Beyond Pairwise Clustering

Sameer Agarwal[1], Jongwoo Lim[1], Lihi Zelnik-Manor[2], Pietro Perona[2], David Kriegman[1]
Serge Belongie[1]

[1]Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093, USA
http://vision.ucsd.edu

[2]Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125, USA
http://vision.caltech.edu

Abstract

We consider the problem of clustering in domains where the affinity relations are not dyadic (pairwise), but rather triadic, tetradic or higher. The problem is an instance of the hypergraph partitioning problem. We propose a two-step algorithm for solving this problem. In the first step we use a novel scheme to approximate the hypergraph using a weighted graph. In the second step a spectral partitioning algorithm is used to partition the vertices of this graph. The algorithm is capable of handling hyperedges of all orders including order two, thus incorporating information of all orders simultaneously. We present a theoretical analysis that relates our algorithm to an existing hypergraph partitioning algorithm and explain the reasons for its superior performance. We report the performance of our algorithm on a variety of computer vision problems and compare it to several existing hypergraph partitioning algorithms.

1. Introduction

Clustering or partitioning a dataset in a manner that elements of the same cluster are more similar to each other than elements in different clusters is a fundamental task in many fields of study including Computer Vision, Machine Learning, VLSI CAD and Statistics. With a few notable exceptions, formulations of the clustering problem and the proposed algorithms for solving them are based on the assumption that a pairwise (or dyadic) measure of distance between data points is available. A common measure for data points lying in a vector space is the Euclidean distance. The use of a pairwise measure is characteristic of central clustering methods like k-means and k-medoids, as well as pairwise clustering methods [11, 18, 24, 27, 30].

It is not always the case, however, that there exists a similarity measure between pairs of data points. For some clustering problems, one may need to consider three or more data points together to determine if they belong to the same cluster. Consider a k-lines algorithm which clusters data points in a d-dimensional vector space into k clusters where elements in each cluster are well approximated by a line. As every pair of data-points trivially defines a line, there does not exist a useful similarity measure between pairs of points for this problem. Yet there do exist useful measures for triplets of points which indicate how close the three points are to being collinear.

Weighted undirected graphs serve as a combinatorial representation for datasets containing pairwise relationships. For this reason, clustering algorithms are also frequently referred to as graph partitioning algorithms. The corresponding representation for datasets with higher order relationships is a hypergraph. Like a weighted graph, a weighted hypergraph is defined as a set of vertices and a set of weighted hyperedges. Each weighted hyperedge can now be an arbitrary subset of the vertices and has a scalar weight associated with it.

The focus of this work is the largely neglected but fundamental problem of clustering data on the basis of triadic and higher-order affinities. We introduce a general purpose hypergraph partitioning algorithm, based on a novel graph approximation scheme we call *Clique Averaging*, and show that with an appropriate similarity measure, this generic clustering algorithm can be applied to a number of clustering problems that arise in computer vision[1].

As an example of the kind of problems we are interested in, consider the problem of clustering a collection of images of different objects, each of which is imaged in the same pose, but under a collection of different lighting conditions. Belhumeur and Jacobs have shown that for any two images, there exists a Lambertian surface with spatially vary-

[1]For a tensor theoretic approach to the same problem see [14] in these proceedings

ing albedo and a pair of light source directions that could produce the two images [21]. Hence, there is no function of a pair of images that returns zero when the images depict the same object (but under differing lighting) yet returns a non-zero value when the images are depicting different objects. Furthermore, it is well known that the set of images of a Lambertian surface under arbitrary lighting (without shadowing) lies on a 3-D linear subspace in the image space [3]. As any three images span a three dimensional subspace, one needs to consider atleast four images at a time to define a measure of affinity.

As another example, consider the problem of partitioning a set of correspondences into clusters that are related by the same motion model. The usual approaches are based on (i) doing a greedy set covering using RANSAC [31], (ii) performing a Hough Transform [2], or (iii) performing pose clustering in the space of the model parameters [29]. There are fundamental problems with each of these approaches. RANSAC was designed for detecting a single model in the presence of a noise and as we will show it does not scale well to the case of multiple overlapping models. Approaches based on a generalized Hough Transform require a bounded finite parameterization of the model. Finding such a parametrization is not a trivial problem, and even if one is available, the Hough transform for anything but the simplest problems requires huge amounts of memory. Clustering in the space of model parameters, while conceptually attractive, may not be tractable. The problem is, to perform this clustering one needs to be able to define a measure of similaritybetween arbitrary pairs of models. Given that most parameter spaces are non-linear manifolds without a global metric there may not be any easy way of doing so. In contrast, the fitting error of a set of points to a model is a natural and easily available measure of disassociation, without any limitations on the geometric structure of the parameter space of the model[2].

The rest of the paper is organized as follows. In Section 2 we survey related work on hypergraph partitioning. Section 3 presents the theory behind the proposed algorithm. In Section 4 we report the performance of our algorithm on a variety of computer vision problems and compare it to several existing hypergraph partitioning algorithms. Section 5 concludes with a discussion of open problems and avenues for future work.

2. Related Work

The study of distances defined over sets of size greater than two is not new The literature on n-metrics is devoted to constructing and analyzing distance measures defined over $n+1$-tuples. The primary focus of this literature is the study of topological and geometrical properties of these generalized measures [8].

While the work on n-metrics is theoretical, a more practical line of work has emerged in the psychometrics community. Starting with the work of Hayashi, who proposed the area of a triangle as the triadic distance between its vertices [16], a number of researchers have developed generalizations of Multidimensional Scaling (MDS) to the case of triadic data. MDS is a technique for embedding pairwise similarity or dissimilarity data in a low dimensional Euclidean space [5]. This embedding is primarily used for the purposes of visualization, but can also be used as a preprocessing step for data analysis methods that require a coordinate representation of their input. In one of the earliest such works, Carroll and Chang developed an algorithm for n-adic MDS using a generalization of the SVD to the case of n-dimensional matrices [6]. Subsequently, Cox et al. have proposed an MDS algorithm based on a combination of a Gradient Descent and Isotonic regression [7]. Axiomatic theories of triadic distances have been developed by Joly & LeCalvé and Heiser & Bennani [17, 22].

The most extensive and large scale use of hypergraph partitioning algorithms, however, occurs in the field of VLSI design and synthesis. A typical application involves the partitioning of large circuits into k equally sized parts in a manner that minimizes the connectivity between the parts. The circuit elements are the vertices of the hypergraph and the *nets* that connect these circuit elements are the hyperedges [1]. The leading tools for partitioning these hypergraphs are based on two phase multi-level approaches [23]. In the first phase, they construct a hierarchy of hypergraphs by incrementally collapsing the hyperedges of the original hypergraph according to some measure of homogeneity. In the second phase, starting from a partitioning of the hypergraph at the coarsest level the algorithm works its way down the hierarchy and at each stage the partitioning at the level above serves as an initialization for a vertex swap based heuristic that refines the partitioning greedily [9, 25]. The development of these tools is almost entirely heuristic and very little theoretical development exists that analyzes their performance beyond empirical benchmarks.

The set of tools available for partitioning graphs are much better developed than those for hypergraphs. A case in point is the development of algorithms for solving the max-flow min-cut problem on hypergraphs. While extremely efficient algorithms for the case of graphs have been available for sometime now [13], it is only recently that efficient algorithms that operate directly on hypergraphs have become available [28]. Thus it makes sense to consider methods that construct a graph that approximates the hypergraph and partition it; this partition in turn induces a vertex partitioning on the original hypergraph. In fact, it is possible to construct methods that operate directly on the

[2]For the particular case of parameter spaces that are linear subspaces see GPCA [32]

hypergraph while implicitly working on its graph approximation [12]. The two most commonly used graph approximations are *Clique Expansion* and *Star Expansion*. Clique Expansion, as the name implies, expands each hyperedge into a clique. Star expansion introduces a dummy vertex for each hyperedge and connects each vertex in the hyperedge to it [19]. As can be expected, the weights on the edges of the clique and the star determine the cut properties of the approximating graph. We refer the reader to [15, 20] for further discussions that address this problem.

3. Theory

In this section, we describe our proposed hypergraph partitioning algorithm. It is a two-step procedure. In the first step we construct a weighted graph that approximates the hypergraph. This approximation is based on a novel algorithm that we call *Clique Averaging*. In the second step we use a spectral clustering algorithm based on the normalized Laplacian of the graph to partitioning its vertex set. As the second step of this algorithm is well known, we will mainly focus on the development and properties of Clique Averaging. As part of our analysis we will also show the relation of Clique Averaging to Clique Expansion, which is a commonly used graph approximation in the VLSI CAD literature. Finally as a consequence of this relationship we are able to show the relationship between the algorithm proposed by Gibson et al., and our algorithm. We begin with some notation.

A weighted undirected hypergraph H is a pair (V, h). Here V is the set of vertices of H, and subsets of V are known as hyperedges. The function h associates non-negative weights with each hyperedge. In the special case when the cardinality of the hyperedges is 2, H is a weighted undirected graph and the hyperedges are the same as ordinary graph edges. We use $G = (V, g)$ to denote a weighted undirected graph defined over the same set of vertices V with the weighting function denoted by g. We will assume that the number of vertices in the hypergraph is n, i.e., $|V| = n$. While general hypergraphs can have hyperedges of varying cardinality, and the algorithms we present will work on hypergraphs with arbitrarily sized hyperedges, we will for reasons of notational simplicity assume that all hyperedges are of a fixed known size $k \geq 2$. The two weighting functions h and g can then formally be described as

$$h : V^k \rightarrow R^+ \quad \text{and} \quad g : V^2 \rightarrow R^+. \tag{1}$$

As the hypergraphs we are dealing with are undirected, the functions h and g are symmetric in their arguments, i.e., their value remains the same if the order of the arguments is arbitrarily permuted. Finally we will use the symbols d_k to denote the vector of hyperedge weights obtained from H by ordering the hyperedges in lexicographic order based on their vertex sets. The vector d_2 denotes the corresponding lexicographically ordered weight vector for the graph G.

In the above notation, the problem of approximating the hypergraph with a graph can now be restated as the problem of approximating the weighting function h with the weighting function g. But before we introduce our approximation scheme, it is instructive to consider the feasibility of such an approximation in a purely combinatorial sense.

For a complete weighted hypergraph of order k defined on $|V| = n$ vertices, the weighting function h can take on $\binom{n}{k}$ different values. For a complete graph on the same number of vertices, the number of degrees of freedom is only $\binom{n}{2}$. Even for moderately sized n and k, the number of degrees of freedom for a graph is a tiny fraction of that of a hypergraph. Thus it is not reasonable to expect any graph approximation that preserves the number of vertices to do a good job of approximating every possible hypergraph. However we are not interested in approximating all possible hypergraphs. For a dataset to be *clusterable* the weighting function should be indicative of the cluster structure in the data. For example in the case of bipartitioning a hypergraph, the ideal hypergraph would consist of two completely connected components. The set of hypergraphs of this type is much smaller than $\binom{n}{k}$; in fact it is only $O(n^2)$ and are easily represented by graphs containing two completely connected components on the corresponding vertices. But real data is noisy and the corresponding hyperedge weights reflect that; however, for data that can be divided into well separated partitions, one would expect that the corresponding hypergraph is close to a hypergraph containing two densely connected components, and thus amenable to approximation with a weighted graph.

3.1. Clique Averaging

We are now ready to introduce our hypergraph approximation scheme. Our construction and analysis of the approximation will be based on considering graph approximations of a single hyperedge z. The extension to the whole hypergraph is then a matter of linear superposition. We begin by revisiting the observation that the value of the weighting function $h(z)$ is independent of the order in which we consider the vertices in z. In light of this, when considering the various kinds of graphs that can be associated with the hyperedge z, the only graph structure that satisfies the requirements of symmetry is the k-clique on z. A k-clique is a completely connected graph on k vertices. Thus the task of approximating $h(z)$ boils down to assigning weights to the edges in k-clique associated with z.

As we mentioned earlier, the most widely used such approximation scheme is Clique Expansion, and is based on the assumption that every edge in the clique associated with

z has edge weight equal to $h(z)$. Formally

$$h(z) = g(v_i, v_j), \qquad \forall \quad v_i, v_j \in z \tag{2}$$

Collecting the above set of equations over all hyperedges results in an over-determined linear system consisting of $\binom{k}{2}\binom{n}{k}$ equations. This system has a simple least squares solution given by

$$g(v_i, v_j) = \frac{1}{\mu(n,k)} \left(\sum_{v_i, v_j \in z} h(z) \right). \tag{3}$$

Here $\mu(n,k) = \binom{n-2}{k-2}$ is the number of hyperedges that contain a particular pair of vertices. Thus the weight on an edge is the arithmetic mean of the weights of all the hyperedges that contain both of its vertices. Other choices for $\mu(n,k)$ are also possible and will amount to different weighting schemes when working with hyperedges of varying sizes. The optimal choice of weighting in Clique Expansion when combining information across hyperedges is an area of research in itself [15, 20].

The relationship between a hyperedge and the edge weights in its clique in the above approach was the simplest possible, where we assumed that the hyperedge weight and the edge weights are equal to each other. In an attempt to make this relationship richer, we take a generative view of the problem. Let us assume that there exists a $\binom{k}{2}$-ary function F such that, given the edge weights on a k-clique, it returns the corresponding hyperedge weight. Formally

$$h(z) = F\left(g(v_1, v_2), \ldots, g(v_i, v_j), \ldots, g(v_{k-1}, v_k)\right). \tag{4}$$

Now given a particular generative model F and a hypergraph H, the hypergraph approximation problem can then be stated as the problem of solving for those values of the graph edge weights $g(v_i, v_j)$ that satisfy the above equation over all hyperedges simultaneously. Of course how well the graph G captures the structure of hypergraph H is now a function of F. So what is a good choice of F? We begin our search by demanding some simple properties of F: (i) **Positivity** F should be positive for positive valued arguments, (ii) **Symmetry** F should be symmetric in its arguments, (iii) **Monotonicity** F should be monotonic in each of its arguments. Positivity and symmetry are simple consequences of the definition of h. Monotonicity is a reasonable demand to make of F as one would expect that as the interaction between two vertices increases or decreases the strength of the hyperedge would be indicative of that change. Within these constraints there are still very many choices for F. In this paper we consider the family of functions F_p parameterized by the positive scalar $p > 0$,

$$F_p(x_1, x_2, \ldots, x_u) = \left(\lambda(u) \sum_{i=1}^{u} x_i^p \right)^{1/p}, \quad u = \binom{k}{2} \tag{5}$$

where λ is a scalar function of the arity of F_p. We can now write Equation (4) as

$$h(z) = \left(\lambda\left(\binom{k}{2}\right) \sum_{\substack{v_i, v_j \in z \\ i<j}} g^p(v_i, v_j) \right)^{1/p} \tag{6}$$

For brevity we will write $\lambda(k) = \lambda(\binom{k}{2})$. Using this and taking the p^{th} power on both sides gives us

$$h^p(z) = \lambda(k) \sum_{\substack{v_i, v_j \in z \\ i<j}} g^p(v_i, v_j) \tag{7}$$

We note that the above equation states that the L_p norm of the clique weights is proportional to the hyperedge weight. It is also worth noting that as the value of p increases the L_p norm is biased towards the largest clique weight. For a given h and a fixed p this is a linear system in $g(v_i, v_j)^p$. Thus without any loss of generality we can restrict our analysis to the case $p = 1$. With this in mind let us interpret the above equation. Modulo a constant the above equation states that the weight of a hyperedge is the arithmetic mean of the edge weights in the clique it induces. Thus a natural choice for $\lambda(\binom{k}{2})$ is $\binom{k}{2}^{-1}$. Other choices for $\lambda(k)$ are possible and will amount to different weighting schemes when working with hyperedges of varying sizes. When working with hyperedges of the same size, which is the case in the current study, the choice of $\lambda(k)$ is immaterial as it amounts to a uniform scaling of the graph weights. As spectral clustering algorithm are insensitive to scaling of the edge affinities this is not a problem. For the sake of concreteness we will use the arithmetic mean interpretation of the above equation and the resulting choice of $\lambda(k)$.

Also without loss of generality we will assume that the set of hyperedges has been ordered in a lexicographic order based on the vertices incident on each hyperedge. A similar ordering is done on the set of graph edges too. We can now define the incidence matrix $\mathbf{\Delta}$. $\mathbf{\Delta}$ is a zero-one matrix, that represents the incidence relationship between a hyperedge in H and an edge in G. We say an edge is incident on a hyperedge if the hyperedge contains both of its vertices.

$$\mathbf{\Delta}_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is incident on hyperedge } i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The rectangular matrix $\mathbf{\Delta}$ has $\binom{n}{k}$ rows and $\binom{n}{2}$ columns. Note that $\mathbf{\Delta}$ is an extremely sparse matrix with only $\binom{k}{2}$ non-zero entries. Now recall that d_2 denotes the vector of graph edge weights of size $\binom{n}{2}$ and, d_k denote the vector of hyperedge weights. Then Equation (7) for the case of $p = 1$ can be written in matrix form as

$$\mathbf{\Delta} d_2 = \lambda(k) d_k \tag{9}$$

This equation assumes that $d_2 \geq 0$, i.e., each element of the vector d_2 is non-negative. Hence when solving for d_2 given d_k, we will explicitly enforce this constraint. When working with hypergraphs with edge weights that are bounded above as in the case of affinities; we will enforce an upper bound $d_2 \leq 1$ also. Since the linear system is overdetermined, the solution to Equation (9) has to be determined by minimizing the least squares error. Thus for the case of a hypergraph with hyperedge weights bounded in the interval $[0, 1]$, its graph approximation is given by the edge weight vector d_2 that minimizes the following constrained minimization problem

$$\min_{d_2} \|\lambda(k)\mathbf{\Delta}d_2 - d_k\|_F^2 \qquad 0 \leq d_2 \leq 1 \tag{10}$$

The above optimization problem is an instance of the Bounds Constrained Least Squares problem. However as we noted earlier $\mathbf{\Delta}$ is a sparse matrix and thus we can exploit efficient iterative methods for solving it [4]. We use `lsqlin` in MATLAB's Optimization Toolbox.

3.2. Duality

In this section we analyze the link between Clique Averaging and Clique Expansion. In Section 3.1 we saw that the graph edge weights as a result of Clique Expansion are given by

$$g(v_i, v_j) = \frac{1}{\mu(n,k)} \left(\sum_{v_i, v_j \in z} h(z) \right). \tag{11}$$

using the notation of the previous section we can re-write this as

$$d_2^e = \frac{1}{\mu(n,k)} \mathbf{\Delta}^\top d_k. \tag{12}$$

We use the superscript e to indicate Clique Expansion. From Equation (9), the linear system for Clique Averaging is

$$\lambda(k)\mathbf{\Delta}d_2 = d_k \tag{13}$$

It is readily shown that the above two equations are duals of each other.

Let us now multiply both sides of Equation (12) by $\mathbf{\Delta}$ to get

$$\mathbf{\Delta}d_2^e = \frac{1}{\mu(n,k)} \mathbf{\Delta}\mathbf{\Delta}^\top d_k. \tag{14}$$

Note that modulo a constant, Wquations (13) and (14) differ only in the right hand side by a pre-multiplication by the matrix $\mathbf{S} = \mathbf{\Delta}\mathbf{\Delta}^\top$. To understand the action of this pre-multiplication let us consider the structure of the matrix $\mathbf{S}$.

$\mathbf{S}$ is a symmetric matrix, with rows and columns corresponding to the hyperedges H. The entry in the z_i row and z_j column corresponds to the inner product of the z_i^{th} and the z_j^{th} rows of $\mathbf{\Delta}$. $\mathbf{\Delta}$ as we noted earlier is a zero-one matrix, hence the dot product counts the number of edges in the graph G that the two hyperedges share. These entries are easily calculated, for if $l = |z_i \cap z_j|$ denotes the number of vertices the two hyperedges have in common then $[\mathbf{S}]_{z_i z_j} = \binom{|z_i \cap z_j|}{2} = \binom{l}{2}$ Let the distance between two hyperedges of size k be $k - l$, then multiplication with the z_i^{th} row of $\mathbf{S}$ is equivalent to multiplying each element of d_k by a decreasing function of the distance from the hyperedge z_i and summing over them. This is in fact a convolution of the hyperedge weights by a quadratically decreasing kernel. Thus $\mathbf{S}d_k$ is a low passed version of d_k. This implies that Clique Expansion solves the same approximation problem as Clique Averaging, but instead of operating on the original hypergraph it operates on a low passed version of it. We know from basic signal processing theory that low pass filtering is an operation that loses information and in the limit transforms the weight vector d_k into a constant vector. Hence the approximation produced by Clique Averaging is of a higher quality and better preserves the cluster structure present in the hypergraph H.

3.3. Partitioning the Hypergraph

We now describe our proposed hypergraph partitioning algorithm. Given a dataset, the first step is the construction of the affinity hypergraph H by calculating the affinity for every distinct k-tuple in the dataset. However, calculating $\binom{n}{k}$ hyperedge weights can be computationally prohibitive. In many cases the user has a choice of the size of hyperedge when constructing the hypergraph. Using a simple counting argument one can show that since the number of within cluster hyperedges to the number of between cluster hyperedges goes down geometrically with increasing hyperedge size, the smallest possible value of k should be chosen. We get around this problem by sub-sampling the hypergraph and instead considering a hypergraph H' obtained by sparsely sampling hyperedges. Since the column rank of $\mathbf{\Delta}$ is $\binom{n}{2}$, we need at least that many rows, which in turn puts a lower bound on the number of hyperedges in H'. In our experiments we fix $n_{samples} = 5pn^2$ where p is the number of partitions that the data is to be divided into. We then use Clique Averaging to construct a graph G. To partition the graph into p parts, we use a spectral clustering algorithm that uses the first p eigenvectors of the normalized Laplacian of the graph and performs k-means clustering on the resulting k-dimensional embedding [26, 30].

4. Experiments

In this section we study the performance of six different algorithms out of which five are hypergraph partitioning al-

gorithms. The sixth algorithm is a multi-round variant of RANSAC. We report the performance of the algorithms on two datasets. The algorithm are

1. Clique Averaging+Ncut (CAVERAGE) The hypergraph is approximated using Clique Averaging and the resulting graph is partitioned using the Normalized Cuts algorithm.
2. Clique Expansion+Ncut (CEXPAND) The hypergraph is approximated using Clique Expansion and the resulting graph is partitioned using the Normalized Cuts algorithm.
3. Gibson's Algorithm-Sum Model (GIBSONS) Gibson et al.'s algorithm operating under the sum model [12].
4. Gibson's Algorithm-Product Model (GIBSONP) Gibson et al.'s algorithm operating under the product model [12].
5. kHMeTiS (KHMETIS) The leading tool for hypergraph partitioning in the VLSI community based on multi-level iterative refinement.
6. Cascading RANSAC (CRANSAC) A simple multi-round extension to the RANSAC algorithm. In the i^{th} round a number of trials are performed to identify that k-tuple that has the highest number of inliers. This k-tuple and its associated inliers are identified as the i^{th} group in the dataset and removed from it.

Reporting unbiased performance comparison of clustering algorithms is a hard problem, since each algorithm that one compares against has one or more free parameters one must set according to the particular problem at hand. Thus while comparing performance across problems, an approach giving each algorithm the best shot would need to perform a sweep over all possible parameter values. While this might report the best behavior of the algorithm it is clearly not informative about the robustness of the algorithm to parameter choice, a property that is of vital importance to a user who is using the algorithm on a novel dataset. Thus it is important to use an experimental protocol that is as close as possible to real world usage.

One of the ways in which algorithms are tuned is by running them on a small pilot dataset similar to the real problem. This is the basis of our experimental protocol. When running an algorithm over a suite of experiments, we choose a problem that lies at the center of the set of experiments in terms of complexity and choose the best performing parameters using a parameter sweep. This parameter setting is used for all the experiments in the test suite. To be fair to CRANSAC in terms of computation resources, we set the total number of trials it could perform to be equal to the number of hyperedges. GIBSONS and CAVERAGE were run with $p = 1$. The only free parameter across all the hypergraph partitioning algorithms was the parameter σ that was used to convert a dissimilarity d into the affinity $e = e^{-d/\sigma}$. In case of CRANSAC the error threshold for inlier detection was the free parameter.

4.1. k-lines Clustering

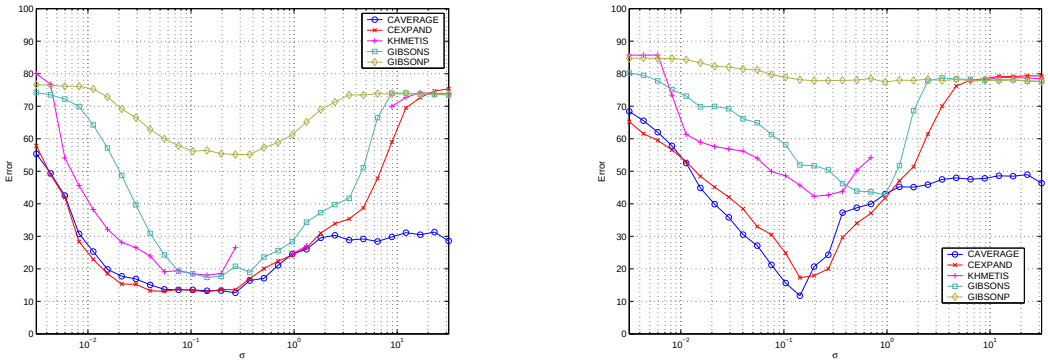
In the first experiment we consider the k-lines problem in spaces of dimension greater than two, i.e., given a set of points in $\mathbb{R}^d$, the task then is to partition them into a number of d-dimensional lines. In the case of lines in two dimensions the Hough transform solves this problem quite effectively, but with three or more dimensions there is no convenient parameterization that can be used. Pairwise measures of similarity are not applicable here since any two points are co-linear, thus it takes at least three points to determine a measure of co-linearity. This is an example of a triadic relationship. The dissimilarity measure on triples of points is their distance to the best fitting line. Our dataset consists of points sampled from gently curving lines with noise added to them. All the lines pass through the origin. Thus all clusters overlap with each other to some degree. The lines are generated as arcs of circles with a controllable radius of curvature. We consider the performance of the six hypergraph partitioning algorithms. The results are reported over a dataset containing 5 lines, in the cube $[-1, 1]^5$. We sample 70 points from each line for a total of 350 points. The hypergraph was generated by sampling $k^2\binom{n}{2} = 549675$ 3-tuples. For this dataset we considered the performance the five hypergraph partitioning algorithms over varying values of σ. Results are reported over 30 trials.

CAVERAGE	12.6	CEXPAND	12.9
GIBSONS	17.3	GIBSONP	55.1
KHMETIS	18.0	CRANSAC	23.4

But a more elaborate picture emerges when one looks at the performance of the algorithms over a range of values of σ. Figure 1(a) plots this behavior. The graph has a number of notable features. We begin by noting that Clique Expansion and Clique Averaging are the two best performing algorithms and for small and moderate values of sigma there is virtually no difference between their performance. It is however interesting to note that as sigma increases further the performance of Clique Expansion sharply degrades and reaches 80% error which is the same as chance. Clique Averaging on the other hand continues to perform well at about 30% error while the other four algorithms are operating at $70\% - 80\%$ error. The error curve for HMETIS is disjointed because for those values of σ the program crashed.

4.2. Illumination Invariant Clustering

It has been shown that all the images of a Lambertian object illuminated by a point light source lie in a three dimensional subspace [3]. This leads to a natural measure of dissimilarity over four(tetradic) or more images and allows us to perform clustering using it. Indeed this is a generalization of the k-lines problem to the k-subspaces problem. If we assume that the four images under consideration form the columns of a matrix, then $d = \frac{s_4^2}{s_1^2+\cdots+s_4^2}$, serves as a



(a) (b)

Figure 1. (a) and (b) show the performance of the five hypergraph partitioning algorithms on the k-lines and Yale face datasets respectively as the scale parameter σ is varied. Note that in both cases despite similar best case performance, CAVERAGE **is much more robust to scale changes than** CEXPAND**.**

measure of dissimilarity. Where s_i is the i^{th} singular of this matrix.

The Yale database contains 45 images each of 10 individuals. The aim of the clustering procedure is to partition the images into groups by identity. Figure 1(b) shows the result of performing a parameter sweep over the parameter σ for the case of 7 identities. The gross behavior of the algorithms in Figure 1(a) and Figure 1(b) is very similar. Again CAVERAGE and CEXPAND are consistently the best performing algorithms and CAVERAGE is much more robust to changes in the value of the scaling parameter σ. This experiment was used as the basis for tuning the parameters for individual algorithms for the following experiment.

The following table presents the results of running the six algorithms on four subsets of the Yale face dataset with increasing number of points and clusters. Each algorithm was run 30 times with parameters picked by running a parameter sweep over σ case of 7 identities. The results are in the form of mean error/standard deviation.

	4	6	8	10
AVERAGE	4.2 / 6.3	12.7 / 8.4	17.4 / 4.0	16.0 / 3.0
CEXPAND	11.8 / 3.4	17.6 / 5.4	21.8 / 5.4	24.9 / 4.3
GIBSONS	25.9 / 7.3	42.2 / 3.8	47.7 / 3.0	51.5 / 2.1
GIBSONP	67.4 / 2.3	75.2 / 1.2	79.7 / 0.8	82.8 / 0.7
KHMETIS	21.5 / 4.3	41.9 / 6.8	38.4 / 4.7	58.3 / 3.3
CRANSAC	16.2 / 9.5	23.6 / 9.2	35.1 / 7.9	37.1 / 6.6

As can be seen in the above table, CVERAGE beats all other algorithms across the board.

While two problem sets do not make for conclusive evidence, but they are indicative of a few general trends. CAVERAGE is much less sensitive to changes in the dynamic range of hyperedge weights, providing empirical verification of the relationship established between CEXPAND and CAVERAGE in Section 3.2. It is also consistently the best performing algorithm amongst the six we have tested. It can be shown that the only difference between GIBSONS and CEXPAND is that the former uses the unnormalized Laplacian, while the latter uses the normalized Laplacian. This set of experiments is further evidence that it is preferable to use the normalized Laplacian over its unnormalized variant.

5. Discussion

In this work we have introduced hypergraph partitioning as the natural formulation for a number of computer vision tasks. Leveraging a simple additive generative model, we have introduced a new class of hypergraph approximation algorithms which have provably better behavior than existing approximations, for which we have presented empirical proof. We also compared the performance of our proposed algorithm to four existing hypergraph partitioning algorithms and a multi-round variant of RANSAC. In all our experiments, our Clique Averaging approach outperformed its competitors both in terms of clustering error as well as insensitivity to parameter changes in the data. There do however remain a number of open questions and directions for future work. The most important question is that of computational complexity. Since we solve for the all the graph edge weights, the sampling complexity for the algorithm is lower bounded by $O(n^2)$. However there is evidence that for data that is clusterable into a small number of clusters, spectral clustering can be performed using far fewer than $O(n^2)$ graph edges [10], thus it seems a signif-

icant reduction in the sampling complexity of Clique Averaging is possible. We have developed a sparse implementation of the current Clique Averaging algorithm that works with an order of magnitude fewer samples; lack of space precludes us from including a discussion of it. There is also the possibility of developing better generative models relating graphs and hypergraphs.

While the methods we discussed in this paper are focused on converting a hypergraph to a graph and then operating upon it, the development and performance of methods that operate directly on the hypergraph without any intermediate or implicit reduction to a graph remains an open question.

6 Acknowledgements

It is a pleasure to acknowledge our various discussions with Andrew Kahng, Fan Chung Graham, Josh Wills, Kristin Branson, Sanjoy Dasgupta and Satya Prakash Mallick.

Sameer Agarwal and Serge Belongie are supported by NSF-CAREER #0448615, DOE/LLNL contract no. W-7405-ENG-48 (subcontracts B542001 and B547328), and the Alfred P. Sloan Fellowship. Jongwoo Lim and David Kriegman are supported by NSF CCR 00-86094. Lihi Zelnik-Manor and Pietro Perona are supported by MURI award number SA3318 and by the Center of Neuromorphic Systems Engineering award number EEC-9402726.

References

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19(1–2):1–81, 1995.

[2] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[3] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions. *IJCV*, 28(3):245–260, 1998.

[4] A. Björck. *Numerical methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[5] I. Borf and P. Groenen. *Modern multidimensional scaling: Theory and Applicatsions*. Springer Series in Statistics. Springer Verlag, 1997.

[6] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckert-Young decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[7] T. Cox, M. Cox, and J. Branco. Multidimensional scaling for n-tuples. *British Journal of Mathematical and Statistical Psychology*, 44:195–206, 1991.

[8] M.-M. Deza and I. . Rosenberg. n-Semimetrics. *European Journal of Combinatorics*, 21:797–806, 2000.

[9] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th conference on Design automation*, pages 175–181. IEEE Press, 1982.

[10] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *PAMI*, 26(2):214–225, 2004.

[11] Y. Gdalyahu, D. Weinshall, and M. Werman. Stochastic image segmentation by typical cuts. In *CVPR*, 1999.

[12] D. Gibson, J. M. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *VLDB*, pages 311–322. Morgan Kaufmann Publishers Inc., 1998.

[13] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. In *FOCS*, pages 2–11, 1997.

[14] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *CVPR*, 2005.

[15] S. W. Hadley. Approximation techniques for hypergraph partitioning problems. *Discrete Appl. Math.*, 59(2):115–127, 1995.

[16] C. Hayashi. Two dimensional quantification based on the measure of dissimilarity among three elements. *Annals of the Institute of Statistical Mathematics*, 24:251–257, 1972.

[17] W. J. Heiser and M. Bennani. Triadic distance models: Axiomatization and least squares representation. *Journal of Mathematical Psychology*, 41:189–206, 1997.

[18] T. Hofmann and J. M. Buhmann. Pairwise data clustering by deterministic annealing. *PAMI*, 19(1):1–14, 1997.

[19] T. Hu and K. Moerder. Multiterimnal flows in hypergraphs. In T. Hu and E. S. Kuh, editors, *VLSI Circuit Layout: Theory and Design*, pages 87–93. IEEE Press, 1985.

[20] E. Ihler, D. Wagner, and F. Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Information Processing Letters*, 45:171–175, 1993.

[21] D. Jacobs, P. Belhumeur, and R. Basri. Comparing images under variable illumination. In *CVPR*, pages 610–677. IEEE, IEEE Press, 1998.

[22] S. Joly and G. L. Calvé. Three-way distances. *Journal of Classification*, 12:191–205, 1995.

[23] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 343–348. ACM Press, 1999.

[24] L. Kaufman and P. J. Rousseeuw. *Finding groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[25] B. Kernighan and L. S. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, 49:291–307, February 1970.

[26] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

[27] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *CVPR*, 2003.

[28] J. Pistorius and M. Minoux. An improved direct labeling method for the max-flow min-cut computation in large hypergraphs and applicatsions. *International Transactions in Operational Research*, 10(1):1–11, 2003.

[29] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, August 2000.

[31] P. H. S. Torr. Geometric motion segmentation and model selection. In *Philosophical Transactions of the Royal Society A*, pages 1321–1340. Roy Soc, 1998.

[32] R. Vidal, Y. Ma, and J. Piazzi. A new gpca algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. In *CVPR*, 2004.