Contextual Object Localization With Multiple Kernel Nearest Neighbor

Brian McFee, *Student Member, IEEE*, Carolina Galleguillos, *Student Member, IEEE*, and Gert Lanckriet, *Member, IEEE*

***Abstract*—Recently, many object localization models have shown that incorporating contextual cues can greatly improve accuracy over using appearance features alone. Therefore, many of these models have explored different types of contextual sources, but only considering one level of contextual interaction at the time. Thus, what context could truly contribute to object localization, through integrating cues from all levels, simultaneously, remains an open question. Moreover, the relative importance of the different contextual levels and appearance features across different object classes remains to be explored. Here we introduce a novel framework for multiple class object localization that incorporates different levels of contextual interactions. We study contextual interactions at the pixel, region and object level based upon three different sources of context: semantic, boundary support, and contextual neighborhoods. Our framework learns a single similarity metric from multiple kernels, combining pixel and region interactions with appearance features, and then applies a conditional random field to incorporate object level interactions. To effectively integrate different types of feature descriptions, we extend the large margin nearest neighbor to a novel algorithm that supports multiple kernels. We perform experiments on three challenging image databases: Graz-02, MSRC and PASCAL VOC 2007. Experimental results show that our model outperforms current state-of-the-art contextual frameworks and reveals individual contributions for each contextual interaction level as well as appearance features, indicating their relative importance for object localization.**

***Index Terms*—Contextual features, multiple kernel learning, multiple object localization, object detection.**

I. Introduction

LOCALIZING and identifying an object in an image is a challenging task in the presence of occlusions, poor quality, noise or background clutter. Therefore, to improve recognition accuracy, many models for object recognition have supplemented appearance features (derived from the object being recognized), with contextual information (derived from surrounding regions or objects). Recent work in computer vision has shown that the inclusion of contextual information can improve recognition of objects in real world images as it captures knowledge about the identity, location and scale of objects. Various types of contextual cues have been exploited to benefit object recognition tasks, including semantic [1]–[3], spatial [2], [4]–[12], scale [6], [9], [13], [14] and geographic [1] information. All of these models incorporate contextual information at either a global or a local image level.

Global context considers statistics from the image as a whole, i.e., the entire scene. This may include information about scene identity, provided by image and scene classification techniques before performing a more detailed analysis of the individual objects in the image. Several models [1], [12], [14] have successfully exploited global context to achieve better object detection results compared to using only objects' appearance information.

Local context considers information from neighboring areas of the object, such as information from neighboring pixels, regions, and objects. Since local context information is extracted after decomposing a scene into many parts, its advantage over global context is that it incorporates more precise, specific information from spatially localized image processing. Moreover, even if large parts of the scene are occluded, which could significantly affect global context features, it is still possible to extract relatively accurate local context features and carry out partial, localized background matching. Therefore, in this work, we will focus on local context, which many approaches for object segmentation and localization [2], [4]–[9], [11], [15], [16] have effectively incorporated into their recognition systems, greatly improving their accuracy.

When considering local context, contextual interactions can be grouped in three different types: pixel, region and object interactions. *Pixel interactions* capture low-level feature interactions between spatially adjacent objects. *Region interactions* capture higher-level information from the region surrounding an object. Finally, *object interactions* capture high-level information from other objects in the scene, which may be separated by large distances. Some of the previously mentioned approaches for object localization use local context from pixels [11], [15], [16], while others incorporate local context from regions [4]–[8], [17], or objects [2], [9]. Although most of these models achieve good results, and some successfully combine different sources of local context at a single level, they do not combine information from different levels of local context, or make explicit the contributions to localization accuracy due to

Manuscript received April 05, 2010; revised July 27, 2010; accepted August 03, 2010. Date of publication August 23, 2010; date of current version January 14, 2011. The work of B. McFee and G. Lanckriet was supported by the National Science Foundation under Grant DMS-MSPA 0625409. The work of C. Galleguillos was supported by DARPA under Grant NBCH1080007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mark (Hong-Yuan) Liao.

B. McFee and C. Galleguillos are with the Department of Computer Science and Engineering, University of California, San Diego, CA 92093 USA (e-mail: bmcfee@cs.ucsd.edu; cgallegu@cs.ucsd.edu).

G. Lanckriet is with the Department of Electrical and Computer Engineering, University of California, San Diego, CA USA 92093 (e-mail: gert@ece.ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2010.2068556

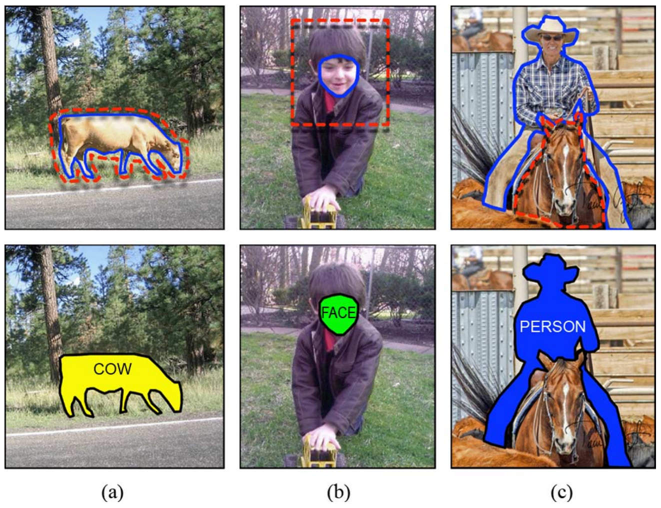


Fig. 1. Examples of contextual local interactions. (a) Pixel interactions capture information such as grass and tree pixels around the cow's boundary. (b) Region interactions are represented by relations between the face and the upper region of the body. (c) Object relationships capture interactions between the objects person and horse.

each individual level. Fig. 1 shows examples of different contextual interaction levels.

Previous work on image and scene classification has shown that by providing a more complete representation of the scene, the combination of multiple contextual interaction levels can improve image classification accuracy [18], [19]. The demonstrated benefits for image classification motivate our work to unify contextual interactions and appearance information for the problem of object localization. Previous methods for combining different interaction levels for image classification are relatively complex and computationally expensive. As a result, the relative contributions due to each contextual level may be obscured by the overall complexity of the system.

To provide an intuitive and interpretable method for integrating multiple contextual interaction levels, we turn to *multiple kernel learning* (MKL). Multiple kernel learning methods [20] have been succesfully applied in image classification [21], [22] and object localization tasks to optimally combine different types of appearance features [23], [24] and pixel interactions [4]. These methods generally entail finding a weighted combination of the given base kernels, where kernel weights are applied uniformly across all points. The weighted combined kernel is then used to produce classifiers, in either a hierarchical or one-versus-all framework. Although learning separate kernel combinations for each binary classification problem has been shown to perform extremely well on these tasks [22]–[24], it poses a great difficulty in scaling to large datasets, and the predictions from each classifier must be combined to yield a single prediction. By contrast, learning a single metric would enable the use of nearest neighbor classification, which naturally supports multiclass problems.

In this work, we present a novel framework for object localization that efficiently and effectively combines different levels of local contextual interactions. We develop a multiple kernel learning algorithm (MKLMNN) to integrate appearance features with pixel and region interaction data, resulting in a unified similarity metric which is optimized for nearest neighbor classification. Our method differs from previous MKL methods found in the computer vision literature, in that it can learn kernel weights which vary across the training set. Thus, a kernel may receive large weight only for the regions (or training points) for which it is informative. The kernel methods used here capture pixel- and region-level interactions, but to model object-level interactions, we apply a conditional random field (CRF), which produces the final label prediction for a test point. By using the algorithmic tools developed here, we are able to study the relative contribution of local contextual interactions for single- and multiobject localization over different data sets and object classes. To stimulate further research, source code will be made publicly available as part of this paper, which originally appeared as [25].

The paper is organized as follows. In Section II, we describe in detail our multiclass multikernel approach, including a detailed derivation of our MKLMNN algorithm. Section III introduces the different levels of contextual interactions studied in this work. We evaluate our algorithm for classification and localization tasks, and compare with related work in Section IV. Finally, conclusions are presented in Section V.

II. Multiclass Multikernel Approach

In our model, each training image $\mathcal{I}$ is partitioned into segments s_i by using ground truth information. Each segment s_i corresponds to exactly one object of class $c_i \in \mathcal{C}$, where $\mathcal{C}$ is the set of all object labels. These segments are collected for all training images into the training set S.

For each segment $s_i \in S$, we extract several types of features, e.g., texture or color. Due to the specific nature of the features used here, including appearance features and context features from pixel interactions and region interactions (see Sections III and IV), we do not expect linear models to adequately capture the important relationships between data points. We, therefore, represent each segment with a set of feature maps $\{\phi^z(s_i)\}$, where the $p\mathrm{th}$ feature space is characterized by a kernel function h^z and kernel matrix K^z, specifying the inner product—or, more intuitively, the similarity—between each pair of data points s_i and s_j:

$$h^z(s_i, s_j) = \langle \phi^z(s_i), \phi^z(s_j) \rangle, \qquad K^z_{ij} = h^z(s_i, s_j). \quad (1)$$

As in support vector machines [26], the kernel formulation allows us to capture nonlinear relations specific to each view of the data. However, each kernel matrix encodes a different feature space, and it is not immediately obvious how to optimally combine them to form a single space. In this section, we develop an algorithm to learn a unified similarity metric over the data, and a corresponding embedding function $g : S \longrightarrow \mathbb{R}^d$. This embedding function is used to map the training set S into the learned space, where it is then used to predict labels for unseen data with a k-nearest neighbor (kNN) classifier.

Because at test time, ground-truth segmentations are not available, the test image must be segmented automatically. To provide more representative examples for nearest neighbor prediction, we augment the training set S, of ground-truth segments, with automatically obtained segments S_A. These

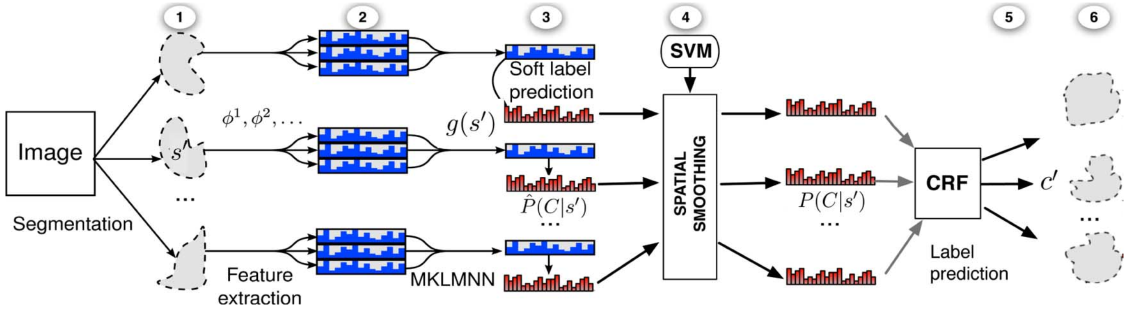


Fig. 2. Our object localization framework. (1) A test image is partitioned into segments s', and (2) several different features $\phi^1, \phi^2, \ldots$ (blue) are extracted for each segment. (3) Segments are mapped into a unified space by the optimized embedding $g(\cdot)$, and a soft label prediction $\hat{P}(C|s')$ (red) is computed using kNN. (4) Label predictions are spatially smoothed using a pairwise SVM, resulting in a new soft prediction $P(C|s')$. (5) CRF estimates the final label for each segment s' in the test image, and (6) segments are combined into an object c' if they overlap and receive the same final label.

additional segments, S_A, are obtained by running the segmentation algorithm [27] on the training images. This algorithm runs multiple times on each image, where each run provides a different number of image segments. Only those segments that are completely contained within or overlap more than 50% with the ground-truth object annotations are considered. These extra segments are then mapped into the learned space by applying $g(\cdot)$, and are also used to make label predictions on unseen data.

To counteract erroneous over-segmentation of objects in test images, we train an SVM classifier over pairs of the extra examples S_A to predict whether two segments belong to the same object. This is then used to spatially smooth the label predictions in test images.

To incorporate context from object interactions within an image, we train a CRF by using co-occurrence of objects within training images.

At test time, object localization for test images proceeds in six steps, depicted in Fig. 2. Specifically:

1) a test image $\mathcal{I}$ is partitioned into stable segments S';
2) for each $s' \in S'$, we apply the learned embedding function $s' \mapsto g(s')$ (Section II-B);
3) the k-nearest neighbors $\mathcal{N} \subset S \cup S_A$ of $g(s')$ are used to estimate a distribution over labels for the test segment $\hat{P}(C|s')$;
4) using the pairwise SVM, the label distribution of s' may be spatially smoothed by incorporating information from other segments in the test image, resulting in a new label distribution $P(C|s')$ (Section II-D);
5) the CRF uses object co-occurrence over the entire image to predict the final labeling of each segment $s' \in S'$ in the test image $\mathcal{I}$ (Section II-E);
6) finally, to produce object localizations from segment-level predictions, we consider segments to belong to the same object if they overlap at least 90% and receive the same final label prediction.

Table I gives a brief summary of the notation used throughout this article.

A. Large Margin Nearest Neighbor Using Kernels

Our classification algorithm is based upon k-nearest neighbor prediction, which naturally handles the multiclass setting. Because raw features (in the original feature space) may not adequately predict labels, we apply the large margin nearest neighbor (LMNN) algorithm to optimally transform the features for nearest neighbor prediction [28].

TABLE I
NOTATION USED THROUGHOUT THIS ARTICLE

Symbol	Definition
$\mathcal{I}$	Image
$S = \{s_1, s_2, \ldots\}$	Training segments (ground truth segmentation)
S_A	Additional segments for kNN (automatic segmentation)
$S' = \{s', \ldots\}$	Segments of a test image (automatic segmentation)
$\mathcal{C}$	Set of class (object) labels
$g(\cdot)$	Learned embedding function
$\phi^z(\cdot)$	Feature map for the zth kernel
$W \succeq 0$	Positive semi-definite matrix
$\|\|x - y\|\|_W$	Mahalanobis distance defined by W
$\mathcal{N}_i$	Nearest neighbors of s_i (in feature space)

1) LMNN: At a high level, LMNN simply learns a linear projection matrix L to transform the data such that the resulting representation is optimized for nearest-neighbor accuracy. If we imagine segments s_i, s_j, and s_ℓ as being represented by vectors in $\mathbb{R}^D$, then the goal is to learn a matrix $L \in \mathbb{R}^{d \times D}$ such that

$$\begin{aligned} &\|Ls_i - Ls_j\| \leq \|Ls_i - Ls_\ell\| \\ \Leftrightarrow\ &\|Ls_i - Ls_\ell\| - \|Ls_i - Ls_j\| \geq 0 \end{aligned} \tag{2}$$

when s_i and s_j belong to the same class, and s_ℓ belongs to a different class. Computationally, it is more convenient to operate on squared Euclidean distances, which can be expressed as follows:

$$\|L(s_i - s_j)\|^2 = (s_i - s_j)^\top L^\top L(s_i - s_j).$$

Note that distance calculations involve quadratic functions of the optimization variables (L), and distance constraints described by (2) require differences of quadratic terms. Therefore, formulating the optimization problem directly in terms of L would lead to a nonconvex problem with many local optima [29]. However, solving for the positive semidefinite (PSD) matrix $W \doteq L^\top L$ gives rise to distance constraints that are linear and, thus, convex in the optimization variables (W).

Neighbors are then selected by using the learned Mahalanobis distance metric W

$$d(s_i,s_j) = \|s_i - s_j\|_W^2 = (s_i - s_j)^{\mathsf{T}} W (s_i - s_j). \quad (3)$$

Formulating the problem in terms of W introduces the constraint $W \succeq 0$, leading to a semidefinite programming problem [29], which is shown in Algorithm 1 [28]. In Algorithm 1, $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ contain the neighbors of segment s_i in the original feature space with similar or dissimilar labels respectively. For each s_i, rather than simply forcing neighboring segments s_ℓ with dissimilar labels to be further away than those with similar labels (s_j), as expressed by (2), the constraints in Algorithm 1 enforce unit margins between the distances to ensure stability of the learned metric. As in support vector machines, slack variables $\xi_{ij\ell}$ allow constraint violations with a hinge-loss penalty.

Algorithm 1 LMNN [28]

```
$\min_{W,\xi} \sum_i \sum_{j \in \mathcal{N}_i^+} \|s_i - s_j\|_W^2 + \beta \sum_{ij\ell} \xi_{ij\ell}$
$\forall i,\ \forall j \in \mathcal{N}_i^+,\ \forall \ell \in \mathcal{N}_i^-$ :
$\|s_i - s_\ell\|_W^2 - \|s_i - s_j\|_W^2 \geq 1 - \xi_{ij\ell}$
$W \succeq 0,\ \xi_{ij\ell} \geq 0$
```

The first term in the objective function minimizes the distance from each s_i to its similarly labeled neighbors s_j. The second term, weighted by a slack tradeoff parameter, $\beta \geq 0$, penalizes violations of the margin constraints. W is a PSD matrix which characterizes the optimal feature transformation.

Once W has been learned, a linear projection matrix L can be recovered by spectral decomposition, so that $W = L^{\mathsf{T}}L$

$$W = V^{\mathsf{T}}\Lambda V = V^{\mathsf{T}}\Lambda^{1/2}\Lambda^{1/2}V \quad \Rightarrow \quad L \doteq \Lambda^{1/2}V. \quad (4)$$

Here, V contains the eigenvectors of W, and Λ is a diagonal matrix containing the eigenvalues.

2) Kernel LMNN (KLMNN): Algorithm 1 assumes that each segment is represented by a vector in $\mathbb{R}^D$, and is limited to linear transformations of these vector representations. To learn nonlinear transformations, the algorithm can be kernelized [26], [30] as follows.

First, a feature map ϕ, possibly nonlinear, is applied to a segment s_i. This can be viewed as projecting the segment into a (high- or potentially infinite-dimensional) feature space. Then, as in Algorithm 1, we learn an optimal linear projection L from that feature space to a low-dimensional Euclidean space in which distances are optimized for nearest neighbor prediction

$$\|L\phi(s_i) - L\phi(s_j)\|^2 + 1 \leq \|L\phi(s_i) - L\phi(s_\ell)\|^2. \quad (5)$$

This projection L, combined with the mapping ϕ, allows to learn nonlinear transformations of the segment representation s_i. Formulating an optimization problem in terms of L in the high-dimensional space could lead to over-fitting. If we introduce a regularization term $\|L\|_F^2 = \mathrm{tr}(L^{\mathsf{T}}L)$ in the objective function to limit the complexity of the learned L, we may then apply the *representer theorem* [31], [32]. It follows that, at the optimum, L takes the form

$$L = \hat{L}\Phi^{\mathsf{T}} \quad (6)$$

where Φ is a matrix where the $i\mathrm{th}$ column is $\phi(s_i)$. Intuitively, this expresses that the rows of any optimal L must lie in the span of the training data in the feature space.

This fact can be exploited to rewrite distance calculations in terms of $\hat{L}$ and $K = \Phi^{\mathsf{T}}\Phi$, the kernel matrix corresponding to the feature map ϕ

$$\begin{aligned} d(s_i,s_j) &= \|L(\phi(s_i) - \phi(s_j))\|^2 \\ &= (\phi(s_i) - \phi(s_j))^{\mathsf{T}} L^{\mathsf{T}} L(\phi(s_i) - \phi(s_j)) \\ &= (\phi(s_i) - \phi(s_j))^{\mathsf{T}} \left(\hat{L}\Phi^{\mathsf{T}}\right)^{\mathsf{T}} \left(\hat{L}\Phi^{\mathsf{T}}\right) \\ &\quad \times (\phi(s_i) - \phi(s_j)) \\ &= (\phi(s_i) - \phi(s_j))^{\mathsf{T}} \Phi\hat{L}^{\mathsf{T}}\hat{L}\Phi^{\mathsf{T}} \\ &\quad \times (\phi(s_i) - \phi(s_j)) \\ &= (K_i - K_j)^{\mathsf{T}} \hat{L}^{\mathsf{T}}\hat{L}(K_i - K_j) \end{aligned} \quad (7)$$

where $K_i = \Phi^{\mathsf{T}}\phi(s_i)$ is s_i's column in K. Similarly, we can rewrite the regularization term

$$\begin{aligned} \mathrm{tr}\left(L^{\mathsf{T}}L\right) &= \mathrm{tr}\left(\left(\hat{L}\Phi^{\mathsf{T}}\right)^{\mathsf{T}}\left(\hat{L}\Phi^{\mathsf{T}}\right)\right) \\ &= \mathrm{tr}\left(\Phi\hat{L}^{\mathsf{T}}\hat{L}\Phi^{\mathsf{T}}\right) \\ &= \mathrm{tr}\left(\hat{L}^{\mathsf{T}}\hat{L}\Phi^{\mathsf{T}}\Phi\right) \\ &= \mathrm{tr}\left(\hat{L}^{\mathsf{T}}\hat{L}K\right) \end{aligned} \quad (8)$$

which allows to formulate the problem entirely in terms of $\hat{L}$ and K without explicit reference to the feature map ϕ. Defining $\hat{W} = \hat{L}^{\mathsf{T}}\hat{L} \succeq 0$, we can substitute $\hat{W}$ into (7) and (8), and solve the KLMNN problem in terms of $\hat{W}$. The KLMNN is listed as Algorithm 2.

Algorithm 2 KLMNN

```
$\min_{W,\xi} \sum_i \sum_{j \in \mathcal{N}_i^+} \|K_i - K_j\|_W^2 + \beta \sum_{ij\ell} \xi_{ij\ell} + \gamma \cdot \mathrm{tr}(WK)$
$\forall i, \forall j \in \mathcal{N}_i^+$,
$\forall \ell \in \mathcal{N}_i^-$ :    $\|K_i - K_\ell\|_W^2 - \|K_i - K_j\|_W^2 \geq 1 - \xi_{ij\ell}$
$\xi_{ij\ell} \geq 0,\quad W \succeq 0$
```

In summary, compared to Algorithm 1, we represent each segment s_i by its corresponding column in the kernel matrix $(s_i \mapsto K_i)$—essentially using similarity to the training set as features—and introduce a regularization term $\gamma \cdot \mathrm{tr}(WK)$, balanced by the parameter $\gamma > 0$ to the objective function. The embedding function then takes the form

$$g(s_i) \doteq LK_i \quad (9)$$

where L is recovered from W by spectral decomposition (4).

This embedding function generalizes to an unseen segment s' by first applying the kernel function

$$h(s', s_i) = \langle \phi(s'), \phi(s_i) \rangle$$

at s' and each s_i in the training set, and then applying the linear transformation L to the vector $(h(s', s_i))_{i=1}^{n}$, where $(\cdot)_{i=1}^{n}$ denotes vertical concatenation.

B. Multiple Kernel LMNN

To effectively integrate different types of feature descriptions, e.g., appearance features and context from pixel and local interactions—we extend the LMNN algorithm to a novel algorithm that supports multiple kernels ($K^1, K^2, \ldots, K^m$ with feature maps $\phi^1, \phi^2, \ldots, \phi^m$).

Previous work approaches multiple kernel learning by finding a weighted combination of kernels $K^* = \sum_z a_z K^z$, where $a_z \geq 0$ is the learned weight for K^z [20]. While this approach has worked for support vector machines, adapting it directly to work with (K)LMNN, i.e., calculating distances by

$$\left(\sum_{z=1}^{m} a_z K_i^z - a_z K_j^z\right)^{\mathsf{T}} W \left(\sum_{z=1}^{m} a_z K_i^z - a_z K_j^z\right) \tag{10}$$

would lead to a nonconvex optimization problem with many local optima.

Instead, we take a different approach, and following [33], we learn a set of linear projections $L^1, L^2, \ldots, L^m$, each corresponding to a kernel's feature space. In this view, the linear projection L^z is tuned specifically to the geometry of the space defined by the feature map ϕ^z. By representing the embedding of a point as the concatenation of projections from each feature space, we obtain the multiple-kernel embedding function

$$g(s_i) = (L^z \phi^z(s_i))_{z=1}^{m}. \tag{11}$$

By linearity, the inner product between the embeddings of two points s_i, s_j can be expressed as

$$\begin{aligned} \langle g(s_i), g(s_j) \rangle &= \sum_{z=1}^{m} \langle L^z \phi^z(s_i), L^z \phi^z(s_j) \rangle \\ &= \sum_{z=1}^{m} \phi^z(s_i)^{\mathsf{T}} L^{z\mathsf{T}} L^z \phi^z(s_j). \end{aligned} \tag{12}$$

Accordingly, distances between embedded points take the form

$$\begin{aligned} d(s_i, s_j) &= \|g(s_i) - g(s_j)\|^2 \\ &= \sum_{z=1}^{m} (\phi^z(s_i) - \phi^z(s_j))^{\mathsf{T}} \\ &\quad \times L^{z\mathsf{T}} L^z (\phi^z(s_i) - \phi^z(s_j)). \end{aligned} \tag{13}$$

Following the argument of the previous section, we introduce a regularization term for each kernel: $\operatorname{tr}(L^{z\mathsf{T}} L^z)$. Now, by independently applying the representer theorem to each L^z, it follows that the optimum lies in the span of the training data (within the zth feature space):

$$L^z = \hat{L}^z \Phi^{z\mathsf{T}}. \tag{14}$$

Finally, by plugging (14) into (13) and following the logic of (7), it follows that distances between embedded points can be decomposed to the sum:

$$d(s_i, s_j) = \sum_{z=1}^{m} \left(K_i^z - K_j^z\right)^{\mathsf{T}} \hat{L}^{z\mathsf{T}} \hat{L}^z \left(K_i^z - K_j^z\right). \tag{15}$$

Similarly, the regularization terms can be collected and expressed as $\sum_{z=1}^{m} \operatorname{tr}(\hat{L}^{z\mathsf{T}} \hat{L}^z K^z)$ (see (8)). As in KLMNN (Algorithm 2), the projection matrices appear only in the form of inner products $\hat{L}^{z\mathsf{T}} \hat{L}^z$, so we can equivalently express the constraints in terms of $W^z = \hat{L}^{z\mathsf{T}} \hat{L}^z$

$$\begin{aligned} d(s_i, s_j) &= \sum_{z=1}^{m} (K_i^z - K_j^z)^{\mathsf{T}} W^z (K_i^z - K_j^z) \\ &= \sum_{z=1}^{m} \|K_i^z - K_j^z\|_{W^z}^2 \end{aligned} \tag{16}$$

and, similarly, the regularization term as $\sum_{z=1}^{m} \operatorname{tr}(W^z K^z)$. This allows to carry out the optimization in terms of the kernel-specific metrics W^z.

We refer to the algorithm that emerges from this formulation as MKLMNN, and the optimization is listed as Algorithm 3. Like Algorithm 2, the optimization problem is still a semidefinite program (and, hence, convex), but now there are m PSD matrices to learn. The optimization is solved by gradient descent on W^z, where each W^z is projected onto the set of PSD matrices after each gradient step (see Appendix A).

Algorithm 3 MKLMNN

```
$\min_{W^z, \xi} \sum_i \sum_{j \in \mathcal{N}_i^+} d(s_i, s_j) + \beta \sum_{ij\ell} \xi_{ij\ell} + \gamma \sum_{z=1}^{m} \operatorname{tr}(W^z K^z)$
$\forall i, \forall j \in \mathcal{N}_i^+,$
$\forall \ell \in \mathcal{N}_i^- :$    $d(s_i, s_\ell) - d(s_i, s_j) \geq 1 - \xi_{ij\ell}$
$d(s_i, s_j) \doteq \sum_{z=1}^{m} \|K_i^z - K_j^z\|_{W^z}^2$
$\xi_{ij\ell} \geq 0, \forall z = 1 \ldots m : W^z \succeq 0$
```

Fig. 3 illustrates the differences between LMNN, KLMMN, and our framework for MKLMNN. The formulation of multiple kernel learning via concatenated projections of feature spaces results in a more flexible model than previous methods, and allows the algorithm to automatically adapt to the case where the discriminative power of a kernel varies over the data set.

Although the optimization problem is convex and can be solved in polynomial time, maintaining the constraints $W^z \succeq 0$ requires a spectral decomposition and projection onto the cone of positive semidefinite matrices after each gradient step. To simplify the process, we add a constraint which restricts W^z to be diagonal. This added constraint reduces the semidefinite program (Algorithm 3) to a (more efficient) linear program, and the diagonals of W^z can be interpreted as weightings of S in each feature space. Moreover, diagonally constraining W^z

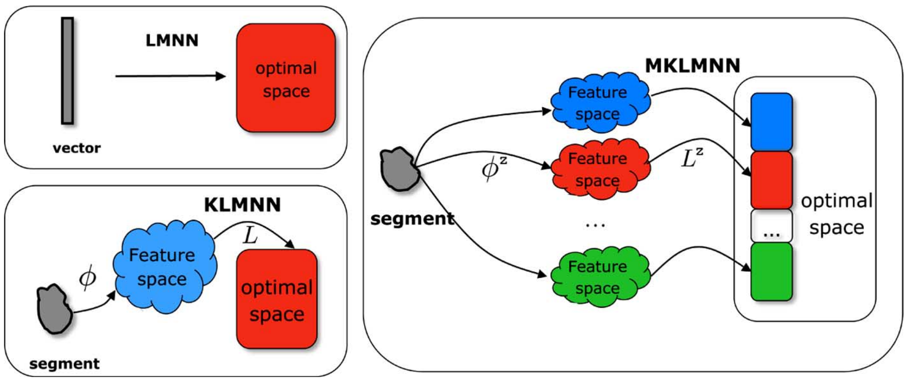


Fig. 3. Diagrams depicting the differences between LMNN, KLMMN, and our framework for MKLMNN.

can be interpreted as a sparse approximation to the full set of PSD matrices, and is equivalent to optimizing over the set

$$\left\{ \sum_{i=1}^{n} W^z_{ii}\phi^z(s_i)\phi^z(s_i)^\mathsf{T} \quad | \quad W^z_{ii} \geq 0 \right\}.$$

In this view, each dimension of the learned embedding $g(\cdot)$ is computed by a single kernel evaluation and scaling by the corresponding learned weight. For diagonal matrices, enforcing positive semidefiniteness can be accomplished by thresholding: $W^z_{ii} \mapsto \max(0, W^z_{ii})$. This operation is much more computationally efficient than the PSD projection for full W^z matrices, and the diagonal formulation still yields good results in practice.

As is usually the case for kernel-based learning algorithms, the complexity of Algorithm 3 (i.e., the dimensionality of W^z) scales with the number of training points. To cope with high-dimensionality, one route is to compress each kernel matrix K, either by row-sampling or principal components analysis. Our formulation remains convex under both of these modifications, which are equivalent to learning a projection LVK (as opposed to LK in (9)), where V is a d-by-n sampling or PCA matrix. This would effectively reduce the number of parameters to learn and the dimensionality of the learned space, leading to a more efficient optimization.

C. Soft Label Prediction

After mapping a test segment s' into the learned space, a probability distribution over the labels is computed by using its k nearest neighbors $\mathcal{N} \subseteq S \cup S_A$, weighted according to distance from $g(s')$

$$\hat{P}(C = c|s') = \frac{\sum_{j \in \mathcal{N}, c_j = c} \exp\left(-d(s', s_j)\right)}{\sum_{c'} \sum_{j \in N, c_j = c'} \exp\left(-d(s', s_j)\right)} \tag{17}$$

where c_j is the label of segment s_j.

D. Spatial Smoothing by Segment Merging

Due to the automatic segmentation, objects may be represented by multiple segments at test time, where each segment might contain only partial information from the object, resulting in less reliable label information $\hat{P}(C|s')$. To counteract this effect, we smooth a segment's label distribution $\hat{P}(C|s')$ by incorporating information from segments which are likely to come from the same object, resulting in an updated label distribution $P(C|s')$.

Using the extra segments S_A automatically extracted from the training images, we train an SVM classifier on pairs of segments to predict whether two segments belong to the same object. Based upon the ground truth object annotations for the training set, we know when to label a pair of training segments as coming from the same object. A training set is constructed as follows. Going through all training images, all segment pairs that come from the same (ground truth) object are collected in a set of positive training examples. An equal number of negative training examples is obtained by randomly selecting pairs of segments coming from a different object, in each of the training images. Based upon this training data set of segment pairs, taken from all training images, one SVM is trained.

The SVM is trained on features extracted from pairs of segments, i.e., given two segments s_i and s_j we compute:

Feature	Description
ϕ_i^{PI}, ϕ_j^{PI}	Pixel interaction features for segments i and j (Section III),
ϕ_i^{RI}, ϕ_j^{RI}	Region interaction features for segments i and j (Section III),
O_{ij}, O_{ji}	Fraction of segment i that overlaps with j and vice versa, where $0 \leq O \leq 1$,
μ_i, μ_j	Normalized centroid coordinates for segments i and j,
q_i, q_j	Total number of segments generated in the segmentation from which i, respectively j was obtained, $2 \leq q \leq 10$ (the segmentation algorithm [27] that generates S_A partitions each image multiple times, resulting in segmentations with $q = 2, 3, \ldots, 10$ segments),
$\|\|\mu_i - \mu_j\|\|_2$	Distance between centroids μ_i and μ_j.

Note that soft label predictions are not included as features, so the SVM provides an independent assessment to smooth the label distributions. At test time, we construct an undirected graph where each vertex is a segment s' of the test image, and edges are added between pairs that the classifier predicts to come from the same object. For each connected component of the graph, we merge the segments corresponding to its vertices, resulting in a new object segment s_o. We then extract features for the merged object segment s_o, apply the embedding function $g(s_o)$, and obtain a label distribution $\hat{P}(C|s_o)$ by (17). The smoothed label distribution for a segment s' is then obtained as the geometric mean of the segment's distribution and its corresponding object's distribution

$$P(C=c|s') = \frac{\sqrt{\hat{P}(C=c|s')\cdot \hat{P}(C=c|s_o)}}{\sum_{c'} \sqrt{\hat{P}(C=c'|s')\cdot \hat{P}(C=c'|s_o)}}. \quad (18)$$

Note that distributions remain unchanged for any segments s' which are not merged (i.e., when $s_o = s'$).

E. Contextual CRF

Unlike pixel and region interactions, which can be described by lower-level features, object interactions require a high-level description of the segment, e.g., its label, or a distribution over possible labels. Because this information is not available until after soft label predictions are known, object interactions cannot be encoded in a base kernel. Therefore, information derived from high-level object interactions is incorporated by introducing a CRF after the soft label predictions $P(C|s')$ have been computed. CRFs are better suited for incorporating contextual cues than other types of graphical models [34]. First, object co-occurrences encode undirected information. This suggests undirected graphical models, like CRFs or Markov random fields (MRFs). Second, by modeling the conditional distribution, CRFs can directly incorporate contextual relationships, as soft constraints between random variables (as opposed to MRFs, which model the joint distribution). This approach has been previously demonstrated to be effective for object localization [2], [3].

Given soft label predictions for all segments $\{s_i\}_{i=1}^{|\mathcal{I}|}$ in an image $\mathcal{I}$, the CRF models the distribution of final label assignments $\vec{c} = (c_1 \ldots c_{|\mathcal{I}|})$ for all segments as follows:

$$P\left(\vec{C}=\vec{c}\,|\,\mathcal{I}\right) = \frac{1}{Z}\Psi(\vec{c})\cdot\prod_{i=1}^{|\mathcal{I}|} P(C_i=c_i|s_i) \quad (19)$$

where $\vec{C} = (C_1 \ldots C_{|\mathcal{I}|})$, Z is the partition function and Ψ is given by

$$\Psi(\vec{c}) = \exp\left(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}\psi(c_i,c_j)\right). \quad (20)$$

The potential function ψ captures long-distance dependencies between objects in images, and is learned from object co-occurrences in training images through maximum likelihood estimation. As it is intractable to maximize the co-occurrence likelihood directly, we approximate the partition function using Monte Carlo integration [35], and apply gradient descent to find $\psi(\cdot)$ that approximately optimizes the data likelihood.

After obtaining soft label predictions for all segments in a test image, the final label vector is determined by maximizing (19) over all possible label assignments. The maximization can be carried out efficiently by using importance sampling, where each segment is a node in the CRF.

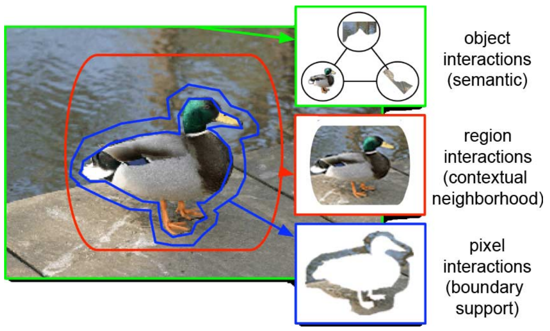


Fig. 4. Local contextual interactions in our model. Pixel interactions are captured by the surrounding area of the bird. Region interactions are captured by expanding the window to include surrounding objects, such as water and road. Object interactions are captured by the co-occurrence of other objects in the scene.

III. Contextual Interactions

In this section, we describe the features we use to characterize each level of contextual interaction.

A. Pixel-Level Interactions

By capturing low-level feature interactions between an object and surrounding pixels, pixel-level interactions implicitly incorporate background contextual information, as well as information about object boundaries. To model pixel-level interactions, we propose a new type of contextual source, which we call *boundary support*. Boundary support computes the surrounding statistics of an object within an image by considering individual pixel values of a surrounding region of the object. This is shown in Fig. 4.

In our model, boundary support is encoded by computing a histogram over the L*A*B* color values in the region immediately surrounding an object's boundary. We compute the χ^2-distance between boundary support histograms H

$$\chi^2(H,H') = \sum_i \frac{(H_i - H_i')^2}{H_i + H_i'} \quad (21)$$

and define the pixel interaction kernel as

$$h^{PI}(s_i,s_j;\sigma) = \exp\left(-\sigma\chi^2(H_i,H_j)\right) \quad (22)$$

where $\sigma > 0$ is a bandwidth parameter.

B. Region-Level Interactions

Region-level interactions have been extensively investigated in the area of context-based object localization. By using large windows around an object, known as contextual neighborhoods [15], regions encode probable geometrical configurations, and capture information from neighboring (parts of) objects (as shown in Fig. 4). Our contextual neighborhood is computed by dilating the bounding box around the object using a disk of diameter d

$$d = \max\left(\sqrt{\frac{I_w}{B_w}}, \sqrt{\frac{I_h}{B_h}}\right) \quad (23)$$

where I_w, I_h, B_w, and B_h are the widths and heights of the image and bounding box respectively. We model region interactions by computing the gist [14] of a contextual neighborhood, G_i. Hence, our region interactions are represented by the χ^2-kernel

$$h^{RI}(s_i, s_j; \sigma) = \exp\left(-\sigma\chi^2(G_i, G_j)\right). \tag{24}$$

C. Object-Level Interactions

To train the object interaction CRF, we derive *semantic* context from the co-occurrence of objects within each training image by constructing a co-ocurrence matrix A. An entry $A(i,j)$ counts the times an object with label c_i appears in a training image that contains an object with label c_j. Diagonal entries correspond to the frequency of the object in the training set. Next, the between-class potential $\psi(c_i, c_j)$ is learned by approximately optimizing the data likelihood, using gradient descent, as explained in Section II-E.

IV. Experiments

To evaluate the localization accuracy of the proposed system and study the relative importance of each contextual interaction level, we perform experiments on the Graz-02 [36], MSRC [37] and PASCAL 2007 [38] databases.

Four different appearance features were computed: SIFT [39], Self-similarity (SSIM) [40], L*A*B* histogram and pyramid of histogram of oriented gradients (PHOG) [41]. SIFT descriptors were computed at random locations and quantized in a vocabulary of 5000 words. SSIM descriptors were computed at the same locations as SIFT, and also quantized in a vocabulary of 5000 words. PHOG descriptors were computed as in Bosch *et al.* [41], but we consider only a 360° orientation (608-dimensional descriptor). L*A*B* histograms were computed and concatenated into a 48-dimensional histogram. Finally, each type of feature is represented by a separate χ^2-kernel.

As explained in Section III, region- and pixel-interaction kernels are computed using GIST (1008-dimensional descriptor) and L*A*B* color (48-dimensional histogram) features, respectively. Boundary support is computed between 0 and 20 pixels away from a segment's boundary.

A. Analyzing MKLMNN for Single-Object Localization

In order to analyze the contribution of the MKLMNN component of our framework, we perform experiments on Graz-02, a single-object detection database. Graz-02 presents one of three object classes—*bikes*, *cars*, and *people*—in each image (usually with only one object instance per image), extreme variability in pose, scale and lighting. Following the experimental setup of [36], the ground truth object segments of the first 150 odd-numbered images of each class are used for training. The first 150 even-numbered images of each class are added to the test set. Since, at test time, some segments will represent background and no object, the discriminative power of MKLMNN is ensured by augmenting the training set with the class *background*. More specifically, 150 background segments are obtained from a random sample of the training images, confined to regions where no object is present.

As there is only one class present in each image, there are no object co-ocurrences from which to learn object interactions, and we, therefore, omit the CRF step in this experiment. For similar reasons, no SVM smoothing is being performed, making the MKLMNN algorithm the focus of this evaluation. Test set performance is measured by segment classification and single-object localization accuracy.

1) Segment Classification: After labeling each segment s' in a test image with the most probable class label from $\hat{P}(C|s')$, the classification accuracy is evaluated by considering s' as correctly classified if it overlaps more than 90% with the ground truth object while predicting the correct label. Table II(a) reports classification results achieved for each object class by combining appearance, pixel and region interactions. For comparison purposes, we also list accuracy achieved by an unweighted kernel combination. We define the *average kernel function* $\bar{h}$ as the unweighted sum of all base kernel functions, from which we construct the *average kernel matrix*

$$\bar{h}(s_i, s_j) = \sum_{z=1}^{m} h^z(s_i, s_j) = \bar{K}_{ij} = \sum_{z=1}^{m} K_{ij}^z. \tag{25}$$

Results show that for each object class, MKLMNN achieves significantly higher accuracy than the unweighted average kernel. While classification accuracy is high for all object classes, we observe slightly lower performance for the object class *people*. This class presents greater variability in scale than other classes, resulting in more erroneous over-segmentations at test time. For example, heads tend to be segmented as part of the background. Table II(b) shows the mean classification accuracy achieved by MKLMNN with different combinations of base kernels. Results show that combining appearance with only one level of context ($\mathrm{App} + \mathrm{PI}$ or $\mathrm{App} + \mathrm{RI}$) outperforms using context ($\mathrm{PI} + \mathrm{RI}$) or appearance alone (App). Furthermore, combining appearance features with both types of local contextual features results in the best performance ($\mathrm{App} + \mathrm{PI} + \mathrm{RI}$).

Fig. 5 visualizes the learned space when optimally combining appearance, pixel and region interactions. Note that images are surrounded by neighbors that depict the same object from a similar viewpoint.

Learning diagonally constrained rather than full W^z, as described in Section II-B, affects the embedding and, thus, the classification accuracy. To quantify the effect on accuracy of this simplifying assumption, we perform a small experiment to compare full and diagonally constrained W^z when learned with the appearance kernels (SIFT, SSIM, and PHOG). Classification accuracy is shown in Table III.

When constraining W^z to be a diagonal matrix, the optimization problem becomes linear and we, therefore, gain substantial efficiency in computation and obtain a sparse solution. However, we also lose a small percentage of classification accuracy when comparing with the results obtained with the full matrix. In this work, we choose to trade accuracy for efficiency, so we constrain W^z to be diagonal in all subsequent experiments.

2) Object Detection: Localization accuracy is obtained by first merging segments in test images that overlap by at

TABLE II
SEGMENT CLASSIFICATION RESULTS FOR GRAZ-02. APPEARANCE (APP), PIXEL (PI) AND REGION (RI) INTERACTIONS ARE COMBINED FOR SEGMENT CLASSIFICATION. (A) CLASSIFICATION ACCURACY PER CLASS FOR THE UNWEIGHTED SUM OF KERNELS (AVERAGE KERNEL) VERSUS LEARNING THE OPTIMAL EMBEDDING BY COMBINING ALL KERNELS $(\mathrm{App}+\mathrm{PI}+\mathrm{RI})$ WITH MKLMNN. (B) AVERAGE CLASSIFICATION ACCURACY FOR DIFFERENT KERNEL COMBINATIONS WITH MKLMNN

Classification Accuracy	Average Kernel	MKLMNN
Bikes	0.52	0.98
Cars	0.74	0.99
People	0.73	0.96
Mean	0.66	**0.98**

(a)

Mean Classification Accuracy	MKLMNN
PI+RI	0.76
App	0.92
App+PI	0.93
App+RI	0.96
App+PI+RI	**0.98**

(b)

TABLE IV
LOCALIZATION RESULTS FOR GRAZ-02. (A) APPEARANCE (APP), PIXEL (PI) AND REGION (RI) INTERACTIONS ARE COMBINED FOR OBJECT LOCALIZATION. (B) LOCALIZATION ACCURACY IMPROVES SIGNIFICANTLY WHEN LEARNING THE OPTIMAL EMBEDDING WITH MKLMNN. THE BEST ACCURACY USING ONLY ONE KERNEL IS OBTAINED USING REGION INTERACTIONS (GIST) FOR GRAZ-02

Localization Accuracy	Bikes	Cars	People	Mean
PI+RI	0.56	0.78	0.42	0.59
App	0.72	0.81	0.50	0.68
App+PI	0.73	0.81	0.51	0.68
App+RI	0.72	0.82	0.54	0.69
App+PI+RI	0.74	0.82	0.56	**0.71**

(a)

Localization Accuracy	Bikes	Cars	People	Mean
MKLMNN (App+PI+RI)	**0.74**	**0.82**	**0.56**	**0.71**
KLMNN on average kernel	0.71	0.78	0.53	0.67
Average kernel (native)	0.57	0.63	0.46	0.56
Best kernel (RI)	0.65	0.82	0.40	0.58

(b)

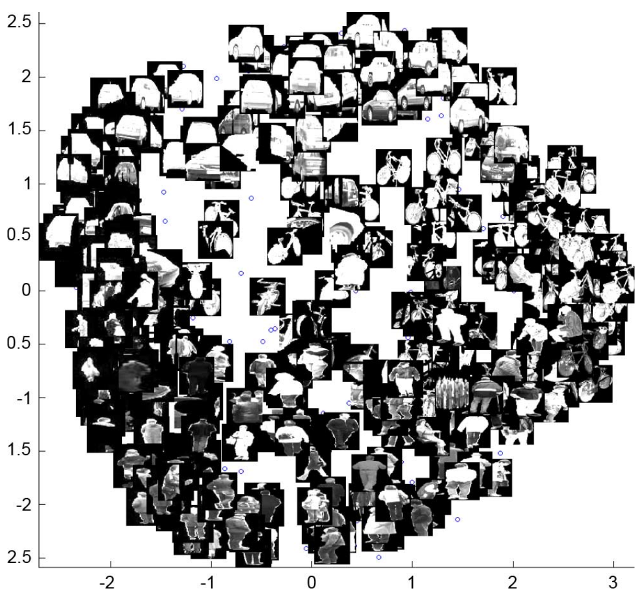


Fig. 5. 2-D projection of the optimal embedding for the Graz-02 training set. We excluded background segments and subsample segments from object categories in order to have a better view of them.

TABLE III
COMPARISON IN CLASSIFICATION ACCURACY FOR LEARNING FULL, RESPECTIVELY DIAGONAL W^z

Kernels	**Classification Accuracy Full** W^z	**Classification Accuracy Diagonal** W^z
PHOG	0.673	0.641
SIFT+SSIM+PHOG	0.853	0.840

least 90% and receive the same final label prediction, and then following the well-known evaluation procedure of [38] on the merged segments. This procedure accounts for label accuracy and overlap with the ground truth object for the (merged) segments in test images. Table IV(a) shows localization accuracy results for each object class. Combining all local contextual interactions with appearance features results in the best localization accuracy. Although localization and classification accuracy cannot be compared directly, the relatively lower localization accuracy can be understood as follows: even though some segments are correctly classified, the resulting (merged) segment fails to overlap significantly with the ground truth bounding box.

For all combinations of features, we achieve better localization accuracy for the classes *bikes* and *cars* than for the class *people*, for reasons discussed earlier. Due to the presence of cluttered backgrounds, boundary support conveys little useful information in this database, as can be seen by comparing the results for App and $\mathrm{App}+\mathrm{PI}$. The MKLMNN optimization detects this phenomenon at training time, and correctly down-weights pixel interactions where they are noninformative. Fig. 10 shows examples of the localization of objects in test images.

Since the Graz-02 data set has traditionally been used for other computer vision tasks, no other object localization results are currently available. Comparisons of our system to state-of-the-art algorithms will be provided for the multiobject localization task in Section IV-B.

3) Feature Combination: To gain a better understanding of how the MKLMNN algorithm contributes to localization performance, we repeat the localization experiment with different methods of kernel combination. Table IV(b) compares the localization accuracy obtained by MKLMNN to that obtained by using the average kernel, as well as the space obtained by optimizing the average kernel with KLMNN (Algorithm 2), and the single best kernel (in this case, RI). MKLMNN achieves significant improvements in accuracy over the unweighted kernel combination, which performs worse than using the single best kernel. We analyze the relative importance of each kernel in forming the optimal embedding by examining the learned weights W^z_{ii}. We observe that the solution is sparse, since some examples are more discriminative than others for nearest neighbor classification. Fig. 6(a) illustrates the sparsity of the solution, and shows the kernel weights for each point in the training set. Previous MKL methods generally learn a set of kernel weights that are applied uniformly across all points. MKLMNN, on the other hand, learns weights that vary across

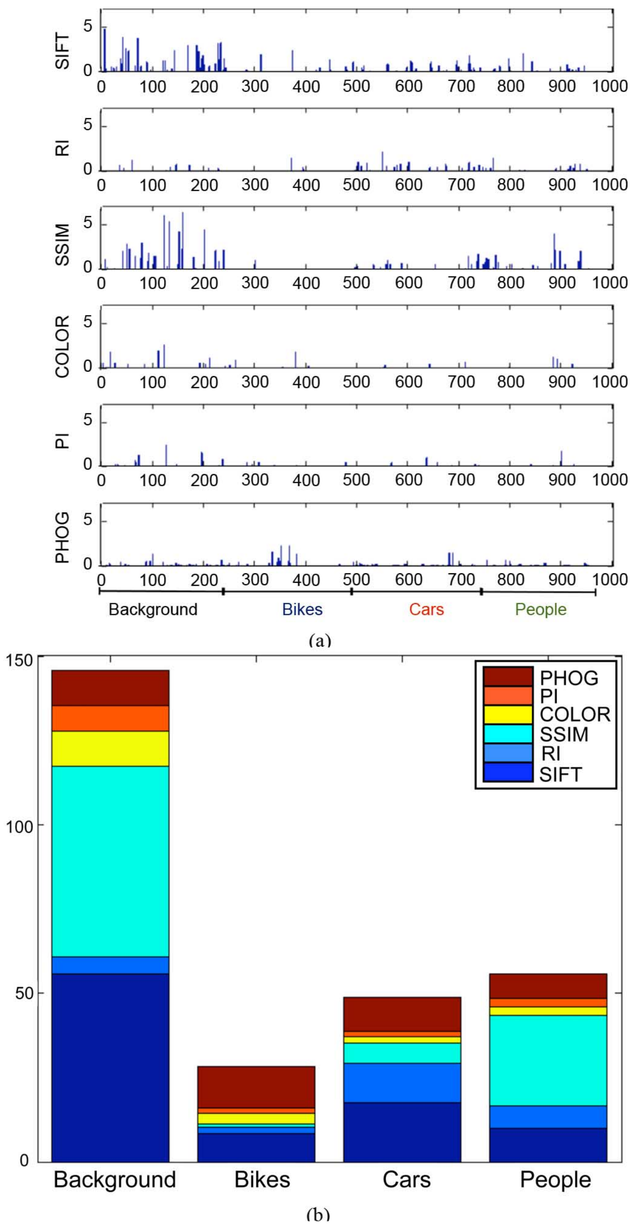


Fig. 6. Learned kernel weights for Graz-02. (a) Kernel weights for each point in the training set, per kernel. (b) Kernel weights grouped by class.

points, so that a kernel may be used only where it is informative; this is demonstrated by the fact that different training points receive weight in the different kernel spaces. Fig. 6(b) shows the learned weights grouped by class. Segments corresponding to background examples receive greater weights for the appearance features SIFT and SSIM than segments corresponding to actual objects. This can be explained by the great dissimilarity between examples in the background class.

Inspecting the kernel weights for each of the object classes in more detail, we observe that appearance kernels are generally important, while region interactions mostly matter to discriminate the object classes *cars* and *people*, capturing typical geometric configurations between the background and these objects. Pixel interactions and color kernels receive low weights across all object classes. The latter can be explained by the high variability in color appearance for the objects in this database, while the former is due to the high levels of clutter, which generally results in a nonuniform background making boundary support relatively uninformative.

4) Implementation Details: For Graz-02, we use the data split of [36] for training and testing. We compute multiple stable segmentations, consisting of respectively 2, 3,…, 9, and 10 segments per image. Together, this results in 54 segments per image. MKLMNN is trained using the 250-nearest neighbors, and the parameters β and γ are found using cross-validation. For χ^2-kernels, the bandwidth is fixed a priori at $\sigma = 3$. For the experiments comparing diagonally constrained and full W^z, the same values of the hyperparameters are used.

B. Multiple Object Localization

To evaluate our framework for multiobject localization, we use the MSRC [37] and PASCAL 2007 [38] databases. These databases present 21 and 20 different object classes, respectively, with images that contain several object instances from multiple classes, as well as occlusions, extreme variability in pose, scale and lighting.

1) Object Detection: Localization accuracy is computed, again, by following the evaluation procedure of [38]. Table V (top) shows the mean accuracy results for MSRC with different combinations of appearance (App) and contextual interactions—pixel interaction (PI), region interaction (RI), and object interaction (OI). We observe that using only appearance information (App) results in a mean localization accuracy of 50%, while including local contextual interactions $(\mathrm{App} + \mathrm{PI} + \mathrm{RI} + \mathrm{OI})$ improves accuracy to 70%. Combining all local context features $(\mathrm{PI} + \mathrm{RI} + \mathrm{OI})$ performs similarly to using appearance only, suggesting that object classes could potentially be learned from cues that don't include appearance information [42]. If only pixel or region interactions are combined with appearance features $(\mathrm{App} + \mathrm{PI}$ or $\mathrm{App} + \mathrm{RI})$, accuracy already improves over using appearance alone, where adding RI realizes a larger improvement than adding PI.

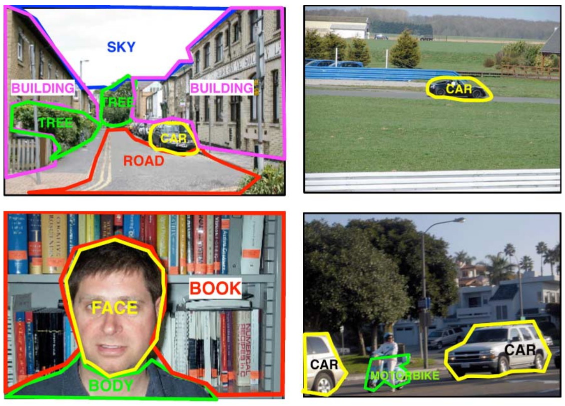
Note that the object interaction model depends directly upon the estimated labels $P(C|s')$, so a more accurate estimate of $P(C|s')$ allows the CRF to contribute better to the final localization accuracy. The segment-merging SVM predicts same-object segment pairs correctly 81% of the time, and contributes constructively to the localization accuracy without making a significant difference: omitting this step only reduces localization accuracy by approximately 1%.

We repeat the experiment on PASCAL07, and, again, evaluate the localization accuracy and the contribution of the different contextual interactions. Table V (bottom) shows the results for combining appearance with different levels of local context. As for MSRC, combining appearance with all contextual interactions $(\mathrm{App} + \mathrm{PI} + \mathrm{RI} + \mathrm{OI})$ improves the mean accuracy dramatically: in this case, from 26% (for appearance only) to 39%. Pixel interactions account for the largest individual gain, improving accuracy from 26% (App) to 33% $(\mathrm{App} + \mathrm{PI})$. For PASCAL, we observe that the segment merging step correctly predicts same-object segment pairs 85% of the time, and contributes constructively without making a significant difference.

TABLE V
MEAN LOCALIZATION ACCURACY FOR THE MSRC AND PASCAL07 DATA SETS. APPEARANCE (APP), PI, (RI), AND (OI) ARE COMBINED FOR OBJECT LOCALIZATION

	Features	Mean Localization Accuracy	Features	Mean Localization Accuracy
MSRC	PI+RI	0.42	App+ RI	0.61
	PI+RI+OI	0.49	App+ OI	0.52
	App	0.50	App+ PI + RI	0.66
	App+ PI	0.54	App + PI + RI + OI	**0.70**

	Features	Mean Localization Accuracy	Features	Mean Localization Accuracy
PASCAL 2007	PI+RI	0.23	App+ RI	0.29
	PI+RI+OI	0.24	App+ OI	0.27
	App	0.26	App+ PI + RI	0.37
	App+ PI	0.33	App + PI + RI + OI	**0.39**



(a) (b)

Fig. 7. (a) Examples from MSRC (left column). (b) Examples from PASCAL 07 (right column). The background in most MSRC images is segmented and labeled with one or more specific object classes, like, e.g., *sky*, *road*, and *building*. In PASCAL images, the background lacks such structure, and is generally unlabeled. Background structure allows region interactions to incorporate more consistent information from neighboring (parts of) objects in MSRC, compared to PASCAL. Moreover, this increases the number of object classes which co-occur in an MSRC image, enabling object interactions to make a greater contribution to localization than in PASCAL.

As in the MSRC experiment, omitting the segment merging step reduces localization accuracy by approximately 1%.

Comparing both data sets, we notice that the different contextual interaction levels contribute differently to localization in the different data sets. For example, for PASCAL, adding object interactions ($\mathrm{App} + \mathrm{PI} + \mathrm{RI} + \mathrm{OI}$ versus $\mathrm{App} + \mathrm{PI} + \mathrm{RI}$) improves localization accuracy by only 2%, compared to the 4% improvement for MSRC. This is not surprising, since MSRC presents more co-occurrences of object classes per image than PASCAL, which provides more information to the object interaction model. Region interactions also contribute more in MSRC where the background tends to exhibit more structure, due to the presence of specific background classes in the scene, i.e., *sky*, *grass*, *water*, *road*, and *building*. Fig. 7 illustrates these differences.

2) Feature Combination: Table VI shows that for both MSRC and PASCAL, learning the optimal embedding with MKLMNN again results in substantial improvements over the average kernel (native or optimized), and the single best kernel. For MSRC, we achieve 66% localization accuracy with MKLMNN, compared to 54% when optimizing the average kernel with KLMNN. Similarly, in PASCAL we observe 37% with MKLMNN, compared to 25% for the average kernel.

TABLE VI
BOTH FOR MSRC AND PASCAL07, LOCALIZATION ACCURACY IMPROVES SIGNIFICANTLY AFTER LEARNING THE OPTIMAL EMBEDDING. THE BEST ACCURACY USING ONLY ONE KERNEL IS OBTAINED USING SIFT FOR MSRC AND RI (GIST) FOR PASCAL

Mean Localization Accuracy	MSRC	PASCAL 07
MKLMNN (App+PI+RI)	**0.66**	**0.37**
KLMNN on average kernel	0.54	0.25
Average kernel (native)	0.51	0.25
Best kernel (SIFT/RI)	0.36	0.20

To analyze the relative importance of each kernel in forming the optimal embedding, we examine the learned W^z matrices. As with the Graz-02 data set, the solution is sparse, which, again, can be explained by some examples being more discriminative than others for kNN classification. Figs. 8(a) and 9(a) depict the sum of the weights assigned to each kernel for MSRC, respectively PASCAL07. We observe that SIFT and PHOG are the most important kernels for both data sets, and that color-based kernels receive relatively more weight in MSRC than in PASCAL. The latter is explained by the presence of background classes in MSRC such as *water*, *sky*, *grass* and *tree* which tend to be more homogeneous in color and, therefore, can be more efficiently described using a color kernel. PASCAL, on the other hand, lacks these homogeneous background classes, and, instead, contains more "man-made objects" where color features exhibit higher variance and less discriminatory power.

Figs. 8(b) and 9(b) illustrate the learned weights for each kernel, grouped by class. This demonstrates the flexibility of our multiple kernel formulation. Kernel weights automatically adapt to the regions in which they are most discriminative, as evidenced by the nonuniformity of each kernel's weight distribution. Contrast this with the more standard kernel combination approach, which would assign a single weight to each kernel for the entire data set, potentially losing locality effects which are crucial for nearest neighbor performance.

This allows us to examine which features are active for each class. For example, as shown in Fig. 8(b) for MSRC, color kernels are selected for points in the classes *building*, *cat*, *face*, *grass*, *road*, *sky* and *tree*. With respect to contextual kernels, *body*, *face*, and *water* give importance to pixel interactions, but not region interactions. In the particular case of the class *face*, this effect is explained by the fact that faces are often surrounded by (dark) hair.

Similarly, in PASCAL, classes such as *boat*, *bottle*, *chair*, and *motorbike* get weights for pixel interactions and not for region interactions [see Fig. 9(b)]. This is easily explained for *boats*,

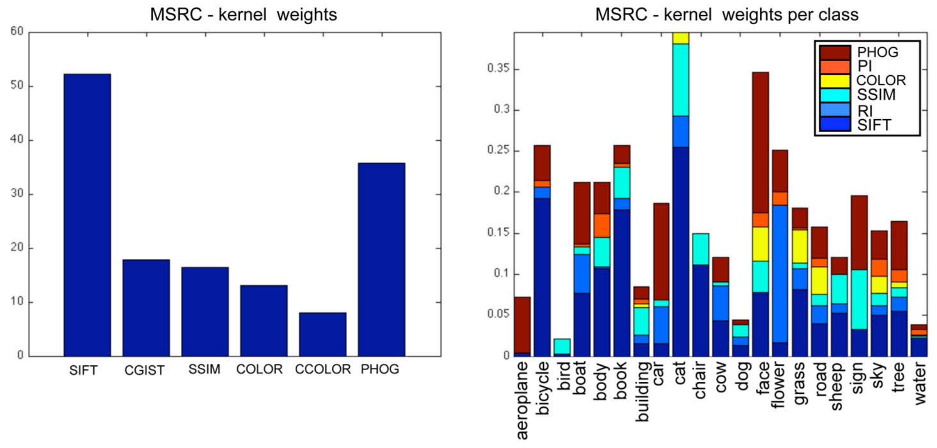


Fig. 8. Learned kernel weights for MSRC. Context gist (CGIST) corresponds to region interactions (RI) and context color (CCOLOR) corresponds to PIs. (a) For kernel K^z, its total weight is $\mathrm{tr}(W^z)$. (b) Weights grouped by class.

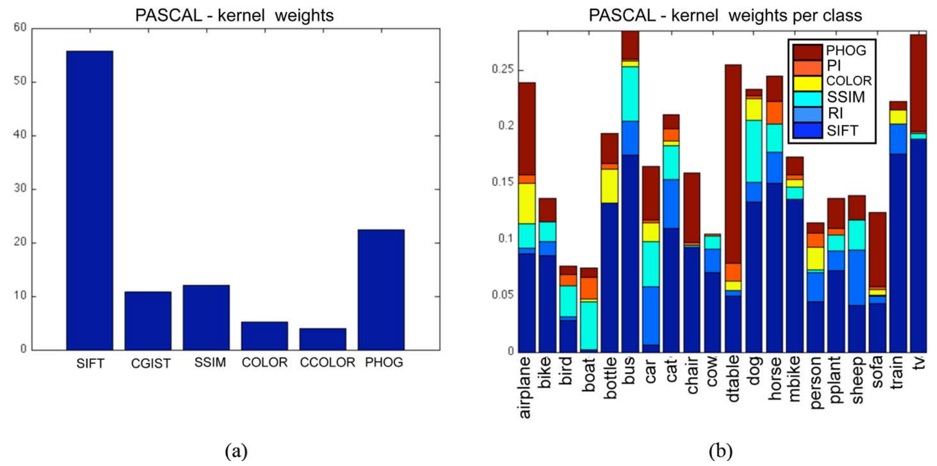


Fig. 9. Learned kernel weights for PASCAL. CGIST corresponds to RI and CCOLOR corresponds to PI. (a) For kernel K^z, its total weight is $\mathrm{tr}(W^z)$. (b) Weights grouped by class.

which are surrounded by water, for which color is highly informative. Region interactions get some weight for the classes *bike*, *bus*, *sheep*, and *train*, as objects in these classes are often found in the proximity of other specific objects. For example, *bike* objects are often overlapped by *person* objects. Fig. 11 shows examples of localization where the different context levels help to improve this task.

3) Comparison to Other Models: To compare our model to the current state-of-the-art, we compute the detection accuracy per class. Table VII shows the per-class accuracy for some of our models, corresponding to different combinations of kernels, and the contextual model from [2], which is the current state-of-the-art for object localization on MSRC. The MSRC data set has been studied as well for object segmentation, e.g., by models such as [11], [17]. Since this is an essentially different task, with different evaluation metrics, no comparison is made to these segmentation approaches. We outperform [2] for half of the classes, and obtain higher average accuracy overall, demonstrating the benefit of combining different contextual interaction levels.

For the PASCAL data set, we compare our model (All) to the current state-of-the-art algorithm for object localization on this data set [2], as well as the best performing system in the PASCAL09 challenge object detection [24] (for which the test set is not publicly available yet), and one other context-based approach [4]. Table VIII shows the per-class localization accuracy, where the bottom line provides the best localization result obtained for each object class in the PASCAL07 challenge [38]. We notice that our model performs best in the largest number of classes (tied with [2]), and we achieve a higher mean localization accuracy.

Our multiple kernel framework for learning a single metric over all classes outperforms models which learn class-specific kernel combinations [4], [24]. This owes to the fact that our embedding algorithm is geared directly toward multiclass prediction, and information can be shared between all classes by the joint optimization. Moreover, models in [4], [24] report only modest gains over the unweighted average of base kernels, while our model achieves significant improvement over both the average and best kernels. This suggests that convex combinations

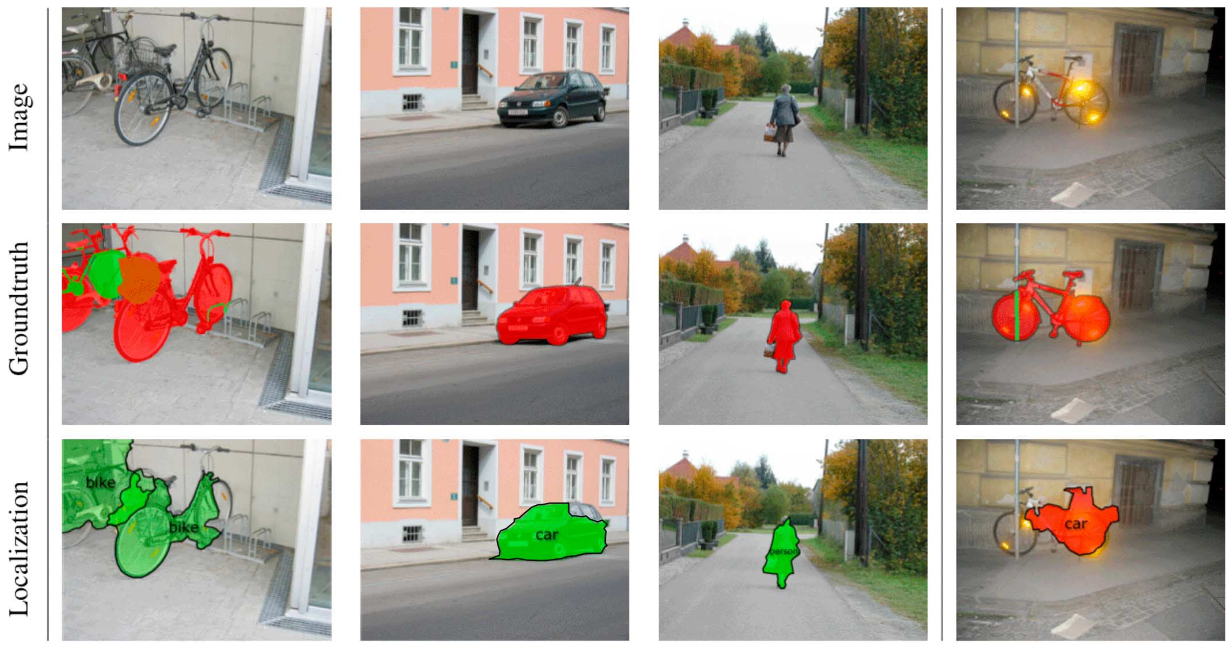


Fig. 10. Examples of images from the Graz-02 database. Images (first row), ground truth labels (second row) and detections (third row) are shown. For images showing ground truth labels (second row), red areas correspond to visible parts of the object and green indicates occluded parts. For detection results, green areas correspond to correct detections by our framework and red areas corresponds to false detections. (a) Examples of localization results for the category *bikes*. (b) Examples of localization results for the category *car*. (c) Examples of localization results for the category *people*. (d) Examples of false localizations for the classes *bikes* (top) and *people* (bottom).

TABLE VII

First Three Rows: Localization Accuracy for Our System Using Appearance Alone (A), Using Appearance Together With Pixel and Region Interactions (A + C), and Using Appearance With All Contextual Levels, i.e., Pixel, Region and Object Interactions (All). The Last Row Provides the Per-Class Localization Accuracy Obtained by the Contextual Model in [2], the Current State-of-the-Art for Object Localization on MSRC. Results in Bold Indicate the Best Performance Per Class. Our System Achieves the Best Average Accuracy

	aeroplane	bike	bird	boat	body	book	building	car	cat	chair	cow	dog	face	flower	grass	road	sheep	sign	sky	tree	water	mean
A	0.49	0.95	0.00	0.31	0.35	0.38	0.65	0.51	0.09	0.66	0.45	0.13	0.40	0.33	0.93	0.62	0.55	0.63	0.53	0.91	0.54	0.50
A+C	0.96	1.00	0.10	0.63	0.66	0.74	0.65	0.86	0.18	0.69	0.76	0.27	0.60	0.72	0.94	0.71	0.95	0.70	0.47	0.70	0.50	0.66
All	**1.00**	**0.98**	0.11	0.63	0.55	**0.78**	0.73	**0.88**	0.11	**0.80**	**0.74**	0.43	0.72	**0.72**	**0.96**	0.76	**0.90**	**0.92**	0.50	0.76	0.61	**0.70**
[2]	0.73	0.60	**0.52**	**0.81**	**0.77**	0.56	**0.91**	0.57	**0.42**	0.37	0.41	**0.46**	**0.81**	0.65	0.95	**0.96**	0.55	0.54	**0.97**	**0.80**	**0.95**	0.68

TABLE VIII

Comparison of Localization Accuracy for Different Systems on the PASCAL 07 Object Classes. Results in Bold Indicate the Best Performance Per Class. The Bottom Line Provides the Best Localization Result Obtained for Each Class in the PASCAL07 Challenge [38]. Our System (All) Achieves the Best Average Accuracy

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	dtable	dog	horse	mbike	person	pplant	sheep	sofa	train	tv	mean
All	0.33	0.24	**0.47**	**0.69**	0.22	0.37	**0.71**	0.33	0.07	0.15	**0.74**	0.21	0.26	**0.55**	0.33	**0.29**	0.38	0.23	**0.51**	0.57	**0.39**
[24]	0.38	**0.48**	0.15	0.15	0.22	**0.51**	0.51	0.30	0.17	**0.33**	0.23	0.22	**0.51**	0.46	0.23	0.12	0.24	0.29	0.45	0.49	0.32
[2]	**0.63**	0.22	0.14	0.42	**0.43**	0.50	0.62	0.32	**0.37**	0.19	0.30	0.29	0.15	0.31	**0.43**	0.33	**0.41**	**0.37**	0.29	**0.62**	0.37
[4]	0.11	0.12	0.09	0.06	0.00	0.25	0.14	**0.36**	0.09	0.14	0.24	**0.32**	0.27	0.34	0.03	0.02	0.09	0.30	0.30	0.08	0.17
[38]	0.26	0.41	0.10	0.09	0.21	0.39	0.43	0.24	0.13	0.14	0.10	0.16	0.34	0.38	0.22	0.12	0.18	0.15	0.33	0.29	-

of kernels may be too restrictive, while our approach of concatenated linear projections provides a greater degree of flexibility to the model.

4) Implementation Details: The data split from Galleguillos *et al.* [2] is used for MSRC evaluations. For PASCAL07, we follow [38] and train models based upon 30 images per object class. Multiple stable segmentations [27] are computed—9 different segmentations for each image—each of which contains between 2–10 segments. This results in 54 segments per image. The computation time for one segmentation is between 60–90 s, resulting in an average of 10 min of computation time to obtain all stable segmentations for one image. As the individual segmentations are independent of one another, they could also be computed in parallel, to improve computational efficiency. For the spatial smoothing step, one SVM is trained for each data set, using the SVM$^{\text{light}}$ implementation [43] with RBF kernels for the classification task. For the MSRC data set, 994 positive and an equal number of negative pairwise examples are used for

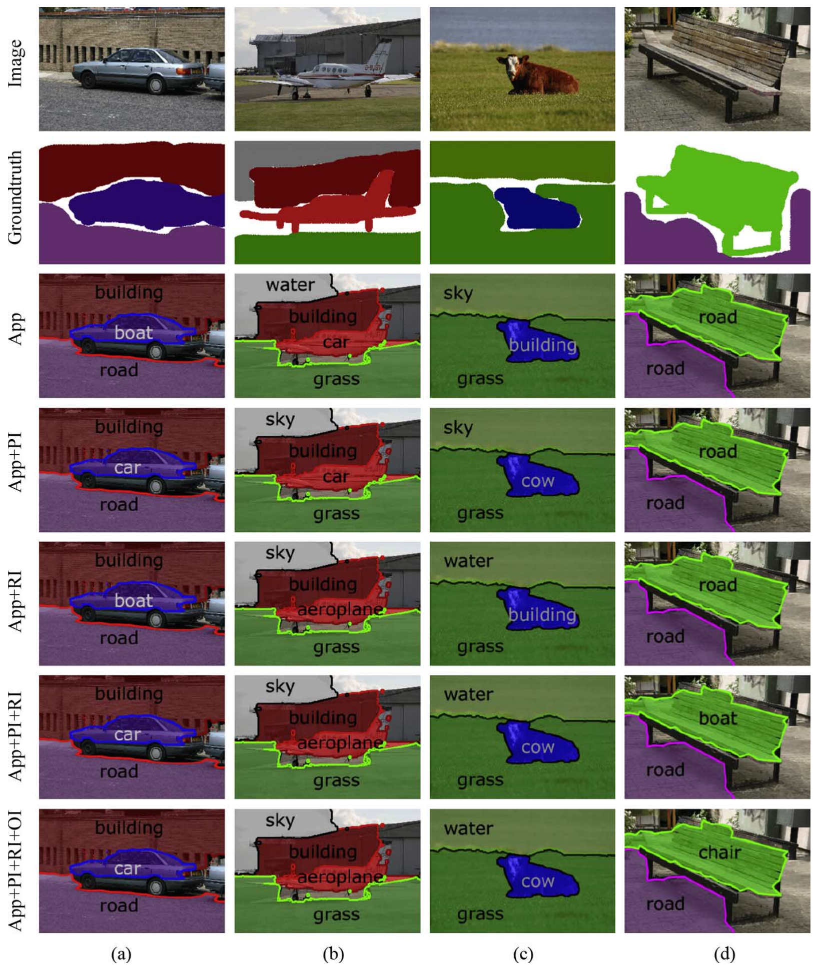


Fig. 11. Examples of images from the MSRC database. Each labeled colored region corresponds to an object localization result performed by our framework. (a) Localization example where pixel interactions improve localization using appearance. (b) Localization example where region interactions improve localization. (c) Localization example where pixel and region interactions together improve localization. (d) Localization example where object interactions improve localization over different feature combinations.

training. For PASCAL07, 350 positive and 350 negative examples are used. Each training example is described by a 2120-dimensional vector, as explained in Section II-D. Hyperparameters for each SVM are determined by 3-fold cross-validation on the training data set. We train MKLMNN with 15 nearest neighbors. For MSRC, the parameters β and γ are obtained with two-fold cross-validation on the training set. The results are stable for a variety of choices of k between 5–15; we select $k = 10$. For PASCAL07, the best β, γ and k are selected on the PASCAL07 validation set and then applied for testing on the test set. The parameter σ for the χ^2-kernels is set to 3. The complexity of MKLMNN scales with the number of training points and the number of neighbors to consider in the constraints. On an Intel 2.53 GHz Core Duo with 4 GB RAM, training time is 35 min with $\sim$ 900 training points (comparable to the number of training segments for MSRC, and for PASCAL07), 15 nearest neighbors, and a diagonal constraint on W. Predicting the soft labeling for all segments in a test image takes under a second (after segmentations have been computed). For the CRF, hyperparameters are determined by two-fold cross-validation on the training set. Training the CRF takes 3 min for MSRC (315 training images) and 5 min for PASCAL07 (600 training images). At test time, running the CRF to obtain the final labeling takes between 2–3 s, depending upon the number of segments.

V. Conclusion

In this work, we have developed a novel framework for optimally integrating multiple feature descriptors into a single, unified similarity space. By learning a single space which is optimized for nearest neighbor prediction, we are able to quantitatively compare the contributions to the learned space due to each base feature descriptor. In particular, we evaluated the importance of various contextual cues at the pixel, region, and object level for object localization tasks. Moreover, the proposed MKLMNN algorithm yields interpretable solutions, and achieves significant improvements over current state-of-the-art contextual frameworks.

Appendix
Gradient Descent Derivation

To solve the optimization problem listed as Algorithm 3, we implemented a gradient descent solver. We show here the derivation of the gradient. We first eliminate the slack variables by moving margin constraints into the objective

$$\min_{W^z \succeq 0} \quad f_1 + \beta \cdot f_2 + \gamma \cdot f_3$$

where

$$f_1 = \sum_i \sum_{i \in \mathcal{N}_i^+} d(s_i, s_j)$$

$$f_2 = \sum_{ij\ell} \eta \left(1 + d(s_i, s_j) - d(s_i, s_\ell)\right)$$

$$f_3 = \sum_{z=1}^{m} \operatorname{tr}(W^z K^z)$$

and

$$\eta(x) = \begin{cases} 0, & x < 0 \\ x, & \text{otherwise} \end{cases}$$

is the hinge-loss function. We can now derive the gradient of the objective with respect to W^z in three pieces, corresponding to the three terms f_1, f_2, f_3.

By the cyclic property of the trace, a distance $\|K_i^z - K_j^z\|_{W^z}^2$ can be expressed as a matrix inner product

$$\begin{aligned} \|K_i^z - K_j^z\|_{W^z}^2 &= (K_i^z - K_j^z)^\mathsf{T} W^z (K_i^z - K_j^z) \\ \|K_i^z - K_j^z\|_{W^z}^2 &= \operatorname{tr}(W^z (K_i^z - K_j^z)(K_i^z - K_j^z)^\mathsf{T}). \end{aligned}$$

It follows that the gradient for the first term is:

$$\frac{\partial f_1}{\partial W^z} = \sum_i \sum_{j \in \mathcal{N}_i^+} (K_i^z - K_j^z)(K_i^z - K_j^z)^\mathsf{T}.$$

Although η is nondifferentiable at 0, we can write down a subgradient for f_2 as follows:

$$\frac{\partial f_2}{\partial W^z} = \sum [d(s_i, s_\ell) - d(s_i, s_j) < 1] \\ \left((K_i^z - K_j^z)(K_i^z - K_j^z)^\mathsf{T} - (K_i^z - K_\ell^z)(K_i^z - K_\ell^z)^\mathsf{T}\right)$$

where $[x]$ is the indicator function of the event x.

Finally, the gradient for the regularization term is simply

$$\frac{\partial f_3}{\partial W^z} = K^z.$$

By linearity, the (sub-)gradient of the objective function is the sum of these three (sub-)gradients. After each gradient step, the updated W^z matrix is projected back onto the PSD cone by calculating its spectral decomposition, $W^z = V \Lambda V^\mathsf{T}$, and thresholding the eigenvalues: $W^z \mapsto V(\max(\Lambda, 0))V^\mathsf{T}$. When each W^z is restricted to be diagonal, the decomposition step is unnecessary since the diagonal elements contain the eigenvalues; diagonal PSD projection can, thus, be accomplished by $W^z \mapsto \max(W^z, 0)$.

Acknowledgment

The authors would like to thank S. Belongie for his insightful comments and support.

References

[1] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1271–1278.

[2] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2008.

[3] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Proc. Int. Conf. Comput. Vis.*, 2007.

[4] M. Blaschko and C. H. Lampert, "Object localization with global and local context kernels," in *Proc. BMVC*, 2009.

[5] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[6] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[7] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *Proc. Eur. Conf. Comput. Vis.*, 2008, vol. 1.

[8] J. J. Lim, P. Arbelaez, C. Gu, and J. Malik, "Context by region ancestry," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[9] D. Parikh, C. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2008.

[10] B. Russell, A. Torralba, C. Liu, R. Fergus, and W. Freeman, "Object recognition by scene alignment," in *Proc. NIPS*, 2007.

[11] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2008.

[12] J. Verbeek and B. Triggs, "Scene segmentation with crfs learned from partially labeled images," in *Proc. NIPS*, 2008, vol. 20.

[13] K. Murphy, A. Torralba, and W. Freeman, "Using the forest to see the trees: A graphical model relating features, objects and scenes," in *Proc. NIPS*, 2003, vol. 16.

[14] A. Torralba, "Contextual priming for object detection," *Int. J. Comput. Vis.*, 2003.

[15] M. Fink and P. Perona, "Mutual boosting for contextual inference," in *Proc. NIPS*, 2004.

[16] H. Kruppa and B. Schiele, "Using local context to improve face detection," in *Proc. BMVC*, 2003, pp. 3–12.

[17] L. Ladicky, C. Russell, P. Kohli, and P. Torr, "Associative hierarchical CRFs for object class image segmentation," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[18] X. He, R. Zemel, and M. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2004.

[19] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *Proc. Int. Conf. Comput. Vis.*, 2005.

[20] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[21] A. Kumar and C. Sminchisescu, "Support kernel machines for object recognition," in *Proc. Int. Conf. Comput. Vis.*, 2007.

[22] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *Proc. Int. Conf. Comput. Vis.*, 2007.

[23] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[24] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[25] C. Galleguillos, B. McFee, S. Belongie, and G. R. G. Lanckriet, "Multi-class object localization by combining local contextual interactions," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2010.

[26] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[27] A. Rabinovich, T. Lange, J. Buhmann, and S. Belongie, "Model order selection and cue combination for image segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2006.

[28] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. NIPS*, 2006, pp. 451–458.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[30] L. Torresani and K. Lee, "Large margin component analysis," in *Proc. NIPS*, 2007, vol. 19, pp. 1385–1385.

[31] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *J. Math. Anal. Appl.*, vol. 33, pp. 82–95, 1971.

[32] B. Scholkopf, R. Herbrich, A. J. Smola, and R. Williamson, "A generalized representer theorem," in *Proc. 14th Annu. Conf. Comput. Learn. Theory*, 2001, pp. 416–426.

[33] B. McFee and G. R. G. Lanckriet, "Partial order embedding with multiple kernels," in *Proc. ICML*, Jun. 2009.

[34] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer Vis. Image Understand.*, vol. 114, pp. 712–722, 2010.
[35] C. P. Robert and G. Casella*, Monte Carlo Statistical Methods*. New York: Springer-Verlag, 2005.
[36] M. Marszalek and C. Schmid, "Accurate object localization with shape masks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2007.
[37] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2006.
[38] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," in *Proc. Visual Recognition Challenge Workshop*, 2007.
[39] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[40] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2007.
[41] A. Bosch, A. Zisserman, and X. Munoz, "Scene classification via pLSA," in *Proc. Eur. Conf. Comput. Vis.*, 2006.
[42] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar, "Attribute and simile classifiers for face verification," in *Proc. Int. Conf. Comput. Vis.*, 2009.
[43] T. Joachims*, Making Large-Scale Support Vector Machine Learning Practical, Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999.

Brian McFee (S'10) received the B.S. degree in computer science from the University of California, Santa Cruz (UCSC) in 2003, the M.S. degree in computer science and engineering from the University of California, San Diego (UCSD) in 2008, and is currently pursuing the Ph.D. degree at UCSD.

He is a member of the Computer Audition Laboratory, UCSD. In 2010, he was a recipient of the Qualcomm Innovation Fellowship. His research interests include machine learning and music information retrieval.

Carolina Galleguillos (S'10) received the Engineering degree in computer science from the University of Chile, Santiago, Chile, in 2005, the M.S. degree in computer science and engineering from the University of California, San Diego (UCSD) in 2008, and is currently pursuing the Ph.D. degree in computer science at UCSD.

She is a member of the Computer Vision Lab at UCSD. In 2007 and 2008, she was awarded the National Science Foundation Integrative Graduate Education and Research Traineeship (NSF IGERT) fellowship. Her research interest include computer vision and machine learning.

Gert Lanckriet (M'10) received the M.S. degree in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 2000, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 2001 and 2005, respectively.

In 2005, he joined the Department of Electrical and Computer Engineering at the University of California, San Diego, where he heads the Computer Audition Laboratory. His research focuses on the interplay of convex optimization, machine learning, and signal processing, with applications in computer music and computer audition.

Dr. Lanckriet was awarded the SIAM Optimization Prize in 2008 and is the recipient of a Hellman Fellowship and an IBM Faculty Award.