FLoSS: Facility Location for Subspace Segmentation

Nevena Lazic
nevena@comm.utoronto.ca

Inmar Givoni
inmar@psi.utoronto.ca

Brendan Frey
frey@psi.utoronto.ca

Parham Aarabi
parham@ecf.utoronto.ca

University of Toronto, Dept. of Electrical and Computer Engineering
10 Kings College Road, Toronto ON, Canada, M5S 3G4

Abstract

Subspace segmentation is the task of segmenting data lying on multiple linear subspaces. Its applications in computer vision include motion segmentation in video, structure-from-motion, and image clustering. In this work, we describe a novel approach for subspace segmentation that uses probabilistic inference via a message-passing algorithm.

We cast the subspace segmentation problem as that of choosing the best subset of linear subspaces from a set of candidate subspaces constructed from the data. Under this formulation, subspace segmentation reduces to facility location*, a well studied operational research problem. Approximate solutions to this NP-hard optimization problem can be found by performing maximum-a-posteriori (MAP) inference in a probabilistic graphical model. We describe the graphical model and a message-passing inference algorithm.*

We demonstrate the performance of Facility Location for Subspace Segmentation, or FLoSS, on synthetic data as well as on 3D multi-body video motion segmentation from point correspondences.

1. Introduction

Many statistical models used for data analysis in vision assume that high-dimensional input data has an intrinsic low-dimensional representation. Furthermore, many such models assume the data can be well approximated as lying on a linear subspace. For instance, principal component analysis (PCA) [13], independent component analysis [12], factor analysis [10], and nonnegative matrix factorization [16] are all highly popular methods that attempt to recover a low dimensional linear representation of the data. Although the linearity assumption is often inaccurate, it nevertheless turns out to be a reasonable and useful approximation in many cases [25, 26, 30]. Even in non-linear dimensionality reduction, many methods assume that data is locally linear, and can be represented as some configuration of local linear subspaces [19, 31].

In *subspace segmentation*, the underlying assumption is that the data is composed of points lying on several distinct linear subspaces, not necessarily of the same intrinsic dimension. The goal of subspace segmentation is to recover the underlying subspaces and to assign the data points to one subspace each. The toy data sets shown in Fig. 1 illustrate this idea. Thus, subspace segmentation is a more flexible model compared to the single linear subspace representation, but it still retains some of the computationally favorable properties of linear subspace models.

Subspace segmentation has a variety of computer vision applications. One example is clustering images of different objects under varying illumination. It has been shown in [11] that a set of images of a Lambertian object under varying lighting conditions forms a convex polyhedral cone in the image space, which is well-approximated by a low dimensional subspace. As images of different objects lie on different subspaces, subspace segmentation can be used for clustering images. Another application is in 3D multi-body video motion segmentation from point correspondences. Given the image coordinates of several keypoints lying on a rigid object, undergoing motion over F video frames, it can be shown that vectors (of length $2F$) of stacked point coordinates lie on a linear subspace of dimension 2, 3 or 4 [24, 29]. When there are several moving objects in the video, with tracked keypoints on each object, the motion segmentation task is to cluster these points - another instance of subspace segmentation.

In this work, we describe a novel subspace segmentation method called Facility Location for Subspace Segmentation (FLoSS), where we formulate subspace segmentation as an instance of the *facility location* problem - a classical NP-hard problem in combinatorial optimization and operational

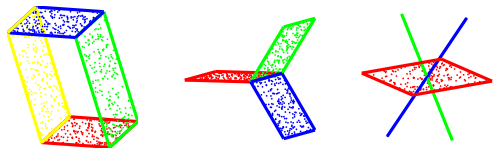

Figure 1. Examples of data lying on multiple linear subspaces

research [17]. The facility location problem can be stated as follows: given a set of customers, a set of potential facilities that can be opened to serve customers, the cost of opening each facility, and the distances between customers and facilities, open a subset of facilities and assign customers to one facility each, such that the sum of facility costs and customer-facility distances is minimized.

FLoSS formulates subspace segmentation as facility location by first constructing a large initial set of candidate subspaces (or facilities), and assigning costs to them that are based on their complexity (dimensionality). The candidate subspaces are initialized from the data by randomly selecting D-tuples of linearly independent points, with $2 \leq D \leq \mathcal{D}$, where $\mathcal{D}$ is the original data dimension. Each data D-tuple defines a linear subspace of dimension $(D-1)$. Given the normal distances of data points to candidate subspaces and the costs for utilizing subspaces, subspace segmentation is framed as finding the optimal subset of subspaces that best explains the data, *i.e*. the subset that minimizes the subspace costs and the point-subspace distances - an instance of facility location.

We find approximate facility location solutions by using a message-passing algorithm on a factor-graph representation of the problem. The approach is closely related to Affinity Propagation [8], an exemplar-based clustering algorithm.

2. Previous work

There exist numerous notable subspace segmentation algorithms, having different underlying approaches to the problem. When the number of subspaces is unknown, a sensible approach is to search for them one at a time, and select the one that represents a large number of points well at each pass. One such algorithm is random sample consensus (RANSAC) [6, 23, 28], a generic algorithm for outlier detection. RANSAC fits a $(D-1)$-dimensional subspace by iteratively (1) constructing a basis from D randomly sampled points, (2) computing the normal distance from all points to this subspace, and (3) labeling those above some distance threshold as outliers. This is repeated until a specified number of inliers is reached, or a sufficient number of points have been sampled. Multiple subspaces are found iteratively, by removing the inliers from the previous step and repeating. A similar idea - that of iteratively searching for a subspace with the most inliers - is used by Da Silva *et al*. [3]. They formulate this task as an unconstrained, but non-convex optimization problem, with improved efficiency over RANSAC. Neither method provides a direct way of estimating subspace dimensionalities. One proposed solution is to start with the highest-dimensional model, and recursively check each found solution for lower-dimensional models [2]. An alternative is to simultaneously apply the algorithm on multiple hypotheses and use model selection [7, 20].

When the number of subspaces and their dimensionalities are specified, it is more intuitive to determine all subspaces at once. One approach is to iterate between assigning points to their nearest subspaces, and re-estimating the subspace bases from the assigned points. k-subspaces [11], an extension of the k-means algorithm, iterates between making hard assignments of points to subspaces based on minimal point-subspace normal distance, and re-computing the subspace bases using PCA. Mixture of pPCA (mpPCA) [22] makes this process probabilistic by using latent variables to indicate the assignment of each point to one of k probabilistic PCA models. The model parameters and the probability distribution over the latent variables are estimated iteratively, using the Expectation Maximization (EM) algorithm [4]. Both methods can be sensitive to initialization and local optima.

Another possible approach, when the subspace number and dimensionality are available, is to construct the solution algebraically. Generalized PCA (GPCA) [27] represents a union of k subspaces embedded in $\Re^{\mathcal{D}}$ by a set of homogeneous polynomials of degree k in $\mathcal{D}$ variables. The polynomial coefficients can be estimated linearly from the data. The complexity of GPCA scales as $k^{\mathcal{D}}$, and the number of data points needed to estimate polynomials is exponential in k; hence, it is only practical for a small number of low-dimensional subspaces. When the number of subspaces is unavailable, the authors determine it by estimating the rank of a matrix. A recursive approach similar to [2] can be used when subspace dimensionalities are unknown.

Subspace separation (SS) [14] is also an algebraic approach. It relies on the observation that when the subspaces are linearly independent and noise-free, it is possible to compute a binary data interaction matrix, indicating whether two points lie on the same subspace or not. Noise is addressed by using model selection to decide whether to merge subspaces.

Overall, none of the methods provide an effective way of estimating the number of subspaces and their dimensionalities. However, there exist applications in which subspace structures are known beforehand, the most notable being *motion segmentation*. In motion segmentation, the subspace dimensionalities are 2, 3, or 4, and the possible challenges posed by the data are well understood. Indeed, many sub-

space segmentation methods were actually designed as motion segmentation algorithms [14, 21, 29].

The multi-stage learning (MSL) algorithm of [21] for motion segmentation refines the subspace segmentation results of SS using three stages of mpPCA of increasing complexity, each corresponding to a different type of motion. The simplest mpPCA model is initialized using SS, and the results at each stage are used to initialize the next stage. In this way, MSL accounts for the cases where SS fails, namely, when the subspaces are co-dependent. This can occur frequently in motion data, especially when the motion of the points is in part due to a moving camera.

Another multi-body motion segmentation method is local subspace affinity (LSA) [29]. It is an algebraic method that first projects points onto the first R principal components and then onto a hyper-sphere $\mathcal{S}^{R-1}$. A local subspace is fit around each point and its k nearest neighbors. The points are then clustered using spectral clustering [18] with pairwise similarities computed using angles between the local subspaces. Misclassification can occur near the intersection of two subspaces (as the nearest neighbors lie on different subspaces), or when the nearest neighbors do not span the selected subspace. Model selection is used to select appropriate subspace dimensionality, which is 2, 3 or 4.

In comparison to previous methods, FLoSS is essentially an iterative method that constructs subspaces from randomly sampled D-tuples of data, similarly to RANSAC. However, it considers all constructed subspaces (of possibly different dimensionality) simultaneously, and selects all k subspaces at once. Its complexity does not depend on the dimensionality of the original data; however, it increases with the total number of subspaces provided at initialization. FLoSS requires subspace dimensionalities or their range as inputs, and discovers the number of subspaces automatically.

3. A graphical model for subspace segmentation

3.1. Problem setup

Given N data points, we begin by creating a large set of M candidate subspaces by randomly drawing sets of $D << N$ linearly independent points for different values of D where each set defines a $(D-1)$-dimensional candidate subspace. We evaluate d_{nm}, the squared normal distances from point n to subspaces $m \in M$. In addition, we associate a cost c_m with each subspace m, which is set to be the sum of all pairwise distances between points defining the subspace. The purpose of assigning costs is to prevent overfitting the data with very high-dimensional subspaces or with too many subspaces. Using the sum of all pairwise distances is a sensible way of setting costs since it assigns lower costs to lower-dimensional subspaces and to subspaces generated by points that are close to one another.

Having generated the set of candidate subspaces, we would now like to select a subset $\mathcal{M} \subset M$ of subspaces, and associate each point with one subspace in $\mathcal{M}$, such that the sum of normal distances from points to their assigned subspaces and the total cost of subspaces in $\mathcal{M}$ is minimized. This optimization problem can be seen as an instance of *facility location* (FL), a well studied NP-hard problem. In FL, given a set of potential facilities (subspaces, in our case), the goal is to select an optimal subset of facilities and assign customers (data points) to one facility each such that the sum of facility costs and the distance between customers to their assigned facilities is minimized.

Let x_{nm} be a binary indicator variable, equal to 1 if point n is assigned to subspace m and 0 otherwise, and let $\mathbf{x} = \{x_{11}, \ldots, x_{nm}\}$ be a vector of all the x_{nm} variables . The FL optimization problem can be stated as:

$$\min_{\mathbf{x}} \quad \sum_m \sum_n d_{nm} x_{nm} + \sum_{m \in \mathcal{M}} c_m \tag{1}$$

$$\text{subject to} \quad \sum_m x_{nm} = 1 \; \forall n \tag{2}$$

$$\mathcal{M} = \{m | \sum_n x_{nm} > 0\} \tag{3}$$

$$x_{nm} \in \{0, 1\} \tag{4}$$

The constraint (3) ensures that each point is assigned to exactly one subspace.

3.2. Factor-graph representation and the max-sum algorithm

The FL problem can be described in terms of a probabilistic graphical model where each x_{nm} is treated as a hidden binary random variable. The graphical model for the problem is shown in Fig. 2, where we have used a factor-graph notation [15]. Recall that a factor-graph is a bipartite graph consisting of variable nodes and factor nodes. The factor nodes evaluate potential functions over the variable nodes they are connected to. The probability distribution described by the graph is proportional to the product of the factor potentials. The potential functions we associate with the graphical model incorporate facility costs and customer-facility distances, and enforce the constraint on the x_{nm} variables.

Given the following functions,

$$h_{nm}(x_{nm}) = -d_{nm}x_{nm} \quad (5)$$

$$f_m(x_{1m},\ldots,x_{Nm}) = \begin{cases} -c_m, & \sum_n x_{nm} > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

$$g_n(x_{n1},\ldots,x_{nM}) = \begin{cases} 0, & \sum_m x_{nm} = 1 \\ -\infty, & \text{otherwise,} \end{cases} \quad (7)$$

the joint probability distribution can be written as

$$p(\mathbf{x}) \propto \prod_{m,n} \exp(h_{nm}(x_{nm})) \quad (8)$$

$$\times \prod_{m} \exp(f_m(x_{1m},\ldots,x_{Nm})) \quad (9)$$

$$\times \prod_{n} \exp(g_n(x_{n1},...,x_{nM})) \quad (10)$$

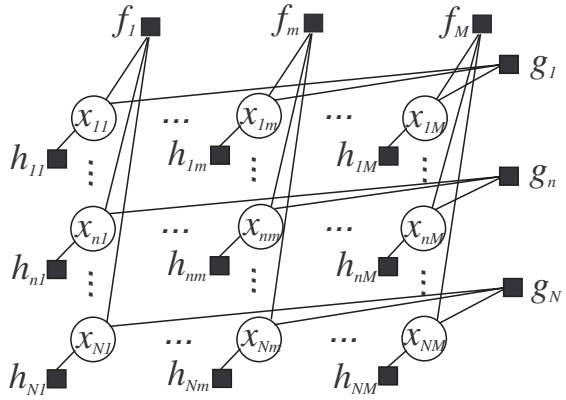

Figure 2. Factor graph representation of $p(\mathbf{x})$

The functions h_{nm} and f_m account for distances and costs, respectively, while the functions g_n enforce the constraint that each point is assigned to exactly one subspace. Maximizing $\log p(\mathbf{x})$ corresponds to the FL optimization problem stated in Equation 1. Finding a solution to the FL problem is carried out by finding maximum-a-posteriori (MAP) estimates for the x_{nm} using the max-product (belief propagation) algorithm [15]. Max-product is a local message passing algorithm known to converge to the MAP values of the variables on cycle-free graphs, and empirically observed to give good results on graphs with cycles, as the one described here. For notational convenience as well as computational stability, we use the log-domain version of the algorithm, max-sum. The general form of the max-sum local messages between a function f and a variable x is [1]:

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_K} [f(x,x_1,\ldots,x_K) \quad (11)$$

$$+ \sum_{x_i \in ne(f)\setminus x} \mu_{x_i\to f}(x_i)] \quad (12)$$

$$\mu_{x\to f}(x) = \sum_{f_l \in ne(x)\setminus f} \mu_{f_l\to x}(x) \quad (13)$$

The messages are passed iteratively between function to nodes and nodes to functions, and the algorithm is said to converge once the message values no longer change. The final variable assignment is based on the sum of all incoming messages to a variable:

$$x^* = \arg\max_x \sum_{f_l \in ne(x)} \mu_{f_l\to x}(x) \quad (14)$$

When the messages are functions of binary random variables (as in the factor graph in Fig. 2), they are of length two. However, in practice it suffices to only pass the difference between the two values, $\mu = \mu(1) - \mu(0)$. For the graph of Fig. 2, it can be shown that these messages are:

$$\mu_{h_{nm}\to x_{nm}} = -d_{nm} \quad (15)$$

$$\mu_{x_{nm}\to g_n} = \mu_{f_m\to x_{nm}} - d_{nm} \quad (16)$$

$$\mu_{x_{nm}\to f_m} = \mu_{g_n\to x_{nm}} - d_{nm} \quad (17)$$

$$\mu_{g_n\to x_{nm}} = -\max_{l\neq m} \mu_{x_{nl}\to g_n} \quad (18)$$

$$\mu_{f_m\to x_{nm}} = \min[0, -c_m + \sum_{l\neq n} \max(0, \mu_{x_{lm}\to f_m})] \quad (19)$$

To find the MAP assignment, we only need to compute the messages received by each x_{mn}. By substituting in, we can reduce the number of message updates to only these two types:

$$\eta_{nm} \equiv \mu_{g_n\to x_{nm}} = \max_{l\neq m}(-d_{nl} + \alpha_{nl}) \quad (20)$$

$$\alpha_{nm} \equiv \mu_{f_m\to x_{nm}} = \min[0, -c_m + \sum_{l\neq n} \max(0, \eta_{lm} - d_{nm})] \quad (21)$$

We also note that all the η_{mn} and α_{mn} message updates can be done in a parallel fashion, i.e. without looping over m and n. At convergence, we calculate the variable assignments as

$$x^*_{nm} = \begin{cases} 1 & [\eta_{nm} + \alpha_{nm} - d_{nm}] > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (22)$$

3.3. Relationship to affinity propagation

Affinity propagation (AP) [8] is an exemplar-based clustering algorithm that selects data exemplars using local message passing. It has been shown in [9] that affinity propagation can be represented using a factor graph that is similar to Fig. 2, having binary random variables indicating the membership of points to clusters. The differences between AP and FL are in that (1) in AP, the available cluster centers are data points (as opposed to general facilities), and (2) there is an additional constraint: if a point i chooses j as its exemplar, j must also choose itself as its exemplar. A somewhat different derivation of facility location as an instance of affinity propagation has been been applied in the past to computational biology problems [5].

4. Experiments

We evaluate the performance of FLoSS on both synthetic and real data sets. We use synthetic data to illustrate and compare the performance of the subspace segmentation methods FLoSS, RANSAC, mpPCA and GPCA on different types of data. We then apply FLoSS to rigid body motion segmentation from video, and compare it both to the above mentioned subspace segmentation methods, as well as to the MSL and LSA motion segmentation algorithms.

We note that the number of subspaces k is not an input to FLoSS. This is due to the FL formulation where the number of subspaces is automatically determined as a trade-off between distances and costs. The value of k can be controlled indirectly by changing the value of the costs c_m; in general, lower costs will result in selecting a larger number of subspaces to use from the set of potential subspaces. To compare FLoSS to methods that specify the number of underlying subspaces k, we use the following procedure to adjust the costs so that FLoSS finds exactly k subspaces

- If $k_{FL} < k$, change costs to $c'_m = 0.75c_m$ and run again.
- If $k_{FL} > k$, iteratively merge subspaces until $k_{FL} = k$. To decide on which subspace to merge, we form a $k_{FL} \times k_{FL}$ matrix F whose entry F_{ij} is the average normal distance of points assigned to subspace i from subspace j; $F_{ij} = \sum_n x_{ni} d_{nj} / \sum_n x_{ni}$. We find the smallest F_{ij} such that $i \neq j$, and merge the corresponding subspaces.

4.1. Synthetic data

We first investigate the case of subspaces of the same dimensionality. We generate several synthetic data sets by sampling data points from planes in $\Re^3$, and adding orthogonal Gaussian noise with variance at 5% of data variance. For all algorithms, we specify the number of subspaces k and their dimensionality. The segmentation results are shown in Fig. 3, where the colors indicate subspace membership.

In general, RANSAC does not give very good results, and its performance mainly depends on the number of iterations. mpPCA performs well on most data sets. However, as it is geared towards modeling mixtures of linear *segments* rather than infinite subspaces, it may assign two disjoint pieces of the same subspace to different mixture components, as illustrated in the top row of Fig. 3. In addition, mpPCA can have difficulties distinguishing linear segments that overlap close to their means, as is the case for the data shown in the middle row of Fig. 3. Although GPCA gives good results on a variety of subspace configurations, its performance degrades as the number of subspaces k increases since the number of data points needed to estimate subspaces is exponential in k. This explains its poor performance on the 4-plane data set in the bottom row of Fig. 3. GPCA can also be susceptible to noise; in fact, as noted in [27], it is suboptimal compared to the other algorithms in the Gaussian noise case when $k > 1$.

FLoSS gives very good results on the example configurations. It treats subspaces as infinite, and its performance is not affected by disconnected segments of the same subspace or the point of intersection of several subspaces. Increasing the number of subspaces k does not degrade its performance either, although higher values of k may require using more facilities at initialization.

We illustrate a case where FLoSS may fail using a more challenging data set, shown in Fig. 4. The data set contains a plane and two co-planar lines, at two levels of noise: 1% and 5% of data variance. We use a fixed dimensionality of 2 for mpPCA, RANSAC and GPCA[1], and initialize FLoSS with both $1D$ and $2D$ subspaces.

On this data set, only mpPCA and GPCA correctly identify the subspaces, and only in the low-noise case. FLoSS, on the other hand, groups the two lines into one plane at both noise levels. In general, FLoSS prefers lower-dimensional subspaces through lower costs. However, having several densely sampled D-dimensional subspaces embedded in a $(D+1)$-dimensional subspace may offset the cost difference, causing FL to choose the $(D+1)$ dimensional subspace. As the structure of the subspaces is unknown in general, it is difficult to set facility costs so as to prevent this; a possible remedy could be the recursive approach of [2].

4.2. 3D motion segmentation

The 3D motion segmentation of points lying on rigidly moving objects can be shown to correspond to segmen-

[1] Although it is possible to specify different dimensionalities for GPCA, we found that fixed dimensionality gives better results using the code available at http://perception.csl.uiuc.edu/gpca/

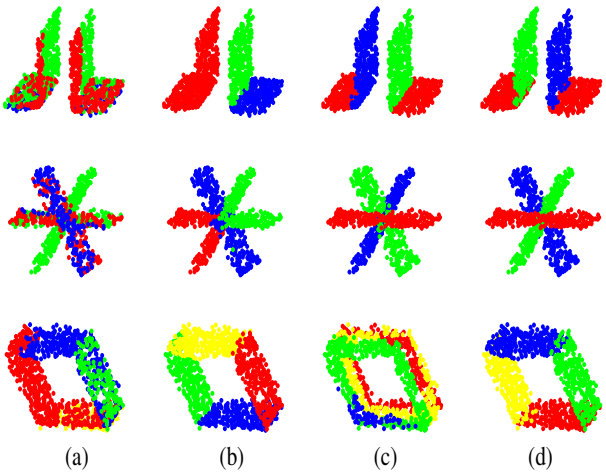

Figure 3. Comparison of different algorithms on data sets consisting of planes, (a) RANSAC, (b) mpPCA, (c) GPCA, and (d) FLoSS

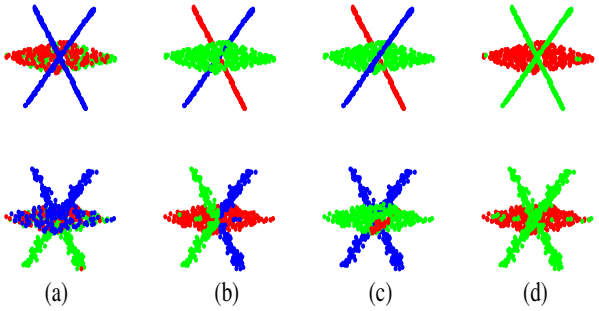

Figure 4. Mixed dimensionality subspaces, two noise levels: $\sigma^2 = 0.01$ (top row) and $\sigma^2 = 0.05$ (bottom row). (a) RANSAC, (b) mpPCA, (c) GPCA, and (d) FLoSS

tation of linear subspaces [24, 29]. Briefly, let $\{w_{fp} \in \Re^2\}_{p=1,\ldots,P}^{f=1,\ldots,F}$ be the image projections of P 3D points $\{X_p \in \mathcal{P}^3\}_{p=1,\ldots,P}$, lying on a rigidly moving object, over F frames of a rigidly moving camera. Under the affine projection model, $w_{fp} = A_f X_p$, where $A_f \in \Re^{2\times 4}$ is the affine camera matrix at frame f. Let $W \in \Re^{2F\times P}$ be a matrix whose columns are the 2D point trajectories. Then,

$$W_{2F\times P} = \begin{bmatrix} A_1 \\ \vdots \\ A_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} X_1 & \cdots & X_P \end{bmatrix}_{2F\times 4} \quad (23)$$

Therefore, the trajectories are embedded in a subspace of dimension $rank(W)$ that can be either 2, 3 or 4, depending on the type of motion. When the points lie on multiple moving objects, the trajectories lie on multiple linear subspaces of $\Re^{2F}$; this observation is the basis of most rigid body 3D motion segmentation algorithms.

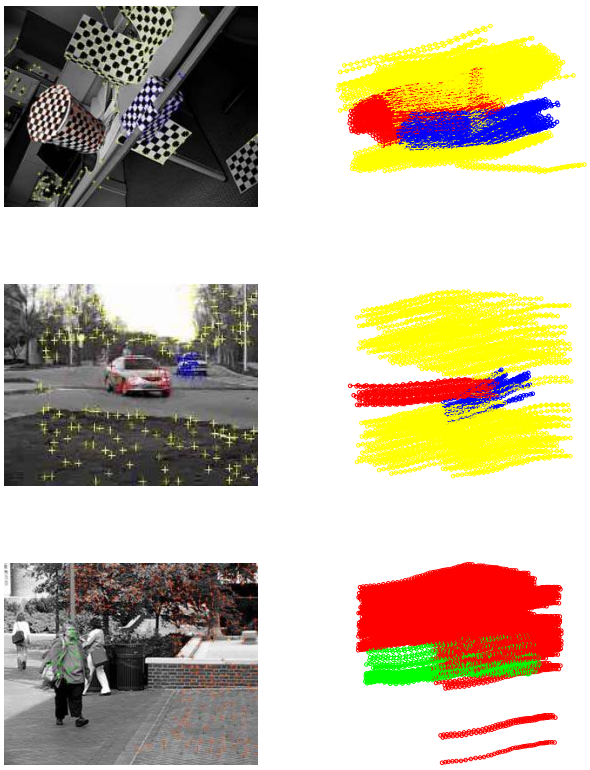

Figure 5. Example frames with keypoints (left) and trajectories (right) of checkerboard, traffic, and articulated motion sequences from the Hopkins155 database. The keypoints colors denote hand labeled objects.

A benchmark database for multi-body motion segmentation from point correspondences is the Hopkins155 database [24]. The database contains 50 video sequences of indoor and outdoor scenes, each containing two or three motions. Additionally, the 35 three-motion videos are split into $\binom{3}{2}$ groups containing only two out of three motions, resulting in a total of 155 sequences. The data contains subspaces of different dimensionalities. The three video types that make up the database are:

- Checkerboard: 104 video sequences with 2 checkerboard-pattern objects. The camera undergoes rotation, translation, or both.
- Traffic: 38 sequences of outdoor traffic scenes, taken by a moving hand-held camera.
- Articulated and non-rigid sequences: 13 video sequences of motions constrained by joints and non-rigid motions.

Example frames from the three types of video sequences are shown in Fig. 5.

We used the Hopkins155 database to evaluate the motion segmentation performance of the subspace segmentation models specified in Section 4.1, as well as that of two motion segmentation algorithms: LSA and MSL. The number of objects in each sequence was specified for all algorithms.

Except for FLoSS and mpPCA, the reported results were obtained from [24], where the following settings were used: GPCA was run on the first 5 principal components of the data matrix W, and LSA was run on the first k principal components, where k was the number of objects present. For RANSAC, the dimension of all subspaces was assumed to be 4; the algorithm was run 1000 times on each sequence, and the average results were recorded. mpPCA was run on the first 12 principal components of W, and the subspace dimensionality was set to 4. FLoSS was also run on the first 12 principal components of W, and initialized with random subsets of 3, 4 and 5 points (corresponding to subspaces of dimension 2, 3, and 4).

The segmentation errors, calculated as the percentage of misclassified points, are summarized in Tables 1 and 2. We note that no single method outperforms all others for all data sets. While GPCA achieves very good results for the 2 objects data, it performs poorly for the 3 objects data. As for the motion segmentation algorithms, LSA performs well, although inconsistently; while it is one of the best methods for the checkerboard sequences, it has the worst performance on traffic. MSL also performs well overall, notably better than mpPCA. Recall that MSL consists of three stages of mpPCA, initialized using the subspace separation algorithm, and adapted to different types of motion including degenerate. The large gap in the performance of the two methods is an indication of the sensitivity of mpPCA to initialization and variable subspace dimensionality.

FLoSS outperforms all other methods on the traffic sequences, and achieves comparable results on the checkerboard and articulated motion sequences. The FLoSS error median is typically low; however, some large errors do occur, most frequently as a consequence of choosing the wrong subspace dimensionality. This is illustrated in Fig. 6, which shows the first 3 principal components of data corresponding to the checkerboard sequence[2] shown in Fig. 5. Here, instead of a higher-dimensional subspace, FLoSS chooses two lower-dimensional subspaces embedded in it. GPCA and LSA correctly group the two embedded subspaces. On the other hand, FLoSS outperforms other methods on data that contains two disjoint parts of the same subspace, such as the data shown in Fig. 7, corresponding to the traffic sequence[3] shown in Fig. 5. In this case, LSA fails due to the non-local structure, and GPCA fails because very few points lie on two of the three groups. Such cases occur more frequently in traffic data when a large number of keypoints are detected on disjoint pieces of the background (due to, for example, trees and grass), in contrast to only a few keypoints per car. In comparison to the other non-motion segmentation specific methods (RANSAC, mpPCA, and GPCA) FLoSS is either better (the traffic and articulated motion data for 3 objects), or performs very closely to the best method (GPCA for 2 objects checkerboard and articulated motion, mpPCA for 3 objects checkerboard).

[2]The sequence $1RT2RTCRT - B$

[3]The sequence $cars2 - 07$

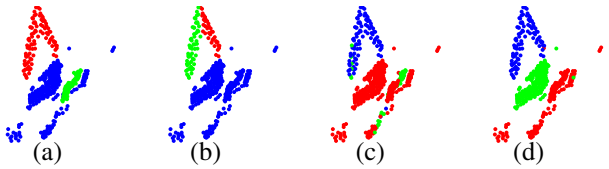

Figure 6. Checkerboard sequence, first 3 principal components. (a) Ground truth, (b) FLoSS, (c) GPCA, and (d) LSA

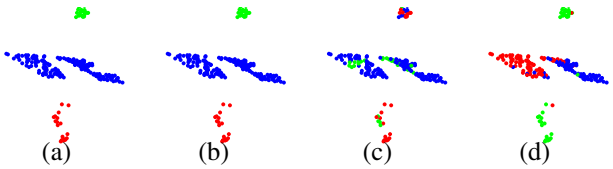

Figure 7. Traffic sequence, first 3 principal components. (a) Ground truth, (b) FLoSS, (c) GPCA, and (d) LSA

5. Conclusions and future work

We described a new subspace segmentation method that discovers linear subspaces in data using a message passing algorithm. We demonstrated its advantages over other methods on synthetic geometrical data, and evaluated its performance on multi-body motion segmentation from video.

The presented framework for subspace segmentation suggests numerous future work directions. We have described a way of finding an approximate facility location solution using the max-sum algorithm, and formulated subspace segmentation as an instance of facility location. The same approach could be applied to any other task that can be formulated as facility location, with different distances and costs. In addition, it is possible to adopt alternative approaches to the suggested method for discovering a candidate set of subspaces such as an iterative refinement procedure to re-sample candidate subspaces by using, for instance, PCA on points assigned to each subspaces.

References

[1] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 4

	Error	RANSAC	mpPCA	GPCA	FLoSS	LSA	MSL
Checkerboard	Average	6.52	9.89	6.09	7.70	**2.57**	4.46
	Median	1.75	2.49	1.03	1.23	0.27	**0.00**
Traffic	Average	2.55	21.41	1.41	**0.14**	5.43	2.23
	Median	0.21	17.61	**0.00**	**0.00**	1.48	**0.00**
Articulated	Average	7.25	25.13	**2.88**	4.69	4.10	7.23
	Median	2.64	19.44	**0.00**	1.30	1.22	**0.00**

Table 1. Motion segmentation percent error, 2 objects

	Error	RANSAC	mpPCA	GPCA	FLoSS	LSA	MSL
Checkerboard	Average	25.7	15.44	31.95	16.45	**5.80**	10.38
	Median	26.01	12.71	32.93	16.79	**1.77**	4.61
Traffic	Average	12.83	37.02	19.83	**0.29**	25.07	1.80
	Median	11.45	30.89	19.55	**0.00**	23.79	**0.00**
Articulated	Average	21.38	53.12	16.85	8.51	7.25	**2.71**
	Median	21.38	53.12	16.85	8.51	7.25	**2.71**

Table 2. Motion segmentation percent error, 3 objects

[2] O. Chum, T. Werner, and J. Matas. Two-view geometry estimation unaffected by a dominant plane. *CVPR*, 2005. 2, 5

[3] N. da Silva and J. Costeira. Subspace segmentation with outliers: A grassmannian approach to the maximum consensus subspace. *CVPR*, 1:1–6, 2008. 2

[4] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977. 2

[5] D. Dueck, B. Frey, N. Jojic, G. G. V. Jojic, A. Emili, G. Musso, and R. Hegele. Constructing treatment portfolios using affinity propagation. *Research in Computational Molecular Biology (RECOMB)*, 4955:360–371, 2008. 5

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[7] D. Forsyth, J. Haddon, and S. Ioffe. The joy of sampling. *IJCV*, 41:109–134, 2001. 2

[8] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007. 2, 5

[9] I. Givoni and B. Frey. A binary variable model for affinity propagation. *Neural Computation*, 21:1–12, 2009. 5

[10] R. Gorsuch. *Factor analysis*. Lawrence Erlbaum, Hillsdale NJ, 1983. 1

[11] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. *CVPR*, 1:11–18, 2003. 1, 2

[12] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. J. Wiley, New York, 2001. 1

[13] I. Jolliffe. *Principal component analysis*. Springer Series in Statistics, Berlin, 1986. 1

[14] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Proc. 8th ICCV*, pages 586–591, 2001. 2, 3

[15] F. Kschischang, B. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2):498 – 519, 2001. 3, 4

[16] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000. 1

[17] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *In Proc. of the 5th Int'l. Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002. 2

[18] A. Ng, Y. Weiss, and M. Jordan. On spectral clustering: analysis and an algorithm. In *NIPS 14*, pages 849–856. MIT Press, 2001. 3

[19] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 1

[20] K. Schindler and D. Suter. Two-view multibody structure-and-motion with outliers. *CVPR*, 2005. 2

[21] Y. Sugaya and K. Kanatani. Geometric structure of degeneracy for multi-body motion segmentation. *Workshop on Statistical Methods in Video Processing*, 2004. 3

[22] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999. 2

[23] P. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London*, 356:1321–1340, 1998. 2

[24] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. *CVPR*, 1:1–8, 2007. 1, 6, 7

[25] M. Turk and A. Pentland. Face recognition using eigenfaces. *CVPR*, 1:586–591, 1991. 1

[26] R. Urtasun, D. Fleet, and P. Fua. Temporal motion models for monocular and multiview 3d human body tracking. *Computer Vision and Image Understanding*, 104(2):157–177, 2006. 1

[27] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (gpca). *IEEE Trans. PAMI*, 27(12):1945–1959, 2005. 2, 5

[28] A. Yang, S. Rao, and Y. Ma. Robust statistical estimation and segmentation of multiple subspaces. In *CVPR workshop on 25 years of RANSAC*, 2006. 2

[29] J. Yanv and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. *ECCV*, 3954:94–106, 2006. 1, 3, 6

[30] J. Zhang, Y. Yan, and M. Lades. Face recognition : Eigenface, elastic matching, and neural nets : Automated biometrics. In *In Proceedings of the IEEE*, volume 85(9), pages 1423–1435, 1997. 1

[31] Z. Zhang. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338, 2004. 1