Markup SVG—An Online Content-Aware Image Abstraction and Annotation Tool

Edward Kim, *Student Member, IEEE*, Xiaolei Huang, *Member, IEEE*, and Gang Tan

***Abstract*—Suppose you want to effectively search through millions of images, train an algorithm to perform image and video object recognition, or research the complex patterns and relationships that exist in our visual world. A common and essential component for any of these tasks is a large annotated image dataset. However, obtaining labeled image data is a complex and tedious task that requires methods for annotating and structuring content. Therefore, we developed a comprehensive online tool and data structure, Markup SVG, that simplifies the collection of annotated image data by leveraging state-of-the-art image processing techniques. As the core data structure of our tool, we adopt scalable vector graphics (SVG), an extensible and versatile language built upon XML. Given the extensibility of our framework, we are able to encode low-level image features, high-level semantics, and further define interactions with the data to assist the user with image annotation. We also demonstrate the ability to merge multiple online and offline datasets into our system in an effort to standardize image collection and its data representation. Lastly, we present our modular design; each component acts as a plug-in to our system. We developed several novel components and algorithms to highlight the possibilities of semi-supervised segmentation and automatic annotation within our proposed framework. Further, our modular design provides the necessary capabilities to incorporate future image features, methods, or algorithms. Our results show that our tool is able to greatly simplify the process of obtaining large annotated image collections in an online collaborative platform.**

***Index Terms*—Data structures, image annotation, image processing, image representation, scalable vector graphics (SVG).**

I. Introduction

As a 19-month-old child points at a picture of a telephone, he exclaims, "ooh, teltone", simultaneously executing an object recognition and object labeling task [1]. Through learning and experience, these tasks become effortless for the infant's human visual system. Yet in computer vision, object detection and recognition remain remarkably difficult problems. Although challenging, we can foresee a computer system learning from multiple data sources and mimicking a high level of reasoning and image understanding. As shown in the literature, by collecting more labeled training data, we can increase the performance of supervised learning algorithms. But collecting large and diverse annotated datasets necessary for training these algorithms is a tedious and expensive task. Several web-based tools and social image sharing sites—such as LabelMe [2], Flickr [3], ALIPR [4], among others [5], [6]—have gained popularity and success by utilizing the collaboration potential of the online community for collecting annotation data. Aside from online systems, offline resources like the Lotus Hill [7] and Caltech 101 [8] datasets contain valuable, manually annotated data that could be used for image understanding or image retrieval tasks. However, despite these efforts, room for improvement exists. First, historically the systems have taken a top-down approach, where the intended application (e.g., collecting training data for object recognition, or text label data for image retrieval) determines what kind of annotation capabilities are supported. Unfortunately, we have found that this approach, although effective in collecting high quality data, can ultimately limit the utility of the data collected. For example, nearly all of the existing systems and storage formats we have encountered are uniquely designed and do not support the sharing of each other's data; see Fig. 1. Also, global tags in ALIPR and Flickr limit image search to text labels, with virtually no possibility of training object classifiers or performing content-based image retrieval. Second, while new and effective image processing and computer vision techniques are being developed, they remain isolated from these annotation efforts. For example, many effective general interactive segmentation methods exist [9]–[13], as well as medical image segmentation techniques [14]–[18] where interactivity helps physicians use their expert knowledge to obtain object boundaries. If some of these methods could be effectively coupled to online annotation systems, this could greatly improve the quality of segmentations and the user experience. However, currently most online systems involve manual annotation, without automated or semi-automated assist tools. A typical problem with these manual annotations, as seen in Fig. 1(c) and (d), is the use of straight line segments to estimate curved borders, resulting in slight approximation errors around the boundary.

To address the aforementioned problems in existing image annotation systems, we present a new image annotation tool, Markup SVG, which has the ability to incorporate heterogeneous data and leverage state-of-the-art image processing techniques to assist users in online annotation. The fundamental component of our online annotation tool is a structured information layer that we referred to as our image *abstraction*. Just as Bentley states, "data structures are frozen algorithms" [19],

Manuscript received September 17, 2010; revised March 11, 2011; accepted June 12, 2011. Date of publication July 05, 2011; date of current version September 16, 2011. This work was supported in part by a grant from the National Science Foundation under contract NSF IIS-0812120. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ajay Divakaran.

The authors are with the P.C. Rossin College of Engineering and Applied Science, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18105 USA (e-mail: edk208@lehigh.edu; xih206@lehigh.edu; gtan@cse.lehigh.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/TMM.2011.2161275

1520-9210/$26.00 © 2011 IEEE

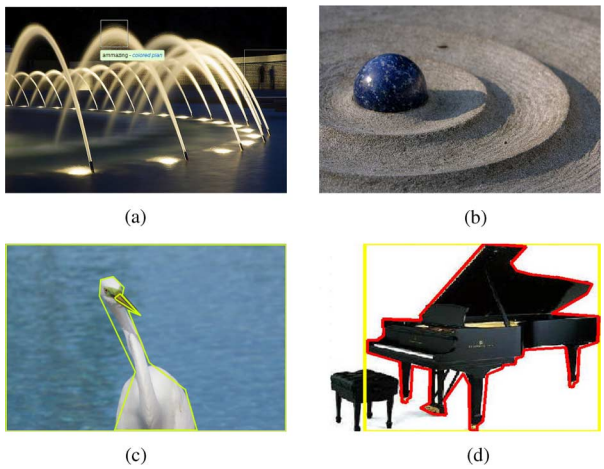


Fig. 1. Sample images from heterogeneous sources and their visualization. (a) Flickr has global tags and boxes. (b) ALIPR has global tags only. (c) LabelMe is online with manually delineated regions. (d) Caltech 101 has ground truth annotations through Matlab. Photo (a) courtesy of Brian Talbot. Photo (b) courtesy of Abra Kassandra.

our abstraction can be thought of as a frozen collection of image processing algorithms that can be visualized and manipulated in a standard web browser. For the underlying technology of our abstraction, we adopt scalable vector graphics (SVG). SVG is an expressive language that is built upon XML and benefits from the extensibility and versatility associated with the XML language. Thus, the abstraction supports the encoding of data from all existing sources such as LabelMe, Flickr, and Lotus Hill. Further, being a W3C (world wide web consortium) recommendation, SVG allows for interaction, visualization, and rendering in major web browsers, naturally supporting online image viewing and annotation.

Building upon the basic abstraction framework, we incorporate extensible *markup modules* that encode heterogeneous image information into symbolic form (numerical or textual) and define the interaction with such information. The motivation behind the modular approach is in our desire to design an abstraction that can represent image information at different levels of granularity. For example, how does one encode a representation of an object when it is unknown *a priori* how many components an object should consist of or even what the definition of an object should be? For different application purposes, we will get different answers to this question. Our modular approach allows our abstraction to encode different types of data, either provided by users or extracted from images using image processing techniques. Thus, we define four classes of markup modules: low-level image processing modules, higher level semantic modules, custom dataset modules, and action modules. First, the low-level image processing modules take a bottom-up technique towards feature collection. For instance, region-specific features (color, texture, etc.) can be extracted and represented numerically as either feature vectors or histograms [20], [21]. More complex features such as shape, Harris corners, SIFT [22], and Gist [23] have motivated the creation of bags of visual features [24], which can also be represented by our low-level module. Additionally, new features from future algorithms can be included by the addition of their corresponding markup modules.

Beyond the low-level information, integrating high-level semantic modules into existing image signatures raises fundamental questions on data structuring and organization. As noted by [25], a low-level descriptor in and of itself cannot encode high-level semantic information. Additionally, [26] states that a system is also needed to integrate content features, similarities methods, interaction with users, visualization of the image database, etc. We design high-level modules that have the ability to merge user annotations or encode high-level automated tasks (like parent-child inferences) into our abstraction. Furthermore, when integrating various datasets into our abstraction, low-level and high-level information may need to be represented simultaneously, e.g., LabelMe polygon with user annotation. The flexibility of our design naturally allows for these combinations, greatly simplifying the design of specific dataset modules.

Given the wealth of information encoded in our abstraction, we perceive our representation to be *content aware*. This awareness truly differentiates us from other online annotation tools. For example, in other tools, the main sources of confusion are what regions the user should label, how accurately the user should segment the object, and what labels the object should get [2]. However, in our tool, through the combination of different low-level, high-level, and dataset modules, we can design agents that have the ability to intelligently assist the user in the major aspects of image annotation, i.e., region segmentation and annotation. These intelligent, content-aware agents make up our final markup module, the action module.

Our innovation and contributions are twofold; first, we introduce an image abstraction—a novel content encoding data structure—and comprehensively demonstrate the capabilities of our abstraction towards supporting online image annotation. Our proposed abstraction seamlessly organizes image features and semantic information with methods to visualize and interact with the data. Additionally, the abstraction serves as a standard format that enables the collaborative annotation and sharing of online and offline datasets from various sources. Second, we develop novel methods based upon our abstraction to facilitate the segmentation and annotation of data in a web-based annotation tool. To our knowledge, our system is the first to incorporate emerging web technologies and image processing algorithms into an online annotation framework. Only through the combination of our tool and abstraction are we able to accomplish these complex assist methods online.

The remainder of this paper is organized as follows. In Section II, we introduce the data format of our image abstraction based on SVG, and the four classes of markup modules that we designed. In Section III, we describe our annotation tool and its functionalities in terms of visualization and interactivity. In Section IV, we present our results comparing our annotation tool to other online tools in terms of the region segmentation and annotation process on several datasets. Finally, we conclude with discussions and future directions in Section V.

II. Image Abstraction

We aim to design our content-aware image abstraction to be extensible, flexible, sharable, while being interactive and visu-

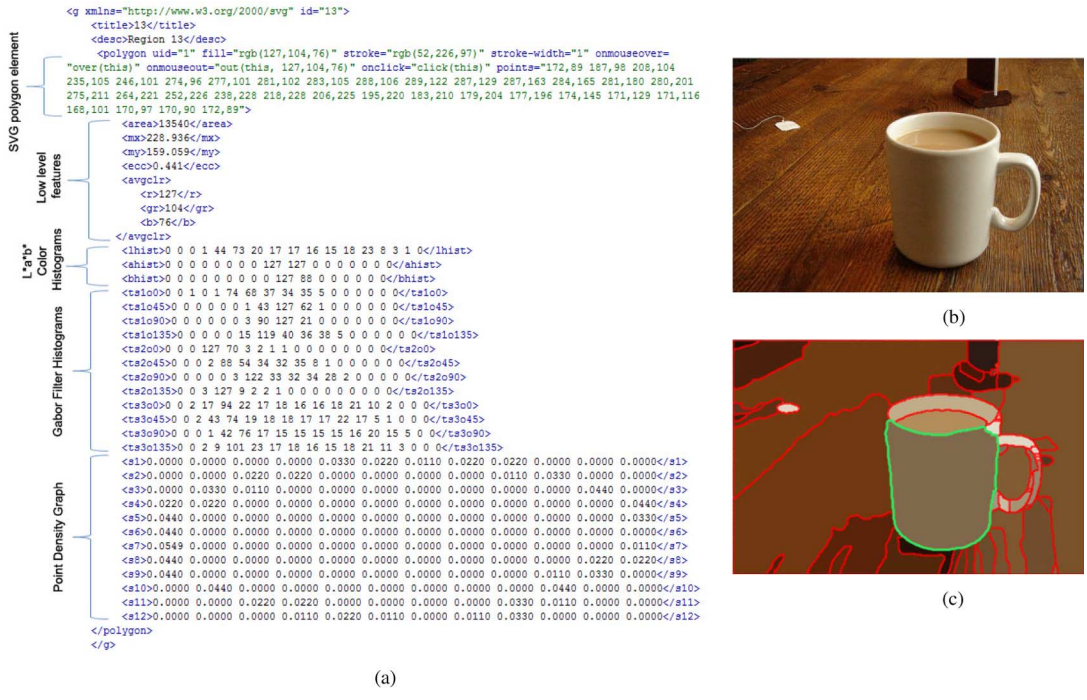


Fig. 2. Partial SVG document (a) corresponds to only the single region highlighted in green (c). The SVG abstraction stores both the visualization information to render the polygon in a browser as well as low-level information extracted from the raw image (b). (a) SVG encoding example. (b) Raw image. (c) gPb segmentation module with a highlighted polygon in green.

alizable online. Given the rich capabilities of SVG, we chose to develop our data structure using the SVG language. In this section, we first describe the basic principles of SVG, and then illustrate how we use SVG to construct our abstraction.

A. Scalable Vector Graphics (SVG)

The SVG standard [27] is an XML-based file format that describes two-dimensional vector graphic shapes, images, and text. Vector images and shapes are defined by mathematical instructions rather than traditional image formats based on individual pixels. In SVG, there are several basic shape objects, including lines, circles, paths, and polygons, which are modifiable by spatial transformations, alpha masks, and other effects. These shapes can also be customized in color, fill, texture, and stroke style. As SVG is the visualization of XML, our model is both renderable and extensible. Further, because SVG is a W3C recommendation, it allows for native web rendering and interactivity by scripting languages such as ECMAScript (Javascript, JScript). Robust XML query languages like XQuery, XPath, and FLWR can be used to efficiently search through the XML Document Object Model (DOM).

B. Markup Modules in Our Abstraction

We define our image abstraction in an SVG document that links to an image and supports the inclusion of various markup modules. We design these markup modules to be extensible plugins that interface with our system. Each module contains all the information necessary to visualize, interact, and utilize its functionality. Thus, future algorithms, datasets, etc. can be integrated into our system by designing a module, without ever having to modify our annotation tool.

We specify four classes of modules, low-level image processing modules, high-level semantic modules, heterogeneous dataset modules, and action modules. The modules are written in XML and contain the following properties/fields: $\{name, type, class, hook, Javascript, SVG\}$.

- The property field, "name", is the name of the module.
- The "type" field corresponds to the type of feature being extracted. For example in the low-level modules, we have developed several types including $\{\text{raw}, \text{edge}, \text{segmentation}, \text{GIST}, \text{SIFT}\}$, but this list can be appended to at any time.
- The "class" types include $\{\text{lowlevel}, \text{highlevel}, \text{dataset}, \text{action}\}$.
- The "hook" field is the name of the Javascript function that will be called upon loading the selected module.
- The "Javascript" field defines the interactivity that the module provides. For example, frequently there will be an onclick, onmouseover, or onmouseout event associated with the SVG elements. The functions declared in the Javascript field will support these actions.
- The "SVG" field describes how to structure and visualize the image features. The image features can be defined via polygons, circles, etc. with varying size, color, and texture.

Both the Javascript and SVG can be described inline, as a link to an external file, or as a reference to an external database table. The high-level semantic and the action modules also contain an additional field, *dependency*, since they may have dependencies on existing loaded modules in our system; see Fig. 4.

When loading a module, our SVG abstraction will create a new group element, $\langle g \rangle$. The group element is a container construct that associates elements together. Because the group el-

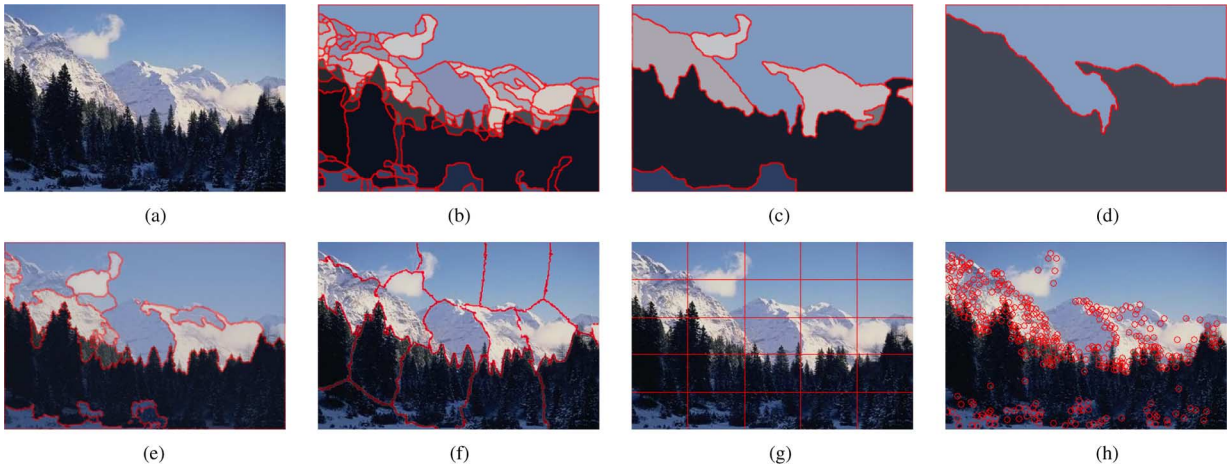


Fig. 3. Creation of the possible segmentations and features stored in our image abstraction's low-level modules for image (a). (b), (c), and (d) represent the varying granularities of segmentation where each region on each layer has associated color, texture, shape, etc. information encoded with it. (e), (f), and (g) display different image processing algorithms supported. (h) displays SIFT features that can be encoded into our abstraction. (a) Original. (b) gPb, 1st segmentation. (c) gPb, 2nd segmentation. (d) gPb, 3rd segmentation. (e) Mean Shift [28], $\mathrm{opacity} = 50\%$. (f) Normalized Cut [29]. (g) Concept Occur. Vector [30]. (h) SIFT [22].

ement can also contain other nested group elements to an arbitrary depth, we can utilize these nodes to hierarchically group different layers. Thus, loading a module involves the following steps: appending a $\langle \mathrm{g} \rangle$, group element, to the DOM with the *name*, *type*, *class* as attributes, then appending the *SVG* field contents to the group. Next, the annotation tool adds an eventlistener callback function for the *hook* and finally, dynamically loads the *Javascript* field contents into memory.

1) Low-Level Image Processing Module: Our low-level image processing techniques are completely automatic methods that extract image features from the raw image. We preprocess the images and store the result in low-level modules. The easiest way to describe these modules is through several illustrations. In the case of the *gPb-owt-ucm* [31] segmentation method (abbreviated *gPb*) [Fig. 3(b)–(d)], an example of our module properties could be $\{'gPb', 'segmentation', 'lowlevel', 'initgpb', gPb.js, gPb.svg\}$. Contained in the gPb SVG field, we store region boundary information as a point set in the SVG $\langle \mathrm{polygon} \rangle$ element. Next, we extend the basic instance of the $\langle \mathrm{polygon} \rangle$ to incorporate the region-specific features extracted from the raw image by adding elements to our polygon. In general, low-level features or MPEG-7 content descriptors extracted from the raw image can be represented here as an element in alphanumeric form. For example, the $\langle \mathrm{area} \rangle$, $\langle \mathrm{mx} \rangle$, $\langle \mathrm{my} \rangle$, elements would encode the area and centroid coordinates of the $\langle \mathrm{polygon} \rangle$ region. Additionally, for each region we collect $L^*a^*b^*$ color histogram, mean color, and responses of Gabor filters [32] of three scales and four orientations for a total of 12 texture histograms. From the shape information, we also encode eccentricity and indexable shape features. We explored Shape Context descriptors [33] and Curvature Scale Space [34] representations, but chose a modified Chord-based shape histogram [35] for faster indexing, and additional robustness on smaller regions with fewer polygon points. Our modifications use the principal eigenvector as our chord and compute the point density graph normalized around this chord. These features can be seen in our SVG example in Fig. 2(a).

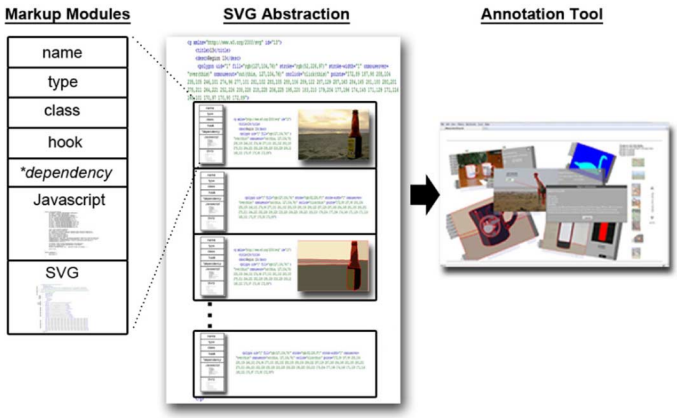
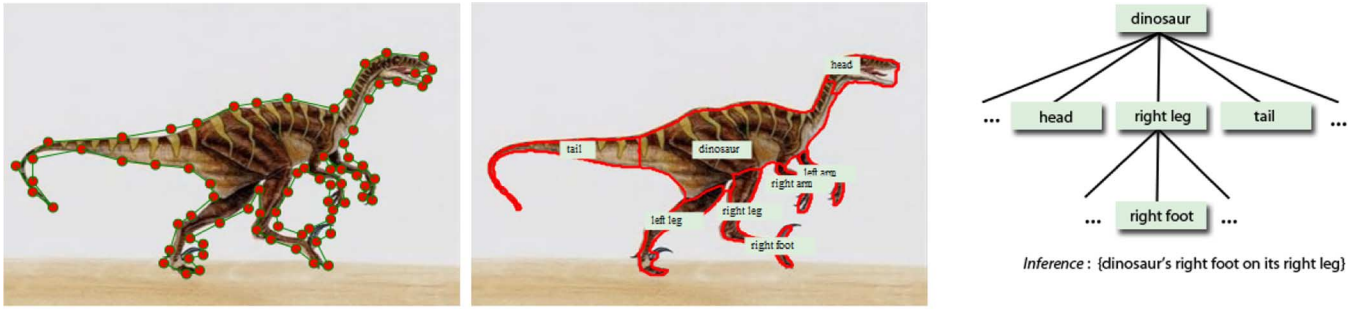


Fig. 4. Markup module properties. Any number of modules can be incorporated in our SVG abstraction. The $^*\mathrm{dependency}$ field is only for high-level or action modules.

For greater simplicity and because the SVG standard allows for the inclusion of foreign namespaces and private data, we chose to directly append to the XML.

Fig. 3 illustrates the results of different segmentation and feature extraction modules that can be stored in our model. In regards to segmentation, our flexible framework allows the encoding of not only several layers [Fig. 3(b)–(d)] of segmentation detail, but also of completely different segmentation techniques [28], [29] [Fig. 3(e) and (f)]. Alternatively, we could have chosen to encode edge information from Sobel or Prewitt convolutions, block segmentations as seen in Fig. 3(g) [30], or more advanced image features, such as SIFT features [22] [Fig. 3(h)]. This encoding stores SIFT histograms in the SVG, visualized by $\langle \mathrm{circle} \rangle$ elements.

2) High-Level Semantic Modules: When dealing with user input or semantic concepts, we employ our high-level semantic modules. These modules belong to the class, $\{\mathrm{highlevel}\}$, and have a new extendable set of types, $\{\mathrm{manual}, \mathrm{label}, \mathrm{inference}\}$. Additionally, this class includes a dependency field to ensure proper functionality.



(a) (b) (c)

Fig. 5. Several examples of high-level semantic modules that have been designed for our abstraction. In (a), the manual selection module functionality enables the clicking of points around the object. In (b), the region label module allows for the semantic annotation of predefined regions. In this case, the labels are added to a *gPb* segmentation. (a) Manual selection module. (b) Region label module. (c) Parent-child inference module.

As further explanation, we illustrate the design of several high-level semantic modules. For our first example, we design a module called the *manual selection* module, that can manually delineate regions within an image; see Fig. 5(a). The user clicks around an object, and the interaction and resulting region boundary is encoded as a polygon in the SVG document. This module is primarily Javascript centered, defining onclick methods and appending child elements to its SVG description.

A more complex example is the *region annotation* module that can be seen in Fig. 5(b). This is the first module we encounter that has a dependency requirement. For this module to work properly, it requires one of the *level:types* to be present, (highlevel:manual) or (lowlevel:segmentation). The reason for this dependency is because this module assigns labels to different regions, where the regions could be obtained from a low-level automated segmentation module or a high-level manual region selection module. For the labels, we encode an $\langle \mathrm{id} \rangle$, $\langle \mathrm{title} \rangle$, and $\langle \mathrm{desc} \rangle$ semantics (derived from the SVG group element).

- The $\langle \mathrm{title} \rangle$ element maintains a list of key words that can be associated with either human or machine classification labels for that region.
- The description $\langle \mathrm{desc} \rangle$ element is provided for a more natural language annotation or to convey other characteristics of a region not suitable for a keyword element.

Also defined in the module is the capability of composite region labeling. A user can create a composite region by merging several regions together (again by defining onclick functions). After merging, a new composite region (consisting of a list of its region components) is created and can be saved in the SVG document.

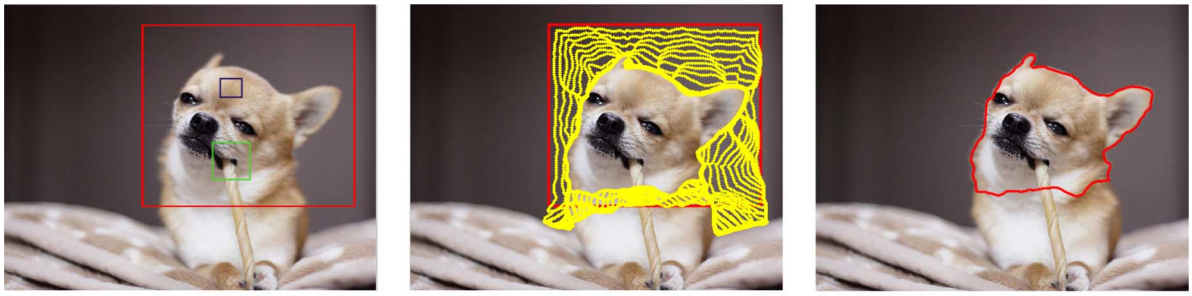
Building on the composite region idea further, we developed another high-level parent-child inference module that has the dependency of (highlevel:label), (lowlevel:segmentation) as seen in Fig. 5(c). From the component list of a merged region, R, e.g., the "dinosaur" region in Fig. 5(b), we notice the region is composed of the union of several, n, labeled regions, $L_{1...n}$ and m unlabeled regions, $U_{1...m}$ or

$$R = L_1 \cup \cdots L_n \cup U_1 \cup \cdots U_m. \tag{1}$$

These internal regions, L and U, can further be composed of the union of other, smaller segmentation regions. Thus, we can make parent/child inferences based on the components that comprise R. For example, in Fig. 5(c), an analysis of the right foot region returns its high- and low-level descriptor information, but also ascertains that this region, the *dinosaur's right foot*, is a child node on the parent node, the *dinosaur's right leg*. Reversing our logic and looking at the parent/child relationships in a top-down perspective, we can also infer a statistical model regarding the children of a *dinosaur*. Meaning, from a database with sufficient training data, our system can reason that a *dinosaur* is typically composed of a *head, body, tail, right arm, etc.*

3) Heterogeneous Dataset Modules: One of the key benefits of our system is the flexibility to immediately incorporate the results of other datasets and annotations. If we were to generalize annotation systems, the data are represented by a region/bounding box plus user tag. However, the representation of these results varies greatly, e.g., image bitmaps, XML definitions, and Matlab data structures, from system to system. On close examination of virtually all of the datasets we have encountered, we find that their inclusion can simply be defined as a mapping from heterogeneous data to two of our existing high-level modules, the *manual selection* module and *region annotation* module. The mapping function from these datasets to our abstraction modules is a preprocessing step that typically involves a Matlab CGI script or PHP XML parsing script. For example, we create an XML mapping for the LabelMe and Lotus Hill datasets to transform their XML region definitions (//region/pt/x,//region/pt/y) to our SVG polygon points. The XML names (//object/name) are mapped to the SVG $\langle \mathrm{title} \rangle$ elements. In the Caltech 101 dataset, we create a Matlab CGI script that maps the Matlab object data structure to our SVG polygon points. For the Flickr annotations, we create a PHP script that utilizes the Flickr API to pull XML bounding boxes and Flickr notes from the web and transform this data to SVG polygons and title elements. A visualization of some of our results can be seen later in our results section, Fig. 12.

4) Action Modules: Given the wealth of information that can be collected by our modules, it becomes evident that our abstraction has the property of content-awareness. Our abstraction no longer sees only individual pixels, but rather sees objects, edges, groupings, labels, etc. Utilizing this knowledge, we can design interesting content-aware action modules to assist in the anno-



(a) (b) (c)

Fig. 6. Example of our segmentation action module 1 using active contours on the bounding boxes obtained from Flickr. The active contour in this example uses the equal weight edge information from the gPb and Mean Shift segmentation results and assigns the manually obtained box contour zero weight. (a) Flickr bounding boxes. (b) Intermediate active contours. (c) Final segmentation result.

tation process. Specifically, we target the main tasks in the annotation process, namely, segmenting regions to annotate, and giving these labels relevant annotations.

Segmentation Assist Action Module 1: In the case where we are manually delineating boundaries or importing manual/bounding box data from other sources into our system, we often would like to correct the segmentation results to create a tighter more accurate boundary. We introduce a segmentation assist module based on an active contour model [11] to automate this task. The active contour model is a parametric curve, $\mathbf{x}(s) = [x(s), y(s)]$, $s \in [0, 1]$, that deforms to minimize its total energy

$$E = \int_0^1 \frac{1}{2}[\alpha|\mathbf{x}'(s)|^2 + \beta|\mathbf{x}''(s)|^2] + E_{ext}(\mathbf{x}(s))ds \quad (2)$$

where α and β are parameters that control the weighting of the active contour's internal energy terms related to curve tension and rigidity; E_{ext} defines the external energy of the contour derived from image information.

This action module has a dependency on either a (lowlevel:edge), or (lowlevel:segmentation) module. Content information from a low-level module, either edges or region boundaries, can be utilized in defining the active contour's external energy function E_{ext}. For example, if we had a gPb segmentation and a Sobel edge module, we could define our E_{ext} in a discrete formulation as

$$E_{ext} = -(w_1 \cdot d(M_{gPb}) + w_2 \cdot d(M_{sobel}) + w_3 \cdot d(M_{manual})) \quad (3)$$

where M_{method} represents the edge or object boundary described by the specified module (method), and $d(M_{method})$ represents the distance transform of this edge information. The M_{manual} edge information typically comes from the original manually delineated edges input by the user. This exists for situations where the user is able to infer a boundary where the low-level methods cannot. The weights, w, are user defined weights between 0 and 1 that specify the confidence the user has in the edge boundaries of that specific module. In LabelMe data, the manual delineation is probably close to an edge so the M_{manual} would have a nonzero weight; however, in Flickr data, the manual delineation is only a bounding box. Thus, the weight of M_{manual} should be set to zero. In general, we can define the E_{ext} as

$$E_{ext} = -\sum_{i=1}^{n} w_i \cdot d(M_i) \quad (4)$$

where n is the total number of supported contour modules. Fig. 6 demonstrates our segmentation assist module on a Flickr annotation bounding box. Similarly, in an interactive setting, the user can delineate a rough segmentation by manually clicking a sparse set of points around the boundary. This initial process is similar to the manual segmentation; however, upon closing this rough boundary, our active contour method will evolve the boundary to fit the edges in the image.

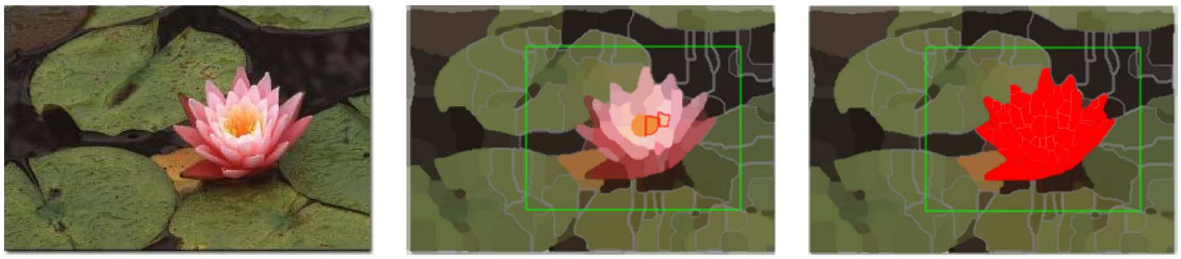
Segmentation Assist Action Module 2: We develop a complementary segmentation assist module based upon an interactive conditional random field (CRF) [36] Graph Cuts [10], [15] method. Particularly, we are interested in developing our algorithm on top of an automatic segmentation method that partitions an image into small coherent groups, e.g., superpixels [37], [38]. Therefore, this module is dependent upon the existence of a (lowlevel:segmentation) module. Consider the set of superpixels, $\mathcal{S}$, and a neighborhood system, $\mathcal{N}$, of unordered neighboring pairs $\{s, q\}$. We map our superpixels to an undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ are the graph nodes corresponding to our $s \in \mathcal{S}$, and $\mathcal{E}$ are the undirected edges that connect these nodes, and we add two more terminal nodes, S, T, the source (foreground) and sink (background). Let l_s be the binary label $\{1, 0\}$, assigned to a superpixel s in $\mathcal{S}$, where $l_s = 1$ indicates the superpixel belongs to the foreground and $l_s = 0$ indicates the superpixel belongs to the background. Then, let $L = l_1, l_2, \ldots l_{\mathcal{S}}$ be the binary vector that assigns a label to all superpixels in $\mathcal{S}$. We aim to minimize the energy function

$$E(L) = \sum_{s \in \mathcal{S}} R_s(l_s) + \sum_{\{s,q\} \in \mathcal{N}} B_{\{s,q\}} \cdot \delta(l_s, l_q) \quad (5)$$

where

$$\delta(l_s, l_q) = \begin{cases} 1, & \text{if } l_s \neq l_q \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where $R_s(l_s)$ represents the region properties, or the costs associated with labeling a superpixel to the foreground or background label l_s, and $B_{\{s,q\}}$ represents the boundary proper-



(a) (b) (c)

Fig. 7. Example of our segmentation assist action module 2 using superpixels on a Caltech 101 image. In figure (b), the green bounding box and foreground seeds outlined in red are selected by the user. The superpixels outlined in gray intersect the bounding box, creating our background seeds. The final region of interest (ROI) highlighted in red in (c) is obtained from the min-cut between foreground and background. (a) Caltech 101 image. (b) Superpixel gPb segmentation. (c) Final segmentation ROI.

ties, or costs associated with assigning differing labels to neighboring superpixels, $\{s, q\}$.

Our formulation resembles [15] and [37], but additionally includes a novel pairwise edge term, $B_{\{s,q\}} \cdot \delta(l_s, l_q)$. But in order to use this term, we need to know a region's neighbors and shared border length between neighbors. The neighboring regions and border lengths can be obtained by computing the gray level co-occurrence matrix (GLCM) [39]. Although typically used for texture analysis, the GLCM with a symmetric offset of 1 pixel in all directions on the labeled connected components of our segmentation efficiently computes superpixel neighbors. The indices of the nonzero entries in the matrix (off the diagonal) identify the neighbors, and the values of these entries specify the border length. As our abstraction is easily extendable, we are able to store these additional features (neighboring regions and border length) into our encoding. The superpixel regions are uniquely numbered and a neighbor list is generated and stored within the $\langle \text{neighbors} \rangle$ element. This list defines $\mathcal{N}$.

Previously included with our region segmentation abstraction are the $L^*a^*b^*$ color histograms whose distance between regions s and q can be computed via the χ^2 measure

$$\chi^2(s,q) = \frac{1}{2}\sum_{k=1}^{K} \frac{[h_s(k) - h_q(k)]^2}{h_s(k) + h_q(k)} \tag{7}$$

where h_s and h_q represent the histograms of s and q, respectively, and K is the number of bins in the histogram. With our neighborhood list and region distance measure, we can compute $B_{\{s,q\}}$ based upon a CRF formulation

$$B_{\{s,q\}} = exp\left(\frac{-\sum_{n=1}^{3}\chi_n^2(s,q)}{2\sigma^2}\right) \cdot \frac{\lambda_{\{s,q\}}}{dist(s,q)} \tag{8}$$

where n represents the different color channels, $\lambda_{\{s,q\}}$ represents the neighbor border length, and $dist(s,q)$ is the Euclidean distance between the centroid points of the regions. The exponential term encourages region coherency while the $\lambda_{\{s,q\}}/dist(s,q)$ term acts as a regularizing component, penalizing isolated regions and superpixel neighbors whose centroids are large distances from each other. The effect of this term is illustrated in Fig. 8. The σ term is a user-defined parameter that we set to 1/2.

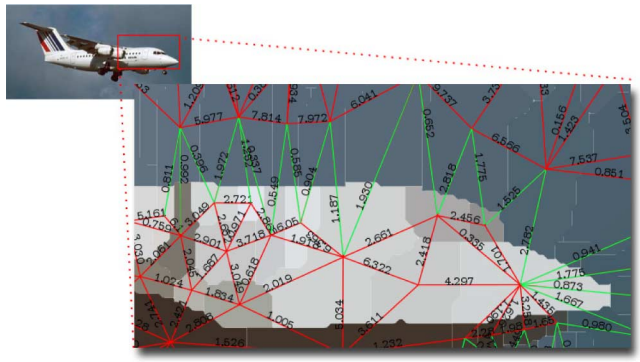


Fig. 8. Superpixel graph displaying the weights of our regularizing term, $\lambda_{\{s,q\}}/dist(s,q)$, visualized in SVG. The red lines indicate superpixel neighbors that belong to the same class whereas the green lines indicate neighbors that belong to different classes. In this example for the airplane class, our regularizing term maintained an average intra-class weight of 2.51, while inter-class links between the airplane and sky had an average weight of 1.32.

As previously mentioned, our region term, $R_s(l_s)$, represents the cost of assigning label l_s to superpixel s. For each superpixel, we encode its bounding box information in our SVG abstraction. Then, the user indicates a region of interest on the image by interactively placing an SVG rectangle bounding box. By computing the bounding box intersection between the user rectangle and superpixel bounding boxes, we can discriminate between the superpixels that lie outside the user's rectangle, intersect the rectangle, and are fully enclosed by the rectangle. The goal of our segmentation method is to assign a label, l_s, to all superpixels fully enclosed within the user-provided rectangle that minimizes the total energy. To initialize our algorithm, we use the superpixels that intersect the user rectangle as our background seeds and use several of the fully enclosed superpixels interactively selected by the user as our foreground seeds. In the beginning, all fully enclosed nodes have equal weight to S and T, but after our initialization procedure, we set our $R_s(l_s) = (\infty, 0)$ to (S,T) if the superpixel belongs to the source or $(0, \infty)$ if the superpixel is assigned to the sink. From this formulation, the minimum energy cut is computed by a max-flow/min-cut algorithm [15] as shown in Fig. 7.

Annotation Assist Action Module: With training through existing annotated regions in our database, our system has the ability to suggest annotations to the user with our annotation

assist action module. This module has the dependencies of (lowlevel:segmentation), or (highlevel:label).

Given the low-level features collected by a segmentation module like color, shape, texture, size, and location (full list described in Section II-BI), we can estimate a probability that a given region, r_s, is assigned to a certain classification label. If we consider m low-level features in a vector $\mathcal{P} = \{p_1, p_2 \ldots, p_m\}$, and have a possible of $\mathcal{K}$ classes to choose from, where the classes are defined as $\mathcal{C} = c_1, c_2, \ldots, c_{\mathcal{K}}$, we can compute a likelihood or confidence score of having classification c by

$$S(c|\mathcal{P}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{f(\mathcal{P}, \hat{\mathcal{P}}_i)}. \tag{9}$$

Here, N is the number of exemplar regions in category, c, and $\hat{\mathcal{P}}_i$ are the low-level features in the exemplar region, i. The function, $f(\mathcal{P}, \hat{\mathcal{P}}_i)$ computes the distance between the m low-level feature vectors, $\mathcal{P}$ and $\hat{\mathcal{P}}_i$. This function will vary according to the specific feature being evaluated. For example, when evaluating the difference between two histograms like color or texture histograms, the difference is computed by a χ^2 measurement. If the pixel feature being computed is a scalar value (eccentricity, image moment, etc.), the distance measurement is a $\mathcal{L}_1$ difference. All distance measurements derived from features in the feature vector are normalized so that they contribute equally to the total distance, $f(\mathcal{P}, \hat{\mathcal{P}}_i)$. At smaller distances, there is a higher similarity between the two regions and thus a higher probability of them belonging to a certain class.

Furthermore, we can improve on (9) if we leverage more information to estimate the class probability of a region by including data from our high-level semantic label module. The majority of our annotations occur in an image where some regions have already been labeled. Thus, when considering a new region to be annotated, we can exercise the existing $\mathcal{H}$ annotated regions, $\mathcal{A} = \{a_1, a_2, \ldots, a_{\mathcal{H}}\}$, in a novel Bayesian formulation, and estimate the probability of each class given the new region's low-level features $\mathcal{P}$, and information about the annotated regions $\mathcal{A}$

$$p(c|\mathcal{P}, \mathcal{A}) \propto p(\mathcal{P}|c)p(\mathcal{A}|c)p(c). \tag{10}$$

We can rearrange the term, $p(\mathcal{P}|c)$, and obtain

$$p(c|\mathcal{P}, \mathcal{A}) \propto p(c|\mathcal{P})p(\mathcal{P})p(\mathcal{A}|c). \tag{11}$$

We can drop the $p(\mathcal{P})$ term to obtain our final probability equation

$$p(c|\mathcal{P}, \mathcal{A}) \propto p(c|\mathcal{P})p(\mathcal{A}|c). \tag{12}$$

We can compute $p(c|\mathcal{P})$ by utilizing our above score equation defined in (9)

$$p(c|\mathcal{P}) = \frac{S(c|\mathcal{P})}{\sum_{n=1}^{\mathcal{K}} S(c_n|\mathcal{P})}. \tag{13}$$

The probability, $p(\mathcal{A}|c)$, is

$$p(\mathcal{A}|c) = \frac{p(\mathcal{A}, c)}{p(c)} \tag{14}$$

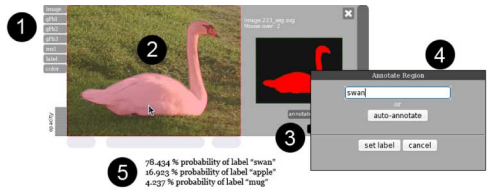


Fig. 9. Illustration of our automatic annotation module. In step 1, the user chooses a segmentation module or manual module to visualize. In step 2, the user can select a region or combination of regions to annotate. In step 3, the user can preview the selected region and elect to annotate that region. In step 4, the user can either provide a label or ask our annotation assist module to compute the most probable label for that region. In step 5, we overlay the top three probabilities calculated for the region and suggest the maximum likelihood label, "swan".

where the joint probability, $p(\mathcal{A}, c)$, can be computed via a search through our dataset for the number of co-occurrences. We then choose

$$\hat{c} = \arg\max_{c_n} p(c_n|\mathcal{P}, \mathcal{A}) \tag{15}$$

as our final region classification/annotation label, to be suggested to the user. In Fig. 9, we illustrate how and where this module can be used in our annotation tool.

III. Annotation Tool Overview

In this section, we describe our online annotation tool and the functionality provided by our system in terms of visualization and interactivity.

A. Visualization

Because of the visualization support of SVG in all of the major web browsers (Firefox, Safari, Chrome, IE 9), our annotation system is built as a web-based tool using standard web technologies such as HTML, CSS, AJAX, and PHP. The visualization of the different SVG elements is handled adeptly by the SVG elements and their associated attributes. As an example, the SVG elements can be controlled through different color and opacity effects as shown in Fig. 11. The user interface of our tool is shown in Fig. 10, which provides an area for displaying an image, positioning and scaling controls, opacity and color controls, and an information window. Using transformation matrices, any element (e.g., an image, polygon, line, etc.) can be translated, rotated, or scaled as shown in our UI. Loading our markup modules to the annotation tool enables additional functionality. When enabled for a certain image, the modules appear on the left-hand side of an image. To the right of the image exists the information panel. This panel is controlled by the specific Javascript definitions contained in the loaded markup module. Our abstraction provides access to this panel, enabling the markup modules to display text, or even append elements such as buttons or interactive elements.

B. Interactivity

Similar to the visualization effects, interactivity is added using Javascript and manipulating the XML DOM. The basic interactivity provided by the annotation tool is simply an interface to the modules and datasets. For image modules, we

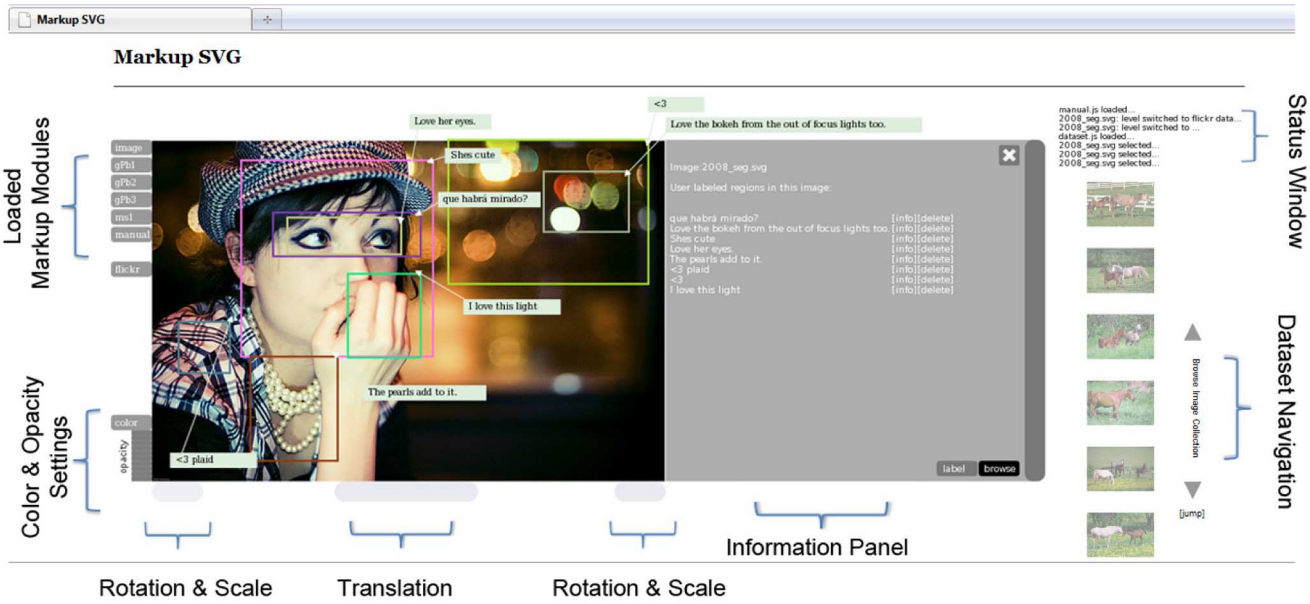


Fig. 10. Annotation tool user interface in the Firefox 3.0 web browser for various modules and datasets. The image shown utilizes the Flickr dataset module and displays the bounding boxes on the image and user provided tags in the information window. Our interface allows for the scaling/rotation of an image, interaction with loaded markup modules, and navigation through various datasets. Photo courtesy of Jon Clay.

provide access to the transformation matrix, and since SVG consists of vector objects, scaling or rotating these objects does not result in pixelation artifacts. Modifying the DOM on the fly, we can easily implement control boxes/tabs around an image frame to allow dragging, rotating, zooming, switching to different views, and so on. For our datasets, our tool provides easy access and navigation via thumbnails (scaled SVG images) with page-up and page-down controls; see Fig. 10.

When interfacing with an image abstraction in an SVG document, our annotation tool reads the "Javascript" field of the loaded modules. This allows a module to define interactions associated with that module, independent of the functionality of our tool. This flexibility enables new functions through our modular design.

IV. Results

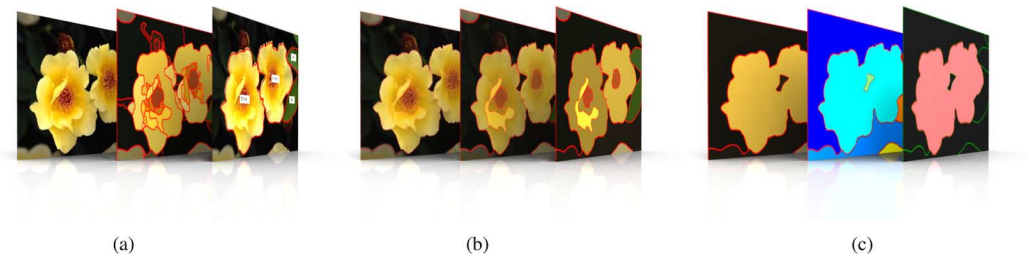
In our experiments, we evaluated how effectively our tool could be used to collect annotation data from the general population. We were interested in several factors, including how easy is our tool to use, how satisfied are the users with their segmentation result, how much training is necessary to utilize the different methods available in our tool, and does our tool more efficiently collect data over the current state-of-the-art. For these experiments, we imported annotation data from Flickr, LabelMe, Lotus Hill, and the Caltech 101 datasets and created their SVG abstractions; see Fig. 12. We also created SVG abstractions and collected annotations for two previously unannotated datasets, ETHZ Shape [40] and Corel 1K (WANG database).

For all datasets, we encode four low-level automatic segmentation modules—three levels of detail from *gPb* and one from Mean Shift. We enable the high-level manual region selection module and the region label annotation module. We also enable two action modules, the superpixel segmentation assist and annotation assist. Additionally, the active contour assist module is enabled in our first segmentation user study. The abstraction's modules linked their SVG documents to a MySQL database for fast indexing and storage, and the SVG abstraction is parsed via XPath and XQuery.

Next, we present results from various region segmentation and annotation experiments.

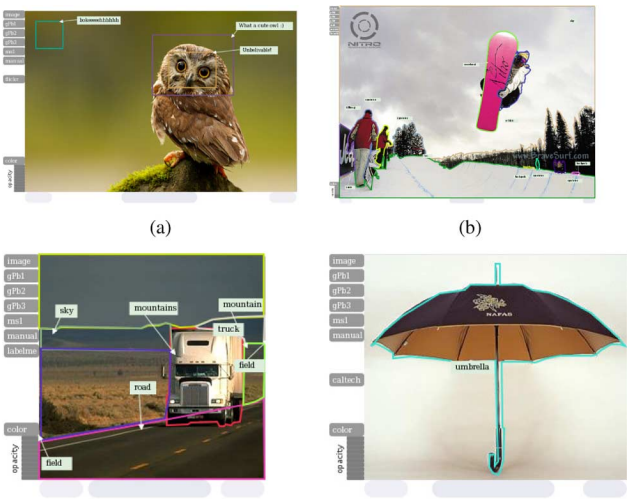
A. Region Segmentation User Study 1

In our first segmentation study, we utilized the crowdsourcing tool, Amazon Mechanical Turk, to evaluate four of our segmentation methods. The four methods were a basic manual segmentation method, an active contour segmentation module (assist action module 1), a region annotation module with composite region grouping capabilities (described in Section II-BII), and a superpixel segmentation module (assist action module 2). We were interested in how long it would take for a worker to be trained on how to use a particular segmentation method. We assigned each segmentation method its own task, and for each of the tasks, we asked 30 unique workers to read instructions on how to use our tool. The instructions consisted of 3–5 steps with illustrative examples. To ensure the quality of the user response, at the end, the user is required to use our tool to segment a specific image that could only be segmented properly if the worker fully comprehended how to use the segmentation method. We collected data from 62 unique workers across all 120 tasks (an average of 1.94 tasks per worker) and recorded their responses to three questions. First, how long did it take the user to read and fully understand the instructions on how to use our tool. Second, on a scale of 1–7 ($1 = \text{very disappointed}, 7 = \text{very satisfied}$), the user rated how satisfied they were with the final object outline. Third, on a scale of 1–7 ($1 = \text{very difficult to use}, 7 = \text{very easy to use}$), the user rated how easy it was to use our



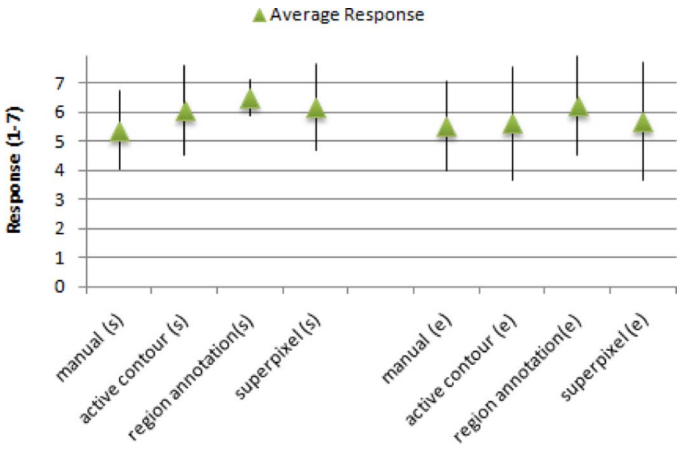
(a) (b) (c)

Fig. 11. Different visualizations encoded in the SVG abstraction. In (a), we demonstrate the raw image to the high-level semantics. In (b), we illustrate different opacity levels. In (c), we alter the polygon fill information (a) Raw, segmentation, semantic module. (b) 25%, 60%, 100% opacity. (c) Average, Jet, ROI color.



(a) (b) (c) (d)

Fig. 12. Resulting encodings of a (a) Flickr image, (b) Lotus Hill boundary, (c) LabelMe polygon, and (d) Caltech 101 ground truth, into our annotation tool using our dataset modules. Photo (a) courtesy of Jason Idzerda.



(a) Amazon Mechanical Turk Results.

Fig. 13. Satisfaction (s) and Ease of use (e) responses for each of our segmentation methods obtained through Amazon Mechanical Turk. The graph shows the average response as well as the standard deviation across 30 unique users for each tool.

tool to obtain the object outline. Workers also had the option of leaving general comments about our tool.

As expected, the training time for simple methods like the manual annotation method took the least amount of time. On average, it took 1.89 min for users to fully understand how to use manual annotation methods. However, surprisingly, our other more complex methods did not require much more training. The active contour module took 2.35 min, the region annotation module with composite region grouping took 2.05 min, and the superpixel module took 2.34 min. In terms of computational time for each method, naturally the manual method has no computational load, the active contour method takes approximately 1–2 s to compute 30 deformation iterations, and the superpixel method takes less than 1 s to perform the maxflow/mincut operation. The region annotation method has no associated computation time, but does require an automatic preprocessing segmentation. We describe how to alleviate some of the preprocessing computations in real-time upload situations in Section IV-D.

In terms of satisfaction and ease of use with each method, we present the mean and standard deviation in Fig. 13. From our results, the manual method is least favored both in satisfaction with the end outline and ease of use. Generally, users preferred our region annotation module over all other methods; however, we note that this module is dependent on the correctness of the low-level segmentation results, and does not have the flexibility of our other assist modules. Also, the feedback we obtained through general comments were overwhelmingly positive, further validating the usefulness of our segmentation methods.

B. Region Segmentation User Study 2

In our second user study, we performed an in-depth experiment measuring the usability of our system versus the state-of-the-art LabelMe system in terms of clicks, time, segmentation accuracy satisfaction, and ease of use. For this experiment, we randomly chose 75 images from the Caltech 101 dataset from 15 categories, including dolphins, pianos, laptops, staplers, etc. We recruited 11 users (6 females, 5 males) of varying educational backgrounds, none of whom had any experience with computer vision algorithms, nor had any experience with either our tool or the LabelMe tool. First, we trained them on our interface as well as LabelMe's interface. Then, we compared the total number of clicks it took the users to segment the object of interest from the image and report them in Fig. 14(a). For our system, Markup SVG, and LabelMe, we report the average number of clicks across all subjects. We also report the number of clicks from the original Caltech 101 images, but since there is no interface for the Caltech 101 dataset, we report the number of vertices on their ground truth polygon as the number of user clicks needed to manually delineate the ground truth segmentation.

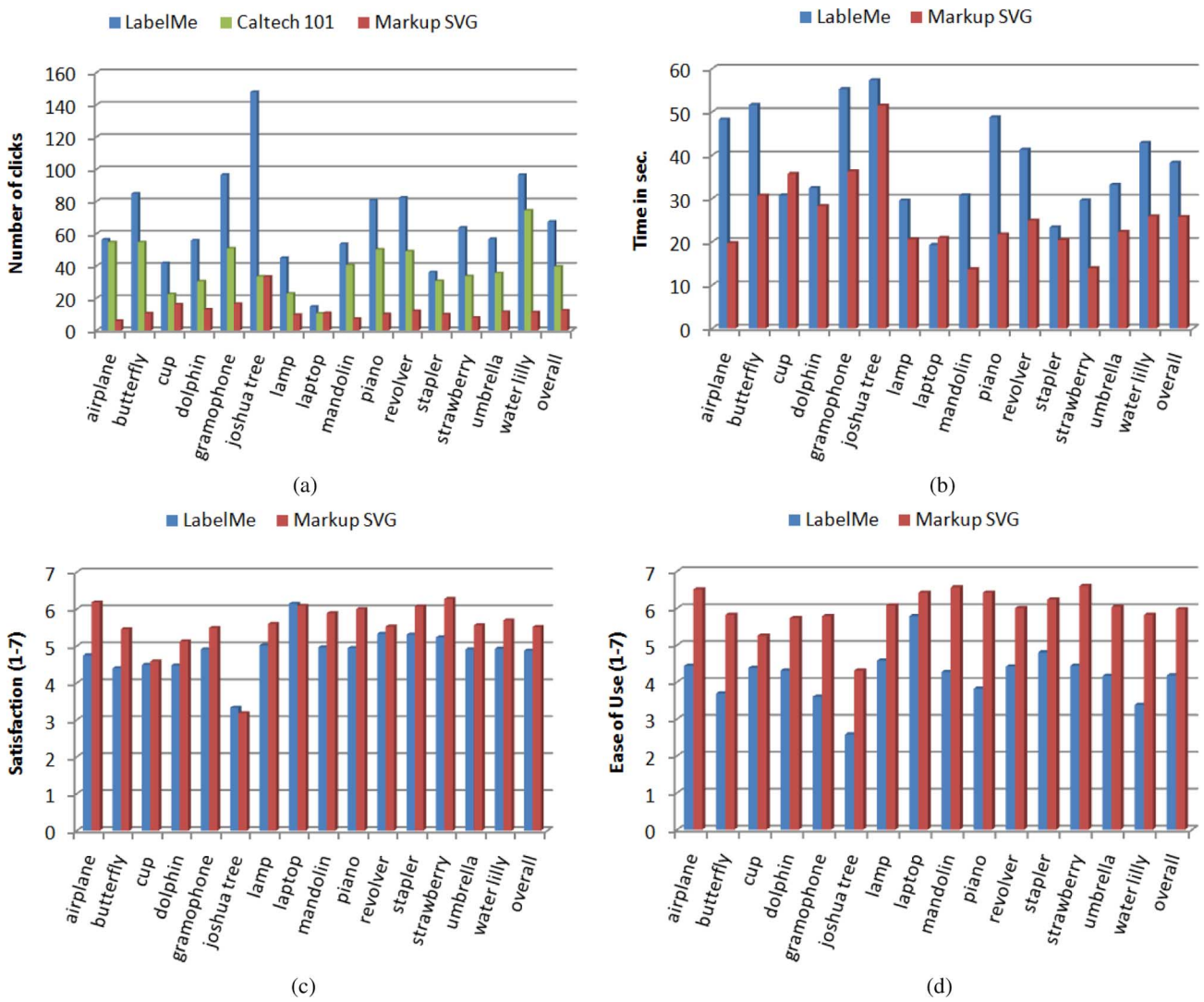
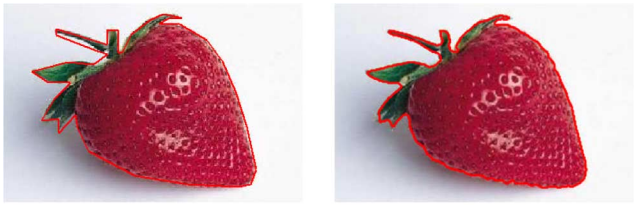


Fig. 14. (a) Click comparison, (b) time(secs.), (c) segmentation accuracy satisfaction, and (d) ease of use results on our user study between Markup SVG, LabelMe, and Caltech 101 (only click comparison) in 15 different object categories averaged on 11 different users.

Next, we timed the users from the time they first load the image to the time they complete an object segmentation in both our system and the LabelMe system. The results are presented in Fig. 14(b). Finally, after segmenting the region, we asked the users to evaluate the segmentation boundary accuracy and rate their experience in terms of ease of use to achieve this boundary. The satisfaction and ease of use results were recorded on a scale from 1 to 7 ($1 = \text{very disappointed}, 7 = \text{very satisfied}$) and ($1 = \text{very difficult to use}, 7 = \text{very easy to use}$), respectively, and the user responses are presented in Fig. 14(c) and (d).

From our results, we notice several interesting trends similar to our first segmentation study. First, as anticipated, our system significantly reduces the number of clicks necessary to delineate the boundary when compared to manual methods, LabelMe and Caltech 101. We also see the ease of use of the Markup SVG system is significantly higher than manual methods. Second, we see that in LabelMe, the time and number of clicks required to segment objects is proportional to the complexity of the boundary. For example, when segmenting a complex object like a piano or gramophone, the users time increases and ease of use decreases. On the other hand, the users in our annotation tool were less affected by the complexity of the object and were able to consistently maintain a faster time and lower number of clicks. Third, we see the users are generally more satisfied with our computer assisted segmentation results than their manual attempts, especially when dealing with high curvature objects. We attribute this improvement to a manual approximation error akin to a discrete method approximating a continuous function. Since the manual methods use straight lines between vertices, the user would need to click numerous times in order to fit lines around curves. This issue became more apparent in the discrepancy between the number of clicks performed by the users on the LabelMe tool versus the Caltech 101 dataset. We originally hypothesized that the regions created by these two systems should be roughly equivalent. However, after visualizing the results of the Caltech 101 dataset, it appears that the ground truth also suffers from the same approximation errors that appeared in the LabelMe segmentation results. However, utilizing our framework and computer-aided methods, we are able to address this issue. As an example, we can use our



(a) (b)

Fig. 15. Ground truth polygon from the Caltech 101 dataset in (a). Refined ground truth by using Markup SVG and our active contour segmentation module in (b). Our system is able to assist in the curvature issues seen in manually created ground truth datasets.

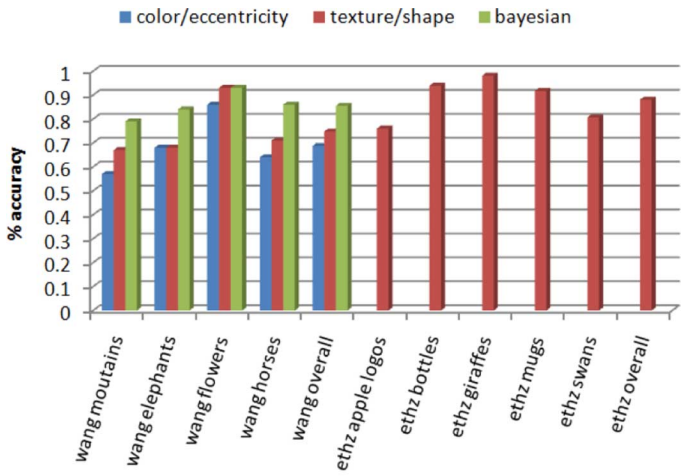
active contour segmentation assist module to refine the existing boundary of the Caltech 101 images to better fit the object boundary as seen in Fig. 15.

C. Annotation Assist Results

For evaluating our annotation assist results, we performed experiments on the ETHZ shape database and the Wang database. The ETHZ shape database was collected by [40] and consists of five shape categories, apple logos, bottles, giraffes, mugs, and swans. This database has a total of 255 images with varying scales and object locations. The WANG database is a 1000 image subset of the Corel stock photo database that consists of 10 category classifications with 100 images in each category.

For our first experiment, we recruited several volunteers to annotate the images in the WANG dataset online using our Markup SVG tool. These volunteers were given the freedom to annotate whatever objects they wanted and use whatever labels they choose. After analyzing their annotations, we found four categories that had an adequate number of exemplars in our database (greater than 30), *mountains, elephants, flowers, and horses*. From each of these categories, we randomly selected 25 unlabeled objects for a total of 100 and had our system automatically annotate these regions. We tested the annotation of the regions using three different methods, one using basic low-level features ($L^*a^*b^*$ color histograms and eccentricity), another using the same features but adding texture and shape matching features, and finally a high-level Bayesian matching using prior image annotations within the same image to adjust the probability as described in (15). We provide the accuracy results in Fig. 16.

Next, we manually annotated the shape regions in the 255 images contained in the ETHZ dataset using our high-level region annotation module. We then performed a leave-one-out automatic annotation experiment using our annotation assist module in each of the five categories. For this automatic annotation experiment, we selected one region in each of the 255 images and then classified the region based upon the maximum likelihood score obtained from (9). The score is calculated based upon the low-level feature comparison between the current selected region and the manually annotated regions (the exemplar regions). For this experiment, we use the low-level features, $L^*a^*b^*$ color histograms and texture histograms with χ^2 distance measure, eccentricity, total area, and centroid positions with $\mathcal{L}_1$ distance, and shape histograms with Kullback-Leibler divergence. Fig. 16



(a) ETHZ and Wang annotation accuracy.

Fig. 16. Automatic annotation accuracy (%) for the ETHZ and WANG database. The WANG results use color/eccentricty, then shape/texture along with color/eccentricty, and finally a Bayesian formulation. The ETHZ dataset specifies one relevant object per image and so only the color/eccentricty/shape/texture results are reported.

shows a comparison of the accuracy of our annotation assist module in obtaining automatically the correct region annotation.

D. Note on Scalability and Preprocessing

As the number of low-level modules imported into our image abstraction increases, naturally the size of the SVG encoding will also follow. On extremely large datasets, the size of the SVG encodings can become increasingly unwieldy. For scalability reasons, we develop an SVG compression technique to manage the size of the SVG file based on the observation that storing the polygon point set attribute from our segmentation modules typically takes more than 50% of the total encoding space. Thus, we use a polynomial fitting technique to reduce the number of points stored in our abstraction when the points do not deviate beyond a certian epsilon, ϵ_p, from the polynomial line. For the ETHZ database with four segmentation modules, we encode a total of 44 105 regions. With no compression, the SVG documents occupy 83.9 MB of space, an average of 329 k per encoding. With an ϵ_p of 2 pixels, we can reduce the total space to 34.6 MB. The uncompressed bitmap size of the raw images is 80.7 MB. For the WANG database and four segmentation modules, we create 271 007 regions in our abstraction. With no compression, the SVG documents occupy 473.8 MB of space, an average of 473 k per encoding. With an ϵ_p of 2 pixels, we can reduce the total space to 209.1 MB. The uncompressed bitmap size of the raw images is 378.1 MB.

Also, many of our methods require preprocessing (e.g., automatic segmentation, feature extraction) which can affect the online user experience when uploading new images. To alleviate this issue, we have converted many of our preprocessing algorithms to perform on CUDA, the NVIDIA GPU architecture. We have developed or implemented numerous automatic segmentation algorithms, including Damascene [41] (the gPb CUDA implementation), Meanshift [42], and Deterministic Annealing [42]. Thus, on our server equipped with an NVIDIA GTX 285,

(a) (b) (c)

Fig. 17. Limitation of automatic methods when dealing with complex objects. In (a), we show the ground truth polygon from the Caltech 101 dataset. In (b), we show our assisted result using our active contour assist module. In (c), we use a manual method to segment the tree.

the preprocessing time is typically reduced two orders of magnitude, from several minutes to several seconds.

V. Discussion and Conclusion

Limitations: In some cases, our segmentation assist modules are not helpful enough when the low-level methods, e.g., automatic segmentation algorithms, fail to provide enough relevant information. Occasionally, this is the case when an object is occluded or extremely complicated as seen in Fig. 17. For these cases, manual methods are still necessary to achieve a better segmentation. Our manual markup module is designed to address these cases; however, time is usually spent by the user testing assisted methods before realizing that the manual method is the best approach for the specific object. In our user study, participants reacted in this manner for approximately 3 (out of 75) images. In the event of object occlusion, the low-level information is insufficient for assisting the user in the segmentation and annotation task. If the complete shape is desired for an occluded object, manual methods would again be the best approach.

Discussion: Given global or generated tags from online systems like ALIPR, Flickr, and others [43], we plan on developing additional action modules that can map image tags to specific regions. Other interesting modules could utilize geographic information to analyze images, similar to the method in Kennedy *et al.* [44] and Hayes and Efros [45]. For handling the synonymy of annotations and building our semantic relationships between regions, we have been experimenting with WordNet [46], Resource Description Framework (RDF), and Web Ontology Language (OWL). These ontology languages are also XML based and thus fully compatible with our tool. Another very interesting direction where we see great potential is in video annotation and non-photo realistic rendering techniques using our SVG abstraction. Further, we plan on encoding different user interactions—type, time stamp, speed, number of iterations—to assist in user intention modeling. Our framework and web-based tool enable a community of users or experts to collaboratively annotate images in a large repository. Thus, we also plan on creating a module API that allows users to independently create different modules that can plug into our annotation tool.

In conclusion, we present a content-aware image abstraction based on SVG and an annotation tool for efficient and accurate image annotation. We present four markup modules, low-level, high-level, heterogeneous data, and action modules. Our abstraction organizes a set of markup modules in an open, extensible framework. Our action modules are able to leverage a combination of low-level and high-level module outputs to assist the user in the major aspects of image annotation including segmentation and annotation. Our system exploits the well-known advantages of SVG and XML such as visualization, flexibility, and interoperability with existing technologies. Additionally, our system can be interacted with in a standard web browser and manipulated using any major query languages build around XML. Finally, we demonstrate the effectiveness of our abstraction on multiple benchmark datasets in efficiency, annotation accuracy, satisfaction, and ease of use.

Acknowledgment

The authors would like to thank the reviewers and editors for their insightful comments and their user study volunteers for their diligent work.

References

[1] J. S. DeLoache, S. L. Pierroutsakos, D. H. Uttal, K. S. Rosengren, and A. Gottlieb, "Grasping the nature of pictures," *Psycholog. Sci.*, vol. 9, no. 3, pp. 205–210, 1998.

[2] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: A database and web-based tool for image annotation," *Int. J. Comput. Vision*, vol. 77, no. 1, pp. 157–173, 2008.

[3] Flickr Photo Sharing Service. [Online]. Available: http://www.flickr.com.

[4] J. Li and J. Wang, "Real-time computerized annotation of pictures," in *Proc. 14th Annual ACM Int. Conf. Multimedia*, 2006, pp. 920–920.

[5] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. SIGCHI Conf. Human Factors in Computing Systems*, 2004, pp. 319–326.

[6] D. Stork, "The open mind initiative," *IEEE Expert*, vol. 14, pp. 16–20, 1999.

[7] B. Yao, X. Yang, and S. Zhu, "Introduction to a large-scale general purpose ground truth database: Methodology, annotation tool and benchmarks," in *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2007, pp. 169–169.

[8] F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[9] E. Mortensen and W. Barrett, "Interactive segmentation with intelligent scissors," *Graph. Models Image Process.*, vol. 60, no. 5, pp. 349–384, 1998.

[10] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[11] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[12] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[13] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 359–369, Mar. 2002.

[14] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation," *Med. Image Comput. Comput.-Assist. Intervent.*, pp. 773–780, 2005.

[15] Y. Boykov and M. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in ND images," in *Proc. Int. Conf. Computer Vision*, 2001, vol. 1, pp. 105–112.

[16] A. Lefohn, J. Cates, and R. Whitaker, "Interactive, gpu-based level sets for 3d segmentation," *Med. Image Comput. Comput.-Assist. Intervent.*, pp. 564–572, 2003.

[17] W. Barrett and E. Mortensen, "Interactive live-wire boundary extraction," *Med. Image Anal.*, vol. 1, no. 4, pp. 331–341, 1997.

[18] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 2, pp. 158–175, Feb. 2002.

[19] J. Bentley, "The most beautiful code I never wrote," in *Beautiful Code*. Sebastopol, CA: O'Reilly Media, 2007, pp. 29–40.
[20] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Comput. Survey*, vol. 40, no. 2, pp. 1–60, 2008.
[21] R. Zhao and W. Grosky, "Bridging the semantic gap in image retrieval," *Distrib. Multimedia Databases: Tech. Appl.*, pp. 14–36, 2001.
[22] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Computer Vision*, 1999, vol. 2, pp. 1150–1150.
[23] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progr. Brain Res.*, vol. 155, pp. 23–36, 2006.
[24] Z. Wu, Q. Ke, M. Isard, and J. Sun, "Bundling features for large scale partial-duplicate web image search," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1643–1650.
[25] J. Hays and A. Efros, "Scene completion using millions of photographs," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 26, no. 3, 2007.
[26] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognit.*, vol. 40, no. 1, pp. 262–282, 2007.
[27] SVG, Scalable Vector Graphics, 2003. [Online]. Available: http://www.w3.org/Graphics/SVG/.
[28] P. Meer and B. Georgescu, "Edge detection with embedded confidence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 12, pp. 1351–1365, Dec. 2001.
[29] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 1997, pp. 731–737.
[30] J. Vogel and B. Schiele, "Semantic modeling of natural scenes for content-based image retrieval," *Int. J. Comput. Vision*, vol. 72, no. 2, pp. 133–157, 2007.
[31] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.
[32] J. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision Res.*, vol. 20, no. 10, pp. 847–856, 1980.
[33] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[34] F. Mokhtarian and A. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 8, pp. 789–805, Aug. 1992.
[35] X. Huang, Z. Li, and D. N. Metaxas, "Learning coupled prior shape and appearance models for segmentation," in *Proc. Medical Image Computing and Computer Assisted Intervention*, 2004, pp. 60–69.
[36] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Int. Conf. Machine Learning*, 2001, pp. 282–289.
[37] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proc. Int. Conf. Computer Vision*, 2009, vol. 5, pp. 7–7.
[38] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," *Eur. J. Comput. Vision*, 2010.
[39] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. 3, no. 6, pp. 610–621, 1973.
[40] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Object detection by contour segment networks," *Eur. J. Comput. Vision*, vol. 3953, pp. 14–14, 2006.
[41] B. Catanzaro, B. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer, "Efficient, high-quality image contour detection," in *Proc. Int. Conf. Computer Vision*, 2010, pp. 2381–2388.
[42] E. Kim, W. Wang, H. Li, and X. Huang, "A parallel annealing method for automatic color cervigram image segmentation," in *Proc. MICCAI-GRID HPC Workshop Medical Image Computing and Computer Assisted Intervention*, 2009.
[43] J. Jia, N. Yu, and X. Hua, "Annotating personal albums via web mining," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 459–468.
[44] L. Kennedy, M. Naaman, S. Ahern, R. Nair, and T. Rattenbury, "How flickr helps us make sense of the world: context and content in community-contributed media collections," in *Proc. 15th ACM Int. Conf. Multimedia*, 2007, pp. 640–640.
[45] J. Hays and A. Efros, "IM2GPS: Estimating geographic information from a single image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
[46] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.

Edward Kim (S'10) received the B.S.E. degree in computer science and the M.S.E. degree in computer graphics and game technology from the University of Pennsylvania School of Engineering and Applied Science, Philadelphia, in 2003 and 2008, respectively. He is currently pursuing the Ph.D. degree in the Computer Science and Engineering Department at Lehigh University, Bethlehem, PA.

His research interests are in the areas of computer vision, computer graphics, and multimedia storage and retrieval.

Xiaolei Huang (M'05) received the bachelor's degree in computer science and engineering from Tsinghua University, Beijing, China, in 1999, and the Master's and doctorate degrees in computer science from Rutgers, New Brunswick, NJ, in 2001 and 2006, respectively.

She is currently a P.C. Rossin Assistant Professor in the Computer Science and Engineering Department at Lehigh University, Bethlehem, PA. Her research interests are in the areas of computer vision, biomedical image analysis, computer graphics, and multimedia retrieval.

Dr. Huang serves on the program committees of several international conferences on computer vision, biomedical image computing, and computer graphics, and she is a regular reviewer for journals including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI) and the IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP). She is a member of the IEEE Computer Society.

Gang Tan received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1999 and the master's and doctorate degrees in computer science from Princeton University, Princeton, NJ, in 2001 and 2005, respectively.

Currently, he is an Assistant Professor of Computer Science and Engineering at Lehigh University, Bethlehem, PA. His research interests include software security and programming languages. He leads Lehigh's Security of Software (SOS) group, which aims to create secure and reliable software systems.