Online Random Forest for Interactive Image Segmentation

Olga Barinova[1], Roman Shapovalov[1], Sergey Sudakov[2], Alexander Velizhev[1]

[1] Lomonosov Moscow State University
{obarinova,shapovalov,avelizhev}@graphics.cs.msu.su
[2] EligoVision Ltd.
svsudakov@gmail.com

Abstract. Many real-world applications require accurate segmentation of images into semantically-meaningful regions. In many cases one needs to obtain accurate segment maps for a large dataset of images that depict objects of certain semantic categories. As current state-of-the art methods for semantic image segmentation do not yet achieve the accuracy required for their use in real-world applications, they are not applicable in this case. The standard solution would be to apply interactive segmentation methods, however their use for a large number of images would be laborious and time-consuming. In this work we present an online learning framework for interactive semantic image segmentation that simplifies processing of such image datasets. This framework learns to recognize and segment user-defined target categories using the ground truth segmentations provided by user. While the user is working on ground truth image segmentation, our framework combines online-learned category models with the standard stroke-propagation mechanisms that are typically used in interactive segmentation methods. Our implementation of this framework in a software system has specific interface features that minimize the required amount of user input. We evaluate the implementation on several datasets from completely different domains (*Sowerby* dataset containing 7 different semantic categories, *sheep & cows* dataset containing 3 categories, and 6 different *flower* datasets with 2 categories each). Usage of our system requires substantially less user effort compared to the traditional interactive segmentation methods.

1 Introduction

Many applications, such as aerial and space image processing, defect detection, and medical imaging require accurate segmentation of large image datasets into some semantically-meaningful zones. Despite the substantial progress made, current state-of-the art methods for automatic semantic image segmentation [1] do not yet achieve the accuracy required for their use in real-world applications. The standard solution to obtain accurate segmentations for a dataset of images would be to apply interactive segmentation methods. Moreover, in some cases interactivity and providing feedback for the computational algorithms to perform segmentation, can not only overcome the inherent difficulties of automatic

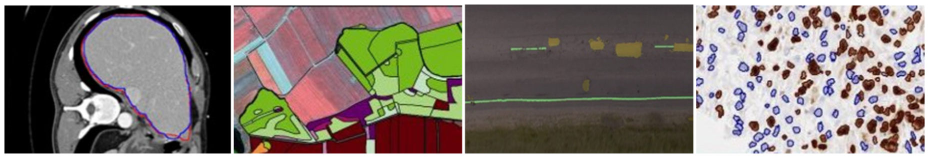


Fig. 1. Image segmentation is used in medical imaging, processing aerial, space images, and detection of road defects. In many applications one needs to accurately segment objects of the some target semantic categories from a large dataset of images. In this work we present a framework that automates this process and substantially reduces the user effort

semantic segmentation, but may also be desirable because the user may want to be able to control the segmentation process and review the results. However applying standard interactive segmentation software for large image datasets would be an extremely laborious and time-consuming task.

In this work we consider the case when one needs to perform segmentation of a large image dataset into a number of semantically-meaningful categories. We aim at developing a general framework that would work with any user-defined target categories and learn these categories from the user input as semantic segmentation methods do. On the other hand, we want to give the user as much control on the segmentation results as interactive image segmentation methods provide. Our main goal is to minimize user effort while allowing her to produce accurate image segmentations. Most existing methods for interactive image segmentation work with a single image [2–4], which limits their power. For example, segmentation of an image from *MSRC* dataset with our system takes less than a minute compared to 15–60 minutes for manual annotation [5].

Related task of inducing segmentation from example was looked at [23]. In this approach a non-parametric model of the provided training pair is constructed by selecting a set of patch-based representatives inside each labeled region in the training image. These representatives are used to quantify the degree of resemblance between small regions in the input image and the labeled regions in the training set.

Adaptive learning of object detection was considered in [6]. The models for new categories may benefit from the detectors built previously for other categories. [7] presented a framework for dynamic visual category learning using incremental support vector machine. That method exploits a previously built classifier to learn the optimal parameters for the current set of training images more efficiently, which is faster than batch retraining. In contrast to those works, we consider a problem of interactive semantic segmentation and our framework enables both adding new categories and incremental learning of the existing ones.

The paper is organized as follows. Next section describes the general workflow of our system and our semantic segmentation algorithm. In Section 3 we describe the online random forest. Section 4 describes the experiments, and the last section is left for conclusions.

2 Interactive Semantic Segmentation Framework

2.1 Interactive Semantic Segmentation Workflow

Suppose one needs to process a large image dataset and obtain accurate segmentation of the objects of certain categories. In framework, a user examines images from the dataset in sequence. Each image is presented as a set of superpixels (Section 2.2). The first image is segmented manually: a user should label each superpixel with one of the category labels. The newly-obtained labelling is used to update the appearance model. When the user opens one of the consequent images, segmentation is performed automatically using the current appearance model. Then the user may correct mistakes of the automatic method by changing superpixel labels. Each time the user approves the (possibly corrected) segmentation result, the system learns from the newly obtained examples of object categories and background. As training goes, user time spent on correction of category map reduces, thus the rate of image labelling increases.

Our **implementation** provides a set of tools to simplify the process of error correction for the user. The brush tool is used to modify superpixel labels. It is possible to change labels of groups of neighbouring superpixels by choosing the appropriate brush size. Each time the user applies it, the system also updates the global labelling using the newly observed labels as context. Therefore, one brush stroke usually changes labels of a large number of pixels. We use hierarchical clustering to obtain superpixels, so the user can switch between different scales of superpixels and choose appropriate scale for correction of errors in the segmentation. We also provide a user with a set of sliders, each one controls the trade-off between false positive and false negative rates of one object category. The sliders help to significantly reduce the amount of manual work in the beginning of the image set processing, when few images have been seen, and the classifiers are likely to be biased towards some categories.

The output of the most time-consuming operations (such as over-segmentation and feature extraction) can be cached, so in practice those operations are performed offline, before the user starts working with the system. We use efficient methods for inference of the optimal segmentation and learning the appearance models of object categories (see Sections 3 and 2.2), thus a user gets immediate response from the system.

2.2 Semantic segmentation algorithm

We obtain pixelwise object segmentation by assigning category labels to a set of superpixels obtained by clustering the joint color and coordinate space with mean-shift algorithm [8]. Usage of superpixels improves computational efficiency as well as makes segmentation more robust. Texture and color features are computed from the image by applying a filter bank [9]. We use texton histograms over superpixels generated by mean shift similarly to [10]. To take the geometric information into account we use simple geometric features like variance, elongation, orientation and area of a superpixel.

We use a simple pairwise conditional random field (CRF) that allows efficient inference. The vector of superpixel labels $\mathbf{c} = \{c_i\}$ is determined as the one that minimizes the following energy function:

$$E(\mathbf{c}) = -\sum_i \Psi_i\left(c_i|I\right) - \sum_{(i,j)} \Phi_{ij}\left(c_i, c_j|I\right), \tag{1}$$

where the first term sums the appearance potentials of individual superpixels, the second sum is over the neighbouring pairs of superpixels.

The unary potential for assigning the object category c_i to the i-th superpixel in the image I is computed as $\Psi_i\left(c_i|I\right) = \log p\left(c_i|S_i, I\right) + \eta(c_i)$, where $p\left(c_i|S_i, I\right)$ is the probabilistic output of the online random forests (Section 3) for c_i-th class on the superpixel S_i. The second term $\eta(c_i)$ is the slider value that can be treated as the prior that prefers some categories over the others.

Pairwise potentials consist of the two terms: $\Phi_{ij}\left(c_i, c_j|I\right) = \theta\left(c_i, c_j|I\right) + \tau\left(c_i, c_j\right)$. The first term is the inverse of the boundary strength provided by the mean-shift segmentation. The second term corresponds to a fraction of neighbouring superpixels of the classes c_i and c_j among all neighbouring superpixels in the images seen so far. There are efficient algorithms for minimization of the energy (1), so inference can be performed every time when the user moves a slider to change the trade-off between false positives and false negative rates.

3 Online Random Forest

Our variant of online random forest[1] builds a set of Hoeffding trees [11]. This method is proven to produce the trees asymptotically arbitrarily close to the ones produced by a batch learner. Therefore the incremental nature of our version of Online Random Forest algorithm does not significantly affect the quality of the model it produces.

In Breiman's Random Forest [12] the training set of each tree is obtained by random resampling. This means that the probability that each of N instances is sampled exactly K times for each tree is binomially distributed:

$$p(K = k) = \binom{N}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{N-k}. \tag{2}$$

As N goes to infinity, the distribution of K converges to the Poisson distribution with the parameter equal to 1: $K \sim \frac{\exp(-1)}{k!}$. Therefore online bootstrapping can be performed as follows: for each base model, choose each example $K \sim \text{Pois}(1)$ times and update the base model accordingly. To diversify the trees in our variant of online random forest each tree operates with a random subset of features.

[1] `http://graphics.cs.msu.ru/en/science/research/machinelearning/bolt`

```
Online Random Forest
Input: Example $(x, y)$
For each base model $h_m = h_1, \ldots, h_M$
    Set $k = Poisson(1)$
    Do $k$ times
        $h_m = Update\_tree\,(x, y)$
Return updated $\{h_1, \ldots, h_M\}$
```

As long as Hoeffding trees can handle multiclass classification, our Online Random Forest naturally performs multiclass classification without any change in the algorithm.

Handling imbalanced classes. It was proven [13] that error balancing can be achieved by resampling the training set. As the expectation of Poisson random variable $p(K = k) = \frac{\lambda^k}{k!}\exp(-\lambda)$ equals to the parameter λ of Poisson distribution, we can balance the errors by introducing various parameters of Poisson distribution for different classes. In this work we use the balanced version of the Online Random Forest and allow the user to control false positive and false negative rates with the same sliders that were discussed in section .

4 Experiments

Image datasets. In the first experiment we used *Sowerby* dataset that contains 100 images of urban scenes. The goal in this experiment was to perform accurate multi-zone segmentation into 7 object categories provided in the ground truth annotation of this dataset.

In the second experiment we used a subset of the *MSRC* dataset[2] composed of 60 images of cows and sheep. In this experiment we considered a 3-zone segmentation problem where the goal was to segment cows and sheep from background.

In the third experiment we used a subset of *17-flower* dataset[3]. We considered 6 different flowers (daffodil, tigerlily, daisy, fritillary, pansy, sunflower), and 80 images of each flower. The goal in this experiment was to segment each flower from the background.

Measuring usability of the system. The typical sequence of user actions to segment an image in our system is the following. The user starts with tuning the sliders to adjust the false positive vs. false negative rates, and then corrects the segmentation errors using brush tool. In most cases the optimal strategy for error correction is to start with fixing the errors in the coarsest scale of superpixels and then proceed to more detailed scales.

To quantify the user input we have implemented a robot-user that emulates the actions of a human user working with the system. Given the initial image

[2] `http://research.microsoft.com/en-us/projects/ObjectClassRecognition/`
[3] `http://www.robots.ox.ac.uk/~vgg/data/flowers/`

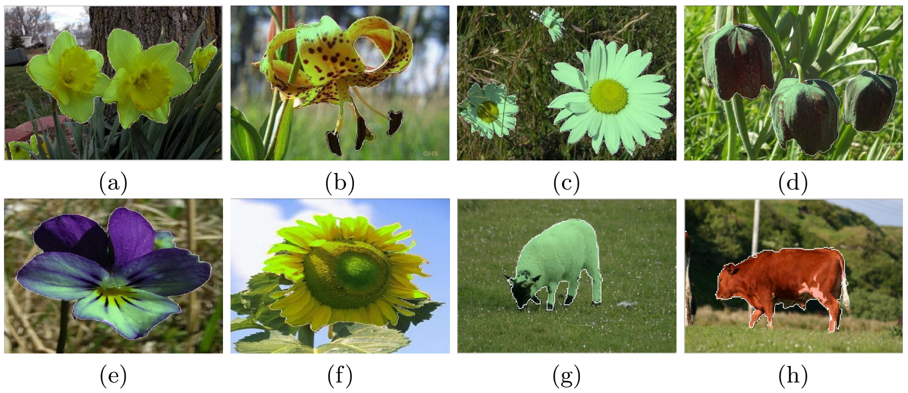


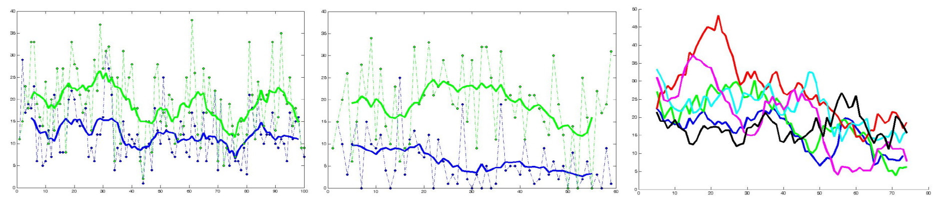
Fig. 2. Example segmentations created using our framework: (a) daffodil, (b) tigerlily, (c) daisy, (d) fritillary, (e) pansy, (f) sunflower; (g),(h) images from our *sheep and cows* dataset. Object is shown with blending, the boundary of an object is marked with white

segmentation, the robot first finds the optimal values of $\eta(c_i)$ for all object classes (i.e. optimal position of the sliders). For that we minimize the total area of misclassified superpixels using Nelder–Mead algorithm. Then we count the number superpixels that need to change labels in order to obtain correct segmentation result. We start by correcting the errors at the coarsest scale and proceed to more detailed scales of superpixels. The resulting metric characterizes overall amount of user input required to obtain correct result using our system, and we refer to it as *usability metric*.

To measure the gain provided by learning the appearance models of object categories, we compared two values of usability metric. First we computed the usability metric for the case of fully manual image segmentation using our brush tool, i.e assuming that all superpixels are initially labelled as background. Second, we computed the usability metric for our semantic segmentation framework.

The results of this experiment for *Sowerby* and *sheep & cows* datasets are shown in Figure 3 (a, b). The green lines show the results in fully manual case, and the blue lines show the results for our framework. The use of automatic segmentation helps to significantly reduce required amount of user input compared to performing fully manual segmentation.

To measure the gain of online learning we looked at the behaviour of the plots of usability metric with respect to the the total number of images processed. As the values of usability metric vary significantly for each particular image, we computed the average over 9 subsequent images to estimate the long-term trends of usability metric. The effect of online learning is most clearly visible for the flowers image datasets (Figure 3 (c)), where the total number of superpixels that require relabelling tends to decrease over time. For *Sowerby* and *sheep & cows* image datasets this metric decreases also decreases, but more slowly.



(a) (b) (c)

Fig. 3. The number of superpixels that have to change their label subject to the number of images processed: (a) *Sowerby* dataset, (b) *sheep & cows* dataset. Performance of our system is shown in blue, total number of clicks required to label an image from scratch is shown in green. Thin lines represent automatically calculated number of superpixels that need to change their labels as described in text, thick lines show the average value over 9 subsequent images. (c) averaged values of usability metric for the *6-flowers* datasets: blue — daffodil, green — fritillary, red — pansy, cyan — sunflower, magenta — tigerlily, black — daisy

Measuring the time. We compared the time required from a human user to obtain high-quality image segmentation with our system and with GrowCut interactive segmentation tool [4] on the *6-flowers* image datasets. The user had practical experience with both systems. The time required for producing high-quality image segmentation for a set of 80 images of the same flower varied from 14 min to 46 min. GrowCut took about twice more time to produce the segmentation of similar quality.

5 Conclusions

We have presented a framework for interactive semantic image segmentation that is based on online learning. The experiments show that online learning of object appearance models helps to significantly reduce user input required to obtain accurate image segmentation.

References

1. Yao, J., Fidler, S., Urtasun, R.: Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In: CVPR. (2012)
2. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.: Interactive image segmentation using an adaptive gmmrf model. In: ECCV. (2004)
3. Grady, L.: Random walks for image segmentatio. Transaction on Pattern Analysis and Machine Intelligence (2006)
4. Vezhnevets, V., Konouchine, V.: "grow-cut" - interactive multi-label n-d image segmentation. In: Graphicon. (2005) 150–156
5. Kohli, P., Ladicky, L., Torr, P.: Robust higher order potentials for enforcing label consistency. In: CVPR. (2008)

6. Opelt, A., Pinz, A., Zisserman, A.: Incremental learning of object detectors using a visual shape alphabet. In: CVPR. (2006)
7. Yeh, T., Darrell, T.: Dynamic visual category learning. In: CVPR. (2008)
8. Paris, S., Durand, F.: A topological approach to hierarchical segmentation using mean shift. In: CVPR. (2007)
9. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: ECCV. (2006) 1–15
10. Yang, L., Meer, P., Foran, D.: Multiple class segmentation using a unified framework over mean-shift patches. In: CVPR. (2007)
11. Domingos, P.., Hulten, G.: Mining high-speed data streams. In: Knowledge Discovery and Data Mining. (2000)
12. Breiman, L.: Random forests. Machine Learning Journal **45**(1) (2001) 532
13. Elkan, C..: The foundations of cost-sensitive learning. In: IJCAI. (2001)