

Semi-Automatically Labeling Objects in Images

Wen Wu, *Student Member, IEEE*, and Jie Yang, *Member, IEEE*

Abstract—Labeling objects in images plays a crucial role in many visual learning and recognition applications that need training data, such as image retrieval, object detection and recognition. Manually creating object labels in images is time consuming and, thus, becomes impossible for labeling a large image dataset. In this paper, we present a family of semi-automatic methods based on a graph-based semi-supervised learning algorithm for labeling objects in images. We first present SmartLabel that proposes to label images with reduced human input by iteratively computing the harmonic solutions to minimize a quadratic energy function on the Gaussian fields. SmartLabel tackles the problem of lacking negative data in the learning by embedding relevance feedback after the first iteration, which also leads to one limitation of SmartLabel—needing additional human supervision. To overcome the limitation and enhance SmartLabel, we propose SmartLabel-2 that utilizes a novel scheme to *sample negative examples automatically*, replace regular patch partitioning in SmartLabel by *quadtree partitioning and applies image over-segmentation (superpixels)* to extract smooth object contours. Evaluation on six diverse object categories have indicated that SmartLabel-2 can achieve promising results with a small amount of labeled data (e.g., 1%–5% of image size) and obtain close-to-fine extraction of object contours on different kinds of objects.

Index Terms—Gaussian fields, harmonic functions, labeling objects, quadtree, SmartLabel, superpixels.

I. INTRODUCTION

THE fast growth of visual data on the Internet has created new challenges for the image processing (IP) community. Most IP tasks, such as image annotation [1], content-based image retrieval [2]–[7], and object detection and recognition [8], [9], require training data. Manually labeling images is not only a labor-intensive and time-consuming task but also subject to human operation errors and variances. There has been much attention on attracting Internet users to label images manually such as MIT LabelMe [10] and ESPGame.org, but research on the semi-automatic manner has not been fully studied. Our goal is to offer a semi-automatic framework to label objects in images effectively (focusing on *things* as opposed to *stuff* [11]). We propose a family of semi-automatic labeling methods in this

paper based on a graph-based semi-supervised learning algorithm [12].

To appreciate the difficulties in manually labeling objects, let us go through the labeling process of LabelMe [10]. Users need mark contours of as many objects in the image and for as many images as they like. Often users click mouse >30 times along the outline to finish working on one object. Within 20 months since its inception, about 55 thousand polygons are manually labeled among which about 5 thousand are described. LabelMe’s impact is influential, but the numbers of labeled objects and descriptions are small compared to the amount of contributed human efforts. These difficulties motivate us to seek semi-automatic methods to label objects in images. One solution is to let a user mark a small region of interest (ROI) on the object (e.g., dragging a rectangle) and the computer automatically extracts the object outline and its other instances in the image. Fig. 1 shows examples of user inputs and results by SmartLabel [13] we have proposed and GrabCut [14].

Some research has been done in semi-automatic object extraction using various techniques [11], [14]. Semi-supervised learning (SSL), which leverages the availability of unlabeled data to improve classification, has attracted a lot of attention in the past decade and been proved useful for many problems [12], [15]–[20]. A survey on SSL can be found in [21]. We focus on addressing the problem within an SSL framework. We formulate the problem as follows. Given an image, an ROI is provided by the user. We divide the image into nonoverlapping square patches; any patches overlapping the ROI are considered as labeled samples and put in L and the rest are considered as unlabeled samples and put in U . Formulating the problem in this way allows us to apply any SSL algorithm to classify patches in U . However, the formulation still presents two challenges. First, labeled patches contain only positive samples. Second, L may contain noise because patches which overlap ROI can include background. These two challenges become critical hurdles for many existing SSL algorithms.

We have come to a recently proposed graph-based SSL method [12], which we refer as Zhu’s SSL method in the following. Zhu’s SSL method represents labeled and unlabeled samples as vertexes in a weighed graph. Edge weights represent the similarity between connected vertexes. It adopts Gaussian fields over a continuous state space rather than random fields over a discrete label set. The mean of the field is characterized in terms of harmonic functions and its solution can be efficiently obtained using matrix methods or belief propagation. This *relaxation* to a continuous space has some attractive properties. For instance, lack of negative samples and continuous label values can be naturally handled by Zhu’s SSL method. It is proposed for problems in other domains so there are still difficulties in directly applying it to our problem. In [13], we have proposed SmartLabel based on it. SmartLabel has four novelties. 1) Real numbers from 0 to 1 are used as label values for positive patches. 2) The weighed graph is constructed from

Manuscript received March 25, 2008; revised February 16, 2009. First published April 24, 2009; current version published May 13, 2009. This work was supported in part by General Motors, in part by Carnegie Mellon University CRL grant, and in part by a gift grant from Microsoft. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Til Aach.

W. Wu is with the Language Technologies Institute, School of Computer Science at Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: wenwu@cs.cmu.edu).

J. Yang is with Human-Computer Interaction Institute, School of Computer Science at Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: jie.yang@cs.cmu.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.2009.2017360

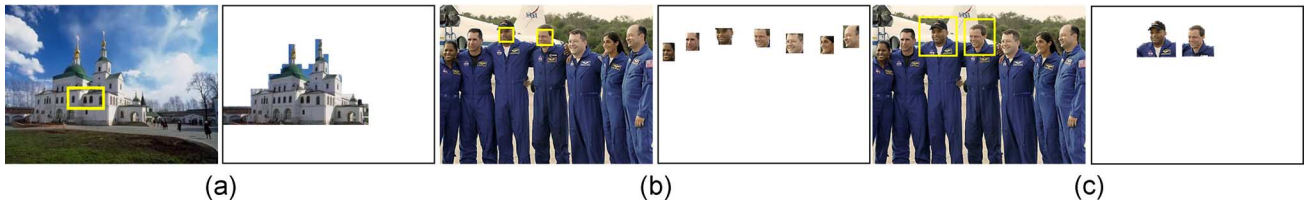


Fig. 1. Illustration of SmartLabel [13] and comparison with GrabCut [14]. SmartLabel allows users loosely drag a rectangle (or a loop stroke) inside an object. It can label both single and multiple objects. In each group, left are input images with user specified ROI (in yellow) and right are extracted object regions. Note GrabCut asks users to mark ROI outside of the object. (a) SmartLabel—single object; (b) SmartLabel—multiple objects; (c) GrabCut [Rother et al. 04].

the input image using two spatial constraints. 3) Harmonic energy minimization is applied iteratively to estimate labels for patches in U and newly labeled patches are added to L for the next iteration. 4) It brings the human in the loop by plugging in relevance feedback (RF) to collect negative samples for learning.

Unfortunately, SmartLabel has some shortcomings such as demand of human supervision for RF and zigzag object boundaries in resulting object areas, shown in [13]. These issues limit SmartLabel’s application to many tasks. To overcome these weaknesses, we propose SmartLabel-2 which improves in three aspects. In particular, we propose applying quadtree to partition images instead of using regular gridding. Secondly, we introduce a novel saliency-based method to sample negative samples. Finally, we adopt image superpixel representation [22] to refine labeling results and extract smooth object boundaries. The evaluation of proposed SmartLabel-2 and comparison with SmartLabel and Zhu’s SSL method in a dataset of six object classes indicate that SmartLabel-2 not only works effectively with a small amount of user input (e.g., 1%–5% of image size) but also achieve very promising results. In some class, SmartLabel-2 obtains pixel-level high performance.

II. RELATIONS TO OTHER OBJECT EXTRACTION TOOLS

Labeling objects in images is closely related to image segmentation, object extraction or image editing, but they are different in terms of applications. Image segmentation has been an active research area for decades. We describe some standard or state-of-the-art interactive object extraction tools: Magic Wand, GrabCut [14], ClickRemoval [23], and central object extraction [24].

Magic Wand as seen in Adobe Photoshop and many other photo-editing tools applies a basic seeded region growing algorithm [25]. It is based on absolute color differences among adjacent pixels. It lets a user select a consistently colored area without having to trace its outline. While the user interface is straightforward, finding the appropriate tolerance level is often cumbersome and it cannot extract an object with multiple colored areas.

ClickRemoval [23] is an interactive tool to erase an object in an image. The user indicates the undesired object by pinpointing it with the mouse cursor. The object extraction step relies on a statistical region-growing segmentation technique using color information and the hole-filling step applies background texture-based synthesis. This method does not support extracting objects at multiple locations.

Central Object Extraction [24] is developed for object-based image retrieval. It does not require user interactions

and automatically extracts central objects. It relies on two underlying assumptions: interesting objects are located near the image center and contain regions with significant color distributions. It cannot extract objects at various places and its assumptions limit its applications.

GrabCut [14] and SmartLabel solve a similar task but are different in three respects. 1) *User initial inputs are different* as shown in their original papers. GrabCut’s common input is an ROI *outside* the object while SmartLabel needs an ROI *within* the object.¹ 2) *Inference models are different*. SmartLabel adopts Gaussian fields with harmonic functions. GrabCut extends the graph-cut approach [17]. 3) *Outcomes are different*. GrabCut achieves fine foreground and background segmentation while SmartLabel can extract multiple similar/same objects. Fig. 1 shows a comparison.

To summarize, SmartLabel is different from other tools in terms of user inputs, underlying models and outcomes.

III. LABELING OBJECTS IN IMAGES USING ZHU’S SSL METHOD

We first describe SSL notations and the application of Zhu’s SSL method [12] to label objects.

SSL is a problem of learning from labeled and unlabeled data. Given a data set, $X = \{x_1, \dots, x_l, x_{l+1}, \dots, x_n\}$, and a label set, $\mathcal{C} = \{1, \dots, c\}$, the first l samples have labels $\{y_1, \dots, y_l\} \in \mathcal{C}$ and remaining samples are unlabeled. We call $L = \{(x_1, y_1), \dots, (x_l, y_l)\}$ the labeled set and $U = \{(x_{l+1}, y_{l+1}), \dots, (x_n, y_n)\}$ the unlabeled set. Graph-based SSL methods consider a connected graph, $G = (V, E)$, with n vertexes in V corresponding to n samples, where vertexes $L = \{1, \dots, l\}$ are labeled samples and vertexes $U = \{l + 1, \dots, n\}$ are unlabeled samples. The edges, E , are weighted by an $n \times n$ affinity matrix, W , computed by certain distance metrics.

The goal of Zhu’s SSL method is to compute a real-valued labeling function, $g(\cdot) : V \rightarrow \mathcal{R}$, on G with certain nice properties, and then to assign labels for U based on $g(\cdot)$. The labeling function is constrained to assign labels such as $g(i) = y_l(i) \equiv y_i$, on L , $i = 1, \dots, l$. Gaussian field configuration aims to make unlabeled samples that are nearby in the graph have similar labels. This motivates choosing a quadratic energy function, $E(g) = (1/2) \sum_{i,j} w_{ij} \cdot (g(i) - g(j))^2$. In the n -dim space, $x \in \mathbb{R}^n$, the weight matrix, W , can be defined as, $w_{ij} = \exp(-\sum_{d=1}^n ((x_{id} - x_{jd})^2 / 2\sigma_d^2))$, where x_{id} is d th component of the feature vector $x_i \in \mathbb{R}^n$, and $\sigma_1, \dots, \sigma_n$ are length scale hyperparameters for each dimension.

¹GrabCut and SmartLabel both can work by being given a few background or foreground pixels/patches.

To assign a probability distribution on $g(\cdot)$, a Gaussian field is formed as $p_\beta(g) = e^{-\beta E(g)} / Z_\beta$, where β is the inverse temperature parameter and Z_β is the partition function $Z_\beta = \int_{g|g_l=L} \exp(-\beta E(g)) dg$, which normalizes over all functions constrained to g_l on L , where $(x_i, y_i) \in L, i = 1, \dots, l$. To compute the harmonic solutions of $g(\cdot)$, we minimize $E(g)$ subject to the labeled data constraint

$$g = \arg \min_{g|L=g_l} E(g) = \arg \min_{g|L=g_l} \frac{1}{2} \sum_{i,j} w_{ij} \cdot (g(i) - g(j))^2 \quad (1)$$

in other words, it satisfies $\Delta g = 0$ on all unlabeled samples in U and is subject to $g|L = g_l$. Here Δ is called *combinatorial Laplacian*, and defined as $\Delta = D - W$, where $D = \text{diag}(d_i)$, $d_i = \sum_j w_{ij}$.

The harmonic property indicates $g(\cdot)$ at each unlabeled sample is average of $g(\cdot)$ at its nearby samples

$$g(j) = \frac{1}{d_j} \sum_{ij} w_{ij} \cdot g(i), \quad \text{for } j = l+1, \dots, l+u \quad (2)$$

which maintains the smoothness constraint of $g(\cdot)$ with respect to G . Expressed in the matrix form, (1) can be rewritten as $g(\cdot) = Qg(\cdot)$, where $Q = D^{-1}W$. To compute the harmonic solution, W is split into four blocks based on the separation of L and U . $W_{n \times n} = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}_{n \times n}$

Denote the target labeling function $g(\cdot) = \begin{bmatrix} g_l(\cdot) \\ g_u(\cdot) \end{bmatrix}_{n \times c}$, and $g_l(\cdot)$ denotes labels on L , $g_u(\cdot)$ denotes labels on U , and c is the number of classes. The unique harmonic solution $\Delta g(\cdot) = 0$ subject to $g(\cdot)|L = g_l(\cdot) \equiv y_i, i = 1, \dots, l$ is given as a $u \times c$ matrix $g_u(\cdot)$

$$g_u(\cdot) = (D_{uu} - W_{uu})^{-1} \cdot W_{ul} \cdot g_l(\cdot). \quad (3)$$

Algorithm 1 shows Zhu's SSL method for labeling objects. SmartLabel improves it in several aspects.

Algorithm 1: Labeling Objects in Images using Zhu's SSL method

Input An image (I) and input ROI(s).

Initialize Divide I in patches of $B \times B$ pixels. patches in ROI are put in L and others in U .

Object Labeling using Zhu's SSL method

1. *Construct the weight matrix on L and U .* Form the weight matrix W using the weight factor defined in Section III and $W_{ii} = 1$.
 2. *Construct the combinatorial Laplacian.* Compute the matrix $\Delta = D - W$, in which D is a diagonal matrix with D_{ii} equal to the sum of the i th row of W .
 3. *Compute the harmonic solution.* Compute the label prediction on U using the equation $g_u(\cdot) = (D_{uu} - W_{uu})^{-1} \cdot W_{ul} \cdot g_l(\cdot)$.
 4. *Assign labels to unlabeled data.* For each x_i in $U (i > l)$, assign the label as $y_i = g_u(x_i)$.
 5. *Output object regions.* Produce object regions as combination of patches which are labeled positive.
-

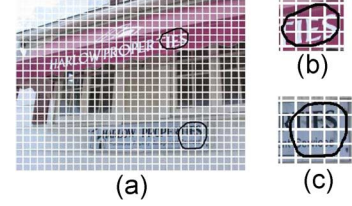


Fig. 2. Illustration of necessity of soft labeling. After regular gridding, two ROIs (in black) include a few complete patches but overlaps with other patches on the borders. Similar problems happen too when the user drags a rectangle ROI.

IV. SMARTLABEL

A. Soft Labeling

To formulate an SSL-based labeling problem, we construct L and U . Given an image and ROI(s), we partition the image into nonoverlapping patches of size $B \times B$ (pixels). Each patch is treated as a sample. Patches that are in the ROI or overlap with it are treated as positive samples and added in L . Other patches are treated as unlabeled samples and put in U . The number of labeled samples is l and the number of unlabeled samples is u and $n = l + u$. Note L does not contain negative samples yet.

Most SSL methods use the 0 or 1 labeling strategy (0 means negative and 1 means positive). Problems rise with this strategy because samples in L are different: some patches are within ROI but others partially overlap with ROI. If we use uniform labeling $y_i = 1, i = 1, \dots, l$ for all samples in L , the labeled data can be noisy because some patches may contain background. Fig. 2 shows an example sign image with two ROIs. Some patches just partially overlap ROIs and some only contain canvas background.

One solution is to give up all patches which only partially overlap with ROI and put them in U . But this is expensive by sacrificing already small L . Instead, we use soft labeling by relaxing the label values to $y_i \in [0, 1]$. The label value for each sample in L is computed as the ratio of its area within ROI to the patch size. Since the mapping function $g(\cdot)$ in Zhu's SSL method is real-value defined, this relaxation appears to be naturally handled by it.

B. Graph Construction With Spatial Constraints

Labeling objects in an SSL manner possesses some unique properties which other SSL problems do not have. One property is that location and context are important in images. Pixels/patches nearby tend to belong to one object instance rather than those far away. Regions at nearby locations are more likely to contain similar color distributions than those far away. By observing this spatial continuity property, we next describe two spatial constraints which we embed to build the weighed graph: a location-based similarity distance metric and separation of U into U_r and U_n .

We first define a new distance metric by introducing an additional spatial distance term

$$w_{ij} = \exp \left(- \sum_{d=1}^m \frac{(x_{id} - x_{jd})^2}{\sigma_d^2} \cdot \frac{\sum_{k=1}^2 (z_i^k - z_j^k)^2}{\epsilon} \right) \quad (4)$$

where the first term shows the feature distance as in [12] and the second term measures the spatial distance between two sam-

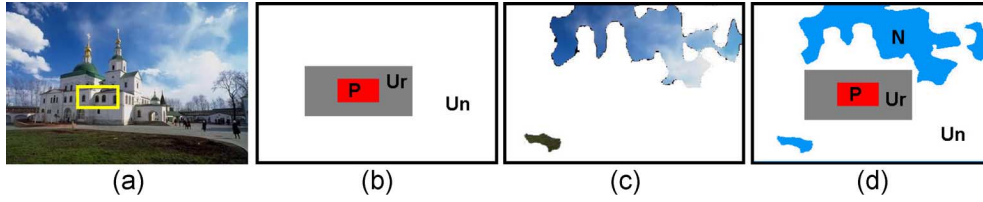


Fig. 3. SSL settings in SmartLabel (b) and SmartLabel-2 (d). P : the positive set; U_r : the relevant unlabeled set; U_n : the irrelevant unlabeled set; N : the negative set. (a) Input image with an ROI. (c) Detected less informative regions (Section V).

ples. m is the number of dimensions of feature vector x_i , and $\sigma_1, \dots, \sigma_m$ are length scale hyperparameters for each dimension. z_i^1 and z_i^2 are image coordinates of x_i and ϵ is a normalization term. The new distance metric approximates the relevancy between two samples by combining Mahalanobis distance in the feature space and Euclidean distance in $2 - D$.

In most SSL algorithms, samples in U are treated equally in learning, but this assumption may conflict with the location-dependent property. We propose treating unlabeled samples differently in terms of their image locations. We separate U into two subsets, U_r and U_n , based on the location of L . U_r includes unlabeled patches that are close to L and used in learning. U_n includes the rest. Fig. 3 shows an illustration of this setting. With this scheme, we incrementally estimate labels on U_r without sacrificing the global configuration. SmartLabel is an iterative approach and T iterations guarantee all unlabeled patches will be included in the graph eventually. Furthermore, separation of U prevents the graph learning from adding many false positives in L , which can easily happen due to small number of samples in L . Also, searching globally instead of incrementally using L is problematic because the target object concept is under-represented by L without negative samples. By incrementally exploring neighborhood of L and adding high-confidence predicted positive patches in L iteratively, our method can quickly converge and achieve robust results.

C. Iterated Harmonic Energy Minimization

SmartLabel introduces an iterated harmonic energy minimization in place of the one-shot minimization in Zhu's SSL method. This scheme allows an automatic increment of L after each iteration. Note in most SSL other algorithms L is fixed but in SmartLabel L is dynamically updated.

Algorithm 2: SmartLabel* (w/o. relevance feedback)

Input An image (I) and input ROI(s).

Initialize Divide I in patches of $B \times B$ pixels. Create L and U as introduced in Section IV-A Soft Labeling. Set the max iteration as T and neighborhood size as H . $U_r = \emptyset$, $U_n = \emptyset$.

SmartLabel*

for $t = 1$ to T **do**

1. *Update* U_r & U_n . Extract neighboring patches from L in H pixels and put them into U_r , and remaining unlabeled patches are put in U_n .
2. *Compute the matrix* W' *based on* L *and* U_r . Form the weight matrix W' using the weight factor defined in (4) and $W'_{ii} = 1$.

3. *Construct the Laplacian*. Compute the matrix $\Delta = D' - W'$, in which D' is a diagonal matrix with D'_{ii} equal to the sum of the i th row of W' .
4. *Compute the harmonic solution*. Compute the label prediction on U_r using the equation $g_{u_r}(\cdot) = (D_{u_r u_r} - W_{u_r u_r})^{-1} \cdot W_{u_r l} \cdot g_l(\cdot)$.
5. *Augment* L . Add newly predicted unlabeled patches from U_r to L using the predictions g_{u_r} .

Output results. Produce extracted object regions from patches in L .

Algorithm 2 shows main elements of SmartLabel* (without relevance feedback). In each iteration, samples in U_r and L are studied and labels are predicted for U_r . SmartLabel* runs a fixed number of iterations or until no more unlabeled samples can be added into L . False positives are restrained during the label propagation by applying soft labeling and separation of U_r and U_n . In the following we analyze the time complexity of SmartLabel*. Step (1) takes $O(lu)$ time for update U_r and U_n computation. Sizes of L , U and U_r vary for applications. Step (2) takes $O((l + u_r)^2)$ time for computing the weight matrix of L and U_r . Step (3) takes $O(l + u_r)$ for Laplacian construction. Step (4) takes $O(u_r^2 + u_r l)$ for computing the harmonic solution. And finally the augmenting of L takes $O(u_r)$ time. Thus, the time complexity, $C(T)$, of SmartLabel* in T iterations is $C(T) = O(T \cdot lu)$ when $l \sim u_r$. One way to speed up is to compute W for the whole image once and shuffle rows and columns to create W' based on updated L and U_r at every iteration.

To compute g_{u_r} , W is split in nine blocks based on L , U_r and U_n as $W_{n \times n} = \begin{bmatrix} W_{l \times l} & W_{l \times u_r} & W_{l \times u_n} \\ W_{u_r \times l} & W_{u_r \times u_r} & W_{u_r \times u_n} \\ W_{u_n \times l} & W_{u_n \times u_r} & W_{u_n \times u_n} \end{bmatrix}$ similarly for the matrix D .

Since only U_r are considered, we denote W' as the first *four* blocks in upper left of W as $W'_{p \times p}$, where $p = l + u_r$. The harmonic solution of U_r is given as

$$g_{u_r}(\cdot)_{u_r \times c} = (D_{u_r \times u_r} - W_{u_r \times u_r})^{-1} \cdot W_{u_r \times l} \cdot g_l(\cdot)_{l \times c}. \quad (5)$$

SmartLabel* supports marking of multiple ROIs, which either show many instances of one object or different objects. Then one-against-all classifiers compete using $g_{u_r}^c(j)$, $c = 1, \dots, K$ as posterior probability. In particular, an unlabeled sample j is labeled as follows. First, compute the harmonic solution $g_{u_r}(\cdot)$, a $u_r \times c$ matrix and obtain the j th row which corresponds to the sample j . Second, identify the highest value in the row vector and label the sample j accordingly: $g_{u_r}(j) = \arg \max_{c \in C} g_{u_r}^c(j)$.

D. Relevance Feedback

Note that initial ROI(s) provides only positive samples but lacks negative samples. This is problematic in SSL inference. The harmonic energy minimization in the first iteration has a difficult task of discriminating *nonobject* patches that have similar appearance (color and/or texture) of the *object* from true *object* patches in U_r , since no negative samples are available initially in L , which is crucial to reduce false positives. To alleviate the problem of lacking negative samples, we bring the human in the loop by embedding relevance feedback (RF) [2] in SmartLabel*. Our aim is to improve performance by including RF data, but to reduce human supervision, we apply RF only in the first iteration. In other cases where the amount of human supervision is not a concern while the performance is crucial, the strategy will be to apply RF in all iterations to achieve the best results. In this paper, we only employ the first strategy.

Algorithm 3: SmartLabel (w. relevance feedback)

Input & Initialization are the same as SmartLabel*.

SmartLabel

1. For $t = 1, 2, \dots, T$, similar to Step 1 in **SmartLabel*** except the program (or user) gives relevance feedback before augmenting L in sub-step (e) at Iteration 1.
 2. *Output resulting extractions.* Produce extracted regions of the object from patches in L .
-

We present patches that are labeled as the *object* by SmartLabel* to the user, ranked by $g_u(\cdot)$. When the user labels a patch as a negative example, its neighboring patches in the graph including itself are added to L as negative samples. The same process is done to *nonobject* patches labeled by SmartLabel*. Other correctly labeled patches are also added in L . After RF, the graph learning in later iterations has more positive samples and additional negative samples to represent the target object concept discriminatively. The introduction of RF in SmartLabel* makes it possible to learn *object* and *nonobject* regions in the image simultaneously and also utilize the underlying graph structure from the unlabeled samples to further improve labeling performance. Algorithm 3 shows the details of SmartLabel (w. RF).

V. SMARTLABEL-2

SmartLabel is not good enough. Its limitations include lack of negative samples, demanding human feedback, zigzag object boundaries and holes in extracted regions. To overcome these issues we propose SmartLabel-2 by extending SmartLabel [13] in three aspects.

- 1) SmartLabel needs RF to collect negative samples. With the goal of minimizing human effort, SmartLabel-2 revise a spectral residual approach to sample negative samples from the input image.
- 2) Furthermore, SmartLabel-2 apply quadtree to partition instead of regular gridding used in SmartLabel. This not only increases granularity of patches to reach object boundaries but also improves labeling accuracy by reducing mixture of object and background in one patch.

- 3) Finally, SmartLabel-2 refine labeled object patches via superpixels [22]. Each image's superpixels are computed offline and used with labeled patches to extract object boundaries.

A. Sampling Negative Samples From Single Image

Negative samples are often used with positive samples to train a supervised classifier or a semi-supervised classifier. However, in our problem, ROI(s) inside the object only contains positive patches. Zhu's SSL method can learn a classifier using positive data and unlabeled data, but its performance is affected for missing negative samples. SmartLabel tackles the issue by requiring the user to give relevance feedback on newly predicted patches. This practice is expensive and inefficient because of the large number of images to label and limited manpower.

To overcome this challenge, we need an automatic way to sample negative examples from the input image. One solution is to sample along image borders, but this invalidates the advantage of SmartLabel-2—*extracting objects anywhere in the image including borders*. Instead, we adopt a spectral residual approach (SR) [26] to detect uninformative image areas to sample negative patches. Algorithm 4 shows the scheme and Fig. 3(c) and (d) shows an example. Hou's method detects information rich regions. Using Inverse Fourier Transform, the method outputs the image's *saliency map* which contains the nontrivial image parts. We revise the method to detect less informative regions. We apply the inverse of saliency map to obtain a *background map* which mainly contains uninformative regions. Fig. 3 shows detected uninformative regions by our scheme (top 20%) in (c) and the SSL setting in (d).

Algorithm 4: Saliency-based sampling of negative samples from a single image

Input and Initialization: input image (I), user-input ROI (P), (U_r) set and a threshold (T_s).

Saliency-based Sampling of Negative Samples from Single Image

1. Compute *saliency map* of I using Hou's approach in [26].
 2. Sort pixels in *saliency map* in *ascending* order, preserve top T_s portion to obtain negative set N .
 3. Combine P , U_r and N as the SSL labels of the image.
-

Our scheme cannot cope with all cases and suffers when the object(s) contains uninformative regions decided by SR. Fortunately, this happens only in a few cases when labeling sky and ground among other uninformative objects. Even so, SmartLabel-2 works by skipping the sampling of negative samples step.

B. Quadtree Partitioning

A quadtree [27] is a tree data structure in which each internal node has up to four children. Quadtree is often used to partition a 2-D space by recursively subdividing it into four quadrants or regions. A similar partitioning is also known as a Q-tree. All forms of quadtrees share some common features. 1) They decompose space into adaptable cells. 2) Each cell has a maximum capacity.

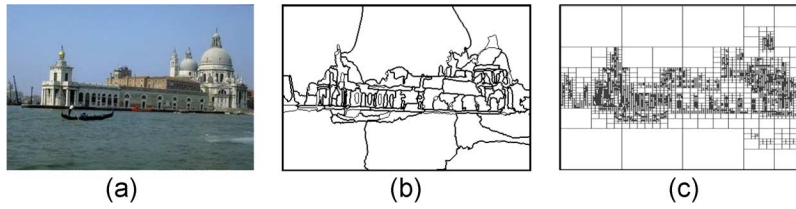


Fig. 4. For SmartLabel-2, (a) an input image; (b) its superpixel map ($P = 100$); (c) its quadtree partition (min patch = 2×2).

When maximum capacity is reached, the cell splits. 3) The tree directory follows the spatial decomposition of the quadtree.

SmartLabel-2 replaces regular gridding by quadtree-based partitioning which changes the underlying patch structure. Fig. 4 depicts quadtree partitioning of an example image. We adopt *qtdecomp* function in Matlab to perform quadtree decomposition, which needs a preprocessing step to resize the input image to a square image at resolution of 2^n . Resizing input images to be square changes the height-to-width, it does not affect patch-based feature extraction much. This resembles unaccessible selection of optimal size and ratio of sliding windows which are commonly used in the image processing and computer vision fields. Quadtree partitioning not only increases granularity of patches to reach arbitrary object boundaries but also improves accuracy by minimizing mixture of object and background in one patch. It takes less than half second to process a 512×512 image using Matlab on a 3.2-GHz CPU.

C. Labeling Refinement Based on the Image's Superpixels

Many existing image processing algorithms use image pixel as the basic representation unit. However, a pixel is not a natural representation unit of real scenes [22]. It is only a representation unit of digital imaging. It would be more natural to work with perceptually meaningful units obtained from a low-level clustering process. One well-known way [22] to achieve this natural representation is to apply Normalized Cuts (NCut) [15] to partition an image into P segments, called superpixels. Typically with some over-segmentation, NCut preserves most object structures and boundaries.

NCut is a well-known segmentation algorithm which uses spectral clustering to exploit pairwise brightness, color and texture affinities between pixels. As in [22], we only perform the preprocessing over-segmentation using NCut. This is done *offline* because it normally takes 3 min to process a 320×240 image. Fig. 4 shows an image's superpixel map. The labeled regions are compared with superpixels to generate refined object boundaries. Algorithm 5 shows the process.

Algorithm 5: Refine quadtree-based labeling results using superpixels

Init: input image (I), labeling mask (M), image superpixels (S) and number (P), and result map (R).

Refine quadtree-based labeling results using superpixels.

1. For $i = 1, 2, \dots, P$; Compute overlap between $S(i)$ and M , if the overlap is $> 50\%$, add $S(i)$ in R .
 2. Use R to cut out predicted object regions from I .
-

VI. EXPERIMENTS

A. Experimental Setting

To evaluate SmartLabel-2 and compare it with SmartLabel, we have collected images from public datasets [28]–[30] which are the same data used in [13]. Two sets of experiments are conducted, *quantitative analysis* with simulated ROI and RF, and *qualitative analysis* with user input ROIs. Six classes of objects are studied: *airplane, animal, building, car, flower and text*. These classes cover a range of man-made and natural objects.

Two kinds of low-level features are used: color information and Gabor texture. Both features are extracted for each patch. The color info consists of the first and second moments for each RGB channel and color histograms with 32 bins for each channel. This results in a 102-dim color vector for each patch. The texture features are obtained from convolving each patch with various Gabor filters. Here, 4 scales and 8 angles are used. The center and second-order moments are computed from each filter output. This results in a 64-dim texture vector. After normalization, we concatenate color and texture vectors into a 166-dim vector. Hyperparameters, $\sigma_1, \dots, \sigma_m$, in (4) are set as variance for each dimension.

To quantitatively compare SmartLabel and SmartLabel-2, we simulate input ROIs by randomly selecting a 10% of object regions. In SmartLabel, after the first iteration, we simulate RF by labeling top 5 false positives as negative based on ground truth and adding them to L . For every new negative sample, we extract its 3 nearest neighbors in the graph and add them to L . Duplication is not allowed. F_1^M value is used as overall performance metric and defined as $F_1^M = 2 \sum_{i=1}^K R_i \sum_{i=1}^K P_i / K (\sum_{i=1}^K P_i + \sum_{i=1}^K R_i)$, where P_i and R_i are precision and recall computed for i th image and K is the number of images in the class. Except SmartLabel-2, other methods in our study use regular gridding as in [13]. We report results based on two patch sizes, $B = 8$ or $B = 16$ and for various sizes of U_r , such as $H = 32$ and $H = 64$.

B. Performance Evaluation

We compare four semi-supervised labeling methods as follows. Zhu's SSL method (**Zhu's**) takes ROI(s) as positive samples and makes one-shot labeling prediction on all unlabeled patches. **SmartLabel*** utilizes the iterative labeling scheme without RF. **SmartLabel** includes RF. **SmartLabel-2** requires no human supervision during learning and incorporates quadtree partitioning and sampling of negative samples.

Fig. 5 depicts performance curves on three object categories, (a) airplane, (b) text, and (c) flower. Each subfigure contains the curves of Zhu's, SmartLabel*, SmartLabel and SmartLabel-2

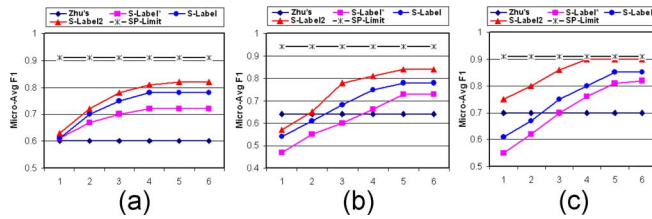


Fig. 5. Comparing four algorithms: Zhu's SSL method [12] (baseline), S-Label* (SmartLabel w/o. relevance feedback), S-Label (SmartLabel), and S-Label2 (SmartLabel-2). Each curve plots F_1^M against iterations. SP-Limit: superpixel-based upper bound. (a) Airplane; (b) text; (c) flower.

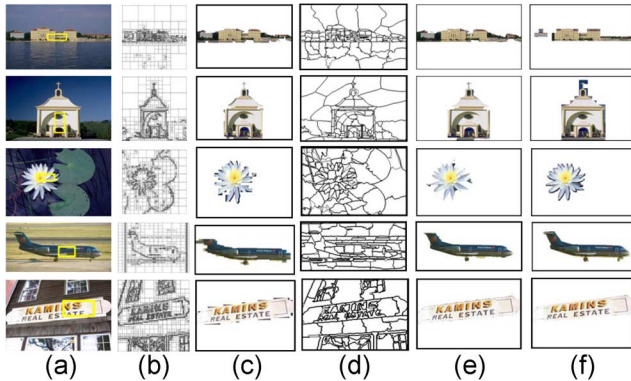


Fig. 6. Results by SmartLabel-2. (a) Input images with ROIs. (b) Quadtree (QT) partitions in square images ($thr = 0.27$ and minimal patch size is 2 pixels). (c) SmartLabel-2's labeling results based on QT. (d) Superpixel (SP) maps ($P = 100$). (e) SmartLabel-2's results using SP. (f) Ground truth. For complete results of all six classes, see our online paper link.

and the superpixel-based upper bound (SP-Limit), which is reachable performance by using superpixel representation. Since Zhu's is an one-shot approach, its performance does not change as iterations. In the first loop, Zhu's does better than or is comparable to other methods. This is because Zhu's predicts labels on U while other methods predict only on U_r . We observe that after three or four iterations, the other methods pick up and do better than Zhu's. Comparing SmartLabel*, SmartLabel and SmartLabel-2, we see that their performances stand in the order of SmartLabel* \ll SmartLabel \ll SmartLabel-2. This observation corroborates the benefits of new improvements proposed in SmartLabel-2. We also notice SmartLabel-2 achieves near SP-Limit F_1^M on the *flower* class. With the number of loops increasing, all three SmartLabel methods monotonically improve and normally converge at the 5th iteration. In summary, SmartLabel (w. RF) performs better than SmartLabel* but underperforms SmartLabel-2.

Fig. 6 shows results by SmartLabel-2 on several classes. The first column lists input images with input ROIs. The second column shows corresponding superpixel maps. The third column shows labeling results produced by SmartLabel-2. The last column depicts ground truth. We have several interesting observations from the results. First, SmartLabel-2 generalizes very across man-made objects such as airplanes, buildings and text signs, and natural objects such as flowers and animals. Second, superpixels are proven to be effective in SmartLabel-2 to extract most of object boundaries despite its heavy computation. Third, the results suggest that SmartLabel-2 can assist

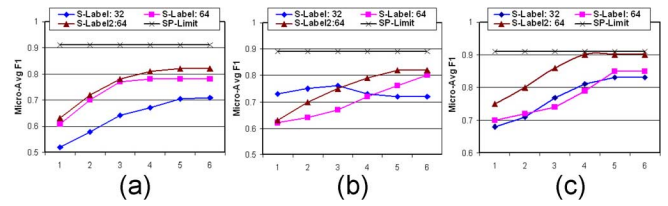


Fig. 7. Impact of U_r 's size on performance. Comparison on three classes. $B = 16$. Two settings for SmartLabel: $H = 32$ and $H = 64$. For SmartLabel-2, $H = 64$. Each curve plots F_1^M against iterations. SP-Limit: superpixel-based upper bound. (a) Airplane; (b) building; (c) flower.

humans to generate ground truth of object locations in images. Top two building examples in Fig. 6 show SmartLabel-2 does a nice on extracting man-made objects. For complete results of all six classes, please see our submitted supplementary package.

Fig. 7 shows labeling curves in terms of the size of U_r on three classes: (a) airplane, (b) building and (c) flower. $B = 16$ is used for both SmartLabel and SmartLabel-2. We compare them in two cases: $H = 32$ and $H = 64$. Each subplot shows F_1^M curves. Fig. 7 shows several interesting points. First, a large U_r ($H = 64$) results in higher performance than $H = 32$ along iterations for the *airplane* class. However, the observation is opposite for the *building* class in first few iterations. To investigate the reasons, we examine images in both classes and find that backgrounds in *airplane* images are simpler than those in *building* images. This explains why a large U_r boosts performance on the *airplane* class but degrades performance in complex *building* images. Also, performance trends seem interesting in *building*. We examine images and intermediate results at every iteration, and we find that *building* regions across the class have different sizes and shapes. $H = 64$ gives better performance on some *building* images than others and $H = 32$ can also favor some images not others. This is why the blue curve slightly decreases after the third iteration. In contrast SmartLabel-2 consistently outperforms two other SmartLabel variants.

To have a qualitative comparison between SmartLabel-2 and SmartLabel [13], Fig. 8 shows various labeling results by two methods. The first row shows input images with ROIs. The second row shows SmartLabel's results and the third row is SmartLabel-2's results. SmartLabel obtains reasonable results but with zigzag boundaries, holes and some false positives. SmartLabel-2 gives results with smooth object boundaries and no holes. The *quantitative* improvement by F_1^M may not be significant on these results, but the *qualitative* improvement in terms of user satisfaction is distinctive.

Table I lists a comparison of SmartLabel-2 and methods in [13] on a same dataset via the F_1 metric. The average of 10 runs' F_1^M is shown for each method. Input images are resized to 512×512 as we use the Quadtree implementation in Matlab in the experiments. Results of published methods are slightly different from those in [13] due to different experiment settings. We observe that large patches ($B = 16$) improve Zhu's because segmenting images into small patches would create more patches which merely contain background and bring noise to L . We see that SmartLabel-2 achieves near SP-Limit F_1^M on the *flower* class. For other classes, SmartLabel-2 performs very

TABLE I
COMPARING METHODS IN [13] WITH SMARTLABEL-2 (S-LABEL-2) ON A SAME DATASET VIA F_1 . FOR S-LABEL* AND S-LABEL, $H = 64$, $B = 8$ OR $B = 16$. S-LABEL-2 USES $B = 16$ AND $H = 64$. SP: USING SUPERPIXELS AND NSP: NOT USING; SPL: SUPERPIXEL LIMIT; $c = .5$: 50% OVERLAP TO SELECT A SUPERPIXEL; $a-F_1^M$: AVERAGE F_1^M

	Zhu's SSL		S-Label*		S-Label		S-Label-2		SPI
	B=8	B=16	B=8	B=16	B=8	B=16	nSP	SP	c=.5
plane	0.56	0.60	0.71	0.72	0.78	0.77	0.78	0.82	0.91
animal	0.57	0.53	0.72	0.75	0.74	0.76	0.79	0.85	0.92
bldg.	0.60	0.63	0.75	0.70	0.80	0.78	0.77	0.82	0.89
car	0.54	0.59	0.72	0.69	0.76	0.73	0.74	0.78	0.91
flower	0.66	0.70	0.82	0.81	0.85	0.83	0.85	0.90	0.91
text	0.64	0.61	0.73	0.71	0.78	0.78	0.80	0.84	0.94
$a-F_1^M$	0.60	0.61	0.74	0.73	0.79	0.78	0.79	0.84	0.91

TABLE II
SUMMARY OF FOUR SSL LABELING METHODS DESCRIBED IN THIS PAPER

	Zhu's	S-Label*	S-Label	S-Label-2
User input	✓	✓	✓	✓
Regular partitioning	✓	✓	✓	
Quadtree partitioning				✓
No human supervision	✓	✓		✓
Negative sampling				✓
Iterative labeling		✓	✓	✓
Boundary refinement				✓



Fig. 8. Comparison between SmartLabel [13] and SmartLabel-2. In two groups of three rows. The first row shows input images with ROIs. The second row shows results by SmartLabel. The third row shows results by SmartLabel-2. Two merits of SmartLabel-2 compared to SmartLabel: 1) smooth object boundaries instead of zigzag lines and 2) less misclassified patches.

well and better than SmartLabel. The *car* class is harder than others because its images are all gray so color info is not explored. SmartLabel runs for 15–30 s processing an image of 384×256 using 16×16 patches on an Intel Pentium 4 CPU 3.2 GHz (dual-core). SmartLabel-2 requires additional 10–30 s

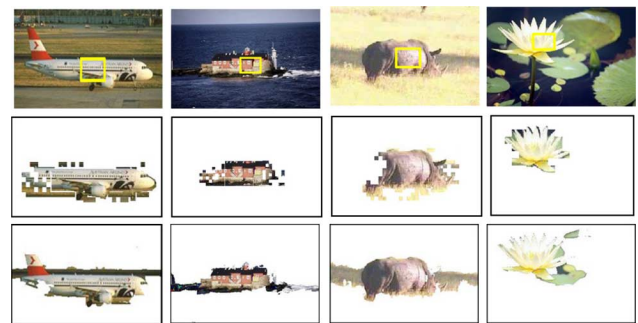


Fig. 9. Some failure examples by SmartLabel-2 (bottom row). Results by SmartLabel are shown in the middle row.

on average. Computing superpixels takes several minutes per 320×240 image using NCut Matlab code.²

Fig. 9 shows some failure cases by SmartLabel-2. After monitoring the intermediate results at each iteration, we found that most of these failures resulted from two reasons: 1) likeness between background and the object appearance and 2) background regions are added in (L) as positive data in initial iterations. Table II summarizes properties of four semi-supervised labeling methods proposed in this paper.

VII. DISCUSSIONS AND SUMMARY

We have presented a family of semi-automatic object labeling methods based on Zhu's SSL method [12] in a semi-supervised learning framework. Given an image with a small-size ROI, our methods can extract the contour of the object and also the same or similar objects at other locations in the image. In particular, we have proposed SmartLabel-2 to enhance SmartLabel [13] by overcoming three of its limitations. Our experiments on various

²<http://www.seas.upenn.edu/timothee>.

TABLE III
COMPARING QUADTREE (QT) AND SUPERPIXEL (SP, $P = 100$) ON
COMPUTATION (AVG.) AND FR. 512 \times 512 IMAGES

	QT	SP
Running Time in Matlab (s)	0.4	836.5
Feature Representation (FR)	✓	N/A

object classes have demonstrated that SmartLabel-2 not only outperforms SmartLabel but also achieves close-to-fine extraction of object contours in many classes. One future direction is to reinvent SmartLabel-2 for labeling objects in videos.

One question rises when using over-segmentation in SmartLabel-2—*why not use over-segmentation patches (superpixels) directly as the labeling units?* We offer three thoughts for discussion: 1) *Computation*. Generating 100 superpixels on a 512 \times 512 image can take >10 min but *quadtree* normally takes <1 second, both using Matlab. We do not seek real time computing in this work. 2) *Feature representation (FR)*. FR is not well addressed for superpixels yet. Many current FR methods are for representing features on regular regions not arbitrary ones. Apart from the FR issue, normalization and similarity measurement can be problematic too. 3) *Effectiveness*. Applying quadtree partitioning and refinement by superpixels has led to promising results in our experiments. We would like to leave this interesting direction for future audience. Table III shows comparison of quadtree and superpixels based on computation and FR.

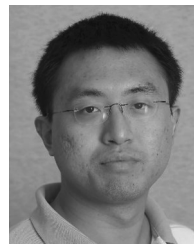
ACKNOWLEDGMENT

The authors would like to thank X. Zhu for insightful advice to help improve their manuscript. They would also like to thank the anonymous reviewers for their insightful comments and detailed suggestions.

REFERENCES

- [1] T. Volkmer, J. R. Smith, and A. P. Natsev, "A web-based system for collaborative annotation of large image and video collections: An evaluation and user study," presented at the ACM Multimedia, 2005.
- [2] Y. Rui and T. S. Huang, "A novel relevance feedback technique in image retrieval," presented at the ACM Multimedia, 1999.
- [3] A. W. M. Smeulders, M. Worring, A. G. S. Santin, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Patten Anal. Mach. Intell.*, 2000.
- [4] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLicity: Semantics-sensitive integrated matching for picture libraries," *IEEE Trans. Patten Anal. Mach. Intell.*, 2001.
- [5] F. Jing, B. Zhang, F. Lin, W.-Y. Ma, and H.-J. Zhang, "A novel region-based image retrieval method using relevance feedback," presented at the ACM Workshops on Multimedia: Multimedia Information Retrieval, 2001.
- [6] T. Deselaers, D. Keysers, and H. Ney, "FIRE—Flexible image retrieval engine: ImageCLEF 2004 evaluation," presented at the Cross-Language Evaluation Forum (CLEF) Workshop, 2004.
- [7] Y. Chen, J. Z. Wang, and R. Krovetz, "CLUE: Cluster-based retrieval of images by unsupervised learning," *IEEE Trans. Image Process.*, 2005.
- [8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," presented at the CVPR, 2001.
- [9] L. Li, W. Huang, I. Y. H. G., and Q. Tian, "Foreground object detection from videos containing complex background," presented at the ACM Multimedia, 2003.

- [10] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: A database and web-based tool for image annotation," *Int. J. Comput. Vis.*, 2008.
- [11] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation-maximization and its application to image querying," *IEEE Trans. Patten Anal. Mach. Intell.*, 2002.
- [12] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," presented at the ICML, 2003.
- [13] W. Wu and J. Yang, "SmartLabel: An object labeling tool using iterated harmonic energy minimization," presented at the ACM Multimedia, 2006.
- [14] C. Rother, V. Kolmogorov, and A. Blake, "'GrabCut'—Interactive foreground extraction using iterated graph cuts," presented at the ACM SIGGRAPH, 2004.
- [15] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Patten Anal. Mach. Intell.*, 2000.
- [16] Y. Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Patten Anal. Mach. Intell.*, 2001.
- [17] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," presented at the ICCV, 2001.
- [18] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," presented at the ICML, 2001.
- [19] M. Belkin and P. Niyogi, "Semi-supervised learning on riemannian manifolds," *Mach. Learn.*, 2004.
- [20] D. Zhou, O. Bousquet, and T. N. Lal, "Learning with local and global consistency," presented at the NIPS, 2004.
- [21] X. Zhu, *Semi-Supervised Learning Literature Survey*, Univ. Wisconsin-Madison, Computer Sciences TR 1530, 2008.
- [22] X. Ren and J. Malik, "Learning a classification model for segmentation," presented at the ICCV, 2003.
- [23] F. Nielsen and R. Nock, "Clickremoval: Interactive pinpoint image object removal," presented at the ACM Multimedia, 2005.
- [24] S. Kim, S. Park, and M. Kim, "Central object extraction for object-based image retrieval," presented at the CIVR, 2003.
- [25] R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. Patten Anal. Mach. Intell.*, 1994.
- [26] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," presented at the CVPR, 2007.
- [27] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Inf.*, 1974.
- [28] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Patten Anal. Mach. Intell.*, 2006.
- [29] P. Silapachote, J. Weinman, A. Hanson, R. Weissy, and M. A. Mattar, "Automatic sign detection and recognition in natural scenes," presented at the IEEE Workshop on Computer Vision Applications for the Visually Impaired, 2005.
- [30] S. Kumar and M. Hebert, "Discriminative random fields: A discriminative framework for contextual interaction in classification," presented at the ICCV, 2003.



Wen Wu (S'03) received the B.S. degree in computer science and technology from Tsinghua University, China, in 2001, and the M.Sc. degree in computer science from the National University of Singapore in 2003. He is currently pursuing the Ph.D. degree in language and information technologies at the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

His research experiences center around two themes: real-world applications and high-performance systems. He has researched several problems in learning, human-machine interface, and multimedia. His research domains range from traditional media (text, video, audio) to creative media and data (full-windshield display, social network).

Mr. Wu has published more than a dozen of papers in international journals and conferences such as the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the *ACM Transactions on Information Systems*, and ACM Multimedia Conference.



Jie Yang (M'91) received the Ph.D. degree from the University of Akron, Akron, OH, 1994.

He is currently on leave from Carnegie Mellon University (CMU), Pittsburgh, PA, and serves as a Program Director in the Division of Information Intelligence Systems at the National Science Foundation. He pioneered hidden Markov models for human performance modeling in his Ph.D. dissertation research. He joined the Interactive Systems Laboratories, CMU, in 1994, where he has been leading research efforts to develop visual tracking and recognition systems for multimodal human computer interaction. He developed adaptive skin color modeling techniques and demonstrated a software-based real-time face tracking system in 1995. He has been involved in the development of many multimodal systems in both intelligent working spaces and mobile platforms. His current research interests include multimodal interfaces, computer vision, and pattern recognition.