THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCE

SPARSE MOTION ANALYSIS

By

LIANGJING DING

A Dissertation submitted to the
Department of Scientific Computing
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2013

Liangjing Ding defended this dissertation on June 13, 2013.

The members of the supervisory committee were:

Adrian Barbu
Professor Directing Thesis

Anke Meyer-Baese
Co-Professor Directing Thesis

Xiuwen Liu
University Representative

Dennis Slice
Committee Member

Xiaoqiang Wang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with the university requirements.

To my wife, Tingting

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my advisor, Prof. Adrian Barbu. It is my real fortune and pleasure to be a student of Prof. Barbu, for his thoughtful guidance and inspiring advice. I have enjoyed every minute in research. I would also like to thank my co-advisor Prof. Anke Meyer-Baese for her consistent support and warm encouragement.

I am also deeply indebted and grateful to my committee members, Prof. Xiuwen Liu, Prof. Dennis Slice and Prof. Xiaoqiang Wang. I learned a great amount of knowledge from your courses and seminars.

I would like to thank Dana Lutton, Xiaoguang Li, Jim Wilgenbusch and Kyle Gower-Winter for their generous help when I worked as a graduate assistant. Finally, I deeply appreciate the consistent financial support from Department of Scientific Computing for my Ph.D study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Motion segmentation is an essential pre-processing task in many computer vision problems. In this dissertation, the motion segmentation problem is studied and analyzed. At first, we establish a framework for the accurate evaluation of the motion field produced by different algorithms. Based on the framework, we introduce a feature tracking algorithm based on RankBoost which automatically prunes bad trajectories. The algorithm is observed to outperform many feature trackers using different measures. Second, we develop three different motion segmentation algorithms. The first algorithm is based on spectral clustering. The affinity matrix is built from the angular information between different trajectories. We also propose a metric to select the best dimension of the lower dimensional space onto which the trajectories are projected. The second algorithm is based on learning. Using training examples, it obtains a ranking function to evaluate and compare a number of motion segmentations generated by different algorithms and pick the best one. The third algorithm is based on energy minimization using the Swendsen-Wang cut algorithm and the simulated annealing. It has a time complexity of $O(N^2)$, comparing to at least $O(N^3)$ for the spectral clustering based algorithms; also it could take generic forms of energy functions. We evaluate all three algorithms as well as several other state-of-the-art methods on a standard benchmark and show competitive performance.

CHAPTER 1

INTRODUCTION

This chapter is a brief overview of the sparse motion segmentation problems. In section 1.1, we sketch the idea of sparse motion segmentation and present the main challenges. In section 1.2 and 1.3, we introduce some preliminary knowledge, such as concepts and algorithms of feature detection and optical flow that form the basis of generation of trajectories, a.k.a. tracked feature points. In section 1.4, we present the mathematical models of motion segmentation. In section 1.5, we review the spectral clustering, which has been widely used in the sparse motion segmentation literature recently. In section 1.6, we briefly talk about several datasets that are used in this dissertation. Finally, the outline of the dissertation is given in in section 1.7.

1.1 Motion Segmentation and its Main Challenges

The goal of motion segmentation is to separate a sequence of images into different regions, each of which should correspond to a distinct motion. It is an important fundamental step for many applications in computer vision, such as robotics, video surveillance, action recognition, tracking, traffic monitoring, etc. There are two kinds of motion segmentation, sparse motion segmentation and dense motion segmentation. In the sparse motion segmentation, the moving objects are represented by a limited number of feature points, whereas dense motion segmentation compute pixel-wise motion. In this dissertation, we will only talk about sparse motion segmentation.

There are several issues that need to be resolved to make motion segmentation algorithms applicable in practice. One of the main problems is the presence of noise. In some

scenarios, the noise level can become critical, for example, underwater imaging [11, 67]. Blurring is also a common issue when motion is involved. Another common problem is occlusion, or even worse, the disappearance and reappearance of objects in the scene. Furthermore, one does not always have knowledge about the objects or the number of motions in the scene. The main challenges of motion segmentation could be summarized as follows.

- Occlusion: is the algorithm able to deal with pixels that appear or disappear between frames?
- Sequentiality: is the algorithm able to handle the motion that is not present at some frames in a video sequence?
- Robustness: does the algorithm work well when the noise level is relatively large or there are many outliers existing?
- Number of motions: is the algorithm able to detect the number of motions in the scene dynamically?
- Non-rigid object: is the algorithm able to deal with motions of non-rigid objects?
- Dependent motions: is the algorithm able to deal with motions that are not totally independent?

Furthermore, for a generic motion algorithm that is supposed to work in different unpredictable situations, the requirement of prior knowledge or training might be considered as a drawback.

In a word, motion segmentation is a crucial but difficult task in computer vision. Many researchers have focused on this problem, but the performance of most algorithms is still far behind human perception. More effort is still needed to completely solve the problem.

1.2 Feature Detection

Detection of features is the prerequisite of feature tracking algorithms. It plays a big role in the process of tracking [54]. A local feature is an image pattern which differs from its neighborhood. It is usually associated with some change of image properties, such as intensity, color, and texture. The exact definition of features often depends on the type of application. The resulting features could be in the form of isolated points, continuous curves, or connected regions.

Feature detection is usually performed as an starting point for many computer vision algorithms. The desirable property for a good feature detector is Repeatability. Given a object or a scene, people could obtain many images by changing the viewing conditions, repeatability requires that a high percentage of the features detected are identical on all images. Moreover, the feature detection methods should be insensitive to some small deformations, such as image noise, discretization effects, compression artifacts, blur, etc. In other words, the accuracy of the detection should not drastically decrease due to such deformations. Repeatability is the most important property of feature detection.

Efficiency is also important in feature detection. Preferably, the detection of features in an image should be applied for time-critical applications. Additionally, the number of features detected in a image should not be too few.

Commonly used features are edges, corners, blobs, ridges, etc. but for tracking, one main approach is to find suitable features in the first image, and then track those features in the other frames of the sequence by a local search technique. In the process, people usually use corner features. So in the following, we only introduce corner feature detection algorithms.

1.2.1 Corner Detector

The corner points are point-like features with high curvature in the 2D image. The terms 'corners' and 'interest points' are used interchangeably in literature. They don't necessarily correspond to projections of 3D corners. A corner can be defined as the intersection of two edges. A frequently used corner detection algorithm is proposed by Harris and Stephens [30].

Intuitively, image patches with large contrast change (gradients) are easier to localize than textureless ones, especially those with gradients in at least two different orientations. The intuition can be formalized by looking at the simplest possible matching criterion for comparing two image patches, i.e., their (weighted) summed square difference,

$$E_{\mathrm{WSSD}}(u) = \sum_i w(x_i)[I_1(x_i + u) - I_0(x_i)]^2,$$

where I_0 and I_1 are the two images being compared, $u = (u, v)$ is the displacement vector, $w(x)$ is a spatially varying weighting (or window) function, and the summation i is over all the pixels in the patch.

When performing feature detection, for the purpose of stability, we can compute the metric with respect to small variations in position Δu by comparing an image patch against itself, which is known as an auto-correlation function or surface

$$E_{\mathrm{AC}}(\Delta u) = \sum_i w(x_i)[I_1(x_i + \Delta u) - I_0(x_i)]^2.$$

Using a Taylor Series expansion of the image function $I_0(x_i+\Delta u) \approx I_0(x_i)+\nabla I_0(x_i)\cdot\Delta u$, we can approximate the auto-correlation surface as

$$\begin{aligned} E_{\mathrm{AC}}(\Delta u) &= \sum_i w(x_i)[I_1(x_i + \Delta u) - I_0(x_i)]^2 \\ &\approx \sum_i w(x_i)[I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u - I_0(x_i)]^2 \\ &= \sum_i w(x_i)[\nabla I_0(x_i) \cdot \Delta u]^2 \\ &= \Delta u^T A \Delta u, \end{aligned}$$

where

$$\nabla I_0(x_i) = (\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y})(x_i)$$

is the image gradient at x_i. This gradient can be computed in various ways. Early works usually use a fixed filter due to the limited computational resource, for example, the Harris detector uses a [-2 -1 0 1 2] filter, but modern variants convolve the image with horizontal and vertical derivatives of a Gaussian.

The auto-correlation matrix A can be written as

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where we have replaced the weighted summations with discrete convolutions with the weighting kernel w. If we use the gaussian kernel, the auto-correlation matrix is

$$A = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x(\sigma_D) I_y(\sigma_D) \\ I_x(\sigma_D) I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

with

$$\begin{aligned} I_x(\sigma_D) &= \frac{\partial}{\partial x} g(\sigma_D) * I \\ g(\sigma) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned}$$

where the σ_D and the σ_I is the *differentiation scale* and the *integration scale*, respectively. The eigenvalues of A represent the principal signal changes in two orthogonal directions in a neighborhood around the point, and corners can be found as locations in the image for which the image signal varies significantly in both directions, which means both eigenvalues are large.

(a) The example image

(b) Harris features

(c) Shi-Tomasi features

Figure 1.1: Harris and Shi-Tomasi corner features generated for the example image, represented with green circles.

Based on this property, Harris detector uses the following measure for *cornerness*,

$$\det(A) - \alpha \operatorname{trace}(A)^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

with $\det(A)$ the determinant and $\operatorname{trace}(A)$ the trace of the matrix A. A typical value for α is 0.06. The measure does not require to actually compute the eigenvalue decomposition of

the matrix A which is computationally expensive at that time, and instead it is sufficient to evaluate the determinant and trace to make sure the two eigenvalues are big.

The Shi-Tomasi corner detector [54] directly use

$$\min(\lambda_1, \lambda_2) > \lambda \tag{1.1}$$

as a feature selection criterion to find features which can be tracked well, where λ is a predefined threshold.

According to [60], the stages of typical corner extraction process are,

1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussian.
2. Compute the three images I_x^2, $I_x I_y$, I_y^2.
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas mentioned above, e.g. (1.1).
5. Find local maxima above a certain threshold and report them as detected feature point locations.

A example of Harris features and Shi-Tomasi features is shown as Fig. 1.1. Note that many Harris features and Shi-Tomasi features share the same locations.

1.3 Optical Flow

The goal of optical flow is to compute an independent estimate of motion at each pixel. It generally involves minimizing the brightness or color difference between corresponding pixels summed over the image

$$E(\mathrm{v}_i) = \sum_i [I_1(x_i + \mathrm{v}_i) - I_0(x_i)]^2,$$

where I_0 and I_1 are two consecutive frames in a video and v is the displacement $(u, v)'$. Because the number of variables of v_i is twice the number of measurements, the problem is unconstrained. One way to solve the problem is to perform the summation locally over overlapping regions and use differential techniques. The first instances used first-order derivatives and were based on image translation [19, 32]

$$I(\mathrm{x}, t) = I(\mathrm{x} - \mathrm{v}t, 0).$$

From the Taylor expansion, the gradient constraint equation could be easily derived

$$\nabla I(\mathrm{x},t)\cdot \mathrm{v} + I_t(\mathrm{x},t) = 0, \tag{1.2}$$

where $I_t(\mathrm{x},t)$ denotes the partial time derivative of $I(\mathrm{x},t)$ and $\nabla I(\mathrm{x},t) = (I_x(\mathrm{x},t), I_y(\mathrm{x},t))'$. The equation yields the normal component of motion of spatial contours of constraint intensity

$$\mathrm{v_n} = s\mathrm{n}.$$

where the normal speed s and the normal direction n are given by

$$s(\mathrm{x},t) = \frac{-I_t(\mathrm{x},t)}{\|\nabla I(\mathrm{x},t)\|}, \qquad \mathrm{n}(\mathrm{x},t) = \frac{\nabla I(\mathrm{x},t)}{\|\nabla I(\mathrm{x},t)\|}.$$

Because there are two unknown components of v in (1.2) constrained by one linear equation, further constraints are needed to solve for v.

It is also possible to use second-order derivatives to constrain 2-d velocity [45, 46]

$$\begin{bmatrix} I_{xx}(\mathrm{x},t) & I_{yx}(\mathrm{x},t) \\ I_{xy}(\mathrm{x},t) & I_{yy}(\mathrm{x},t) \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} I_{tx}(\mathrm{x},t) \\ I_{ty}(\mathrm{x},t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

There are many different approaches to set the constraints. For example, Horn and Schunck [32] develop a regularization-based framework where the gradient constraint (1.2) with a global smoothness term to constrain the estimated velocity field $\mathrm{v}(\mathrm{x},t)$. They minimize

$$\int_D (\nabla I\cdot \mathrm{v} + I_t)^2 + \alpha^2(\|\nabla u\|_2^2 + \|\nabla v\|_2^2)d\mathrm{x} \tag{1.3}$$

defined over a domain D, where the weighting factor α reflects the influence of the smoothness term.

Eq. (1.3) can be solved iteratively to obtain the image velocity

$$u^{k+1} = \bar{u}^k - \frac{I_x[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$
$$v^{k+1} = \bar{v}^k - \frac{I_y[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$

where k is the iteration number, u^0 and v_0 denote initial velocity approximations which are set to zero, and $\bar{u}^k$ and $\bar{v}^k$ denote neighborhood averages of u^k and v^k.

Black and Anandan use another smoothness constraint, for an image of size $n \times n$ pixels, they define a grid of sites as

$$S = \{s_1, s_2, \ldots, s_{n^2} | \forall w, 0 \leq i(s_w), j(s_w) \leq n-1\},$$

where $(i(s), j(s)$ is the pixel coordinates of site s. The energy function (1.3) would become

$$\sum_{s \in S}[\sum_{R}(I_x u + I_y v + It, \sigma_1) + \lambda \sum_{n \in G}[\rho_2(u_s - u_n, \sigma_2) + \rho_2(v_s - v_n, \sigma_2)]]$$

where subscripts s and n indicate sites in S, G_s represents the set of north, south, east, west neighbors of s in the grid, σ_1 and σ_2 are scale parameters, and ρ_1 and ρ_2 are different estimators.

Brox et al. [6] try to minimize the energy $E(u, v) = E_{Data} + \alpha E_{Smooth}$, where the data term is defined based on several constraints that they think should be considered in any motion models.

$$E_{Data}(u, v) = \int_D \Psi(I(\mathrm{x+w}) - I(\mathrm{x})|^2 + \gamma |\nabla I(\mathrm{x+w}) - \nabla I(\mathrm{x})|^2) d\mathrm{x},$$

where γ is some weighting and function Ψ is some increasing concave function. The smoothness term is introduced to achieve piecewise smoothness flow field by penalizing the total variation

$$E_{Smooth}(u, v) = \int_D \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2) d\mathrm{x},$$

where $\nabla_3 := (\partial_x, \partial_y, \partial_t)'$ is the spatio-temporal gradient. The functions u and v that minimize the energy could be solved using the Euler-Lagrange equations.

By utilizing the median filtering during optimization to denoise the flow field, Sun et al. [58] minimizes

$$\begin{aligned} E(\mathrm{u}, \mathrm{v}, \hat{\mathrm{u}}, \hat{\mathrm{v}}) = & \sum_{i,j} \{\rho_d(I_0(i,j) - I_1(i + u_{i,j}, j + v_{i,j})) + \\ & \lambda[\rho_s(u_{i,j} - u_{i+1,j}) + \rho_s(u_{i,j} - u_{i,j+1}) + \rho_s(v_{i,j} - v_{i+1,j}) + \rho_s(v_{i,j} - v_{i,j+1})]\} \\ & + \lambda_2(\|\mathrm{u} - \hat{\mathrm{u}}\|^2 + \|\mathrm{v} - \hat{\mathrm{v}}\|^2) + \sum_{i,j} \sum_{(i',j') \in N_{i,j}} \lambda_3(|\hat{u}_{i,j} - \hat{u}_{i',j'}| + |\hat{v}_{i,j} - \hat{v}_{i',j'}|), \end{aligned}$$

where ρ_d and ρ_s are the data and spatial penalty functions, $\hat{\mathrm{u}}$ and $\hat{\mathrm{v}}$ denote an auxiliary flow field, $N_{i,j}$ is the set of neighbors of pixel (i, j), λ, λ_2 and λ_3 are weights. The function

could be optimized by alternately minimizing

$$E_O(\mathrm{u},\,\mathrm{v}) = \sum_{i,j} \rho_d(I_0(i,j) - I_1(i+u_{i,j}, j+v_{i,j})) + \\ \lambda[\rho_s(u_{i,j}-u_{i+1,j}) + \rho_s(u_{i,j}-u_{i,j+1}) + \rho_s(v_{i,j}-v_{i+1,j}) + \rho_s(v_{i,j}-v_{i,j+1})] \\ + \lambda_2(\|\mathrm{u}-\hat{\mathrm{u}}\|^2 + \|\mathrm{v}-\hat{\mathrm{v}}\|^2)$$

and

$$E_M(\hat{\mathrm{u}},\hat{\mathrm{v}}) = \lambda_2(\|\mathrm{u}-\hat{\mathrm{u}}\|^2 + \|\mathrm{v}-\hat{\mathrm{v}}\|^2) + \sum_{i,j} \sum_{(i',j')\in N_{i,j}} \lambda_3(|\hat{u}_{i,j} - \hat{u}_{i',j'}| + |\hat{v}_{i,j} - \hat{v}_{i',j'}|).$$

1.3.1 Kanade-Lucas-Tomasi (KLT) Feature Tracker

The Kanade-Lucas-Tomasi (KLT) algorithm [42, 62] is a popular optical flow method. The displacement vector v is chosen to minimize the residual error defined by

$$\epsilon = \int_{\mathcal{W}} [I(\mathrm{x}-\mathrm{v},t) - I(\mathrm{x},t+\tau)]^2 w \; d\mathrm{x} \tag{1.4}$$

over the given window $\mathcal{W}$. In this expression, w is a weighting function. In the simplest case, it could be set to 1. Alternatively, w could be a Gaussian-like function to emphasize the central area of the window.

By Taylor expansion, we have

$$I(\mathrm{x}-\mathrm{v}) = I(\mathrm{x}) - \mathrm{g}\cdot\mathrm{v},$$

where g is the first derivative of $I(\mathrm{x})$, and the residue defined in equation (1.4) could be written as

$$\epsilon = \int_{\mathcal{W}} [I(\mathrm{x}) - g\cdot\mathrm{v} - J(\mathrm{x})]^2 w \; d\mathrm{x} = \int_{\mathcal{W}} (h - \mathrm{g}\cdot\mathrm{v})^2 w \; d\mathrm{x},$$

where $h = I(\mathrm{x},t) - I(\mathrm{x},t+\tau)$.

The residue is a quadratic function of the displacement v. The minimum could be obtained when the first derivative equals zero:

$$\int_{\mathcal{W}} (h - \mathrm{g}\cdot\mathrm{v})\mathrm{g} w \; dA = 0.$$

Since $(\mathrm{g}\cdot\mathrm{v})\mathrm{g} = (\mathrm{g}\mathrm{g}^T)\mathrm{v}$, and v is assumed to be constant within $\mathcal{W}$, we have

$$(\int_{\mathcal{W}} \mathrm{g}\mathrm{g}^T w \; dA)\mathrm{v} = \int_{\mathcal{W}} h\mathrm{g} w \; dA.$$

This is a system of two scalar equations in two unknowns. It can be rewritten as

$$G\mathrm{v} = e,$$

where the coefficient matrix is the symmetric, 2×2 matrix

$$G = \int_{\mathcal{W}} \mathrm{g}\mathrm{g}^T w \, dA,$$

and the right-hand side is the two-dimensional vector

$$e = \int_{\mathcal{W}} (I(\mathrm{x}, t) - I(\mathrm{x}, t + \tau)\mathrm{g} w \, dA.$$

In practice, for discrete images, the displacement is

$$\mathrm{v} \approx [\sum_{\mathrm{x}} (\frac{\partial I}{\partial \mathrm{x}})^T [I(\mathrm{x},\, \mathrm{t}) - I(\mathrm{x}, t + \tau)]][\sum_{\mathrm{x}} (\frac{\partial I}{\partial \mathrm{x}})^T (\frac{\partial I}{\partial \mathrm{x}})]^{-1}.$$

In implementation the displacement between the feature and its estimated position are calculated in an iterative Newton-Raphson style search. Smoothing and multi-resolution are used to improve the search range.

The KLT algorithm usually combines with the Shi-Tomasi features to obtain better performance. In this process, the Shi-Tomasi features are extracted from the first frame, and the KLT algorithm is applied to find the corresponding features in the following frames. Because the KLT algorithm is much faster than traditional techniques for examining far fewer potential matches between the images, it is widely used in practice.

1.4 Mathematical Background

Two kinds of models are widely used in motion segmentation. One is the affine camera model, which generalizes orthographic, weak-perspective and paraperspective projection. Under the affine camera model, the tracked feature points from a rigid moving object lie on a linear subspace of dimension at most 4. Thus the motion segmentation problem could be changed to subspace clustering problem. The other one is the perspective projection model. Under this model, tracked feature points from each moving object live in a multilinear variety for example, bilinear for two views, trilinear for three views, etc. Therefore the motion segmentation is solved by clustering these multilinear varieties. Typically, most

prior work that adapts the perspective projection model has utilized algebraic methods for factorizing the multilinear variety [31, 71] or statistical methods for two and multiple views [64, 52].

At present, the performances of the perspective projection model based methods are still much worse than those of the affine camera model based methods. Also we use the affine camera model in the dissertation, so in this section, we will only introduce the affine camera model.

1.4.1 Affine Camera Model

Many works on motion segmentation [37], [12], [18], [41] use the affine camera model, which is approximatively satisfied when the objects are far from the camera. Under the affine camera model, a point on the image plane (x, y) is related to the real world point (X, Y, Z) by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}}_{A \in \mathbb{R}^{2\times 4}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1.5}$$

where A is the affine motion matrix, which is determined by the camera calibration matrix $K \in \mathbb{R}^{2\times 3}$ and the relative orientation of the image plane with respect to the world coordinates $(R, t) \in SE(3)$.

Let $t = (x^1, y^1, x^2, y^2, \ldots, x^F, y^F)^T$ be a trajectory of a tracked feature point in F frames. Given P trajectories undergoing the same rigid motion, the *measurement matrix* $W = [t_1, t_2, \ldots, t_P]$ is constructed. From equation (1.5), W can be decomposed into a *motion matrix* $M \in \mathbb{R}^{2F\times 4}$ and a *structure matrix* $S \in \mathbb{R}^{4\times P}$ as

$$W = MS$$

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_P^1 \\ y_1^1 & y_2^1 & \cdots & y_P^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \cdots & x_P^F \\ y_1^F & y_2^F & \cdots & y_P^F \end{bmatrix} = \begin{bmatrix} A^1 \\ \vdots \\ A^F \end{bmatrix} \begin{bmatrix} X_1 & \cdots & X_P \\ Y_1 & \cdots & Y_P \\ Z_1 & \cdots & Z_P \\ 1 & \cdots & 1 \end{bmatrix}, \tag{1.6}$$

where A^f is the affine motion matrix at frame f. It implies that $\text{rank}(W) \leq 4$. In other words, under the affine camera model, the trajectories from a rigidly moving object reside in a subspace of dimension at most 4. Also, it is worth noting that the rows of each A^f

involve linear combinations of the first two rows of the rotation matrix R^f, hence $\text{rank}(W) \geq \text{rank}(A^f) = 2$.

Additionally, the entries of the last row of the structure matrix S are identically 1. It is easy to derive the orthographic camera model [63]. Define the registered trajectories as

$$\tilde{t}_i = t_i - \frac{\sum_{i=1}^{P} t_i}{P}, \tag{1.7}$$

then the registered measurement matrix

$$\tilde{W} = [\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_P] \tag{1.8}$$

is at most rank 3. This means that the trajectories are in a 3-D affine subspace within the 4-D space.

Ideally, the measurement matrix should contain perfect information about the object being tracked. Unfortunately, in practice most trackers can only provide inaccurate point tracks if placed in an unstructured environment.

1.4.2 Segmentation of Multiple Rigid Motions

Assume that there are P trajectories $\{t_i\}_{t=1}^{P}$ corresponding to k rigid moving objects relative to a moving camera. People want to cluster these trajectories according to their corresponding motion. From the affine camera model, the trajectories from one motion span a linear subspace of $\mathbb{R}^{2F}$ whose dimension d is not fixed, thus the motion segmentation problem is equivalent to clustering a set of points into k subspaces of $\mathbb{R}^{2F}$ of unknown dimension $d_i \in [2, 3, 4]$ for $i = 1, \ldots, k$.

If we put the measurement matrices from k motions together as

$$W = [W_1, W_2, \ldots, W_k]\, \Gamma \in \mathbb{R}^{2F \times P},$$

where $W_i \in \mathbb{R}^{2F \times P_i}$ is the measurement matrix of i-th motion which has P_i trajectories, and Γ^T is an unknown matrix permuting the $P = \sum_{i=1}^{k} P_i$ trajectories.

Based on the factorization in equation (1.6), we have $W_i = M_i S_i$. Thus W can be

factorized into matrices $M \in \mathbb{R}^{2F \times \sum_{i=1}^{n} d_i}$ and $S \in \mathbb{R}^{(\sum_{i=1}^{n} d_i) \times P}$ as

$$\begin{aligned} W &= [M_1, M_2, \ldots, M_k] \begin{bmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_k \end{bmatrix} \Gamma \\ &= MS\,\Gamma. \end{aligned} \tag{1.9}$$

From equation (1.9), it is intuitive to solve the motion segmentation problem by finding a permutation matrix Γ, such that the matrix $W\Gamma^T$ could be decomposed into a motion matrix M and a block diagonal structure matrix S. However, the motion subspaces $\{\mathcal{W} \subset \mathbb{R}^{2F}\}_{i=1}^{k}$ must be independent to factor W as (1.9) [34]. In other words, for every $i \neq j$, we must have $\dim(\mathcal{W}_i \cap \mathcal{W}_j) = 0$, so that

$$\operatorname{rank}(W) = \sum_{i=1}^{k} \dim(\mathcal{W}_i).$$

Unfortunately, most pratical motion sequences show partially dependent motions, which means

$$0 < \dim(\mathcal{W}_i \cap \mathcal{W}_j) < \min(\dim(\mathcal{W}_i), \dim(\mathcal{W}_j)) \qquad i, j \in [1, \ldots, k], i \neq j.$$

For instance, when two objects have the same rotational but different translational motion [57], or for articulated motions [74].

1.5 Spectral Clustering

The goal of clustering is to partition data points into several groups according to their similarities. Spectral clustering is one of the widely used clustering techniques, which has been used in many applications. Compared to the traditional clustering algorithms, such as k-means, expectation-maximization (EM) algorithm, spectral clustering often gives better clustering results, and it is also very simple to implement and has relatively fast computational speed.

Given a set of data points $x_1, \ldots, x_n$, spectral clustering needs a affinity metric to measure the similarity between all pairs of data points x_i and x_j, for example, the Gaussian similarity function

$$A_{ij} = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}},$$

where the parameter σ controls the strength of the connection between points. The affinity is always a non-negative number in the range $[0, 1]$. Consider an undirected graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$. Each vertex v_i represents a point x_i in the graph, and $w_{i,j}$ is the weight of the edge connecting vertices v_i and v_j. The larger the weight is, the stronger the connection between the two nodes. If weight $w_{ij} = 0$, vertices v_i and v_j are not connected. It is of interest to find a partition of the graph such that the weights between different groups are small and the edges within a group have large weights.

Given a subset of vertices $A \subset V$, and its complement $V \setminus A$ as $\bar{A}$, and define the connections between two sets $A, B \subset V$ which are not necessarily disjoint as

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

and the degree matrix D as a diagonal matrix in which

$$D_{ii} = \sum_{j=1}^{n} w_{ij}.$$

Several different cuts are introduced to formalize the goal. For example, the mincut approach simply consists in choosing the partition $A_1, \ldots, A_k$ which minimizes

$$\text{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i).$$

However, the mincut is not robust enough to obtain satisfactory partitions in practice. The two common objective functions used today are the RatioCut [29] and the normalized cut (Ncut) [53]. RatioCut measures the size of a subset A of a graph by the number of vertices in it while the normalized cut measures it by the weights of its edges.

$$\text{RatioCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$

$$\text{Ncut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{\text{vol}(A_i)} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)},$$

where $|A_i|$ is the number of vertices in A_i and $\text{vol}(A) = \sum_{i \in A} D_{ii}$ is the sum of the edge weights between vertices in A. We only discuss the normalized cut in the following since the case for RatioCut is very similar.

Input: a set of points $x_1, \ldots, x_n$ of dimensionality of d from k clusters.
1. Construct the affinity matrix $A \in \mathbb{R}^{n \times n}$ by some affinity metric if $i \neq j$, and $A_{ii} = 0$.
2. Construct D as a diagonal matrix whose (i, i)-element is the sum of A's i-th row and $S = D^{-1/2} A D^{-1/2}$.
3. Compute the leading k eigenvectors $u_1, \ldots, u_k$ of S. and build the matrix $U \in \mathbb{R}^{n \times k}$
4. Normalize the rows of U.
5. Treat each row of U as point in $\mathbb{R}^k$ and cluster them into k clusters by k-means or any other algorithm.
6. Assign points x_i to their corresponding clusters.

Figure 1.2: The normalized spectral clustering algorithm [48].

Suppose there are $k = 2$ clusters and define f as the cluster indicator vector by

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} & \text{if } v_i \in A \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} & \text{if } v_i \in \bar{A}. \end{cases} \tag{1.10}$$

It could be approved that the problem of minimizing Ncut could be rewritten to the equivalent problem

$$\min_A f' L f \quad \text{subject to} \quad f \text{ as in (1.10)}, \, Df \perp \mathbb{1}, f' D f = \mathrm{vol}(V).$$

where the Laplacian matrix L is defined as

$$L = D - W.$$

By allowing f to take arbitrary real values, the problem could be relaxed as

$$\min_{f \in \mathbb{R}^n} f' L f \quad \text{subject to} \quad Df \perp \mathbb{1}, f' D f = \mathrm{vol}(V).$$

After substituting $g = D^{1/2} f$, the problem is

$$\min_{g \in \mathbb{R}^n} g' D^{-1/2} L D^{-1/2} g \quad \text{subject to} \quad g \perp D^{1/2}\mathbb{1}, \|g\|^2 = \mathrm{vol}(V).$$

One can observe that the problem is in the form of the standard Rayleigh-Ritz theorem, and the solution g is given by the second eigenvector of $D^{-1/2} L D^{-1/2}$.

When $k > 2$, the indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})'$ is defined by

$$h_{i,j} = \begin{cases} 1/\sqrt{\mathrm{vol}(A_j)} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \ldots, n; j = 1 = 1, \ldots, k.$$

Then build matrix H by using k indicator vectors as its columns. Since $H'H = I$, $h_i'Dh_i = 1$, and $h_i'Lh_i = \text{cut}(A_i, \bar{A}_i/\text{vol}(A_i)$, the problem of minimizing Ncut could be changed to

$$\min_{A_1,\ldots,A_k} \text{Tr}(H'LH) \quad \text{subject to } H'DH = I.$$

By relaxing the discreteness condition and substituting $T = D^{1/2}H$, we obtain the relaxed problem

$$\min_{T\in\mathbb{R}^{n\times k}} \text{Tr}(T'D^{-1/2}LD^{-1/2}T) \quad \text{subject to } T'T = I.$$

The solution of the standard trace minimization problem is the matrix T which contains the first k eigenvectors of $D^{-1/2}LD^{-1/2}$.

Finally one could yield the normalized spectral clustering method as shown in Figure 1.2.

The computational cost of the Algorithm 1.2 could be analyzed as the following. Some inexpensive steps are omitted.

1. **Construct the affinity matrix**: The cost of obtaining an affinity is $O(d)$. Constructing the whole affinity matrix A requires $O(n^2d)$.
2. **Construct matrix** S: Since A is a symmetric matrix and D is a diagonal matrix, calculating S needs $O(n^2)$ operations.
3. **Eigendecomposition**: The complexity of generalized eigendecomposition is $O(n^3)$.
4. k**-means**: Calculating the distance between two points is $O(k)$. For each point, we need to compare its distance to k centroids and assign it to the cluster with the smallest distance. The complexity is $O(nk^2)$. Plus, k-means runs in an iterative way. If the iteration number is m, the whole complexity would be $O(nk^2m)$.

Combining all the steps, because the number of points n is always much bigger than the number of clusters k, the dimensionality of points d, and the number of iterations in k-means m, the overall complexity of spectral clustering is $O(n^3)$, in which the eigendecomposition is the most time-consuming step.

1.6 Dataset

Datasets for quantitatively evaluating algorithms are important to boost the development of better algorithms. Scientific research always benefits from objective measurement to compare among methods. Several datasets are used in the dissertation for comparison with different algorithms in different situations.

1.6.1 The Hopkins 155 Dataset

The Hopkins 155 Dataset[1] [66] is a public benchmark for testing feature based motion segmentation algorithms. It contains video sequences along with some feature points extracted on the first or the second frame and tracked in the following frames. The datset also provides the ground-truth segmentation of all the sequences for comparison purposes. Figure 1.3 shows some sample frames from videos in the Hopkins 155 database with feature points added on. The sequences contain a great variety of motions, such as degenerate and non-degenerate motions, independent and partially dependent motions, articulated motions, nonrigid motions [66], etc.

Based on the content of the video and the motion types, the 155 sequences can be categorized into three main groups:

Checkerboard sequences: this group includes 104 sequences of indoor scenes which were taken with a camera under controlled conditions. The checkerboard pattern which appears on most of the objects is used to assure a large number of tracked points.

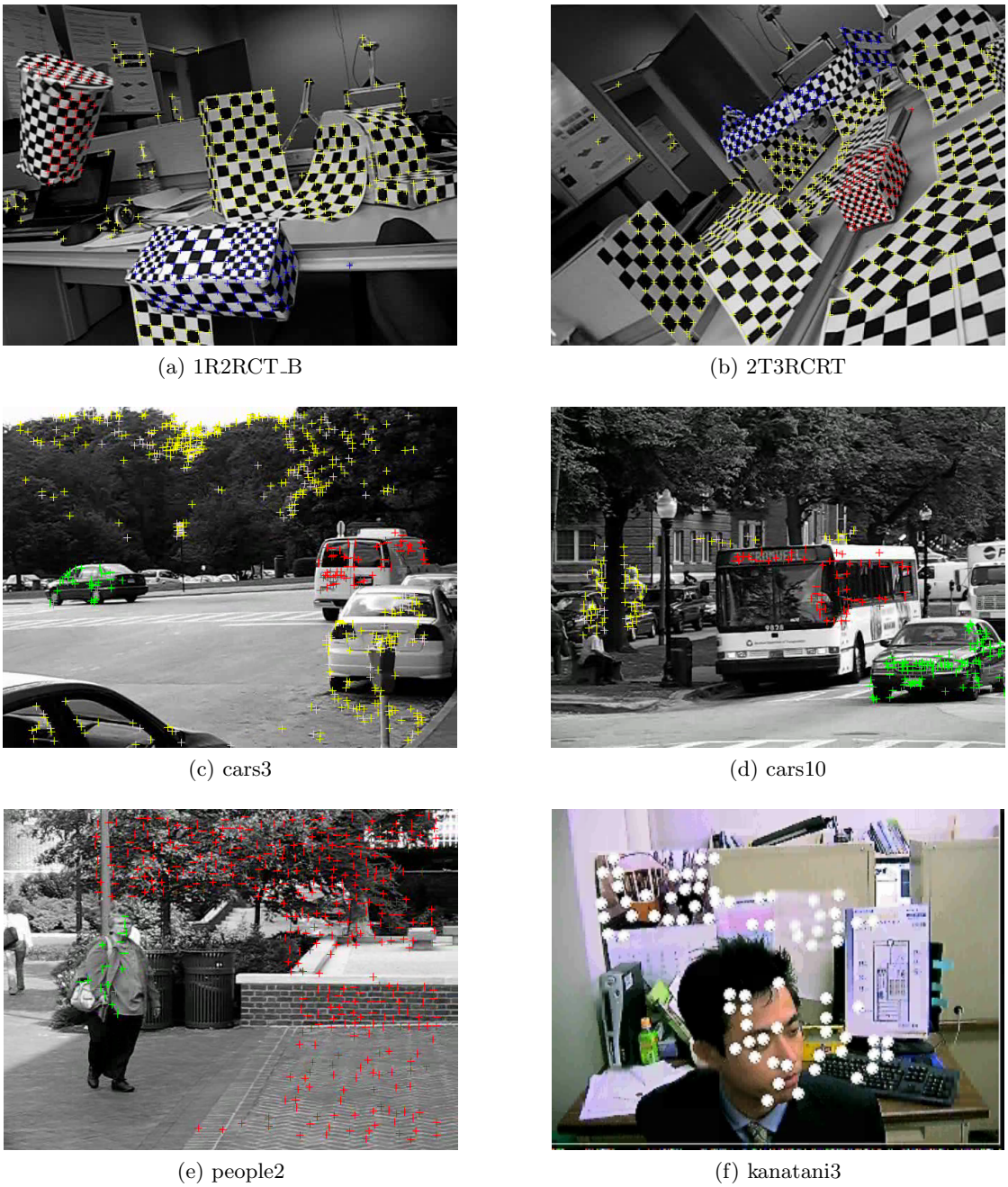
Traffic sequences: this group includes 38 sequences of outdoor traffic scenes which were taken by moving cameras.

Articulated/non-rigid sequences: this group has 13 sequences showing motions from non-rigid objects, for example, head and face motions, people walking, etc.

Table 1.1: Distribution of the number of points and frames in Hopkins 155 Dataset

	2 Groups			3 Groups		
	Sequences	Points	Frames	Sequences	Points	Frames
Checkerboard	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Articulated	11	155	40	2	122	31
All	120	266	30	35	398	29

[1] Available at http://www.vision.jhu.edu/data/hopkins155/.



(a) 1R2RCT_B (b) 2T3RCRT

(c) cars3 (d) cars10

(e) people2 (f) kanatani3

Figure 1.3: Sample frames from the Hopkins 155 dataset with tracked points superimposed.

In the 155 sequences, some point trajectories were provided in the respective datasets. some are produced by a tool based on a tracking algorithm implemented in OpenCV [2].

[2]The OpenCV library, http://opencv.org/

The ground-truth segmentation was assigned by an operator manually. The operator also removed obviously wrong tracked feature trajectories. The 155 sequences are obtained from 50 videos. Most sequences with 2 motions are extracted from sequences with 3 motions. The dataset chooses 2 clusters from a sequence with 3 motions and names it as a new sequence.

Table 1.1 reports the number of sequences and the average number of tracked points and frames for each category. The number of points per sequence ranges from 39 to 556, and the number of frames from 15 to 100 [66].

The Hopkins 155 dataset provides a platform for researchers to compare the efficiency of different algorithms, it boosts the research on multi-motion segmentation. Since its publication, many sparse motion segmentation methods [69, 50, 37, 18, 41, 78] were developed and benchmarked on the dataset.

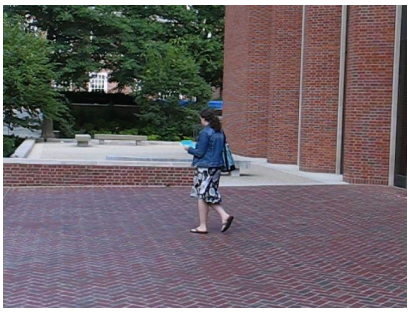
1.6.2 Moseg Dataset

Brox, et al [7] provide a dataset of 26 video sequences with pixel-accurate segmentation annotation of moving objects[3]. The dataset has 14 sequences from detective movies and the 10 car and 2 people sequences from Hopkins 155 dataset. 189 frames are annotated in pixel-wise accuracy, and not every frame is annotated. More frames are annotated at the beginning of a shot so that methods which could not handle long sequences can also be evaluated. In figure 1.4 are shown sample frames and their corresponding ground truth in the dataset.

The annotated frames allows one to obtain the ground truth of many tracked feature points. Compared to the Hopkins 155 dataset, the Moseg dataset is more suitable to test algorithms that could deal with a large number of trajectories or long trajectories. The only drawback is that there is no ground truth for every frame, so that the segmentation may not be that accurate, compared to the Hopkins 155 dataset. Nonetheless, the performance of different algorithms still could be revealed based on the relative ranking on the dataset.

[3] http://lmb.informatik.uni-freiburg.de/resources/datasets/

(a) cars9 (frame 1)



(b) people1 (frame 1)

(c) marple1 (frame 1)

(d) GT of cars9 (frame 1)

(e) GT of people1 (frame 1)

(f) GT of marple1 (frame 1)

(g) cars9 (frame 60)

(h) people1 (frame 40)

(i) marple1 (frame 300)

(j) GT of cars9 (frame 60)

(k) GT of people1 (frame 40)

(l) GT of marple1 (frame 300)

Figure 1.4: Sample frames from the Moseg dataset and the corresponding ground truth. Note that cars9 and people1 are sequences from the Hopkins 155 dataset.

1.6.3 Middlebury Optical Flow Database

Army frame 0 **Army** frame 1 **Army** GT flow flow color coding

(a) Hidden texture data

Grove frame 0 **Grove** frame 1 **Grove** GT flow flow color coding

(b) Synthetic data

Backyard frame 0 **Backyard** frame 1 GT interpolated frame

(c) High-speed data for interpolation

Teddy frame 0 **Teddy** frame 1 **Teddy** GT flow flow color coding

(d) Stereo data

Figure 1.5: Different types of data in the Middlebury database.

The Middlebury optical flow database [3] is used for the quantitative evaluation of different optical flow algorithms. There are four types of data to test different aspects of

optical flow algorithms in the database,

1. sequences with nonrigid motion where the ground-truth is determined by tracking a hidden fluorescent texture.
2. realistic synthetic sequences.
3. high frame-rate video used to study interpolation error.
4. modified stereo sequences of static scenes.

The third type doesn't have the ground truth. To the other types, there are two kinds of sequences, one kind with public ground truth which could be used for training, the other kind with hidden ground truth for test purpose. Sample data is shown in figure 1.5.

This database also proposes a set of evaluation methods to measure the performance of optical flow algorithms. The average angular error (AE) between a flow vector (u, v) and the ground truth flow $(u_{\mathrm{GT}}, v_{\mathrm{GT}})$ is the angle in 3D space between $(u, v, 1.0)$ and $(u_{\mathrm{GT}}, v_{\mathrm{GT}}, 1.0)$

$$\mathrm{AE} = \arccos\left(\frac{1.0 + u \times u_{\mathrm{GT}} + v \times v_{\mathrm{GT}}}{\sqrt{1.0 + u \times v + v \times v}\sqrt{1.0 + u_{\mathrm{GT}} \times u_{\mathrm{GT}} + v_{\mathrm{GT}} \times v_{\mathrm{GT}}}}\right).$$

The absolute flow endpoint error (EE) is

$$\mathrm{EE} = \sqrt{(u - u_{\mathrm{GT}})^2 + (v - v_{\mathrm{GT}})^2}.$$

The interpolation error (IE) is the root-mean-square difference between the ground truth image and the estimated interpolated image

$$\mathrm{IE} = \left[\frac{1}{N}\sum_{(x,y)}(I(x,y) - I_{\mathrm{GT}}(x,y))^2\right]^{\frac{1}{2}},$$

where N is the number of pixels. The normalized interpolation error (NE) between an interpolated image I and a ground truth image I_{GT} is given by:

$$\mathrm{NE} = \left[\frac{1}{N}\sum_{(x,y)}\frac{(I(x,y) - I_{\mathrm{GT}}(x,y))^2}{\|\nabla I_{\mathrm{GT}}(x,y)\|^2 + \epsilon}\right]^{\frac{1}{2}}.$$

The database is freely available on the web at http://vision.middlebury.edu/flow/. Many researchers have uploaded their results to the database and the database will rank those algorithms based on the proposed measures.

1.7 Outlines

The dissertation is organized as follows. In chapter 2, we will create a framework to evaluate different feature tracking algorithms. Also in this framework, we will introduce a feature tracking algorithm based on RankBoost that automatically prunes bad trajectories obtained by an optical flow algorithm. In chapter 3, we will develop a motion segmentation algorithm based on spectral clustering. In chapter 4, we will propose a novel motion segmentation algorithm based on learning. In chapter 5, we will establish a scalable motion segmentation algorithm which is based on the Swendsen-Wang Cuts. In chapter 6, we will give some concluding remarks and study some interesting problems to guide our future works.

CHAPTER 2

LEARNING A QUALITY-BASED RANKING FOR FEATURE POINT TRAJECTORIES

2.1 Introduction

Motion segmentation and estimation over tens or hundreds of frames is a difficult problem that still poses many challenges. Two of these challenges are finding good algorithms for tracking feature points over long sequences and a framework for accurately evaluating and comparing these algorithms.

While there exist frameworks and datasets for evaluating optical flow algorithms, such as the Middlebury dataset [3], the optical flow is evaluated only on two frames in these datasets. When estimating motion over long sequences, occlusion handling becomes very important because a large percentage of the image pixels will sooner or later be occluded in the sequence.

Many motion analysis algorithms based on tracking feature points [62] only track a subset of the image pixels, and these subsets can be different for different algorithms. Evaluation of the obtained motion fields would ideally require dense ground truth correspondences over the entire sequence, which is difficult to obtain.

There are efforts on evaluating the performance of a tracker [47] [76] objectively using a manually annotated dataset. For this reason, a number of metrics are defined in [47] [76]to evaluate motion trackers comprehensively. However, this raises the question: which metric is the best for ranking? Or should the metrics be combined in some way, such as by a weighted sum? A similar issue exists in the Middlebury optical flow dataset [3] where a number of error metrics are evaluated based on the ground truth.

Due to the complexity of reality scenes, no methods usually obtain the best performance on all sequences being tested. Also, when comparing the performance of trackers, people are always talking about the average performance. It is not strange that the best tracker would generate some bad trajectories, while a bad tracker would produce some good ones. Thus, in order to control the quality of the output trajectories, pruning out the bad trajectories seems to be a reasonable strategy.

In this chapter we bring two contributions. First, we introduce an error measure for evaluating feature tracking algorithms on sequences containing objects undergoing rigid motion under the affine camera model. The proposed measure was observed to be consistent with the relative ranking of several selected optical flow algorithms on the Middlebury dataset. Moreover, the proposed error measure was observed to be more robust than an indirect measure based on evaluating the segmentation error of the generated trajectories.

Second, we introduce a feature tracking algorithm based on RankBoost that ranks the feature trajectories obtained by any optical flow algorithm from well tracked to badly tracked and removes a percentage of the badly tracked ones.

The proposed feature tracking algorithm is evaluated using two different error measures: the proposed error measure and an indirect measure based on motion segmentation accuracy. Both evaluations show that the proposed feature tracker outperforms other feature trackers based on optical flow on a number of publicly available image sequences.

2.2 A Method for Evaluating Feature Trackers

As mentioned in the introduction, it is difficult to evaluate feature tracking algorithms because dense ground truth motion fields over tens or hundreds of frames have not been obtained so far. There exist indirect error measures such as the segmentation error [7] returned by a predefined motion segmentation algorithm on the obtained feature trajectories. However, such a measure depends on the segmentation algorithm and tends to be quite unstable, as it will be seen in experiments.

In this section we investigate a direct error measure that can be used for sequences containing rigid motions under the affine camera model. The proposed measure is based on

the observation that it is unlikely for a badly tracked feature point to accurately follow the rigid motion model of the object it belongs.

To use the proposed measure, the following are needed for each sequence being evaluated:

1. A set of full length *ground truth (GT) feature point trajectories* for each motion of the sequence. These trajectories have been verified to be correct and manually adjusted if necessary.
2. A dense (manual) segmentation of the first frame of the sequence (optional).

The proposed measure evaluates any given feature point trajectory by obtaining the motion label based on the dense segmentation of the first frame and the rigid motion model from the GT feature trajectories for that object.

Assume the set of full length GT trajectories for the rigid motion of interest is $t_i = (x_i^1, y_i^1, ..., x_i^T, y_i^T)', i = 1, ..., k$. Assume the sequence obeys the affine camera model [66], which generalizes orthographic, paraperspective and weak perspective projections. From the trajectories $t_1, ..., t_k$, we construct the *measurement matrix*

$$W = (t_1, t_2, ..., t_k) = \begin{pmatrix} x_1^1 & x_2^1 & ... & x_k^1 \\ y_1^1 & y_2^1 & ... & y_k^1 \\ .. & . & . & . \\ x_1^T & x_2^T & ... & x_k^T \\ y_1^T & y_2^T & ... & y_k^T \end{pmatrix}$$

The affine camera model [63, 66] factorization allows us to write

$$W = MS$$

where M is a $2T \times 4$ *motion matrix* and S is a $4 \times k$ matrix representing the 3D structure of the points. We will also refer to the matrix M as the motion model for the rigid object containing the trajectories $t_1, ..., t_k$.

There is an inherent ambiguity in M and S but we will show that it is irrelevant for our evaluation purpose.

2.2.1 RMSE Error for One Trajectory

Given a feature point trajectory $t = (x^b, y^b, ..., x^e, y^e)'$ that needs to be evaluated, assumed to belong to this motion, beginning at frame b and ending at frame e will have a

corresponding 3D structure point P obtained by least squares:

$$P = \underset{P}{\operatorname{argmin}} \|t - M_{be}P\|^2 = \underset{P}{\operatorname{argmin}} (t - M_{be}P)'(t - M_{be}P) = (M'_{be}M_{be})^{-1}M'_{be}t \tag{2.1}$$

where M_{be} is the submatrix of M corresponding to the frames from b to e.

We can estimate P in a least square sense from the entire trajectory t and generate a most likely rigid motion trajectory as the projection $t^G = M_{be}P$ of P to the frames from b to e through the motion matrix M.

Finally define the SSE tracking error of the trajectory t as the sum of the squared errors between the trajectory points and the corresponding points of t^G:

$$\mathrm{SSE}_W(t) = \min_P (t - M_{be}P)'(t - M_{be}P) = t't - t'M_{be}(M'_{be}M_{be})^{-1}M'_{be}t \tag{2.2}$$

Remark 2.2.1. *The $SSE_W(t)$ is invariant to the choice of M and S in the decomposition $W = MS$.*

Proof. Multiplying M by any 4×4 invertible matrix A results in multiplying M_{be} by A. It can be easily verified that $\mathrm{SSE}_W(t)$ does not change when M_{be} is multiplied by an invertible matrix A. ■

We can now define the *RMSE error* for a trajectory t as:

$$\mathrm{RMSE}_W(t) = \sqrt{\frac{\mathrm{SSE}_W(t)}{e - b + 1}}, \tag{2.3}$$

which is measured in pixels and tells how well the trajectory fits the motion model of W.

The error measures $\mathrm{SSE}_W(t), \mathrm{RMSE}_W(t)$ can be computed for any trajectory t of a sequence that has the GT feature trajectories, against any measurement matrix W from the same sequence.

The $\mathrm{RMSE}_W(t)$ error can be used to evaluate the fitness of the trajectory t to the motion W, when the GT trajectories are available. If there is more than one motion in the ground truth, the error $\mathrm{RMSE}(t)$ of a trajectory can be obtained as follows:

1. If a dense motion segmentation is available, it can be used to obtain the motion label of the trajectory and the corresponding measurement matrix W can be used, so $\mathrm{RMSE}(t) = \mathrm{RMSE}_W(t)$.

2. If a dense segmentation is not available, the most likely motion is chosen as the motion that leads to the smallest RMSE error. Thus, the RMSE error is computed against all motions, and the smallest one is selected, so $\text{RMSE}(t) = \min_W \text{RMSE}_W(t)$.

In what follows we will use the second alternative since we don't have manual segmentations for the first frame of all 47 Hopkins sequences that will be used for evaluation.

We can now define the measure for evaluating the quality of the trajectories obtained by a feature tracking algorithm as the percentage of trajectories with RMSE larger than a threshold τ.

$$\text{RMSE}_\tau = \frac{1}{N}|\{t_i, \text{RMSE}(t_i) \geq \tau, i = \overline{1, N}\}| \tag{2.4}$$

2.2.2 Comparison with the Middlebury Dataset

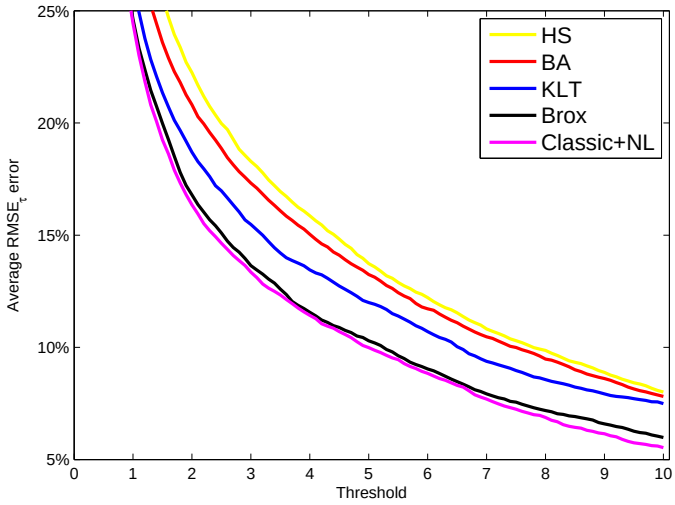


Figure 2.1: The average RMSE_τ measure vs the threshold τ for five optical flow algorithms. The relative order of the algorithms is consistent with the Middlebury ranking for $\tau \geq 3$.

The accuracy of the proposed error metric was evaluated on five standard optical flow algorithms: Kanade-Lucas-Tomasi (KLT) [62], Brox [6], Classic+NL [58], Black & Anandan (BA) [5] and Horn & Schunck (HS) [32]. An brief introduction about the five algorithms could be found in section 1.3.

The algorithms were evaluated on 47 image sequences from the Hopkins 155 dataset. The trajectories were obtained using the different algorithms starting from the same points in the first frame, as described in Section 2.4.1.

The relative ranking of these algorithms on the Middlebury dataset was compared with the ranking obtained from the proposed measure RMSE_τ with a range of thresholds τ.

In Figure 2.1 is shown the average RMSE_τ of the five algorithms for a range of values τ. The relative order based on the RMSE measure is consistent with the Middlebury relative ranking for a large range of values $\tau \geq 3$.

The only exception is the KLT algorithm that ranks better than in the Middlebury ranking.

We will see in the experimental section that the KLT performance changes with the strength of the feature points, so it would make sense that it ranks better than in the Middlebury dataset because the Hopkins sequences contain a majority of checkerboard images with strong feature points.

On the other hand, in Figure 2.6 are shown two indirect error measures based on motion segmentation. One can see from the graph that the measures are quite unstable and are not consistent with the Middlebury ranking.

2.3 A Trajectory Pruning Algorithm based on RankBoost

In this section we present a method for ordering feature point trajectories by predicting their relative quality using RankBoost [22]. This ranking is used to remove a percentage (e.g. 20%) of the trajectories predicted to have the worst quality.

2.3.1 The RankBoost Algorithm

We will use RankBoost [22] to learn an ordering of the feature point trajectories obtained by any feature tracker or optical flow algorithm.

RankBoost is a boosting algorithm that combines a number of ranking functions $h_i : \mathcal{X} \to \mathbb{R}, i = \overline{1, M}$ into a single ranking function $H : \mathcal{X} \to \mathbb{R}$ for instances $x \in \mathcal{X}$.

Given a set of training instances $S = \{x_i \in \mathcal{X}, i = \overline{1, n}\}$, we assume that a ground truth ranking is given on these instances as a function $\Phi : S \times S \to \mathbb{R}$, where $\Phi(x_0, x_1) > 0$ means x_1 should be ranked above x_0 and vice versa.

RankBoost attempts to find a ranking that is similar to the given function Φ. In order to formalize this goal, construct the distribution D by $D(x_0, x_1) = c \cdot \max\{0, \Phi(x_0, x_1)\}$,

Given: initial distribution D over $\mathcal{X} \times \mathcal{X}$.
Initialize: $D_1 = D$.
for $t = 1, \ldots T$ **do**
1. Train weak learner using distribution D_t to get weak ranking h_t.
2. Choose $\alpha_t \in \mathbb{R}$.
3. Update: $D_{t+1}(x_0, x_1) = \frac{D_t(x_0,x_1)exp(\alpha_t(h_t(x_0)-h_t(x_1))))}{Z_t}$ where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

end for
Output the final ranking: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

Figure 2.2: The original RankBoost algorithm [22].

where c is a constant to make $\sum_{x_0,x_1} D(x_0, x_1) = 1$. The learning algorithm tries to find a final ranking $H : \mathcal{X} \to \mathbb{R}$ that minimizes the weighted sum of wrong orderings:

$$\text{rloss}_D = \sum_{(x_0,x_1)\in S\times S} D(x_0, x_1)[\![H(x_1) \leq H(x_0)]\!]$$

where the notation $[\![\pi]\!]$ is defined to be 1 if predicate π holds and 0 otherwise.

Like other boosting algorithms, RankBoost operates in rounds. In each round t, RankBoost maintains a distribution D_t on $S \times S$ and selects the best weak ranker h_t along with its corresponding weight α_t from a large set of candidate weak rankers. The weight $D_t(x_0, x_1)$ will be decreased if $h_t(x_1) > h_t(x_0)$, and increased otherwise. Thus D_t will tend to concentrate on the pairs that are hard to rank. The final ranking H is a weighted sum of the weak rankers selected in each round. The algorithm is given in more detail in Figure 2.2.

2.3.2 Features Used by the Weak Learners

The input to a weak learner $h : \mathcal{X} \to \mathbb{R}$ is a trajectory $x \in \mathcal{X}$. Based on the trajectory, shape and appearance features are generated. The process of obtaining the features is illustrated in figure 2.3.

The shape features are based on the position information along the trajectory. The features are parametrized by two attributes: the start time s, and the time gap t between two adjacent selected points. Based on these parameters we generate features that are supposed to measure constant velocity along the trajectory. For example, if we denote the

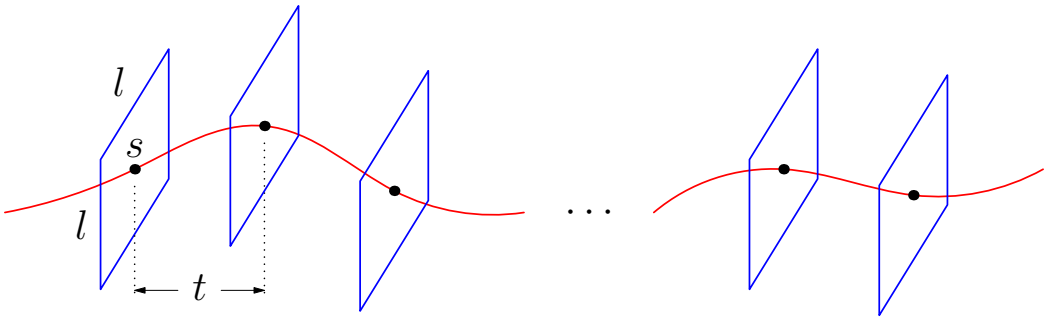


Figure 2.3: The features used for ranking are trajectory shape features and intensity coherence features. By controlling the parameters s, t, and l, an over-complete feature representation of the trajectory can be obtained.

positions of the points along a trajectory as $p_i, i = 1, \ldots, T$, the features are

$$f_{st} = \sum_{i=1}^{L-1} \|2 * p_{s+it} - p_{s+(i-1)t} - p_{s+(i+1)t}\|.$$

where L is the index of the last point that we could pick. The appearance features are intensity coherence features based on square image patches along the trajectory. The parameters of these features are s, t as in the geometric features and the side length of the square l. Different features are obtained using different brightness constancy measures such as SSD, normalized cross-correlation, etc. By changing the s, t, l and brightness constancy measure f, we get an over-complete feature representation of each trajectory. From the range of parameters used in our experiments, we obtained about 1200 features.

These features are easy to obtain and fast to compute. Each weak learner is based on one of these features, and RankBoost will select the ones that are most useful in obtaining an accurate ranking of the training set.

2.3.3 Training the Weak Learners

We want the weak rankers have range $[0, 1]$ rather than the actual values of the ranking features. For this reason, a threshold-based weak ranking h is adopted

$$h(x) = \begin{cases} 1 & \text{if } f_i(x) > \theta \\ 0 & \text{if } f_i(x) \leq \theta \end{cases} \tag{2.5}$$

where $\theta \in \mathbb{R}$. A weak ranking is derived from a ranking feature f_i by comparing the score of f_i to a threshold θ.

A set of candidate thresholds $\{\theta_j\}_{j=1}^{J}, \theta_1 \geq \ldots \geq \theta_J$ are chosen for each weak learner. We pick the thresholds evenly in the range of a feature in the training set. For a feature f_i, the maximum and minimum value in the training set is $\max(f_i)$ and $\min(f_i)$. The thresholds for it are set to

$$\theta_j = \max(f_i) - \frac{j-1}{J-1}(\max(f_i) - \min(f_i)), \qquad j = 1, \ldots, J. \tag{2.6}$$

Based on the discussion in [22], if a weak ranker has the form as equation ((4.7)), it should be trained to maximize $|r|$ where r is defined as

$$r = \sum_{x: f_i(x) > \theta} \pi(x)$$

where $\pi(x) = \sum_{x'}(D(x', x) - D(x, x'))$.

At each boosting iteration, we exhaust all weak rankers to find the one that maximizes $|r|$ along with its associated θ value. If $r_{\max}$ is the maximal value of $|r|$, then the corresponding weight α is calculated as

$$\alpha = \frac{1}{2}\ln(\frac{1 + r_{\max}}{1 - r_{\max}}).$$

2.3.4 Training the Ranking Algorithm

Training of the trajectory ranking algorithm requires a pool of weak rankers and ground truth ranking information in the form of a distribution D over all pairs of training examples.

Given a set of trajectories obtained by a feature tracking algorithm, the weak rankers are trained at each boosting iteration as described in Section 2.3.3, based on features extracted from the given trajectories and the image sequences.

The ground truth ranking of the trajectories is obtained from the RMSE error (2.3) based on the GT labeling of the trajectories into a number of rigid motions. All trajectories belonging to the same moving object from a video sequences can be ranked by their RMSE error. Obviously, a trajectory with a smaller error should be ranked above another one with a larger error. The initial distribution D is constructed based on the relative rank of the trajectories. Actually, the exact value of $D(i, j)$ is not important, so we simply put it 0 or 1 to indicate the rank between trajectories. Let the label and RMSE error of a trajectory x_i be l_i, e_i, respectively. The ranking $D(i, j)$ between trajectories x_i and x_j is calculated as

$$D(i,j) = \begin{cases} 1 & \text{if } l_i == l_j \text{ and } e_i > e_j \\ 0 & \text{Otherwise} \end{cases} \tag{2.7}$$

With the initial distribution D, the feature set and candidate weak rankers, the RankBoost training algorithm can be applied to get the boosted ranking algorithm. The process is explained in Algorithm 1.

Algorithm 1 Training the Trajectory Ranking Algorithm

Given: A set of trajectories with their labels and RMSE errors (2.3)

1. Compute features for each trajectory as described in section 2.3.2.
2. Calculate the candidate thresholds (equation (2.6)) for each feature.
3. Build the initial distribution D by equation (2.7), and normalize it.
4. Perform T iterations of RankBoost with weak rankers (4.7). In each iteration, obtain a weak ranking h_t with threshold θ_t and weight α_t.

Output: The final ranking function: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

2.3.5 Pruning Feature Trajectories with the Ranking Algorithm

The trained ranking function can be used to rank a set of trajectories produced by any feature tracker from a video sequence. Based on the final rank, a percentage of the worst trajectories can be discarded. The pruning algorithm is summarized as Algorithm 2.

In this way, one could obtain a more refined set of trajectories. It is worth noting that in this process no a priori information – such as segmentation, or RMSE error – is needed to obtain the rank. Thus, the algorithm could be widely used for videos with or without ground truth.

Algorithm 2 Feature Pruning using RankBoost

Given: a video sequence.

1. Detect interest points in the first frame.
2. Track interest points in all frames using a feature tracker,
3. Apply the trained ranker to sort the trajectories.
4. Discard the worst $p\%$ of the trajectories.

2.4 Experiments

In this section we present an evaluation of the proposed pruning-based ranking algorithm on the 47 video sequence of the Hopkins 155 Dataset [66] that contain ground truth trajectories for all motions that are present.

2.4.1 Trajectory Generation

For the comparison purpose, five optical flow algorithms were employed to generate trajectories. They are, the Kanade-Lucas-Tomasi (KLT) algorithm [62], the Brox algorithm [6], the Classic+NL algorithm [58], the Black & Anandan (BA) algorithm [5] and the Horn & Schunck (HS) algorithm [32]. When generating trajectories, Shi-Tomasi features [55] were detected on the first frame of the video, and then the corresponding feature points in the following frames were extracted by the five optical flow algorithms. Because many motion segmentation algorithms [37, 18] require the input trajectories to have the same length, the incomplete trajectories that were stopped before the last frame were discarded. Moreover, in the spirit of fairness, if any of the five algorithms stopped the trajectory from a feature point early, the trajectories of all five algorithms starting at that point were discarded. So in the end, the trajectories produced by all algorithms for one video sequence will share the same set of starting points.

2.4.2 Dataset and Evaluation Methods

We evaluated the proposed algorithm and error measure on the Hopkins 155 Dataset. The 155 sequences in the Hopkins 155 dataset are actually extracted from 50 video sequences. Among them, there are three sequences missing the ground truth for the background motion: articulated, three-cars and arm. Since we need ground truth trajectories for all motions in the video, these three sequences were not included in the evaluation, remaining with 47 video sequences.

RMSE Error Measure. The ground truth trajectories provided by the Hopkins dataset enable us to calculate the RMSE_τ error (2.4) for each algorithm averaged over the 47 sequences. It should be mentioned that the number of motions in the ground truth may be smaller than that in the video, but most of the motions that are not in the ground truth

are not present in all frames, and they always cover small regions in the frame, so it is reasonable to ignore their impact and simply set the number of motions as given in the ground truth.

Segmentation Error Measure. Of the 50 video sequences of the Hopkins 155 dataset, Brox et.al. [7] annotated 10 car and 2 people sequences in the Moseg dataset. The first frame is always annotated to allow the evaluation of segmentation methods which could not deal with long trajectories. Based on this ground truth, any segmentation result obtained on any set of trajectories can be evaluated. This facilitates us to use the segmentation as an indirect measure of the performance of different trackers. There are five measure about segmentation proposed in [7]:

density The density of the tracked points.

overall clustering error The number of bad trajectory labels divided by the total number of labeled trajectories.

average clustering error Similar to the overall clustering error, the average clustering error is the average of the clustering error for each region separately. It gives more weight to small objects.

over-segmentation error The number of clusters that need to be merged to obtain the ground truth segmentation. This error is used to prevent obtaining a small error by producing a severe over-segmentation.

number of extracted objects The number of regions covered with a small error.

Among the five measures, the density is important for dense motion segmentation. In our experiments, the density is fixed at the very beginning since we use the Shi-Tomasi features as the starting points and we use the same starting points for all algorithms being compared. Moreover, the sparse motion segmentation methods [37] [18] that will be used for indirect evaluation ask for the number of motions in advance, so it has no meaning to measure the number of extracted objects here. In this case, the over-segmentation is hard to happen, so we do not measure the over-segmentation error either. As a result, two significant measures are adopted, the overall clustering error and the average clustering error. These two error measures will be evaluated on the 12 sequences that have a dense GT motion segmentation of the first frame.

2.4.3 Results

The RankBoost algorithm was trained on trajectories generated by the Classic+NL method from four image sequences and tested on all 47 sequences. From the pool of more than 1200 features, 150 were selected by RankBoost during training. The number of candidate thresholds J was set to 65. The ground truth ranking was obtained using the RMSE errors and trajectory labels based on the GT trajectories, as described in Section 2.3.4.

We evaluate the performance of the proposed algorithm using the two error measures described above. The first error measure is the proposed RMSE_τ error (2.4) that measures directly the quality of the trajectories based on rigid motion models. The second error measure is an indirect measure in which the obtained trajectories are segmented using a motion segmentation algorithm and the segmentation error is evaluated on the dense GT motion segmentation of the first frame. The performance using these two error measures is discussed in more details below.

RMSE Error. First, we evaluate the average RMSE_τ error (2.4) of all the trajectories. Starting with a large pool of trajectories, the pruning rate $p\%$ is changed for the RankBoost algorithm, to obtain different numbers of trajectories. For the other algorithms, the trajectory pruning can be controlled based on the strength of the Shi-Tomasi feature points. For different values of the pruning rate, the corresponding average RMSE error can be computed for the sets of trajectories obtained by the different algorithms. The average RMSE_5 error measure vs. the pruning rate from 0% to 80% is shown in figure 2.4, left.

From figure 2.4, left, one could find that the RMSE_5 error of BA, HS, Classic+NL and Brox algorithm is almost unchanged with respect to the pruning rate. This is understandable because these methods are independent of the selection of feature points (up to a point). However, the RMSE error of the KLT algorithm decreases moderately as the pruning rate is increased. This is indication that the Shi-Tomasi features have an impact on the performance of the KLT tracker. Better Shi-Tomasi features will lead to better performance of the KLT algorithm. There is no wonder that the two are always used together. Among all the algorithms, the RankBoost algorithm performs the best according to the RMSE_5 measure. When the pruning rate is 80%, its average RMSE_5 error is less than 1%, while that of the second best (the Classic+NL algorithm) is more than 9%. Also, as the prun-

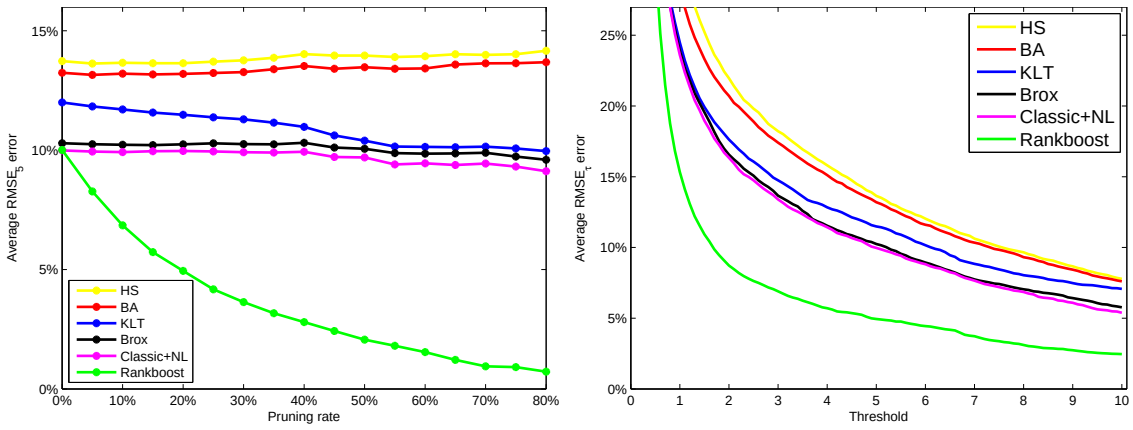


Figure 2.4: Left: the RMSE_5 error (2.4) of the trajectories sets generated by different algorithms for different pruning rates. Right: the RMSE_τ error vs. the threshold τ when the pruning rate is set to 20%. The relative rank of the algorithms is consistent with the Middlebury ranking for a large range of thresholds.

ing rate increases, the error decreases much faster than the KLT algorithm. In particular, one could see the large difference in RMSE error between the RankBoost and Classic+NL algorithm, and keep in mind that the trajectories evaluated in the RankBoost algorithm and the Classic+NL algorithm were actually the same before pruning. The reason that the RankBoost algorithm could get better results is that the trained algorithm can predict very well which trajectories might not have been tracked properly.

In figure 2.4, right, is shown the RMSE_τ error vs the threshold τ for the pruning rate of 20%. From the figure, one could find that the RankBoost method with 20% pruning outperforms the other algorithms according to the RMSE_τ measure for any value of $\tau \in [2, 10]$. Moreover, we find that the relative performance of the tracking algorithms could be measured by the order of the curves for a given threshold τ. In figure 2.4, right, the performance order is Classic+NL $>$ Brox $>$ KLT $>$ BA $>$ HS for all values of the threshold $\tau \in [2, 10]$. This order is very similar to the order of the average endpoint error and average angle error in the Middlebury dataset, which is Classic+NL $>$ Brox $>$ BA $>$ HS $>$ KLT. As mentioned above, the Shi-Tomasi corner features could boost the performance of the KLT algorithms, it is better to take the KLT out of the list, then the orders are exactly the same. This study validates the power of the proposed RMSE error. We could pick other

pruning rates than 20%, but the results will be almost the same.

Figure 2.5 shows the generated trajectories from different algorithms on a sample sequence in the Hopkins 155 dataset. The number of trajectories is kept the same and the pruning rate is 20%. It is clear the Rankboost algorithm prunes the trajectories in the path of the car and obtains better trajectories.

Segmentation Error. Another way to compare the performance of different feature tracking algorithms is by an indirect measure such as the segmentation error. The trajectories obtained by different tracking algorithms are segmented using a motion segmentation method and a measure of segmentation error is reported. Observe that this is an indirect measure since it depends on an additional step, that could introduce additional noise in the evaluation.

We use the state-of-the-art motion segmentation algorithm sparse subspace clustering (SSC) [18]. As explained before, there may be some moving objects in the first frame that are not present in all frames. In order to evaluate the segmentation error accurately, we only consider the 'major' motions in the video, so the number of motions for the motion segmentation is set to the value given by the ground truth in the Hopkins 155 dataset. We omit other motions when calculating the segmentation error.

The segmentation error is averaged over the 12 video sequences that have the dense ground truth motion segmentation of the first frame. Because of using fewer image sequences for this evaluation than for the RMSE measure (12 instead of 47), this measure is expected to be less accurate.

Furthermore, the motion segmentation imposes limitations on the maximum pruning rate that can be used. Since the rank of the measurement matrix from one rigid motion is at most four, many segmentation methods require that the number of trajectories in one motion is at least four. Because of this requirement, the maximum pruning rate that can be handled by the motion segmentation is 25%. The RMSE_τ error described above does not suffer from this limitation, allowing a pruning rate of 80% or more.

In figure 2.6 are shown the overall clustering error and the average clustering error for different pruning rates, averaged over the 12 sequences with dense GT motion segmentation. These measures were explained in Section 2.4.2.

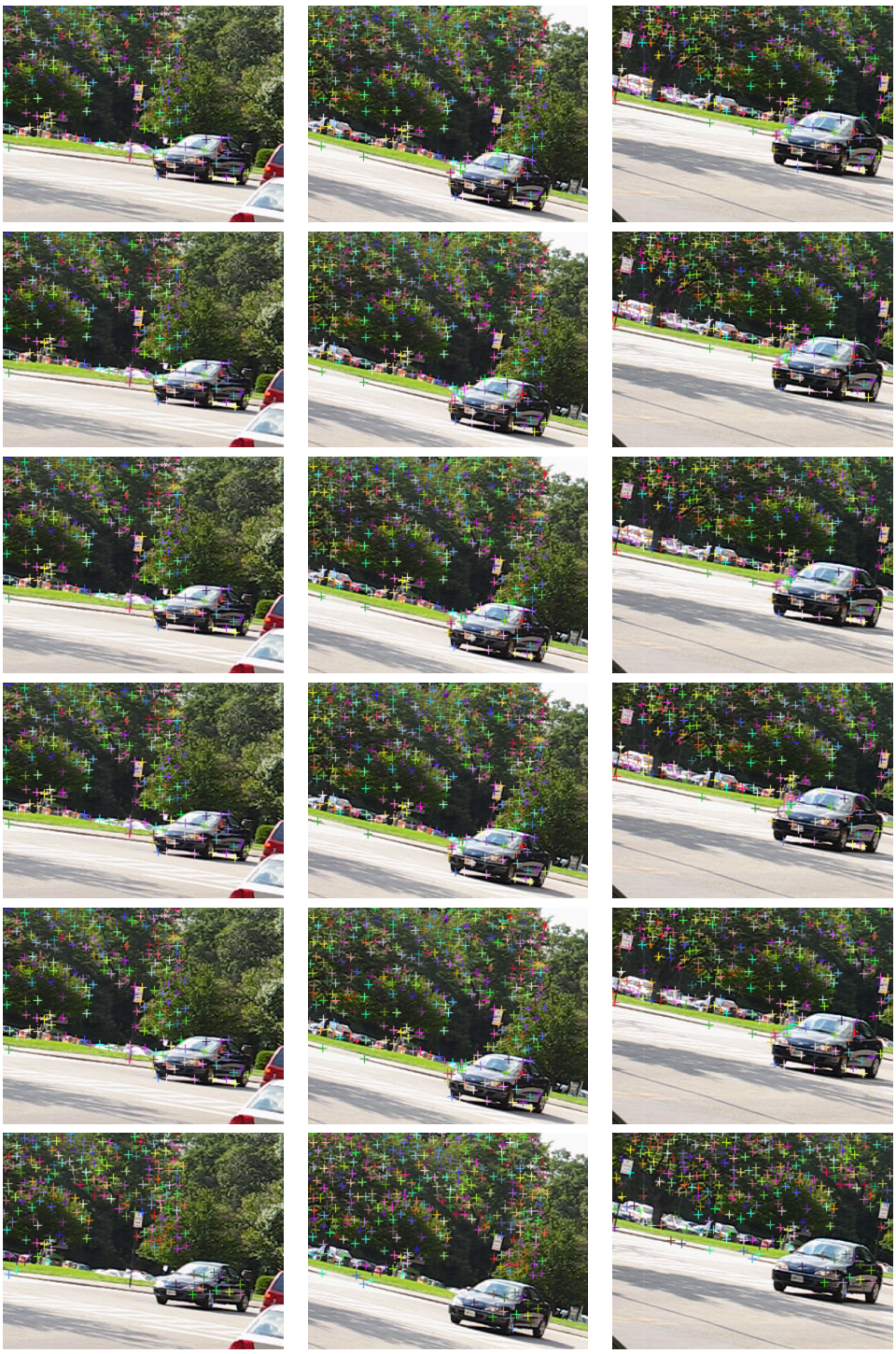


Figure 2.5: Trajectories from different tracking algorithms on the `cars7` sequence from the Hopkins 155 dataset. Row 1 to row 6: Brox, Classic+NL, BA, HS, KLT, Rankboost. The number of trajectories is kept the same and the truncation rate is 20%. From left to right: frame 1, 13, 25. The images were cropped for clarity.

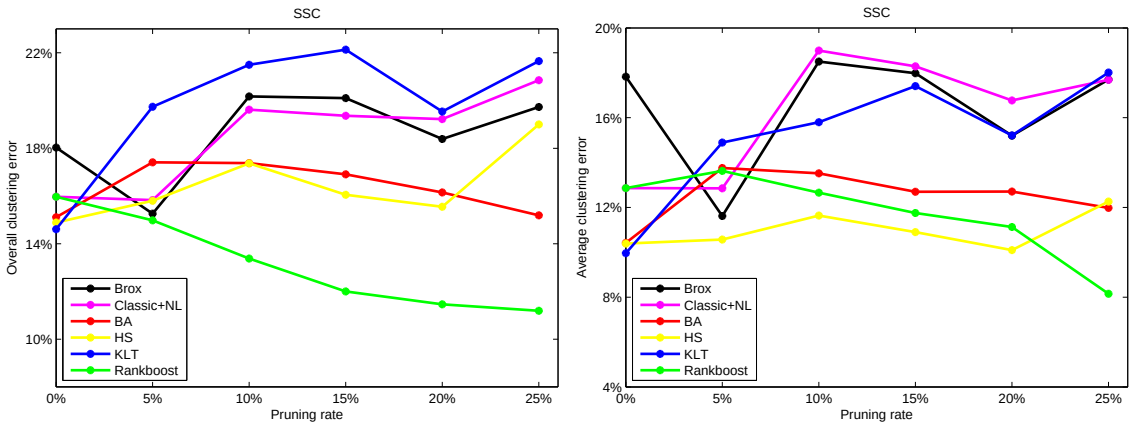


Figure 2.6: The overall clustering error (left) and average clustering error (right) from 12 annotated sequences for different pruning rates.

One could find that the RankBoost algorithm enables to reduce the overall clustering error effectively, constantly decreasing as the pruning rate is increased. Moreover, the RankBoost algorithm also constantly reduces the average clustering error when the pruning rate is at least 10%. These segmentation experiments also show that the RankBoost algorithm improves the quality of the feature trajectories.

2.5 Conclusions

In this chapter, we presented a RMSE error measure based on factorization of the affine camera model. By comparing the percentage of trajectories whose RMSE error is above a threshold for a number of tracking algorithms, we find the relative order of the algorithms is consistent with that in the Middlebury dataset. Based on this RMSE trajectory error measure and RankBoost, we introduce an algorithm for ranking the quality of feature point trajectories. One feature of the algorithm is that it does not require any priori information to rank trajectories, and it can be used to rank the trajectories obtained with any feature tracker. The comparative study has demonstrated that the proposed algorithm obtains smaller RMSE errors than other selected feature tracking algorithms after pruning. An indirect measure based on motion segmentation was also employed to evaluate the performance of different trackers. The segmentation evaluation shows again that the RankBoost algorithm can effectively improve the quality of the obtained feature point trajectories.

CHAPTER 3

MOTION SEGMENTATION BY VELOCITY CLUSTERING WITH ESTIMATION OF SUBSPACE DIMENSION

3.1 Introduction

Motion segmentation has been studied mostly in the case of the affine camera model, under which the vectors of feature points from each rigid motion lie in a subspace of dimension four or less [63], thus the motion segmentation problem can be posed as a subspace separation problem. The main difficulty in subspace separation is that it is usually hard to determine the number of subspaces and their dimension. For example, tracked feature points from a static background might lie on a 2-dimensional subspace, while points from other motions might lie on subspaces of dimension 3 or 4. Moreover, practical motion scenes usually exhibit partially dependent motions.

Many methods [18], [37], [50], [68], [75] project the feature trajectories onto a smaller dimensional space and perform clustering on the projected points. This approach not only provides computational advantages, but also imposes some sort of a spatial prior on the point trajectories.

Unlike earlier attempts to find a best projection dimension for subspace separation, this chapter proposes a new motion segmentation method to perform subspace separation for all possible dimensions. Based on this idea, we propose a motion segmentation approach which performs spectral clustering in many dimensions, and then carefully selects the result with the best separability using a novel clustering error measure.

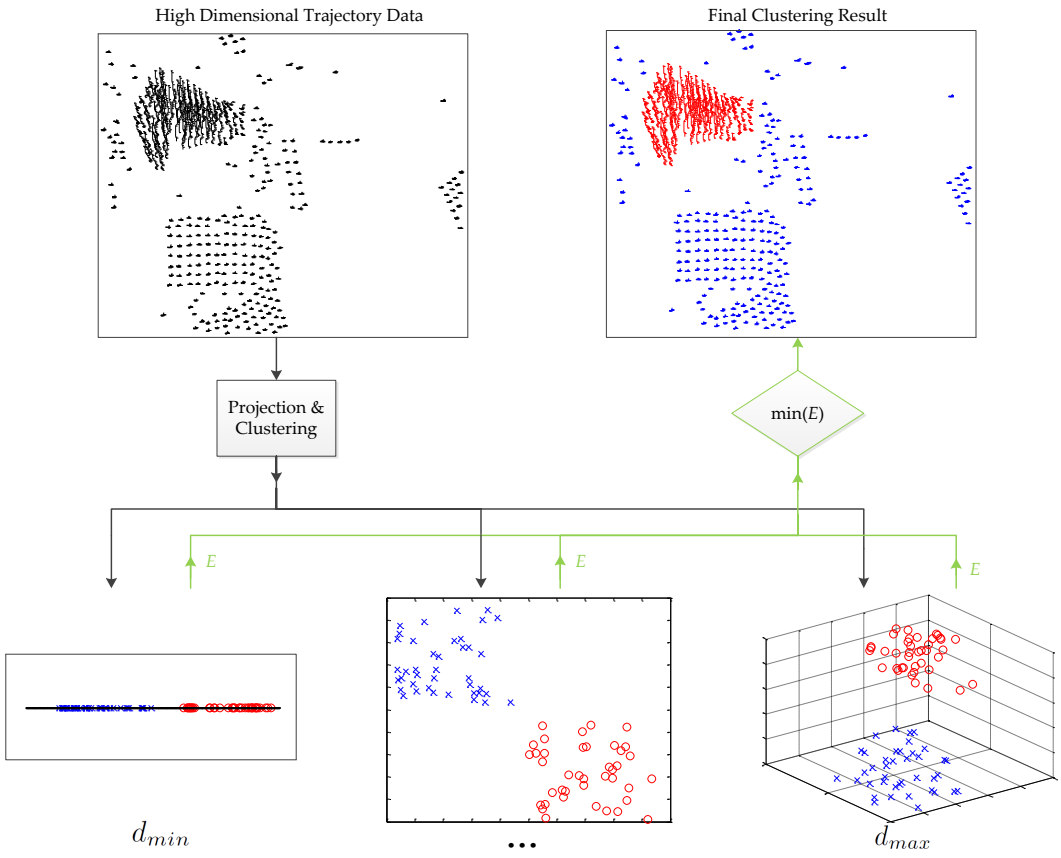


Figure 3.1: Illustration of the process of selection of the best result after performing spectral clustering in spaces of dimensions in the range $[d_{min}, d_{max}]$. The selection is based on a clustering error measure described in Section 3.2.4.

3.1.1 Related Work

Early works of multiframe 3-D motion segmentation based on matrix factorization [14], [25] find the segmentation by thresholding the entries of a similarity matrix built from the factorization of the matrix of data points. However, the thresholding process is very sensitive to noise and such methods are only provably correct when the subspaces are independent. The Generalized Principal Component Analysis (GPCA) [68] is an algebraic method for subspace separation which could deal with dependent motions, but it is not robust to data contaminated by outliers and noise. Some statistical methods, such as Agglomerative Lossy Compression (ALC) [50], RANSAC [20], Multi-Stage Learning (MSL) [57], etc, can

handle noise in the data, but their assumptions about the distribution of the noise are not optimal. In recent years, spectral clustering has become a widely used method in motion segmentation. Based on the fact that a point and its k-nearest neighbors (k-NNs) often belong to the same subspace, Local Subspace Affinity (LSA) [75], Spectral Local Best-fit Flats (SLBF) [78], Locally Linear Manifold Clustering (LLMC) [27] use the angle or distance between a point and the subspace fitted through the point and its k-NNs to construct the affinity measure for spectral clustering. However, the neighbors of a point could belong to different spaces, especially when close to the intersection of two subspaces. Also, the selected neighbors may not span the underlying subspace. The spectral clustering (SC) method [37], which uses the angular information between trajectories as affinity, is simple and efficient, but its criterion to select the best subspace dimension is noise-sensitive. More recently, some approaches such as Spectral Curvature Clustering (SCC) [12], Sparse Subspace Clustering (SSC) [18], and Low-Rank Representation (LRR) [41], use the so-called *sparsity* information as the affinity measure. Optimization is always involved in these methods, which makes them computationally expensive.

3.1.2 Our Contributions

In this work, we provide two main contributions. First, we use the velocity vector as a preprocessing step to reduce the influence of the errors accumulated during feature point tracking. This step proves to be very important for improving performance. Second, we present a method for estimating the optimal projection dimension for spectral clustering. We use the angular information between the points proposed in SC [37] to build the affinity matrix. Compared to the SC algorithm, the proposed method presents a different strategy for selecting the best subspace dimension. The SC finds the best dimension before performing the spectral clustering, and the dimension is determined by the so called *relative gap* which is related to the eigenvalues of a Laplacian matrix L . However, when the noise level is large, the relative gap is not very effective. Instead, our method performs spectral clustering after projecting to each of the possible dimensions in a range $[d_{min}, d_{max}]$, and then selects the best result based on a novel clustering error measure. The advantage of the proposed strategy is that the performance is much more robust to data corrupted by noise. Moreover, the complexity of the resulting algorithm remains low as long as the number

of motions is small. When applied to the motion segmentation data from the Hopkins155 database [66], the proposed method is competitive with the current state-of-the-art methods both in terms of segmentation accuracy and computational speed.

3.2 Motion Segmentation by Spectral Clustering

In this chapter, we only focus on the problem of segmentation of tracked feature point trajectories. The goal is to find labels for all trajectories, to group them according to their corresponding motions. Also, we assume that the number of different motions is already known.

3.2.1 Noise Reduction using Velocity Vectors

Methods for reducing the noise level in the trajectory data is an area that did not receive enough attention in previous work. Noise is an inevitable by-product of feature tracking. Tracking errors are introduced with each new frame, due to factors such as aliasing, non-constant brightness, lack of texture, occlusion, and so on. These errors tend to accumulate and the total tracking error tends to grow as the number of frames increases.

In order to reduce the effect of the accumulated error in the motion segmentation, we use the velocity vector to characterize the trajectories, which is defined by

$$[x^1 - x^2, y^1 - y^2, \dots x^{F-1} - x^F, y^{F-1} - y^F, x^i, y^i]^T, i \in [1, \cdots, F] \tag{3.1}$$

With the exception of the last two rows, the entries of the other rows are replaced with the corresponding velocities. In the last two rows, the feature locations of the i-th frame are kept. The selection of i is not crucial. In this chapter, we use $i = F$ but we could as well use $i = 1$ for example. The advantage is that the velocities in each frame contain only the tracking error from the previous frame to the current frame, and not the error accumulated from the starting frame. A similar velocity has been used to measure the distance between trajectories for motion segmentation in [7].

It is easy to see that when the measurement matrix W' is built from the velocity vectors, no information is lost since the original measurement matrix W can be recovered from W' by simple row operations. Because of this, the ranks of W and W' are the same. In other

words, the subspace clustering problem has not been changed. However, even though the velocity matrix differs from the original measurement matrix only by row operations, the subspace projections are different because these row operations cannot be represented by a rotation matrix.

Table 3.1: The SSE and variance of the distances from the projected points to the fitted subspaces in 3D for a synthetic experiment. The projected points were generated from trajectories with different noise levels.

	SSE	Variance
Distance Vector (No noise added)	0	0
Velocity Vector (No noise added)	0	0
Distance Vector (SNR = 10)	0.256e-5	0.0011e-5
Velocity Vector (SNR = 10)	0.106e-5	0.0004e-5
Distance Vector (SNR = 5)	1.058e-5	0.005e-5
Velocity Vector (SNR = 5)	0.208e-5	0.001e-5

The noise reduction effect of using the velocity vector can be well observed in a synthetic experiment. For this purpose, 242 synthetic trajectories of length 20 were generated for two different motions, perfectly following the affine camera model. The starting feature points were randomly chosen in the first frame, and different levels of Gaussian tracking errors were introduced based on the displacement of feature points. If denote the tracker as f, and noise as n, to a point p_i in frame i, the tracked point in the next frame would be $p_{i+1} = f(p_i) + n$.

The trajectories were projected to a 3D subspace by truncated SVD. A plane was fitted in a least squares sense to the projected points of each motion. The sum of squared error (SSE) and variance of the distances from projected points to the fitted planes are shown in table 3.1. One could see that by using velocity vectors the noise is reduced, and the reduction is greater when the tracking errors are larger. Since the projected points obtained by velocity clustering are closer to satisfying the planarity assumption, it should be expected that the segmentation results would also be better.

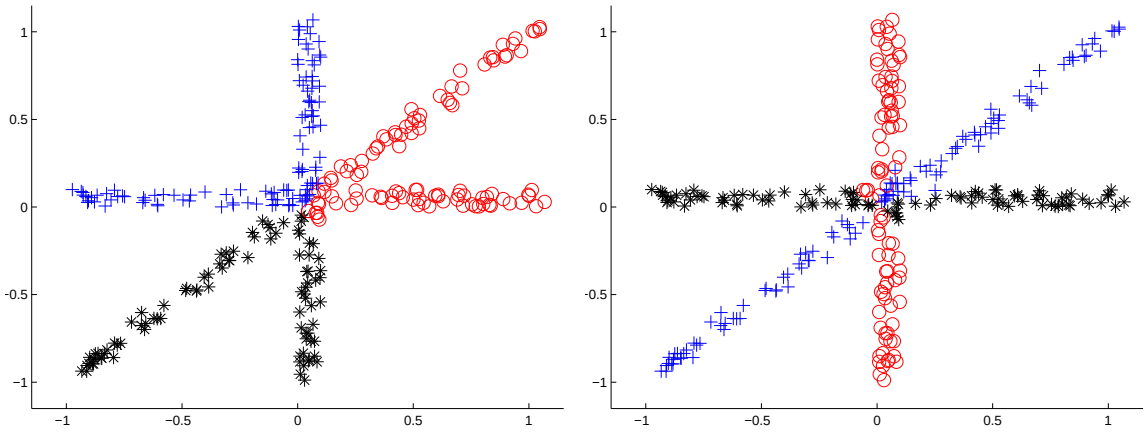


Figure 3.2: Spectral clustering of lines with a distance-based affinity mixes points from different subspaces (left), while the angle-based affinity (3.2) separates them very well (right).

3.2.2 Spectral Clustering of Subspaces

Spectral clustering [53], [48] is a popular technique for solving motion segmentation problems [73], [49], [75], [78], [27], [18], [41], [12]. One challenge in applying spectral clustering is the construction of a good affinity matrix. Two points that lie in two different subspaces and are near the intersection of the subspaces may be close to each other. Conversely, a pair of points in the same subspace could be far from each other. As a consequence, one cannot use the typical distance-based affinity.

SC [37] proposes an affinity measure based on the angle between two vectors, defined by

$$A_{ij} = \left(\frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}\right)^{2\alpha}, i \neq j, \alpha \in \mathbb{N} \tag{3.2}$$

where x_i, x_j are two vectors. The parameter $\alpha > 1$ is used to increase the separation and should be tuned according to the noise level. It has been proved [37] that the proposed affinity measure (3.2) guarantees that each point x_i has a higher connection with its own group than the others. Figure 3.2 shows the power of angle-based affinity over the distance-based affinity in clustering 1D subspaces in 2D. This chapter also uses the angular information to build the affinity matrix. While SC [37] suggests to set $\alpha = 4$ for motion segmentation, in the experiments of section 3.4, we find that $\alpha = 2$ could produce better results for our algorithm.

3.2.3 Best Subspace Dimension

Most motion segmentation methods usually require the projection to a low dimensional space where the clustering is performed. The dimension of this projection space has a large impact on the speed and accuracy of the final result. GPCA [68] suggests to project trajectories onto a 5-dimensional space. However, five dimensions are not sufficient to complex scenes, such as scenes with articulated or nonrigid motions. Motivated by compressive sensing [17], ALC [50] chooses to use the sparsity-preserving dimension

$$d_{sp} = \min_{d \geq 2D \log(2F/d)} d$$

for D-dimensional subspaces (with $D = 4$ for motion segmentation). SC [37] wants the intersection of different subspaces to have minimal dimension and proposes to set dimension $d = kD + 1$, where k is the number of motions and $1 \leq D \leq 4$ for motion segmentation; the d used for clustering is searched in range $[k + 1, 4k + 1]$ by some relative gap.

The main difficulty for selecting the best subspace dimension is that the dimension of one affine subspace is not fixed. If one tries to find the correct dimension by setting a threshold of noise, this scheme will not work well because different scenes may have different thresholds.

The search strategy in SC [37] is innovative, but the range of possible dimensions that are searched is a parameter that needs to be tuned. Moreover, the criterion to select the best dimension in SC [37] is related to the noise level, and is not optimal in some scenarios, as we will see in experiments.

In this chapter, we don't look for the best subspace dimension directly. Instead, we employ an exhaustive strategy. Since the best dimension is unknown and hard to determine, our method performs clustering after projecting to spaces of all possible dimensions, then the best result is chosen by a clustering measure. Based on this idea, finding the best subspace dimension is not necessary here. What we need to do is to find a bound on the possible dimensions.

The dimension of one affine subspace S is not fixed but is bounded by

$$2 \leq \dim(S) \leq 4.$$

If there are k linear affine subspaces in general position embedded in space S_k, we would expect

$$2k \leq \dim(S_k) \leq 4k.$$

This is the range of space dimensions that will be used in our method. The best dimension will be determined using the clustering error measure defined in the next section.

3.2.4 Motion Error Measure

When the spectral clustering is performed in the selected spaces, a number of results will be obtained. A question is raised naturally: how to select the best one? In this paper we investigate two types of estimators of the segmentation error, both based on a RMSE error measure for each trajectory.

The Tomasi-Kanade factorization [63] allows us to write the registered matrix $\tilde{W}$ in equation (1.8) as

$$\tilde{W} = \tilde{M}\tilde{S}$$

where $\tilde{M}$ is a $2F \times 3$ matrix and $\tilde{S}$ is a $3 \times P$ matrix. There is an inherent ambiguity in $\tilde{M}$ and $\tilde{S}$ but we will show that it is irrelevant for the error measure.

Any registered trajectory $\tilde{t}$ in $\tilde{W}$ will have a corresponding point $\tilde{P} \in \mathbb{R}^3$ obtained by least squares:

$$\tilde{P} = \underset{\tilde{P}}{\operatorname{argmin}} \|\tilde{t} - \tilde{M}\tilde{P}\|^2.$$

We define the RMSE error of $\tilde{t}$ as

$$\mathrm{RMSE}_{\tilde{W}}(\tilde{t}) = \sqrt{\frac{\min_{\tilde{P}} \|\tilde{t} - \tilde{M}\tilde{P}\|^2}{F}} \tag{3.3}$$

The RMSE error is measured in pixels and can be viewed as the tracking error for one trajectory.

Remark 3.2.1. *The $RMSE_{\tilde{W}}(\tilde{t})$ is invariant to the choice of $\tilde{M}$ and $\tilde{S}$ in the decomposition $\tilde{W} = \tilde{M}\tilde{S}$.*

Proof. To any 3×3 invertible matrix A, $\tilde{W} = \tilde{M}AA^{-1}\tilde{S}$. It can be easily verified that $\mathrm{RMSE}_{\tilde{W}}(\tilde{t})$ in equation (3.3) does not change when $\tilde{M}$ is multiplied by an invertible matrix

A. Moreover, any decomposition $\tilde{W} = \tilde{M}'\tilde{S}'$ has $\tilde{M}' = \tilde{M}A$ for some invertible matrix A.

■

Given a labeling L of the trajectories, obtain for each label l the registered measurement matrix $\tilde{W}^l$ containing all trajectories with label l. Based on $\tilde{W}^l$ we define two types of estimators of the segmentation error.

The first type is just the sum of the RMSE errors of all registered trajectories based on their corresponding motion matrices

$$E(L) = \sum_{l=1}^{k} \sum_{i,L(i)=l} \text{RMSE}_{\tilde{W}^l}(\tilde{t}_i). \tag{3.4}$$

The second type makes the contribution of each registered trajectory comparing to a threshold τ

$$E_\tau(L) = \sum_{l=1}^{k} \sum_{i,L(i)=l} I(\text{RMSE}_{\tilde{W}^l}(\tilde{t}_i) \geq \tau). \tag{3.5}$$

where $I(\cdot)$ is the indicator function taking on value 1 if its argument is true or 0 otherwise.

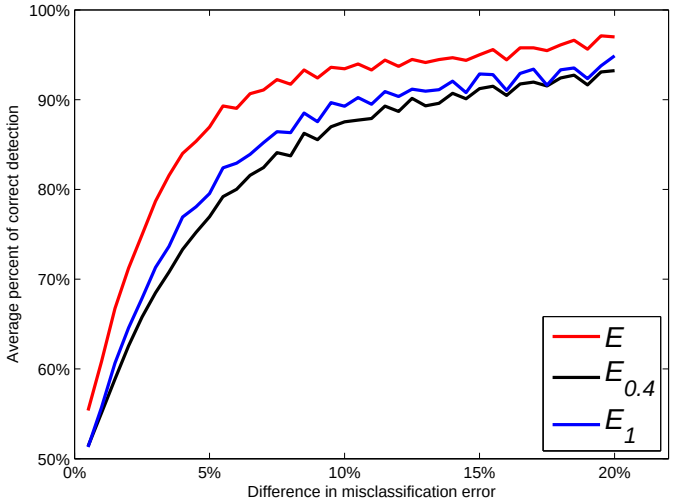


Figure 3.3: The average percentage of times the proposed error estimators find the better segmentation out of two segmentations vs. their difference in misclassification rates for sequences with 3 motions in the Hopkins 155 dataset. If one segmentation is much better than the other, it will be found most of the time.

In a perfect segmentation result, each trajectory would have a small RMSE error because of the affine camera model, resulting in a small clustering error $E(L)$ and $E_\tau(L)$.

A number of segmentation results can be obtained by projecting the original trajectories to spaces of different dimensions and performing clustering in those spaces. The problem is how to select from the obtained segmentations the one with the smallest error. For that we can use an estimator that correlates well with the segmentation error.

We propose to use the measures $E(L)$ and $E_\tau(L)$ to rank the obtained segmentations. We evaluated the capability of these error measures to find the better one out of two segmentations on the sequences with 3 motions in the Hopkins 155 dataset (See section 3.4). It is expected that when one segmentation is much better than the other (i.e. the error difference is large), the better segmentation should be found more often. Different segmentations were obtained in this way: for one sequence, a random set of $p\%$ of trajectories ($p \leq 50$ is a random number) were assigned random labels, while the labels of the remaining trajectories were untouched. 500 sets of segmentation were generated for each sequence (25000 segmentations in total for all sequences). At last, we calculated the difference in misclassification rate and the average correct detection rate shown in Figure 3.3. One can see that if one segmentation is much better than the other, it will be found most of the time. Also, $E(L)$ always outperforms the other two estimators. Thus in this paper, $E(L)$ is adopted.

3.3 Complete Procedure

Dimension Reduction. Dimension reduction seems to be a standard procedure for motion segmentation by spectral clustering in [37], [50], [68]. It can improve the computational tractability without adversely affecting the quality of the segmentation, since in general the projection onto an arbitrary d-dimensional space preserves the multi-subspace structure of data lying on subspaces with dimensionality strictly less than d. There are two different strategies in dimension reduction: the random sampling [12], [18] and the truncated SVD [14], [37], [57]. This paper uses the latter method for dimension reduction from $W' \in \mathbb{R}^{2F\times P}$ to $X = [x_1, ..., x_P]^T \in \mathbb{R}^{D\times P}$ in our framework, where D is the dimension of the subspace. The truncated SVD is related to the factorization-based methods [13], [34], which use the SVD, $W = U\Sigma V^T$, to obtain a shape interaction matrix $Q = VV^T$. In order to deal with the noise and dependencies, we use the truncated SVD of the velocity measurement matrix, $W' \approx U_D\Sigma_D V_D^T$.

Details of Spectral Clustering. After the projection for dimension reduction, the spectral clustering method is applied to obtain the clustering result.

The affinity matrix is constructed using the angular affinity metric in equation (3.2). In fact, the affinity matrix can be easily calculated as $Q = (\tilde{V}_D \tilde{V}_D^T)^{2\alpha}$, where $\tilde{V}_D$ is the V_D with normalized rows. This normalization ensures that only the angular information is taken into account.

From the affinity matrix, the corresponding Laplacian matrix L is obtained. Then the k largest eigenvectors of L are found, where k is the number of clusters. A matrix A is formed by stacking the k eigenvectors in columns. Finally, the segmentation of the trajectories follows by applying K-means clustering to the rows of $\tilde{A}$, which is obtained by normalizing the rows of A .

Algorithm 3 Velocity Clustering with Estimation of Subspace Dimension (VC)

Input: The measurement matrix $W = [t_1, t_2, \ldots, t_P] \in \mathbb{R}^{2F \times P}$ whose columns are point trajectories, and the number of clusters k.

Preprocessing: Build the velocity measurement matrix W' by row transformations of W given by equation (3.1).

for $D = d_{min}$ **to** d_{max} **do**

1. Perform SVD: $W' = U\Sigma V^T$
2. Build the N-by-D data matrix

$$X_D = [v_1, ..., v_D]$$

where v_i is the i-th column of V.

3. Apply spectral clustering to the N points in X_D using the affinity measure (3.2).
4. Compute the clustering error E_D of the segmentation result using equation (3.4).

end for

Output: The segmentation result with the smallest error E_D.

Selection of the best result. According to section 3.2.3, to ensure that the best result is not missed, an exhaustive search strategy is employed. Let $d_{min} = 2k$ and $d_{max} = 4k$ be the minimal and maximal subspace dimensions, motion segmentation is performed in spaces with all dimensions D in the range $D \in [d_{min}, d_{max}]$. Then the best result is selected among all results based on the smallest clustering error (3.4) or (3.5). The whole procedure

is illustrated in Figure 3.1 and described in Algorithm 6.

3.4 Experiments

We have tested our algorithm on the image sequences from the Hopkins 155 database, as well as several other state-of-the-art algorithms: ALC [50], SC [37] and SSC [18]. For each algorithm on each sequence, we recorded the misclassification rate defined as

$$\text{Misclassification Rate} = \frac{\#\text{ of misclassified points}}{\text{total }\#\text{ of points}} \tag{3.6}$$

The parameter setting in our method are $\alpha = 2$, $d_{min} = 2k$, $d_{max} = 4k$, and the locations of the last frame are kept to build the velocity matrix. The results on sequences with 2, 3 motions and the whole dataset are presented in Table 3.2 and compared with the three state-of-the-art and baseline methods. We also show in the table the results of the algorithm with fixed subspace dimension $D = 4k$ as well as results without using the velocity preprocessing step.

One could see that by using the velocity for clustering the misclassification rate decreases by about 0.8% while by using the clustering error measure to decide the best segmentation the error decreases from 4.91% to 0.99%. Thus the clustering error measure has a large impact in the spectral clustering performance while the velocity clustering has a smaller but also important impact.

Compared to other motion segmentation algorithms, our approach outperforms for the 3 motion sequences and for all the sequences combined and is outperformed on the two motion sequences by SC [37] and SSC [18]. We achieve an overall misclassification rate of 1.10% for 3 motions, around half of the best reported result (SC [37]); an overall error of 0.96% for 2 motions, coming close to the best performing SSC [18]; and an overall error of 0.99% for the whole database, which is better than the other methods. Our method always obtains good results for checkerboard sequences which have the most complicated scenes (including both translation and rotation motions) in the dataset.

The performance on the articulated sequences with 3 motions is worse than the SC, possibly because these sequences don't obey the rigid motion model and thus the RMSE measure might not be accurate. On the other hand, when the motions follow the rigid

model, the RMSE measure helps obtain very good results. This is clearly visible in the three motion checkerboard sequences, where our algorithm obtains errors less than half of the other algorithms.

Table 3.2: Misclassification rate (in percent) for sequences of full trajectories in the Hopkins 155 dataset (Subscript $4k$ means using fixed dimension $4k$ instead of dimension search, and superscript $*$ means not using velocity for clustering).

Method	ALC	SC	SSC	VC^*_{4k}	VC^*	VC_{4k}	VC
Checkerboard (2 motion)							
Average	1.55	0.85	1.12	2.07	1.38	1.38	**0.67**
Median	0.29	0.00	0.00	0.30	0.00	0.00	0.00
Traffic (2 motion)							
Average	1.59	0.90	**0.02**	6.87	1.35	8.25	0.99
Median	1.17	0.00	0.00	1.33	0.30	1.09	0.22
Articulated (2 motion)							
Average	10.70	1.71	**0.62**	6.02	2.56	2.46	2.94
Median	0.95	0.00	0.00	0.99	0.88	0.88	0.88
All (2 motion)							
Average	2.40	0.94	**0.82**	3.67	1.48	3.25	0.96
Median	0.43	0.00	0.00	0.51	0.00	0.00	0.00
Checkerboard (3 motion)							
Average	5.20	2.15	2.97	4.38	1.06	2.28	**0.74**
Median	0.67	0.47	0.27	1.37	0.58	0.51	0.21
Traffic (3 motion)							
Average	7.75	1.35	**0.58**	27.80	8.22	19.21	1.13
Median	0.49	0.19	0.00	32.27	1.42	28.28	0.21
Articulated (3 motion)							
Average	21.08	4.26	**1.42**	6.18	6.18	18.95	5.65
Median	21.08	4.26	0.00	6.18	6.18	18.95	5.65
All (3 motion)							
Average	6.69	2.11	2.45	9.17	2.78	6.62	**1.10**
Median	0.67	0.37	0.20	1.99	0.67	0.85	0.22
All sequences combined							
Average	3.37	1.20	1.24	4.91	1.78	4.01	**0.99**
Median	0.49	0.00	0.00	0.57	0.00	0.24	0.00

From the cumulative distributions in Figure 3.4, we see that for 2 motions, our method is comparable to the best method SSC; and for 3 motions, our method outperforms all others. Moreover, the largest error of our method for 3 motions is about 10%, while that

of the other methods is around 40%.

Table 3.3 shows that the average computing time per sequence (obtained on a 2.66GHz Core 2 Duo computer with Matlab on Linux) for sequences with 2 motions is less than 1 second, while that for sequences with 3 motions is less than 2 seconds. In comparison to other methods, our method is much faster than ALC and SSC, but slightly slower than SC.

Table 3.3: Average computing time for sequences in the Hopkins 155 database.

	ALC	SC	SSC	Our Method
2 motions	7.85m	0.53s	2.27m	0.72s
3 motions	16.77m	1.34s	4.08m	1.81s

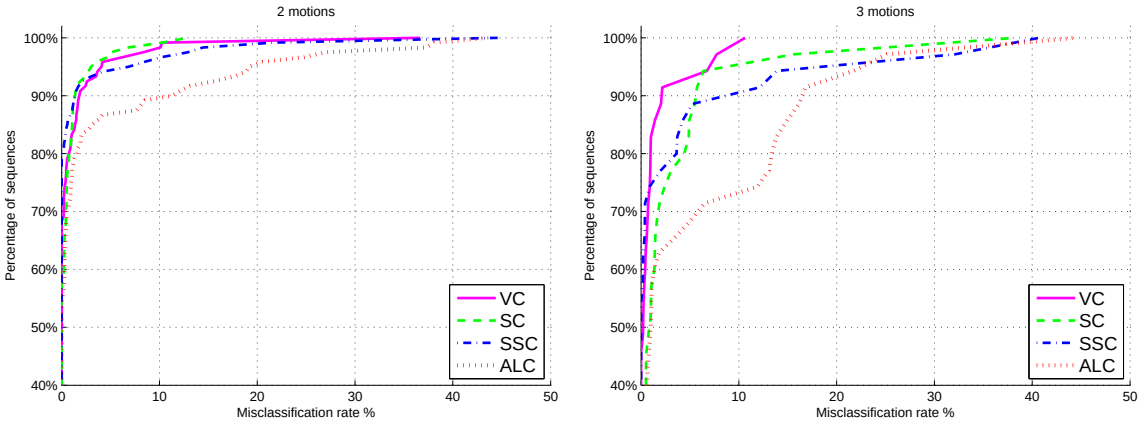


Figure 3.4: The cumulative distribution of the misclassification rate for two and three motions in the Hopkins 155 database.

3.5 Conclusion

In this paper, we presented a method for segmenting moving objects using spectral clustering. The method uses the velocity vectors as the input for clustering, which is more robust to accumulated errors, and then applies spectral clustering in all possible subspace dimensions. The final segmentation is selected from the obtained results using a novel clustering error measure. Our evaluation on the Hopkins 155 database shows that the method is competitive with current state-of-the-art methods, both in terms of overall

performance and computational speed. The algorithm has been shown to be robust to different types of scenes and motions present in the Hopkins 155 database, while remaining very efficient in computation time.

CHAPTER 4

A RANKING BASED METHOD FOR MOTION SEGMENTATION

4.1 Introduction

Learning based method have seen great popularity in recent years due to their flexibility and proved robustness on large datasets. Many applications of learning in computer vision and medical imaging are for detecting and segmenting objects in 2D or 3D images.

In this work we study a learning based approach to sparse motion segmentation. Given a number of feature points tracked in all frames of an image sequence, the sparse motion segmentation problem is to group the feature point trajectories into a number of clusters based on their common motion.

Many works [18, 37, 41, 70, 75] address this problem using spectral clustering, but the approaches depend on a number of parameters that have a great influence in the quality of the obtained segmentations.

This chapter proposes a learning based approach to sparse motion segmentation and brings the following contributions:

1. It proposes a novel method for training a ranking function based on the newly introduced Feature Selection with Annealing (FSA) algorithm. The obtained ranking function depends nonlinearly on a small number of selected features, which helps in accuracy and generalization power.

2. It introduces two types of features for ranking motion segmentations. The first type are likelihood features that evaluate how well the points of each motion lie in a low dimensional subspace. The second type are prior features that measure how compactly clustered together are the points of each motion label.

3. It introduces a ranking-based method for motion segmentation. A number of candidate segmentations are generated using different segmentation parameters. The obtained segmentations are ranked by the trained ranking function and the best ranked segmentation is reported as the final result.

The proposed motion segmentation method is compared with other state of the art methods on the Hopkins 155 dataset. The experiments show that the proposed FSA-based ranking outperforms RankBoost [22] in learning a ranking function from the same set of features. Moreover the experiments show that the proposed motion segmentation method is competitive with the other state of the art motion segmentation methods in terms of average segmentation error.

4.2 Related Work

Aside from the works in sparse motion segmentation that were already discussed, there are a number of works in image segmentation that use different ways of ranking segmentation results.

A ranking function trained by Rankboost has been used in [77] to compare object segmentations parameterized by a PCA model. The simplex algorithm was used to find the PCA parameters of the final segmentation result. The ranker was trained based on Haar features extracted from the image. In contrast, our work ranks arbitrary segmentations of the feature point trajectories, not parameterized by a PCA model. The features are also different, namely we use appearance (likelihood) features and prior features. Finally we use a novel ranking algorithm based on FSA that outperforms Rankboost.

A ranking based method for face alignment was presented in [24], using Gradient Boosted Regression Trees, which is a Random Forest trained for regression. Similar to [77], it uses only appearance features in constructing the ranking function.

A regression based approach to ranking image segmentations was used in [9, 10]. Instead of using a ranking approach, the authors directly try to regress the overlap measure of each segmentation with the ground truth. The features used can be regarded as appearance features and prior features. Some of their prior features could in principle be used in our motion segmentation application. The regressor used was either a linear regressor or a Random forest in [10] and based on support vector regression in [9].

4.3 The Feature Selection with Annealing Algorithm

Let $\mathbf{x}_i \in \mathbb{R}^M, i = \overline{1, N}$ be N training examples in an M-dimensional instance space.

The Feature Selection with Annealing algorithm is designed for optimizing a loss function $L(\beta)$ defined on these training examples. We assume that each variable i has associated a coefficient β_i and $\beta_i = 0$ if $L(\beta)$ does not depend on variable i.

We are interested in the constrained optimization of the loss function

$$\beta = \underset{|\{i,\beta_i \neq 0\}| \leq s}{\operatorname{argmin}} L(\beta) \tag{4.1}$$

where the number s of relevant features is a given parameter. Further assume that the loss function $L(\beta)$ is differentiable with respect to β.

The FSA approach starts with an initial value of the parameters β, usually $\beta = 0$, and alternates two basic steps:

- one step of gradient descent parameter updates towards minimizing the loss $L(\beta)$

$$\beta \leftarrow \beta - \eta \frac{\partial L(\beta)}{\partial \beta} \tag{4.2}$$

- one variable selection step that removes some variables according to a criterion $c_j(\beta), j = \overline{1, M}$.

In this chapter we will use the criterion $c_j(\beta) = \|\beta_j\|^2, j = \overline{1, M}$ for which there are theoretical guarantees of convergence and variable selection consistency for classification.

Usually many variables are removed at each iteration, keeping only a number M_e of variables that have the largest values of the criterion $c_j(\beta)$. The number M_e of variables that are kept after each iteration $e = \overline{1, N^{iter}}$ is similar to an annealing schedule.

Through this schedule, the constraint after iteration e is $|\{i, \beta_i \neq 0\}| \leq M_e$, thus the constraint is gradually tightened and after a large number of iterations we reach $|\{i, \beta_i \neq 0\}| \leq s$. This way we obtain a suboptimal solution to the constrained problem (4.1).

In this work we use an inverse schedule $M_e, e = \overline{1, N^{iter}}$ with a parameter v or $\mu = \frac{M}{sv}$

$$M_e = \max(s, \frac{M}{1 + e\mu} - \frac{1}{N^{iter}}). \tag{4.3}$$

Because the M_e quickly decreases after the first iteration, the total computation time of the algorithm is about $N(M + sv \ln \frac{M}{s} + sN^{iter})$ operations, where N is the number of training examples, assuming one iteration with M variables takes MN operations. Thus when $N^{iter} = 500$, the whole algorithm takes about $2MN$ operations.

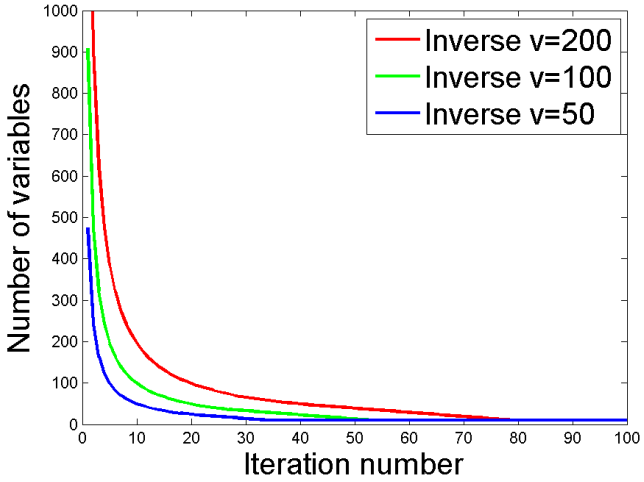


Figure 4.1: The value of the number of features M_e vs iteration e for three annealing schedules, where $M = 10,000, s = 10$.

In Figure 4.1 are shown the value of M_e vs the iteration number e for three schedules $v = 50, 100$ and 200, where $M = 10,000$ and $s = 10$. We also experimented with an exponential schedule $M_e = \max(s, M\xi^e)$ with a parameter $0 < \xi < 1$, but observed that the inverse schedule worked better for the same computational expense.

The FSA algorithm is summarized in Algorithm 4. In practice, the gradient update (4.2) can be replaced by one epoch of stochastic gradient updates.

The performance of the FSA algorithm depends on the following parameters:

- Gradient learning rate η, which can be arbitrarily small provided that the number of iterations is large enough. If η is too large, the coefficients β_i will not converge.
- Parameter v or μ for controlling the number of variables left at each iteration. Parameter μ should have a value proportional to the parameter η so that if the learning rate is small, the variables are removed at a slower pace.
- Number of iterations N^{iter}, large enough to obtain in the end a desired number s of variables and to insure the parameters have converged to a desired tolerance.

Algorithm 4 Feature Selection with Annealing (FSA)

Input: Training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

Output: Trained model parameters β.

1: Initialize $\beta = 0$.
2: **for** e=1 to N^{iter} **do**
3: Update $\beta \leftarrow \beta - \eta \frac{\partial L(\beta)}{\partial \beta}$
4: Compute $c_j(\beta), j = 1, ..., M$
5: Keep the M_e variables with highest $c_j(\beta)$ and renumber them $1, ..., M_e$.
6: Set $M = M_e$
7: **end for**

Experiments showed that the FSA algorithm obtains good and stable results for a large range of values of these parameters, but are omitted for lack of space.

4.4 Ranking Using FSA

Let $\mathbf{x}_i \in \mathbb{R}^M$ be the training instances and $r_{ij} \in [0, 1]$ be the true rankings between observations $\mathbf{x}_i, \mathbf{x}_j$, with $(i, j) \in C \subset \{1, ..., N\} \times \{1, ..., N\}$. Observe that the true ranking might not be given for all pairs $(i, j) \in \{1, ..., N\} \times \{1, ..., N\}$ but only for the subset C. A criterion (e.g. an error measure) can be used to compare instances $\mathbf{x}_i$ and $\mathbf{x}_j$ and generate the true rankings $r_{ij} \in [0, 1]$, which can be for example 0 is $\mathbf{x}_i$ is "better" than $\mathbf{x}_j$, 0.5 if they are "equally good" and 1 if $\mathbf{x}_i$ is "worse" than $\mathbf{x}_j$.

Training can be achieved by finding a ranking function $h_\beta(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}$ specified by a parameter vector β such that $h_\beta(\mathbf{x}_i) - h_\beta(\mathbf{x}_j)$ agrees as much as possible with the true rankings r_{ij}.

There are different criteria that could be optimized to measure this degree of agreement, but we will use the differentiable criterion from [8]

$$\begin{aligned} L(\beta) = &- \sum_{(i,j)\in C} r_{ij}(h_\beta(\mathbf{x}_i) - h_\beta(\mathbf{x}_j)) + \\ &+ \sum_{(i,j)\in C} \ln(1 + e^{h_\beta(\mathbf{x}_i) - h_\beta(\mathbf{x}_j)}) + \sum_{i=1}^{M} \rho(\beta_i) \end{aligned} \tag{4.4}$$

where we added the prior term $\sum_{i=1}^M \rho(\beta_i)$ that helps improve the generalization ability.

4.4.1 Piecewise Linear Learners for Nonlinearity

A simple way to introduce nonlinear dependence on the feature values is by specifying the ranking function as a sum of univariate functions

$$h_\beta(\mathbf{x}) = \sum_{k=1}^{M} h_k(x_k, \beta_k)$$

where each $h_k(x_k)$ is a nonlinear function that depends only on the variable x_k of the feature vector $\mathbf{x} = (x_1, ..., x_M)$.

In this chapter we will use piecewise linear functions for the nonlinear functions $h_k(x_k)$.

To simplify notation we will drop the index k in the rest of this section and we will implicitly assume that we are working with the variable x_k of the feature vector $\mathbf{x} = (x_1, ..., x_M)$.

A piecewise linear (PL) function $h(x) : \mathbb{R} \to \mathbb{R}$ is defined based on the range $[x^{min}, x^{max}]$ of the variable x and a predefined number B of bins.

Let $b = (x^{max} - x^{min})/B$ be the bin length.

For each value x, the learner finds the bin index $j(x) = \big[(x - x^{min})/b\big] \in \{0, ..., B-1\}$ and the relative position in the bin $\alpha(x) = (x - x^{min})/b - j(x) \in [0, 1)$ and returns

$$h(x, \beta) = \beta_{j(x)}(1 - \alpha(x)) + \beta_{j(x)+1}\alpha(x)$$

Let

$$u_b(x) = \begin{cases} 1 - \alpha(x) & \text{if } b = j(x) \\ \alpha(x) & \text{if } b = j(x) + 1 \\ 0 & \text{else} \end{cases}$$

for $b \in \{0, ..., B\}$ be a set of $B+1$ piecewise linear basis functions. Then $h(x)$ can be written as a linear combination:

$$h(x, \beta) = \sum_{b=0}^{B} \beta_b u_b(x) = \beta^T \mathbf{u}(x)$$

where $\mathbf{u}(x) = (u_0(x), ..., u_B(x))^T$ is the vector of responses of the basis functions and $\beta = (\beta_0, ..., \beta_B)^T$ is the parameter vector.

Some recent works [44, 33] also use nonlinear additive models that depend on the variables through one dimensional smooth functions.

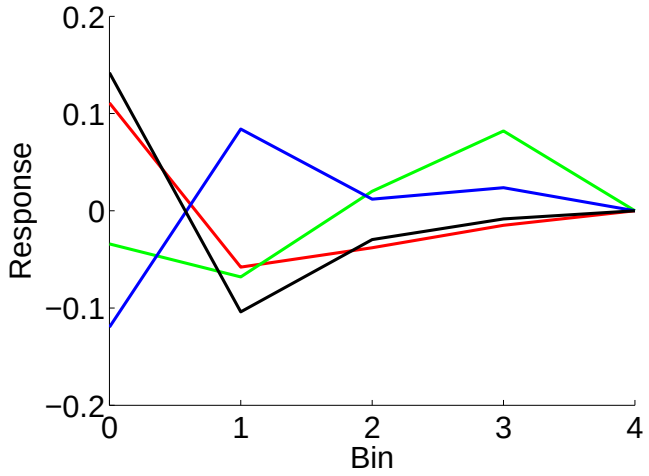


Figure 4.2: Examples of trained piecewise linear response functions $h_k(x_k, \beta_k) = \beta_k^T \mathbf{u}_k(x)$ that are the components of the ranking function $h_\beta(\mathbf{x})$.

4.4.2 FSA-PL For Ranking

Using the notations from the previous section, we can write the nonlinear ranking function as

$$h_\beta(\mathbf{x}) = \sum_{k=1}^{M} \beta_k \mathbf{u}_k(x_k), \tag{4.5}$$

where $\mathbf{u}_k(x_k)$ is the basis response vector and $\beta_k \in \mathbb{R}^{B+1}$ is the coefficient vector of variable k.

The loss function (4.1) in this case has the partial derivatives

$$\frac{\partial L(\beta)}{\partial \beta_k} = \sum_{i,j \in C} \left(\frac{1}{1+e^{h_\beta(\mathbf{x}_j) - h_\beta(\mathbf{x}_i)}} - r_{ij} \right) (\mathbf{u}_k(x_{ik}) - \mathbf{u}_k(x_{jk})) + \frac{\partial \rho(\beta_k)}{\partial \beta_k}$$

where x_{ik} is variable k of observation $\mathbf{x}_i$.

The obtained method for training a ranking function using FSA is summarized in Algorithm 5.

In this work we use the shrinkage prior for each coefficient vector $\beta_k \in \mathbb{R}^{B+1}$

$$\rho(\beta_k) = \lambda \|\beta_k\|^2, \tag{4.6}$$

which discourages large values of the coefficients and gives response functions such as those shown in Figure 4.2.

Algorithm 5 FSA-Rank

Input: Training examples $\{\mathbf{x}_i\}_{i=1}^N$ and ground truth rankings $r_{ij}, (i,j) \in C \subset \{1,...,N\} \times \{1,...,N\}$.

Output: Trained model parameters β.

Initialize $\beta = 0$.
for e=1 to N^{iter} **do**
  for $(i,j) \in C$ **do**
    Compute $d_{ij} = \dfrac{1}{1 + e^{h_\beta(\mathbf{x}_j) - h_\beta(\mathbf{x}_i)}} - r_{ij}$
    Update

$$\beta_k \leftarrow \beta_k + \eta d_{ij}(\mathbf{u}_k(x_{ik}) - \mathbf{u}_k(x_{jk})),\ k = \overline{1, M}$$

  end for
  Update

$$\beta_k \leftarrow \beta_k + \eta \nabla \rho(\beta_k)$$

  Keep the M_e variables with highest $\|\beta_k\|^2$ and renumber them $1, ..., M_e$.
  Set $M = M_e$
end for

The FSA-Rank method will be used in the next section to compare motion segmentations and choose the best one from a set of segmentations with different parameters.

4.5 Ranking for Motion Segmentation

A popular method for sparse motion segmentation is spectral clustering [37]. In this method the feature point trajectories are projected to a lower dimensional space where spectral clustering is performed according to an affinity measure.

A major difficulty in this approach is that a rigid motion lies in a low dimensional space that does not have a fixed dimension. As a result, when there are several motions present in the same video sequence, it is hard to determine the best projection dimension for spectral clustering.

Consequently, some segmentation methods [16, 37] propose to project to a number of spaces of different dimensions and find the best results according to some measure.

However, it is hard to find a versatile measure that is consistent in finding the best dimension in all scenarios. Also, segmentation algorithms always have one or more parameters that need to be tuned according to different scenarios, for example, the noise level, the separability of the affinity measure, etc. It is also hard to expect there exists a set of parameters that work well for all problems.

Moreover, many motion segmentation algorithms have been published in recent years. Different algorithms have different weaknesses and work best on different datasets. It would be of practical importance to segment one sequence by many different algorithms and find an automatic way to select the best segmentation.

In this work, we address the problem of choosing the best segmentation from a larger set of segmentations that are generated by different algorithms or one algorithm with different parameters. We formalize it as a ranking problem and solve it using supervised learning with the FSA-Rank algorithm.

4.5.1 Segmentation by Spectral Clustering

The segmentation candidates are generated by the velocity clustering (VC) algorithm 3. For each sequence, after removing possible repetitive segmentations, around $2K + 1$ segmentations would be generated for each sequence. The performance of a segmentation is characterized by (3.6) which could be easily calculated by comparing with the ground truth segmentation. While the VC proposes a error measure to select the best segmentation, this chapter solves the same problem by learning.

4.5.2 Likelihood and Prior Based Features

A motion segmentation can be described by a labeling $L : \{1, .., P\} \rightarrow \{1, .., K\}$. We will use two types of features that can characterize the ranking of a motion segmentation L: likelihood features and prior features.

Under the orthographic camera assumptions, the point trajectories of each rigid motion should lie in a 3 dimensional affine subspace.

For a segmentation the **likelihood features** are used to measure how far are the point trajectories of the same label from lying in a 3D linear subspace.

For both the original trajectory vectors and the points obtained by projection to space of dimension d, where d is a parameter, we fit in a least squares sense 3-D affine subspaces S_l through the points of motion label $l \in \{1, ..., K\}$. Denote $L(i)$ as the label of trajectory t_i and let $D(t, S)$ be the euclidean distance of point t to plane S. Let N is the total number of trajectories.

We use three types of likelihood features:

- The average distance
$$\frac{1}{N}\sum_{i=1}^{N} D(t_i, S_{L(i)})$$
- The average squared distance
$$\frac{1}{N}\sum_{i=1}^{N} D^2(t_i, S_{L(i)})$$
- The average thresholded distance
$$\frac{1}{N}\sum_{i=1}^{N} I(D(t_i, S_{L(i)}) \geq \tau),$$
where $I(\cdot)$ is the indicator function taking on value 1 if its argument is true or 0 otherwise, and τ is a threshold.

Inspired by VC, the first and second types of features obtained in dimensions $d \in [2K, 4K]$ are sorted and the smallest 4 are used as features.

By changing the threshold τ and dimension d a number of features of the third type can be obtained.

The **prior features** measure the compactness of the partition over different graphs.

For a given k, the k-nearest neighbor (kNN) graph is constructed using a distance measure in a space of a given dimension d. The distance could be either the Euclidean distance or the angular distance defined in eq. (3.2). By changing the dimension d, number of neighbors k and distance measure a number of different graphs and features are obtained.

On the kNN graph $G = (V, E)$ the prior feature is the proportion of the edges that connect vertices with different labels
$$F_G = \frac{|(i, j) \in E, L(i) \neq L(j)|}{|E|}$$

where $L(i)$ is the segmentation label of vertex $i \in V$.

In total, the features described in this section result in more than 2000 features for each segmentation.

4.5.3 Training the Ranking Function

The true rankings $r_{ij}, (i,j) \in C$ is constructed based on the relative misclassification rates of the segmentations. Since at test time only segmentations belonging to the same sequence will be compared, the set C contains only pairs of segmentations obtained from the same sequence.

For any two segmentations i, j obtained from the same sequence, the ranking r_{ij} is based on the misclassification rates of the two segmentations, with value 1 is i is better than j, 0.5 if they have the same error and 0 if j is better than i.

These ground truth rankings and the feature vectors for each segmentation are used in the FSA-Rank Algorithm 5 to obtain the parameter vector β that generates the ranking function $h_\beta(\mathbf{x})$ from eq. (4.5).

4.5.4 Motion Segmentation Algorithm

Given a new sequence, the learned parameter vector β is used to select the best segmentation for that sequence. The whole procedure is described in Algorithm 6.

4.6 Experimental Results

The proposed method was evaluated on the Hopkins 155 dataset [66].

4.6.1 RankBoost

The RankBoost algorithm [22] was used in this chapter as the baseline method to compare performance in learning the ranking function.

Let $S = \{\mathbf{x}_i \in \mathbb{R}^M, i = \overline{1,N}\}$ be the set of training instances. We assume that a ground truth ranking is given on a subset $C \subset \{1,...,N\} \times \{1,...,N\}$ as $r_{ij}, (i,j) \in C$ where $r_{ij} > 0$ means x_i should be ranked above x_j and vice versa.

RankBoost searches for a ranking which is similar to the given ranking r. To formalize the goal, a distribution D is constructed by $D_{ij} = c \cdot \max\{0, r_{ij}\}$, where c is a constant to

Algorithm 6 Motion Segmentation using Ranking

Input: The measurement matrix $W = [t_1, t_2, \ldots, t_P] \in \mathbb{R}^{2F \times P}$ whose columns are point trajectories, and the number of clusters K.

Preprocessing: Build the velocity measurement matrix $W' = (v(t_1), ..., v(t_P))$ where $v(t)$ is given in eq. (3.1).

for $d = d_{min}$ **to** d_{max} **do**

1. Perform SVD: $W' = U\Sigma V^T$
2. Obtain P projected points as the columns of the $d \times P$ matrix

$$X_d = [v_1, ..., v_d]^T$$

where v_i is the i-th column of V.

3. Apply spectral clustering to the P points of X_d using the affinity measure (3.2), obtaining segmentation L_d.
4. Extract feature vector $\mathbf{x}_d$ from segmentation L_d as described in Section 4.5.2.
5. Compute the ranking

$$h(L_d) = \sum_{k=1}^{M} \beta_k \mathbf{u}_k(x_{dk})$$

end for

Output: The segmentation result L_d with the largest value of $h(L_d)$.

make $\sum_{(i,j)\in C} D_{ij} = 1$. The learning algorithm tries to find a ranking function $H : \mathbb{R}^M \to \mathbb{R}$ that minimizes the weighted sum of wrong orderings:

$$\text{rloss}_D = \sum_{(i,j)\in C} D_{ij} I(H(\mathbf{x}_1) \leq H(\mathbf{x}_0))$$

where again $I(\pi)$ is 1 if predicate π holds and 0 otherwise. The ranking function $H(x)$ is a weighted sum of weak rankers which are selected iteratively

$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}).$$

At iteration t, RankBoost selects the best weak ranker h_t along with its weighted ranking score α_t from the pool of candidate weak rankers, and adds $\alpha_t h_t(x)$ to the ranking function $f_{t-1}(x)$.

We used threshold-based weak rankers

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } x_i > \theta \\ 0 & \text{if } x_i \leq \theta \end{cases} \tag{4.7}$$

that depend on the threshold $\theta \in \mathbb{R}$ and the variable index i. The pool of weak rankers is generated using all variables $i = \overline{1, M}$ and $B = 64$ equally spaced thresholds on the range of each feature.

4.6.2 Misclassification Rate

Parameter Settings

The parameters for RankBoost were the following: the number of thresholds $B = 64$, and the number of boosting iterations was set to 100.

The parameters for our FSA-Rank method were: number of bins $B = 4$, the number of features $s = 40$. The other parameters are $N^{iter} = 300, \eta = 0.5, \mu = 1, \lambda = 0.01$.

Ten Fold Cross Validation

The Hopkins 155 dataset contains sequences from 50 videos. The 50 videos were divided at random into 10 subsets, each subset containing 5 videos. For each subset of 5 videos were collected all the Hopkins 155 sequences corresponding to these 5 videos. This way the 155 Hopkins sequences were also divided into 10 subsets. The reason for separating the videos first and then the sequences is fairness. Some 2 motion sequences are subsets of 3 motion sequences, and it is possible that the segmentation from 2 motions is a subset of that of 3 motions. If this happens, then it be unfair to have a 3-motion sequence in the training set and a 2-motion subset from the same sequence for testing.

At round k of the cross validation, we select the k-th of the 10 subsets of sequences as the test set and form the training set from the remaining 9 subsets. After training, we apply the obtained ranking function to rank the motion segmentations for each sequence. The best one is picked as the final result to calculate classification rate.

Ranking Accuracy

Each sequence would be selected in the training set 9 times and in the test set once. In Table 4.1 are shown the average misclassification rates over all sequences when they were in the training set and when they were in the test set. Our method outperforms the RankBoost algorithm in every category on both training and test sets, even though Rankboost uses 100 boosting iterations (thus about 100 features) while FSA-Rank uses only 40 features.

Also the difference in misclassification rate between the training set and test set is very small for FSA-Rank, especially for 2-motion sequences. In comparison, the average

Table 4.1: Misclassification rate (in percent) for sequences of full trajectories in the Hopkins 155 dataset.

Method	SC	SSC	VC	RankBoost		FSARank					
						Likelihood Features		Prior Features		All Features	
				Train	Test	Train	Test	Train	Test	Train	Test
Checkerboard (2 motion)											
Average	0.85	1.12	0.67	0.67	0.74	0.58	0.69	1.09	1.28	**0.12**	**0.12**
Median	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Traffic (2 motion)											
Average	0.90	**0.02**	0.99	0.69	0.72	0.80	0.76	4.25	4.25	0.59	0.58
Median	0.00	0.00	0.22	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00
Articulated (2 motion)											
Average	1.71	**0.62**	2.94	2.05	2.26	2.30	2.27	1.32	1.32	1.32	1.32
Median	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
All (2 motion)											
Average	0.94	0.82	0.96	0.80	0.87	0.80	0.85	1.93	2.05	**0.35**	**0.35**
Median	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Checkerboard (3 motion)											
Average	2.15	2.97	0.74	0.85	2.60	0.74	0.74	4.10	4.22	**0.49**	**0.49**
Median	0.47	0.27	0.21	0.26	0.26	0.21	0.21	0.24	0.24	0.21	0.21
Traffic (3 motion)											
Average	1.35	**0.58**	1.13	4.15	4.24	1.13	1.13	4.05	4.05	1.73	1.07
Median	0.19	0.00	0.21	0.00	0.47	0.00	0.00	0.00	0.00	0.00	0.00
Articulated (3 motion)											
Average	4.26	**1.42**	5.65	3.66	18.09	5.32	5.32	3.19	3.19	3.19	3.19
Median	4.26	0.00	5.65	3.66	18.09	5.32	5.32	3.19	3.19	3.19	3.19
All (3 motion)											
Average	2.11	2.45	1.10	1.67	3.82	1.08	1.08	4.04	4.13	0.90	**0.76**
Median	0.37	0.20	0.22	0.20	0.32	0.20	0.20	0.20	0.20	0.00	0.00
All sequences combined											
Average	1.20	1.24	0.99	1.00	1.54	0.86	0.90	2.40	2.52	0.47	**0.44**
Median	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

misclassification rate of 3 motions on test set of RankBoost is about 50% larger than that on training set, while these two misclassification rates are quite close on our method. This is probably due to the small number of features selected and the shrinkage prior (4.6), which together helped obtain a small training error and good generalization.

Compared to VC [16] which uses a fixed measure to select best segmentation, our method works better on 2 motions sequences which constitutes most of the dataset. For the 3 motion sequences, our method works better on Checkerboard and Articulated categories and worse on the Traffic category. Moreover, the average misclassification rates of our method on both 2 motions and 3 motions are almost half of those from SSC [37].

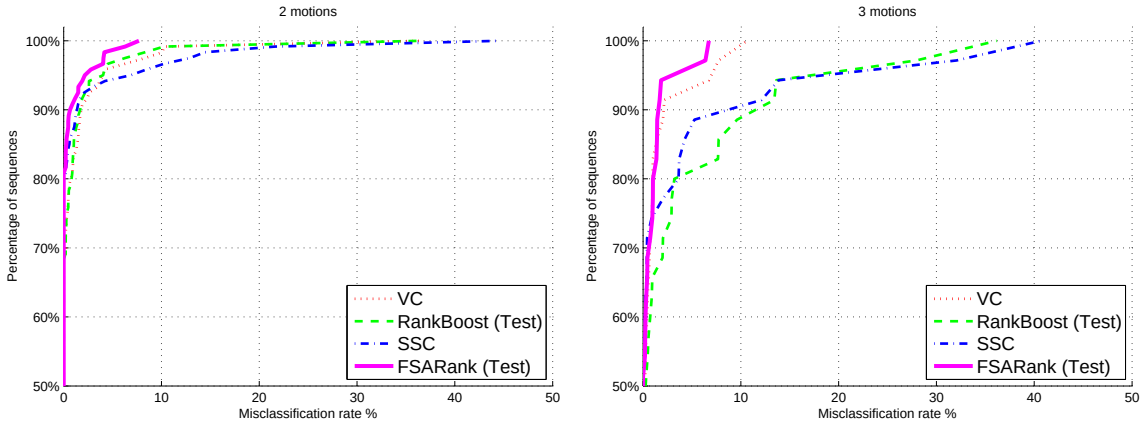


Figure 4.3: The cumulative distribution of the misclassification rate for two and three motions in the Hopkins 155 database.

From the cumulative distributions shown in Figure 4.3, we see that for 2 motions our method performs much better than the other methods compared, while for 3 motions our method is comparable to the best (VC). Nevertheless, our method outperforms RankBoost in both situations.

4.7 Conclusion

This chapter presented a novel method for training a ranking function, based on the Feature Selection with Annealing algorithm. The method allows the use of priors on the ranking function and nonlinear dependence on the feature values, obtaining models that are flexible and generalize well to unseen data.

A second contribution is a set of likelihood and prior based features for ranking motion segmentations. The likelihood features measure how well the points from each motion fit a rigid motion model. The prior features measure how compact are the different motion clusters.

A third contribution is an approach for motion segmentation based on generating a number of motion segmentations with different parameters and ranking them using the proposed ranking function. Experiments on the Hopkins 155 dataset show that the proposed approach outperforms a similar approach based on Rankboost and is competitive with other state of the art motion segmentation methods.

CHAPTER 5

SCALABLE MOTION SEGMENTATION USING SWENDSEN-WANG CUTS

5.1 Introduction

A common approach in the state of the art sparse motion segmentation methods [18, 37, 41, 70, 75] is to project the feature trajectories to a lower dimensional space and use spectral clustering to group the projected points and obtain the motion segmentation.Even though these methods obtain very good results on standard benchmark datasets, the spectral clustering algorithm requires expensive computation of eigenvectors and eigenvalues on an $N \times N$ dense matrix where N is the number of trajectories. In this manner, the computation time for these motion segmentation methods scales as $O(N^3)$, so it can become prohibitive for large problems (e.g. $N = 10^5 - 10^6$).

5.1.1 Our Contributions

In this chapter, we propose a completely different approach to motion segmentation, based on the Swendsen-Wang Cut (SWC) algorithm [4] and bring the following contributions:

- We formulate the sparse motion segmentation as an energy minimization problem in a Bayesian framework with prior and likelihood terms. The data term is based on a rigid motion model, while the prior is an Ising/Potts prior.
- We solve the energy minimization problem using the Swendsen-Wang Cuts (SWC) algorithm and simulated annealing. The SWC algorithm needs a weighted graph to propose good data-driven clusters for label switching. We construct this graph as a k-NN graph from an affinity matrix.

- We study the computation complexity of the SWC algorithm and observe that it scales as $O(N^2)$ for all practical values of N, making the proposed approach more scalable than spectral clustering (an $O(N^3)$ algorithm) to large scale problems.
- We conduct experiments on the Hopkins 155 dataset to compare the performance of the proposed algorithm with the state of the art methods. We observe that it obtains an error less than twice the error of the state of the art methods. We also observed experimentally that the SWC scaling is about $O(N^{1.3})$ and the spectral clustering scaling is about $O(N^{2.5})$.

Compared to other statistical methods [20, 43, 61], our method doesn't require a good initialization, which can be hard to obtain.

Overall, our method provides a new perspective to solve the motion segmentation problem, and demonstrates the power of Swendsen-Wang Cuts algorithm in clustering problems. While it does not obtain a better average error, it scales better to large datasets, both theoretically as $O(N^2)$ and practically as $O(N^{1.3})$.

5.2 The Swendsen-Wang Cuts Method

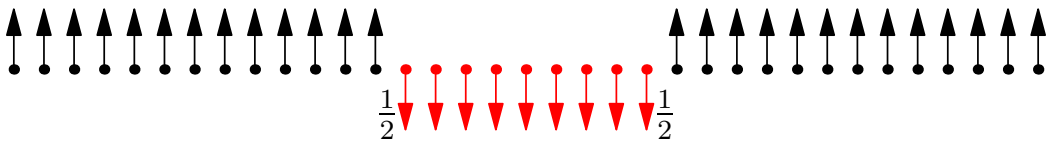


Figure 5.1: Difficulty in sampling the Ising and Potts models

The Swendsen-Wang (SW) method [59] is a Markov Chain Monte Carlo (MCMC) algorithm for sampling partitions (labelings) $\pi : V \to \{1, ..., N\}$ of a given graph $G =< V, E >$. The probability distribution over the space of partitions is the Ising/Potts model

$$p(\pi) = \frac{1}{Z} \exp[- \sum_{<i,j>\in E} \beta_{ij} \delta(\pi(i) \neq \pi(j)]. \tag{5.1}$$

where $\beta_{ij} > 0, \forall < i, j >\in E$ and $N = |V|$. Figure 5.1 shows a string of spins whose label (color) π could be +1 (up) and -1 (down). For a single site update algorithm, like the Gibbs sampler, the probability for flipping each spin from -1 to +1 is 1/2. Thus the expected number of steps to flip a string of n spins from -1 to +1 is 2^n. This is exponential waiting.

The SW algorithm addresses the slow mixing problem of the Gibbs Sampler [26], which changes the label of a single node in one step. Instead, the SW algorithm constructs clusters of same label vertices in a random graph and flips together the label of all nodes in each cluster. The random graph is obtained by turning off (removing) all graph edges $e \in E$ between nodes with different labels and removing each of the remaining edges $< i, j > \in E$ with probability $e^{-\beta_{ij}}$. The whole process is shown in Figure 5.2. While the original SW method was developed originally for Ising and Potts models, the Swendsen-Wang Cuts (SWC) method [4] generalized SW to arbitrary posterior probabilities defined on graph partitions.

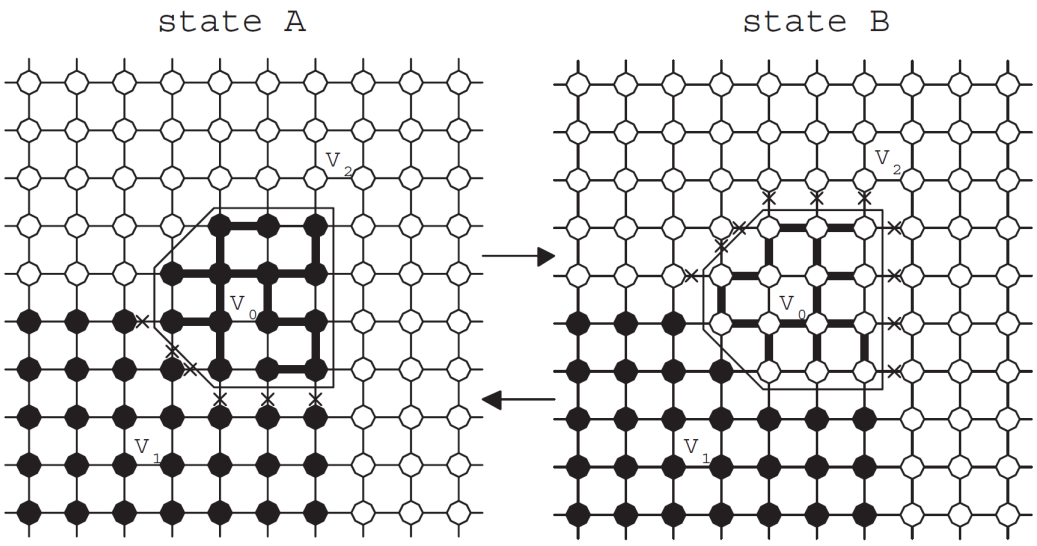


Figure 5.2: The edges between vertices of the same label are turned on/off probabilistically to yield a number of connected components. The Swendsen-Wang algorithm flips the color of a connected component V_0 in one step. The set of edges marked with crosses is called the Swendsen-Wang cut [4].

The SWC method relies on a weighted adjacency graph $G =< V, E >$ where each edge weight $q_e, e =< i, j > \in E$ encodes an estimate of the probability that the two end nodes i, j belong to the same partition label. The idea of the SWC method is to construct a random graph in a similar manner to the SW but based on the edge weights, select one connected component at random and accept a label flip of all nodes in that component with a probability that is based on the posterior probabilities of the before and after states and the graph edge weights.

This algorithm was proved to simulate ergodic and reversible Markov chain jumps in the space of graph partitions and is applicable to arbitrary posterior probabilities or energy functions.

From [4], there are different versions of the Swendsen-Wang Cut algorithm. We use the SWC-1 algorithm which is shown in Figure 5.3.

Input: Graph $G =< V, E >$, with weights q_e, $\forall e \in E$, and posterior probability $p(\pi|I)$.
Initialize: A partition $\pi : V \to \{1, ..., N\}$ by random clustering
for $t = 1, \ldots T$, current state π, **do**

1. Find $E(\pi) = \{< i, j >\in E, \pi(i) = \pi(j)\}$
2. For $e \in E(\pi)$, turn $\mu_e =$ off with probability $1 - q_e$.
3. $V_l = \pi^{-1}(l)$ is divided into N_l connected components $V_l = V_{l1} \cup \ldots \cup V_{ln_l}$ for $l = 1, 2, \ldots, N$.
4. Collect all the connected components in set $CP = \{V_{li} : l = 1, \ldots, N, i = 1, \ldots, N_l\}$.
5. Select a connected component $V_0 \in CP$ with probability $q(V_0|CP) = \dfrac{1}{|CP|}$, say $V_0 \subset V_l$.
6. Propose to assign V_0 a new label $c_{V_0} = l'$ with a probability $q(l'|V_0, \pi)$, thus obtaining a new state π'.
7. Accept the proposed label assignment with probability.

$$\alpha(\pi \to \pi') = \min(1, \frac{\prod_{e\in\mathcal{C}(V_0, V_{l'}-V_0)} (1-q_e)}{\prod_{e\in\mathcal{C}(V_0, V_l-V_0)} (1-q_e)} \cdot \frac{q(c_{V_0} = l|V_0, \pi')}{q(c_{V_0} = l'|V_0, \pi)} \cdot \frac{p(\pi'|I)}{p(\pi|I)}. \tag{5.2}$$

end for
Output: Samples $\pi \sim p(\pi|I)$.

Figure 5.3: The Swendsen-Wang Cut algorithm [4].

The set of edges $\mathcal{C}(V_0, V_{l'} - V_0), \mathcal{C}(V_0, V_l - V_0)$ from eq. (5.2) are the SW cuts defined in general as

$$\mathcal{C}(V_1, V_2) = \{< i, j >\in E, i \in V_1, j \in V_2\}$$

The algorithm could automatically decide the number of clusters. But in this chapter, it is assumed that the number of motions is already known. Thus the new color is sampled

with uniform probability from the number M of motions:

$$q(c_{V_0} = l' | V_0, \pi) = 1/M.$$

5.3 Motion Segmentation Background

In general, we are given a measurement matrix W that contains trajectories from multiple possibly nonrigid motions. The task of motion segmentation is to cluster together all trajectories coming from each motion.

A popular approach [16, 37, 50, 70] is to project the trajectories to a lower dimensional space and to perform clustering in that space.

5.3.1 Dimension Reduction

Dimension reduction is an essential preprocessing step for obtaining a good motion segmentation. To realize this goal, the truncated SVD is often applied [16, 37, 50, 70].

To project the measurement matrix $W \in \mathbb{R}^{2F \times N}$ to $X = [x_1, ..., x_N] \in \mathbb{R}^{D \times N}$, where D is the desired projection dimension, the matrix W is decomposed via SVD as $W = U\Sigma V^T$ and the first D columns of the matrix V are chosen as X^T.

The value of D for dimension reduction is also a big concern in motion segmentation. This value has a large impact on the speed and accuracy of the final result, so it is very important to select the best dimension to perform the segmentation. The dimension of a motion is not fixed, but can vary from sequence to sequence, and since it is hard to determine the actual dimension of the mixed space when multiple motions are present, different methods may have different dimensions for projection. In this chapter, we find that projecting to dimension $D = 2M + 1$ can generate good results.

The computation complexity of computing the SVD of a $m \times n$ matrix M when $m >> n$ is $O(mn^2 + n^3)$ [65]. If $n >> m$ then it is faster to compute the SVD of M^T, which takes $O(nm^2 + m^3)$.

Assuming that $2F << N$, it means that the SVD of W can be computed in $O(NF^2 + F^3)$ operations.

5.3.2 Trajectory Affinity Matrix

A popular clustering method [16, 37, 50] that is applied after the dimension reduction step is spectral clustering, which relies on an affinity matrix that for any pair of trajectories measures how likely they belong to the same motion.

In this work we will use the affinity matrix to compute the weighted adjacency graph $G =< V, E >$ needed in the Swendsen-Wang Cuts algorithm.

When building the adjacency graph of trajectories, one challenge is to find a good distance metric. Two points from two different subspaces that are near the intersection of the subspaces may be close to each other and conversely, two points in the same subspace could be far from each other.

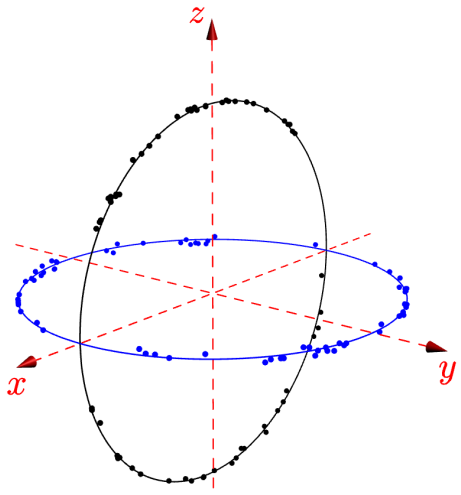


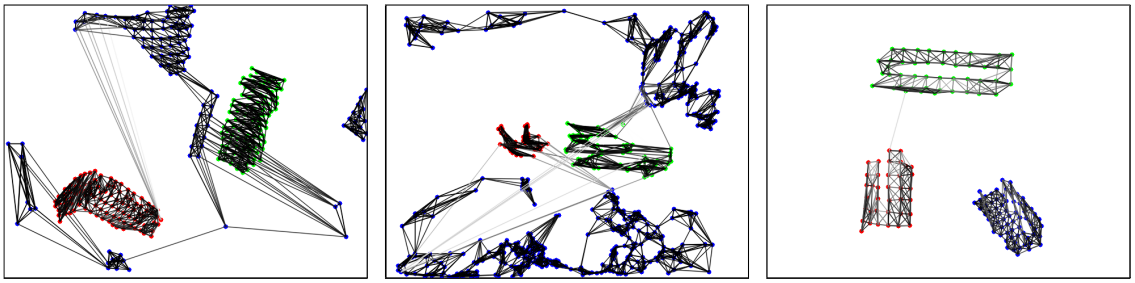
Figure 5.4: Two 2D subspaces in 3D. The points in both subspaces have been normalized to unit length. Due to noise, the points may not lie exactly on the subspace. One can observe that the angular distance may find the neighbors in most places except at the plane intersections.

As a result, the traditional Euclidean-based distance might not be the best choice. Inspired by the affinity measure in [37], we modify the measure to

$$A_{ij} = \exp(-m\frac{\theta_{ij}}{\bar{\theta}}), \quad i \neq j \tag{5.3}$$

where θ_{ij} is the angle between the vectors x_i and x_j, which could be calculated by

$$\theta_{ij} = 1 - (\frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2})^2,$$



(a) 1RT2TC (b) cars3 (c) articulated

Figure 5.5: Examples of weighted graphs constructed for the SWC algorithm for a checkerboard (left), traffic (middle) and articulated (right) sequence. The images show the positions of the feature points in the first frame. The edge intensities represent their weights from 0 (white) to 1 (black).

and $\bar{\theta}$ is the average of all θ. The parameter m is a tuning parameter to control the size of the connected components obtained by the SWC algorithm. In Figure 5.7, left is shown the average performance of the motion segmentation with SWC for a range of values of this parameter.

Fig 5.4 shows two linear subspaces, where all points have been normalized to have norm 1. It is intuitive to find that the point tends to be in the same subspace as their neighbors in angular distance except those near the intersection of the subspaces.

The affinity measure based on the angular information between points enables to obtain the neighborhood graph, for example based on the k-nearest neighbors. After the graph has been obtained, the affinity measure is also used to obtain the edge weights for making the data driven clustering proposals in the SWC algorithm as well as for the prior term of the posterior probability.

5.4 Motion Segmentation by Swendsen-Wang Cuts

To apply the Swendsen-Wang Cuts algorithm to motion segmentation, we need to specify a posterior probability $p(\pi) = p(\pi|I)$ that will be sampled and optimized, as well as an adjacency graph $G = (V, E)$ that will be used by the SWC to generate clusters for label switching.

Most works in motion segmentation use spectral clustering for obtaining the segmen-

tation. Spectral clustering optimizes an approximation of the normalized cut or the ratio cut [72], which are discriminative measures. In contrast, the proposed approach optimizes a generative model where the likelihood term is based on the rigid motion model under the affine camera assumption. It is possible that the discriminative measures are more flexible and obtain better results, and we will study them in future work.

5.4.1 Graph Construction

The graph $G = (V, E)$ has as vertices the set of trajectories that need to be clustered. The edges E are constructed based on the proposed distance measure from eq. (5.3). Since the distance measure is more accurate in finding the nearest neighbors (NN) from the same subspace, the graph is constructed as the k-nearest neighbor graph (kNN), where k is a given parameter.

5.4.2 Posterior Probability

The posterior probability is defined in a Bayesian framework up to a constant

$$p(\pi) \propto \exp[-E_{data}(\pi) - E_{prior}(\pi)]$$

The normalizing constant is irrelevant in the optimization since it cancels out in the acceptance probability.

To construct the data term $E_{data}(\pi)$, we adopt a model based on the affine camera model. Given the current partition (or labeling) π, for each label l an affine subspace L_l is fitted in a least squares sense through all points with label l. Denote the distance of a point x with label l to the linear space L_l as $d(x, L_l)$. Then the data term is

$$E_{data}(\pi) = \sum_{l=1}^{M} \sum_{i,\pi(i)=l} d(x_i, L_l) \tag{5.4}$$

The prior term $E_{prior}(\pi)$ is set to encourage tightly connected points to stay in the same cluster.

$$E_{prior}(\pi) = -\rho \sum_{<i,j>\in E, \pi(i)\neq\pi(j)} \log(1 - A_{ij}), \tag{5.5}$$

where ρ is a parameter controlling the strength of the prior term. This is exactly the Potts prior from (5.1) that would give A_{ij} as the edge weights in the original SW algorithm.

5.4.3 Optimization by Simulated Annealing

The SWC algorithm is designed for sampling the posterior probability $p(\pi)$. To use SWC for optimization, a simulated annealing scheme should be applied while running the SWC algorithm.

Simulated annealing means the probability used by the algorithm is not $p(\pi)$ but $p(\pi)^{1/T}$ where T is a "temperature" parameter that is large at the beginning of the optimization and is slowly decreased according to an annealing schedule. If the annealing scheduled is slow enough, it is theoretically guaranteed [36] that the global optimum of the probability $p(\pi)$ will be found.

In reality we use a faster annealing schedule, and the final partition π will only be a local optimum. We use an annealing schedule that is controlled by three parameters: the start temperature T_{start}, the end temperature as T_{end}, and the number of iterations N^{it}. The temperature at step i is calculated as

$$T_i = \frac{T_{\text{end}}}{\log\left(\frac{i}{N}[e - \exp(\frac{T_{\text{end}}}{T_{\text{start}}})] + \exp(\frac{T_{\text{end}}}{T_{\text{start}}})\right)}, i = \overline{1, N^{it}} \tag{5.6}$$

To better explore the probability space, we also use multiple runs with different random initializations. Then the final algorithm is shown in Figure 5.6.

Input: N trajectories $(t_1, \ldots, t_N)$ from M motions
Dimension reduction: Project the trajectories to a D-dimensional space by truncated SVD, obtaining points $(x_1, \ldots, x_N)$.
Construct the adjacency graph G as a k-NN graph using eq (5.3)..
for $r = 1, \ldots, Q$ **do**
  Initialize the partition π as $\pi(i) = 0, \forall i$.
  for $i = 1, \ldots, N^{it}$ **do**
    1. Compute the temperature T_i using eq (5.6).
    2. Run one step of the SWC algorithm 5.3 using $p(\pi|I) = p^{1/T_i}(\pi)$ in eq (5.2).
  end for
  Record the clustering result π_r and the final probability $p_r = p(\pi_r)$.
end for
Output: Clustering result π_r with the largest p_r.

Figure 5.6: The Swendsen-Wang Cuts algorithm for sparse motion segmentation.

5.4.4 Complexity Analysis

Let N be the number of feature trajectories that need to be clustered. The complexity of the proposed motion segmentation method can be broken down as follows:

- The dimension reduction step is $O(NF^2 + F^3)$ as discussed in Section 5.3.1.
- Adjacency graph construction is $O(N^2 D \log k)$ since for one point, one needs to calculate the distance from it to the other $N-1$ points and use a heap to maintain its k-NNs.
- Each of the N^{it} iterations of the SWC algorithm involves:
  - Sampling the edges at each SWC step is $O(|E|) = O(N)$ since the k-NN graph $G =< V, E >$ has at most $2kN$ edges.
  - Constructing connected components at each SWC step is $O(|E|\alpha(|E|)) = O(N\alpha(N))$ using the disjoint set forest data structure [23, 21]. The function $\alpha(N)$ is the inverse of $f(n) = A(n,n)$ where $A(m,n)$ is the fast growing Ackerman function [1] and $\alpha(N) \leq 5$ for $N \leq 2^{2^{10^{19729}}}$.
  - Computing $E_{data}(\pi)$ involves fitting linear subspaces for each motion cluster, which is $O(D^2 N) + O(D^3)$ where $D = 2M + 1$ is the projection dimension.
  - Computing the $E_{prior}(\pi)$ is $O(N)$.

The number of iterations is $N^{it} = 2000$, so all the SWC iterations take $O(N\alpha(N))$ time.

We see that the entire algorithm complexity in terms of the number N of trajectories is $O(N^2)$ so it should scale better than spectral clustering for large problems.

5.5 Experiments on the Hopkins 155 Dataset

In this section, we apply the Swendsen-Wang algorithm to the motion segmentation problem. We evaluate our algorithm on the Hopkins 155 motion database [66].

Before applying the Swendsen-Wang method, we reduce the dimension of the data from $2F$ to $D = 2M + 1$, where M is the number of motions. After the projection, the points are assigned with same label, and then applied with the SW method. For each sequence, the misclassification rate is measured as equation 3.6.

Parameter settings. In these experiments we held the parameters constant to the following values. The number of NN (nearest neighbors) for graph construction is $k = 7$,

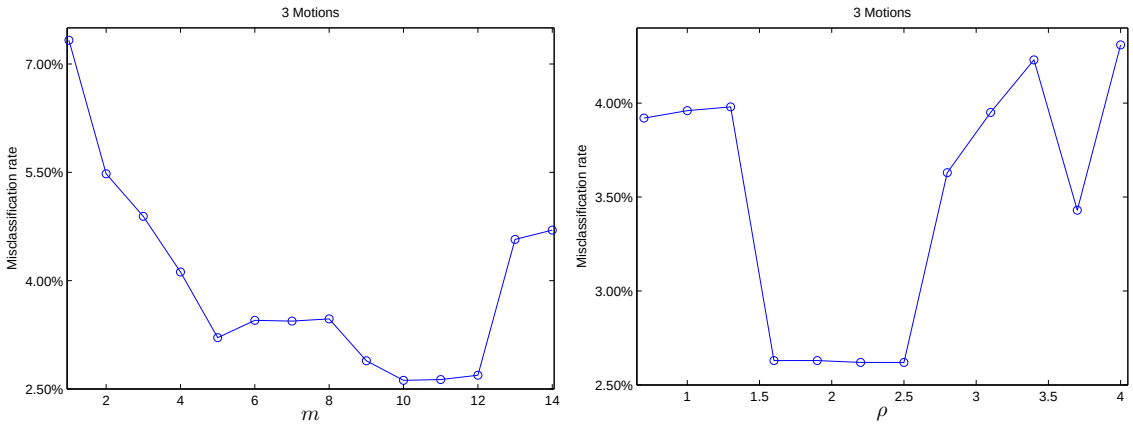


Figure 5.7: Dependence of the misclassification rate on the affinity parameter m (left) and prior strength ρ (right).

the parameter m in the affinity measure is $m = 10$, and the prior coefficient is $\rho = 2.2$. The sensitivity of the average misclassification rate to the parameters m and ρ is shown in Figure 5.7. The annealing parameters are $t_{\text{start}} = 1$, $t_{\text{end}} = 0.01$, $N^{it} = 2000$. The number of independent runs to obtain the most probable partition is $Q = 10$.

Results. In figure 5.8 shows the whole process of segmenting a sample sequence in Hopkins 155 dataset by the SWC algorithm. At first, it starts from one cluster; after a couple of iterations, the correct segmentation is obtained. The average and median classification errors are listed in Table 5.1. For accuracy, the results of the SWC algorithm from table 5.1 are averaged over 10 runs. In parentheses are shown the standard deviations of the averages over the 10 runs. In order to compare the SW method with the state of the art methods, we also list the results of ALC [50], SC [37], SSC [18] and VC [16].

We see that SWC obtains average errors that are less than twice the errors of the other methods. In our experiments we observed that the energy of the final state is usually smaller than the energy of the ground truth state. This fact indicates that the SWC algorithm is doing a good job optimizing the model but the Bayesian model is not accurate enough in its current form and could be improved.

Table 5.1: Misclassification rates (in percent) of different motion segmentation algorithms on the Hopkins 155 dataset.

Method	ALC	SC	SSC	VC	SWC (std)
Checkerboard (2 motion)					
Average	1.55	0.85	1.12	0.67	1.25 (0.11)
Median	0.29	0.00	0.00	0.00	0.00 (0.00)
Traffic (2 motion)					
Average	1.59	0.90	0.02	0.99	1.87 (0.55)
Median	1.17	0.00	0.00	0.22	0.00 (0.0)
Articulated (2 motion)					
Average	10.70	1.71	0.62	2.94	2.15 (0.11)
Median	0.95	0.00	0.00	0.88	0.00 (0.00)
All (2 motion)					
Average	2.40	0.94	0.82	0.96	1.49 (0.19)
Median	0.43	0.00	0.00	0.00	0.00 (0.00)
Checkerboard (3 motion)					
Average	5.20	2.15	2.97	0.74	2.26 (0.04)
Median	0.67	0.47	0.27	0.21	0.67 (0.00)
Traffic (3 motion)					
Average	7.75	1.35	0.58	1.13	2.88 (0.00)
Median	0.49	0.19	0.00	0.21	0.81 (0.00)
Articulated (3 motion)					
Average	21.08	4.26	1.42	5.65	6.33 (1.88)
Median	21.08	4.26	0.00	5.65	6.33 (1.88)
All (3 motion)					
Average	6.69	2.11	2.45	1.10	2.62 (0.13)
Median	0.67	0.37	0.20	0.22	0.81 (0.00)
All sequences combined					
Average	3.37	1.20	1.24	0.99	1.75 (0.15)
Median	0.49	0.00	0.00	0.00	0.00 (0.00)

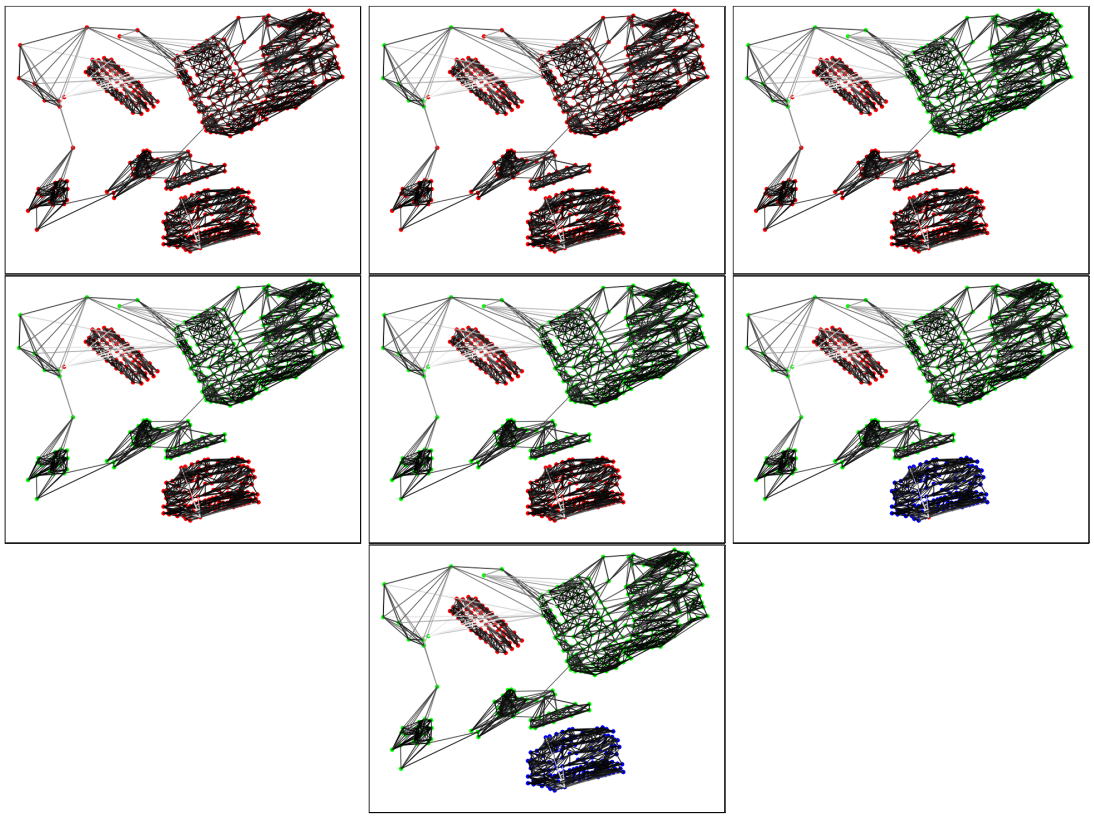


Figure 5.8: Clustering the sequence 1R2TCR of the Hopkins 155 dataset by the SWC algorithm. There are 3 motions in this example. The images show the positions of the feature points in the first frame. The darkness of the edges represents the strength of the connection. The point colors are the labeling states π obtained while running the SWC algorithm from the initial state (top left) to the final state (bottom right). The correct segmentation was successfully obtained in the end.

5.6 Scalability Experiments on Large Data

To show the scalability of different algorithms, we need evaluate on sequences that have a large number of trajectories. The trajectories could be generated by some optical flow algorithm, but it is difficult to obtain the ground truth segmentation and remove bad trajectories caused by occlusions. From the Moseg dataset, we picked the cars10 sequence and tracked every pixels of the first frame using the Classic+NL method [58]. There are two reasons for choosing cars10. Firstly it has three motions, two moving cars and the background. Secondly, the two moving cars are relatively large in the video, so that we could obtain a large number of trajectories from each motion.

(a) Frame 1 (b) Frame 10

(c) Frame 20 (d) Frame 30

Figure 5.9: Selected frames of sequence cars10 with 1000 tracked feature points.

There are 30 frames in the sequence, and 3 of them have a dense manual segmentation of all pixels. We removed trajectories that have different labels on the 3 ground truth frames. To avoid occlusion, the trajectories close to the motion boundaries were also removed. Plus, we only kept the full trajectories for clustering. At last, we obtained around 48000 trajectories as a pool. From the pool, different numbers N of trajectories could be subsampled for evaluation. For each given N, a total of N trajectories were randomly selected from the pool such that the number of trajectories in each of the three motions was roughly the same. For example, to generate $N = 1000$ trajectories, we would randomly pick from the pool 333 trajectories from two of the motions and 334 trajectories from the third motion. If there is not enough trajectories from one motion, we will add more from the motion which have the most trajectories.

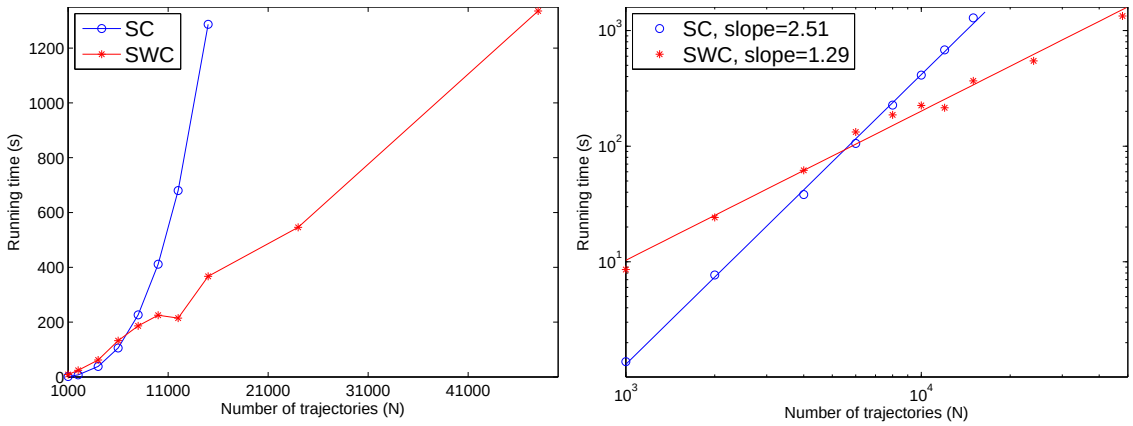


Figure 5.10: Left. Dependence of the computation time (sec) vs number of trajectories N for SC and SWC. Right: log-log plot of the same data with the fitted regression lines.

We compare our method with the SC algorithm which is one of the fastest algorithms [16] based on spectral clustering. We generated data containing between 1000 to 15000 trajectories, and applied the two segmentation algorithms. Sample frames are shown in Figure 5.9. The parameters for SC are kept the same as in the original paper, and those for SWC are identical with Section 5.5. The SC algorithm is implemented in Matlab (which has optimized SVD algorithms), while the SWC code is in C++. The experiments were performed on a Windows machine with an Intel core i7-3970 CPU and 12 GB memory. We also generated data with 24000 and 48000 trajectories for SWC clustering. For SC, the same experiments could not be conducted because Matlab will run out of memory.

The misclassification rate is recorded in Table 5.2 and the running time is shown on Figure 5.10. In Table 5.2, both methods perform well and the misclassification rate of SWC is about one third of that of the SC.

From Figure 5.10, which shows the computation time vs the number N of trajectories, one could find that for a small number of trajectories, the SC is faster than SWC, but for more than $N = 6000$ trajectories, the computation time of SC is greater than that of SWC, and increases much faster. We also plot the log(time) vs. log(N) and use linear regression to fit lines through the data points of the two methods. If the slope of the line is α, then time $= O(N^{\alpha})$. We observe that the slope of SC is 2.52 while the slope for SWC is 1.29, which is consistent with the complexity analysis of Section 5.4.4.

One may note that the complexity of the SC algorithm is different from the complexity analysis in section 1.5. The reason is that in practice, spectral clustering only needs top k eigenvectors. To calculate the top k eigenvectors, the implicitly restarted Arnoldi method (IRAM) [56, 38] is adopted in Matlab (Actually, Matlab wraps the ARPACK implementation [39]). From the computational costs analysis in [2], for each iteration in IRAM, the total cost is

$$\gamma pN + 2[(5k-2)p + 2p^2]N + 2k^2N + O((k+p)^3),$$

where $\gamma = 2N$ for a dense matrix, $p = m - k$, and m is the Arnoldi length. In practice, p is usually set to be several times larger than k. Therefore, if k is very small compared to N, the overall cost of IRAM is

$$O(N^2) \times (\#\text{ of iterations}).$$

The number of iterations in IRAM is very difficult to estimate, since it involves in the setting of tolerance, initial guess, etc. In Matlab, the maximum iteration number is set to $\max(300, \lceil 2N/\max(p,1) \rceil)$, where $\lceil\rceil$ is the ceil function. Thus, even in the worst case, when the maximum iteration number is reached, the overall complexity is $O(N^2)$ (when the iteration number is 300) or $O(N^3)$ (when the iteration number is $\lceil 2N/\max(p,1) \rceil$). Also, when N is much bigger than p, it is unlikely to achieve $O(N^2)$ complexity.

Based on the above analysis, the computation complexity of spectral clustering should be between $O(N^2)$ and $O(N^3)$ if the eigendecomposition is performed using IRAM. It is reasonable to find the complexity $O(N^{2.5})$ in the conducted experiments.

Table 5.2: Average misclassification rate for the sequence cars10 (in percent).

Number of Trajectories N	SC	SWC
1000 to 15,000	2.77	0.99
24,000 to 48,000	-	1.00

On the other hand, the complexity of the SWC is different with the analysis in theory. It is caused by the coefficient of $O(N^2)$ part is very small compared to the coefficient of the $O(N)$ part. The adjacency graph construction is only one step which has $O(N^2)$ cost. The coefficient is around $D \log k$, which is 21 in the experiments. The other steps

have complexity of $O(N)$. Among them, the most time-consuming part is the iterations of the SWC algorithm. The total iteration number would be QN^{it}, which is 20000 in the experiments. As a result, for relatively small number of trajectories N, e.g. thousands of trajectories, one could expect a complexity between $O(N)$ and $O(N^2)$. It is not a surprise to obtain complexity of $O(N^{1.3})$ in the experiments. If the number of the trajectories is much larger, the computational cost would get closer to $O(N^2)$.

5.7 Conclusion

In this chapter we presented a stochastic method based on the Swendsen-Wang Cuts (SWC) algorithm for segmenting feature trajectories obtained by a feature tracker or an optical flow algorithm. We defined the graph used by the SWC algorithm for making informed relabeling proposals as the k-NN graph based on an affinity matrix. The posterior probability is a generative model defined in a Bayesian framework with data and likelihood parts.

The complexity analysis showed that the proposed algorithm is $O(N^2)$ in the number N of trajectories that need to be clustered.This is in contrast to the spectral clustering algorithms that have $O(N^3)$ complexity.

Experiments on the Hopkins 155 motion segmentation dataset showed that the algorithm performs slightly worse than the state of the art motion segmentation algorithms based on spectral clustering. This could probably be due to the rigidity of the generative model in contrast to the flexibility of the Normalized Cut or Ratio Cut or their approximations that are optimized by the spectral clustering algorithms. Experiments also revealed that the spectral clustering is about $O(N^{2.5})$ in reality while the proposed SWC method is $O(N^{1.3})$.

In the future we will experiment with posterior probabilities based on the Normalized Cut or Ratio Cut and see if the Swendsen-Wang Cuts algorithm can obtain better errors in that case.

CHAPTER 6

SUMMARY AND FUTURE WORKS

In this chapter, we first summarize our former works on sparse motion analysis and then point out the possible directions of future work.

6.1 Summary

The generation of high quality feature trajectories is an important preprocessing step for motion segmentation. It has great influence on the results of motion segmentation. It poses many standing challenges that need to be addressed for advancing this field. One of these challenges is how to correctly handle occlusion and detect when a pixel trajectory needs to be stopped. Very few optical algorithms provide an occlusion map and are appropriate for this task. Another challenge is how to accurately evaluate the motion field produced by different algorithms. Our work makes two contributions in these directions. First, it presents a RMSE based error measure for evaluating feature tracking algorithms on sequences with rigid motion under the affine camera model. The proposed measure was observed to be consistent with the relative ranking of a number of optical flow algorithms on the Middlebury dataset. Second, it introduces a feature tracking algorithm based on RankBoost that automatically prunes bad trajectories obtained by an optical flow algorithm. The proposed feature tracking algorithm is observed to outperform many feature trackers based on optical flow using both the proposed measure and an indirect measure based on motion segmentation.

Given some good trajectories for segmentation, such as the sequences in the Hopkins 155 dataset, in this work we proposed three different motion segmentation methods, the

spectral clustering based algorithm, the learning-based algorithm and the Swendsen-Wang cuts based algorithm.

The spectral clustering based motion segmentation method uses the angular information as the affinity measure. Based on the fact that the dimension of the subspace where the point trajectories are projected has a great impact on the performance of different motion segmentation algorithms, we present a strategy for estimating the best subspace dimension using a novel clustering error measure. For each obtained segmentation, the proposed measure estimates the average least square error between the point trajectories and synthetic trajectories generated based on the motion models from the segmentation. The second contribution we make is the use of the velocity vector instead of the traditional trajectory vector for segmentation. The evaluation on the Hopkins 155 video benchmark database shows that the proposed method is competitive with current state-of-the-art methods both in terms of overall performance and computational speed.

In the learning based segmentation methods, firstly our work proposes a novel method for training a ranking function using the recently introduced Feature Selection with Annealing algorithm. The obtained ranking function depends nonlinearly on a small number of selected features, which helps in accuracy and generalization power. Second, it introduces two types of features, likelihood based and prior based, for ranking motion segmentations. Third, it introduces a ranking based motion segmentation method that uses a trained ranking function to evaluate and compare a number of motion segmentations generated with different parameters and select the best one. Experiments on the Hopkins 155 dataset show that the proposed method is competitive with the state of the art. The experiments also show that the ranking functions obtained by the proposed FSA-Rank algorithm outperform those obtained by RankBoost from the same feature pool on both training and unseen test data.

The third motion segmentation algorithm is based on energy minimization using the Swendsen-Wang cuts algorithm and simulated annealing. The energy or probability model is defined in a Bayesian framework with a likelihood based on a rigid motion model and a Ising/Potts prior to regularize the result. The computation complexity of the algorithm is $O(N^2)$, so it should scale better than the spectral clustering based methods to large

datasets, as spectral clustering has time complexity $O(n^3)$. Experiments on the Hopkins 155 dataset show that the error of the proposed method is comparable to that of the state of the art algorithms. Experiments on the running time show that the proposed method has about $O(N^{1.3})$ complexity while the spectral clustering has about $O(N^{2.5})$, confirming the conclusion that the our method scales better to large datasets.

6.2 Future Works

At present, one disadvantage of motion segmentation is that nearly all algorithms assume the number of motions is already known. But the assumption is hardly satisfied in practice. Some initial exploration has been made in [7]. Since the spectral clustering algorithm could only take ratio cuts or normalized cuts as energy, it is not suitable to determine the number of trajectories. In contrast, the Swendsen-Wang cuts could take a generic energy form and could determine the number of motions automatically. Therefore, one possible solution is to design a proper energy for the Swendsen-Wang cuts algorithm.

Another challenge is how to make the motion segmentation algorithms more scalable. Suppose one is already given a segmentation, the question is how to cluster new trajectories or how to refine the segmentation if tracked feature points from new frames are added? Currently, dimension reduction is a necessary step for almost all motion segmentation algorithms. However, either the two common dimension reduction method, truncated SVD or random sampling could not deal with new entries, which means that the segmentation has to be recomputed from scratch. A naive way to solve this problem is to fit subspaces without dimension reduction and assign new points to the cluster that generates the nearest subspace, but this is not accurate and robust at all. These difficulties may stimulate us to design new motion segmentation algorithms that are completely different from the existing ones.

BIBLIOGRAPHY

[1] Wilhelm Ackermann. Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99(1):118–133, 1928. 81

[2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. Templates for the solution of algebraic eigenvalue problems: A practical guide. 2000. 87

[3] S. Baker, D. Scharstein, JP Lewis, S. Roth, M.J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 21, 24

[4] Adrian Barbu and Song-Chun Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1239–1253, 2005. x, 72, 74, 75

[5] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996. 28, 34

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *European Conference on Computer Vision*, pages 25–36, 2004. 8, 28, 34

[7] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*, pages 282–295. 2010. 19, 25, 35, 44, 91

[8] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning*, pages 89–96, 2005. 60

[9] João Carreira, Fuxin Li, and Cristian Sminchisescu. Object recognition by sequential figure-ground ranking. *International journal of computer vision*, pages 1–20, 2012. 57

[10] Joao Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3241–3248, 2010. 57

[11] Andrea Cavallaro, Olivier Steiger, and Touradj Ebrahimi. Tracking video objects in cluttered background. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(4):575–584, 2005. 2

[12] G. Chen and G. Lerman. Spectral curvature clustering. *International Journal of Computer Vision*, 81(3):317–330, 2009. 11, 43, 46, 50

[13] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the 5th International Conference on Computer Vision*, pages 1071–1076, 1995. 50

[14] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. 42, 50

[15] Liangjing Ding, Adrian Barbu, and Anke Meyer-Baese. Learning a quality-based ranking for feature point trajectories. In *Asian Conference on Computer Vision*, 2012.

[16] Liangjing Ding, Adrian Barbu, and Anke Meyer-Baese. Motion segmentation by velocity clustering with estimation of subspace dimension. In *Asian Conference on Computer Vision Workshop on Detection and Tracking in Challenging Environments*, 2012. 63, 70, 76, 77, 82, 86

[17] D.L. Donoho and J. Tanner. Counting faces of randomly-projected polytopes when the projection radically lowers dimension. *American Mathematical Society*, 22(1):1–53, 2009. 47

[18] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 11, 19, 34, 35, 38, 41, 43, 46, 50, 52, 56, 72, 82

[19] Claude L Fennema and William B Thompson. Velocity determination in scenes containing several moving objects. *Computer graphics and image processing*, 9(4):301–315, 1979. 6

[20] M. A. Fischler and R. C. Bolles. RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 26:381–395, 1981. 42, 73

[21] Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989. 81

[22] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4:933–969, 2003. ix, 29, 30, 32, 57, 66

[23] Bernard A Galler and Michael J Fisher. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964. 81

[24] Hua Gao, Hazım Kemal Ekenel, and Rainer Stiefelhagen. Face alignment using a ranking model based on regression trees. 57

[25] C. W. Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998. 42

[26] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984. 74

[27] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 43, 46

[28] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 769–775, 2004.

[29] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-aided design of integrated circuits and systems, ieee transactions on*, 11(9):1074–1085, 1992. 14

[30] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988. 3

[31] Richard Hartley and René Vidal. The multibody trifocal tensor: Motion segmentation from 3 perspective views. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–769. IEEE, 2004. 11

[32] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 6, 7, 28, 34

[33] Jian Huang, Joel L. Horowitz, and Fengrong Wei. Variable selection in nonparametric additive models. *The Annals of Statistics*, 38(4):2282–2313, 2010. 61

[34] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 2, pages 586–591, 2001. 13, 50

[35] Q. Ke and T. Kanade. Robust L_1-norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 739–746, 2005.

[36] Scott Kirkpatrick, MP Vecchi, et al. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983. 80

[37] F. Lauer and C. Schnörr. Spectral clustering of linear subspaces for motion segmentation. In *International Conference on Computer Vision*, 2009. 11, 19, 34, 35, 41, 43, 46, 47, 50, 52, 56, 63, 70, 72, 76, 77, 82

[38] Richard B Lehoucq and Danny C Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996. 87

[39] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998. 87

[40] Fuxin Li, Joao Carreira, and Cristian Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *Conference on Computer Vision and Pattern Recognition*, pages 1712–1719, 2010.

[41] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*, 2010. 11, 19, 43, 46, 56, 72

[42] B.D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, pages 674–679. Citeseer, 1981. 9

[43] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9), 2007. 73

[44] Lukas Meier, Sara van de Geer, and Peter Bühlmann. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779–3821, 2009. 61

[45] Hans-Hellmut Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21(1):85–117, 1983. 7

[46] Hans-Hellmut Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial intelligence*, 33(3):299–324, 1987. 7

[47] Chris Needham and Roger Boyle. Performance evaluation metrics and statistics for positional tracker evaluation. In *Computer Vision Systems*, volume 2626, pages 278–289. 2003. 24

[48] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. ix, 15, 46

[49] J.H. Park, H. Zha, and R. Kasturi. Spectral clustering for robust motion segmentation. In *European Conference on Computer Vision*, pages 390–401, 2004. 46

[50] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010. 19, 41, 42, 47, 50, 52, 76, 77, 82

[51] Lafferty J. Liu H. Ravikumar, P. and L. Wasserman. Spam: Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71:1009–1030, 2009.

[52] Konrad Schindler, Hanzi Wang, et al. Perspective n-view multibody structure-and-motion through model selection. In *European Conference on Computer Vision*, 2006. 11

[53] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), August 2000. 14, 46

[54] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600. IEEE, 1993. 2, 6

[55] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 34

[56] Danny C Sorensen. Implicit application of polynomial filters in ak-step arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13(1):357–385, 1992. 87

[57] Y. Sugaya and K. Kanatani. Geometric structure of degeneracy for multi-body motion segmentation. In *Workshop on statistical methods in video processing*, 2004. 13, 42, 50

[58] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2432–2439, 2010. 8, 28, 34, 84

[59] Robert H Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58(2):86, 1987. 73

[60] R. Szeliski. *Computer vision: algorithms and applications*. Springer-Verlag New York Inc, 2010. 6

[61] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999. 73

[62] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Tech. Rept. CMU-CS-91132, Carnegie Mellon University, 1991. 9, 24, 28, 34

[63] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992. 12, 26, 41, 48

[64] Philip HS Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998. 11

[65] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*. Number 50. 1997. 76

[66] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 17, 19, 26, 34, 44, 66, 81

[67] Emanuele Trucco and Konstantinos Plakas. Video tracking: a concise survey. *Oceanic Engineering, IEEE Journal of*, 31(2):520–529, 2006. 2

[68] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005. 41, 42, 47, 50

[69] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using powerfactorizatin and GPCA. *International Journal of Computer Vision*, 79:85–105, 2008. 19

[70] René Vidal and Richard Hartley. Motion segmentation with missing data using power-factorization and gpca. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–310, 2004. 56, 72, 76

[71] René Vidal, Yi Ma, Stefano Soatto, and Shankar Sastry. Two-view multibody structure from motion. *International Journal of Computer Vision*, 68(1):7–25, 2006. 11

[72] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. 79

[73] H. Wang and P.F. Culverhouse. Robust motion segmentation by spectral clustering. In *Proc. of the British Machine Vision Conference*, pages 639–648, 2003. 46

[74] J. Yan and M. Pollefeys. A factorization approach to articulated motion recovery. In *IEEE conference on computer vision and pattern recognition*, volume II, pages 815–821, 2005. 13

[75] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106, 2006. 41, 43, 46, 56, 72

[76] F. Yin, D. Makris, S.A. Velastin, and J. Orwell. Quantitative evaluation of different aspects of motion trackers under various challenges. *British Machine Vision Association (5)*, pages 1–11, 2010. 24

[77] Jingdan Zhang, Shaohua Zhou, Dorin Comaniciu, and Leonard McMillan. Discriminative learning for deformable shape segmentation: A comparative study. *European Conference on Computer Vision*, pages 711–724, 2008. 57

[78] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1927–1934, 2010. 19, 43, 46

BIOGRAPHICAL SKETCH

Liangjing Ding completed his Bachelor's degree in Electronic Information Science and Technology at the University of Science and Technology of China. He obtained his Master's degree in Biomedical Engineering at the University of Science and Technology of China in 2008. His work was mainly involved in electrocardiography signal analysis and ultrasound imaging. He enrolled in the doctoral program of the Department of Scientific Computing at Florida State University in the fall of 2008. Mentored by Prof. Adrian Barbu and Prof. Anke Meyer-Baese. he worked on object tracking and motion segmentation.

Liangjing's research interests include computer vision, machine learning and data mining.