Ensemble Segmentation Using Efficient Integer Linear Programming

Amir Alush and Jacob Goldberger

***Abstract*—We present a method for combining several segmentations of an image into a single one, that in some sense is the average segmentation, in order to achieve a more reliable and accurate segmentation result. The goal is to find a point in the "space of segmentations" which is close to all the individual segmentations. We present an algorithm for segmentation averaging. The image is first over-segmented into superpixels. Next, each segmentation is projected onto the superpixel map. An instance of the EM algorithm combined with integer linear programming is applied on the set of binary merging decisions of neighboring superpixels to obtain the average segmentation. Apart from segmentation averaging, the algorithm also reports the reliability of each segmentation. The performance of the proposed algorithm is demonstrated on manually annotated images from the Berkeley segmentation dataset and on the results of automatic segmentation algorithms.**

***Index Terms*—image segmentation, ensemble segmentation, integer linear programming, correlation clustering, EM algorithm.**

I. Introduction

Image segmentation is a fundamental process in many image, video, and computer vision applications. It is essentially the partitioning of an image into several constituent components. The basic task of image segmentation is thus to assign each pixel in the image to one of the image components. Segmentation is often not a well-defined task, since the level to which an image is subdivided is determined by the application at hand. Many segmentation algorithms have been proposed and studied in recent decades and new algorithms are continuously emerging. These segmentation algorithms are usually based on various combinations of local low-level features and global optimization methods.

Instead of considering ways to design better segmentation methods, we explore another strategy which involves enhancing performance by combining segmentations of the same image obtained by different algorithms. Similar to the concept of the 'wisdom of the crowd' in social choice theory, it is reasonable to assume that improved segmentation results can be obtained by automatically combining several segmentations of a given image into a single segmentation that in some sense is the average of all the segmentations. Nevertheless, the idea of averaging multiple segmentations poses an intriguing problem. The main difficulty arises when the segmentations have different numbers of segments and are inconsistent (e.g. one is not a strict refinement of another). The goal is to find a point in the "space of segmentations" which is close to all the individual segmentations. This space, however, has a fairly complicated structure. In this study we attempt to tackle this problem by introducing a general framework to form a single 'average' segmentation from a given set of segmentations.

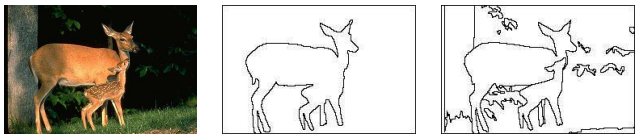
There is still another important hurdle to overcome regarding multiple segmentations. It is reasonable to assume that, given a set of segmentation algorithms, for each image some algorithms will perform well while other algorithms will perform poorly. The relative performance of each algorithm can also vary from one image to another as a function of the image content. In the extreme case where the quality of one of the segmentations is poor, or a segmentation is totally different from the other segmentations, it is preferable to be able to detect this situation and exclude this segmentation from the averaging process. Therefore, in the process of combining segmentations obtained by automatic algorithms, there is a need to estimate the quality of each algorithm in an unsupervised manner based only on the consistency of the segmentation compared to the other algorithms. For this reason, in addition to segmentation 'averaging', the approach proposed here provides a reliability measure for each segmentation.

Another reason to use segmentation averaging is supervised, or ground truth based evaluation, which is commonly employed for empirical comparison of algorithms. The evaluated segmentation is compared to a human segmentation using some type of metric to obtain a quantitative evaluation of the performance. Many evaluation methods have been proposed (see e.g. [21][16] [10]). Some methods focus on the regions and consider segmentation a standard data clustering task. Others focus on the boundaries between the segments and compare them to the reference boundaries, in statistical terms of miss detection and false positives, or precision and recall [14]. Apart from segmentation comparison, the existence of a dataset of manually segmented images can be useful for learning-based designs of segmentation procedures and tuning algorithm parameters. Recently, large image databases associated with manual segmentations have become available. The Berkeley Segmentation Dataset (BSDS300) consists of 300 natural images, each of which was hand-segmented by multiple human subjects [15]. The variability across observers, as is highlighted by the BSDS dataset, is due to the fact that each human chooses to segment an image at different levels of detail and focuses on different objects in the image. Figure 1 shows two examples from the BSDS dataset. The difference between different manual segmentations of the same image is easy to see. This inconsistency can also be systematically analyzed by estimating human performance w.r.t. to a

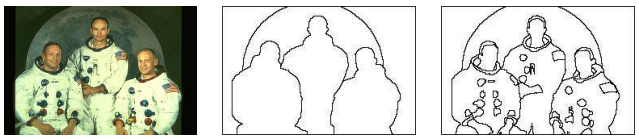
A. Alush is with the Engineering Faculty , Bar-Ilan University, Ramat-Gan 52900, Israel (e-mail: amiralush@gmail.com).

J. Goldberger is with the Engineering Faculty, Bar-Ilan University, Ramat-Gan 52900, Israel (e-mail: goldbej@eng.biu.ac.il).

given metrics by treating each ground-truth segmentation as a test segmentation and computing the metrics w.r.t. the other ground-truth segmentations. The variability inherent to human segmentation prevents annotated datasets from being used as a good 'gold standard'. Note that we do not claim that there is a correct segmentation; this is meaningless since this is not a well-defined task and is also subjective. Rather, we aim to represent a set of annotations of the same image by a single clustering that can be used for improved parameter tuning of clustering algorithms and for better performance evaluation.



(a) Image (b) annotation #1 (c) annotation #2



(d) Image (e) annotation #1 (f) annotation #2

Fig. 1. Examples of inconsistent human annotations from the BSDS dataset.

In the manual annotation collecting process, we also face the problem of inferring individual annotator quality in the absence of gold standard information. Estimating each annotator's quality can prove crucial to improving overall performance and mitigating the effect of annotation noise. In the case of multiple experts, annotation noise is a common problem since the quality of annotations may not be controlled. This problem is even more severe in the recently emerged online annotating marketplaces such as Amazon's Mechanical Turk (http://www.mturk.com) which is an online tool where remote workers are paid to complete small labeling and annotation tasks (see e.g. [28]). In such cases we have no a-priori knowledge or control of the reliability of individual labelers. Our approach can yield a reliability measure for online annotators as well.

Recently several studies have proposed utilizing multiple segmentations for various computer vision applications such as shape matching and object discovery (see e.g. [31], [20], [19], [26], [17], [24], [13]). We are not aware of previous approaches that have addressed the problem of measuring the reliability of each segmentation and taking this information into account in the segmentation fusing process.

The rest of the paper is organized as follows. In the next section we present a probabilistic model and a combined EM-ILP algorithm for averaging the results of multiple unreliable clusterings. Section 3 introduces an efficient method to solve the ILP problem. Section 4 presents an automatic segmentation averaging algorithm that is based on representing a segmentation as a set of binary decisions. Experimental results are shown in Section 5.

II. Learning from Multiple Unreliable Experts

Before delving into the problem of segmentation averaging, we first analyze the simpler case of combining multiple classifiers. Many classification algorithms rely on accurately labeled instances. However, what if there are multiple labeling sources (or experts) with different but unknown reliabilities? How can we jointly learn the accuracy of labeling sources and obtain the most informative labels for a given object set? To handle this challenge we start by defining a formal probabilistic framework. Assume there is a set of n objects or data points and each object has an unknown binary label. There is also a set of m experts and a judgment of expert l on the label of object i is denoted by $y_{li} \in \{0,1\}$. One example is where the experts are human annotators and the judgments are 'ground truth' binary labels assigned by the annotators. The same model, however, applies to a set of classification algorithms that provide binary labels on data elements.

Let the symbol y denote the entire matrix of labels (y_{li}). The i column in the matrix y, denoted by y_i, is the judgment set on object i. We assume that each object i has some (unknown) "correct" label, denoted by $x_i \in \{0,1\}$, and that each expert l is associated with an unknown probability p_l of making the correct decision. The vector of individual probabilities $(p_1, ...p_m)$ is denoted by θ. For simplicity, we assume that, in the absence of any information, the two possible labels of an object are equally likely; that is, for every i, the prior probability $p(x_i = 1) = 1/2$.

Our objective is to find the expert reliability parameters $p_1, ..., p_m$, and the correct object labels $x_1, ..., x_n$. Note that in this probabilistic modeling, the individual skills $p_1, ..., p_m$ are viewed as (unknown) parameters and the "correct" resolutions $\{x_i\}$ are treated as hidden binary random variables. The conditional probabilities of all the label decisions related to object i are:

$$p(y_i|x_i = b;\theta) = \prod_{l=1}^{m} p(y_{li}|x_i = b;\theta) = \tag{1}$$

$$\prod_{l|y_{li}=b} p_l \prod_{l|y_{li}\neq b} (1-p_l), \qquad b = 0,1$$

Therefore, the likelihood of the entire observation set y is:

$$p(y;\theta) = \frac{1}{2^n} \prod_{i=1}^{n} (p(y_i|x_i = 0) + p(y_i|x_i = 1)) \tag{2}$$

Thus optimality is obtained by the values of θ that maximize the probability of the observed data $p(y;\theta)$. That is, we wish to find a maximum likelihood estimator of θ:

$$\hat{\theta} = \arg \max_{\theta \in [0,1]^m} p(y;\theta) \tag{3}$$

Since the model has hidden variables and unknown parameters it is natural here to utilize the EM algorithm [7] to find the maximum-likelihood parameters. The EM algorithm handles the parameter estimation task by iterating the E and M steps. The equations take the following form. The E-step is:

$$p(x_i = b|y_i;\theta) = \frac{p(y_i|x_i = b;\theta)}{p(y_i|x_i = 0;\theta) + p(y_i|x_i = 1;\theta)}, \tag{4}$$

$$i = 1, ..., n \ , \ \ b = 0,1$$

where the terms $p(y_i|x_i = 0;\theta)$ and $p(y_i|x_i = 1;\theta)$ are defined in Eq. (1). The M-step is:

$$p_l = \frac{1}{n}\sum_{i=1}^{n} p(x_i = y_{li}|y_i;\theta), \quad l = 1,...,m \tag{5}$$

The general EM theory ensures that the likelihood function monotonically converges to a (local) maximum. To avoid a local maximum, we can use majority voting as an estimation for the correct labels to start the EM iterations. Variants of this EM procedure have been proposed in several studies (e.g. [32][8][23][27][34][33]). The Staple algorithm, suggested by Warfield et al., [32] addresses the problem of pixel-level object-background classification in the context of medical images and suggests a variant of the method described above to combine multiple expert markings to generate a single ground truth.

The probabilistic model described above is suitable for supervised classification tasks. However, it cannot be directly applied to an unsupervised clustering task. In this study we extend it to the problem of averaging multiple segmentations. The first step is formalizing a clustering as a binary classification task by viewing it as a set of many binary decisions.

A (hard) clustering of a set $S = \{1,...,n\}$ into n_c clusters is a function $C : S \to \{1,...,n_c\}$. The Rand Index [21] is a standard similarity measure between two clusterings C and C' of the set S that checks the consistency of the two clusterings over a range of binary decisions of the form "Are i and j in the same cluster?". Utilizing the Rand Index, we can transform a clustering C into a set of n-over-two binary decisions $x = \{x_{ij}|1 \le i < j \le n)\}$ such that $x_{ij} = 1$ if i and j are in the same cluster and 0 otherwise. Note that the correspondence between clusterings and binary decision sets is not one-to-one. Each clustering is represented by a different set of binary decisions but not every set of binary decisions corresponds to a valid clustering. The pairwise relation 'i and j are in the same cluster' is a transitive relation. If i,j and j,k are in the same cluster then necessarily i,k should be in the same cluster.

Assume the set $\{1,...,n\}$ is grouped into clusters such that the clustering is represented by the binary values $x = \{x_{ij}\}$. The grouping x is unknown but we are given a set of m expert' judgements on the clustering of the object set $\{1,...,n\}$. Each expert l is associated with an unknown probability p_l of making the correct binary decision x_{ij} for each object pair i,j. Each observed clustering is represented by a set of binary decisions as described above. Let $y_{lij} \in \{0,1\}$ be the judgement of the l-th expert whether objects i and j are in the same cluster or not. The conditional probability of the observation set $y = \{y_{lij}\}$ given the true clustering x is:

$$p(y|x;\theta) = \prod_{i<j}\prod_{l=1}^{m} p(y_{lij}|x_{ij};\theta) \tag{6}$$

where the probability that y_{lij} is equal to x_{ij}, is the expert reliability p_l. The only difference between the average classification model discussed above and the binary classification model of clustering averaging is that here the ground truth set x must satisfy the transitivity constraints. Note that in this simplified generative model we assume that the binary judgements of a given expert are done independently and we do not take into account the fact that the expert judgement set also forms a valid clustering that satisfies the transitivity constraints.

The probability of the observation set y is $p(y;\theta) = \sum_x p(x,y;\theta)$ where x goes over all the possible clusterings. The EM algorithm is next utilized to find the unknown parameters $p_1,...,p_m$ and the un-observed clustering x. To apply the EM algorithm we need to compute the marginal posterior probabilities $p(x_{ij} = 1|y;\theta)$ in the E-step. Since this computation is not tractable, we approximate it by computing instead the most likely clustering $\hat{x} = \arg\max_x p(x|y;\theta)$. (This is similar to the case of learning HMM parameters using the EM algorithm where we utilize the Viterbi algorithm instead of computing the exact E-step using the forward-backward algorithm). Given the most likely clustering $\hat{x}$, the (approximated) M-step is:

$$\hat{p}_l = \frac{2}{n(n-1)}\sum_{i<j}(\hat{x}_{ij}y_{lij} + (1-\hat{x}_{ij})(1-y_{lij})), \quad l = 1,...,m \tag{7}$$

where $\hat{p}_l$ is the updated reliability measure of the l-th expert. After the EM converges we obtain both the values of the clustering reliability parameters and the most likely clustering. To complete the learning procedure we only need to find an efficient way to compute the most likely clustering $\hat{x}$. Since we have no informative prior on the set of all clusterings, Bayes rule implies that $\arg\max_x p(x|y;\theta) = \arg\max_x p(y|x;\theta)$. We show next that this optimization problem can be viewed as an instance of Integer Linear Programming (ILP).

Denote the set of all the binary observations $y_{1ij},...,y_{mij}$ related to the object pair i,j by y_{ij}. It can be easily verified that $\log p(y_{ij}|x_{ij};\theta)$ can be written as:

$$\log p(y_{ij}|x_{ij}=1)1_{\{x_{ij}=1\}} + \log p(y_{ij}|x_{ij}=0)1_{\{x_{ij}=0\}} \tag{8}$$

$$\begin{aligned} &= \log\frac{p(y_{ij}|x_{ij}=1)}{p(y_{ij}|x_{ij}=0)}1_{\{x_{ij}=1\}} + \log p(y_{ij}|x_{ij}=0) \\ &= \log\frac{p(y_{ij}|x_{ij}=1)}{p(y_{ij}|x_{ij}=0)}x_{ij} + \log p(y_{ij}|x_{ij}=0) \end{aligned}$$

Hence,

$$\log p(y|x;\theta) = \sum_{i<j}\log p(y_{ij}|x_{ij};\theta) = \sum_{i<j} w_{ij}x_{ij} + \text{const} \tag{9}$$

such that 'const' is a scalar that is not dependent on x and

$$w_{ij} = \log\frac{p(y_{ij}|x_{ij}=1;\theta)}{p(y_{ij}|x_{ij}=0;\theta)} = \sum_l (-1)^{y_{lij}}\log(\frac{1-p_l}{p_l})$$

Therefore, $\arg\max_x p(y|x;\theta) = \arg\max_x \sum_{i<j} w_{ij}x_{ij}$ such that the maximization is done over all the sets of binary decisions x that correspond to a legal clustering (i.e. that are transitive). Observing that the transitivity constraints are linear, we can form this optimization problem as an Integer Linear programming (ILP) in the following way:

$$\begin{aligned} &\max_x \sum_{i<j} w_{ij}x_{ij} && && (10)\\ &s.t. && x_{ij} + x_{jk} - x_{ik} \le 1 && \forall i,j,k \\ &&& x_{ij} \in \{0,1\} && \forall i,j \end{aligned}$$

The constraint $x_{ij} + x_{jk} - x_{ik} \leq 1$ on the binary variables, enforces that whenever $x_{ij} = x_{jk} = 1$, it must also be the case that $x_{ik} = 1$. The constraints in (10) exactly enforce the transitivity relation which ensures that the set of binary decisions x indeed corresponds to a valid clustering. The graph clustering problem (10) is known as "correlation clustering" [3] [2]. The correlation clustering approach has several advantages. It does not require users to specify a parametric form for the clusters, nor to pick the number of clusters. Correlation clustering has been successfully applied in several areas such as computational linguistics [18], [9] and gene expression analysis [3]. In our setup we utilize the correlation clustering framework as a way to perform the E-step part of the EM algorithm. The drawback of the ILP approach is its high complexity which impedes its applicability for clustering of large sets. The ILP problem (10) is known to be NP-hard [2]. However, as we show in the next sections, it is still tractable for image segmentation applied to superpixels. In the next section we derive an efficient method for solving the ILP problem (10) by breaking it into small sub-problems.

III. ILP on Single-Edge Partition Trees

Assume we are given an undirected weighted graph $G = (V, E)$ such that the vertices (nodes) $V = \{1, ..., n\}$ are the data points we want to cluster. Denote the weight of an undirected edge $ij \in E$ by w_{ij}. The weights can be either positive, negative or zero. A positive weight w_{ij} is a local information indicating that i and j are more likely to be in the same cluster. A negative weight indicates that i and j are more likely to be in different groups. If $w_{ij} = 0$ or there is no edge connecting i and j, no local evidence is provided. We want to find a global clustering of the node set V that is most consistent with the local cues. Assuming that the score of a clustering, defined by the binary decision set $x = \{x_{ij} | 1 \leq i < j \leq n)\}$, is $\sum_{i<j} w_{ij} x_{ij}$, the optimal global clustering is the solution of the correlation clustering task defined by the ILP optimization problem (10).

The correlation clustering problem (10) is known to be NP-hard [2]. Our goal is to find an efficient algorithm to compute an exact solution to the ILP problem (10) in certain cases and a good approximation in others. Our approach is based on dividing the problem into smaller problems in which applying standard ILP solvers is still feasible.

We use the following notation. Let $V_1, ..., V_k$ be a partition of V. Denote $E \cap (V_i \times V_i)$ by E_i and for each $i \neq j$, $E \cap (V_i \times V_j)$ is denoted by E_{ij}. For $i \neq j$, an edge in E_{ij} is called a crossing edge. We start with the following observation.

Theorem 1: Assume we can divide V into disjoint subsets V_1, V_2 such that there is no crossing edge with positive weight between the subsets, (i.e., if $(i, j) \in E_{12}$ then $w_{ij} \leq 0$). Then the data clustering which is the optimal solution of (10) is a refinement of the partition V_1, V_2 and is obtained by separately applying the ILP optimization on each subset.

Proof: The cost function (10) can be written as:

$$\sum_{ij \in E} w_{ij} x_{ij} = \sum_{ij \in E_{12}} w_{ij} x_{ij} + \sum_{t=1,2} \sum_{ij \in E_t} w_{ij} x_{ij} \qquad (11)$$

Eq. (11) decomposes the variables that appear in the cost function (10) into three disjoint subsets. Hence by separately maximizing each sub-problem we get an upper bound on the optimal value. Since we assume that $w_{ij} \leq 0$ for all $(i, j) \in E_{12}$, the optimal zero-one solution of: $\max \sum_{ij \in E_{12}} w_{ij} x_{ij}$ is obtained by setting $x_{ij} = 0$ for all $(i, j) \in E_{12}$. Solving an ILP problem on each sub-graph (V_t, E_t), $t = 1, 2$ separately:

$$\begin{aligned} \max \quad & \sum_{ij \in E_t} w_{ij} x_{ij} && (12) \\ s.t. \quad & x_{ij} + x_{jk} - x_{ik} \leq 1 && \forall i, j, k \in V_t \\ & x_{ij} \in \{0, 1\} && \forall i, j \in V_t \end{aligned}$$

we get an upper bound on the optimal global solution. It can be easily verified that the combined solution (with $x_{ij} = 0$ for all $(i, j) \in V_1 \times V_2$) satisfies all the transitivity constraints in (10) and hence it is optimal. ■

Theorem 1 can be easily generalized to the case of dividing V into any number of subsets. The most refined partition $V_1, ..., V_k$ that satisfies the requirement of Theorem 1 (no positive weight crossing edge between the subsets) can be found by utilizing a greedy approach. We begin with some vertex $v \in V$ defining the initial set of vertices $V_1 = \{v\}$. Then, in each iteration, we look for a positive weight edge (u, v), where $u \in V_1$ and $v \notin V_1$. Then vertex v is brought into V_1. This process is repeated until no vertex can be added to V_1. We next choose a vertex outside of V_1 and start constructing V_2 from the remaining vertices, etc. We call the members of the obtained partition the 'positively connected components' (they are actually the connected components of the graph obtained by eliminating all the non-positive weight edges). The complexity of the algorithm applied to a n-vertex graph is $O(n^2)$. (Given a graph data-structure, it is straightforward to compute the connected components of a graph in linear time using either a breadth-first search or a depth-first search [11]).

As a result of Theorem 1, we can solve the ILP problem (10) for each positive connected component separately. Hence, hereafter in this section we assume that the graph is composed of a single positively connected component. In other words, we assume that the graph cannot be decomposed into two disjoint subsets where all the crossing edges are non-positive.

The key observation we use to form an efficient algorithm for the ILP problem (10) is the following.

Lemma 1: Assume we can divide V into two disjoint subsets V_1, V_2 such that there is just a single crossing edge between the subsets. Denote the crossing edge by $(v_1, v_2) \in E$, such that $v_1 \in V_1$ and $v_2 \in V_2$. Then the data clustering which is the optimal solution of (10) is obtained by separately applying the ILP optimization on V_1 and V_2 and then merging the two clusters that contain v_1 and v_2,

Proof: The graph connectivity assumption implies that the crossing edge (v_1, v_2) satisfies $w_{v_1, v_2} > 0$ since otherwise we can break the graph into two positively connected components. Since there is exactly one edge connecting V_1 and V_2, the cost function (10) can be written as:

$$w_{v_1, v_2} x_{v_1, v_2} + \sum_{ij \in E_1} w_{ij} x_{ij} + \sum_{ij \in E_2} w_{ij} x_{ij} \qquad (13)$$

$E_1, E_2, \{(v_1, v_2)\}$ form a partition of the variables that appear in the cost function (13) into three disjoint parts. Hence by solving the three sub-problems we get an upper bound on the optimal global solution. The solution of the sub-problems is obtained by setting $x_{v_1,v_2} = 1$ and solving the two smaller ILP problems on V_1 and V_2. It can be easily validated that the combined solution (and setting $x_{ij} = 0$ for all $i, j \in (V_1 \times V_2) \backslash E$) satisfies the transitivity constraints; hence it is the optimal solution. Setting $x_{v_1,v_2} = 1$ implies that the two clusters that contain the nodes v_1 and v_2 are merged into a single cluster. ■

Next we generalize Lemma 1, to the case of partitioning the graph into more than two subsets.

Let $V_1, ..., V_k$ be a partition of V into k disjoint subsets. We can define a new graph $G'(V_1, ..., V_k) = (V', E')$ where $V' = \{V_1, ..., V_k\}$ and for each $i \neq j$ $(V_i, V_j) \in E'$ if there is an edge between V_i and V_j, i.e. $E_{ij} \neq \varnothing$.

Definition : A partition $V_1, ..., V_k$ is a Single-Edge Partition Tree (SEPT) if $G'(V_1, ..., V_k)$ is a cycle-free graph (tree) and $|E_{ij}| \leq 1$ for all $i \neq j$. We call $V_1, ..., V_k$ the tree components of the SEPT representation. Figure 2 shows an example of a SEPT graph based on a partition of 13 nodes into three tree components.

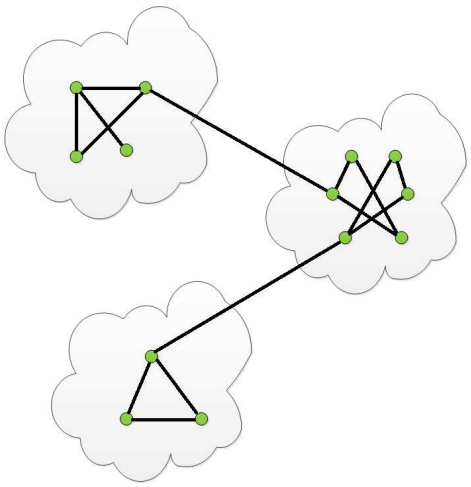


Fig. 2. An example of a SEPT with a three tree components of a 13 node graph.

Theorem 2: Let $V_1, ..., V_k$ be a partition of V that is a SEPT. Then the data clustering which is the optimal solution of (10) is obtained by separately applying the ILP optimization on each subset and then for each crossing edge $(v_i, v_j) \in V_i \times V_j$ merging the two clusters that contain the nodes v_i and v_j.

Proof: Since G has a single positively connected component and $G'(V_1, ..., V_k)$ is a tree, for each edge $(V_i, V_j) \in E'$ the corresponding edge $\{(v_i, v_j)\} = E \cap (V_i \times V_j)$ satisfies $w_{v_i,v_j} > 0$, otherwise we can break the graph G into two positively connected components.

We now prove the theorem using induction. The case of $k = 2$ follows directly from Lemma 1. Assume that $k > 2$ and that the theorem is true for every value less than k. Let $(V_i, V_j) \in E'$ be an edge in the SEPT. This edge corresponds to a single crossing edge $(v_i, v_j) \in V_i \times V_j$. Since G' is a tree, the edge (V_i, V_j) splits G' into two disjoint trees G'_1, G'_2, each of which has fewer than k nodes. Eliminating the edge (v_i, v_j) splits G into two sub-graphs G_1 and G_2 such that G'_1 is the SEPT partition of G_1 and G'_2 is a SEPT partition of G_2. The proof is concluded by utilizing the induction assumption on G_1 with SEPT G'_1 and G_2 with SEPT G'_2 and applying Lemma 1 on the graph G with a SEPT partitioning into two sets G_1, G_2 and the crossing edge (v_i, v_j). ■

To apply Theorem 2 to the ILP optimization problem (10), we first need to transform the given graph $G = (V, E)$ into a SEPT. Since for each tree component we still need to apply an ILP solver, we look for a SEPT structure with subsets that are as small as possible. The most refined SEPT partition of a connected graph $G = (V, E)$ can be found by utilizing the following greedy approach.

We begin with some vertex $v \in V$ defining the initial set of vertices $V_1 = \{v\}$. In each iteration of the algorithm we add one more element of V into the SEPT we are constructing. Assume we have already arranged some of the vertices into k disjoint subsets $V_1, ..., V_k$ that form a SEPT on $U = \cup_{i=1}^{k} V_i$. We look for a positive weight edge (u, v) where $u \in U$ and $v \notin U$ (since the graph is singly connected, there should be such v). Let $E(U, v) = E \cap (U \times \{v\})$ be the set of edges between v and nodes in U. Let $V_{i_1}, ..., V_{i_s}$ be the subsets in the current SEPT that are connected by an edge to v. If (u, v) is the only edge between v and U, define $\{v\}$ as a new subset and add an edge in the SEPT we are building between $\{v\}$ and the subset V_{i_1} that contains u. If (u, v) is not the only edge between v and U but all these edges are between v and V_{i_1}, add v to the subset V_{i_1}. The most complicated case is when $s > 1$, i.e. when there are edges between v and several subsets. In that case adding v and the associated edges will introduce loops and break the tree structure. In order to maintain a tree structure we have to merge all the subsets $V_{i_1}, ..., V_{i_s}$ and all other subsets that are in a path connecting any two of these subsets and add v to the merged subset. Any edge between one of the merged subsets and another subset is replaced with an edge from the merged subset and that subset. The algorithm terminates once all the nodes are incorporated into the constructed SEPT. The complexity of the algorithm applied to a n-vertex graph is $O(n^2)$. The algorithm is summarized in Algorithm Box 1.

After obtaining the subsets of the SEPT representation, for each subset U we need to solve the ILP problem (10) restricted to the subset U. If $|U|$ is small, we can use an ILP solver to find the optimal solution. Otherwise since the problem is NP-hard we need to use an approximation. A simple standard iterative approximation algorithm is the following [18] (see [9] for a comparative review of approximation methods). Start with an empty clustering and add the nodes one by one. Add each node v to the cluster which maximally improves the clustering score function or to a new singleton if all other options decrease the score. Then iterate over the nodes; at each step a single node v is removed from its cluster and placed in the cluster (that can be the singleton $\{v\}$) which yields the highest clustering score. This is repeated until the value of the score function cannot be further improved by moving a single node from one cluster to another. This iterative approach is a standard sequential clustering algorithm and can be easily done in linear

Algorithm 1 Constructing a Single-Edge Partition Tree

Input: A weighted undirected graph on $\{1, ..., n\}$ with weights w_{ij} that is composed of a single positively connected component.
$k = 1$
$V_1 \leftarrow \{1\}$
$U \leftarrow \{1\}$
$E' = \varnothing$
repeat
  Look for a positive weight edge $(u, v) \in E$ where $u \in U$ and $v \notin U$. Let $E(U, v) = E \cap (U \times \{v\})$ be the edge set between U and v. Let $V_{i_1}, ..., V_{i_s}$ be the s subsets that are connected by an edge to v.
  if $|E(U, v)| = 1$ **then**
    $k \leftarrow k + 1$
    $V_k \leftarrow \{v\}$
    $E' \leftarrow E' \cup \{(V_{i_1}, V_k)\}$
  else if $s = 1$ **then**
    $V_{i_1} \leftarrow V_{i_1} \cup \{v\}$
  else
    Merge all the subsets $V_{i_1}, ..., V_{i_s}$ and all other subsets that are in a path connecting any two of these subsets. Update the value of k, Add v to the merged subset.
  end if
  $U \leftarrow U \cup \{v\}$
until U = V

Algorithm 2 An efficient Solver for the ILP problem (10).

Input: A weighted undirected graph $G = (V, E)$ with weights $\{w_{ij}\}$.
Output: A clustering of the graph nodes.

Break the graph into positively connected components $V_1, ..., V_k$.
for $i = 1, ..., k$ **do**
  Apply Algorithm 1 on V_i to find its SEPT subsets $V_{i1}, ..., V_{is_i}$.
  for $j = 1, ..., s_i$ **do**
    Solve the ILP problem restricted to the subset V_{ij}. If $|V_{ij}|$ is small enough use an ILP solver to find the optimal solution, otherwise use an iterative approximation.
  end for
  Apply Theorem 2 to combine the solutions of the tree subsets $V_{i1}, ..., V_{is_i}$ into a consistent clustering solution of V_i.
end for
The clustering of V is the union of the clusters of its positively connected components.

time by computing the score for re-attaching v to all possible clusters. The approach presented in this section is summarized in Algorithm Box 2.

We note in passing that the ideas presented in this section cannot be further generalized in a straight-forward manner. Theorem 1 addresses the situation where all the edges between two subsets are non-positive. Theorem 2 addresses the situation where there is exactly one edge between two subsets and it is positive. If there are several positive edges between the subsets or there are negative edges in addition to a single positive edge between the subsets, there are cases where the optimal ILP solutions obtained on the subsets are no longer part of the global optimal solution.

The success of the approach described in this section depends on the ability to break the similarity weighted graph into a SEPT with small-size tree elements. In a general clustering problem setup where similarity information is provided for each object pair, the graph is fully connected and it cannot be decomposed into small subsets. In a segmentation task local similarity information is provided only for neighboring elements. The similarity graph here is relatively sparse and therefore there is a good chance of obtaining a SEPT representation that breaks the graph into small subsets. There is yet another factor determining the success of the algorithm - namely how consistent the input segmentations are. We can assume that reasonably good segmentation algorithms of the same image should not be very different.

IV. Averaging Multiple Unreliable Segmentations

Suppose now that our task is an unsupervised segmentation of a given image dataset and there are multiple experts with different but unknown reliabilities such that each of them provides a segmentation, possibly with a different number of segments. The challenge here is finding a way to jointly learn the accuracy of the experts and obtain the 'average' segmentation of the given image.

Although image segmentation is an instance of a clustering problem, there is yet another specific source of information that does not exist in a general clustering setup; namely the image spatial consistency. Neighboring pixels are more likely to be in the same cluster than pixels that are far apart. The pixels that belong to a single cluster are expected to be spatially connected. This spatial consistency results in a strong dependency among the pixel-pair binary decisions. One approach to overcome this problem is modeling the spatial structure using an MRF [32] and replacing the E-step (4) with an MRF optimization that encourages neighboring pixels to be in the same cluster. We, however, take another approach which is standard in many other vision applications, namely shifting from pixels to superpixels.

We propose a two-step algorithm to find an 'average' segmentation and expert reliabilities given several segmentations of the same image:

- Apply an unsupervised algorithm to group pixels into superpixels and project each segmentation onto the superpixel map.
- Apply the EM algorithm on binary decisions of the form: "Should we merge these two neighbor superpixels"? and finally, solving the ILP (10), we get the average segmentation.

Next, we describe the algorithm steps in a detailed way. The image is first preprocessed using a low-level segmentation based on local cues such as color and edges. That is, we oversegment the image into small, homogeneous regions,

known as superpixels. The number of superpixels ranges from several hundreds to several thousands. We tried several segmentation algorithms (e.g. the Watershed transform [30] and the Normalized Cuts [25]) and found that the Mean-Shift (MS) algorithm [5] is best suited for our application in the sense that the edges of a human annotation are almost always included in the Mean-Shift edge map. The use of superpixels as primitive objects for clustering reduces considerably the spatial dependence problem that exists between neighbor pixels. Much of this spatial dependency is already captured by the superpixel map. It also reduces the computational burden substantially.

Given the Mean-shift superpixel map, the pixel-level segmentation of each one of the segmentations is projected onto the superpixel map in the following way. Denote the expert clustering label of pixel x_i by $C(x_i)$. The induced expert clustering label of a superpixel is:

$$C(\text{superpixel}) = \arg\max_j |\{x_i \in \text{super-pixel} | C(x_i) = j\}| \tag{14}$$

where $|\cdot|$ is the size of the set. In other words, the induced superpixel label is the most frequent expert label among the pixels in the superpixel. The result of this projection step is a segmentation of the superpixels. Next we merge pairs of superpixels for which all the experts agree that are part of the same segmentation. This significantly reduces the problem size since the superpixel map is much more refined than the segmentations.

Note that there is a natural set of superpixels to use which is just that set defined by the segmentations themselves. Simply declare two pixels to be in the same superpixel if all the segmentations agree on their label. However, when implementing this approach we found that most of the superpixels are located along the original segmentation boundaries. This is due to the fact that the marking of the same boundary can be shifted by several pixels in each segmentation. This results in many very small superpixels that have a non-convex shape and may not even be connected. The problem is that the small marking differences yields a very noisy superpixel map with complicated neighborhood relations which is not suitable for the efficient ILP algorithm we want to apply. Merging the MS superpixels, based on consensus among the experts, yields a robust version of this approach.

The next step is transforming a superpixel segmentation into a set of binary decisions. Since we can assume that each segment is spatially connected, in our implementation we only consider pairs of superpixels that are neighbors (i.e. their boundaries intersect). The binary decision corresponding to a pair of neighbor superpixels is whether they are part of the same segment or not. If an expert assigns the same label to these two superpixels, the expert has decided that the two superpixels are part of the same object and should be merged and vice versa. Therefore the input to the EM-ILP algorithm is a binary set (y_{lij}) where l goes over all the input segmentations and i, j go over all the pairs of neighboring superpixels. $y_{lij} = 1$ means that the expert l assigns the same label to the two superpixels i and j.

The last step is applying the EM algorithm described in Section 2 where the approximated E-step is implemented using the ILP algorithm described in Section 3. The results of the algorithm are both the average segmentation and a reliability score for each input segmentation. We dub the proposed algorithm "Averaging Multiple Unreliable Segmentations" (AMUS). The AMUS algorithm is summarized in Figure 3. The AMUS algorithm was implemented in C using the EDISON [4] Mean-Shift source code. We used the lp-solve (http://lpsolve.sourceforce.net) to solve the ILP optimization sub-problems. Applying the AMUS algorithm on a multiple annotated image takes less than a second.

Input: An image with multiple segmentations.

Goal: Find the 'average' segmentation and the segmentations reliabilities.

Preprocessing: Project the segmentations onto to MS superpixel map and merge superpixels for which there is a consensus that they are part of the same object. Denote the binary decision of expert l on neighbor superpixels i and j by y_{lij}. Denote the number binary decisions by n.

The EM Algorithm:

E-Step:

$$w_{ij} = \sum_l (-1)^{y_{lij}} \log(\frac{1-p_l}{p_l})$$

Apply Algorithm 2 to solve the ILP problem:

$$\begin{aligned} &\max_x \sum_{i<j} w_{ij} x_{ij} \\ &s.t. \quad x_{ij} + x_{jk} - x_{ik} \le 1 \qquad \forall i,j,k \\ &\quad\;\; x_{ij} \in \{0,1\} \qquad \forall i,j \end{aligned}$$

M-Step:

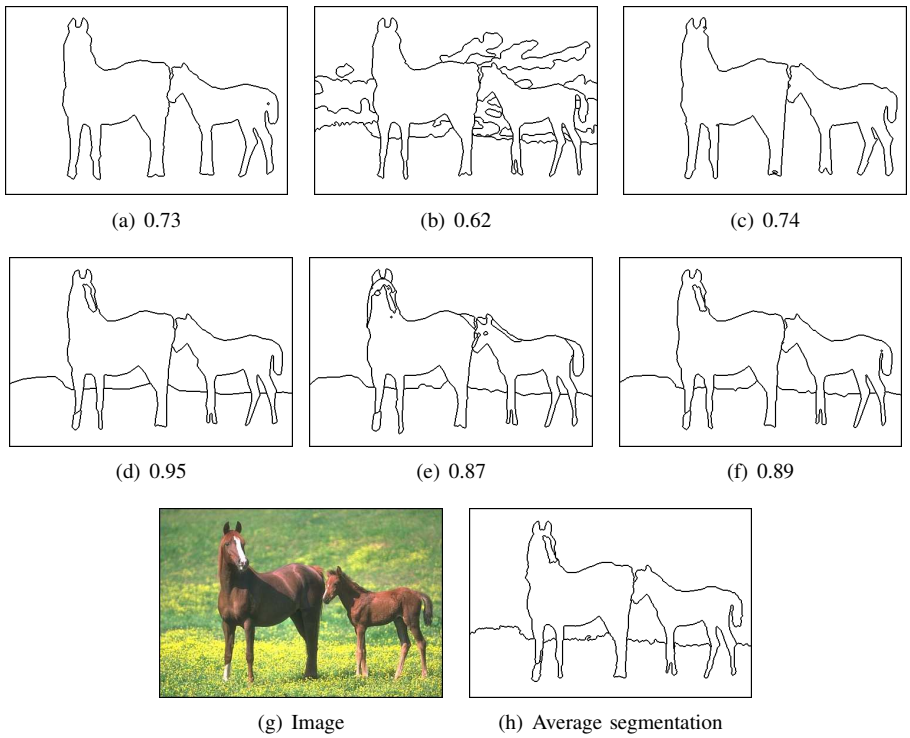
$$p_l = \frac{2}{n(n-1)} \sum_{i<j} (x_{ij} y_{lij} + (1-x_{ij})(1-y_{lij}))$$

The experts' reliabilities are the parameter set $\{p_l\}$ that is learned in the M-step. The average segmentation $\{x_{ij}\}$ is obtained from the ILP optimization.

Fig. 3. The Averaging Multiple Unreliable Segmentations (AMUS) Algorithm.

V. Experimental Results

We present visual and quantitative results of the AMUS algorithm on the BSDS300 dataset [15]. We first apply the AMUS algorithm on the manual segmentations that are part of the BSDS dataset. Then we apply the AMUS averaging procedure on the results of state-of-the-art automatic segmentation algorithms. Finally we show the effect of the efficient ILP algorithm on the segmentation averaging procedure.



(a) 0.73 (b) 0.62 (c) 0.74

(d) 0.95 (e) 0.87 (f) 0.89

(g) Image (h) Average segmentation

Fig. 4. An example of averaging multiple human annotations from the BSDS dataset.

(a) 0.93 (b) 0.69 (c) 0.93

(d) 0.74 (e) 0.70 (f) 0.65

(g) Image (h) Average segmentation

Fig. 5. An example of averaging multiple human annotations from the BSDS dataset.

A. Experiments based on manual segmentations

Figures 4 and 5 present two examples of averaging multiple human segmentations. Figures 4(a-f) are six human segmentations of the image shown in Figure 4g. Figure 4h shows the 'average' AMUS segmentation. The reliability level of each annotator is shown below the segmentation. It can be seen that the average segmentation describes the consensus among the annotators. In the average segmentation the horizon line is shown as suggested by most of the annotators. No fine details of the horses, however, are shown apart from the white spot on the head and one part of the leg as suggested by most of the annotators. The reliability of annotator (b) is low (0.69) because the annotation is too fine. The reliability of annotators (a) and (c) are low because the annotations are too coarse. Figure 5(a-f) shows six different manual annotations of the image shown in Figure 5g. In the 'average' segmentation shown in Figure 5h, the details of the background are not shown as suggested by four annotators. Each face is described by a single segment without explicit indication of the hair. This was also suggested by four annotators (not the same four as above). The reliability of annotator (b) is low (0.69) because the annotation is relatively too coarse. The reliability of annotator (f) is low (0.65) because the annotation is relatively too fine.

The segmentation created by the AMUS algorithm is expected to be the center of the individual segmentations. To systematically validate that this is indeed the case we conducted the following experiment. For each multiple annotated image from the BSDS dataset we first computed the average segmentation using the AMUS algorithm. Next, we measured the average distance between the AMUS result and each one of the manual segmentations of the same image. We compared this value to the average distance between pairs of manual segmentations of the same image. We used several standard methods for objective segmentation evaluation: the probabilistic Rand index (PRI) [29], the variation of information (VOI) [16], the Global Consistency Error (GCE) [14] and the boundary-based F-measure [14]. The results are shown in Table I. Note that the PRI and F-measure are similarity measures (higher is better) and the VOI and GCE are distance measures (lower is better). In all four measures we obtained improved results. The next experiment validates that the result in Table I is mainly due to the averaging algorithm and not to the mean-shift superpixel regularization. Table II shows the results of the same experiment where manual segmentations were first projected onto the mean-shift superpixel map. This projection aligns the segmentation boundaries and makes all the expert segmentations be more similar. Hence all the results in Table II are better than the corresponding results in Table I. However, on all four measures the AMUS algorithm still led to significantly improved results.

In the next experiment we consider AMUS as an additional manual segmentation and evaluate each segmentation with respect to the others. For each segmentation we compute the average distance score (after projecting the manual segmentations onto the superpixel map) from all the other segmentations of the same image. A fair comparison would be to compare the AMUS with each human annotator and show that the AMUS performs better (this is the way we compared AMUS with the automatic algorithms). The problem is that the manual annotator set is different for each image of the BSDS300 dataset. Instead, for each image we chose the best manual segmentation according to a defined criterion and averaged the distance results of the best segmentation across the dataset. The results are shown in Table III. The first line in Table III corresponds to choosing the best manual segmentation for each image using the averaged PRI score. The lines below show results of similar experiments using other scores to define the best manual annotation for each image. For one measure AMUS was first and for the other measures AMUS result was comparable to the best result.

We do not claim that there is a single correct segmentation; this is meaningless since segmentation is not a well-defined task and is also subjective. The BSDS dataset is mainly used here to exemplify the algorithm averaging performance. Representing a set of annotations of the same image by a single clustering can be used for improved parameter tuning of clustering algorithms and for better performance evaluation. Computing annotation quality and removing outliers if necessary is definitely applicable to a manually annotated dataset.

	PRI	VoI	GCE	F-measure
AMUS	**0.881**	**0.994**	**0.081**	**0.794**
Manual	0.867	1.116	0.084	0.787

TABLE I
COMPARISON OF THE AMUS SEGMENTATION AND THE MANUAL SEGMENTATIONS ON THE BSDS DATASET.

	PRI	VoI	GCE	F-measure
AMUS	**0.917**	**0.608**	**0.032**	**0.867**
Manual	0.893	0.824	0.048	0.827

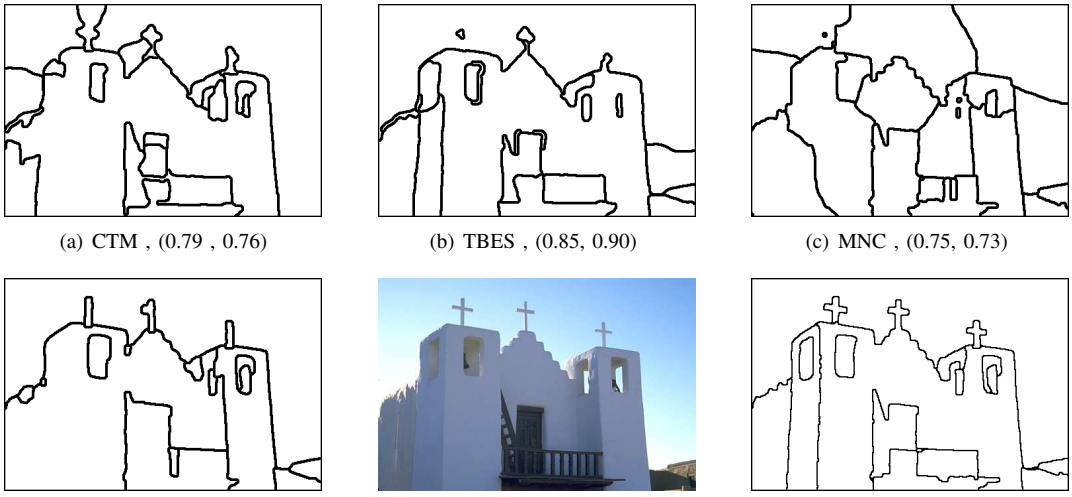
TABLE II
COMPARISON OF THE AMUS SEGMENTATION AND THE MANUAL SEGMENTATIONS ON THE BSDS DATASET AFTER PROJECTING THE MANUAL SEGMENTATIONS ONTO THE MEAN-SHIFT SUPERPIXEL MAP.

	PRI	VoI	GCE	F-measure
Best PRI	**0.920**	0.646	0.040	0.864
Best VoI	0.919	0.639	0.039	0.859
Best GCE	0.879	0.838	**0.027**	0.811
Best F	0.908	0.781	0.045	**0.898**
AMUS	0.917	**0.608**	0.032	0.867

TABLE III
COMPARISON OF THE AMUS SEGMENTATION AND THE BEST MANUAL SEGMENTATIONS IN EACH IMAGE.

B. Averaging automatic segmentations

We next applied the AMUS algorithm to a set of state-of-the-art segmentation algorithms that were applied on the test part of BSDS300. We compared the quality of the individual algorithm's segmentations to the average segmentation obtained by the AMUS algorithm. The four segmentation algorithms we used are Compression-based Texture Merging



(a) CTM , (0.79 , 0.76) (b) TBES , (0.85, 0.90) (c) MNC , (0.75, 0.73)

(d) UCM , (0.87, 0.69) (e) Image (f) AMUS, 0.88

Fig. 7. An example of averaging four automatic segmentation algorithms (CTM [35], TBES [22], MNC [6], UCM [1]). In each algorithm the number on the left is the PRI score and the number on the right is the reliability probability obtained by the EM algorithm.

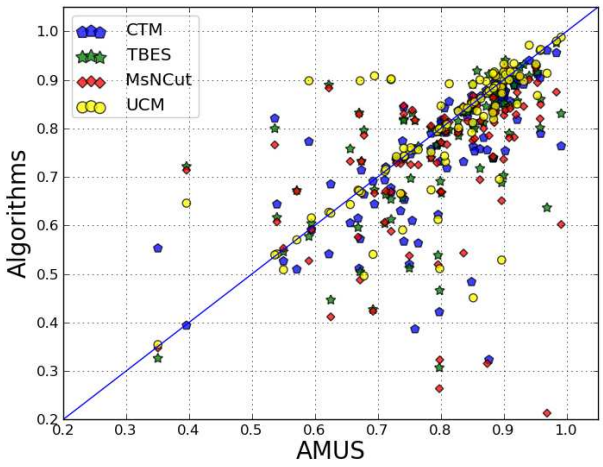
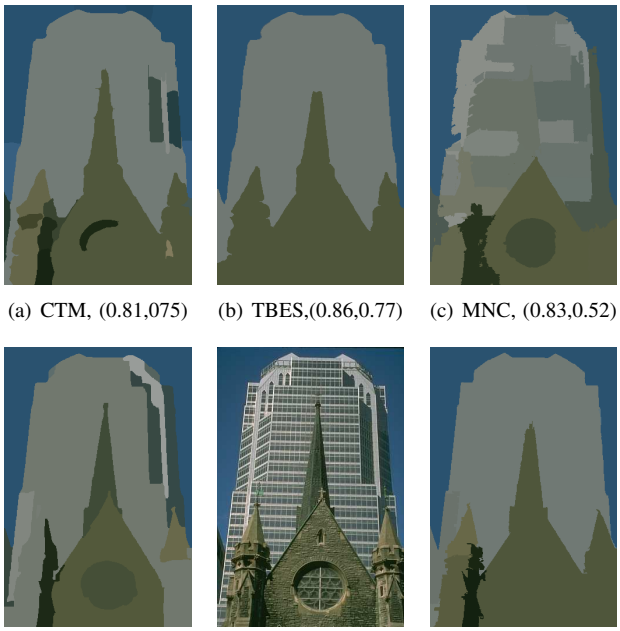


Fig. 6. A scatter plot of the PRI score of the AMUS algorithm versus the PRI scores of four automatic segmentation algorithms.



(a) CTM, (0.81,075) (b) TBES,(0.86,0.77) (c) MNC, (0.83,0.52)

(d) UCM, (0.82,0.79) (e) Image (f) AMUS, 0.88

Fig. 8. An example of averaging four automatic segmentation algorithms. In each algorithm the number on the left is the PRI score and the number on the right is the reliability probability obtained by the EM algorithm.

(CTM) [35], Texture Boundary Encoding-based Segmentation (TBES) [22], Multiscale Normalized Cut (MNC) [6] and Ultrametric Contour Maps (UCM) [1] (with ODS scale parameter 0.2). The segmentation results for each method were produced by publicly available implementations. Note that we only used state-of-the-art segmentation algorithms that produce region segmentation. Thus we did not use the "Global Probability of Boundary (color)" algorithm [12] even though it scored the highest on the BSDS dataset, since it produces only boundary probability maps. Figures 7 and 8 present two examples of averaging automatic segmentations. Comparative results over the BSDS dataset for the four automatic algorithms and the AMUS averaging are shown in Table IV. On three measures the AMUS segmentation obtained the best results and on one measure it came in second. Figure 6 shows a scatter plot of the PRI score [29] of the AMUS algorithm versus the PRI scores for the four automatic segmentation algorithms on the 100 test images of the BSDS dataset.

We also applied the AMUS algorithm on three variations of the CTM algorithm [35] by using three values of its texture parameter, 0.1,0.2 and 0.4. The standard method of parameter tuning is using a train image set. By utilizing the AMUS algorithm we avoid the need for parameter tuning. The most suitable parameter values are chosen by the AMUS algorithm. Moreover, for each image the AMUS algorithm automatically selects the parameter values that are suitable for that image. The results of applying the AMUS algorithm on the three variants of the CTM algorithm are shown in Table V. The results are averaged over the test portion of the BSDS dataset.

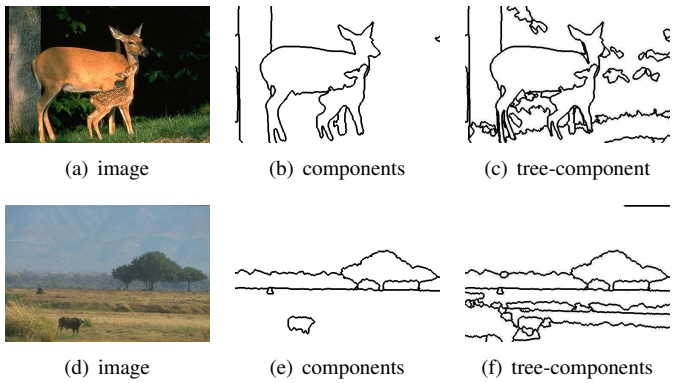
	PRI	VOI	GCE	F-measure
AMUS	**0.799**	**1.682**	**0.174**	0.634
CTM [35]	0.739	2.522	0.198	0.541
TBES [22]	0.792	1.792	0.194	0.581
MNC [6]	0.742	2.651	0.197	0.590
UCM [1]	0.791	1.740	0.176	**0.669**

TABLE IV
COMPARISON OVER THE BSDS DATASET OF FOUR SEGMENTATION ALGORITHMS AND THE AMUS AVERAGING ALGORITHM.

On three measures out of four, the AMUS performed better than all the CTM variants that were used to create the average segmentation.

	PRI	VoI	GCE	F-measure
AMUS	**0.761**	**1.874**	0.180	**0.598**
0.1	0.753	2.275	0.202	0.595
0.2	0.742	2.023	0.198	0.544
0.4	0.698	1.920	**0.166**	0.528

TABLE V
COMPARISON (OVER THE BSDS DATASET) OF THREE VARIANTS OF THE CTM ALGORITHM [35] AND THE AMUS AVERAGING ALGORITHM.



(a) image (b) components (c) tree-component

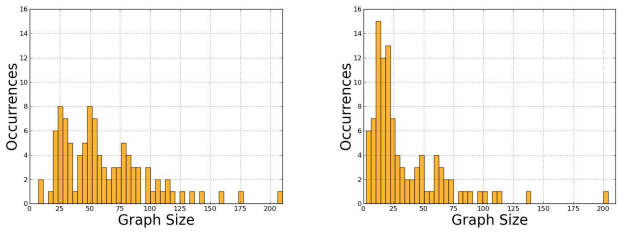
(d) image (e) components (f) tree-components

Fig. 9. Examples of breaking the global ILP problem into small sub-problems.

C. The efficient ILP algorithm

As we described in Section 3, to solve the ILP optimization we first break the graph into positively connected components. Then, for each component we construct its SEPT tree and apply the ILP optimization to each tree-subset separately. Since usually in most of the image regions the segmentations are consistent and the superpixel map is much more refined than the manual annotation segmentation maps, this step significantly improves run-time.

Figure 9 demonstrates the efficiency of the ILP computation on examples from the manually annotated BSDS dataset. For each example we show the original image, the image superpixels decomposition into connected components and the refined image superpixels decomposition of each component into tree-components of the SEPT tree. It is nice to see that the connected components have a visual interpretation of high level areas that elicited a consensus among annotators that they should be separated.



(a) Entire graph (b) Largest component

Fig. 10. Histograms of the ILP problem size. (a) Size of the superpixel map, (b) Size of the largest ILP subsystem.

The computational complexity of the AMUS algorithm depends on the size the ILP problem (10) we need to solve. Note that an ILP problem corresponding to a graph with 50 nodes has 1225 variables and 125,000 constraints.

Figure 10(a) shows histogram, computed on the test part of the BSDS300 dataset, of the number of superpixels in each image (after merging superpixels for which there is a consensus that they are part of the same object.). The average graph size is 62. Figure 10(b) shows histogram of the largest ILP subproblem we need to solve in each image using the method described in Section 3. The average graph size is reduced to 34. Reducing the size of the ILP problem from 62 to 34 enables us finding the exact solution, using standard ILP solvers, in milliseconds instead of hours.

To conclude, in this paper we proposed a framework for unsupervised learning in the presence of multiple (human or automatic) segmentations. The proposed algorithm iteratively establishes an average segmentation and measures the performance of the annotators given that average. Using this approach, annotations from many non-expert (and even anonymous) individuals can be potentially as good as annotations provided by an expert. Note that in the current study the AMUS algorithm was applied to each image separately. In cases where annotators segment a set of images, the AMUS algorithm can be applied simultaneously to the entire dataset. The only small modification needed to apply the algorithm globally is to modify the M-step such that the annotator updated reliability parameters are based on all the annotated images.

REFERENCES

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning Journal*, pages 86–113, 2004.
[3] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, pages 281–297, 1999.
[4] C. M. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. *Int. Conf. on Pattern Recog.*, 2002.
[5] D. Comaniciu and P. Meer. Mean Shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intell.*, pages 603–619, 2002.
[6] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, pages 1–38, 1977.

[8] P. Donmez, J. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. *Knowledge Discovery and Data Mining (KDD)*, 2009.
[9] M. Elsner and W. Schudy. Bounding and comparing methods for correlation clustering beyond ILP. *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 19–27, 2009.
[10] J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cuff. Yet another survey on image segmentation. *European Conference on Computer Vision*, 2002.
[11] J. Hopcrofta and R. Tarjan. Efficient algorithms for graph manipulation. *Communications of the ACM*, pages 372–378, 1972.
[12] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. *Computer Vision and Pattern Recognition*, 2008.
[13] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. *Brit. Machine Vision. Conf.*, 2007.
[14] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. on Pattern Analysis and Machine Intell.*, pages 530–549, 2004.
[15] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Int Conf. Computer. Vision*, 2001.
[16] M. Meila. Comparing clusterings: an axiomatic view. *Int. Conf. on Machine Learning*, 2005.
[17] M. Mignotte. A label field fusion bayesian model and its penalized maximum rand estimator for image segmentation. *IEEE Trans. on Image Processing*, pages 1610–1624, 2010.
[18] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. *Annual Meeting of the Association for Computational Linguistics*, pages 104–111, 2002.
[19] C. Pantofaru, C. Schmid, and M. Hebert. Object recognition by integrating multiple image segmentations. *European Conference on Computer Vision*, 2008.
[20] A. Rabinovich, T. Lange, J. Buhmann, and S. Belongie. Model order selection and cue combination for image segmentation. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2006.
[21] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(366):846–850, 1971.
[22] S. Rao, H. Mobahi, A. Yang, S. Sastry, and Y. Ma. Natural image segmentation with adaptive texture and boundary encoding. *Asian Conf. on Computer Vision*, 2009.
[23] V. Raykar, S. Yu, L. Zhao, A. Jerebko, C. Florin, G. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. *Intl. conf. on Machine Learning (ICML)*, 2009.
[24] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2006.
[25] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 888–905, 2000.
[26] V. Singh, L. Mukherjee, J. Peng, and J. Xu. Ensemble clustering using semidefinite programming with applications. *Machine Learning*, 79:177–200, 2010.
[27] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of Venus images. *Neural Information Processing Systems*, 1995.
[28] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. *IEEE Workshop on Internet Vision at CVPR*, 2008.
[29] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Trans. on Pattern Analysis and Machine Intell.*, pages 929–944, 2007.
[30] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 583–598, 1991.
[31] H. Wang and J. Oliensis. Rigid shape matching by segmentation averaging. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 32:619–635, 2010.
[32] S. K. Warfield, K. H. Zou, and W. M. Wells. Simultaneous truth and performance level estimation (STAPLE): an algorithm for the validation of image segmentation. *IEEE Trans. Med. Imag.*, 23:903–921, 2004.
[33] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Neural Information Processing Systems*, 2010.
[34] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Neural Information Processing Systems*, 2009.
[35] A. Yang, J. Wright, Y. Ma, and S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 110(2):212–225, 2008.

Amir Alush received the B.Sc. degree (cum laude) in bio-medical engineering in 2007 and the M.Sc. degree in bio-medical engineering in 2009 both from the Tel-Aviv university. He is currently a Ph.D. student at the Bar-Ilan University working on discrete optimization problems and their applications in image and video data and medical imaging.

Jacob Goldberger received the B.Sc. degree (cum laude) in mathematics and computer science from the Bar-Ilan University, Ramat Gan, Israel, the M.Sc. degree (cum laude) in mathematics, and the Ph.D. degree in Engineering, from the Tel-Aviv University, Tel-Aviv, Israel. In 2004 he joined the Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel, where he is currently a faculty member. His main research interests include machine learning with applications to information theory, computer vision, medical imaging, speech recognition and natural language processing.