Recovering Articulated Non-rigid Shapes, Motions and Kinematic Chains from Video

by
Jingyu Yan

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Marc Pollefeys, Advisor

Carlo Tomasi, Reader

Ming Lin, Reader

Gary Bishop, Committee Member

Leonard McMillan, Committee Member

ABSTRACT

JINGYU YAN: Recovering Articulated Non-rigid Shapes, Motions and Kinematic Chains from Video.
(Under the direction of Marc Pollefeys.)

Recovering articulated shape and motion, especially human body motion, from video is a challenging problem with a wide range of applications in medical study, sport analysis and animation, etc. Previous work on articulated motion recovery generally requires prior knowledge of the kinematic chain and usually does not concern the recovery of the articulated shape. The non-rigidity of some articulated part, e.g. human body motion with non-rigid facial motion, is completely ignored. We propose a factorization-based approach to recover the shape, motion and kinematic chain of an articulated object with non-rigid parts altogether directly from video sequences under a unified framework. The proposed approach is based on our modeling of the articulated non-rigid motion as a set of intersecting motion subspaces. A motion subspace is the linear subspace of the trajectories of an object. It can model a rigid or non-rigid motion. The intersection of two motion subspaces of linked parts models the motion of an articulated joint or axis. Our approach consists of algorithms for motion segmentation, kinematic chain building, and shape recovery. It is robust to outliers and can be automated. We test our approach through synthetic and real experiments and demonstrate how to recover articulated structure with non-rigid parts via a single-view camera without prior knowledge of its kinematic chain.

ACKNOWLEDGMENTS

This work would not have been completed without the following people. Special acknowledgments go to them.

Professor Marc Pollefeys, my advisor, who has always been a great tutor and collaborator. He makes the Ph.D. study and research experience fruitful and enjoyable.

The rest of my Ph.D. committee members: Professor Min Lin, Gary Bishop, Leornard McMillan and Carlo Tomasi, who have been spending time and efforts throughout the whole process toward my completion of this thesis.

The full faculty of the Department of Computer Science, the University of North Carolina at Chapel Hill, especially Professor Frederick P. Brooks Jr., Henry Fuchs, Guido Gerig, Anselmo A. Lastra, Dinesh Manocha, Stephen M. Pizer, Jan F. Prins, Timothy L. Quigg, Jack S. Snoeyink, Stephen F. Weiss.

The full staff of the Department of Computer Science, the University of North Carolina at Chapel Hill, especially Janet Jones, Mellisa Wood and Katrina Coble.

My parents who have been giving me the most generous and continuous support on my study and research.

The support of the NSF ITR grant IIS-0313047 is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Shape and motion recovery is an essential task for computer vision, which reconstructs 3D scene geometry and camera motions from 2D images. One particular interest in this area is how to recover articulated shape and motion from images or video. It has been attracting research interests for decades. It is relevant to human motion, one of the most interesting motions in nature. Articulated motion with non-rigid parts is a good approximation to human motion, e.g. human body motion with non-rigid facial motion. A system that can capture and recover articulated shape and motion has a wide range of applications in medical study, sport analysis and animation, etc.

While most research of articulated objects focuses on estimating and tracking the pose of the articulated object using images or video as input *given* a 3D articulated model, few start the analysis from building its kinematic structure and recover the shape and motion at the same time. In this thesis, we propose a factorization-based approach to analyze and recover the shape, motion and kinematic chain of articulated objects with non-rigid parts, e.g. the human body with non-rigid facial motion, under affine projection from feature trajectories. We model the articulated non-rigid motion using a set of intersecting motion subspaces. A motion subspace is the linear subspace of the trajectories of an object. It models the motion of either a rigid or non-rigid articulated part. The intersection of two connected subspace models an articulated joint or axis. Based on this model, we proposes a general subspace clustering algorithm to segment the motion subspaces. The kinematic chain of the articulated object is built automatically by analyzing the segmented motions. The shape recovery are done via the factorization method for rigid parts and the non-rigid factorization method for non-rigid parts. Our approach consists of algorithms for motion segmentation, kinematic chain building, and shape recovery. Unlike most research on articulated body estimate which assumes a kinematic model as a prior, our approach build the kinematic chain automatically from feature trajectories. We test our approach through synthetic and real experiments. We demonstrate a potential to recover articulated structure with non-rigid parts via a single-view camera without prior knowledge of its kinematic structure.

Before we start describing the approach, we first introduce the background knowledge on which our research is built. These include shape from motion, camera models, epipolar-geometry and the factorization methods. In the end, we will show how our research fit into the full picture by pushing the state-of-the-art of shape and motion recovery of dynamical scenes.

1.1 Shape from Motion

One essential task in computer vision is to acquire the 3D shape of rigid object(s) from 2D images or image sequences. One particular way is that, by taking different shots of the object, we first establish correspondences of the same point between the images. Then via the geometric knowledge we recover the 3D position of these points. This method is what we call *Shape from Motion*.

Finding 2D correspondences can be automated if image sequences from video are used. Because the video frames are continuous in space and time, point features in one frame can be detected and tracked in the subsequent frames. This method is called feature tracking. The 2D correspondence problem is solved by feature tracking.

The second step requires knowledge of geometric relationship between 3D points and their 2D images. The basic idea is that, from every pair of images and the correspondences of the same points, we back-project two rays into 3D space and compute the 3D position of their intersection. This, in turn, requires the knowledge of the relative camera positions and orientations from which both images are taken. Usually the back-projection and the camera motion are variable, which result in more than one solution of 3D positions of the points. This ambiguity can be reduced by using other constraints.

Besides recovering the shape and motion of a rigid object, recovering those of different kinds of scenes has also been being studied and researched. For example, scenes of objects that move independently, non-rigidly or articulately. It may require an analysis step to separate each object by motion segmentation before 3D shape can be recovered for each object. In this thesis, we focus on non-rigid articulated objects and the techniques to recover their shapes and motions from video.

1.2 Camera Models

In this section, we will describe different camera models used in shape and motion recovery problems. The camera model that we adopt affects the algorithm. We are going to introduce two models and explain our reason for using the second one.

The first camera model is one that is used to describe cameras with finite centers; the other, cameras with centers at infinity. Finite cameras are what we use in practice, for example, a pinhole camera. Cameras at infinity are ideal whose centers are supposed to be infinitely far way, so the perspective effect, i.e. the image size of an object depending on its distance to the camera, is diminished. For cameras at infinity, the image size of an object is fixed regardless of the distance between the object and the camera. In other words, in terms of the image, only the relative orientation between the object and the camera matters.

We will describe both camera models mathematically and show their properties.

1.2.1 Finite cameras

A finite camera can be described by the following 3×4 matrix P.

$$\mathbf{P} = \mathbf{KR}[\mathbf{I}|-\mathbf{C}]$$

where $\mathbf{R}$ and $\mathbf{C}$ are the external parameters, which specify the camera orientation and the camera center position in the 3D world coordinate system. $\mathbf{R}$ is a 3×3 rotation matrix. $\mathbf{C}$ is a 3D vector. $\mathbf{K}$ is the calibration matrix of the camera.

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{pmatrix}$$

It represents the internal parameters of the camera. α_x and α_y are the focal lengths in terms of the pixel dimensions in x and y direction; x_0 and y_0 are the 2D coordinates of the principal point on the image plane; s is the skew parameter.

The projection of a 3D point $\mathbf{X}$ in the world onto the image plane by the above camera model is described as follows.

$$\mathbf{x} = \mathbf{PX}$$

$\mathbf{x}$ is a 2D homogeneous coordinates. The inhomogeneous form is like the following.

$$\mathbf{x}_{in} = \begin{pmatrix} \frac{\mathbf{P}^{(1)}\mathbf{X}}{\mathbf{P}^{(3)}\mathbf{X}} \\ \frac{\mathbf{P}^{(2)}\mathbf{X}}{\mathbf{P}^{(3)}\mathbf{X}} \end{pmatrix} = \begin{pmatrix} \frac{(P_{11}P_{12}P_{13})\mathbf{X}_{in}+P_{14}}{(P_{31}P_{32}P_{33})\mathbf{X}_{in}+P_{34}} \\ \frac{(P_{21}P_{22}P_{23})\mathbf{X}_{in}+P_{24}}{(P_{31}P_{32}P_{33})\mathbf{X}_{in}+P_{34}} \end{pmatrix}$$

$\mathbf{P}^{(i)}$ is the ith row of $\mathbf{P}$;$\mathbf{P}_{ij}$ is the element in ith row and jth column of $\mathbf{P}$; $\mathbf{X}_{in}$ is the inhomogeneous coordinate of $\mathbf{X}$.

One important observation of the above equation is that the mapping between the 2D inhomogeneous coordinates $\mathbf{x}_{in}$ and 3D inhomogeneous coordinates $\mathbf{X}_{in}$ is non-linear due to the quotients. On the other hand, the affine camera that we are going to introduce has a mapping that is linear.

1.2.2 Affine cameras

Among cameras at infinity, affine cameras are of particular interest for the simple reason that the mapping between a 3D homogeneous point and its 2D inhomogeneous image position is linear. An affine camera has the third row of its projection matrix as $\mathbf{P}^{(3)} = (0, 0, 0, \alpha)$ $(\alpha \neq 0)$. The 2D projection is described as the following equation.

$$\mathbf{x}_{in} = \begin{pmatrix} \frac{(P_{11}P_{12}P_{13})\mathbf{X}_{in}+P_{14}}{\alpha P_{34}} \\ \frac{(P_{21}P_{22}P_{23})\mathbf{X}_{in}+P_{24}}{\alpha P_{34}} \end{pmatrix} = \frac{1}{\alpha P_{34}} \left(\begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \end{pmatrix} \mathbf{X}_{in} + \begin{pmatrix} P_{14} \\ P_{24} \end{pmatrix} \right)$$

As shown above, the linear mapping relies on the fact that the third row of the projection matrix being $\mathbf{P}^{(3)} = (0, 0, 0, \alpha)$ $(\alpha \neq 0)$. $\mathbf{X}_{in}$ is the inhomogeneous coordinate of the 3D point.

There are a hierarchy of affine cameras in the order of generality. They all have a projection matrix with $\mathbf{P}^{(3)} = (0, 0, 0, \alpha)$ $(\alpha \neq 0)$. Thus the mappings between the 3D and 2D inhomogeneous coordinates are linear.

- Orthographic projection. Suppose in the camera coordinate system, the projection matrix is as follows assuming the projection is along the z-axis.

$$\mathbf{P}_{orth} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  In the orthographic projection, the z coordinate is dropped and the x and y coordinate remain the same. In the world coordinate system, there is an Euclidean transformation between the world and the camera coordinates.

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

A general orthographic projection is the product of these two matrices.

$$\mathbf{P} = \mathbf{P}_{orth}\mathbf{H} = \begin{pmatrix} \mathbf{R}^{(1)} & t_1 \\ \mathbf{R}^{(2)} & t_2 \\ \mathbf{0} & 1 \end{pmatrix}$$

where $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ are the first and second row of the matrix that represents the camera rotation. t_1 and t_2 are the first and second component of the translation vector.

The orthographic projection has 5 degrees of freedom, 3 for the rotation and 2 for the translation.

- Scaled orthographic projection. It is an orthographic projection followed by an isotropic scaling.

$$\mathbf{P} = \begin{pmatrix} k & & \\ & k & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}^{(1)} & t_1 \\ \mathbf{R}^{(2)} & t_2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}^{(1)} & t_1 \\ \mathbf{R}^{(2)} & t_2 \\ \mathbf{0} & 1/k \end{pmatrix}$$

The scaled orthographic projection has 6 degrees of freedom.

- Weak perspective projection. It is an orthographic projection followed by an anisotropic scaling.

$$\mathbf{P} = \begin{pmatrix} \alpha_x & & \\ & \alpha_y & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}^{(1)} & t_1 \\ \mathbf{R}^{(2)} & t_2 \\ \mathbf{0} & 1 \end{pmatrix}$$

The weak perspective projection has 7 degrees of freedom. $P^3 = (0, 0, 0, 1)$. The mapping between 3D and 2D inhomogeneous coordinates is linear.

- Affine projection.

$$\mathbf{P} = \begin{pmatrix} \alpha_x & s & \\ & \alpha_y & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}^{(1)} & t_1 \\ \mathbf{R}^{(2)} & t_2 \\ \mathbf{0} & 1 \end{pmatrix}$$

  This projection has 8 degrees of freedom. $\mathbf{P}^{(3)} = (0, 0, 0, 1)$ so the mapping is linear as well. Notice that it has the same degrees of freedom as the most general form of affine cameras.

$$\mathbf{P}_{aff} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & t_1 \\ m_{21} & m_{22} & m_{23} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  In fact, $\mathbf{P}_{aff}$ is able to be decomposed as the above $\mathbf{P}$. They are equivalent.

Which camera model to use is a practical problem. Generally speaking, the more general the model is, the more accurate it is to approximate the real-world situation. For example, the finite camera is more closely approximate to reality than the affine camera; the scaled orthographic is no less accurate than the orthographic camera. But for algorithmic consideration, a more restricted model is preferred because it is simpler. So it is important to understand the tradeoffs on different situations. For the orthographic or weak perspective camera models, they are fairly accurate if the object's depth dimension is relatively small compared to the camera-object distance. For objects that are close, finite camera models achieve higher accuracy. A rule of thumb is that in a certain situation we choose the simplest model that satisfies the accuracy requirement. Another strategy is to use a simpler model even if it is less accurate but use the result as an initialization for a more general model and refine the initial result.

The camera model that is chosen affects the algorithm. For affine cameras, due to its linear mapping between inhomogeneous coordinates, most algorithms using this model are linear too. For finite cameras, the algorithms are non-linear.

1.3 The Factorization Method

A widely used approach to recover the shape of a scene is called the factorization method. The first one was proposed in (Tomasi and Kanade, 1992). This method assumes that the projection model is orthographic. So in a real case, it is more accurate for scene objects that are relatively far away from the camera compared to their depth changes. Regardless of this limitation, the factorization method has several advantages that make it a practical method. One is that it does not require non-linear computation. The second is that its inputs are from multiple views and processed in a batch fashion. Because of its advantages, lots of works have been done to extend the original factorization method to more camera models such as para-perspective camera models (Poelman and Kanade, 1992) and perspective camera models (Triggs, 1996). It is this method upon which we base our algorithms for articulated non-rigid shape recovery. The details of the factorization method is described in Section 2.1.1.

1.4 Dynamical Scenes and Our focus

Besides recovering the shape of a *static* scene, efforts have been made to extend the factorization method to scenarios of dynamical scenes, e.g. multiple independently moving objects, non-rigid deformable objects, etc. Our interest is to recover articulated non-rigid shapes using the factorization method. Articulated shapes cover a large range of shapes of machines and robots. Articulated shapes with non-rigid parts cover a larger range of shapes of animals, insects, and, most importantly, human beings. We believe that these form an interesting set of general shapes and motions, the solution of which will find its applications in a number of areas and fields.

Due to the high degree of freedom of the articulated shape and its motion, research efforts have been made using strong priors, like the kinematic structure of the object, or using multiple sensor devices like multiple cameras or hybrid cameras with depth sensors. Even so, it is still a hard problem

We attack the problem from a different angle and use a minimum of priors and sensors. We assume no knowledge of the articulated structure of the object and we base our analysis of the shape and motion only on data obtained from a single video camera. These assumptions (or the lack of assumptions, to be more accurately) impose great challenges to the problem. However, an algorithm that can automatically detect the articulated structure and recover the articulated shape from a single camera is much more desirable and more easily adopted in practice. This thesis demonstrates our effort in this direction. Based on our knowledge, it is also the first effort in this direction.

Chapter 2

Previous Work

Our topic involves different areas of research in computer vision. This chapter is to introduce readers to previous work in these areas and outline their relevance to our research as well as our contributions.

2.1 The Factorization Method and Its Extensions

Our approach is factorization-based. It particularly deals with articulated non-rigid shape and motion recovery. This section is to introduce previous work on the factorization method and its extensions to different kinds of shape and motion recovery problems.

One approach for shape and motion recovery that has proven versatile and reliable is the factorization method proposed in (Tomasi and Kanade, 1992). Assuming an affine projection model, they derive from the imaging process a rank constraint of the trajectory matrix whose column vectors are 2D point feature trajectories across frames. Assuming orthographic projection, the trajectory matrix of a rigid object generally has rank-4. Subtracted by the trajectory of the centroid, the trajectory matrix is reduced to rank-3. This rank constraint of the motion subspace allows for noise reduction as well as for factorization of the trajectory matrix into the product of two matrices up to a linear transformation matrix. Enforcing the orthonormal constraints on the first matrix and inversely transforming the second matrix, the scene shape and the camera motion are recovered directly from these two factorized matrices of the trajectory matrix. See Section 2.1.1 for a detailed description of the method.

Later work (Poelman and Kanade, 1992) extends the factorization method to a paraperspective projection model, a first-order approximation of the perspective projection model without losing the linear property of the affine projection model. So the factorization method is applied and the scaling effect of perspective projection is obtained. A paraperspective projection has two steps. First, the world point is projected to a hypothetical plane that is parallel to the image plane. For example, a hypothetical plane is a plane that passes the center of mass of the scene object; the projection direction is along the line of the center of mass and the camera center. Secondly, the hypothetical plane projection is projected toward the camera center like a normal perspective projection. Under the paraperspective projection model, the motion subspace of a rigid scene is also of rank 4 generally. Like in the factorization method, the factorization of the trajectory matrix is determined up to a linear transformation 4 by 4 matrix. Again, a special structure of the first matrix is exploited similarly (this special structure is the major difference between this method and the factorization method). The shape and motion is recovered from the factorization.

The above extension of the factorization method is still dealing with rigid shape and motion recovery. The following literatures extend the method into techniques that deal with different types of dynamic scenes.

(Bregler et al., 2000) extends the factorization method to handle a non-rigid object's shape and motion recovery if the deformation of the object can be approximated by a linear combination of some key shapes. This assumption is fairly general. By factorizing the trajectory matrix into two matrices with a scaled rotation structure in the first matrix via nonlinear iterations, the keys shapes, the weights of the key shapes for each frame and the motion are recovered. Further work (Torresani et al., 2001; Brand, 2001; Xiao et al., 2004; Brand, 2005) investigate different techniques mainly to solve the factorization ambiguity that happens in the non-rigid case due to more than one key shapes (a rigid object is regarded as having only one key shape). See Section 2.1.2 for more information of the factorization approach for non-rigid shape recovery and related work.

(Costeira and Kanade, 1998) shows how to recover via the factorization method the shapes and motions of multiple independently moving objects, for example, vehicles running in a parking lot. It first segments the trajectories into each object. Without knowing the centroid of each object, the trajectory matrix of each object generally has rank-4. The independently moving objects have shape subspaces orthogonal to each other (a formal proof is in (Kanatani, 2002)). In (Costeira and Kanade, 1998), it factorizes the trajectory matrix using SVD and obtain the shape subspace up to a projective transformation. Then it computes the cross-products between the columns of V^T. The cross-products form a so-called *shape interaction matrix.* By sorting this matrix into a block-diagonal form, the segmentation is retrieved. After the segmentation, the factorization method is applied to each object's trajectories. The shape and motion is recovered independently.

It is important to notice that if the objects do not move independently, for example, the linked parts of an articulated object, the property of orthogonality between trajectories of different objects does not exist any more. The segmentation algorithm of (Costeira and Kanade, 1998) is not applicable in articulated shape and motion recovery.

(Han and Kanade, 2000) applies the factorization method to scenes with multiple linearly moving objects (static objects are included in this category because they are regarded as having zero velocity). Aerial video is a major application area for ground objects usually appear as points and their motions appear linear. (Han and Kanade, 2000) points out that the motion subspace of linearly moving objects is at most of rank 6 regardless of how many objects. The trajectory matrix is factorized into two matrices, the first of which has a special structure of its own and the second of which contains the 3D point position and its 3D velocity. By enforcing the structure of the first matrix, the factorization is determined up to a Euclidean transformation. The positions of the points and their velocities is recovered.

2.1.1 The Factorization Method

We describe the factorization method (Tomasi and Kanade, 1992) for rigid motion recovery in this section.

Suppose the motion and shape are general and the projection model is affine. Stack the 2D trajectories of feature points tracked over f frames as follows.

$$W = \begin{pmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ v_{1,1} & v_{1,2} & \dots & v_{2,n} \\ \dots & \dots & \dots & \dots \\ u_{f,1} & u_{f,2} & \dots & u_{f,n} \\ v_{f,1} & v_{f,2} & \dots & v_{f,n} \end{pmatrix} \tag{2.1}$$

$(u_{i,j}, v_{i,j})$ are the image coordinates of point j in frame i.

The trajectory matrix W has rank 4 due to general shape and motion. We can factorize W into M and S using SVD up to rank-4.

$$W = U\Sigma V^T = (U\Sigma^{1/2})(\Sigma^{-1/2}V^T) = MS$$

This factorization is not unique because a pair of M and S can be transformed into another pair of M' and S' related by an invertible 4 by 4 matrix Q'.

$$W = MS = (MQ')(Q'^{-1}S) = M'S'$$

The objective is to find an appropriate Q' such that the motion and shape is recovered from M' and S'. Let $Q' = [Q'_R|q'_t]$, where Q'_R are the first 3 columns, q'_t are the last column. In total, Q' has 16 unknowns.

$$\begin{pmatrix} q_1 & q_2 & q_3 & q_4 \\ q_5 & q_6 & q_7 & q_8 \\ q_9 & q_{10} & q_{11} & q_{12} \\ q_{13} & q_{14} & q_{15} & q_{16} \end{pmatrix}$$

The first 3 columns of two rows of M' corresponding to the same frame are two orthogonal unit vectors. They provide 3 linear constraints on Q'_R for each frame.

$$m_i Q'_R Q'^T_R m_i^T = 1 \qquad (2.2)$$

$$m_j Q'_R Q'^T_R m_j^T = 1 \qquad (2.3)$$

$$m_i Q'_R Q'^T_R m_j^T = 0 \qquad (2.4)$$

where m_i and m_j are two rows of M corresponding to the same frame. Without loss of generality, we fix the first two rows of M' as $(1, 0, 0)$ and $(0, 1, 0)$. So we have the following.

$$m_1 Q'_R = (1, 0, 0) \qquad (2.5)$$

$$m_2 Q'_R = (0, 1, 0) \qquad (2.6)$$

where m_1 and m_2 are the first and second row of M.

Given all the constraints, we have an overconstrained linear system for computing the symmetric matrix $Q'_R Q'^T_R$. Then we use SVD to factorize the symmetric matrix and get Q'_R.

To compute q'_t, we will need the translation of the object in each frame. Assuming the centroid of the object as the world coordinate, we have the following.

$$Mq'_t = \frac{1}{n}\sum_i w_i$$

where w_i is the ith column of W. q' is computed from these overconstrained linear system using the least square technique.

2.1.2 The Non-Rigid Shape and Motion Recovery via the Factorization Method

Suppose the motion and shape are general and the projection model is affine. Stack the 2D trajectories of a non-rigid object as in Equation 2.1.

The shape of a non-rigid object at a time is approximated by a linear combination of a number of, e.g. K, key shapes (Bregler et al., 2000; Brand, 2001). For the fth frame, the projected image coordinates of these features are m^f.

$$m^f_{2\times P} = [R^f_{2\times 3}|T^f_{2\times 1}] \begin{pmatrix} \sum_{i=1}^K c^f_i S^i_{3\times P} \\ \mathbf{1}_{1\times P} \end{pmatrix} \tag{2.7}$$

Putting all frames together, the trajectory matrix is written as follows, which has a linear subspace of no more than $3K+1$ dimensions.

$$W = \begin{pmatrix} c^1_1 R^1_{2\times 3}|...|c^1_K R^1_{2\times 3}|T^1_{2\times 1} \\ ... \\ c^F_1 R^F_{2\times 3}|...|c^F_K R^F_{2\times 3}|T^F_{2\times 1} \end{pmatrix} \begin{pmatrix} S^1_{3\times P} \\ ... \\ S^K_{3\times P} \\ \mathbf{1}_{1\times P} \end{pmatrix} \tag{2.8}$$

where c^i_j $(1 \leq i \leq F, 1 \leq j \leq K)$ are the linear coefficients of the key shapes.

Similarly, a factorization of W into two matrices enforcing the rank $3K+1$ is determined up to a linear transformation $(3K+1)\times(3K+1)$ matrix. By enforcing the orthonormal constraints on the first matrix, we expect to determine the factorization. The shape bases, the weights and the camera motion are recovered. This is basically what (Bregler et al., 2000) proposes.

There are several variants (Torresani et al., 2001; Brand, 2001; Xiao et al., 2004; Brand, 2005) of recovering non-rigid motion using the factorization method. One improves upon another. The major idea is the same: factorize W into M and S; find a Q' such that $M' = MQ'$ and $S' = Q'^{-1}S$ that satisfy the orthonormal constraints for the trajectory matrix or other constraints for the shape matrix.

We will not describe the detail of each approach. We will point out the improvements that each approach makes. Readers are referred to the literature for implementation details. The approach proposed in (Bregler et al., 2000) enforces the orthonormal constraints on the trajectory matrix in a similar way of the rigid case except for a higher dimension of $3K$ by $3K$ ($K > 1$) for Q'. (Torresani et al., 2001) proposes a nonlinear iteration technique to alternatively estimate the rotation, the weights and the shape bases given initial estimations of them. The initial estimation of camera rotations are retrieved by factorizing the trajectory matrix as in (Tomasi and Kanade, 1992) treating the non-rigid shape as a rigid object assuming the non-rigid shape has a major rigid component. From the rotation for each frame, they estimate the weights and the shape bases as well. With these initial values, it fixes the weights and rotations and re-estimates the shape bases; fixes the rotations and the shape bases, re-estimates the weights; fixes the weights and shape bases, re-estimate the rotations. This process repeats until it converges. (Brand, 2001) points out that the orthonormal constraints are not sufficient for determining the factorization and proposes an additional constraint of minimizing the deformation part of the shape matrix via an optimization method. (Xiao et al., 2004) rigorously proves that the orthogonal constraint is not sufficient whatsoever and proposes to specify key frames to fix the shape bases so the factorization is determined up to a global 3D rotation. (Brand, 2005) proposes an optimization approach to globally derives the motion, shape bases and weights so that it is not necessary to choose key frames as proposed in (Xiao et al., 2004).

2.2 Motion Segmentation

Motion segmentation analyzes a scene with multiple objects and segment the motions of different objects. It allows for easier analysis of the shape and motion of an individual object once the segmentation is done. In our approach, motion segmentation is an important step. This section will introduce previous work in this field and our contribution.

Under the introduction of the factorization method, motion segmentation of multiple rigid objects can leverage linear dependence among the trajectories of the same object and linear independence between independently moving objects. The segmentation becomes a subspace clustering problem. This new insight spans a lot of new literature exploring different techniques for motion subspace clustering.

To be more concrete, let W be the trajectory matrix of n tracked feature points of m independently moving objects. Suppose these tracked points are grouped by the objects that they belong to. The shape matrices of the objects are S_1, ..., S_m. Their motion matrices and translation vectors are R_1,...,R_m and T_1,...,T_m respectively. W is written out as follows.

$$W = (R_1|T_1|R_2|T_2|...|R_m|T_m) \begin{pmatrix} S_1 & & \\ \mathbf{1} & & \\ & ... & \\ & & S_m \\ & & \mathbf{1} \end{pmatrix} \tag{2.9}$$

W is at most of rank $4 \times m$ because the trajectories of each rigid object has at most rank-4 motion subspace. Note that $(R_i|T_i)$ corresponds to the relative motion between the object and the camera. If the subspace of $(R_i|T_i)$ is independent of that of $(R_j|T_j)$, we say that the object i and j are moving independently. We have defined independent motions in such a way that their motion subspaces are guaranteed to be independent. As seen in Equation 2.9, the independence between any $(R_i|T_i)$ and $(R_j|T_j)$ guarantees that the trajectories of the object i and the object j belong to independent motion subspaces.

(Boult and Brown, 1991) uses a singular value decomposition to decompose the trajectory matrix W, with unknown permutation of trajectories of independent objects, into L,Σ and R up to the detected rank of W, $\Re(W)$. The columns of R retain the linear dependence of the trajectories but have a smaller dimension. R is used to generate potential segmentations, e.g. to generate one segmentation, choose 4 trajectories and group all trajectories that are within their subspace. For each potential segmentation, the trajectories are removed from W yielding W_i. If $\Re(W) = \Sigma_i \Re(W_i)$, the segmentation is good at this level; otherwise, regenerate potential segmentations. Repeat this generate-and-check process until $\Re(W_i) \leq 4$. It may be necessary to go back a higher level if W_i can not be segmented further while $\Re(W_i) > 4$. This segmentation method is iterative and potentially expensive.

(Gear, 1994) uses the reduced row echelon form of the trajectory matrix W for direct segmentation of the trajectories of independent objects. The columns of W are the trajectories and those belonging to the same motion lie in a subspace of rank 4, independent of the others. Using Gauss-Jordan elimination to the rows of W, we get the reduced row echelon form of W. The Gauss-Jordan elimination to the rows does not change which subspace a column belongs to. From the reduced row echelon form of a matrix, we can identify the independent subspaces of a matrix and the columns of the matrix that belong to the same subspace. The following criterion is used to group the trajectories: if two columns of the echelon form have non-zero entries on the same row, the corresponding trajectories belong to the same object. This method does not require iterations and provide a direct way for segmentation. For example, the following is the reduced row echelon form of a matrix W without noise of two independent motions. The columns have been grouped such that column 1 to 7 are of the same motion and the rest of the columns, of the other motion.

$$\begin{pmatrix}
0.2 & -0.3 & 0 & 1.0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0.9 & 0 & 0 & 1.0 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.3 & -0.2 & 1.0 & 0 & 0 & -0.4 \\
0.5 & 0.5 & 0 & 0 & 1.0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0.5 & -0.1 & 0 & 0 & 0 & 0.3 \\
1.7 & 0.9 & 0 & 0 & 0 & 1.0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.5 & 0 & 1.0 & 0 & 0.5 \\
-0.8 & -0.2 & 1.0 & 0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

(Costeira and Kanade, 1998) proposes an influential approach for independent bodies motion segmentation. Given the trajectory matrix W of multiple independently moving objects, it first decompose W into U, Σ and V^T using the singular value decomposition up to the rank of W, $\Re(W)$. Then with the ith row of V representing the ith trajectory, a shape interaction matrix Q is formed whose (i, j) and (j, i) entries are equal to the inner product of row i and j.

$$Q = VV^T$$

If the ith and jth trajectory are of independent motion subspaces, the entry in the shape interaction matrix is zero because their row vectors in V are independent as well. A permutation of the rows of V corresponds to a permutation of the rows and columns of the shape interaction matrix. The rest of the algorithm is to permute the rows of V such that the shape interaction matrix has a block diagonal form. When the row vectors of the same subspace are grouped together, a block of entries of the shape interaction matrix are non-zeros and the rest entries on the same rows are zeros.

The relationship between (Costeira and Kanade, 1998) and other powerful segmentation techniques based on affinity matrix using eigenvectors is revealed in (Weiss, 1999). An affinity matrix describes the distances between every pair of data. The distance of data i and j is the value of the (i, j) and (j, i) entry of the affinity matrix. The data dimension and the distance function are problem specific. The segmentation is trying to segment the data into two groups so that the distances between data of the same group normalized by the distances between all data are minimized. (Weiss, 1999) describes 4 segmentation algorithms using eigenvectors. For the first 3 algorithms, the difference consists of which eigenvector(s) are used for segmentation. The first one of (Perona and Freeman, 1998) uses the first largest eigenvector of the affinity matrix for thresholding to segment the data; the second one of (Shi and Malik, 2000) uses the second generalized eigenvector for thresholding; the third one of (Scott and Longuet-Higgins, 1990) uses all k eigenvectors. While thresholding is not possible for k eigenvectors, it thresholds the inner product between the every normalized row vector of the matrix whose k columns are the eigenvectors. The fourth algorithm described in (Weiss, 1999) is the motion segmentation of (Costeira and Kanade, 1998). Except for the normalization step, (Costeira and Kanade, 1998) is identical to (Scott and Longuet-Higgins, 1990). (Weiss, 1999) provides a more solid theoretical foundation for the multibody motion segmentation of (Costeira and Kanade, 1998).

Due to the large body of works of factorization-based motion segmentation, we need to draw the distinction between our motion segmentation work and the previous work. Most of the previous work assume independence between motion subspaces, (Boult and Brown, 1991; Gear, 1994; Costeira and Kanade, 1998) and its refinements, (Kanatani, 2002; Ichimura, 1999) etc. Our goal is to segment articulated motions which are a mixture of independent and dependent motions. Without the assumption of the independence between motion subspaces, these approaches may not work. Another challenge in dealing with a mixture of motions is to do it in a unified way.

(Zelnik-Manor and Irani, 2003) addresses the dependency problem between motion subspaces and deals with it using a method related to (Costeira and Kanade, 1998) to derive an affinity matrix, followed by the technique of (Kanatani, 2002) to separate the dependent motions. In their case, the angle between every pair of vectors, expressed by a dot product, are used as the affinity measure. However, unlike the independent motion subspace cases, angles, or any other distance measure between the data, do not reflect the true geometric constraints, the subspace constraints, that we use to segment the data. Instead, our method uses the distance between two locally estimated subspaces of each data, expressed by subspace principal angles, as the affinity measure. This new affinity measure reflects the true nature of the constraint for segmentation and leads to more accurate results, which is confirmed by our experiments.

Vidal et al.(Vidal and Hartley, 2004; Vidal et al., 2003; Vidal et al., 2004) propose an algebraic framework called GPCA that can deal with dependent and independent subspaces with unknown dimensionality uniformly. It models a subspace as a set of linear polynomials and a mixture of n subspaces as a set of polynomials of degree n. Given enough sample points in the mixture of subspaces, the coefficients of these high degree polynomials are estimated. By differentiating at each data point, the normals of each data are estimated. Then it also uses standard methods like principal angles and spectral clustering to segment the data. However, because GPCA first brings the problem to a high degree nonlinear space and then solves it linearly, the number of sample points required by GPCA to estimate the polynomial coefficients becomes its Achilles' heel, which grows exponentially with the number of subspaces and the dimensions ($\bigcirc((d+1)^n)$, d is the dimension of the largest underlying subspace and n is the number of subspaces). In practice, the number of trajectories can hardly satisfy GPCA's requirement for it to handle more than 3 subspaces. And for non-rigid motion subspaces whose dimensions are more than 4, the situation gets even worse. Our method requires $\bigcirc(d \times n)$ trajectories which makes it practical to handle not only multiple motions but also non-rigid motions that have a higher dimension.

Though our approach is motivated and derived independently, in a different context (Ferrari-Trecate et al., 2003) uses similar techniques, such as local sampling and clustering, to identify discrete-time hybrid systems in piecewise affine form.

2.3 Articulated Structure from Motion

Researchers have been working on algorithms to compute articulated shape and motion; some of them use factorization-based methods as well. The originality of our approach lies in: 1. our perspective of viewing articulated motion with rigid or non-rigid parts uniformly as a set of intersecting motion subspaces; 2. the recovery approach, whose simplicity benefits from this perspective and which only requires basic operations such as subspace intersection, clustering and segmentation; 3. automatical kinematic chain building.

We summarize others' works in the following and make comparison with ours. In

(Zhou and Huang, 2003) each part of an articulated object is treated as independently moving objects using the same approach of (Costeira and Kanade, 1998) at the initial stage and then apply translation constraints on top of the initial result. The translation constraint fixes two end points of the articulated parts on a link. During the process, the root object needs to be recovered first and the rest are recovered hierarchically. An important problem is that articulated shapes do not satisfy the orthogonality assumption of (Costeira and Kanade, 1998). In our approach, the articulated parts are treated equally as different intersecting subspaces and no heuristics are needed. The translation constraint is intrinsically expressed in the intersection of motion subspaces and does not need to be imposed explicitly.

The approach proposed in (Sinclair et al.,1997) first recovers the projective structure including the rotation of the camera before it uses a minimization approach to recover an axis. The joint case is not addressed. Our method directly achieves the motion subspace of an axis or joint from the trajectories without recovering any shape.

There are also a large group of articulated pose estimation method from an image or image sequence, (Taylor, 2000; Bregler and Malik, 1998; Hogg, 1983; Sminchisescu and Triggs, 2001), to name a few. Notice that they require a kinematic model as a prior. Our approach does not require such a prior model as the kinematic structure is inferred from the trajectory data, thus it can potentially be used to provide prior models for these tracking and pose estimation approaches.

For example, (Taylor, 2000) proposes an algorithm to recover the pose of an articulated model, e.g. a person, from a single snapshot. The assumptions include the correspondence between the image points and the joints of the 3D kinematic model, the relative lengths of the segments of the model and the projection being able to be modeled as a scaled orthographic projection. It does not need calibrated images as inputs. The scaled orthographic projection of a 3D point is described by the following equation.

$$\begin{pmatrix} u \\ v \end{pmatrix} = s \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

The only variable for the reconstruction is the scale s. This provides a simple parameter for pose estimation compared to other common models using joint angles.

(Bregler and Malik, 1998) introduces a mathematical technique of the product of exponential maps and twist motions, the former of which models the joint angles of an articulated model and the latter of which models the camera motions. This new technique provides a linear method for frame-by-frame tracking of an articulated model from image sequences when an image motion estimation based on intensity is integrated with the technique. The initialization of the tracking includes specifying the articulated model and providing the correspondence between the image points and the articulated joints.

(Sminchisescu and Triggs, 2001) proposes a hybrid search algorithm that aims to attack the high degree of freedom in the parameter space for 3D body tracking.

Furthermore, compared to previous works which assume that the parts of an articulated motion are rigid (Tresadern and Reid, 2005; Zhou and Huang, 2003), our approach can recover articulated motion with rigid or non-rigid parts and treat them in a unified framework.

To summarize, our contributions extend the state-of-the-art in articulated shape and motion recovery. Our approach is based on the factorization method. It models the non-rigid articulated shape and motion using a combination of rigid or non-rigid motion subspaces. It proposes algorithms to segment motion subspaces that are not totally independent, to detect articulated links and establish the kinematic chain automatically and to recover the whole articulated motion and shape in the end.

The following chapters are organized as followed: Chapter 3, detailed discussion of our modeling of articulated motions using subspaces and its extension to non-rigid articulated parts; Chapter 4 and Chapter 5, our algorithms for motion segmentation; Chapter 6, our algorithm for automatic kinematic chain building from feature trajectories; Chapter 7, a summary of our approach, conclusions and future work.

Chapter 3

Modeling Articulated Motion Using Subspaces and Its Extension to Non-Rigid Parts

Before we introduce our algorithms for articulated shape and motion recovery, we shall spend this chapter discussing the mathematical model that we use to represent articulated motions.

We are going to start discussing the motion of a rigid object by investigating the subspace of its trajectories, followed by a discussion on the motions of independently moving objects. Then we will focus on the motion of an articulated object and extend the discussion to articulated objects with non-rigid parts.

The conclusions of this chapter include:

- Articulated motion with non-rigid parts is modeled as a set of intersecting subspaces
- The intersection between two motion subspaces implies a link between two parts and the type of the link, either a joint or an axis, can be detected by the dimension, either 1 or 2, of the intersection .
- The intersection is the motion subspace of the link.

3.1 The Motion Subspace of A Rigid Object

The coordinate of a point of the rigid object in the object's coordinate system is $(a_\alpha, b_\alpha, c_\alpha)$ (Fig. 3.1). Viewed by a camera at the κ frame, its coordinate in the camera's

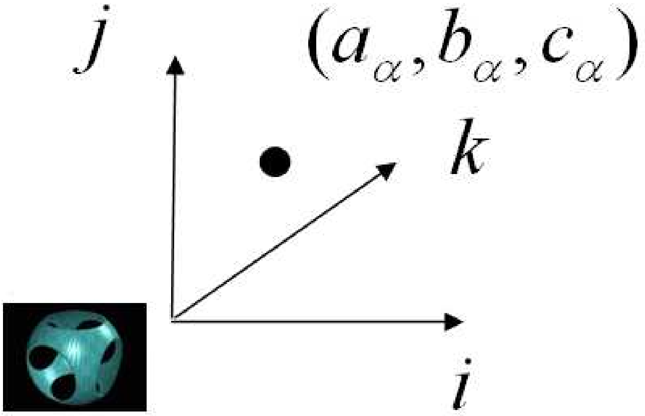


Figure 3.1: A point in the object's coordinate system

coordinate system becomes

$$\begin{pmatrix} x_{\alpha\kappa} \\ y_{\alpha\kappa} \\ z_{\alpha\kappa} \end{pmatrix} = a_\alpha \overrightarrow{i}_\kappa + b_\alpha \overrightarrow{j}_\kappa + c_\alpha \overrightarrow{k}_\kappa + \overrightarrow{t}_\kappa$$

where $\overrightarrow{t}_\kappa$ is the coordinate of the object's origin in the camera's coordinate system; $\overrightarrow{i}_\kappa$, $\overrightarrow{j}_\kappa$ and $\overrightarrow{k}_\kappa$ represent the object's i, j and k axis (Fig. 3.2).

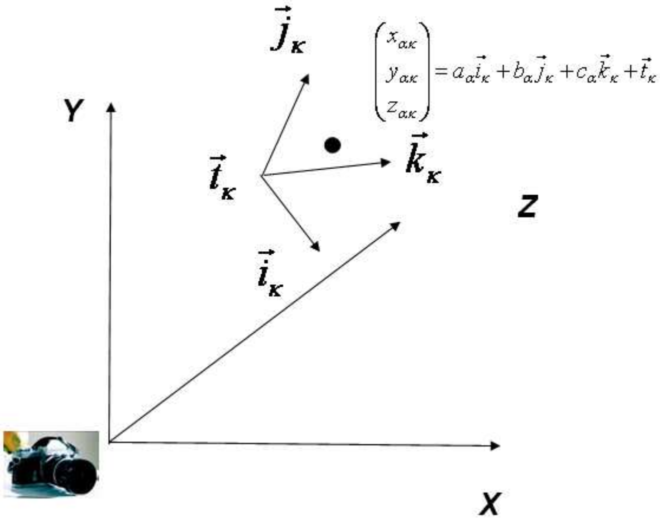

$\vec{j}_\kappa$
$\begin{pmatrix} x_{\alpha\kappa} \\ y_{\alpha\kappa} \\ z_{\alpha\kappa} \end{pmatrix} = a_\alpha \vec{i}_\kappa + b_\alpha \vec{j}_\kappa + c_\alpha \vec{k}_\kappa + \vec{t}_\kappa$
Y
$\vec{t}_\kappa$
$\vec{k}_\kappa$
Z
$\vec{i}_\kappa$
X

Figure 3.2: A point and the object coordinate system in the camera's coordinate system

By an orthographic projection, the image coordinate of the point is

$$\begin{pmatrix} x_{\alpha\kappa} \\ y_{\alpha\kappa} \end{pmatrix} = a_\alpha \widetilde{\overrightarrow{i}}_\kappa + b_\alpha \widetilde{\overrightarrow{j}}_\kappa + c_\alpha \widetilde{\overrightarrow{k}}_\kappa + \widetilde{\overrightarrow{t}}_\kappa$$

where $\widetilde{\overrightarrow{i}}_\kappa$, $\widetilde{\overrightarrow{j}}_\kappa$, $\widetilde{\overrightarrow{k}}_\kappa$ and $\widetilde{\overrightarrow{t}}_\kappa$ are $\overrightarrow{i}_\kappa$, $\overrightarrow{j}_\kappa$, $\overrightarrow{k}_\kappa$ and $\overrightarrow{t}_\kappa$ with the z coordinate chopped off.

Stack the image coordinates over F frames, the trajectory of the point is

$$\begin{pmatrix} x_{\alpha 1} \\ y_{\alpha 1} \\ \dots \\ x_{\alpha F} \\ y_{\alpha F} \end{pmatrix} = a_\alpha \overrightarrow{m}_i + b_\alpha \overrightarrow{m}_j + c_\alpha \overrightarrow{m}_k + \overrightarrow{m}_t$$

It is a linear combination of four vectors, $\overrightarrow{m}_i$, $\overrightarrow{m}_j$, $\overrightarrow{m}_k$ and $\overrightarrow{m}_t$. The trajectory of a different point is another linear combination of these four vectors. The trajectories of a rigid object is in a rank-4 linear subspace. Furthermore, the trajectory of the object origin is $\overrightarrow{m}_t$.

The trajectory matrix is written as:

$$W = \begin{pmatrix} x_{11} & x_{21} & \dots & x_{p1} \\ x_{12} & x_{22} & \dots & x_{p2} \\ \dots & \dots & \dots & \dots \\ x_{1F} & x_{2F} & \dots & x_{pf} \end{pmatrix} = (\overrightarrow{m}_i \, \overrightarrow{m}_j \, \overrightarrow{m}_k \, \overrightarrow{m}_t) \begin{pmatrix} a_1 & a_2 & \dots & a_{p1} \\ b_1 & b_2 & \dots & b_p \\ c_1 & c_2 & \dots & c_p \\ 1 & 1 & 1 & 1 \end{pmatrix} = (R|T) \begin{pmatrix} S \\ 1 \end{pmatrix}$$

The trajectory matrix is of rank-4. When the shape of the object is linear or planar, the rank is 2 and 3 respectively.

3.2 The Motion Subspace of Multiple Independently Moving Objects

Now we will discuss the trajectories of multiple independently moving objects and the motion subspace of them. "Independently moving" is defined as the motion subspaces of each object's trajectories are linearly independent from each other. Loosely speaking, if the objects are rotating and moving on its own in front of the camera, they are moving

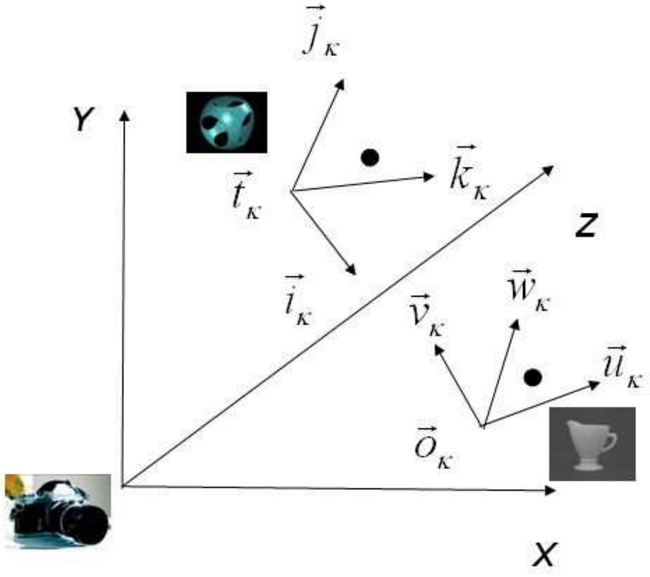


Figure 3.3: Independently moving objects in the camera's coordinate system

independently.

We start with two independently moving objects. Stack the image coordinates over F frames, the trajectories of the two points of the different objects (Fig. 3.3) are

$$\begin{pmatrix} x_{\alpha 1} \\ y_{\alpha 1} \\ \ldots \\ x_{\alpha F} \\ y_{\alpha F} \end{pmatrix} = a_\alpha \overrightarrow{m}_i + b_\alpha \overrightarrow{m}_j + c_\alpha \overrightarrow{m}_k + \overrightarrow{m}_t$$

and

$$\begin{pmatrix} x_{\beta 1} \\ y_{\beta 1} \\ \ldots \\ x_{\beta F} \\ y_{\beta F} \end{pmatrix} = a_\beta \overrightarrow{n}_u + b_\beta \overrightarrow{n}_v + c_\beta \overrightarrow{n}_w + \overrightarrow{n}_o$$

Each of them is a linear combination of four vectors while the first four are independent from the later four.

The trajectory matrix of each object is written separately as:

$$W_1 = (R_1|T_1)\begin{pmatrix} S_1 \\ 1 \end{pmatrix}$$

and

$$W_2 = (R_2|T_2)\begin{pmatrix} S_2 \\ 1 \end{pmatrix}$$

Or putting them together into:

$$W = (W_1|W_2) = (R_1|T_1|R_2|T_2)\begin{pmatrix} S_1 & \\ 1 & \\ & S_2 \\ & 1 \end{pmatrix} \tag{3.1}$$

The rank of the above matrix is 8. The trajectory matrix of the independently moving object has a rank of 4 × N (N is the number of objects).

3.3 The Motion Subspace of an Articulated Object

After the introduction of the motion subspace of a rigid object and multiple independently moving objects, we are going to discuss the motion subspace of an articulated object in this section. An articulated object is an object consisting of multiple parts that are linked by a joint or an axis. An articulated object can move as a whole and its part can move separately under the constraint of its link(s) to other part(s).

3.3.1 Rank constraint of two linked articulated parts

For simplicity, we will discuss an articulated object with two linked parts. The conclusions apply to an articulated object with multiple linked parts.

We will discuss the case of a joint link first. Without loss of generality, we specify that the joint coincides with the origins of both object coordinate systems. Like a single rigid object, a point on the first or second part has a trajectory that is a linear

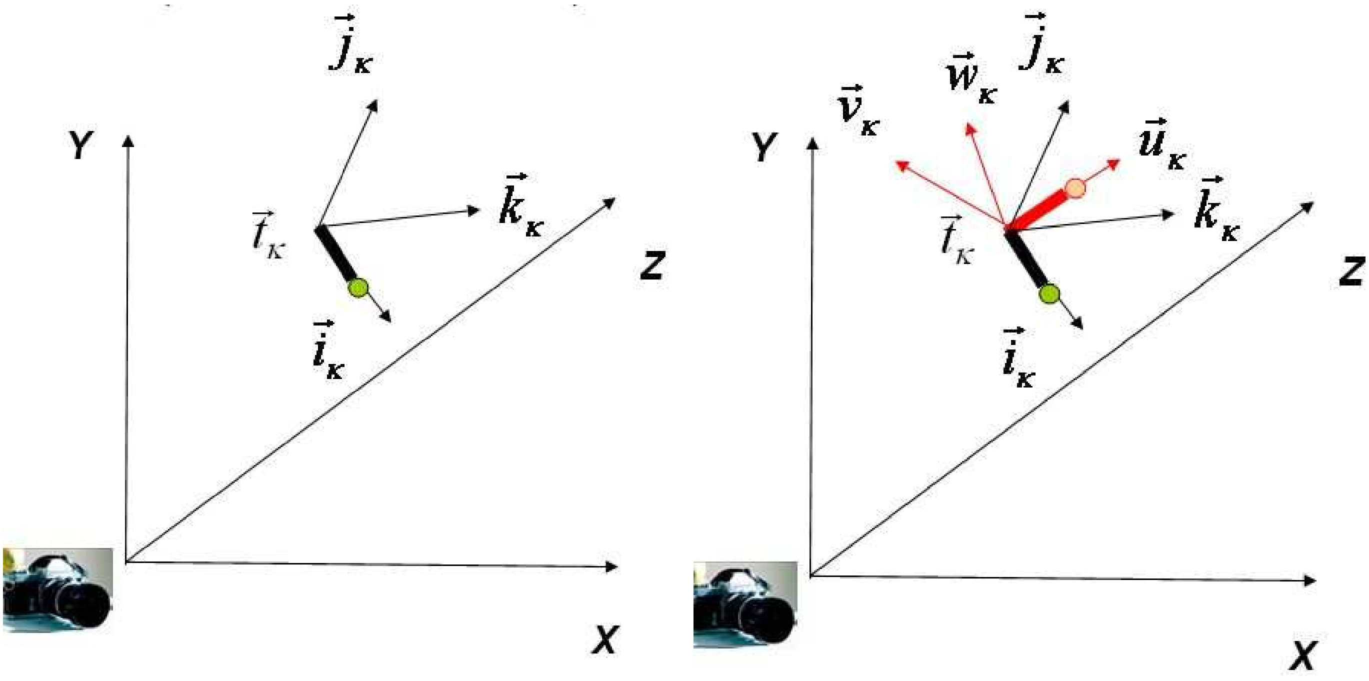


Figure 3.4: *left* An articulated part in the camera's coordinate system. *right* Two articulated parts in the camera's coordinate system.

combination of four vectors (Fig. 3.4).

$$\begin{pmatrix} x_{\alpha 1} \\ y_{\alpha 1} \\ \dots \\ x_{\alpha F} \\ y_{\alpha F} \end{pmatrix} = a_\alpha \overrightarrow{m}_i + b_\alpha \overrightarrow{m}_j + c_\alpha \overrightarrow{m}_k + \overrightarrow{m}_t$$

and

$$\begin{pmatrix} x_{\beta 1} \\ y_{\beta 1} \\ \dots \\ x_{\beta F} \\ y_{\beta F} \end{pmatrix} = a_\beta \overrightarrow{n}_u + b_\beta \overrightarrow{n}_v + c_\beta \overrightarrow{n}_w + \overrightarrow{m}_t$$

Notice that $\overrightarrow{m}_t$ appears in both equations. It implies that there is a 1-D intersection between both linear subspaces and the intersection is the motion subspace of the joint, i.e. the trajectory of the joint is in this subspace.

Another way to write out the trajectory matrix of the articulated object is:

$$W = (W_1|W_2) = (R_1|R_2|T) \begin{pmatrix} S_1 & \\ & S_2 \\ 1 & 1 \end{pmatrix} \tag{3.2}$$

In general, the trajectory matrix of an articulated object with two parts linked by a joint is of rank-7. We assume that the shape and motion is general without degeneracy. Degenerate shapes are shapes that are linear or planar instead of 3 dimensional. Degenerate motion can be defined as insufficient rotation or translation of the object. Either degenerate shapes or motions result in a motion subspace of the trajectories with less ranks. Degenerate shapes will be discussed in Section 8.2. Degenerate motion basically makes the condition insufficient to recover the shape and motion from feature trajectories. We will mainly focus our discussion on general shapes and motions.

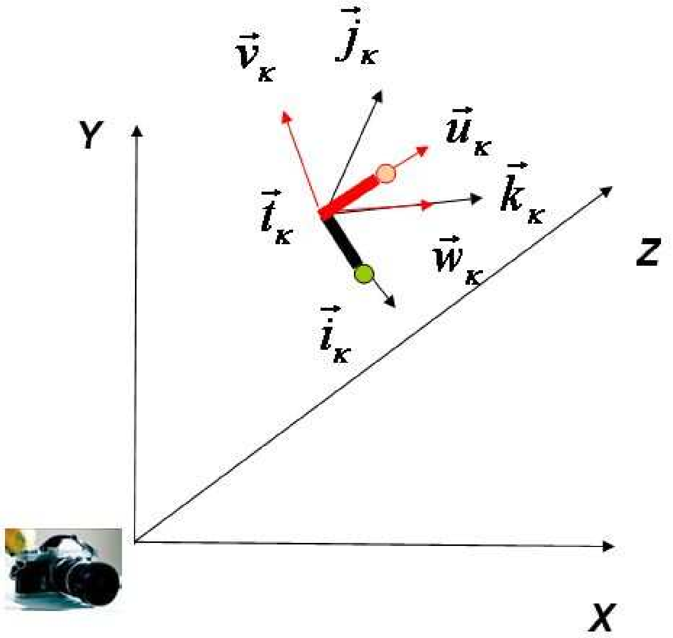


Figure 3.5: *left* An articulated part in the camera's coordinate system. *right* Two articulated parts in the camera's coordinate system.

For the axis link case, we can specify that the rotation axis coincides with a coordinate axis in each part's coordinate system (Fig. 3.5).

$$\begin{pmatrix} x_{\alpha 1} \\ y_{\alpha 1} \\ ... \\ x_{\alpha F} \\ y_{\alpha F} \end{pmatrix} = a_\alpha \overrightarrow{m}_i + b_\alpha \overrightarrow{m}_j + c_\alpha \overrightarrow{m}_k + \overrightarrow{m}_t$$

and

$$\begin{pmatrix} x_{\beta 1} \\ y_{\beta 1} \\ ... \\ x_{\beta F} \\ y_{\beta F} \end{pmatrix} = a_\beta \overrightarrow{n}_u + b_\beta \overrightarrow{n}_v + c_\beta \overrightarrow{m}_k + \overrightarrow{m}_t$$

In this case, $\overrightarrow{m}_k$ and $\overrightarrow{m}_t$ appear in both equations. There is a 2-D intersection between both linear subspaces and the intersection is the motion subspace of the axis, i.e. the trajectories of the points on the axis is in this subspace.

The trajectory matrix of the articulated object is:

$$W = (W_1|W_2) = (\overrightarrow{m}_i \, \overrightarrow{m}_j \, \overrightarrow{n}_u \, \overrightarrow{n}_v \, \overrightarrow{m}_k \, \overrightarrow{m}_t) \begin{pmatrix} a_1 & ... & a_q & & & \\ b_1 & ... & b_q & & & \\ & & & a_{q+1} & ... & a_p \\ & & & b_{q+1} & ... & b_p \\ c_1 & ... & c_q & c_{q+1} & ... & c_p \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{3.3}$$

The trajectory matrix of an articulated object with two parts linked by an axis is of rank-6.

3.4 Extension to non-rigid parts

In this section, we extend our discussion to articulated objects with non-rigid parts. An example is the human motion whose facial motion is non-rigid and whose head and body motions combined can be considered as articulated. We will discuss non-rigid motion subspace first; then we will focus on a typical non-rigid case and prove some results; lastly, we discuss how this typical case can fit into the articulated motion subspace discussed above.

The trajectories of a non-rigid object are approximated by different linear combinations of a number of key shapes(Bregler et al., 2000; Brand, 2001), e.g. K. For the fth frame, the image coordinates of the features are m^f.

$$m^f_{2\times P} = [R^f_{2\times 3}|T^f_{2\times 1}] \begin{pmatrix} \sum_{i=1}^K c^f_i S^i_{3\times P} \\ \mathbf{1}_{1\times P} \end{pmatrix} \tag{3.4}$$

Putting all frames together, the trajectory matrix is written as follows, which forms a linear subspace with a dimension no more than $3K+1$ because each of the shape basis S^i is of rank 3.

$$W = \begin{pmatrix} c^1_1 R^1_{2\times 3}|...|c^1_K R^1_{2\times 3}|T^1_{2\times 1} \\ ... \\ c^F_1 R^F_{2\times 3}|...|c^F_K R^F_{2\times 3}|T^F_{2\times 1} \end{pmatrix} \begin{pmatrix} S^1_{3\times P} \\ ... \\ S^K_{3\times P} \\ \mathbf{1}_{1\times P} \end{pmatrix} \tag{3.5}$$

where c^i_j $(1 \le i \le F, 1 \le j \le K)$ are the linear coefficients of the key shapes.

Let us consider a typical case: the non-rigid shape has rigid points. This includes human facial motion which deforms on top of rigid head motion. More formally, we are considering such a case that a non-rigid shape is represented by linear combinations of a number of key shapes S^1,...,S^K where $S^1_r = ... = S^K_r$ if its rth point is rigid.

We prove the following.

Proposition 3.4.1: If a non-rigid shape is represented by linear combinations of S^1,...,S^K that satisfy $S^1_r = ... = S^K_r$ for any rigid point r, the sum of the linear coefficients for any frame f is 1, i.e. $\sum_{i=1}^K c^f_i = 1$.

Proof: Let S_r be a rigid point of the non-rigid shape. For any frame f, its 2D coordinates are:

$$W^f_r = [c^f_1 R^f|...|c^f_K R^f|T^f] \begin{pmatrix} S_r \\ ... \\ S_r \\ 1 \end{pmatrix} \tag{3.6}$$

Because S_r is rigid, W^f_r can also be written as follows.

$$W^f_r = [R^f|T^f] \begin{pmatrix} S_r \\ 1 \end{pmatrix} \tag{3.7}$$

By comparing Equation 3.6 and 3.7, we have $\sum_{i=1}^K c^f_i = 1$. ■

Corollary 3.4.2: If a non-rigid shape is represented by linear combinations of S^1,...,S^K that satisfy $S_r^1 = ... = S_r^K$ for any rigid point r, the rigid motion subspace formed by the rigid points is embedded in the non-rigid motion subspace.

Proof: From Proposition 3.4.1, we know $\sum_{i=1}^{K} c_i^f = 1$ for any frame f. Let S_H be the set of all rigid points. We have the following.

$$\begin{pmatrix} c_1^1 R^1|...|c_K^1 R^1|T^1 \\ ... \\ c_1^F R^F|...|c_K^F R^F|T^F \end{pmatrix} \begin{pmatrix} S_H \\ ... \\ S_H \\ 1 \end{pmatrix} = \begin{pmatrix} R^1|T^1 \\ ... \\ R^F|T^F \end{pmatrix} \begin{pmatrix} S_H \\ 1 \end{pmatrix} \tag{3.8}$$

Notice the left are trajectories in the non-rigid motion subspace; the right are trajectories in the rigid motion subspace formed by all rigid points. So the rigid motion subspace formed by those points must be embedded in the non-rigid motion subspace.

■

Based on the above corollary, we can have a conclusion that applies to both the rigid and non-rigid case. Essentially it is the embedded rigid motion subspace that intersects with that of its linked part. For cases where the non-rigid deformations between the articulated parts are dependent, it might be possible that higher dimensional intersections are obtained.

- If the link is a joint, two subspaces have in general a one-dimensional intersection.
- If the link is an axis, two subspaces have in general a two-dimensional intersection.

Notice that for either case, we do not need to extract the embedded rigid motion subspace out of the non-rigid one in order to find the intersection. We intersect the motion subspaces directly.

We will end the discussion of the non-rigid articulated motion subspace using an example (Fig. 3.6). The non-rigid face has a motion subspace of rank-10 which corresponds to 3 key shapes. Each other part has a rigid motion subspace of rank-4. The joint links results in 1 rank reduction; the axis links, 2. The actual rank of the articulated motion subspace is 31 compared to 46, the sum of the ranks of all parts.

3.5 Finding axes and joints in articulated motion

We provide an algorithm to find the motion subspace of the link in this section.

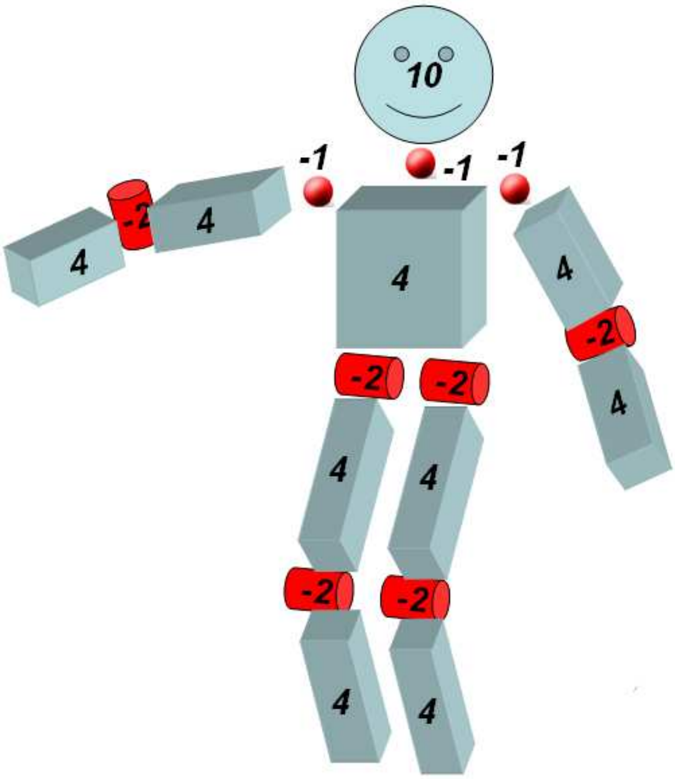


Figure 3.6: The sum of the ranks of all parts is 46. The actual rank of the articulated motion is 31. The rank of each part and the rank reduction resulting from each link are shown.

Let W_1, W_2 be the motion matrices of two linked parts and $\Re(W_1) = E$ and $\Re(W_2) = F$. Due to the rank reduction related to the link as discussed in Section 3.3.1, $\Re(W_1|W_2) = E + F - 2$ for the case of an axis link and $\Re(W_1|W_2) = E + F - 1$ for the case of a joint link.

To find the intersection of two subspaces, we can decompose $W_1 = U_1 \cdot \Sigma_1 \cdot {V_1}^\top$ and $W_2 = U_2 \cdot \Sigma_2 \cdot {V_2}^\top$. Let $U_i^{1:k}$ represent the first k columns of U_i. Let $\begin{pmatrix} N_1 \\ N_2 \end{pmatrix}$ represents a right null vector of $(U_1^{1:E}|U_2^{1:F})$. We have:

$$(U_1^{1:E}|U_2^{1:F}) \cdot \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} = \mathbf{0} \tag{3.9}$$

$$\Rightarrow U_1^{1:E} \cdot N_1 = -U_2^{1:F} \cdot N_2 \tag{3.10}$$

Let $I = U_1^{1:E} \cdot N_1 = U_2^{1:F} \cdot -N_2$. Therefore, I is in the intersection of the subspaces W_1 and W_2. For the joint case, there is only one right null vector. I is the only vector in the intersection. For the axis case, there are two null vectors, I_1 and I_2, in the intersection.

3.5.1 Finding the joint

The trajectory of the joint, T, is in the 1-D intersection. We need to find a proper scale α so that $T = \alpha I$ (Fig. 3.7).

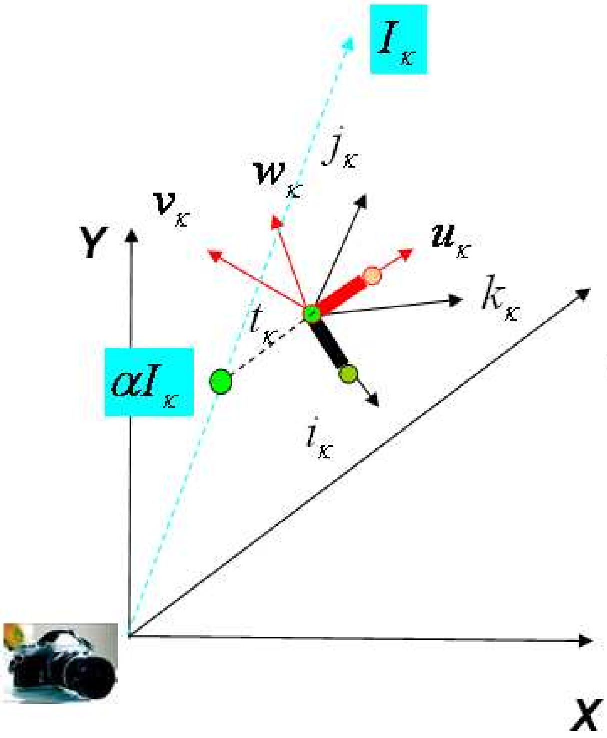


Figure 3.7: This figure illustrates that I and αI in the κth frame. αI_κ coincides with the image of the joint in the κth frame.

The joint is a rigid point attached to either part. The trajectories of a part in respect to the trajectory of the joint will result in a reduction of the rank by one. This is similar to (Tomasi and Kanade, 1992), by placing the world origin on the centroid, the rank of the trajectory matrix goes down from 4 to 3. The equation for obtaining α is:

$$\Re(W_1 - \alpha I(1 \cdots 1)) = \Re(W_1) - 1 = E - 1$$

Given the SVD of $W_1 = \tilde{U}_1 \tilde{\Sigma}_1 \tilde{V}_1^T$ limited to the E non-zero singular values, we have:

$$\Re(W_1 - \alpha I(1 \cdots 1)) = \Re(\tilde{\Sigma}_1 - \tilde{U}_1^\top \alpha I(1 \cdots 1) \tilde{V}_1)$$

Due to rank deficiency, the determinant of $\tilde{\Sigma}_1 - \tilde{U}_1^\top \alpha I(1 \cdots 1)\tilde{V}_1$ is 0. Using the multilinear properties of determinants and introducing the notation $A_i = [0, ..., 0, 1, 0, ...0]$ (with the 1 in i-th place), $B_i = I - diag(A_i)$ (i.e. an identity matrix, except for a 0 in i-th diagonal position) and $C = \tilde{U}_1^\top \alpha I(1 \cdots 1)\tilde{V}_1$, we obtain the following constraint:

$$|\tilde{\Sigma_1} + \sum_{i=1}^{E} |\tilde{\Sigma_1} B_i + C diag(A_i)| = 0 \tag{3.11}$$

Note that this constraint is linear in α as all higher order terms disappear due to linearly dependent columns.

3.5.2 Finding the axis

Similar to the joint case, we need to find the proper scales α and β so that αI_1 and βI_2 are the trajectories of two points on the axis (Fig. 3.8).

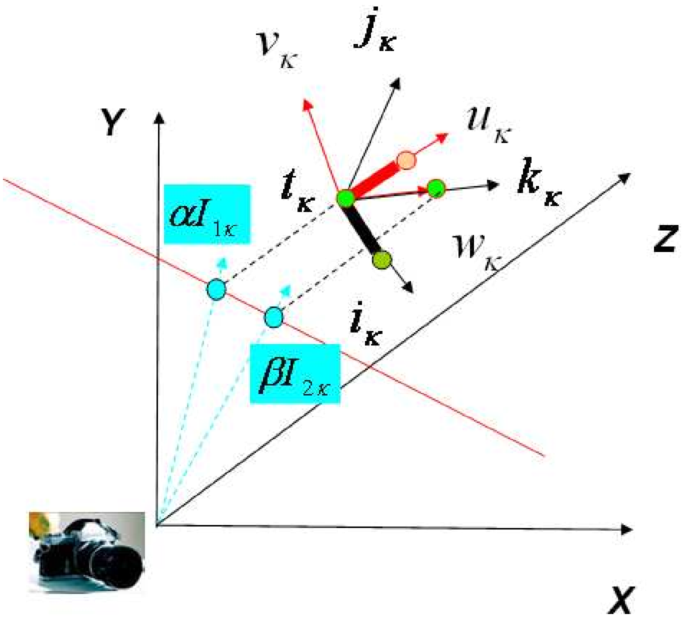


Figure 3.8: This figure illustrates that I_1 and I_2 in the κth frame. $\alpha I_{1\kappa}$ and $\beta I_{2\kappa}$ coincide with the images of two points on the axis in the κth frame.

The equations for obtaining α and β are:

$$\Re(W_1 - \alpha I_1(1 \cdots 1)) = \Re(W_1) - 1 = E - 1$$
$$\Re(W_1 - \beta I_2(1 \cdots 1)) = \Re(W_1) - 1 = E - 1$$

As in the joint case above, the scales of α and β are determined. The axis is the line across both image points in a frame.

3.6 Experiments

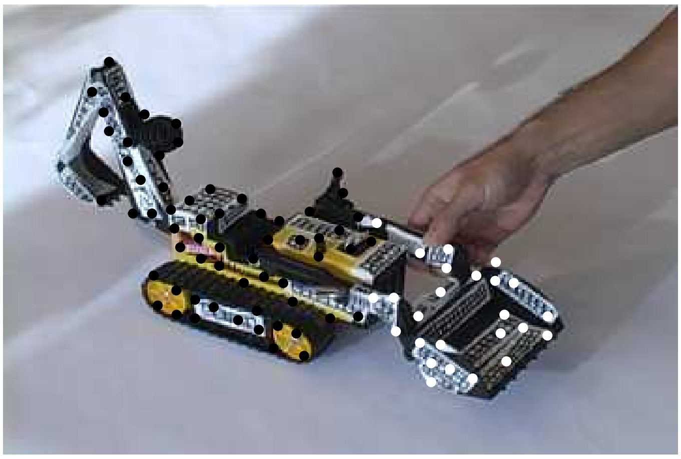


Figure 3.9: Two subspaces are identified from the feature trajectories and segmented accordingly. The color of a feature shows the subspace it belongs to

In a first experiment, a toy truck with a moving shovel is videotaped. A KLT tracker successfully tracks 99 features over 70 frames while the truck moves and the shovel rotates along an axis on the truck. 87 features are left after the first outlier rejection using the rank constraint of the articulated motion. We use an algorithm called GPCA (Vidal and Hartley, 2004) to identify two subspaces from the feature trajectories and segment the trajectories accordingly (Fig. 3.9). GPCA does not scale well with an increasing number of dimensions and subspaces, which are necessary in the non-rigid articulated shapes and motions. Its limitation motivates us to design new algorithms for subspace identification and segmentation which are described in Chapter 4 and Chapter 5.

74 features remain after the second outlier rejection using the rigidity constraint of each part. An axis is identified by intersecting the motion subspaces of the articulated parts. Its image positions are recovered using our algorithm (Fig. 7.2).

Each articulated part is recovered as a rigid shape using the factorization method (Tomasi and Kanade, 1992). By putting all parts into the camera coordinates, shape and motion of the articulated object gets recovered. Furthermore, with the axis recovered in the camera coordinates, we can reanimate the articulated motion by rotating a

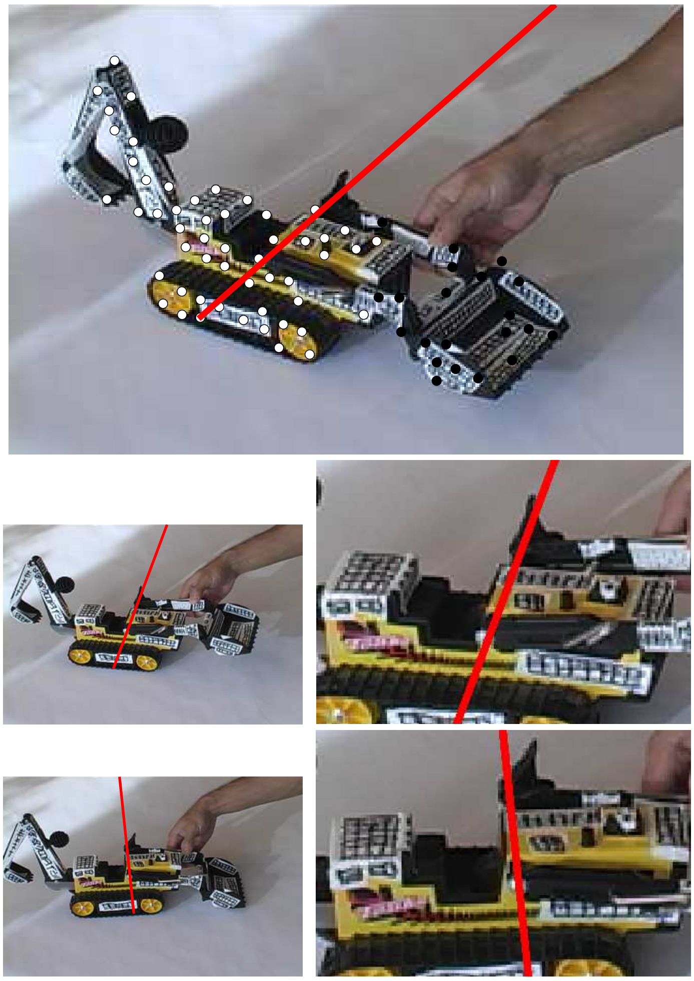


Figure 3.10: (top) An axis is identified by intersecting the motion subspaces of the articulated parts. (middle and bottom) The recovered axis in two frames are shown.

part around the axis and generate not only novel views but also novel motions (Fig. 7.3).

In a second experiment, a person moving his upper body and head is videotaped. 99 features over 60 frames are tracked while the upper body moves and the head rotates around the neck. 93 features are left after the first outlier rejection. The GPCA algorithm identifies two subspaces from the feature trajectories and segment the trajectories accordingly (Fig. 4.5). 92 features remain after the second outlier rejection. The articulated joint is identified (Fig. 3.13).

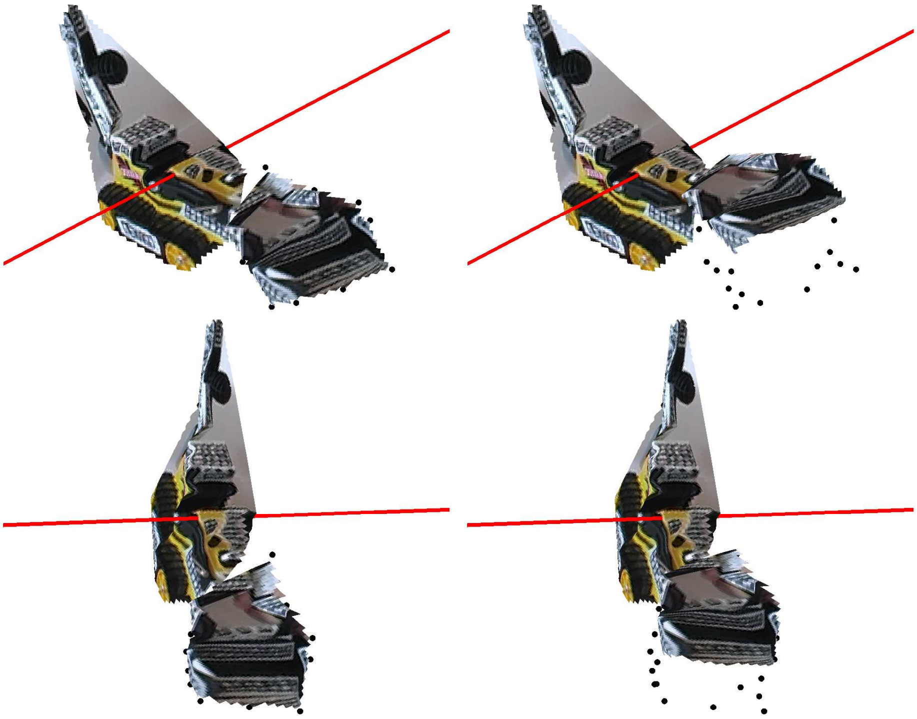


Figure 3.11: The shape and motion of the truck get recovered and reanimated. The black dots show the original position of the shovel. Not only novel views but also novel motions are generated by rotating the shovel around the axis.

3.7 Conclusions

We have shown that the subspace of articulated motion is a set of intersecting rigid motion subspaces. We have derived the rank constraint of articulated motion and proved that the intersection of the motion subspaces is the motion subspace of an axis or a joint. We demonstrate a factorization-based approach to articulated motion recovery based on subspace clustering. The articulated shape and motion, as well as the axes and joints, are recovered using our approach.

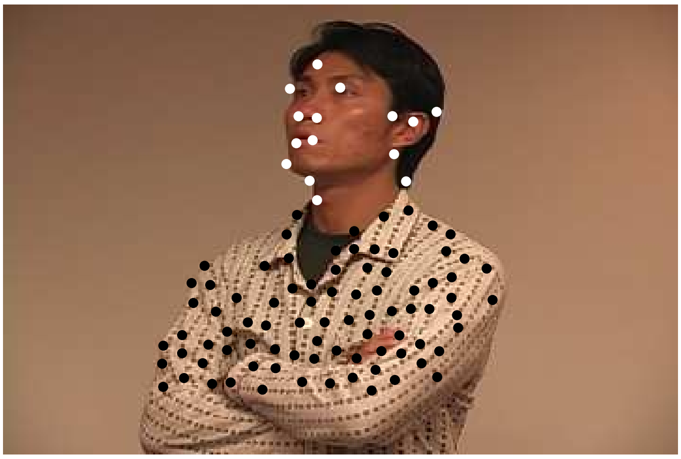


Figure 3.12: The GPCA algorithm identifies two subspaces from the feature trajectories and segments them accordingly. The color of a feature shows the subspace it belongs to

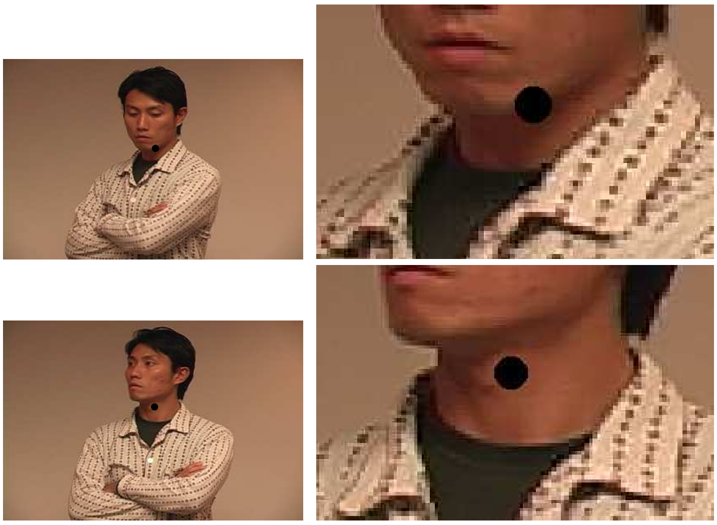


Figure 3.13: A joint is identified by intersecting the motion subspaces of the articulated parts. The recovered joint in two frames are shown.

Chapter 4

Motion Segmentation I: Local Subspace Affinity

Before insights of Section 3.3 can be used, motion segmentation is required. The problem that motion segmentation tries to solve is to segment trajectories according to the underlying motion subspaces that they belong to. It is a challenging problem for the articulated motion due to dependencies between motion subspaces of linked parts. As discussed in Section 2.2, current techniques are insufficient. We will discuss two of our algorithms (Yan and Pollefeys, ECCV 2006; Yan and Pollefeys, ICCV WDV 2005) in this and the following chapter.

The first algorithm is called local subspace affinity. It utilizes two properties of trajectory data for the segmentation purpose, geometric constraint and locality. The geometric constraint is that the trajectories of the same motion lie in a low dimensional linear subspace and different motions form different linear subspaces; locality is that in a transformed space a data and its neighbors tend to lie in the same linear subspace. These two properties provide cues for efficient estimation of underlying subspaces. Our algorithm estimates a number of linear subspaces, whose dimensions are unknown beforehand, and segment the trajectories accordingly. It first transforms and normalizes the trajectories; secondly, for each trajectory it estimates a local linear subspace through local sampling; then it derives the affinity matrix based on the principal subspace angles between these estimated linear subspaces; at last, spectral clustering is applied to the matrix and gives the segmentation result. Our algorithm is general without restriction on the number of linear subspaces and without prior knowledge of the dimensions of the linear subspaces. We demonstrate in our experiments that it can segment a wide range of motions including independent, articulated, rigid, non-rigid, degenerate, non-degenerate or any combination of them. In some very challenging cases where other state-of-the-art motion segmentation algorithms do not work well, our algorithm gives satisfactory results.

4.1 The Algorithm

In this section, we first outline our algorithm and discuss the details of each step. In the end, we will discuss the issue of outliers.

Our algorithm first transforms the trajectory data; then it estimates a local linear subspace for each trajectory by local sampling; it derives an affinity matrix based on principal subspace angles between each pair of local linear subspaces; spectral clustering is applied to the matrix and gives the segmentation result.

4.1.1 Motion data transformation

There are two operations at this step. We will first describe them and then explain their purposes.

- Given a trajectory matrix $W_{2f\times p}$, decompose it into $U_{2f\times K}$,$D_{K\times K}$ and $V^T_{K\times p}$ by SVD, assuming $\Re(W)$ is K (an algorithm for rank detection is described in Section 4.1.6).

- Normalize each column of $V(:, 1 : K)^T$. Each column unit vector $v_i(i = 1...p)$ becomes a representation of the corresponding trajectory.

The columns of the trajectory matrix $W_{2f\times p}$ are to be segmented based on their underlying subspaces. Trajectories from the same motion lie in the same rigid or non-rigid motion subspace while those from different motions do not. These underlying motion subspaces may or may not intersect with each other.

This step serves two purposes. One is to reduce the noise on the data; the second, to transform non-intersecting subspaces into orthogonal subspaces and intersecting subspaces into intersecting subspaces with orthogonal subspaces except for the intersection (Fig. 4.1)[1]. It will be obvious in the Chapter 4.1.3 that this transformation actually increases the distances between subspaces.

[1] The proof of non-intersecting subspaces is in (Kanatani, 2002); the proof for intersecting subspaces is in Appendix

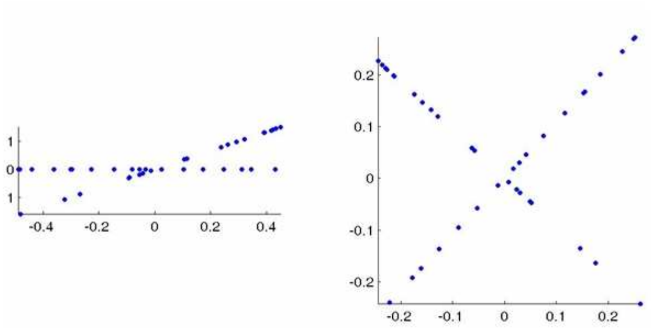


Figure 4.1: Two intersecting subspaces have been transformed into two intersecting subspaces with orthogonal subspaces except for the intersection (the point subspace in this case).

After the transformation, the magnitude of each column vector in $V(:, 1 : K)^T$ bears no direct correlation with its corresponding trajectory. So we remove this factor altogether by the normalization operation.

We are going to perform the segmentation on these unit vectors. It is equivalent to state that we are trying to find a set of R^t spheres $(1 \leq t < K)$ whose union is the R^K sphere. And each vector is grouped according to this set of spheres unless it lies at the intersection of some spheres, in which case it is grouped to either of these intersecting spheres.

4.1.2 Subspace estimation by local sampling

In the transformed subspace, e.g. Fig. 4.2, most points and their closest neighbors lie on the same underlying subspace, which allows us to estimate the underlying subspace of a point α by local samples from itself and its n closest neighbors, i.e. computing the subspace of $[\alpha, \alpha_1, ..., \alpha_n]_{K\times(n+1)}$. This is done using SVD. For rank detection of local estimated subspaces, due to small number of samples, noise level is higher, so we prefer a larger κ (Defined in Section 4.1.6). We can use the angle $\arccos(\alpha^T\beta) \in [0, \frac{\pi}{2}]$ to find the n closest neighbors. Our algorithm is not very sensitive to the choice of n as long as $n + 1$ is not less than the dimension of the underlying subspace (Fig. 4.3).

Two naturally raised questions: what happens to a point near an intersection of subspaces, whose neighbors are from different subspaces (Fig. 4.2). Secondly, what if a point and its n neighbors do not span the whole underlying subspace? We will discuss these two important situations in the following section after introducing the concept of distance between subspaces.

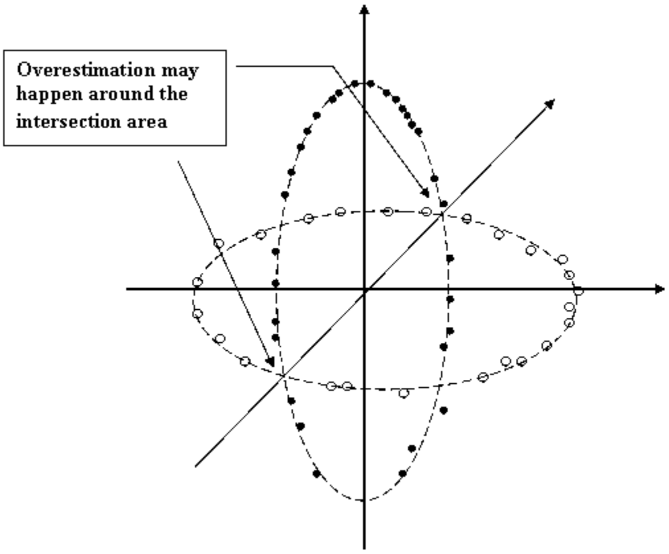


Figure 4.2: There are two underlying subspaces of dimension 2 for the data which are transformed onto the $\mathbf{R^3}$ unit sphere. The empty dots represent a group of transformed data belonging to one subspace and the black dots represent another. Due to noise, the dots may not lie exactly on the $\mathbf{R^2}$ spheres. And the intersection area is where "overestimation" may happen, by which we mean that local sampling results in a local subspace estimation that crosses different underlying subspaces.

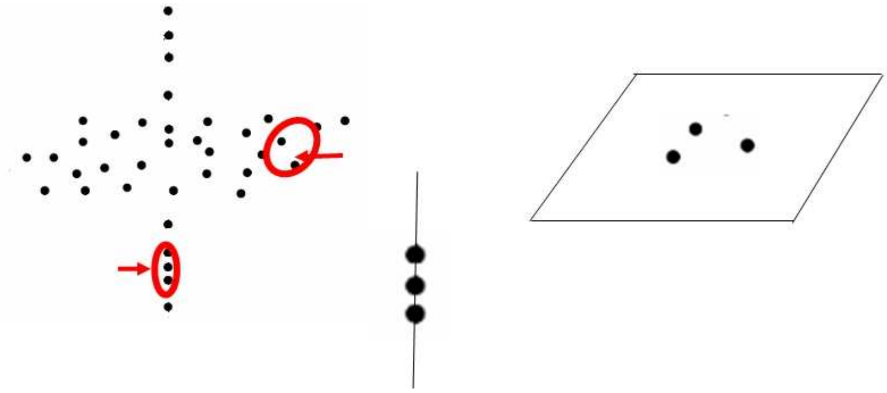


Figure 4.3: Estimation of two underlying subspaces by local sampling.

4.1.3 Distances between local subspaces

We will first introduce the definition of the subspace distance widely used in statistics (Golub and Loan, 1996). Then we will use this definition to define our affinity measure between two points, which is the subspace distance between two points' locally estimated subspaces.

The subspace distance between two subspaces is measured by principal angles. The principal angles between two subspaces P and Q are defined recursively as a series of angles $0 \leq \theta_1 \leq, ..., \leq \theta_M \leq \pi/2$ (M is the minimum of the dimensions of P and Q):

$$\cos(\theta_m) = max_{u \in P, v \in Q} u^T v = u_m^T v_m$$

where

$$\begin{aligned} \|u\| &= \|v\| = 1 \\ u^T u_i &= 0 \quad i = 1, ..., m-1 \\ v^T v_i &= 0 \quad i = 1, ..., m-1 \end{aligned}$$

Consequently the distance between two subspaces P and Q is defined using their principal angles:

$$dist(P, Q) = e^{-\sum_{i=1,...,M} sin^2(\theta_i)}$$

We define the affinity of two points, α and β, as the distance between their estimated local subspaces denoted $S(\alpha)$ and $S(\beta)$.

$$a(\alpha, \beta) = e^{-\sum_{i=1,...,N} sin^2(\theta_i)}$$

where θ_1,...,θ_N are the principal angles of $S(\alpha)$ and $S(\beta)$.

The sine of the principal angle between two locally estimated subspaces of two points instead of the angular or Euclidean distance between two points widely used in previous work in the affinity measure is our contribution that proves to achieve better segmentation results. This is because the distance between two different subspaces where two points lie in is a reliable geometric property for segmentation while the distance between the two points is not *in the case of articulated motions*. Most previous work, e.g. (Costeira and Kanade, 1998; Ichimura, 1999; Kanatani, 2002; Zelnik-Manor and Irani, 2003), use the dot product of two points or some normalized form in the affinity measure for clustering. The dot product measures the angle between vectors and is a "distance" in essence. They assume that the points of the same subspace are closer in "distance". This assumption mostly stems from works for independent motions (Costeira and Kanade, 1998), in which the dot product of two points in two different motion subspaces is 0 and the dot product of two points of the same motion subspace is generally not. That is a strong affinity measure for segmentation in the case of independent motions. But for dependent motions whose subspaces intersect like in Fig. 4.2, the dot product of two points in two different motion subspaces is an arbitrary value between 0 and 1. So is the dot product of two points that are in the same subspace. This diminishes the power of using the distance of two points as a criterion for segmentation. On the other hand, in our proposed affinity measure, the sine of the principal angles between two local estimated subspaces of two points in the same subspace should be 0 without noise. When two points are in different subspaces, the sine is non-zero. So it is a better affinity measure for segmentation in the case of dependent motions compared to the distance of two points.

We are going to build an affinity matrix for spectral clustering in the following section.

Before we proceed to the next section, let us take a closer look at the two scenarios pointed out at the end of Section 4.1.2.

- If an estimated local subspace crosses different underlying subspaces, which happens to points near an intersection as shown in Fig. 4.2 (this usually happens to the trajectories of features very close to or at an articulated axis or joint), we call this estimation "overestimated". An overestimated subspace is usually distanced from the underlying subspaces that it crosses because it has dimensions from other underlying subspace(s). However, points near an intersection are usually small in amount compared to the total. So overestimated subspaces do not have a dominant effect for clustering. Besides, which cluster a point near an intersection may be classified to relies on which underlying subspaces have a larger portion of its overestimated subspace. So in the end it tends to cluster the point to its real underlying subspace. If not, it results in a misclassification. Our experiments show that if there are misclassifications, mostly it happens to points that are close to an intersection.

- If the estimated local subspace is a subspace of the underlying subspace, we call this estimation "underestimated" since it only estimates a part of it. This occurs when the local neighbors may not span the whole underlying subspace. However, this will not affect the effectiveness of the segmentation introduced in the following section. To explain why, we use an example in rigid motions and allow other cases. Suppose two underlying subspaces of dimension 4 having a 2-dimension intersection. This happens when two articulated parts are linked by an axis(Yan and Pollefeys, CVPR 2005; Tresadern and Reid, 2005). The total dimension is 6. The two underlying subspaces, S^1 and S^2, and their underestimated subspaces, A, B, C and D, E, F, of dimension 3 (2 is rare because the features corresponding to the point and its neighbors need to be exactly on a line for that to happen) are as follows.

Subspace \ *Dim*	1	2	3	4	5	6
S^1	X	X	X	X		
A	X	X	X			
B		X	X	X		
C	X		X	X		
S^2			X	X	X	X
D			X	X	X	
E				X	X	X
F			X		X	X

(4.1)

The number of non-zero principal angles between these subspaces and their underestimated subspaces are shown as follows.

Subspaces	S^1	A	B	C	S^2	D	E	F
S^1	0	0	0	0	2	1	2	2
A	0	0	1	1	2	2	3	2
B	0	1	0	1	1	1	2	2
C	0	1	1	0	1	1	2	2
S^2	2	2	1	1	0	0	0	0
D	1	2	1	1	0	0	1	1
E	2	3	2	2	0	1	0	1
F	2	2	2	2	0	1	1	0

(4.2)

Generally, intra-subspaces have smaller number of non-zero principal angles compared to inter-subspaces. So expectedly, an underestimated subspace tends to be closer to all possible estimated subspaces of its underlying subspace than to those of another underlying subspace. Furthermore, in general (not for this example) the non-zero principal angles of intra-subspace local subspace distances are smaller than 90 degrees, while the non-zero principal angles between local subspaces from different subspaces would be 90 degrees.

4.1.4 Spectral clustering

We can apply spectral clustering, e.g.(Shi and Malik, 2000; Ng et al., 2002), to the affinity matrix that we build from the previous step. We advocate recursive 2-way clustering(Shi and Malik, 2000). Thus we can re-estimate the local subspaces within the current clusters so that points belonging to different clusters will not affect each other any more. Secondly, recursive 2-way clustering gives a more stable result than k-way clustering(Ng et al., 2002). The recursive 2-way clustering is as follows:

- Compute the affinity matrix using the approach in Section 4.1.2 and 4.1.3 and segment the data into two clusters $\{C_1, C_2\}$ by spectral clustering.

- For the current clusters, we examine their ranks or other criteria, which will be explained soon, to find those that need further partition. We stop when there is no such cluster. Otherwise, we partition those clusters; evaluate the normalized cut (Shi and Malik, 2000) of each partition to decide which is the best; replace the parent-cluster with the best partition then repeat this step.

  Normalized cut is used to evaluate how good a partition is. Suppose the parent cluster is V which is partitioned into A and B. It is defined as follows (Shi and Malik, 2000).

  $$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

  In our case, $cut(A, B)$ is the sum of the affinities between every pair of points, one from A and the other from B; $assoc(A, V)$ is the sum of the affinities between every pair of points, one from A and the other from V, i.e. A or B; $assoc(B, V)$ is defined similarly. The smaller $Ncut(A, B)$ is, the looser the connection between A and B is.

  The criteria to decide whether a partition of a cluster is needed is the rank of the cluster, the intersection of its partition, etc. In the experiment, we use both. First, we determine if a cluster's rank is less or equal to 4. If so, no partition is needed. Otherwise, it could be either a non-rigid motion or a mixture of articulated or independent motions. We partition it into two clusters. By inspecting the subspace intersection of both clusters, we can determine whether their parent-cluster is from a non-rigid motion or a mixture of motions because two clusters from a non-rigid motion have an intersection of a rank higher than 2. For a non-rigid motion, we shall not replace the parent-cluster with its partitions.

4.1.5 Outliers

In practice, the trajectories may have outliers. We will discuss its effects on our algorithm.

First of all, an outlier will be segmented as other trajectories to one of the segments. However, they are easier to be detected and rejected after the segmentation because the segmented subspaces are more constrained.

Secondly, suppose an outlier is sufficiently far from other points and close to orthogonal to everything else. At the local sampling step, an outlier will of course have some neighbors even if they are far away, but there is no reason for these neighbors to have the outlier as one of their neighbors. There will be other points that are much closer to these points. This neighboring relation is not symmetric. So an outlier that is far away have no effect on the local subspace estimation of a good point.

Thirdly, even if an outlier happens to be close to a good point and corrupts the local subspace estimation of that point. This bad effect will not propagate under our algorithm and only remains limited to those points whose "closest" points include the outlier.

In conclusion, our method is not very sensitive to outliers. And outliers have no or limited effect on most good points.

4.1.6 Effective Rank Detection

In practice, a trajectory matrix is corrupted by noise and outliers and thus its rank is usually full. Without prior knowledge of the scene and object, we may use a model selection algorithm inspired by a similar one in (Vidal et al., 2004) to detect an effective rank.

$$r_n = \arg\min{}_r \frac{\lambda_{r+1}^2}{\sum_{k=1}^{r} \lambda_k^2} + \kappa\, r$$

with λ_i the i^{th} singular value of the matrix and κ a parameter. If the sum of all λ_i^2 is below a certain threshold, the effective rank is 0.

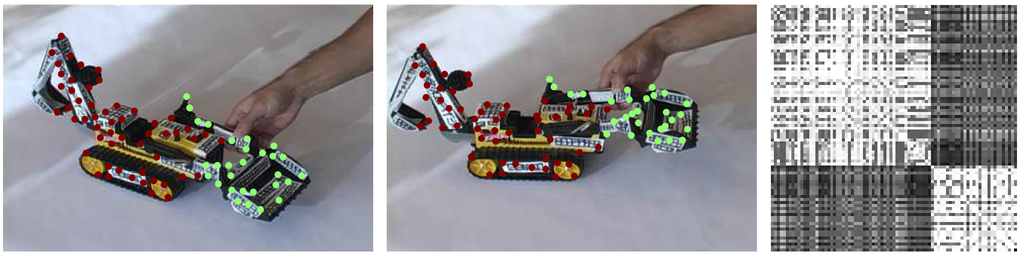


Figure 4.4: (*left and middle*) A sequence of a truck moving with the shovel rotating around an axis. The color of a dot, red or green, shows the segmentation result. (*right*) The affinity matrix of local estimated subspaces is shown. The row and columns are rearranged based on the segmentation.

The advantage of the above algorithm is its simplicity. It only requires a κ parameter. The disadvantage is that a manual-tuning of κ is needed. Other more sophisticated algorithms (P.H.S. Torr, 2002) of model selection may be used as well.

4.2 Experiments

We test our approach in various real dynamical scenes with 2 to 6 motions and a combination of independent, articulated, rigid, non-rigid, degenerate and non-degenerate motions.

For the experiments, we choose $\kappa = 10^{-6}$ to 10^{-5} for trajectory rank estimation depending on the noise level of the trajectories, and $\kappa = 10^{-3}$ for local subspace rank estimation (See Section 4.1.6 for more detail). We let $n = d$ where n is the number of neighbors for local sampling and d is the highest possible dimension of the underlying subspaces. That is 4 for rigid motions and 7 for non-rigid motions in our experiments.

Misclassification errors vs. total number of trajectories and the number of outliers vs. total number of trajectories for the experiments is summarized in Table 4.1. Outliers may be clustered to any segments and are not counted as misclassification errors.

The first experiment is from a scene with non-degenerate data of an articulated object with a rotating axis. The detected rank of the trajectories is 6. The segmentation result is shown in Fig. 4.4. The ranks of the segmented trajectories are both 4.

The second experiment is from a scene of 2 non-degenerate motions of an articulated body with a joint. The detected rank of the trajectories is 7. There are one misclassification, the red dot on the left shoulder. The other red dot on the left arm is an outlier. The segmentation result is shown in Fig. 4.5. The ranks of the segmented trajectories are both 4.

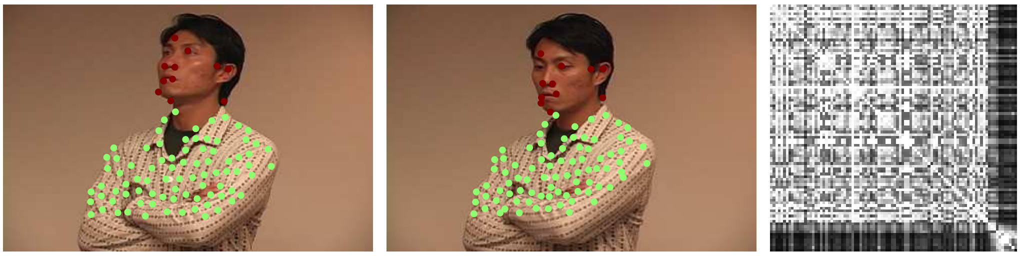


Figure 4.5: (*left and middle*) A sequence of a person moving with his head rotating around the neck. The color of a dot, red or green, shows the segmentation result. There is one misclassification, the red dot on the left shoulder. The other red dot on the left arm is an outlier. (*right*) The affinity matrix of locally estimated subspaces is shown. The row and columns are rearranged based on the segmentation.

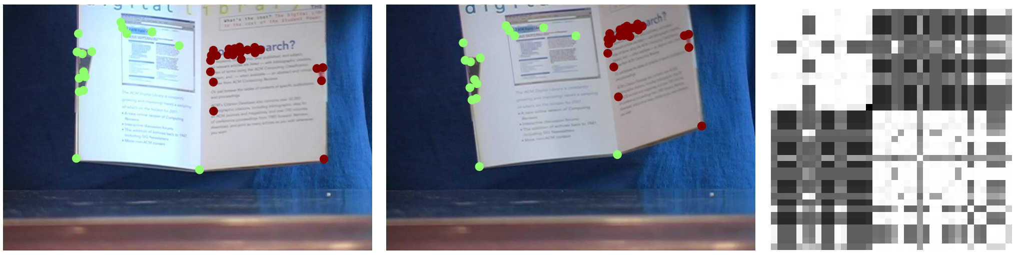


Figure 4.6: (*left and middle*) A sequence of a booklet whose two pages are being opened and closed around an axis. The color of a dot, red or green, shows the segmentation result. There is no misclassification error. The green dot on the rotating axis can be grouped to either page. (*right*) The affinity matrix of local estimated subspaces is shown. The row and columns are rearranged based on the segmentation.

The third experiment is from a scene of 2 degenerate shapes of an articulated object. Two pages of a booklet is being opened and closed. The detected rank of the trajectories is 4. The segmentation result is shown in Fig. 4.6. The ranks of the segmented trajectories are both 3.

The fourth experiment has 3 motions. It comes from a scene of 2 independently moving bulldozers, one of which has an articulated arm rotating around an axis. Only the side of the articulated arm is seen so it is a degenerate shape. The detected rank of the trajectories is 8 before the first segmentation. Both of the segments have rank 4. The next subdivision is automatically detected (See Section 4.1.4) for points from the right bulldozer and the segmented trajectories are of rank 3. The segmentation result is shown in Fig. 4.7.

The fifth experiment has 3 motions. It comes from a scene with an articulated object of 3 parts. The bulldozer has its forearm and upper-arm moving articulately

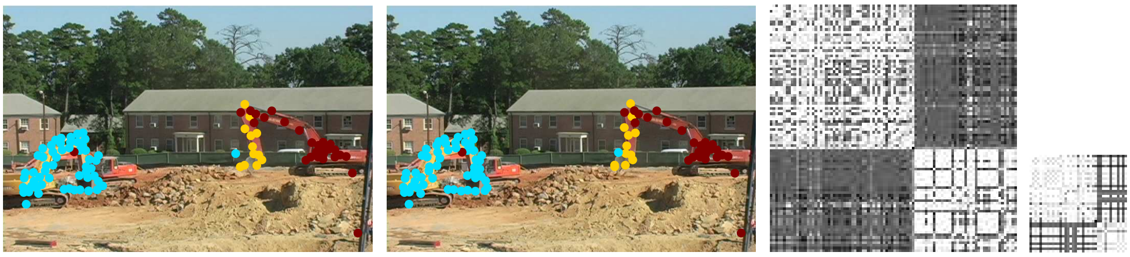


Figure 4.7: (*left 2*)A sequence of two bulldozers moving independently, one of which moves articulately with its arm rotating around an axis. The color of a dot, red, blue or yellow, shows the segmentation result. There is one misclassification error which is the red dot on the forearm near the axis. Besides that, there are several outliers. (*right 2*) The affinity matrices for 2-stage segmentations are shown. The row and columns are rearranged based on the segmentation.

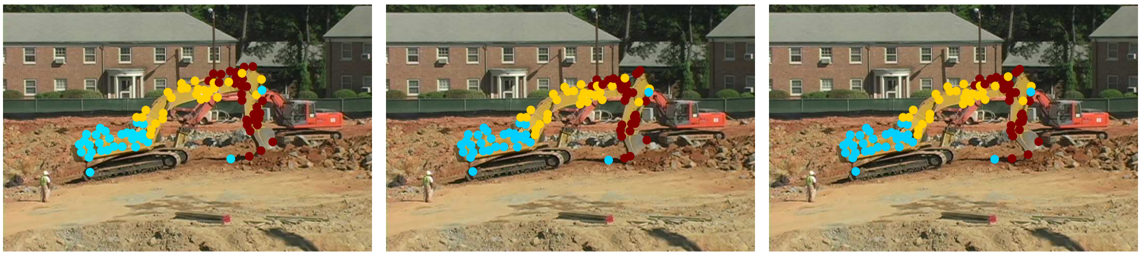


Figure 4.8: A sequence of a bulldozer with its upper-arm and forearm moving articulately around some axis. The color of a dot, red, blue or yellow, shows the segmentation result. There are 4 misclassification errors. Two are the yellow dots on the forearm and two are the red dots on the upper-arm. All of them are near the axis connecting both arms. Besides these, there are several outliers in the trajectories and they are clustered to one of the segments.

rotating around two axes. The detected rank of the trajectories is 6 before the first segmentation. The segmented trajectories have a rank 3 and 4. The rank-4 cluster gets subdivided into 2 rank-3 clusters. There are outliers in this experiment. They have been clustered to one of the segments. Besides outliers, there are 4 misclassifications, two of which are the yellow dots on the forearm and two of which are the red dots on the upper-arm and all of which are near the axis. The segmentation result is shown in Fig. 4.8.

The final experiment has 6 motions. It comes from a scene with a person dancing with his upper body, his head, both upper arms and both forearms moving. Besides, his mouth movement generates a non-rigid motion. This is a highly challenging case not only because of the total number of motions but also because of the dependence between these articulated motions and the non-rigid kind of motion on the person's

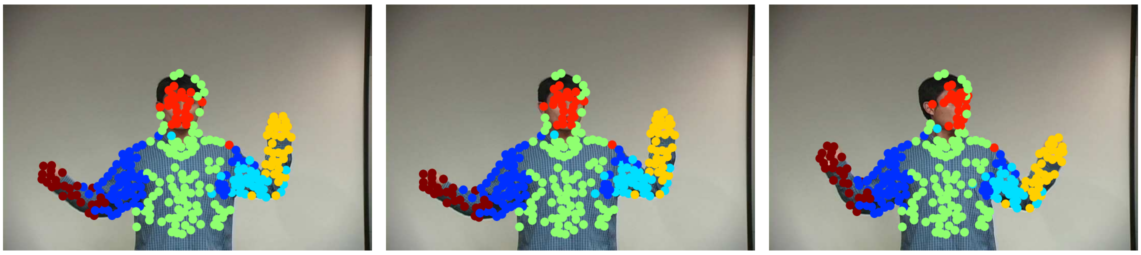


Figure 4.9: (*top*) A sequence of a person dancing with his upper body, his head and both of his upper arms and forearms moving. His mouth motion is non-rigid. The color of a dot shows the segmentation result. Besides outliers, there are about 8% misclassifications.

face. The detected rank of the trajectories is 12 before the first segmentation. Both of the segmented rank-7 and rank-6 clusters get subdivided into rank-6 and -3 clusters, and rank-4 and -3 clusters repectively. In the end, the rank-4 cluster gets subdivided into two rank-3 subspaces. There are outliers and there are about 8% misclassifications, most of which are near the articulated axes or joints. Interestingly, the green dots on the head are those features not turning as the head turns. Instead, they move like the features on the upper body. And indeed, our algorithm classifies them to those features on the upper body. A second interesting observation is the misclassification of the dark blue dots near the joint between the body and the person's right arm. Though they are far away from the left upper arm of the person, they are actually very close to the intersection between the motion subspaces of the left and right upper arms because they both are linked to the body. The segmentation result is shown in Fig. 4.9.

Comparisons

We compare our method with GPCA (Vidal and Hartley, 2004) and trajectory angle based approach, e.g. (Zelnik-Manor and Irani, 2003) except for that we use spectral clustering to segment the affinity matrix (Table 4.1) while in (Zelnik-Manor and Irani, 2003) they use an agglomerative clustering algorithms which starts with some initial groups, adds points to them or merge them subsequently. The numbers in the table are misclassification errors vs. the total number of trajectories and outliers vs. trajectories. Outliers are not counted as misclassification. Angle based approach performs the worst. It proves that the "distance" between points is not a good affinity measure for segmenting dependent subspaces. GPCA performs better than the angle based approach. But it fails to handle the "dancing" experiment because it requires too large a

Table 4.1: A comparison of between our method, GPCA and trajectory angle based method (misclassifications / trajectories)

Experiment	**Our Method**	**GPCA**	**Angle based**	**Outliers**
Truck	0/83	5/83	16/83	0/83
Head and Body	1/99	10/99	9/99	6/83
Booklet	0/38	2/38	2/38	1/38
Two bulldozers	1/94	4/94	24/94	8/94
One bulldozers	4/85	6/85	11/85	9/85
Dancing	21/268	not enough samples[2]	78/268	7/268

number of samples. Our method achieves higher accuracy in segmentation results and can handle a wider range of cases in terms of the number of motions.

4.2.1 Conclusions

We propose a general framework for motion segmentation of a wide range of motions including independent, articulated, rigid, non-rigid, degenerate and non-degenerate. It is based on local estimation of the subspace to which a trajectory belongs through local sampling and spectral clustering of the affinity matrix of the these subspaces. We demonstrate our approach in various situations. In some highly challenging cases where other state-of-the-art motion segmentation algorithms may fail, our algorithm gives expected results.

[2] For the last experiment of 6 motions, GPCA requires a huge number of trajectories for it to work. Roughly, it needs $\bigcirc((d+1)^6)$. d is the dimension of the largest subspace. For non-degenerate rigid motions, $d = 5$(Vidal and Hartley, 2004); for non-rigid motions, d is even larger. That many number of trajectories are normally not available in practice.

Chapter 5

Motion Segmentation II: RANSAC with Priors

We are going to introduce another algorithm for articulated motion segmentation in this chapter.

We have established the concept of motion subspaces and particularly pointed out that an articulated motion subspace consists of a number of intersecting motion subspaces either rigid or non-rigid (See Chapter 3). Most of the existing segmentation methods, e.g. (Costeira and Kanade, 1998), assume the independence between motions and thus can not be applied to articulated motions.

We propose a novel algorithm for articulated motion segmentation called RANSAC with priors. It does not require prior knowledge of the number of articulated parts. It is both robust and efficient. Its robustness comes from its RANSAC nature. Its efficiency is due to the priors, which are derived from the spectral affinities between every pair of trajectories.

The limitation of the method compared to the previous one is that it requires the dimensions of motion subspaces to be known as a prior. However, the advantages of being able to handle an unknown number of motions and being linear to the number of motions in time complexity override the limitation in some cases and make it attractive in cases like dealing with the articulated motion of a humanoid robot whose parts are rigid and whose number of parts could be high.

We test our algorithm with synthetic and real data. In some highly challenging case, where other motion segmentation algorithms may fail, our algorithm still achieves robust results.

Though our algorithm is inspired by articulated motions, it also applies to independent motions which is regarded as a special case and treated uniformly.

5.1 RANSAC

We are going to describe RANSAC (Fischler and Bolles, 1981) briefly in this section. Then we discuss RANSAC for multiple models.

Suppose that there is a dataset of $X_1, X_2, ...X_N$. Some of the X_i's are generated from a model $f_{(a,b,...)}(X) = 0$ and the rest are outliers. $(a, b, ...)$ are the unknown parameters and there is noise within these data. Suppose that the parameters of $f_{(a,b,...)}(X) = 0$ can be estimated from M data.

The purpose of RANSAC is to robustly estimate the parameters $(a, b, ...)$ of the model $f_{(a,b,...)}(X) = 0$ from the dataset even if there are outliers and noise in the dataset.

- Randomly choose M data from the dataset $X_1, ..., X_N$ and use them to estimate $(a, b, ...)$ of the model.

- Find all the data in the dataset that fits the model $f_{(a,b,...)}(X) = 0$ within a user-specified tolerance T, i.e. if $\|f(X_i)\| < T$ then X_i is regarded as fitting the model.

- If the total number of fitted data is over a user-specified number C, we exit with success.

- Repeat the above steps K times.

- Failed if this step is reached.

Notice that after successfully finding a consensus set the parameters of the model are re-estimated using the consensus set of data. This will generally result in a more accurate estimate of the parameters.

There are advanced topics on how to choose K, T and C by probability and statistics analysis. Interested readers are referred to (Fischler and Bolles, 1981) and other related research (P.H.S. Torr, 1995; Meer et al.,1991) for details.

Fig. 5.1 illustrates the algorithm for finding the parameter of a linear model from a set of 2-D data.

We can take the discussion of the original RANSAC to the case of fitting data generated from multiple models.

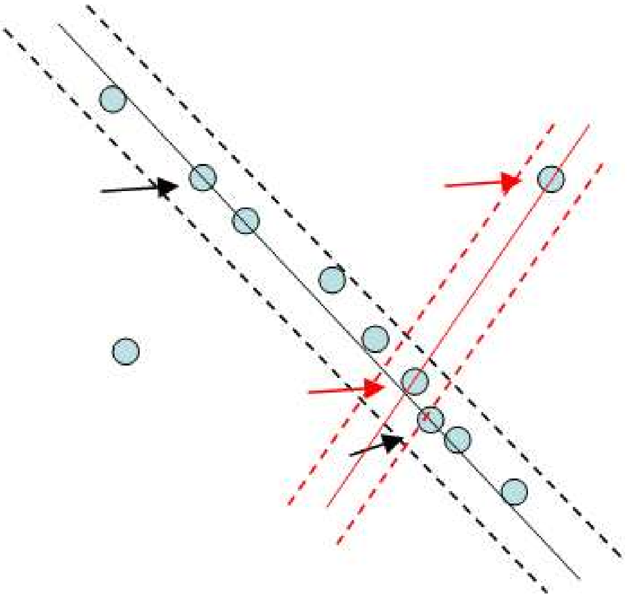


Figure 5.1: RANSAC illustrated: the blue dots are the 2-D dataset generated by a linear model with outliers and noise. The solid black line is the estimated model from sampling two data pointed by two black arrows. The dotted black lines indicate the thresholds of tolerance. All blue dots within the tolerance range are a consensus set. If any of the outliers are sampled during the estimation, like the two red arrows show, the linear model does not have a large consensus set.

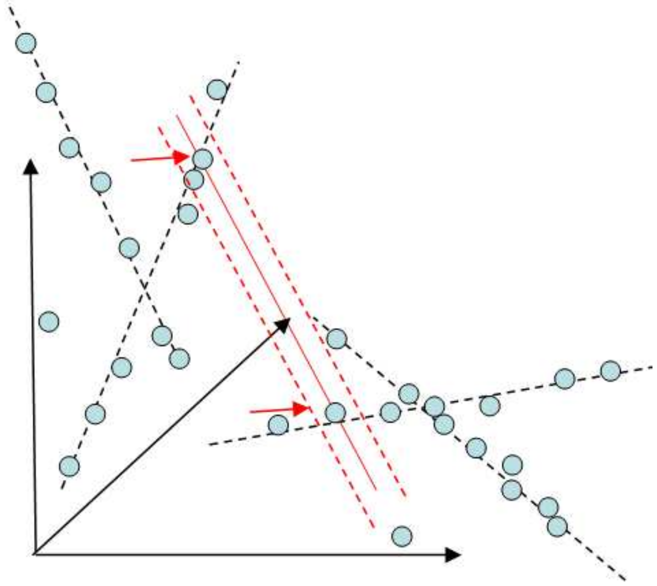


Figure 5.2: The blue dots are the 3-D dataset generated by 4 linear models (shown by 4 black dotted lines) with outliers and noise. The red line illustrates the scenario of two sampled data belonging to different models. The estimated model does not have a large consensus set.

Fig. 5.2 illustrates the case. Suppose there is a dataset generated by 4 linear models, we try to find the parameters of these models so they fit the data best. The data may contain outliers and noise.

A major difficulty of applying RANSAC to multiple models is that it needs a large number of samplings before finding a consensus set. As illustrated in Fig. 5.2, because most of the sample data come from different underlying models, they can not generate a large consensus set. It is when both of the sample points happen to be within the same model that we can find a consensus set and estimate the parameters of one underlying model. One possible approach is to provide prior information to RANSAC to enhance the probability of successful samplings.

5.2 Articulated Motion Subspaces

In this section, we will formulate the articulated motion segmentation problem as a RANSAC-for-multiple-model problem and discuss how we may provide priors to RANSAC to overcome the inefficiency of RANSAC for multiple models.

As discussed in Chapter 3, the articulated motion subspace consists of a number of intersecting linear subspaces. Suppose that the articulated parts are rigid and the shape and motion are general, the trajectories are generated from different 4-D subspaces. The problem of articulated motion segmentation is formulated as fitting the trajectory dataset with noise and outliers to N rank-4 motion subspaces. We need to estimate the parameters of the N subspaces, i.e. 4 bases for each subspace.

Next, we will discuss how the dependency between articulated motion subspaces can provide us priors for RANSAC in motion segmentation.

5.2.1 The Shape Interaction of Articulated Motion Subspaces

Each trajectory has a corresponding column vector in the shape matrix which is the right most matrix in Equation 3.1, 3.2 and 3.3 for multiple independently moving objects, articulated objects with links and those with axes respectively.

For independent motions (Equation 3.1), column vectors of different shape subspaces have zero inner products while column vectors of the same subspace generally do not. The shape interaction matrix proposed in (Costeira and Kanade, 1998) consists of these inner products of every pair of trajectories, so it is used to group features of the same motion.

For articulated motions (Equation 3.2 and 3.3), though the shape subspaces are not orthogonal, column vectors of the same shape subspace generally have larger inner products than those from different shape subspaces in magnitude. We will show that in the following. The first shape subspace in Equation 3.2 is represented by a base (e_1, e_2, e_3, e_7) where $e_i = [0, ..., 1, ...0]^T$ with i indicating the position of 1. Similarly, the second shape subspace is represented by a base (e_4, e_5, e_6, e_7). It is easy to see that the inner product of column vectors from different shape subspaces has only one coefficient not canceled out while that of column vectors from the same shape subspace has four. This observation implies that the magnitude of the former is generally smaller than that of the later. A similar analysis applies to Equation 3.3.

So the inner products of column vectors may tell us how likely two trajectories are of the same motion. This key observation is what RANSAC with priors builds upon. In the following section we will describe how to estimate the priors with regard to how likely every pair of trajectories belong to the same motion and present our segmentation approach, RANSAC with priors.

5.3 RANSAC With Priors

In this section, we will first describe how to build the priors to guide RANSAC. Then we will discuss RANSAC with priors.

5.3.1 The Prior Matrix

Similar to the construct of the shape interaction matrix in (Costeira and Kanade, 1998), we compute the prior matrix that provides cues for how likely two trajectories are of the same part of an articulated object. The procedure is described as follows.

- SVD the trajectory matrix W into U, D, V^T up to rank k.
- Normalize each column of V^T. The new matrix is Q.
- Compute the prior matrix $P = Q^T Q$.

$$P_{ij} = \frac{2}{\sqrt{\pi}} \int_0^{q_i^T q_j} e^{-t^2} dt \tag{5.1}$$

  P_{ij} represents the probability of trajectory i belonging to the same motion subspace of trajectory j; q_i and q_j are the ith and jth column of Q.

A few discussions:

- The choice of rank k. Ideally, k should be the rank of $4N$ (N is the number of articulated parts). In practice, due to noise, the rank k can only be estimated. We may use a model selection algorithm inspired by a similar one in (Vidal et al., 2004) to detect the rank.

$$r_n = \arg\ \min_r \frac{\lambda_{r+1}^2}{\sum_{k=1}^r \lambda_k^2} + \kappa\ r$$

  with λ_i, the i^{th} singular value of the matrix, and κ, a parameter. If the sum of all λ_i^2 is below a certain threshold, the estimated rank is zero.

  Notice that due to outliers the estimated rank may be larger than the rank of the motion subspace. However, the prior matrix is not very sensitive to a larger k.

- Any reasonable distribution function may substitute for Equation 5.1. The point is to use the spectral affinity to build priors with regard to how likely two trajectories belong to the same motion.

5.3.2 RANSAC with Priors

P_{ij} represents the probability of trajectory (or data) i belonging to the same motion (or model) as trajectory (or data) j.

We outline our segmentation approach, RANSAC with priors, as follows.

- Find consensus sets using the priors P_{ij}:

  1. Randomly choose the first data s_1 based on a probability distribution formed by the sums of each row of the prior matrix. A larger row sum indicates that the data is more likely in the same motion as other data.

  2. Randomly choose the 2nd to the kth data, s_2,...,s_k, based on a probability distribution formed by the priors related to data s_1.

  3. Instantiate a model from this sample set.

  4. Determine the set of data S_i that are within a threshold t of the model.

  5. Repeat the steps of 2, 3 and 4 for M times without changing the first data s_1. Each time, we should have a new consensus set S_i.

  6. Repeat the steps of 1, 2, 3, 4 and 5 for N times.

- The largest consensus set is selected and the model is re-estimated using all the points in that consensus set. If the largest consensus set has a size less than some threshold T, terminate.

- Remove the data of the largest consensus set from the original data and repeat the above to find another largest consensus set and its model until either the data is exhausted or no more models can be found from the remaining data.

A few discussions:

- The model that we use is the factorization model (Tomasi and Kanade, 1992) which states that the trajectories of a full rigid motion generally span a rank-4 subspace. So k is 4 in our experiments.

- We introduce M times of sampling by fixing the first data chosen. First, with the priors, the first data is very possible from one of the models. Secondly, with the priors, the 2nd to the kth data chosen are very likely to be within the same models. We add this M times of sampling to increase the probability that we find the consensus set for this model. Without doing this, we may pass by a good chance to get to the consensus set of this model because we would have chosen the first data again and it may be from another model instead.

M is possible to be determined without knowing the total number of models. Because with the priors, the 2nd to kth data should be chosen with high probabilities from only a small number of models instead of the total number of models. For example, in articulated motions, only trajectories of linked parts have priors higher than 0. Parts that are not linked generally are independent from each other. So after the first data is chosen, the 2nd to kth sampling are most probable from the part of the first data; less but still probable from its linked parts which we know it is usually 1 or 2; very unlikely from parts that are not linked to it. So we can determine M based on this and use it to save us unnecessary repetitions. Suppose that it is 2 times likely that the 2nd to kth data is chosen from the part of the first data compared to any of its 2 linked parts, it is $\frac{1}{2}$ chance that it is from the part itself and the same chance that it is from its linked parts. So it is $\frac{1}{2^{k-1}}$ chance that all data are chosen from the same part or $1 - \frac{1}{2^{k-1}}$ chance that they are not. With $k = 4$ in our case, the chance that all data are not from the same part is $\frac{7}{8}^{M}$. If $M = 20$, that chance is around 6%. We use $M = 20$ in our experiments.

The total number of parts is irrelevant in determining M except for the part that has links to all other parts. Fortunately, we can ignore that without hurting the algorithm from finding other parts first. After removing data of other parts are removed from the data set, the part that has links to all other parts has less and less links until it is either 1 or 2.

- N is possible to determined now without knowing the total number of models too given the above M discussed. Basically increasing N will decrease the possibilities of two bad cases. One is that we do not get any valid consensus set from the fixed first data in its M loops. Notice that there is still 6% chance of that by the assumption described above. By having $N = 3$, that chance is decrease to around 0.02%. The second bad case is that N avoids that the first data is always an outlier, which is unlikely anyway because the sum of the priors of an outlier is much lower compared to that of a valid data. We use $N = 3$ in our experiments.

- Model selection is naturally combined with RANSAC with priors to deal with degenerate shape and motion. This will be discussed in Section 5.5.

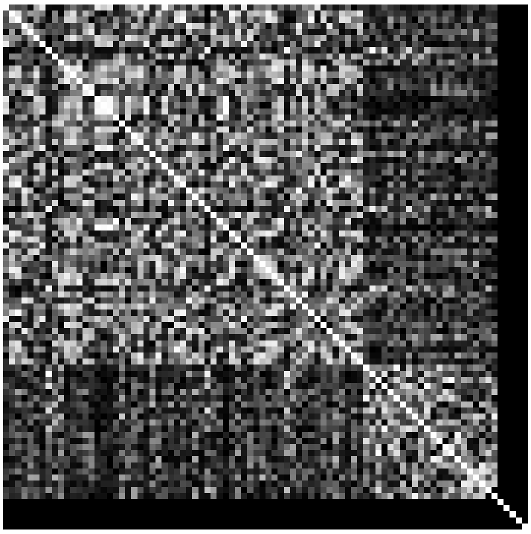


Figure 5.3: *The prior matrix of the truck sequence with outliers. Lighter color indicates higher probability.*

5.4 Experiments

We test RANSAC with priors in three experiments.

The first experiment consists of a truck sequence with a moving shovel. Connected by an axis, the motion dependence is the highest for articulated motions. To demonstrate the robustness of our approach, besides those erroneous trajectories due to tracking, outliers are created by adding large random noise (larger than 10%) to some existing trajectories. The prior matrix is shown in Fig. 5.3. The actual rank of the articulated motion subspace is 6 while the detected rank is 13 because of outliers and noise. For illustration purpose, the trajectories have been grouped into the truck body, the shovel and random outliers. Notice the priors for random outliers have very small values which makes them unlikely to be selected into a sample set. And the erroneous trajectories are rejected when RANSAC with priors tries to find a largest possible consensus set. With $M = 20$ and $N = 3$ (See Section 5.3.2), 60 samples are tried each time to find the largest possible consensus set from the current data. RANSAC with priors finds 2 motions and terminates if the largest possible consensus set that it can find has a size 6 which is less than the threshold $T = 8$. Those 6 trajectories are some of the erroneous trajectories on the shovel and on the body. The remaining data consist of erroneous trajectories and the outliers that we add (Fig. 5.4).

The second experiment is from a sequence of synthetic data of 4 linked parts. Each parts has 10 features to represent its 3D shape. Small random noise (less than 1%) are

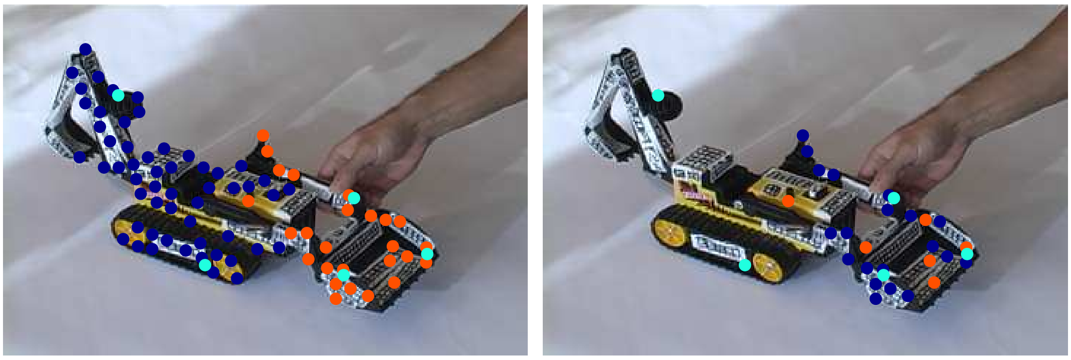


Figure 5.4: *(left) RANSAC with priors finds the first consensus set indicated by blue dots. The orange and light blue dots are the remaining data. The light blue dots are outliers. The orange dots on the truck body are erroneous trajectories. (right) The blue dots indicate the second consensus set found by RANSAC with priors. The orange and light blue dots are the remaining data. The light blue dots are the outliers. The orange dots are erroneous trajectories.*

added to the trajectories. 4 outliers are created by adding large random noise (larger than 10%) to some existing trajectories.

This experiment is challenging. First, each part has a small number of trajectories which provides too few data for GPCA (Vidal and Hartley, 2004) to work; secondly, RANSAC WITHOUT priors will require a large number of times of sampling before it may obtain a valid sample set,i.e. a sample set consisting of trajectories from the same part. In this experiment only $\frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} = \frac{1}{64}$ would yield a valid sample set. Furthermore, without knowing the number of motions beforehand, it is impossible to set a fixed threshold for the number of sampling times.

However, RANSAC with priors generally gets one valid sample set out of every three in this experiment. And this rate does not depend on the total number of motions. It only depends on the number of dependent motions. As discussed in Section 5.3.2, with $M = 20$ and $N = 3$, 60 samples are tried, during each of which it finds a largest possible consensus set from the current data in this experiment. The prior matrix of 4 linked parts is shown in Fig. 5.5. The actually rank of the articulated motion subspace is 13 but the detected rank is 17 due to outliers. RANSAC with priors finds 4 motions within the trajectories and the segmentation is shown in Fig. 5.6 with reference lines representing each part of the object for better illustration. The remaining data are 4 random outliers after RANSAC with priors can not find any consensus set of size more than $T = 8$. The result matches the ground truth.

Last but not the least, we test RANSAC with priors in a more complex scenario.

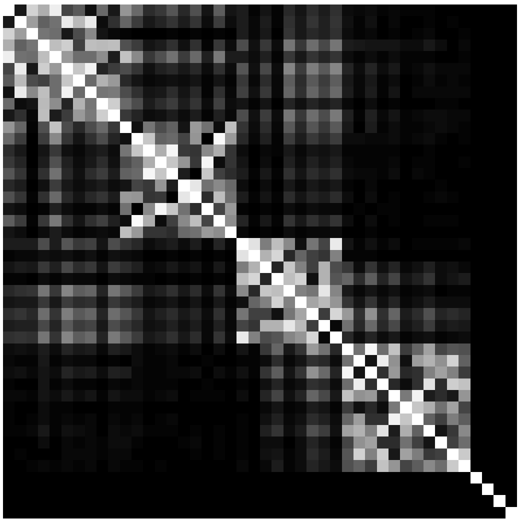


Figure 5.5: *The prior matrix of multiple linked parts with outliers. Lighter color indicates higher probability.*

Using our approach, independent motions are only a special case and are treated in the same fashion. The prior matrix for two independently moving articulated objects from a real sequence is shown in Fig. 5.7. Each of these two articulated objects has two parts. Notice the priors between every pair of trajectories from different objects are very small. RANSAC with priors is able to segment 4 motions from these trajectories.

5.5 Conclusions and Future Work

We describe and demonstrate a motion segmentation algorithm called RANSAC with priors. It can segment articulated motions as well as independent motions. It does not require prior knowledge of how many motions there are. It is both efficient and robust. The priors are derived from the spectral affinity between every pair of trajectories.

Future work will involve combining model selection to deal with degenerate shape and motion, as well as non-rigid ones. After having a sample set, we can estimate the model from several models. Furthermore, with priors, we may even consider forming a larger size of a sample set. This will not increase the computation too much as compared to common RANSAC WITHOUT priors because with the help of the priors the sample set has a far better chance consisting of data belonging to the same model. Applying RANSAC with priors to handle incomplete trajectories is also one important

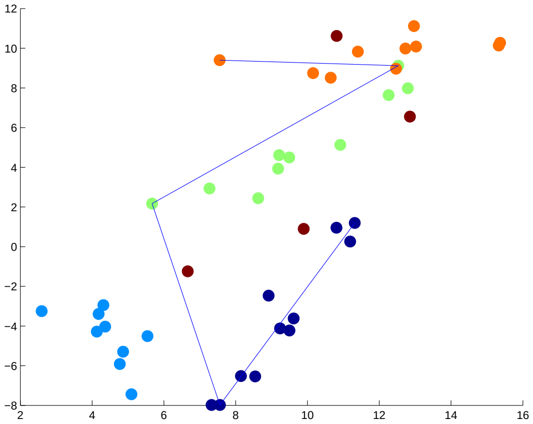


Figure 5.6: *The segmentation result of* 4 *linked parts using RANSAC with priors. Orange, green, light blue and dark blue indicate* 4 *parts of the articulated object.* 4 *Red dots are the remaining data rejected by the algorithm, which are the added outliers.*

future research area for motion segmentation.

We also plan to apply RANSAC with priors to highly challenging cases of complex articulated motions like human motions and complex scenes consisting of partially dependent and independent motions as well as non-rigid motions.

5.6 Comparisons Between Motion Segmentation Algorithms

At the end of this chapter, We compare our two motion segmentation methods described in this and the previous chapter with the other popular ones. Fig. 5.8 shows the comparisons in 4 aspects. There 4 aspects are important to consider to choose the right motion segmentation method for a specific problem. For example, if the motions are articulated, the method Shape Interaction Matrix method should be avoided. In addition, if the number of motions is large, our methods are the only choices. If we need to handle both rigid and non-rigid motions whose dimensions not known beforehand, the method of Local Subspace Affinity is the only choice.

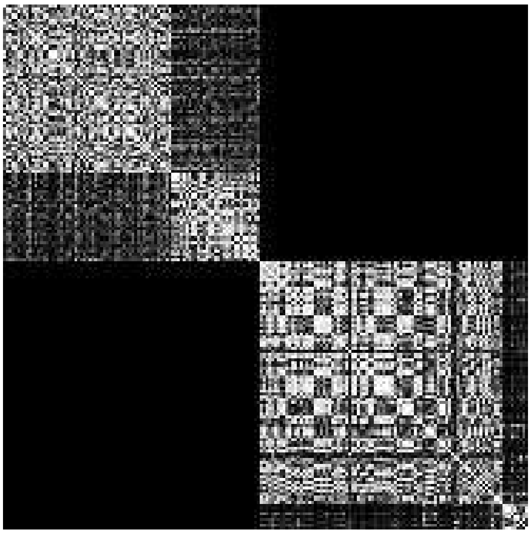
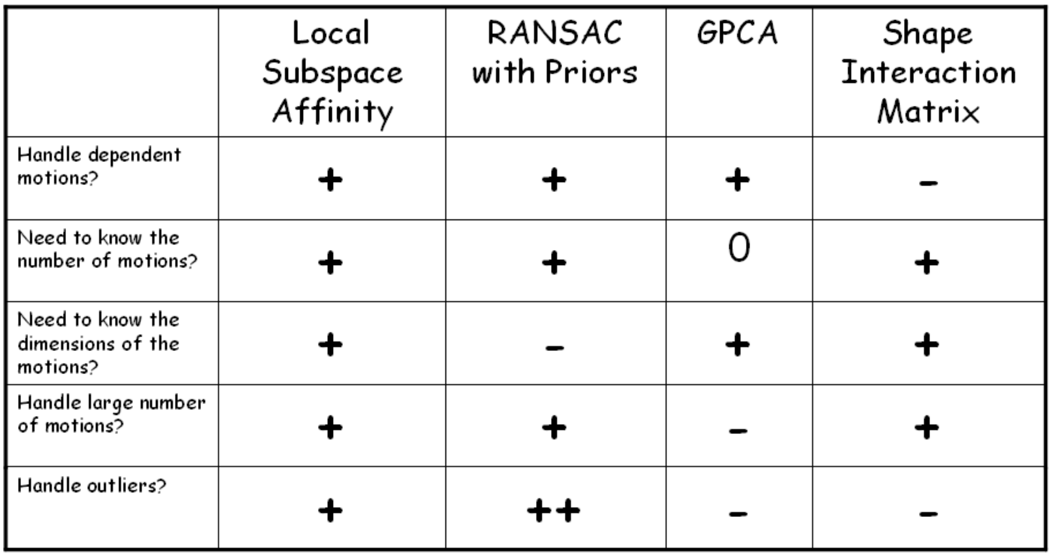


Figure 5.7: *The prior matrix of two independently moving articulated objects. One articulated object has two parts linked by an axis. The other object has two parts linked by a joint. Lighter color indicates higher probability.*



	Local Subspace Affinity	RANSAC with Priors	GPCA	Shape Interaction Matrix
Handle dependent motions?	+	+	+	-
Need to know the number of motions?	+	+	0	+
Need to know the dimensions of the motions?	+	-	+	+
Handle large number of motions?	+	+	-	+
Handle outliers?	+	++	-	-

Figure 5.8: *+: pros; ++: super pros; -: cons; 0: not relevant.*

Chapter 6

Learning the Kinematic Chain

We investigate the problem of learning the structure of an articulated object, i.e. its kinematic chain, from feature trajectories under affine projections. We demonstrate this possibility by proposing an algorithm which first segments the trajectories by local sampling and spectral clustering, then builds the kinematic chain as a minimum spanning tree of a graph constructed from the segmented motion subspaces. We test our method in challenging datasets and demonstrate the ability to automatically build the kinematic chain of an articulated object from feature trajectories. The algorithm also works if there are multiple articulated objects in the scene. Furthermore, we take into account non-rigid articulated parts that exist in human motions. We believe this advance will have impact on articulated object tracking and structure from motion. These results are first shown in our paper (Yan and Pollefeys, CVPR 2006).

6.1 Automatic Kinematic Chain Building From Feature Trajectories

In this section, we describe the algorithm of building the kinematic chain from feature trajectories of an articulated object under affine projection. It consists of two stages: at the first stage, trajectories are segmented according to the articulated parts and the motion subspaces are formed after rejecting outliers; at the second stage, the proximities between these motion subspaces is computed to build a proximity graph, then a minimum spanning tree algorithm is performed on the graph to retrieve the kinematic chain information of the articulated object.

We assume that the kinematic chain is noncyclic, which covers most cases of articulated objects. A cyclic kinematic chain is rare, e.g. four articulated parts that are joined end to end forming a cyclic articulated object. We will discuss how to adapt our algorithm to this special case at the end of this section.

6.1.1 Motion Segmentation

The trajectories of an articulated object are from different rigid or non-rigid parts which form different motion subspaces that may intersect one or the other. The motion segmentation stage is to segment them accordingly. We adopt the local subspace affinity approach proposed in Chapter 4.

After segmenting the trajectories, we perform outlier rejection within each segment. This is done using a RANSAC approach (Fischler and Bolles, 1981) that robustly fit the data into a subspace and reject outliers. The motion subspaces are formed by the remaining trajectories in each group.

6.1.2 Kinematic Chain Building

Given the motion subspaces, either rigid or non-rigid, their dependence on each other are measurable by their minimum principal angles between every pair of them.

For two linked parts, either rigid or non-rigid, either for a joint link or an axis link, their motion subspaces are intersecting on at least one dimensional subspace (See Section 3.4), thus have at least one zero principal angle. In practice, the value will not be exact zero so a threshold is required. For parts that are not linked, the motion subspaces do not have this property and have a larger minimum principal angle. Depending on how independent the motion subspaces are, the minimum principal angles may vary.

Based on the above analysis, we will describe our kinematic chain building algorithm in the following.

- Build the proximity graph
  We use a graph to represent the proximity between every pair of motion subspaces.

  $$G = (V, E)$$

  where $V = \{v_1, ..., v_S\}$ (v_i is the ith motion subspace; S is the number of motion subspaces) and $E(v_i, v_j) = \theta_{ij}$ (θ_{ij} is the minimum principal angle between subspace i and j).

- Find the minimum spanning tree(s)

Based on the proximity graph we find a minimum spanning tree using Algorithm 1. The spanning tree corresponds to the kinematic chain that we compute. Because this tree consists of edges of the smallest possible minimum principal angles between motion subspaces of the parts. If the parts are connected by articulated links, these edges should correspond to the links.

Algorithm 1 Finding single minimum spanning tree

```
Let $T$ be the graph of the smallest edge of $G$
while $T$ has fewer than $S - 1$ edges do
    find the smallest edge in $G$ connecting $T$ to $G - T$
    add it to $T$
end while
```

With a small modification, our algorithm can handle multiple articulated objects in the scene and find multiple kinematic chains (See Algorithm 2). The key is that whenever the smallest edge connecting T to G-T is over some threshold, we stop spanning the current tree and start building another spanning tree using the same procedure. The threshold chosen is not very sensitive. For example, in the experiments, a minimum principal angle between two linked parts is in the scale of 0.01 radians while that between parts of different objects is in the scale of 0.1 radians.

Algorithm 2 Finding multiple minimum spanning trees

```
Let $P = G$
while $P$ has more than one edge $<$ threshold do
    let $T$ be the minimum edge of $P$
    while the smallest edge connecting $T$ to $P - T <$ threshold do
        add it to $T$
    end while
    save $T$ as a kinematic chain
    let $P = P - T$
end while
```

As for cyclic articulated objects like 4 sticks connected as a loop by 4 joints, our algorithm needs to be adapted to find the cyclic kinematic chain. We use a new stopping criteria that we add new minimum edges to the graph until all remain edges are over the threshold. In this way, all links are found.

We also can use the second principal angles if several potential links are equally good for spanning the tree. This will be useful when, for example, the second best edge is almost as good as the best. Then we check their second principal angles to prefer choosing the link that has a smaller second principal angles. For the experiments, we kept it simple and did not use this strategy.

6.2 Experiments

We test our algorithm in two kinds of datasets, synthetic and real.

6.2.1 Synthetic tests

The first synthetic test demonstrates our method in a highly challenging case in which the kinematic chain is automatically built from a human model of 10 parts including the head, the body, two upper arms, two lower arms, two thighs and two legs. Each part has 20 trajectories. And the trajectories are perturbed by uniform noise equivalent to 1 pixel in a 600×400 image. The segmentation result is shown in Fig. 6.1 with bigger black dots showing the 10 misclassified points. Notice that the misclassification happens mostly around the joints and axes. After outlier rejection within each segment, the remaining 171 features are shown in Fig. 6.1.

The minimum principal angles (the proximity graph) between 10 motion subspaces are shown in Table 6.1. The bold font indicates the edges of the minimum spanning tree of the graph. The kinematic chain are built from that. The recovered kinematic chain is correct.

We show the second minimum principal angles between parts in Table 6.1 to see how the second dimension intersection between motions may be detected. The bold font shows the angles that are much lower than the average, which indicates a second dimensional subspace intersection between motions and thus indicates that there is an axis link (See Section **??**). The data shown in the table matches our synthetic model in which the upper arms are connected with the lower arms with axis links and so are the thighs with the legs.

In summary, in order to detect links for kinematic chain building, inspecting the minimum principal angle is enough. The second minimum principal angle is used to determine the type of the link.

The second synthetic test will demonstrate our algorithm applied to multiple articulated objects. We generate two synthetic human models in a scene. The motion segmentation and the outlier rejection steps proceed as described before. At the kine-

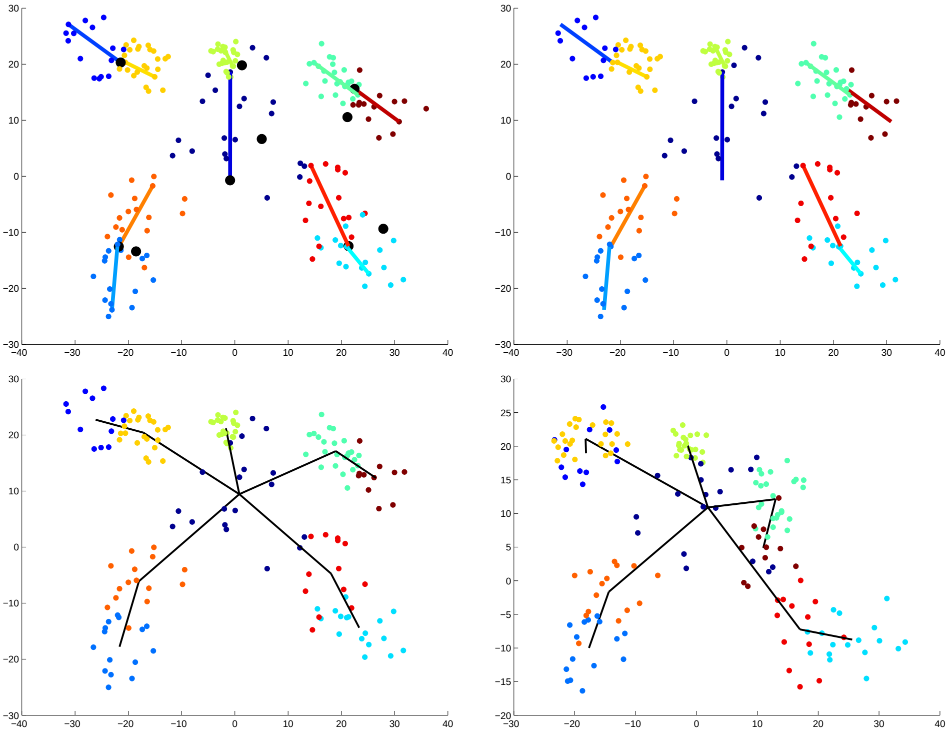


Figure 6.1: (*top left*) The segmentation of trajectories over 10 articulated parts of a synthetic human model. The sticks are shown for illustration purpose. (*top right*) Trajectories after rejecting outliers. (*bottom left and right*) The kinematic chain (black lines) built from trajectories.

Table 6.1: Synthetic human model: Proximity graph and its minimum spanning tree (upper triangle) and the second minimum principal angles between parts (lower triangle)

radians	body	head	luarm	llarm	ruarm	rlarm	lthign	lleg	rthign	rleg
body		**0.004**	**0.003**	0.184	**0.001**	0.189	**0.003**	0.136	**0.004**	0.127
head	0.306		0.321	0.396	0.317	0.439	0.277	0.278	0.245	0.245
luarm	0.632	0.746		**0.002**	0.609	0.619	0.215	0.230	0.135	0.152
llarm	0.786	0.859	**0.015**		0.606	0.620	0.264	0.271	0.187	0.219
ruarm	0.574	0.691	0.806	0.923		**0.002**	0.140	0.150	0.208	0.230
rlarm	0.724	0.754	0.856	0.931	**0.015**		0.187	0.192	0.251	0.265
lthign	0.257	0.476	0.839	0.934	0.685	0.751		**0.002**	0.210	0.220
lleg	0.259	0.482	0.872	0.942	0.700	0.757	**0.008**		0.219	0.233
rthign	0.248	0.430	0.707	0.856	0.762	0.805	0.342	0.345		**0.002**
rleg	0.249	0.470	0.786	0.876	0.770	0.815	0.355	0.355	**0.013**	

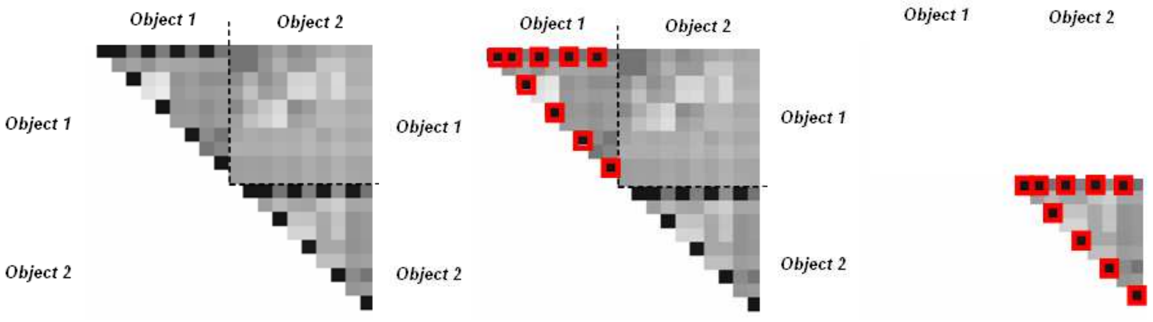


Figure 6.2: (*left*) The proximity graph: the minimum principal angles between the motion subspaces of two synthetic human models. (*middle*) The first minimum spanning tree illustrated by the red squares; the tree stops spanning at the threshold of 0.01 radians. (*right*) The second minimum spanning tree illustrated by the red squares.

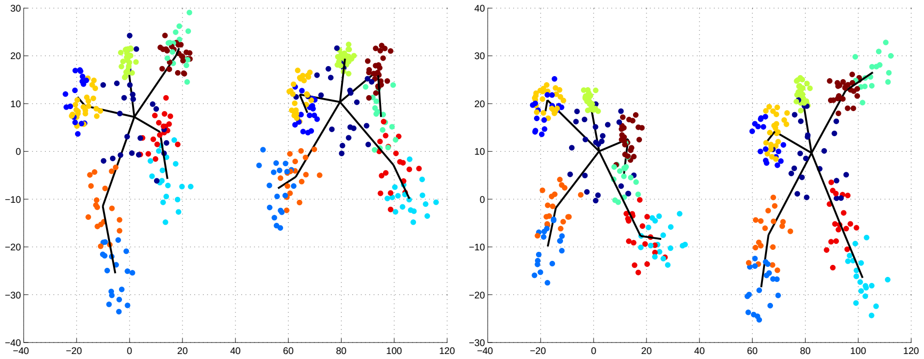


Figure 6.3: (*left and right*)Two kinematic chains (black lines) built from trajectories of a two-person scene.

matic chain building step, we use a threshold of 0.01 radians for determining whether the algorithm should stop adding new links to the current chain.

The proximity graph (the minimum principal angles) of the motion subspaces of the two different synthetic human models and the process of finding two minimum spanning trees from the graph are illustrated in Fig. 6.2.

The kinematic chains built from the trajectories are shown in Fig. 6.3.

The minimum principal angles between the motions of the two different synthetic human models is shown in Table 6.2. Obviously, the magnitude of these angles is far above the threshold. In fact, the threshold value for building multiple kinematic chains is not very sensitive.

Table 6.2: The minimum principal angles between motions of two synthetic human models

radians	body	head	luarm	llarm	ruarm	rlarm	lthign	lleg	rthign	rleg
body	0.158	0.200	0.205	0.214	0.210	0.214	0.207	0.211	0.205	0.216
head	0.161	0.470	0.488	0.493	0.490	0.485	0.372	0.434	0.324	0.421
luarm	0.165	0.372	0.464	0.489	0.437	0.433	0.300	0.358	0.283	0.404
llarm	0.237	0.382	0.461	0.485	0.429	0.425	0.339	0.389	0.312	0.402
ruarm	0.193	0.467	0.624	0.627	0.595	0.590	0.332	0.365	0.287	0.328
rlarm	0.269	0.478	0.626	0.631	0.602	0.597	0.362	0.389	0.327	0.354
lthign	0.195	0.192	0.196	0.204	0.208	0.212	0.194	0.202	0.196	0.216
lleg	0.279	0.276	0.284	0.289	0.293	0.300	0.280	0.277	0.278	0.282
rthign	0.200	0.197	0.201	0.209	0.208	0.213	0.201	0.205	0.203	0.223
rleg	0.220	0.220	0.220	0.226	0.227	0.239	0.220	0.228	0.235	0.255

Table 6.3: Proximity graph and its minimum spanning tree – Puppet

radians	larm	lleg	hip	rarm	body
rleg	0.0111	0.0007	**0.0002**	0.0126	0.0006
larm		0.0110	0.0060	0.0250	**0.0008**
lleg			**0.0002**	0.0170	0.0006
hip				0.0175	**0.0005**
rarm					**0.0003**

6.2.2 Real tests

The first real example is an articulated puppet with 6 rigid parts: the head, the upper body, the hip, 2 arms and 2 legs. A KLT tracker tracks a total of 114 features over 564 frames. The segmentation result is shown in Fig. 6.4. After outlier rejection within each segment, the remaining 97 features are shown in Fig. 6.4.

The minimum principal angles (the proximity graph) between 6 motion subspaces are shown in Table 6.3. The bold font indicates the edges of the minimum spanning tree of the graph. The kinematic chain are built from that and the links are recovered by intersecting linked subspaces (See Section 3.5) based on the kinematic chain (Fig. 6.4).

The second real example is an upper body motion of a person with 6 parts: the head, the upper body, 2 upper arms and 2 lower arms. The head has some non-rigid facial motion. A KLT tracker tracks the total of 268 features over 40 frames. The segmentation result is shown in the top left of Fig. 6.5. After outlier rejection within each segment, the remaining 97 features are shown in the top right of Fig. 6.5. The

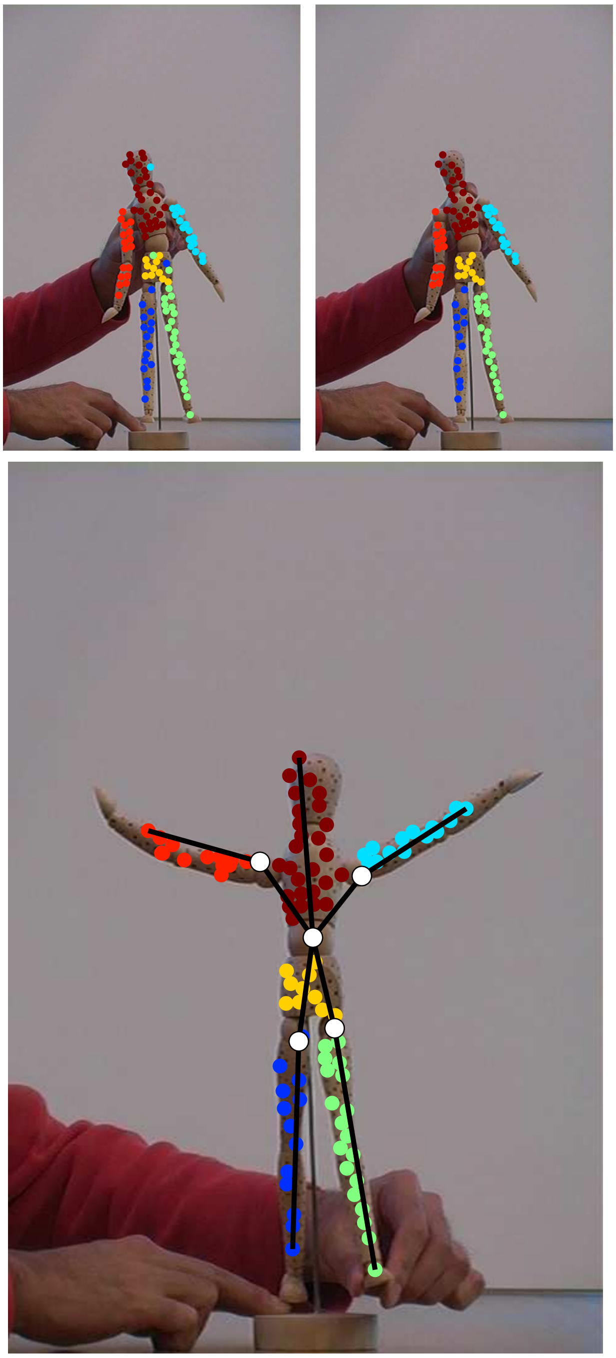


Figure 6.4: (*top left*) The segmentation of trajectories over 6 articulated parts of a puppet. (*top right*) Trajectories after rejecting outliers. (*bottom*) The kinematic chain built from trajectories and the links (white dots) recovered by intersecting the linked subspaces based on the kinematic chain.

Table 6.4: Proximity graph and its minimum spanning tree – Person

radians	luarm	ruarm	body	rlarm	llarm
head	0.0015	0.0033	**0.0011**	0.0035	0.0065
luarm		0.0036	**0.0008**	0.0058	**0.0009**
ruarm			**0.0008**	**0.0003**	0.0145
body				0.0018	0.0033
rlarm					0.0103

minimum principal angles (the proximity graph) between 6 motion subspaces are shown in Table 6.4. The bold font indicates the edges of the minimum spanning tree. The kinematic chain are built from the 6 motion subspaces and the links are recovered by intersecting subspaces based on the kinematic chain, shown in the bottom of Fig. 6.5.

The non-rigid part is the head which has a joint link with the upper body. The link is recovered simply by finding the 1-dimensional intersection between both motion subspaces as discussed in Section 3.4.

6.3 Conclusions and Future Work

We propose an algorithm that builds a kinematic chain from feature trajectories of an articulated object under affine projections. The algorithm is extended to handle scenes of multiple articulated objects and articulated non-rigid parts. It first segments trajectories according to the articulated parts; then it rejects outliers; in the end, it builds a kinematic chain from a minimum spanning tree of a proximity graph that is constructed from the minimum principal angles between the motion subspaces of the articulated parts.

We plan to handle occlusions and missing trajectories in the near future which will make our algorithm more practical. Ultimately, we aim to recover articulated human motion with non-rigid parts.

By learning the structure of an articulated object, constraints are automatically imposed on the motions of the object on the fly which may extend the existing tracking or shape from motion applications of articulated objects.

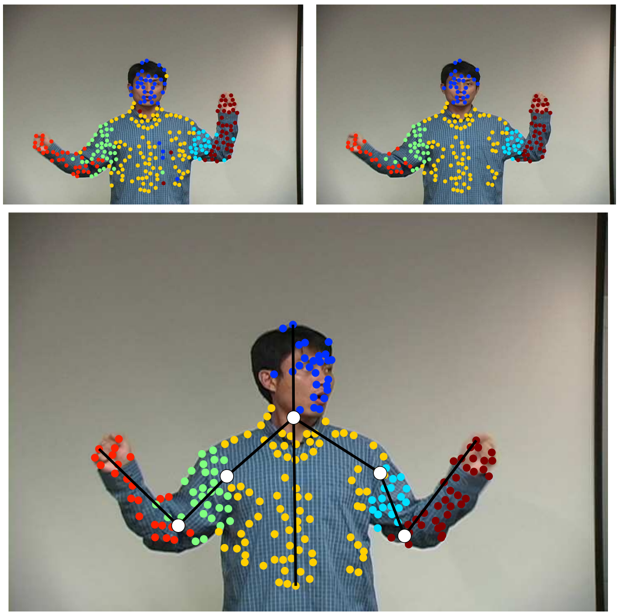


Figure 6.5: (*top left*) The segmentation of trajectories over 6 articulated parts of an upper-body human motion. (*top right*) Trajectories after rejecting outliers. (*bottom*) The kinematic chain built from trajectories and the links (white dots) recovered by intersecting the linked subspaces based on the kinematic chain.

Chapter 7

Articulated Motion Recovery with Non-rigid Parts

7.1 Recovering articulated motion, shape and kinematic chain from video

We summarize our approach to recover articulated motion with non-rigid parts from a single-view image sequence in this section. We assume that the shape and motion is general. Degenerate cases are briefly discussed in Section 8.1.

7.1.1 Feature Tracking

We use a KLT tracker to track features in image sequences and build the trajectory matrix.

With prior knowledge of the object, we derive and impose the rank constraint of the articulated motion directly to the trajectory matrix. Without that knowledge, we propose to detect an effective rank k (Section 4.1.6) of the trajectory matrix.

We reject outliers based on the rank constraint. We iteratively reject a column, i.e. one feature trajectory, that deviates from the constrained motion subspace most, and reimpose the rank constraint on the raw data of the remaining features until the largest deviation is below a certain threshold. The threshold can be chosen to be about twice the size of a feature patch for every frame. So if a feature patch size is 7 pixels, the threshold can be 14 pixels.

7.1.2 Motion Segmentation

The objective of this stage is to segment the trajectories according to the part (motion subspace) that they belong to. The global motion subspace of an articulated object with non-rigid parts is a mixture of intersecting subspaces.

If the number of parts and the dimensions of the motion subspaces are both unknown, we can use the algorithm that we propose in Chapter 4. If the dimensions of the motion subspaces are known, we can use an efficient and robust algorithm that we propose in Chapter 5.

After motion segmentation, we can further reject outliers based on each segmented motion subspace. We impose the rank constraint on a motion subspace and iteratively reject a trajectory that deviates from the constrained motion subspace most until the largest deviation is below a certain threshold. The threshold can be chosen to be twice the size of a feature patch for every frame as in Section 7.1.1.

7.1.3 Kinematic chain building and computing joints or axes

The segmented motion subspaces do not tell us about the structure of the articulated object. We use the algorithm in Chapter 6 to build the kinematic chain. Furthermore, we can compute the joints and axes of linked parts using the technique introduced in Section 3.5.

7.1.4 Shape and Motion Recovery

The segmented trajectories belong to different articulated parts, either rigid or non-rigid. The shape and motion of each part are recovered using algorithms such as (Tomasi and Kanade, 1992; Bregler et al., 2000) (See Section 2.1.1 and 2.1.2). Notice that, for shape and motion recovery, we limit our discussion to the orthographic case. For the shape and motion recovery from a more general camera models like affine cameras etc., please refer to (Hartley and Zisserman, 2002).

After recovering the shape and motion of each individual part of the articulated object, we need to recover 3D positions of the articulated joints and axes in reference to its connected parts before we can align all parts to recover the full articulated shape. In Section 3.5, we describe an algorithm to recover the image positions of the axis or joint *without* recovering the 3D shape of the object. Since we already recover each individual part here, we can use a simpler algorithm to recover those. Moreover, the

3D positions of the axis or joint are recovered at the same time.

Suppose W_1 and W_2 are the feature trajectories of two connected parts and $W_1 = (R_1|T_1)\begin{pmatrix} S_1 \\ 1 \end{pmatrix}$, $W_2 = (R_2|T_2)\begin{pmatrix} S_2 \\ 1 \end{pmatrix}$ are the recovered shape and motion matrices of both parts using the factorization method (See Section 2.1.1). Suppose both parts are connected by a joint. As described in Section 3.5, we can find the intersection subspace W_{12} of W_1 and W_2. It is a 1-dimensional subspace. By determining a scale factor α, αW_{12} is the trajectory of the joint in the image. Suppose $\begin{pmatrix} x & y & z & 1 \end{pmatrix}^T$ is the unknown 3D position of the joint. The following equation is used to recover both α, x, y and z. We simply need to left-multiply the Moore-Penrose pseudoinverse of $(R_1|T_1)$ on both sides.

$$\alpha W_{12} = (R_1|T_1)\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

We can use the motion matrix $(R_2|T_2)$ as well to recover the same joint. The recovered α, x, y and z are in reference to the 3D object coordinate system of the second part.

To recover an axis, we try to recover two points on the axis. The way to recover a point on the axis is very similar to recover the joint. As described in Section 3.5, we can find the intersection subspace W_{12} of W_1 and W_2. It is a 2-dimensional subspace. By determining two scale factor α and β which are related by $a\alpha + b\beta + 1 = 0$, $W_{12}\begin{pmatrix} \alpha & \beta \end{pmatrix}$ is the trajectory of any point on the axis. Without loss of generality, let $\alpha = 0$, then we can obtain a β value and the 3D position of a point on the axis by using the following equation in the same way as to recover a joint. Notice that $W_{12}\begin{pmatrix} 0 & \beta \end{pmatrix}^T$ is the trajectory of the point in the image.

$$W_{12}\begin{pmatrix} 0 \\ \beta \end{pmatrix} = (R_1|T_1)\begin{pmatrix} x_\beta \\ y_\beta \\ z_\beta \\ 1 \end{pmatrix}$$

Similarly, let $\beta = 0$, we can obtain a α value and the 3D location of anther point on the axis using the following equation.

$$W_{12}\begin{pmatrix}\alpha\\0\end{pmatrix}=(R_1|T_1)\begin{pmatrix}x_\alpha\\y_\alpha\\z_\alpha\\1\end{pmatrix}$$

Obviously, we can use the motion matrix $(R_2|T_2)$ in the above equations to recover the same points except for that the 3D positions are in reference to the 3D object coordinate system of the second part. Given the image and 3D positions of these two points, we recover the axis.

Now each individual part and its connected point or axis can be recovered independently in its 3D object coordinate system. The next task is to transform each part into a global coordinate system and align them correctly to recover the whole articulated object. One convenient candidate as the global coordinate system is the camera coordinate system because each part is only undetermined, i.e. unaligned, in its camera depth value. This is due to the orthographic camera model we use. We can not determine how far away an object is from the camera. However, the x and y values of all parts are determined already once transformed to the camera coordinate system. So we only need to align each part's the camera depth correctly. The links between the parts are the joints or axes. We need to align the camera depths of the same joints or axes of linked parts.

For a specific frame, suppose the motion and shape matrix of the part i are the following.

$$\begin{pmatrix}r_1^i & t_1^i\\ r_2^i & t_2^i\end{pmatrix}\begin{pmatrix}S_{obj}^i\\1\end{pmatrix}$$

where r_1^i and r_2^i are the two rows of the rotation matrix. t_1^i and t_2^i are the translation. The shape matrix can be transformed from the object coordinate system to the camera coordinate system as the following.

$$\begin{pmatrix}S_{cam}^i\\1\end{pmatrix}=\begin{pmatrix}r_1^i & t_1^i\\ r_2^i & t_2^i\\ r_3^i & t_z^i\\ 0 & 1\end{pmatrix}\begin{pmatrix}S_{obj}^i\\1\end{pmatrix}$$

where r_3^i is orthogonal to both r_1^i and r_2^i. t_z^i is the only variable, which affects the z coordinates of the transformed shape. t_z^i is subject to the following minimization.

Let

$$f^i(x) = \begin{pmatrix} r_1^i & t_1^i \\ r_2^i & t_2^i \\ r_3^i & x \\ 0 & 1 \end{pmatrix}$$

It represents the transformation matrix with the variable x. Let

$$\begin{pmatrix} J_{cam}^i \\ 1 \end{pmatrix} = f^i(t_z^i) \begin{pmatrix} J_{obj}^i \\ 1 \end{pmatrix}$$

or

$$\begin{pmatrix} A_{cam}^{\alpha i} \\ 1 \end{pmatrix} = f^i(t_z^i) \begin{pmatrix} A_{obj}^{\alpha i} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} A_{cam}^{\beta i} \\ 1 \end{pmatrix} = f^i(t_z^i) \begin{pmatrix} A_{obj}^{\beta i} \\ 1 \end{pmatrix}$$

$$min_{t_z^i} \|t_z^1\| + \sum \|J_{cam}^m - J_{cam}^n\| + \sum(\|A_{cam}^{\alpha p} - A_{cam}^{\alpha q}\| + \|A_{cam}^{\beta p} - A_{cam}^{\beta q}\|)$$

Adding $\|t_z^1\|$ to the minimization means $t_z^1 = 0$. We need to fix the first part's depth. The other parts are going to be aligned according to it through the links. $\sum \|J_{cam}^m - J_{cam}^n\|$ aligns all of the same joint in linked parts m and n. $\sum(\|A_{cam}^{\alpha p} - A_{cam}^{\alpha q}\| + \|A_{cam}^{\beta p} - A_{cam}^{\beta q}\|)$ aligns all of the two points of the same axis in linked parts p and q.

Once we have all t_z^i values, we have the transformation matrix to transform each part from its object coordinate system to the camera coordinate system with the correct camera depths. This completes the recovery of the shape and motion of the whole articulated object.

7.2 Experiments

7.2.1 Synthetic Experiment

In the synthetic test we reconstruct the human model of 10 parts used in Section 6.2.1. The factorization method is applied to the 10 parts and the shapes are reconstructed. We compare the reconstruction errors against the added noise levels in Fig. 7.1. The

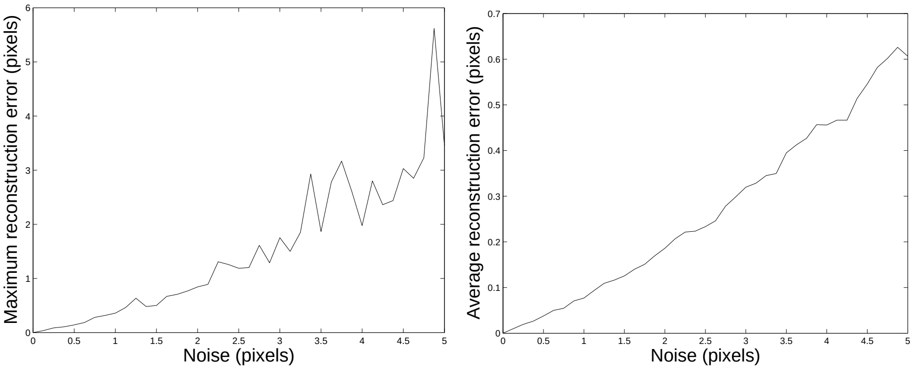


Figure 7.1: (*left*) Noise level vs. Maximum Reconstruction Error (*right*) Noise level vs. Average Reconstruction Error

reconstruction errors are measured by the average deviations of the reconstructed shape from the original shape.

7.2.2 Real Experiment

In the first experiment, we reconstruct the toy truck based on the segmentation result in Section 4.2.

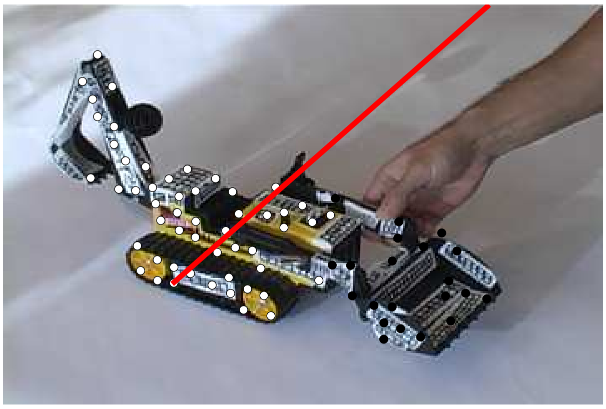


Figure 7.2: An axis is identified by intersecting the motion subspaces of the articulated parts.

An axis is identified by intersecting the motion subspaces of the articulated parts. Its image positions are recovered using our algorithm (Fig. 7.2).

Each articulated part is recovered as a rigid shape using the factorization method. By putting all parts into the camera coordinates, shape and motion of the articulated

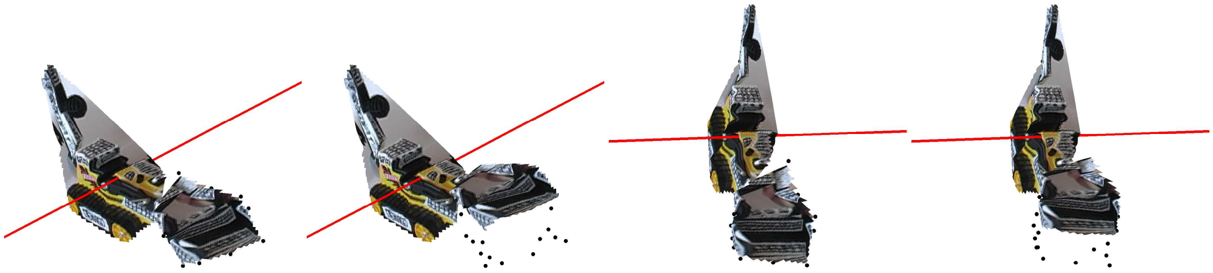


Figure 7.3: The shape and motion of the truck get recovered and reanimated. The black dots show the original position of the shovel. Not only novel views but also novel motions are generated by rotating the shovel around the axis.

object gets recovered. Furthermore, with the axis recovered in the camera coordinates, we can reanimate the articulated motion by rotating a part around the axis and generate not only novel views but also novel motions (Fig. 7.3).

In the second real experiment, we use the puppet data. Based on the motion segmentation and kinematic chain building results, the factorization method is applied to the 6 parts and the shapes are reconstructed. For demonstration purpose, the recovered articulated motion is used to animate a synthetic model. The result is shown in Fig. 7.4.

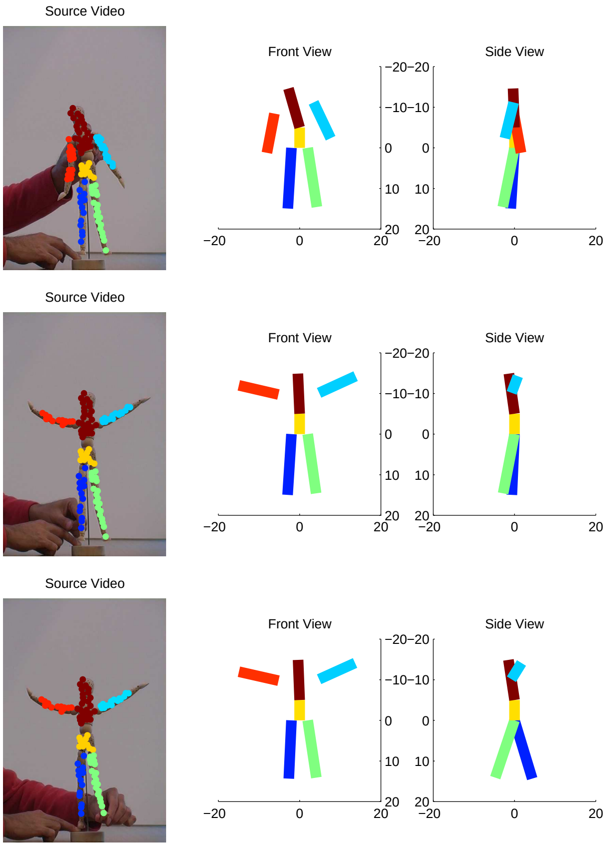


Figure 7.4: The front and side views of the reconstruction of the puppet of different frames. The recovered articulated motion is used to animate a synthetic human model.

Chapter 8

Final Conclusions and Future work

8.1 Conclusions

We describe an approach to analyze and recover articulated motion with non-rigid parts from video. We model the articulated motion with non-rigid parts using a set of intersecting subspaces. The motion of a rigid part or a non-rigid part is uniformly modeled with a motion subspace. The linked parts have 1 or 2 dimensional subspace intersections depending on the link type, a joint or an axis. By local sampling and spectral clustering, the motion subspaces are segmented regardless of their dimensionality and the dependencies between them using our method. This is especially important for articulated motion with unknown rigid or non-rigid parts. The kinematic chain is automatically built by examining the intersections between every two motion subspaces and using a minimum spanning tree to retrieve. And the shape of each part is recovered by the factorization-based methods for rigid or non-rigid shapes after the motion segmentation as well.

8.2 Future Work

We identify the following areas that can improve our approach in the future.

8.2.1 Feature tracking and direct method

Our approach is based on feature trajectories. They are obtained from a KLT tracker described in (Shi and Tomasi, 1994). It is a point-based tracker that detects point features and tracks its 2D position in images across frames. A point feature is a small image patch, 5 by 5 pixels for example, that has high gradients in both x and y direction. A strong point feature, i.e. with high gradients, guarantees a reliable tracking.

For a complex articulated object, the resolution of the imaging device and its dynamic range have direct effect on the result of our approach. Generally the higher the resolution and the dynamic range, the better. The number of point features that are tracked also affect the result. In some cases, artificial markers may be necessary to provide these point features.

One alternative is to derive the image motion using the direct method. It poses the tracking problem as a maximum likelihood problem, to be optimized with respect to the entire image sequence(Irani, 2002). Besides, it has even been shown that non-rigid shape recovery problem can be posed as a maximum likelihood problem too under this framework, to be optimized with respect to the entire image sequence(Brand and Bhotika, 2001; Brand, 2001; Torresani et al., 2001; Torresani and Hertzmann, 2004).

8.2.2 Missing Data and Reappearing Features

Due to self occlusion and illumination change, point features may be lost in some frames and reappear in later frames. This results in two new challenges for shape and motion recovery. The first one is how to handle trajectories that have missing data; the second, how to handle reappearing features.

Different techniques of handling missing data in shape from motion problems have been proposed, e.g. (Vidal and Hartley, 2004; Gruber and Weiss, 2004). They may be applied in the articulated shape and motion recovery. Power factorization (Vidal and Hartley, 2004) can project incomplete data onto low dimensional subspaces, complementing SVD which can only work on complete data. It is useful for estimation of the missing data of individual part of the articulated object. Estimation-Maximization based technique(Gruber and Weiss, 2004) can be used to deal with the missing data in articulated shape and motion recovery. There are other works that handle incomplete trajectories other than rigid shapes such as (Bartoli and Olsen, 2005).

It will be ideal for new trajectories of reappearing features to be merged with previous trajectories instead of treating them as new trajectories in order to obtain a complete 3D model. To our knowledge, not much work has been done dealing with this problem except (Goncalves and Aguiar, 2004). It merges the trajectories by seeking a simplest rigid object that explains the trajectories.. We may use similar techniques for individual parts of the articulated object. How to merge trajectories not at the level of the individual parts but at the level of the global articulated object is a challenging problem. Not in the factorization context, (Koch et al., 1999) has an approach which explicitly tried to pick up lost tracks before instantiating new ones.

8.2.3 Degenerate Shapes

We have discussed our approach assuming that the shapes are general. For degenerate shapes, our motion segmentation algorithms are general enough to deal with them because it does not rely on the knowledge of the dimensionality of motion subspaces. The kinematic chain building is based on the dependence between motion subspaces of linked parts so it works with degenerate shapes as well. The shape recovery of degenerate shapes requires some model selection algorithm, e.g. the rank detection algorithm, to detect the degeneracy so a special treatment is given for the degenerate case. However, a more robust model selection algorithm is highly desirable.

8.2.4 Projective Reconstruction

Our approach assumes an affine projection model and the shape recovery is affine. It will be interesting to extend our work to projective reconstruction and to study the properties of the articulated motions under perspective projection.

Another possibility is to use the affine reconstruction of our approach as an initialization for perspective reconstruction.

A factorization method for projective reconstruction is usually done in the following way.

Suppose a point in 3D is represented as $\mathbf{X}$; the projection matrix, $\mathbf{P}$. The homogeneous 2D coordinates are the result of the projection.

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{PX}$$

We have the trajectories of feature points and write them out in the homogeneous 2D coordinates. We have the following factorization form.

$$\mathbf{W_s} = \begin{pmatrix} \lambda_{11}\begin{pmatrix} u_{11} \\ v_{11} \\ 1 \end{pmatrix} & ... & \lambda_{1P}\begin{pmatrix} u_{1P} \\ v_{1P} \\ 1 \end{pmatrix} \\ ... & ... & ... \\ \lambda_{F1}\begin{pmatrix} u_{F1} \\ v_{F1} \\ 1 \end{pmatrix} & ... & \lambda_{FP}\begin{pmatrix} u_{FP} \\ v_{FP} \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \mathbf{P_1} \\ ... \\ \mathbf{P_F} \end{pmatrix} \begin{pmatrix} \mathbf{X_1} & ... & \mathbf{X_P} \\ 1 & ... & 1 \end{pmatrix}$$

$\mathbf{W_s}$ is called the scaled trajectory matrix.

We do not know λ_{ij}, the so-called projective depth. So we can not factorize $\mathbf{W_s}$. We can estimate λ_{ij} iteratively.

- Set $\lambda_{ij} = 1$
- Factorize $\mathbf{W_s}$ using SVD upto rank 4; get the motion and shape matrix
- Update $\lambda_{ij} = \mathbf{P_i}^{(3)}\mathbf{X_j}$. $\mathbf{P_i}^{(3)}$ is the third row of $\mathbf{P_i}$
- If λ_{ij}'s are the same, stop; otherwise, go to step 2

(Triggs, 1996) points out that the iterative approach above is very stable even with arbitrary initialization λ_{ij}. Once λ_{ij}'s are estimated, a rank-4 factorization of $\mathbf{W_s}$ generates the motion and shape matrix upto a projective transformation. Given more information, e.g. the camera calibration, further reconstructions, e.g. Euclidean reconstruction, can be recovered from the projective reconstruction.

8.2.5 Applications

One promising application for this approach is articulated motion capture. Motion capture is widely used in areas like animation, medical study and sports analysis etc. This approach can be used to analyze motion capture data. It can segment the articulated object's motion into different parts, detect and recover the articulated object's joints and axes, recover the object's 3D motion. It also can be applied to the full human motion capture data including non-rigid facial motion.

Secondly, current motion capture methods use all kinds of technologies including optical or magnetic markers to solve the feature tracking problems. Due to the motion subspace restrictions on the feature trajectory data of an articulated object, it is possible to obtain a high quality of feature tracking results through a single camera. If the missing data and reappearing features are better solved, this approach has the potential to simplify the current motion capture methods by using only a single camera to acquire the image sequence and obtain the motion data from it.

Appendix

Let $W = (W_1|W_2)$, $W_1 = (\mathbf{p_1} \cdots \mathbf{p_{N_1}})$, $W_2 = (\mathbf{p_{N_1+1}} \cdots \mathbf{p_N})$ where $\Re(W_1) = r_1$, $\Re(W_2) = r_2$ and $\Re(W) = r_1 + r_2 - t$, i.e. the subspaces of W_1 and W_2 intersect and the rank of the intersection is t.

Let $W = UDV^T$ up to rank $(r_1 + r_2 - t)$ using SVD. Let $V^T = (V_1^T|V_2^T) = (\mathbf{v_1^T}, ...\mathbf{v_{N_1}^T}|\mathbf{v_{N_1+1}^T}, ..., \mathbf{v_N^T})$. We are going to prove that except for the intersection subspace of r dimensions the subspaces consisting of the other dimensions of V_1^T and V_2^T are orthogonal, i.e. if we remove the dimensions and get $\widehat{V_1}^T$ and $\widehat{V_2}^T$, $\widehat{V_1}\widehat{V_2}^T = \mathbf{0}$.

We will prove it by contradiction. Assume that the subspaces of $\widehat{V_1}^T$ and $\widehat{V_2}^T$ are not orthogonal. There exists a subspace intersection of dimension k $(k > 1)$. So the subspaces of V_1^T and V_2^T have an intersection of dimension $(r + k)$. so do the subspaces of W_1 and W_2 because $W_1 = UDV_1^T$ and $W_2 = UDV_2^T$. This conclusion contradicts with that the intersection of the subspaces of W_1 and W_2 is of dimension r. $\square$

Bibliography

A. Bartoli, S. I. Olsen, “A batch Algorithm for Implicit Non-Rigid Shape and Motion Recovery”, *Workshop on Dynamical Models for Computer Vision*, ICCV, Bejing China, 2005.

T. Boult and L. Brown,“Factorization-based segmentation of motions”, *IEEE Workshop on Visual Motion*, 1991

M. Brand, “Morphable 3D models from video”, *CVPR*, pp. II:456-463, 2001.

M. Brand, R. Bhotika,“Flexible Flow for 3D Nonrigid Tracking and Shape Recovery”, *CVPR*, ISSN: 1063-6919, Vol. 1, pp. 315-322, December 2001

M. Brand, “A Direct Method for 3D Factorization of Nonrigid Motion Observed in 2D”, *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2005 (CVPR 2005)

C. Bregler, A. Hertzmann, H. Biermann, “Recovering Non-Rigid 3D Shape from Image Streams”, *CVPR*, June 2000.

C. Bregler, J. Malik, “Tracking people with twists and exponential maps”, *CVPR*, pp. 8–15, 1998.

J.P. Costeira, T. Kanade, ”A Multibody Factorization Method for Independently Moving Objects”, *IJCV*, Vol. 29, Issue 3 pp. 159-179, 1998.

G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari,“A clustering technique for the identification of piecewise affine and hybrid systems”,*Automatica*, 39:205–217, 2003.

M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”, *Comm. of the ACM*, Vol 24, pp 381-395, 1981.

C.W. Gear, “Feature grouping in moving objects”, *Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, Texas, 1994

G. Golub and A. van Loan, *Matrix Computations*, Johns Hopkins U. Press, 1996

B.B. Goncalves, P.M.Q. Aguiar, "Complete 3-D models from video: a global approach", *ICIP*, 2004

A. Gruber, Y. Weiss, "Multibody factorization with uncertainty and missing data using the EM algorithm", *CVPR*, 2004

M. Han and T. Kanade, "Reconstruction of a Scene with Multiple Linearly Moving Objects", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, June, 2000

R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge Press, 2002

D. Hogg. "Model-based vision: A program to see a walking person", *Image and Vision Computing*, 1(1):5–20, 1983.

R. J. Holt, T. S. Huang, A. N. Netravali, and R. J. Qian, "Determining articulated motion from perspective views: A decomposition approach", *Pattern Recognition*, 30:14351449, 1997.

N. Ichimura, "Motion segmentation based on factorization method and discriminant criterion", *ICCV*, pages 600605, 1999.

M. Irani, "Multi-Frame Correspondence Estimation Using Subspace Constraints", *Int. J. of Comp. Vision* 48 (2002) 173-194

K. Kanatani, "Motion segmentation by subspace separation and model selection: model selection and reliability evaluation", *Intl. J. of Image and Graphics*, 2(2):179197, 2002.

R. Koch, M. Pollefeys, B. Heigl, L. Van Gool and H. Niemann, "Calibration of Hand-held Camera Sequences for Plenoptic Modeling", *Proc. International Conference on Computer Vision*, pp.585-591, Corfu (Greece), 1999.

P. Meer, D. Mintz, A. Rosenfeld and D. Y. Kim, "Robust regression methods for computer vision: A review", *International Journal of Computer Vision*, Vol. 6, Num. 1, April 1991.

A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm", *Advances in Neural Information Processing Systems* 14. MIT Press, 2002.

P. Perona and W.T. Freeman, “A Factorization Approach to Grouping”, *ECCV*, 1998

C. Poelman and T. Kanade, “A Paraperspective Factorization Method for Shape and Motion Recovery”, *tech. report CMU-CS-92-208*, Computer Science Department, Carnegie Mellon University, October, 1992.

M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, “Visual modeling with a hand-held camera”, International Journal of Computer Vision 59(3), 207-232, 2004.

G.L. Scoot and H.C. Longuet-Higgins, “Feature grouping by relocalisation of eigenvectors of the proximity matrix”, *Proc. British Machine Vision Conference*, 1990

J. Shi and J. Malik, “Normalized Cuts and Image Segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

J. Shi and C. Tomasi, “Good Features to Track”, *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593-600, 1994

D. Sinclair, L. Paletta and A. Pinz, “Euclidean Structure Recovery through Articulated Motion”, *Proc. 10th Scandinavian Conference on Image Analysis*, Lappeenranta, Finland, 1997, pp. 991-998.

C. Sminchisescu and B. Triggs, “Covariance Scaled Sampling for Monocular 3D Body Tracking”, Proc. CVPR, vol. I, pp. 447-454, 2001.

C.J. Taylor, “Reconstruction of Articulated Objects from Point Correspondences in a Single Image”, *CVPR*, pp. 677, 2000.

C. Tomasi, T. Kanade, “Shape and motion from image streams under orthography: a factorization method”, *IJCV*, Vol. 9, Issue 2 pp. 137-154, 1992.

P.H.S. Torr, “Bayesian Model Estimation and Selection for Epipolar Geometry and Generic Manifold Fitting”, *IJCV*, 50(1), 35-61, 2002.

P.H.S. Torr, A. Zisserman, S.J. Maybank, “Robust detection of degenerate configurations for the fundamental matrix”, *ICCV*, 1995.

L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler, “Tracking and modeling non-rigid objects with rank constraints”, *CVPR*, pages 493–500, 2001.

L. Torresani, A. Hertzmann, "Automatic Non-Rigid 3D Modeling from Video", *ECCV* 2004

P. Tresadern and I. Reid, "Articulated Structure From Motion by Factorization", *Proc IEEE Conf on Computer Vision and Pattern Recognition*, 2005

B. Triggs, "Factorization methods for projective structure and motion", *CVPR*, 1996.

R. Vidal and R. Hartley, "Motion Segmentation with Missing Data using PowerFactorization and GPCA", *IEEE Conference on Computer Vision and Pattern Recognition*, 2004

R. Vidal, Y. Ma and S. Sastry, "Generalized Principal Component Analysis (GPCA) ", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, June 2003.

R. Vidal, Y. Ma and J. Piazzi, "A New GPCA Algorithm for Clustering Subspaces by Fitting, Differentiating and Dividing Polynomials", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, June 27 - July 02, 2004.

Y. Weiss, "Segmentation using eigenvectors: A unifying view",*ICCV*, pages 975982, Corfu, Greece, September 1999.

J. Xiao, J. Chai, and T. Kanade, "A closed-form solution to non-rigid shape and motion recovery", *Proceedings of the European Conference on Computer Vision*, 2004.

J. Yan, M. Pollefeys, "A Factorization-based Approach to Articulated Motion Recovery", *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005

J. Yan, M. Pollefeys, "A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate", *ECCV* 2006

J. Yan, M. Pollefeys, "Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects", *CVPR'06 (IEEE Conf. on Computer Vision and Pattern Recognition*, New York City, 2006

J. Yan, M. Pollefeys, "Articulated Motion Segmentation Using RANSAC With Priors", *Workshop on Dynamical Vision 2005 (in conjunction with ICCV'05)*

J. Yan, M. Pollefeys, “Recovering Articulated Non-rigid Shapes,Motions and Kinematic Chains From Video”, *TPAMI (IEEE Transactions on Pattern Analysis and Machine Intelligence)* Volumne 30, Issue 5, May 2008

L. Zelnik-Manor and M. Irani, “Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations”, *IEEE Computer Vision and Pattern Recognition*, 2003.

H. Zhou, T. Huang, “Recovering Articulated Motion with a Hierarchical Factorization Method,” *Gesture Workshop* 2003: 140-151