2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Scalable real-time object recognition and segmentation via cascaded, discriminative Markov random fields

Paul Vernaza, Daniel D. Lee
{vernaza,ddlee}@seas.upenn.edu
GRASP Laboratory
University of Pennsylvania, Philadelphia, PA 19104

Abstract— **We present a method for real-time simultaneous object recognition and segmentation based on cascaded discriminative Markov random fields. A Markov random field models coupling between the labels of adjacent image regions. The MRF affinities are learned as linear functions of image features in a structured max-margin framework that admits a solution via convex optimization. In contrast to other known MRF/CRF-based approaches, our method classifies in real-time and has computational complexity that scales only logarithmically in the number of object classes. We accomplish this by applying a cascade of binary MRF-classifiers in a way similar to error-correcting output coding for general multiclass learning problems. Inference in this model is exact and can be performed very efficiently using graph cuts. Experimental results are shown that demonstrate a marked improvement in classification accuracy over purely local methods.**

I. Introduction

We consider the problem of simultaneous object recognition and segmentation of color images. This is a problem that appears in the literature under many names and variations, some of which we will review later. Simply stated, our problem is this: given a training set of fully labeled images, label new images in a way similar to the way the training images were labeled. We define a labeled image to be an image in which every pixel is labeled with a class label, such as "robot," "dog," or "camera."

We are motivated by robotics applications, where it is necessary to make such inferences in real-time. The method presented in this paper is accordingly efficient enough for evaluation in real-time. Unlike other known methods that perform real-time object recognition and/or segmentation, our method is able to learn contextual relationships via a MRF model and is able to efficiently and exactly evaluate the associated global inference problem in real-time.

Our method is also scalable in a number of ways that make it particularly suitable for practical applications. First, its worst-case computational complexity scales approximately linearly in the number of objects present in the scene, but only logarithmically in the total number of object classes. As we will show later, this is in contrast to more obvious approaches whose complexities scale at least linearly in the total number of object classes. From a computational perspective, this particular attribute easily allows our system to scale to tens of thousands of object classes or more—although we note that other practical considerations would likely become limiting factors before our system could be scaled to such magnitudes.

At training time, our method is also scalable in the sense that it can easily cope with vast amounts of training data. We employ a subgradient-based training algorithm that needs space that is only linear in the number of training examples. The algorithm is also inherently parallelizable in the sense that, given N processors, one could solve the training problem with N examples in approximately the same amount of time as necessary to solve it with only one example. The training algorithm is also inherently parallelizable with respect to the number of classes.

We begin our exposition with a brief summary of how our method works and how it relates to comparable approaches. We then describe in detail our method and its derivation in sections III and IV. Finally, we show experimental results in section V that show the method's promise compared to purely local appraches.

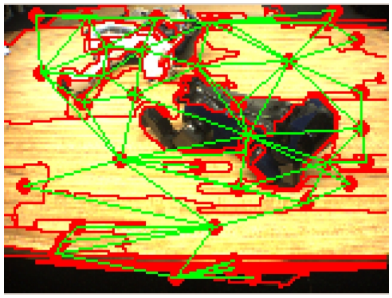
II. Method overview

In this section, we first describe at a high level how the classifier is evaluated and trained.

First, given an image to be classified, we perform an initial unsupervised oversegmentation of the image, as shown in Figure 1. We use the "superpixel" segmentation method of Felzenszwalb and Huttenlocher [1] for this step, primarily because it is efficient enough for real-time evaluation. We then create a graph where each node is a superpixel, and the graph has an edge between two nodes if and only if their corresponding superpixels are adjacent. Features that are properties of the nodes (superpixels) are computed and notionally attached to the graph, in addition to "edge features" that are properties of pairs of adjacent superpixels. This step results in a concise "feature graph" that represents the contextual structure of the image.

Suppose for now that we have a good classifier that is able to use this contextual information, but this classifier is only a binary classifier. Our idea is to leverage this good binary classifier to build a multiclass classifier. The way in which this is done is similar to the way error-correcting output codes are used to transform multiclass learning problems into binary problems [2]; we assign a binary code to each class, and train binary classifiers to generate those codes from input examples.

In particular, the binary code we assign to each class is an encoding of a path in a binary tree where each leaf is a class. The predicted code for a new example is obtained



(a) Superpixels, graph structure

Fig. 1. Initial superpixel segmentation and MRF graph structure of an input image.

by evaluating a sequence of classifiers, each of which adds one bit to the predicted code. Since the length of any code is logarithmic in the number of classes, evaluating a single example requires only a logarithmic number of classifier evaluations. Equivalently, our method can be described as a spatial cascade of contextual binary classifiers.

Therefore, the key to this strategy is the underlying binary classifier. We take advantage of a contextual model that admits exact, tractable learning and inference methods when restricted to the binary case; namely, the maximum margin Markov (M^3N) network [3]. By using M^3N in this binary coding framework, we sidestep the intractability associated with the direct application of M^3N to the multiclass case.

In a nutshell, M^3N learns mappings from features to MRF affinities that will hopefully induce the desired labeling as the maximum probability configuration of the MRF generated by applying these mappings to a given feature graph. The tractable version that we use here is able to learn "associative" potentials; i.e., it can learn positive correlations between the labels of adjacent nodes in the graph, also known as "guilt by association."

A. Related work

There is a growing body of literature concerning contextual image segmentation that bears some resemblance to our work [4] [5] [6] [7]. As in our method, these methods build a probabilistic model that introduces spatial correlation between labels of neighboring pixels or image regions. These models are then trained in a discriminative way, usually with the goal of maximizing the likelihood of data under a parametrized likelihood function. Inference and learning in these models is typically intractable, and one must resort to approximate methods for both. None of these methods are suitable for real-time applications.

With respect to these methods, the type of M^3N our method is built on has an advantage in that inference and learning are both tractable and exact, although some expressivity must be sacrificed to get these benefits. Such is the case in [8], which uses the same general framework as the one used here, but has the disadvantage of being restricted to binary prediction. We will show in this work how to circumvent this restriction while staying in the realm of tractable contextual inference and learning.

The work of Shotton et. al. [9] is noteworthy in that, like our method, it also performs segmentation and recognition in real-time; however, the real-time performance of their method is only achieved without applying a global contextual model, the use of which would most likely improve their classification results. In fact, our method could be used to add global contextual reasoning to such a classifier with little performance penalty simply by using features generated from the output of the classifier. Unfortunately, a detailed discussion of this idea is outside the scope of this paper.

We are also concerned with the issue of computational scaling in terms of number of object classes, which has been examined before in [10]. That work also claims approximately logarithmic scaling of computational complexity with respect to number of classes. However, this is framed in the paradigm of using sliding window detectors for object detection, and does not leverage a contextual model.

Finally, we note that Isukapalli et. al. have previously applied cascaded binary classifiers to solve the multiclass classification problem [11] and also commented on its applicability to problem of scaling with respect to classes. In contrast to that work, however, we employ this trick to escape the inherent computational difficulty in evaluating multiclass contextual models.

III. Learning discriminative MRFs

We begin our detailed discussion by considering first the binary case that is the basis for the multiclass algorithm. Although an equivalent derivation of the following appears in [8], we present a simpler, more intuitive derivation of the algorithm here.

Suppose we have generated a feature graph from an image as described in section II. Our first task is to define the "best" binary labeling of the nodes given this information. Let $y \in \{0,1\}^N$ be a labeling of the N nodes in the graph, and let x_i be a feature vector associated with the ith node. We define a unary energy function $E_w^U(y)$ parametrized by weights w:

$$\begin{aligned} E_w^U(y) &= \sum_i \begin{cases} -\langle w_0, x_i\rangle & \text{if } y_i = 0 \\ -\langle w_1, x_i\rangle & \text{if } y_i = 1 \end{cases} \\ &= -[\sum_i \langle w_1, x_i\rangle y_i + \langle w_0, x_i\rangle (1-y_i)] \end{aligned} \quad (1)$$

Consider what happens if w_0 and w_1 are both equal to a fixed vector w, and we find $y^* = \text{argmin}_y E_w^U(y)$. Clearly $y_i^* = (\text{sgn}\langle w, x_i\rangle + 1)/2$; i.e., we choose y_i according to which side x_i falls with respect to the hyperplane w. Conversely, given the labels y, finding w amounts to the classical problem of finding a hyperplane that separates the positively labeled x_i from the negatively labeled x_i.

We add context to this model by adding the following binary energy function:

$$\begin{aligned} E_w^B(y) &= \sum_{<ij>} \begin{cases} -\langle w_{00}, x_{ij}\rangle & \text{if } y_i = y_j = 0 \\ -\langle w_{11}, x_{ij}\rangle & \text{if } y_i = y_j = 1 \end{cases} \\ &= -[\sum_{\langle ij\rangle} \langle w_{11}, x_{ij}\rangle y_i y_j \\ &\quad + \langle w_{00}, x_{ij}\rangle (1-y_i)(1-y_j)] \end{aligned} \quad (2)$$

This energy function is dependent on edge features x_{ij}. Minimizing the combined energy $E_w(y) = E_w^U(y) + E_w^B(y)$ is like minimizing $E_w(y)$, except that adjacent nodes are given feature- and label-dependent bonuses for agreeing on a label. We note that a limitation of this model is that $\langle w, x_{ij} \rangle$ is required to be nonnegative, which is usually achieved by enforcing that w_{00}, w_{11}, x_{ij} all be nonnegative. This is the so-called *submodularity* assumption required for tractable inference via graph cuts [12].

This energy function therefore resolves both the matter of what is the best labeling given a parameter vector w, and how to find the best labeling efficiently. The solution to an energy function such as $E_w(y)$ can be obtained in polynomial time via network flow optimization [12]. We need not concern ourselves with the details of this optimization, except to note that very efficient codes exist to solve this problem [13].

We now turn to the issue of learning w. Let $\hat{y}$ be a desired labeling. The M^3N objective is that the energy of the desired labeling $E_w(\hat{y})$ induced by the parameters w should be less than the energy of any other labeling by a margin equal to the Hamming loss function

$$L(\hat{y}, y) = \sum_i [y_i \neq \hat{y}_i] = \sum_i y_i(1 - \hat{y}_i) + \hat{y}_i(1 - y_i) \quad (3)$$

This yields the objective

$$E_w(\hat{y}) \leq E_w(y) - L(\hat{y}, y), \ \forall y \in \{0,1\}^N \quad (4)$$

We can replace this exponential number of constraints by the most restrictive constraint, yielding

$$E_w(\hat{y}) \leq \min_{y \in \{0,1\}^N} E_w(y) - L(\hat{y}, y) \quad (5)$$

Adding regularization on w and a slack variable yields the optimization problem

$$\begin{array}{ll} \min_{w \in \mathcal{W}} & \|w\|^2 + C\xi \\ \text{subject to} & E_w(\hat{y}) \leq \xi + \min_{y \in \{0,1\}^N} E_w(y) - L(\hat{y}, y) \\ & \xi \geq 0 \end{array} \quad (6)$$

where $w \in \mathcal{W}$ indicates necessary constraints on w— i.e., the components corresponding to edge features should be nonnegative, as previously discussed. Writing the constraint as

$$E_w(\hat{y}) - (\min_{y \in \{0,1\}^N} E_w(y) - L(\hat{y}, y)) \leq \xi \quad (7)$$

should make it clear that this constraint must hold with equality at an optimum solution. This allows us to substitute the expression on the left directly into the objective yielding the following convex, nondifferentiable objective function:

$$\min_{w \in \mathcal{W}} \|w\|^2 + C[E_w(\hat{y}) - (\min_{y \in \{0,1\}^N} E_w(y) - L(\hat{y}, y))] \quad (8)$$

This objective can be minimized by a subgradient algorithm. The description of the subdifferential ∂ is given by Danskin's theorem [14]. Let $\phi(w, y) = -(E_w(y) - L(\hat{y}, y))$, and let $Y_0(w) = \{\bar{y} | \phi(w, \bar{y}) = \max_{y \in \{0,1\}^N} \phi(w, y)\}$. By Danskin's theorem,

$$\partial \max_w \phi(w, y) = \text{conv}\{\frac{\partial \phi(w, y)}{\partial w} | y \in Y_0(w)\} \quad (9)$$

Therefore, we can find a subgradient of the objective by finding a gradient of ϕ with respect to w at a y^* that minimizes $E_w(y) - L(\hat{y}, y)$. To be explicit, we have the following subgradients of the slack term ξ:

$$\sum_i x_i(\hat{y}_i - y^*) \in \partial_{w_0} \xi \quad (10)$$

$$\sum_i x_i(y^* - \hat{y}_i) \in \partial_{w_1} \xi \quad (11)$$

$$\sum_{<ij>} x_{ij}((1 - y_i^*)(1 - y_j^*) - (1 - \hat{y}_i)(1 - \hat{y}_j)) \in \partial_{w_{00}} \xi \quad (12)$$

$$\sum_{<ij>} x_{ij}(y_i^* y_j^* - \hat{y}_i \hat{y}_j) \in \partial_{w_{11}} \xi \quad (13)$$

In summary, this leads to the following iterative algorithm for minimizing the objective:

$$w \leftarrow \Pi \left(w - \eta \left(2w + \begin{bmatrix} \partial_{w_0} \xi \\ \partial_{w_1} \xi \\ \partial_{w_{00}} \xi \\ \partial_{w_{11}} \xi \end{bmatrix} \right) \right) \quad (14)$$

where $\partial \xi$ denotes any vector in the subgradient, and Π denotes projection onto the feasible set:

$$\Pi \left(\begin{bmatrix} w_0 \\ w_1 \\ w_{00} \\ w_{11} \end{bmatrix} \right) = \begin{bmatrix} w_0 \\ w_1 \\ \max(w_{00}, 0) \\ \max(w_{11}, 0) \end{bmatrix} \quad (15)$$

This is only one possible subgradient update rule. In practice, we have found that it is helpful to employ a "heavy ball" method [14]. In this method, the weight update is passed through an IIR filter that smooths out discontinuous jumps caused by the nondifferentiability of the objective.

We also remark that, as noted in [8], this type of subgradient method scales well for the case of vast amounts of training data, for a few reasons. First, the memory requirements are linear in the input size, since we only need to compute multiples and sums of the training features. Second, as mentioned earlier, the subgradient calculation itself is embarassingly parallelizable, the reason being that the required graph-cut optimization can be solved independently for each training image.

IV. Multiclass extension

The inference and learning method of the previous section may be used to solve the complete object recognition and segmentation problem given that there are only two object classes; this method takes as input a feature graph and outputs a binary labeling of the nodes that optimizes a global energy function that induces hopefully-correct segmentations on the training set. We now consider how to leverage this binary contextual learning algorithm to generate an effective multiclass learning algorithm.

Suppose we are given a binary tree such as the one depicted in Figure 2. Each node is associated with a *scope* of object classes. The root node contains all classes, each leaf node is assigned a distinct class, and each internal node evenly splits the classes within its scope between its two

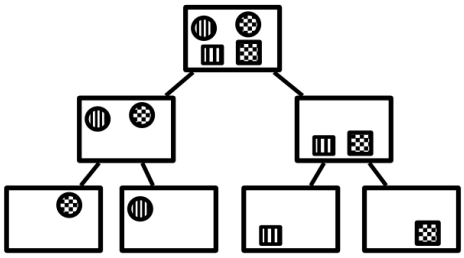


Fig. 2. An illustration of the tree structure used for multiclass classification in a toy problem with four classes.

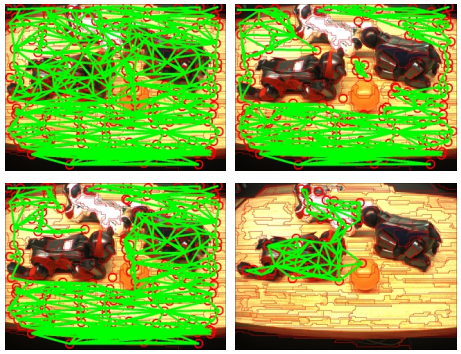


Fig. 3. Feature graphs and subgraphs generated during the training process.

children. Given such a tree, our method works as follows. Each internal tree node takes as input a feature graph from its parent. It then attempts to label each node in this graph according to which child's scope contains the true class of the node. The tree node then creates two disjoint subgraphs. Each subgraph contains only those nodes within the scope of one of the child nodes, and only the edges between such nodes; i.e., it is the subgraph induced by the set of nodes belonging to the scope of a child. Classification begins by submitting the original image feature graph to the root node, and it ends when every node in the graph has propagated down to a leaf node. The final classification of a node (superpixel) is the class assigned to the leaf node it eventually reaches. As mentioned earlier, this method can be viewed as a certain type of binary coding of a multiclass problem—the code we assign to each class is the sequence of "left-right" moves along the unique path from the root to the class's leaf.

Each of these tree node classifiers is exactly an instance of a binary discriminative MRF classifier, as discussed in section III. As in ECOC, this allows us to use that classifier as a subroutine in constructing the multiclass classifier. In order to train a given classifier on a given input feature graph, we label each node in the feature graph with the identity of the child (0 or 1) whose scope encompasses the node's class. We then use the discriminative binary MRF classifier to learn parameters for this classifier that induce this labeling of the feature graph. We then create subgraphs induced by this split, as before, and feed the subgraphs as input to this classifier's children classifiers. Some sample subgraphs generated during the training process are shown in Figure 3.

It is worth noting that each of the classifier nodes may be trained independently and in parallel, yielding a computational complexity that does not scale with the number of classes provided that enough processors are available.

A. Performance considerations

We can now analyze the performance of our approach and how it compares to other possible variations of the M^3N framework. First, we note that, in the worst case, our method requires $\min(|V|, N-1)$ binary classifier evaluations, where $|V|$ is the number of nodes in the feature graph. This occurs when every node in the image is labeled with a different class. At the other extreme, if the classifier performs perfectly, and there are K objects in the scene, the classifier performs no more than $K \log N$ internal classifier evaluations—in this case, exactly K groups propagate down to the leaves of the tree, evaluating $\log N$ classifiers. Some of these are shared between classes, so $K \log N$ is an upper bound on the total number of binary classifier evaluations. Making the reasonable assumption that the number of predicted classes is about equal to the number of classes present, we thus expect the running time to be $O(K \log N)$.

We note that we initially considered other variations of the M^3N framework, but none of the variations we considered have the potential for sublinear scaling in the total number of classes. One such variation arises by using a multi-label variation of Eq. 8. The derivation in III is fairly general in that Eq. 8 admits a solution via a subgradient method under weak conditions, assuming that an inference method exists to perform the inner energy minimization. In the multi-label (Potts model) case, the α-expansion and $\alpha-\beta$-swap algorithms [15] have enjoyed much success for certain problems; however, these methods require iterations of a number of graph-cut operations that is at least linear in the number of classes. Furthermore, these inference methods are only approximate.

We also note that the solution of the M^3N objective, under certain restrictions, has a convex relaxation that has also seen success in the multi-label case [3]. However, this solution is based on the solution of a quadratic program that does not scale well with very large training sets.

V. Experiments

We will first briefly discuss the practical consideration of feature selection before discussing experimental results. Our features were fairly simple for two reasons—the first being that we are only interested in features that can be computed in real-time, and the second being that we wish to show that even very simple features have the potential to perform well when endowed with the benefit of context.

For node features, we chose to compute histogram-based features. The image was converted to HSV and independent histograms were computed for each image channel over the area of each superpixel. These histograms were then normalized and concatenated into the node feature vector. We additionally compute histograms of image gradients within each superpixel. A constant bias feature was also added.

Edge features were computed as functions of node features. In particular, for each adjacent pair i, j of nodes with kth node features $x_i^{(k)}$ and $x_j^{(k)}$, respectively, we computed the kth edge feature $\sqrt{x_i^{(k)} x_j^{(k)}}$. These features are intended to convey co-occurrence of the kth feature. Finally, we computed the cosine of the angle between the adjacent nodes' feature vectors, which is motivated from an entropy minimization perspective; if joining two superpixels results in a relatively low-entropy feature distribution, we may want to learn a preference to join them.

We implemented our method in C++ (open-source code available[1]), using the graph cut code provided by Boykov and Kolmogorov [13], and the graph-based image segmentation code provided by Felzenszwalb [1]. We scaled our input images to 160x120 pixels before doing any subsequent processing. At this resolution, running on a 1.83 GHz Core 2 Duo computer, the total processing time per frame (including pre-segmentation, feature computation, and classifier evaluation) approximately varied between 90 and 120 ms.

Training images were obtained by recursive background segmentation. First, a static background would be learned, and an object placed against that background. Differencing was then used to obtain a segmentation mask for that object. This new scene was then held static while a new object was introduced to the scene, and so on. This resulted in somewhat imperfect ground truth segmentations, as seen in Figure 5.

We evaluated our method on a small dataset of 57 images obtained in this way, featuring 11 object classes, including the background (Figure 4). We performed 5-fold cross-validation to estimate the classification accuracy. Classification accuracy was calculated as the percentage of non-background pixels correctly classified. Background pixels were excluded because of their prevalence in the image, which makes it possible to get a high accuracy rate by simply guessing all background, unless these are excluded from the accuracy calculation.

We compared our method to a linear classifier on the node (local) features alone. This classifier was obtained by training the contextual classifier after constraining the edge weights to be zero.

A. Results

In summary, the cascaded MRF classifier achieved a mean accuracy of 75.5%—nearly a 60% improvement in classification accuracy over the purely local classifier, which scored a mean accuracy of 42.2%. Figure 5 shows some typical scenes labeled by both methods compared to the ground truth segmentation. The first thing immediately apparent from Figure 5 is the noisy nature of many of the images classified by the local classifier. The cascaded MRF results are significantly smoother, and tend to exhibit more "decisive" labelings, where an object will be either entirely correct or incorrect. Notable mistakes by the cascaded MRF happen on one of the robots, which is sometimes misclassified as a mouse pad, and the mouse pad, which is sometimes misclassified as a robot.

[1] `http://www.seas.upenn.edu/~vernaza/mrfseg/`

Fig. 4. A sample of images from a test set

Class confusion matrices for the two approaches are also shown in Table I. Context seems to help especially for "grayRedAibo" and "grayBlueAibo", which are both gray with red and blue patches, respectively. Without context, these classes exhibit significant confusion between themselves and miscellaneous other classes.

Conclusions

We have demonstrated a unique method for real-time object recognition and image segmentation based on cascades of discriminative MRFs. This method is scalable with respect to parallelism, memory usage, and number of recognized classes, which makes it especially suitable for practical applications. Experimental results have shown that the method has significant potential to boost the performance of existing purely local classifiers.

A limitation of our method is that it seems to be sensitive to random variation in the pre-segmentation output, which causes temporal noise in the classified output. We are currently investigating what effects this instability has on learning performance, and how this might be ameliorated by the use of segmentation routines with alternative objectives.

We have also not discussed how to generate the assignment of classes to each scope in the tree of classifiers. Currently, we have observed that a random assignment produces adequate results. However, it is possible that better results could be achieved by averaging over different random trees, in a manner similar to random forests, or by building a single tree with an information-like criterion.

Acknowledgements

Most of this work was performed while the primary author was at Willow Garage, Inc. We would also like to thank Ben Taskar, Kurt Konolige, and Gary Bradski for their support.

References

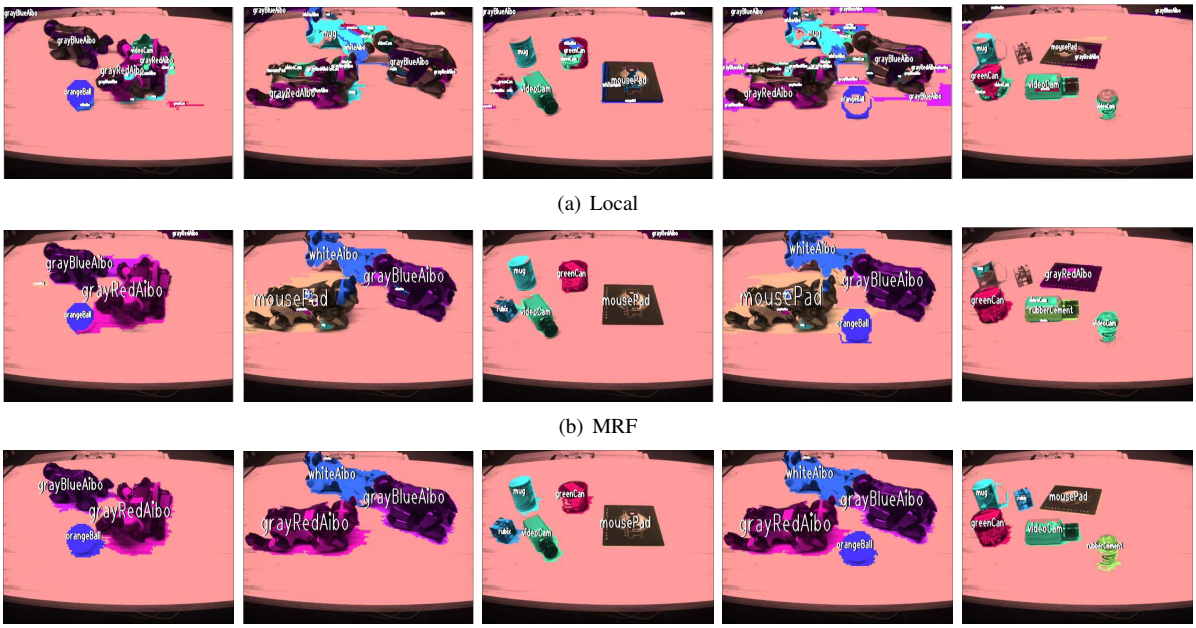
[1] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, 2004.

[2] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[3] B. Taskar, V. Chatalbashev, and D. Koller, "Learning associative Markov networks," in *Proceedings of the International Conference on Machine Learning*, 2004.

[4] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[5] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision*, 2008.



(a) Local

(b) MRF

(c) Ground truth

Fig. 5. A comparison of typical labelings generated by the purely local classifier and the cascaded MRF classifier.

	background	mousePad	rubberCement	videoCam	mug	rubix	whiteAibo	orangeBall	grayBlueAibo	grayRedAibo	greenCan
PURELY LOCAL CLASSIFIER											
background	**936798**	904	752	3239	2197	196	584	281	7756	3180	2239
mousePad	3598	**5008**	0	999	110	660	1663	256	694	1521	751
rubberCement	221	0	**0**	240	0	0	0	0	17	0	0
videoCam	3214	595	919	**8442**	171	1	348	0	1214	608	819
mug	3974	337	214	581	**9182**	0	136	374	577	4	511
rubix	2084	152	7	96	10	**2838**	0	668	442	341	410
whiteAibo	642	0	0	189	1569	19	**1002**	0	38	29	52
orangeBall	896	0	0	0	0	0	41	**1360**	0	0	0
grayBlueAibo	5587	28	0	151	22	182	203	111	**2894**	1323	0
grayRedAibo	4014	588	0	963	331	0	112	605	997	**4891**	190
greenCan	2589	0	675	1889	327	143	1124	65	552	366	**6063**
CASCADED MRF CLASSIFIER											
background	**947528**	2068	54	1846	929	220	320	418	2270	1182	1134
mousePad	848	**8849**	0	0	0	0	0	0	0	5525	38
rubberCement	156	0	**34**	213	0	0	0	0	0	0	75
videoCam	2520	0	361	**12834**	0	27	0	0	0	355	234
mug	3139	50	2	804	**11765**	0	0	3	0	0	99
rubix	2223	0	0	48	133	**3907**	0	0	0	339	398
whiteAibo	436	120	0	0	0	0	**2782**	0	130	72	0
orangeBall	231	24	0	0	0	0	0	**1927**	0	115	0
grayBlueAibo	1051	61	0	0	0	0	31	0	**9358**	0	0
grayRedAibo	925	3865	0	0	66	0	283	3	17	**7532**	0
greenCan	1828	0	695	0	588	0	364	12	0	0	**10306**

TABLE I

CLASS CONFUSION MATRICES ACCUMULATED OVER 5-FOLD CROSS-VALIDATION

[6] A. Torralba, K. P. Murphy, and W. T. Freeman, "Contextual models for object detection using boosted random fields," in *Neural Information Processing Systems*, 2004.

[7] S. Kumar and M. Hebert, "Discriminative fields for modeling spatial dependencies in natural images," in *Neural Information Processing Systems*, 2003.

[8] P. Vernaza, B. Taskar, and D. D. Lee, "Online, self-supervised terrain classification via discriminatively trained submodular Markov random fields," in *IEEE International Conference on Robotics and Automation*, May 2008.

[9] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *CVPR*, 2008.

[10] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *Computer Vision and Pattern Recognition*, 2004.

[11] R. Isukapalli, A. Elgammal, and R. Greiner, "Learning to detect objects of many classes using binary classifiers," in *European Conference on Computer Vision*, 2006.

[12] V. Kolmogorov, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Februrary 2004.

[13] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 2004.

[14] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2003.

[15] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, p. 2001, 2001.