

SmartLabel: An Object Labeling Tool Using Iterated Harmonic Energy Minimization

Wen Wu and Jie Yang
School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
wenwu@cs.cmu.edu, jie.yang@cs.cmu.edu

ABSTRACT

Labeling objects in images is an essential prerequisite for many visual learning and recognition applications that depend on training data, such as image retrieval, object detection and recognition. Manually creating labels in images is not only time-consuming but also subject to human labeling errors, and eventually, becomes impossible for a large scale image database. Semi-supervised learning (SSL) algorithms such as Gaussian random field (GRF) can be applied to labeling objects in images since they have the ability to include a large amount of unlabeled data while requiring only a small amount of labeled data. However, the one-shot property of GRF prevents it from achieving good labeling performance. In this paper, we present a novel object labeling tool, SmartLabel, to semi-automatically label objects in images. The algorithm of SmartLabel has four innovations over GRF: 1) soft labeling, 2) graph construction with spatial constraints, 3) iterated harmonic energy minimization, and 4) using relevance feedback to incorporate human interaction in the loop. As demonstrated in datasets of six object categories, the proposed SmartLabel not only works effectively even with a very small amount of user input (e.g., 1 – 5% of image size) but also achieves significant improvement over GRF.

Categories and Subject Descriptors:

I.4.6 [Image Processing and Computer Vision]: Segmentation - *Region growing, partitioning*

I.2.6 [Artificial Intelligence]: Learning-parameter learning

General Terms: Algorithms.

Keywords: Semi-supervised learning, object labeling, Gaussian random field, harmonic energy minimization.

1. INTRODUCTION

The increasing growth of multimedia data on the Internet has created new challenges for the multimedia community. Many fields in multimedia, such as video retrieval and an-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'06, October 23–27, 2006, Santa Barbara, California, USA.

Copyright 2006 ACM 1-59593-447-2/06/0010 ...\$5.00.

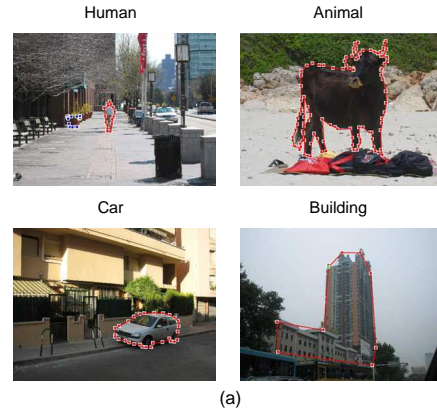


Figure 1: MIT LabelMe tool. (a) Four examples. (b) Statistics of user inputs from LabelMe website, x-axis: dates, and y-axis: accumulated number of labeled objects.

notation [17, 27], content based image retrieval [24, 23, 11, 26, 25, 22], object detection and recognition [15, 28], require the labeling and annotation of data. Manually labeling image data, for example, is not only a labor intensive and time demanding task, but also subject to human labeling and annotation errors and variances. While research has been focused on online massive user labeling (e.g., MIT LabelMe¹, The ESP Game²), little attention has been paid to semi-automatically labeling objects in images. The goal of this research is to address the problem of labeling objects in images by proposing semi-automatic labeling methods within a semi-supervised learning problem setting.

To get a sense of the difficulties in a task of manually labeling objects in images, let us consider LabelMe, the state of the art online annotation tool. LabelMe attempts to build a large collection of annotated images. The users simply

¹<http://people.csail.mit.edu/brussell/research/LabelMe/>

²<http://www.espgame.org/>



Figure 2: Examples of SmartLabel. The user loosely drags a rectangle inside an object. The object and others at different locations if any are then extracted automatically. Left is the input image with user specified ROI and right is extracted object region(s) using SmartLabel.

trace the outline of as many different objects from the image and for as many images as they like. Figure 1 (a) shows examples of manually labeled objects in images taken from the LabelMe database and (b) shows the statistics of labeled images. In 11 months, 26976 images were manually labeled among which 80234 objects were annotated. Although the impact of LabelMe for the research community is influential, this number of labeled objects and images is still fairly small compared to some large scale image databases. Therefore, manually labeling even larger image databases in a short period of time is virtually impossible.

These difficulties motivate us to turn back to semi-automatic methods to label objects in images with less required human effort. One solution is to ask a user to only mark a tiny region of interest (ROI) of the object on the image with some simple inputs (e.g., dragging a rectangle), and the algorithm can take care of the rest of the job by labeling the rest of the object or other similar objects in the entire image. Figure 2 shows examples of user inputs and extraction results from the proposed tool. Semi-supervised learning (SSL), which leverages the availability of a large amount of unlabeled data to enhance classification performance, has attracted a lot of attentions within the last few years and has been proved to be useful for many problems [10, 3, 18, 2, 20, 1, 19]. Some research has been done in semi-automatic object extraction using other techniques [29]. In this paper, we focus on addressing the problem within the framework of SSL.

To formulate the problem as an SSL task, we do the following. Given an image, an ROI is initially marked by the user. We then divide the image into non-overlapping blocks; the blocks that are in ROI or overlap with it are considered as labeled data, L , and the rest of the blocks in the image are considered as unlabeled data, U . Formulating the problem in this way allows us to apply any SSL algorithm to predict labels for U . However, this approach presents two challenges. First, the labeled data initially contain only positive examples because the user only specifies ROI of the object of the interest. Secondly, L from the divide-into-blocks process is not noise-free because blocks overlapping ROI can contain part of the background. These two challenges become critical hurdles to apply existing SSL algorithms to our problem because they normally assume that L contains both positive and negative data and is noise-free.

These challenges motivate us to come to a recently developed graph-based SSL method, known as Gaussian random field (GRF) [20]. GRF represents labeled and unlabeled data as vertexes in a weighed graph, with edge weights representing the similarity between connected vertexes. This method adopts Gaussian fields over a continuous state space rather than random fields over the discrete label set. The mean of the field is characterized in terms of harmonic func-

tions, and the solution can be efficiently obtained using matrix methods or belief propagation. This "relaxation" to a continuous rather than a discrete sample space results in many attractive properties. For instance, the lack of negative data and need of continuous label values to differentiate positive data can be naturally handled by GRF. Since the original GRF method was proposed for the problems in other domains, there are still some difficulties in applying it directly to our problem.

In this paper, we propose a novel iterative algorithm for object labeling in images. We call it SmartLabel. The SmartLabel algorithm has four novelties over GRF. 1) Real numbers ranging between 0 and 1 are used as the label value for labeled points. In other words, we label blocks in L using a continuous label value. 2) The weighed graph is constructed from the input image using two spatial constraints, a) separation of U into two subsets and b) spatial distance based edge weights, both constraints are for down weighing blocks spatially far from ROI. 3) The harmonic energy minimization is applied iteratively to estimate labels for blocks in U and newly labeled blocks are added to L for the next iteration. The algorithm runs a fixed number of iterations or stops until the graph structure is stable. 4) We bring the human in the loop through plugging in relevance feedback to alleviate the lack of negative data problem. Combining semi-supervised learning and relevance feedback offers an even more powerful tool to leverage both human knowledge and a large amount of unlabeled data. The experimental results indicate that SmartLabel works effectively even with a tiny user-specified ROI (e.g., 1 – 5% of image size). The overall performance can yield 84% macro-averaging F_1 over datasets of six object categories.

2. RELATION TO IMAGE SEGMENTATION

Labeling objects in images can also be called image segmentation, foreground extraction, object extraction or image editing, although they are slightly different in terms of applications and domains. Image segmentation has been an active area of research for decades and its application has been widely adopted in many research fields including content-based image retrieval [23, 11, 30]. The goal is to create systems capable of segmenting foreground objects from the background accurately and to achieve convincing segments of the image. Recently, research has focused on the problem of *interactive* extraction of a foreground object from an image [5].

There are three key differences between SmartLabel and other state of the art interactive segmentation tools. First, the goal of SmartLabel is to create labels for interesting object(s) in the image, not to segment the image into a

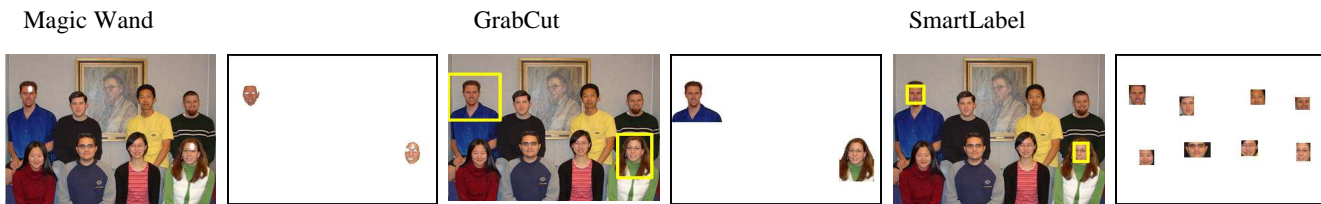


Figure 3: Comparison of three tools. The left image in each group shows user interactions required to complete object extraction: white brush (Magic Wand) and yellow rectangles (GrabCut and SmartLabel). The right image shows the resulting extraction. SmartLabel is able to extract faces at different locations in the image although only two initial locations are given. Note the difference between user inputs for tools.

number of blobs. Secondly, although SmartLabel is a semi-automatic labeling tool, it does not rely much on user input; only initial specification and relevance feedback after the 1st iteration are required. Finally, SmartLabel can extract a foreground object at multiple locations in the image even though the user only specifies part of the object at one location, while most iterative segmentation tools extract objects within or around the user marked ROI.

In the following, we describe and compare several state of the art interactive object extraction (segmentation) tools: Magic Wand, GrabCut [5], ClickRemoval [6], and central object extraction [8]. Figure 3 shows comparisons among Magic Wand, GrabCut, and SmartLabel since they are similar in terms of user interactions.

Magic Wand³ lets you select a consistently colored area (for example, a red flower) without having to trace its outline. You specify the color range, or tolerance, for the Magic Wand’s selection. While the user interface is straightforward, finding the appropriate tolerance level is often cumbersome and extracting the object containing multiple colored areas is problematic and requires a higher degree of user interaction.

ClickRemoval [6] 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. The main difference between this tool from SmartLabel is apparent: for images containing objects at multiple locations, extracting all of them is impossible, given that some of the objects that user does not initially pinpoint,. A considerable degree of user interaction is required to extract objects at multiple locations.

Central Object Extraction [8] is developed for object-based image retrieval. It does not require user interactions and automatically extracts central objects. However, it relies on two underlying assumptions: that interesting objects are generally located near the center of the image and contain regions with significant color distributions. Like the ClickRemoval, this tool lacks the ability to propagate extraction for the object at different locations.

GrabCut [5] is an interactive foreground extraction tool that extends the graph cut mechanism [3, 16]. While GrabCut and SmartLabel attempt to solve similar tasks, the two algorithms are very different in three respects. 1) *User initial inputs are different.* GrabCut requires the user to drag

a rectangle *outside* (around) the desire object, but SmartLabel requires an arbitrary close area (can be a rectangle) *within* an object. 2) *Extraction results are different.* SmartLabel attempts to extract object(s) outside of the ROI, which makes it possible to extract the object or similar objects at multiple locations in the image. In contrast, GrabCut extracts the desired object within the ROI and does not address the problem of multiple object extraction. Figure 3 shows the comparison. 3) *Goals are different.* GrabCut attempts to achieve fine and accurate segmentation by computing continuous alpha value over the entire inference region, while SmartLabel aims to obtain a good but not optimal object extraction. This can be useful in problems which do not require high accuracy of segmentation, e.g., reducing human efforts in a semi-automatic labeling tool, capturing the essence of the user query image(s) or feedback image(s) in CBIR.

3. OBJECT LABELING USING GAUSSIAN RANDOM FIELD

The Gaussian random field (GRF) algorithm [20] belongs to a class of graph-based SSL algorithms. In this section, we first describe standard SSL notations, and next, harmonic energy minimization, the procedure to estimate label predictions in GRF, is described in some detail.

3.1 Semi-Supervised Learning (SSL) Setting

SSL is a general 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 data points have labels $\{y_1, \dots, y_l\} \in \mathcal{C}$ and the remaining points 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 algorithms consider a connected graph, $G = (V, E)$, with nodes V corresponding to n data points, where nodes $L = \{1, \dots, l\}$ are the labeled points and nodes $U = \{l + 1, \dots, n\}$ are the unlabeled points. The edges, E , are weighted by the $n \times n$ affinity matrix, W , computed using certain distance metrics.

3.2 Gaussian Random Field

Recent work has successfully applied Gaussian random fields over a continuous state space and minimized the harmonic energy using matrix operations [20]. The goal of GRF is to first 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 $g(i) = g_l(i) \equiv y_i$, on

³Adobe Photoshop 7: <http://www.adobe.com/>

Input An image and user-specified region(s) of interest.

Initialize Divide the image into $B \times B$ blocks. Blocks in ROI are added in L and the rest are added in U .

Object Labeling using Gaussian Random Field

1. *Construct the weight matrix on L and U .* Form the weight matrix W using the weight factor defined in Section 3.2 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 $g_u = (D_{uu} - W_{uu})^{-1}W_{ul}g_l$.
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 resulting extractions.* Produce extracted regions of the desired object as blocks which are labeled as object.

Figure 4: Labeling Objects in Images using GRF.

L , $i = 1, \dots, l$. The aim of the Gaussian field configuration is to make unlabeled points that are nearby in the graph have similar labels. This motivates the choice of quadratic energy function, $E(g) = \frac{1}{2} \sum_{i,j} w_{ij}(g(i) - g(j))^2$. In the n -dim Euclidean space, $x \in \mathbb{R}^n$, the weight matrix, W , can be defined as, $w_{ij} = \exp\left(-\sum_{d=1}^n \frac{(x_{id} - x_{jd})^2}{2\sigma_d^2}\right)$, 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. Therefore, nearby data points in the Euclidean space are assigned large weights. The diagonal of the matrix W is set to be 1, $w_{ii} = 1$. Other weighting strategies are possible, and a spatial constrained weighting will be described in next section for our problem.

In order to assign a probability distribution on $g(\cdot)$, a Gaussian random field is formed as $p_\beta(g) = \frac{e^{-\beta E(g)}}{Z_\beta}$, where β is an "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 solution of functions $g(\cdot)$, we minimize the quadratic energy function $E(g)$ subject to the labeled data constraint.

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

in other words, it satisfies the equation, $\Delta g = 0$ on all unlabeled points in U and is subject to be equal to g_l on L . Here Δ is called the *combinatorial Laplacian*, and given in matrix form as

$$\Delta = D - W, \text{ where } D = \text{diag}(d_i), d_i = \sum_j w_{ij}. \quad (2)$$

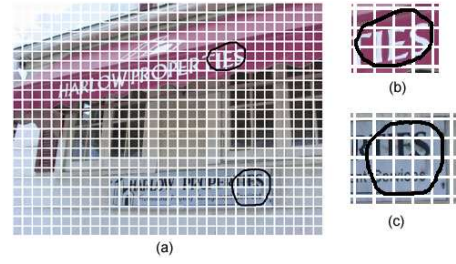


Figure 5: An illustration of necessity of soft labeling. After dividing the image into blocks, those two user specified ROIs (in black circles) only cover a few complete blocks, while other blocks on the ROI boundaries are only partially included, among which some are even completely background. Similar problem happens too when the user drags a rectangle of ROI.

The harmonic property indicates that the value of f at each unlabeled point is the average of f at its neighborhood points,

$$g(j) = \frac{1}{d_j} \sum_{i \in \mathcal{N}(j)} w_{ij}g(i), \text{ for } j = l + 1, \dots, l + u, \quad (3)$$

which maintains the smoothness constraint of $g(\cdot)$ with respect to the graph G . Expressed in the matrix form, Eq(4) can be rewritten as $g(\cdot) = Qg(\cdot)$, where $Q = D^{-1}W$. Because of the maximum principle of harmonic functions [14], the solution of g is unique and it is either a constant or satisfies $0 < g(j) < 1$ for $j \in U$. To compute the harmonic solution in terms of matrix operations, W is split into 4 blocks in terms of separation of L and U .

$$W_{n \times n} = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}_{n \times n}. \quad (4)$$

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 $g_u(\cdot)$.

$$g_u(\cdot) = (D_{uu} - W_{uu})^{-1}W_{ul}g_l(\cdot) = (I - Q_{uu})^{-1}Q_{ul}g_l(\cdot). \quad (5)$$

The intuition of harmonic energy minimization is that points nearest to each other in the graph have similar labels. The resulting classification algorithm for GRF can be viewed as a form of nearest neighbor approach, where the nearest labeled points are computed in terms of a random walk on the graph. In this paper we propose to formulate labeling objects in images as a semi-supervised problem and solve it by GRF. One benefit from the SSL setting is labeling performance can be independent of the number of object instances in the image since the underlying data similarity is embedded in W no matter how many times the object appears in the image. Figure 4 shows the GRF algorithm for object labeling, which is used as a baseline in our experiments. Furthermore, we extend GRF in several ways to improve the labeling performance.

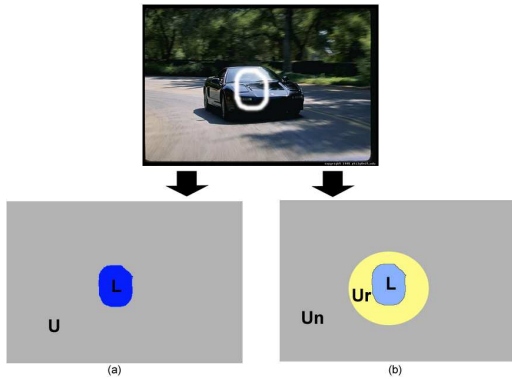


Figure 6: An illustration of formulating semi-supervised object extraction. (a) Apply the usual SSL setting by treating blocks inside the user specified region of interest as L and rest are in U . (b) SmartLabel splits U into two subsets, i.e., the relevant unlabeled set, U_r , and irrelevant unlabeled set, U_n , at each iteration.

4. SMARTLABEL

In this section, we present the novel parts of SmartLabel. We describe the novelties of the SmartLabel algorithm in four respects: soft labeling, graph construction with spatial constraints, iterated harmonic energy minimization, and bringing the human in the loop.

4.1 Soft Labeling

In order to formulate the object labeling problem in the SSL setting, we first need to construct L and U sets. Given an image and user-specified ROI(s), we divide the image into non-overlapping blocks of size $B \times B$ (in pixels). Each block is treated as a data point. Blocks that are in the ROI or overlap with it are considered as labeled blocks and added into the labeled set, L . The rest of the blocks are considered as unlabeled data and added to U . As noted before, the number of labeled data is l and the number of unlabeled data is u and $n = l + u$. Note L only contains the positive examples after this preprocess since the user only specifies ROI and not background.

Most SSL algorithms use the hard labeling strategy. Assume a label set $\mathcal{C} = \{1, \dots, c\}$, which indicates there are c classes, we have a labeling matrix $Y_{n \times c}$ in which $y_{ij} \in \{0, 1\}$. $y_{ij} = 1$ indicates the point x_i belongs to the class C_j , $y_{ij} = 0$ otherwise. In binary case like our task, we focus on only two classes, *object* and *non-object*. One problem rises with using hard labeling strategy because blocks in L are different: some are completely in ROI while some only partially overlap with ROI and can contain background as well. If we use uniform labeling $y_i = 1, i = 1, \dots, l$ for all blocks in L , the labeled data will be noisy because blocks which partially contain background are also labeled as *object*. Figure 5 shows an example where the user specifies two ROIs (in black circles) in the image. As we can see, these two ROIs only completely cover a few image blocks, while other blocks on the boundary are partially included, some of which can be complete background. One strategy to solve this problem is to give up all blocks which only partially overlap with ROI and just add them in U . But this is not a good idea because L becomes even smaller as a result. To alleviate the problem, we relax the possible label value to real numbers

Input An image and user-specified region(s) of interest.

Initialize Divide the image into the size of $B \times B$ (in pixels) block. Create the sets L and U using the soft labeling strategy in section 4.1. The number of iterations T . Neighborhood size H . $U_r = \emptyset, U_n = \emptyset$.

SmartLabel*

1. For $t = 1, 2, \dots, T$
 - a) *Update U_r & U_n .* Extract neighboring blocks from L in H pixels and put them into U_r , and the rest unlabeled blocks are put in U_n .
 - b) *Compute the matrix W' based on L and U_r .* Form the weight matrix W' using the weight factor defined in Equation (6) and $W'_{ii} = 1$.
 - c) *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' .
 - d) *Compute the harmonic solution.* Compute the label prediction on U_r using the Equation (8), $g_{u_r} = (D_{u_r, u_r} - W_{u_r, u_r})^{-1} W_{u_r, l} g_l$.
 - e) *Augment L .* Add newly predicted unlabeled blocks from U_r to L using the predictions g_{u_r} .
2. *Output resulting extractions.* Produce extracted regions of the user-desired object from blocks in L .

Figure 7: SmartLabel* (w/o. relevance feedback).

between 0 and 1 as $y_i \in [0, 1]$. We call it soft labeling. The label value for each block in L is computed as the ratio of its area within ROI to the block size. Since the mapping function $g(\cdot)$ in GRF is real-value defined, this "relaxation" appears to be naturally handled by it.

4.2 Graph with Spatial Constraints

Treating labeling objects in an image as an SSL problem holds some unique properties which are different from other SSL problems, e.g., document classification. One of the most important properties of our task is that location information is very important. Intuitively, pixels or blocks nearby tend to belong to the same object rather than those far away. Regions at nearby locations are more likely to contain similar color distribution than those that are far away. By observing this location dependent property, we next describe two spatial constraints which we incorporate into construction of the weighed graph: a location-dependent similarity measure and the separation of unlabeled set U into two subsets.

Motivated by the observation that nearby blocks tend to belong to the same object, we enrich the distance metric defined in Section 3.2 with an additional spatial distance term. The new distance measure is defined as

$$w_{ij} = \exp \left(- \sum_{d=1}^m \frac{(x_{id} - x_{jd})^2}{2\sigma_d^2} \cdot \frac{\sum_{k=1}^2 (x_i^k - x_j^k)^2}{\epsilon} \right), \quad (6)$$

where the first term represents the similarity distance between two feature vectors and the second term characterizes

Input & Initialization are the same as SmartLabel*.

SmartLabel

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

Figure 8: SmartLabel.

the distance between the centers of two blocks in the image. m is the number of dimensions of feature vector x_i , and $\sigma_1, \dots, \sigma_m$ are length scale hyperparameters for each dimension. x_i^1 and x_i^2 are the coordinates of the block, x_i , and ϵ is a normalization term. The new distance measure approximates the relevancy between two points by combining Euclidean distance in the feature space and the spatial distance in $2D$. Thus, nearby blocks measured by both the image feature and spatial distance are assigned a higher edge weight and they are close to each other in the graph.

In most SSL algorithms, unlabeled data in U are treated equally in learning, but this assumption may conflict with the location-dependent constraint mentioned above. To further embed location information in constructing the graph, we treat unlabeled blocks differently in terms of their image locations. In other words, we separate U into two subsets, U_r and U_n , based on user-specified ROI. U_r includes all blocks that are close to blocks in L and are to be included to build the graph. U_n includes all other unlabeled blocks that are left unconsidered currently. In more detail, blocks that are within the neighborhood of L in H pixels are put into U_r and the rest of the blocks in U are moved to U_n . Figure 6 shows an illustration of this setting. With this scheme, we incrementally estimate the labels on U_r without sacrificing the global configuration. The reason is, SmartLabel is an iterative algorithm, and the total number of iterations, T , guarantees all unlabeled blocks in the image will be included in the graph eventually. Nevertheless, the benefits from this separation of U are significant because it prevents the algorithm from adding many false positives in L at initial learning stages. The reason for the occurrence of false positives is that the number of labeled data in L initially is very small. Searching globally instead of locally using L is problematic because the target object concept is under-represented by L . By incrementally searching neighborhood regions of L and adding high-confidence predicted positive blocks in L iteratively, the labeling procedure can quickly search all unlabeled blocks in the image.

4.3 Iterated Harmonic Energy Minimization

The new iterated harmonic energy minimization in SmartLabel works in place of the previous one-shot minimization in GRF. As mentioned above, this advantage allows an automatic increment of the labeled set L at each iteration, with newly labeled blocks from U_r being added into L iteratively. Note that the definition of L is slightly different from the standard SSL setting because L is dynamically updated in SmartLabel while L is fixed in most SSL algorithms.

Figure 7 shows the main elements of the SmartLabel algorithm. In each iteration only points in U_r are included in the graph, and labels are predicted for each of them. The points which are predicted as *object* are added into set L . The algorithm runs a fixed number of iterations or until no more unlabeled points can be added into L . The algorithm then works in a label propagation manner, and the labeled set, L , is propagated out from initial input to rest of the image. False positives are restrained during the propagation by applying soft labeling and separation between U_r and U_n . The intrinsic idea for this iterative algorithm is to first predict labels on the unlabeled points of high relevancy in terms of both image feature and spatial distance, and then use them to continue predicting labels for points which are initially far away from the labeled data. In the following we analyze the time complexity of the new algorithm described above. Step (a) in the SmartLabel* algorithm takes $O(lu)$ time for update U_r & U_n computation. Step (b) takes $O((l + u_r)^2)$ time for computing the weight matrix of L and U_r . Step (c) takes $O(l + u_r)$ for Laplacian construction. Step (d) takes $O(u_r^2 + u_r l + l) = O(u_r(u_r + l))$ for computing a harmonic solution. And finally the augmenting of L takes $O(u_r)$ time. Thus, the time complexity, $C(T)$, of the algorithm in T iterations is $C(T) = O(T(lu + (l + u_r)^2 + l + u_r + u_r(u_r + l) + u_r))$. For images with small user specified ROI(s), where the label set size, l , is much smaller than the U_r size, u_r , the time complexity is simplified to $O(Tu_r^2)$. One way to speed up the algorithm computation is to compute weight matrix W for the whole image only once at the first iteration and shuffle rows and columns to create W' afterwards based on updated L and U_r sets.

In order to compute the harmonic solution g_{u_r} , we split the weight matrix W into *nine* blocks in terms of three sets (i.e., L , U_r and U_n), similarly for the matrix D .

$$W_{n \times n} = \begin{bmatrix} W_{l \times l} & W_{l \times u_r} & W_{l \times u_n} \\ W_{u_r \times l} & W_{u_r \times u_r} & W_{u_r \times u_n} \\ W_{u_n \times l} & W_{u_n \times u_r} & W_{u_n \times u_n} \end{bmatrix}. \quad (7)$$

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

$$g_{u_r}(\cdot)_{u_r \times c} = (D_{u_r \times u_r} - W_{u_r \times u_r})^{-1} \cdot W_{u_r \times l} \cdot g_l(\cdot)_{l \times c}. \quad (8)$$

If it is a multi-class labeling task ($C > 2$), one-against-all classifiers compete using $g_{u_r}^c(j)$, $c = 1, \dots, K$ as a posterior probability. In other words, an unlabeled data point j is labeled as follows. First, the algorithm computes the harmonic solution $g_{u_r}(\cdot)$, a $u_r \times c$ matrix and obtain the j -th row in the matrix which corresponds to the data point j . Secondly, the algorithm identifies the largest column value in the row vector and label the data point j accordingly: $g_{u_r}(j) = \operatorname{argmax}_{c \in C} g_{u_r}^c(j)$.

4.4 Relevance Feedback

Note that the user-specified ROI provides only positive data but lacks negative data, which can cause problems in our task. The harmonic energy minimization in the first iteration has the difficult task of discriminating *non-object* blocks that have similar appearance (colors and/or textures) of the *object* from true *object* blocks in U_r , since no negative data are available initially in L . The availability of nega-

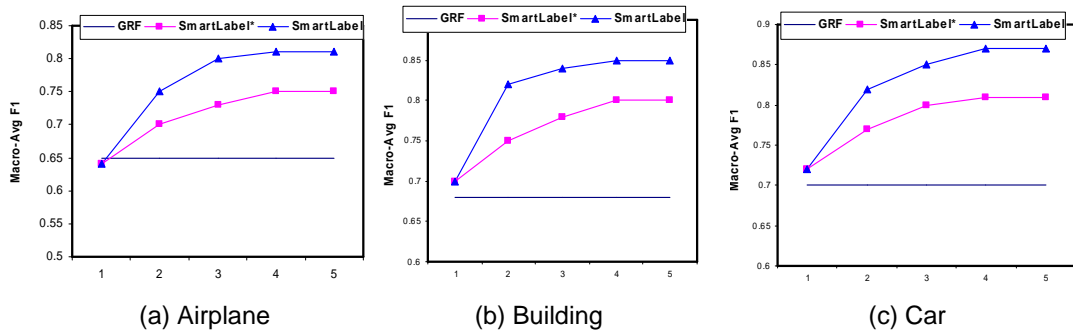


Figure 10: Comparing three algorithms. Performance curves on three object categories, i.e., (a) Airplane, (b) Building and (c) Car. Three algorithms are: baseline (GRF), SmartLabel* (w/o. relevance feedback) and SmartLabel. $B = 8$ for all and $H = 32$ for SmartLabel* and SmartLabel. Each curve plots macro-averaging F_1 against the number of iterations.



Figure 9: An example of image and its label.

tive data is crucial to reduce false positives. To alleviate the problem of lacking negative data, we bring the human in the loop by embedding relevance feedback (RF) in the algorithm. Note that the main focus is to study the performance gain from including negative labeled data in our task, so we apply RF only in the first iteration not in all iterations. This strategy is effective and can be exemplified in many other large scale applications where we want to minimize human supervision as much as possible when no (or few) negative data are available initially. 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.

We present a list of blocks that are labeled as the *object* by SmartLabel to the user, ranked by $g_u(\cdot)$. When the user labels a block as a negative example, its neighboring blocks in the graph including itself are added to L as negative data for later iterations of minimization. The same process is done to other user labeled negative blocks while the rest of the newly labeled blocks are confirmed as positive data. After the relevance feedback in the first iteration, the Gaussian field in the second and later iterations has more positive data and additional negative data to represent the target object concept discriminatively. The introduction of relevance feedback in SmartLabel makes it possible to learn the target *object* concept and *non-object* regions in the image simultaneously and also utilize the underlying graph structure from the unlabeled data to further improve labeling performance. Figure 8 shows the details of the algorithm.

5. EXPERIMENTS

5.1 Experimental Setting

To evaluate the performance of the proposed SmartLabel system, we collected images from several publicly available image datasets [9, 4, 7]. All the data were good-quality images and have ground truth which are the locations of objects in the images. Two sets of experiments were conducted to perform the evaluation, a *qualitative analysis* on real application with user input and feedback, and a *quantitative analysis* with simulated user input and feedback. Six objects were selected for the task, i.e., *Airplane*, *Animal*, *Building*, *Car*, *Flower* and *Text*. These objects cover a range of interesting topics in object labeling research. Figure. 9 shows an example image and its label from our database.

Two kinds of low-level image features were used in the experiments: color features and texture features. Each image was divided into non-overlapping blocks of size $B \times B$ (in pixels). We generate both types of features for each block. The color features consist of two parts. First, the first and second color moments for each RGB color channel were calculated. Then, we used 32 bins for each channel to extract color histograms. In total, we produced a 102-dimensional color feature vector for each block. The texture features were obtained from the convolution of the image block with various Gabor filters. In our implementation, 4 scales and 8 angles were used. The center and second-order moments were computed from each filter output. Thus, the texture feature was a 64-dim vector. After normalization, we concatenated color and texture features into a longer 166-dimensional vector. We adopted the Euclidean distance as the similarity measure between two data points. Hyperparameters, $\sigma_1, \dots, \sigma_m$ in Equation (6) were set as variance for each dimension. Other robust features, e.g., bag-of-word model, can also be used since the SmartLabel algorithm does not make any assumption about image features.

To quantitatively analyze the performance of SmartLabel, we simulated the input of user specified ROI by randomly selecting a 10% portion of the object area as the initial labeled data while the rest of the image was considered as the unlabeled data. After the selected unlabeled blocks were returned from the first iteration, we simulated user feedback by automatically labeling the first 5 false positives as negative examples based on the ground truth and added them to L . For each false positive, we extracted its 3 nearest

neighbors in the graph, and added them to L as well as negative examples if they are not in L . The macro-averaging F_1 score was used as the overall performance metric for each object category, which is defined as $F_1^M = \frac{2 \sum_{i=1}^K R_i \sum_{i=1}^K P_i}{K(\sum_{i=1}^K P_i + \sum_{i=1}^K R_i)}$, where P_i and R_i are the precision and recall computed on the resulting labeling for the i -th image and K is the total number of images in each object category. Macro-averaging metric treats every image equally, and calculates the global measures as the mean of all images’ local measures.

5.2 Performance Evaluation

We compared three semi-supervised labeling algorithms as follows. The Gaussian random field algorithm (**GRF**) simply uses the initial user-specified ROI as labeled data and makes one-shot labeling on all unlabeled blocks. **SmartLabel*** utilized iterative labeling prediction. The maximum iteration number is set as $T = 10$ while SmartLabel* normally converges within 5 or 6 iterations. To study the impact of block size on the labeling performance, we report results based on the setting as $B = 8$ or $B = 16$. Moreover, to examine the benefit from discriminating U_r and U_n , we report results based on the size of U_r , respectively as $H = 32$ and $H = 64$. Finally, in order to show the advantages of bringing the human in the loop, we also study **SmartLabel** performance.

Figure 10 depicts the labeling performance curves on three object categories, i.e., (a) Airplane, (b) Building and (c) Car, which in general represents the labeling patterns of general objects. Each subfigure contains the curves of three labeling algorithms, i.e., GRF, SmartLabel* and SmartLabel. Since GRF is a one-shot algorithm, its performance does not change in terms of iterations. In the first iteration, GRF’s result is slightly better than the SmartLabel results on the category "Airplane" while a little worse on the other two categories. We can observe that after the second iteration the SmartLabel algorithm consistently performs better than SmartLabel* across the categories, which is better than GRF. Also, with the number of iterations increasing, the performance of both SmartLabel* and SmartLabel monotonically increases and normally converges at Iteration 3 or 4. This observation corroborates the robustness of SmartLabel in various object labeling scenarios and the importance of adding relevance feedback in the system. To illustrate the quality of the SmartLabel algorithm’s performance, Figure 12 shows various object labeling results using SmartLabel. Images with a yellow rectangle are the input images with superimposed user-specified ROI and the corresponding labeling results are shown below each of them. As we can see, SmartLabel achieves very good labeling results across different object categories, although they are not perfect yet. Some post-labeling events can be introduced to further improve the performance. For example, using connected component analysis we can fill segments with holes inside. Other structure and context based methods can also be applied.

Figure 11 shows labeling curves in terms of the size of U_r on three categories, i.e., (a) Airplane, (b) Animal and (c) Flower. SmartLabel* is used as the labeling algorithm with $B = 16$. We compare two cases, i.e., $H = 32$ and $H = 64$. Each subplot shows the curves of F_1 values in terms of the number of iterations. The figure shows several interesting points. First, setting a big U_r size ($H = 64$) appears to obtain higher performance in both the first and final iterations than $H = 32$ for the category "Airplane". However, it

	GRF		S-Label*		S-Label	
	B=8	B=16	B=8	B=16	B=8	B=16
Airplane (128)	0.61	0.63	0.75	0.75	0.81	0.80
Animal (59)	0.53	0.60	0.76	0.79	0.78	0.83
Building (42)	0.63	0.68	0.80	0.73	0.85	0.81
Car (123)	0.51	0.69	0.81	0.79	0.86	0.84
Flower (37)	0.68	0.77	0.85	0.82	0.89	0.85
Text (57)	0.65	0.69	0.76	0.72	0.82	0.79
F_1^M	0.60	0.68	0.79	0.77	0.84	0.82

Table 1: Comparison on six object categories. Two different block size, 8×8 pixels ($B=8$) and 16×16 ($B=16$). The number in the first column is the # of images in each class. Results of **S-Label*(SmartLabel*)** and **S-Label(SmartLabel)** are reported in final iterations. The bold number shows higher performance between $B=8$ and $B=16$. The neighborhood size is set as $H = 32$.

is the opposite case for the category "Flower". To investigate the reasons, we examined the images in both categories and found that the object normally occupies a much bigger region of the image in "Airplane" dataset than in "Flower" dataset. This explains why a big U_r set gives higher labeling performance on the "Airplane" category because it makes U_r include more unlabeled regions in the graph at each iteration. Second, we can see that the performance actually monotonically decreases in terms of iterations on the category "Animal". We examined the images from this category and image results after the first iteration, and we found that most images in this category are wild nature images. Image background is complex and sometimes looks similar to animal skin in terms of color and texture. Also, some object regions are so small that labeling is already good enough after the first iteration. Thus, when continuing to the next iteration, falsely labeled background blocks are added to L which can degrade the performance.

To further examine the proposed algorithms, Table 1 lists a more comprehensive quantitative comparison for all six object categories. There are about a million blocks after dividing the images into 8×8 patches. Macro-averaging F_1 is reported for each category in two settings, $B = 8$ and $B = 16$. SmartLabel* consistently performs better than GRF for all six categories under the same block size setting. Overall, the macro-averaging F_1 increases 33%, from 0.60 to 0.79, for $B = 8$. By using relevance feedback, SmartLabel achieves even more improvement over SmartLabel*, increasing 6%, from 0.79 to 0.84. In order to show how block size affects the performance and how the algorithms gain from fine segments, we also show the performance of three algorithms when $B = 16$. As we can see, large blocks improve the performance of GRF. The reason is that segmenting the image into small blocks would create more blocks which merely contain background and thus brings noises into the labeled set and degrades the performance. In contrast, SmartLabel can much better handle the problem than GRF with its arsenal. SmartLabel usually runs for a couple of seconds processing an input image of 384×256 size in 16×16 blocks on a PC with 3.2GHz CPU and 2GB RAM.

One issue we did not explore in this paper due to lack of data is, to study SmartLabel’s performance on noisy images. That is, how the performance of the Smartlabel would change under poor image conditions (e.g., motion blur, salt

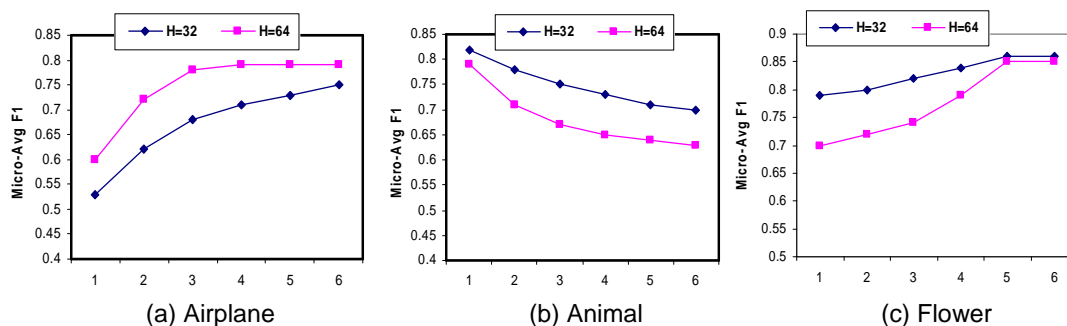


Figure 11: Impact of size of U_r on labeling performance. Comparisons on Airplane, Animal and Flower categories. Two settings: $H = 32$ and $H = 64$. $B = 16$. Each curve plots macro-averaging F_1 against the number of iterations.

and pepper) and a poorly defined ROI. This will be our future work.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a new object labeling tool called SmartLabel. Given an image with a small-size user specified ROI, SmartLabel can efficiently label the object at different locations in the image. Compared to the one-shot Gaussian random field (GRF) algorithm, the iterativeness and extensibility of SmartLabel makes it more suitable for problems of labeling objects in images and image retrieval applications. Our experiments based on six different object categories have demonstrated that SmartLabel not only significantly outperforms GRF in this task, but also has its unique practicabilities compared to other state of the art object extraction tools. Introducing relevance feedback in SmartLabel can further improve performance by bringing the human in the loop. SmartLabel does not make any assumption about the features used as most of semi-supervised learning algorithms, so we can imagine other types of features, e.g., bag-of-words model, can also be applied for this task. In the future, it may be interesting to apply SmartLabel into a content-based image retrieval system to capture the essence of query images or feedback images. Another interesting direction is to apply SmartLabel for object labeling in video, but we expect some twists needed in the algorithm in order to use it for video tasks because SmartLabel is originally proposed for an image object labeling problem.

7. ACKNOWLEDGEMENT

This work was partially supported by the General Motors & Carnegie Mellon University CRL, the NSF under grant No. IIS-0205219, and DARPA under contract DAAH01-02-C-R082. The authors would like to thank three anonymous reviewers for their helpful comments and suggestions and also acknowledge the assistance of Dr. Amarnath Gupta for shepherding our camera-ready version.

8. REFERENCES

- [1] M. Belkin and P. Niyogi, "Semi-Supervised Learning on Riemannian Manifolds," *Machine Learning*, 2004.
- [2] A. Blum and S. Chawla, "Learning from Labeled and Unlabeled Data using Graph Mincuts," *Proceedings of ICML*, 2001.
- [3] Y. Boykov *et al.*, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Trans. on PAMI*, 2001.
- [4] P. Silapachote *et al.*, "Automatic Sign Detection and Recognition in Natural Scenes," *IEEE Workshop on Computer Vision App. for the Visually Impaired*, 2005.
- [5] C. Rother *et al.*, "Interactive Foreground Extraction using Iterated Graph Cuts," *Proceedings of SIGGRAPH*, 2004.
- [6] F. Nielsen and R. Nock, "ClickRemoval: Interactive Pinpoint Image Object Removal," *In ACM Multimedia*, 2005.
- [7] S. Kumar and M. Hebert, "Discriminative Random Fields: A Discriminative Framework for Contextual Interation in Classification," *Proceedings of ICCV*, 2003.
- [8] S. Kim *et al.*, "Central Object Extraction for Object-Based Image Retrieval," *Proceedings of CIVR*, 2003.
- [9] L. Fei-Fei *et al.*, "One-Shot learning of object categories," *IEEE Trans. on PAMI*, 2006.
- [10] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. on PAMI*, 2000.
- [11] J. Z. Wang *et al.*, "SIMPLiCity: Semantics-Sensitive Integrated Matching for Picture Libraries," *IEEE Trans. on PAMI*, 2004.
- [12] J. Sun *et al.*, "Poisson Matting," *In SIGGRAPH*, 2004.
- [13] M. Szummer and T. Jaakkola, "Partially Labeled Classification with Markov Random Walks," *Proceedings of NIPS*, 2001.
- [14] P. Doyle and J. Snell, "Random walks and electric networks," *Mathematical Assoc. of America*, 1984.
- [15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of CVPR*, 2001.
- [16] G. D. Porteous and A. Seheult, "Exact MAP estimation for binary images," *J. Roy. Stat. Soc.*, 1989.
- [17] TRECVID: TREC Video Retrieval Evaluation, <http://www-nlpir.nist.gov/projects/trecvid>.
- [18] Y. Boykov and M.P. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," *Proceedings of ICCV*, 2001.
- [19] D. Zhou *et al.*, "Learning with Local and Global Consistency," *Proceedings of NIPS*, 2004.
- [20] X. Zhu *et al.*, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *Proceedings of ICML*, 2003.
- [21] D. Zhou *et al.*, "Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning," *Proceedings of ICML*, 2005.
- [22] Y. Chen *et al.*, "CLUE: Cluster-based retrieval of images by unsupervised learning," *IEEE Trans. on Image Proc.*, 2005.
- [23] A. W. M. Smeulders *et al.*, "Content-based image retrieval at the end of the early years," *IEEE Trans. on PAMI*, 2000.
- [24] Y. Rui and T. S. Huang, "A novel relevance feedback technique in image retrieval," *Proceedings of ACM Multimedia*, 1999.
- [25] T. Deselaers *et al.*, "FIRE - Flexible Image Retrieval Engine: ImageCLEF 2004 Evaluation," *CLEF 2004, LNCS 3491*, 2004.
- [26] F. Jing *et al.*, "Image retrieval I: A novel region-based image retrieval method using relevance feedback," *ACM Workshops on Multimedia: Multimedia Information Retrieval*, 2001.
- [27] T. Volkmer *et al.*, "A Webbased System for Collaborative Annotation of Large Image and Video Collections," *Proceedings of ACM Multimedia*, 2005.
- [28] L. Li *et al.*, "Foreground object detection from videos containing complex background," *ACM Multimedia*, 2003.
- [29] E. Saykol *et al.*, "A Semi-Automatic Object Extraction Tool for Querying In Multimedia," *Workshop on Multimedia Information Systems*, 2001.
- [30] J. Fan *et al.*, "Automatic image segmentation by integrating color-edge extraction and seeded region growing," *IEEE Trans. on Image Processing*, 2001.



Figure 12: Results using SmartLabel. The images with yellow rectangle are the input images with user specified ROI (region inside rectangle). The results obtained by SmartLabel are shown below each input image.