1

Scalable Subspace Clustering with Application to Motion Segmentation

Liangjing Ding

Adrian Barbu

CONTENTS

1.1 Introduction

Subspace clustering is the problem of grouping an unlabeled set of points into a number of clusters corresponding to subspaces of the ambient space. This problem has applications in unsupervised learning and computer vision. One of the computer vision applications is motion segmentation, where a number of feature point trajectories need to be grouped into a small number of clusters according to their common motion model. The feature point trajectories are obtained by detecting a number of feature points using an interest point detector and tracking them through many frames using a feature point tracker or an optical flow algorithm.

A common approach in the state of the art sparse motion segmentation methods [6][12][13][24][27] is to project the feature trajectories to a lower dimensional space and use a subspace clustering method based on 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 data points. In this manner, the computation time for these subspace clustering/motion segmentation methods scales as $O(N^3)$, so it can become prohibitive for large problems (e.g. $N = 10^5 - 10^6$).

This chapter proposes a completely different approach to subspace clustering, based on the Swendsen-Wang Cut (SWC) algorithm [2] and brings the following contributions:

- The subspace clustering problem is formulated as Maximum A Posteriori (MAP) optimization problem in a Bayesian framework with Ising/Potts prior [16] and likelihood based on a linear subspace model.
- The optimization problem is solved 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. This graph is constructed as a k-NN graph from an affinity matrix.
- The computation complexity of the SWC algorithm is evaluated and observed to scale as $O(N^2)$, making the proposed approach more scalable than spectral clustering (an $O(N^3)$ algorithm) to large scale problems.
- Motion segmentation experiments on the Hopkins 155 dataset are conducetd and the performance of the proposed algorithm is compared with the state of the art methods. The SWC obtains an error less than twice the error of the state of the art methods. The experiments obtain an observed scaling of about $O(N^{1.3})$ for the SWC and about $O(N^{2.5})$ for the spectral clustering.

Compared to other statistical methods [7, 14, 21], the proposed SWC method does not require a good initialization, which can be hard to obtain.

Overall, the proposed method provides a new perspective to solve the subspace clustering 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 $O(N^{1.3})$.

1.2 Subspace Clustering by Spectral Clustering

Given a set of points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \in \mathbb{R}^D$, the subspace clustering problem is to group the points into a number of clusters corresponding to linear subspaces of

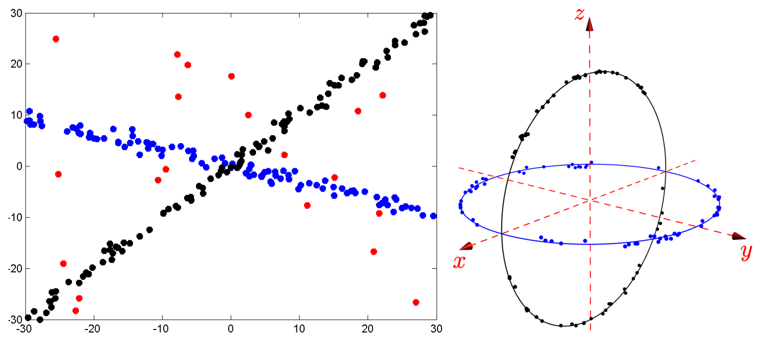

FIGURE 1.1
Left. two subspaces in 2D. Right. two 2D subspaces in 3D. The points in both 2D subspaces have been normalized to unit length. Due to noise, the points may not lie exactly on the subspace. One can observe that the angular distance finds the correct neighbors in most places except at the plane intersections.

$\mathbb{R}^D$. The problem is illustrated in Figure 1.1, left, showing two linear subspaces and a number of outliers in $\mathbb{R}^2$.

A popular subspace clustering method [5][12][17] is based on spectral clustering, which relies on an affinity matrix that measures how likely any pair of points belong to the same subspace.

Spectral clustering [15, 18] is a generic clustering method that groups a set of points into clusters based on their connectivity. The point connectivity is given as an $N \times N$ affinity matrix A with A_{ij} close to 1 if point i is close to point j and close to zero if they are far away. The quality of the affinity matrix is very important for obtaining good clustering results. The affinity matrix for spectral subspace clustering will be described below.

The spectral clustering algorithm proceeds by computing the matrix $S = L^{-1/2}AL^{-1/2}$, where L is a diagonal matrix with L_{ii} as the sum of row i of A. It then computes the k-leading eigenvectors of S and obtains points in $\mathbb{R}^k$ from the eigenvectors. The points are then clustered by k-means or other clustering algorithm. The number k of clusters is assumed to be given. The spectral clustering algorithm is described in detail in Figure 1.2.

```
Input: A set of points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^D$ and the number $k$ of clusters.
1. Construct the affinity matrix $A \in \mathbb{R}^{N\times N}$.
2. Construct the diagonal matrix $L$, with $L_{ii} = \sum_{j=1}^N A_{ij}$.
3..Compute $S = L^{-1/2}AL^{-1/2}$.
4. Compute the matrix $U = (u_1, \dots, u_k) \in \mathbb{R}^{n\times k}$ containing the leading $k$ eigenvectors of $S$.
5. Treat each row of $U$ as point in $\mathbb{R}^k$ and normalize them to unit length.
6. Cluster the $n$ points of $\mathbb{R}^k$ into $k$ clusters by $k$-means or other algorithm.
7. Assign the points $x_i$ to their corresponding clusters.
```

FIGURE 1.2
The spectral clustering algorithm [15].

Affinity Matrix for Subspace Clustering. A common practice [5][12][17] before computing the affinity matrix is to normalize the points to unit length, as shown in Figure 1.1, right.. Then the following affinity measure based on the angle between the vectors has been proposed in [12]

$$A_{ij} = (\frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2})^{2\alpha}, \tag{1.1}$$

where α is a tuning parameter, and the value $\alpha = 4$ has been used in [12].

Fig 1.1, right shows two linear subspaces, where all points have been normalized. It is intuitive to find that the points tend to lie in the same subspace as their neighbors in angular distance except those near the intersection of the subspaces.

1.3 Scalable Subspace Clustering by Swendsen-Wang Cuts

This section presents a novel subspace clustering algorithm that formulates the subspace clustering problem as a MAP estimation of a posterior probability in a Bayesian framework and uses the Swendsen-Wang Cuts algorithm [2] for sampling and optimization.

A subspace clustering solution can be represented as a partition (labeling) $\pi : \{1, ..., N\} \to \{1, ..., M\}$ of the input points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$. The number $M \leq N$ is the maximum number of allowed clusters.

In this section is assumed that an affinity matrix A is given, representing the likelihood for any pair of points to belong to the same subspace. One form of A has been given in (1.1) and another one will be given in section 1.3.3.

1.3.1 Posterior Probability

A posterior probability will be used to evaluate the quality of any partition π. A good partition can then be obtained by maximizing the posterior probability in the space of all possible partitions. The posterior probability is defined in a Bayesian framework

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

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

The data term $E_{data}(\pi)$ is based on the fact that the subspaces are assumed to be linear. Given the current partition (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

$$E_{data}(\pi) = \sum_{l=1}^{M} \sum_{i,\pi(i)=l} d(x_i, L_l) \tag{1.2}$$

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

$$E_{prior}(\pi) = -\rho \sum_{<i,j>\in E, \pi(i)\neq\pi(j)} \log(1 - A_{ij}), \tag{1.3}$$

where ρ is a parameter controlling the strength of the prior term. It will be clear in the next section that this prior is exactly the Potts model (1.4) that would have A_{ij} as the edge weights in the original SW algorithm.

1.3.2 Overview of the Swendsen-Wang Cuts Algorithm

The precursor of the Swenden-Wang Cuts algorithm is the Swendsen-Wang (SW) method [20], which 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 [16]

$$p(\pi) = \frac{1}{Z} \exp[- \sum_{<i,j>\in E} \beta_{ij} \delta(\pi(i) \neq \pi(j)]. \tag{1.4}$$

where $\beta_{ij} > 0, \forall < i, j >\in E$ and $N = |V|$.

The SW algorithm addresses the slow mixing problem of the Gibbs Sampler [10], 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 $e^{-\beta_{ij}}$. While the original SW method was developed originally for Ising and Potts models, the Swendsen-Wang Cuts (SWC) method [2] generalized SW to arbitrary posterior probabilities defined on graph partitions.

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, then 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 [2], there are different versions of the Swendsen-Wang Cut algorithm. The SWC-1 algorithm is used in this chapter, and is summarized in Figure 1.3.

The set of edges $\mathcal{C}(V_0, V_{l'} - V_0), \mathcal{C}(V_0, V_l - V_0)$ from eq. (1.5) 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 SWC algorithm could automatically decide the number of clusters,

```
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 $t = 1, \dots T$, current state $\pi$, 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 $V_l = V_{l1} \cup \dots \cup V_{ln_l}$ for $l = 1, 2, \dots, N$.
  4. Collect all the connected components in set $CP = \{V_{li} : l = 1, \dots, N, i = 1, \dots, N_l\}$.
  5. Select a connected component $V_0 \in CP$ with probability $q(V_0|CP) = \frac{1}{|CP|}$, 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 $\pi'$.
  7. Accept the proposed label assignment with probability
     $$\alpha(\pi \to \pi') = \min(1, \frac{\prod_{e\in\mathcal{C}(V_0,V_{l'}-V_0)}(1 - q_e)}{\prod_{e\in\mathcal{C}(V_0,V_l-V_0)}(1 - q_e)} \cdot \frac{q(c_{V_0} = l|V_0, \pi')}{q(c_{V_0} = l'|V_0, \pi)} \cdot \frac{p(\pi'|I)}{p(\pi|I)}. \quad (1.5)$$
end for
Output: Samples $\pi \sim p(\pi|I)$.
```

FIGURE 1.3
The Swendsen-Wang Cut algorithm [2].

however in this chapter, as in most motion segmentation algorithms, it is assumed that the number of subspaces M is known. Thus the new label for the component V_0 is sampled with uniform probability from the number M of subspaces:

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

1.3.3 Graph Construction

Section 1.2 described a popular subspace clustering method based on spectral clustering. Spectral clustering optimizes an approximation of the normalized cut or the ratio cut [25], which are discriminative measures. In contrast, the proposed subspace clustering approach optimizes a generative model where the likelihood is based on the assumption that the subspaces are linear. It is possible that the discriminative measures are more flexible and work better when the linearity assumption is violated, and will be studied in future work.

The following affinity measure, inspired by [12], will be used in this work

$$A_{ij} = \exp(-m\frac{\theta_{ij}}{\bar{\theta}}), \quad i \neq j \tag{1.6}$$

where θ_{ij} is based on the angle between the vectors x_i and x_j,

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

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. The subspace clustering performance with respect to this parameter will be evaluated in Section 1.4.2 for motion segmentation.

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.

The graph $G = (V, E)$ has as vertices the set of points that need to be clustered. The edges E are constructed based on the proposed distance measure from eq. (1.6). 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.

Examples of obtained graphs will be given in Section 1.4.2.

```
Input: N points $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$ from $M$ subspaces
Construct the adjacency graph G as a k-NN graph using eq (1.6).
for $r = 1, \dots, Q$ do
  Initialize the partition $\pi$ as $\pi(i) = 1, \forall i$.
  for $i = 1, \dots, N^{it}$ do
    1. Compute the temperature $T_i$ using eq (1.7).
    2. Run one step of the SWC algorithm 1.3 using $p(\pi|I) = p^{1/T_i}(\pi)$ in
    eq (1.5).
  end for
  Record the clustering result $\pi_r$ and the final probability $p_r = p(\pi_r)$.
end for
Output: Clustering result $\pi_r$ with the largest $p_r$.
```

FIGURE 1.4
The Swendsen-Wang Cuts algorithm for subspace clustering.

1.3.4 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 [11] 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

$$T_i = \frac{T_{\text{end}}}{\log\left(\frac{i}{N}[e - \exp(\frac{T_{\text{end}}}{T_{\text{start}}})] + \exp(\frac{T_{\text{end}}}{T_{\text{start}}})\right)}, i = \overline{1, N^{it}} \tag{1.7}$$

To better explore the probability space, we also use multiple runs with different random initializations. Then the final algorithm is shown in Figure 1.4.

1.3.5 Complexity Analysis

This section presents an evaluation of the computation complexity of the proposed SWC-based subspace clustering algorithm. To our knowledge, the complexity of SWC has not been calculated yet in the literature.

Let N be the number of points in $\mathbb{R}^D$ that need to be clustered. The computation complexity of the proposed subspace clustering method can be broken down as follows:

- The adjacency graph construction is $O(N^2 D \log k)$ where D is the space dimension. This is because one needs to calculate the distance from each point 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 [9, 8]. 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 $O(D^2 N + D^3)$
  - 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.

In conclusion, the entire algorithm complexity in terms of the number N of points is $O(N^2)$ so it scales better than spectral clustering for large problems.

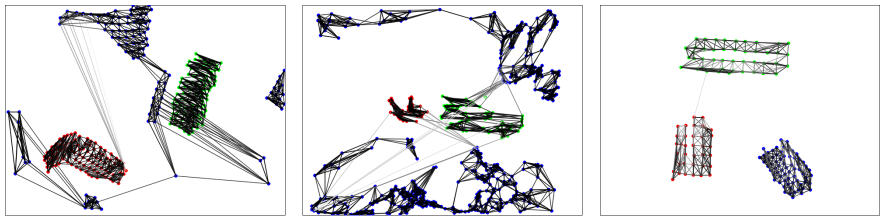
(a) 1RT2TC (b) cars3 (c) articulated

FIGURE 1.5
Examples of SWC weighted graphs for a checkerboard (left), traffic (mid) and articulated (right) sequence. Shown are the feature point positions in the first frame. The edge intensities represent their weights from 0 (white) to 1 (black).

1.4 Application: Motion Segmentation

This section presents an application of the proposed subspace clustering algorithm to motion segmentation.

Most recent works on motion segmentation 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 3D point X by

$$\begin{bmatrix} x \\ y \end{bmatrix} = A \begin{bmatrix} X \\ 1 \end{bmatrix}, \tag{1.8}$$

where $A \in \mathbb{R}^{2\times 4}$ is the affine motion matrix.

Let $t_i = (x_i^1, y_i^1, x_i^2, y_i^2, \ldots, x_i^F, y_i^F)^T, i = 1, \ldots. N$ be the trajectories of tracked feature points in F frames (2D images), where N is the number of trajectories. Let the *measurement matrix* $W = [t_1, t_2, \ldots, t_N]$ be constructed by assembling the trajectories as columns.

If all trajectories undergo the same rigid motion, equation (1.8) implies that W can be decomposed into a *motion matrix* $M \in \mathbb{R}^{2F\times 4}$ and a *structure matrix* $S \in \mathbb{R}^{4\times N}$ as

$$W = MS$$

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ y_1^1 & y_2^1 & \cdots & y_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \cdots & x_N^F \\ y_1^F & y_2^F & \cdots & y_N^F \end{bmatrix} = \begin{bmatrix} A^1 \\ \vdots \\ A^F \end{bmatrix} \begin{bmatrix} X_1 & \cdots & X_N \\ 1 & \cdots & 1 \end{bmatrix}$$

where A^f is the affine object to world transformation matrix at frame f. It implies that $\text{rank}(W) \leq 4$. Since the entries of the last row of S are always 1, under the affine camera model, the trajectories of feature points from a rigidly moving object reside in an affine subspace of dimension at most 3.

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 [5][12][17][24] is to project the trajectories to a lower dimensional space and to perform subspace clustering in that space, using spectral clustering as described in section 1.2. These methods differ in the projection dimension D and in the affinity measure A used for spectral clustering.

1.4.1 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 [5, 12, 17, 24].

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 major 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.

The GPCA [24] suggests to project the trajectories onto a 5-dimensional space. ALC [17] chooses to use the sparsity-preserving dimension $d_{sp} = \text{argmin}_{d \geq 2D\log(2T/d)} d$ for D-dimensional subspaces. The SSC [6] and LRR [13] simply projects the trajectories to the $4M$ subspace, where M is the number of motions. Some methods [5, 12] use an exhaustive search strategy to perform the segmentation in spaces with a range of possible dimensions and pick the best result. 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 U when $m >> n$ is $O(mn^2 + n^3)$ [22]. If $n >> m$ then it is faster to compute the SVD of U^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.

After projecting to the subspace of dimension $D = 2M + 1$, the SWC subspace clustering algorithm from Section 1.3 is applied and the clustering result gives the final motion segmentation result.

1.4.2 Experiments on the Hopkins 155 Dataset

This section presents experiments with the proposed SWC-based motion segmentation algorithm on the Hopkins 155 motion database [23]. The database has been created with the goal of providing an extensive benchmark for testing feature based motion segmentation algorithms. It consists of 155 sequences of two and three motions. The ground-truth segmentation is also provided for evaluation purposes. Based on the content of the video, the sequences could be categorized into three main categories: checkerboard, traffic, and articulated sequences, with examples shown in Figure 1.6. The trajectories are extracted automatically by a tracker, so they are slightly corrupted by noise.

As already mentioned, before applying the SWC algorithm, the dimension of the data is reduced from $2F$ to $D = 2M + 1$, where M is the number of motions. After the projection, the initial labeling state in the SWC algorithm has all points having the same label.

The motion segmentation results are evaluated using the misclassification error rate

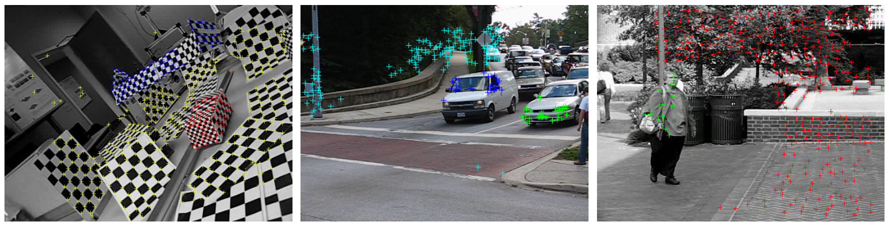
(a) Checkerboard (b) Traffic (c) Articulated

FIGURE 1.6
Sample images from some sequences of three categories in the Hopkins 155 database with ground truth superimposed.

$$\text{Misclassification Rate} = \frac{\text{\# of misclassified points}}{\text{total \# of points}} \tag{1.9}$$

Parameter settings. The proposed motion segmentation algorithm has a number of tuning parameters that were held constant to the following values. The number of NN (nearest neighbors) for graph construction is $k = 7$, the parameter m in the affinity measure (1.6) is $m = 10$, and the prior coefficient in (1.3) is $\rho = 2.2$. The sensitivity of the average misclassification error to the parameters m and ρ is shown in Figure 1.7. 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 $Q = 10$. An example of all the partition states during an SWC run is shown in Figure 1.8.

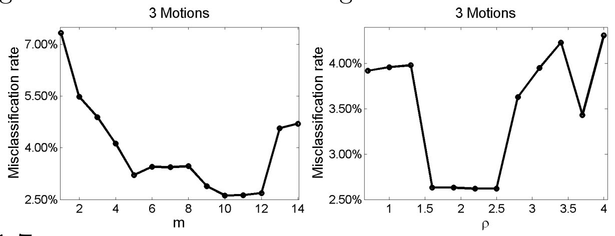

FIGURE 1.7
Dependence of the misclassification rate on the affinity parameter m (left) and prior strength ρ (right).

Results. The average and median misclassification errors are listed in Table 1.1. For accuracy, the results of the SWC algorithm from Table 1.1 are averaged over 10 runs and the standard deviations are shown in parentheses. In order to compare the SWC method with the state of the art methods, we also list the results of ALC [17], SC [12], SSC [6] and VC [5].

The SWC based algorithm 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 needs to be improved.

Also shown in Table 1.1 are the columns labeled SC^4 and SC^{4k} representing the misclassification errors of the SC method [12] with an affinity matrix with

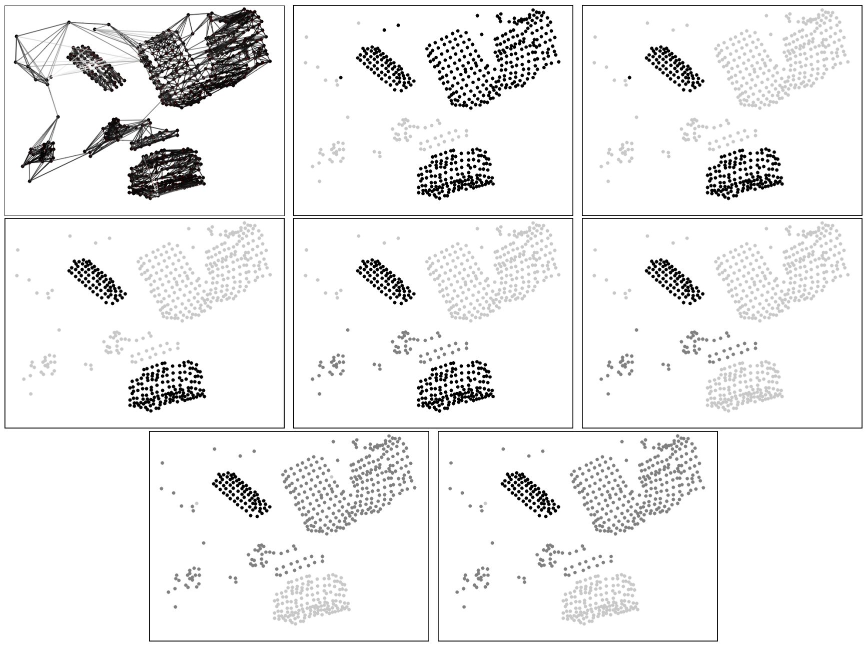

FIGURE 1.8
SWC clustering of the Hopkins 155 sequence 1R2TCR, containing $M = 3$ motions. Shown are the feature point positions in the first frame, having colors as the labeling states π obtained while running the SWC algorithm from the initial state (top left) to the final state (bottom right).

4-NN and 4k-NN respectively. These errors are 13.32 and 8.59 respectively and indicate that the Spectral Clustering really needs a dense affinity matrix to work well, and it cannot be accelerated using sparse matrix operations.

Finally, the performance of the SWC-based algorithm is compared with the KASP algorithm [26], which is a fast approximate spectral clustering and was used in place of the spectral clustering step in the SC method [12]. The data reduction parameter used was $\gamma = 10$, which still results in a $O(N^3)$ clustering algorithm. The total misclassification error is 5.72, about three times larger than the SWC method.

1.4.3 Scalability Experiments on Large Data

In order to evaluate the scalability of different algorithms, sequences with a large number of trajectories are needed. The trajectories can be generated by some optical flow algorithm, but it is difficult to obtain the ground truth segmentation and remove bad trajectories caused by occlusions. Brox et.al. [3] provided a dense segmentation for some frames in 12 sequences in the Hopkins 155 dataset[1]. From them, we picked the cars10 sequence and tracked all pixels

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

TABLE 1.1
Misclassification rates (in percent) of different motion segmentation algorithms on the Hopkins 155 dataset.

Method	ALC	SC	SSC	VC	SWC (std)	SC^4	SC^{4k}	KSP
Checkerboard (2 motion)								
Average	1.55	0.85	1.12	0.67	1.25 (0.11)	10.98	5.25	1.25
Median	0.29	0.00	0.00	0.00	0.00 (0.00)	1.16	0.00	0.00
Traffic (2 motion)								
Average	1.59	0.90	0.02	0.99	1.87 (0.55)	12.85	12.48	8.65
Median	1.17	0.00	0.00	0.22	0.00 (0.00)	7.55	1.30	0.53
Articulated (2 motion)								
Average	10.70	1.71	0.62	2.94	2.15 (0.11)	11.38	12.94	18.67
Median	0.95	0.00	0.00	0.88	0.00 (0.00)	0.95	0.95	0.95
All (2 motion)								
Average	2.40	0.94	0.82	0.96	1.49 (0.19)	11.50	7.82	4.76
Median	0.43	0.00	0.00	0.00	0.00 (0.00)	2.09	0.27	0.00
Checkerboard (3 motion)								
Average	5.20	2.15	2.97	0.74	2.26 (0.04)	16.74	6.58	4.86
Median	0.67	0.47	0.27	0.21	0.67 (0.00)	14.82	0.49	1.46
Traffic (3 motion)								
Average	7.75	1.35	0.58	1.13	2.88 (0.00)	30.04	24.87	21.80
Median	0.49	0.19	0.00	0.21	0.81 (0.00)	26.82	30.91	26.36
Articulated (3 motion)								
Average	21.08	4.26	1.42	5.65	6.33 (1.88)	19.48	24.27	18.42
Median	21.08	4.26	0.00	5.65	6.33 (1.88)	19.48	24.27	18.42
All (3 motion)								
Average	6.69	2.11	2.45	1.10	2.62 (0.13)	19.55	11.25	9.00
Median	0.67	0.37	0.20	0.22	0.81 (0.00)	18.88	1.42	1.70
All sequences combined								
Average	3.37	1.20	1.24	0.99	1.75 (0.15)	13.32	8.59	5.72
Median	0.49	0.00	0.00	0.00	0.00 (0.00)	6.46	0.36	0.31

of the first frame using the Classic+NL method [19]. There are two reasons for choosing cars10. First, it has three motions, two moving cars and the background. Second, the two moving cars are relatively large in the video, so that a large number of trajectories can be obtained from each motion.

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. Finally, we obtained around 48,000 trajectories as a pool. From the pool, different numbers N of trajectories were 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

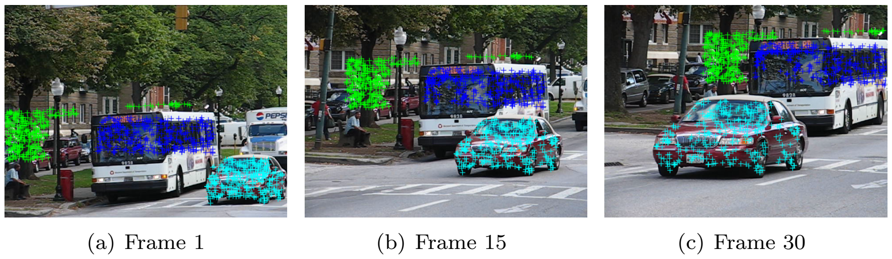
(a) Frame 1 (b) Frame 15 (c) Frame 30

FIGURE 1.9
Selected frames of sequence cars10 with 1000 tracked feature points.

334 trajectories from the third motion. If there were not enough trajectories available from one motion, we added more from the motion that has the most trajectories.

We compared the SWC method with the SC algorithm [12] discussed in Section 1.2, which is one of the fastest and most accurate algorithms [5] based on spectral clustering. We generated data containing between 1,000 to 15,000 trajectories, and applied the two segmentation algorithms. Sample frames are shown in Figure 1.9. The parameters for SC were kept the same as in the original paper, and those for SWC were identical with Section 1.4.2. 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 $N = 24,000$ and $N = 48,000$ trajectories for SWC clustering. For SC, the same experiments could not be conducted because Matlab ran out of memory.

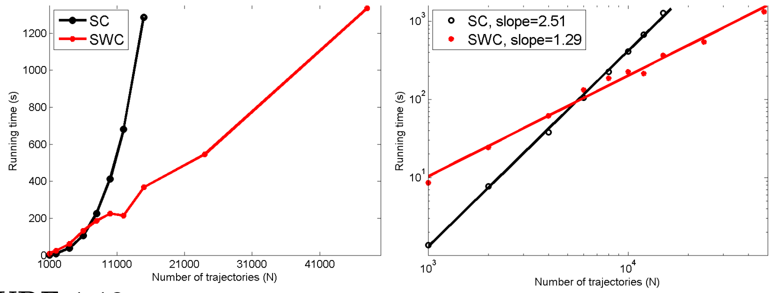

FIGURE 1.10
Left. Computation time (sec) vs number of trajectories N for SC and SWC. Right: log-log plot of same data with the fitted regression lines.

The misclassification rate is recorded in Table 1.2 and the running time is shown on Figure 1.10. Table 1.2 shows that both methods perform well and the misclassification rate of SWC is about one third of that of the SC.

From Figure 1.10, 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 $N = 6,000$ 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 the computation complexity is $O(N^{\alpha})$. 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 1.3.5.

TABLE 1.2
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

1.5 Conclusion

This chapter presented a stochastic method for clustering points belonging to a number of linear subspaces using the Swendsen-Wang Cuts (SWC) algorithm. The posterior probability is a generative model defined in a Bayesian framework with data and likelihood parts. The graph used by the SWC algorithm for making informed relabeling proposals was defined as the k-NN graph based on an affinity matrix.

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

Motion segmentation experiments on the Hopkins 155 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 $O(N^{2.5})$ on the data that was evaluated while the proposed SWC method is $O(N^{1.3})$.

The SWC algorithm uses many Markov Chains to better explore the partition space. These chains are independent and could be run in parallel if desired. Parallel implementations of Spectral Clustering have also been developed recently [4].

In the future we will investigate posterior probabilities based on the Normalized Cut or Ratio Cut instead of the generative model to see if the Swendsen-Wang Cuts algorithm can obtain state of the art errors using such discriminative models.

Bibliography

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

[2] A. Barbu and S. Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Trans. PAMI*, 27(8):1239–1253, 2005.

[3] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, pages 282–295. 2010.

[4] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE Trans. on PAMI*, 33(3):568–586, 2011.

[5] L. Ding, A. Barbu, and A. Meyer-Baese. Motion segmentation by velocity clustering with estimation of subspace dimension. In *ACCV Workshop on Detection and Tracking in Challenging Environments*, 2012.

[6] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.

[7] 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.

[8] 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.

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

[10] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. PAMI*, (6):721–741, 1984.

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

[12] F. Lauer and C. Schnörr. Spectral clustering of linear subspaces for motion segmentation. In *ICCV*, 2009.

[13] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, 2010.

[14] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Trans. on PAMI*, 29(9), 2007.

[15] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *NIPS*, 14:849–856, 2001.

[16] R. B. Potts. Some generalized order-disorder transformations. In *Proc. of the Cambridge Phil. Soc.*, volume 48, pages 106–109, 1952.

[17] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Trans. on PAMI*, 32(10):1832–1845, 2010.

[18] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, 2000.

[19] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439, 2010.

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

[21] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *JRSS: Series B*, 61(3):611–622, 1999.

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

[23] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *CVPR*, pages 1–8. IEEE, 2007.

[24] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *CVPR*, pages II–310, 2004.

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

[26] Donghui Yan, Ling Huang, and Michael I Jordan. Fast approximate spectral clustering. In *SIGKDD*, pages 907–916, 2009.

[27] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV*, pages 94–106, 2006.

Index