Learning an Interactive Segmentation System

Hannes Nickisch *
MPI for Biological Cybernetics
hn@tue.mpg.de

Carsten Rother
Microsoft Research Cambridge
carrot@microsoft.com

Pushmeet Kohli
Microsoft Research Cambridge
pkohli@microsoft.com

Christoph Rhemann
Vienna University of Technology
rhemann@ims.tuwien.ac.at

ABSTRACT

Many successful applications of computer vision to image or video manipulation are interactive by nature. However, parameters of such systems are often trained neglecting the user. Traditionally, interactive systems have been treated in the same manner as their fully automatic counterparts. Their performance is evaluated by computing the accuracy of their solutions under some fixed set of user interactions. This paper proposes a new evaluation and learning method which brings the user in the loop. It is based on the use of an active robot user – a simulated model of a human user. We show how this approach can be used to evaluate and learn parameters of state-of-the-art interactive segmentation systems. We also show how simulated user models can be integrated into the popular max-margin method for parameter learning and propose an algorithm to solve the resulting optimisation problem.

Keywords

Interactive segmentation, interactive learning, SVMstruct.

1. INTRODUCTION

Problems in computer vision are known to be hard, and very few fully automatic vision systems exist which have been shown to be accurate and robust under all sorts of challenging inputs. These conditions in the past had made sure that most vision algorithms were confined to the laboratory environment. The last decade, however, has seen computer vision finally come out of the research lab and into the real world consumer market. This great sea change has occurred primarily on the back of the development of a number of interactive systems which have allowed users to help the vision algorithm to achieve the correct solution by giving hints. Some successful examples are systems for image and video manipulation, and interactive 3D reconstruction tasks. Image stitching and interactive image segmentation are two of the most popular applications in this area. Understandably, interest in interactive vision systems has grown in the last few years, which has led to a number of workshops and special sessions in vision, graphics, and user-interface conferences [1].

Interactive image classification without user model is discussed in the machine learning literature [10] and active inference in random field graphs is already used in the knowledge discovery community [4].

The performance of an interactive system strongly depends on a number of factors, one of the most crucial being the user. This user dependence makes interactive systems quite different from their fully automatic counterparts, especially when it comes to learning and evaluation. Surprisingly, there has been little work in computer vision or machine learning devoted to *learning* interactive systems. This paper tries to bridge this gap.

We choose interactive segmentation to demonstrate the efficacy of the ideas to inspire other researchers to use it e.g. in interactive 3D reconstruction. The framework applies general computer vision problems with an energy function, an error metric and a way of measuring the interaction effort.

Interactive segmentation aims to separate an object of interest from the rest of an image. It is a classification problem where each pixel is assigned one of two labels: foreground (fg) or background (bg). The interaction comes in the form of sets of pixels marked by the user by help of brushes to belong either to fg or bg. We will refer to each user interaction in this scenario as a *brush stroke*.

We address two questions: (1) How to evaluate any given interactive segmentation system? and (2) How to learn the *best* interactive segmentation system? Observe that the answer to the first question gives us an answer to the second by picking the segmentation system with the best evaluation.

We demonstrate the efficacy of our evaluation methods by learning the parameters of the state-of-the-art system for interactive image segmentation. We then extend parameter learning in structured models by including the user effort in the max-margin method. The contributions of this paper are: (1) The study of the problems of evaluating and learning interactive systems. (2) The use of a user model for evaluating and learning interactive systems. (3) A comparison of state-of-the-art segmentation algorithms under an explicit user model. (4) A new algorithm for max-margin learning with user in the loop.

*Corresponding author

ICVGIP '10, December 12-15, 2010, Chennai, India

[1] e.g. ICCV 2007, NIPS 2009 and CVPR 2010

Two recent articles [13, 5] already employ our robot user to learn and compare various different segmentation algorithms, which demonstrates the usefulness of our approach.

Organization of the paper.
In Section 2, we discuss the problem of system evaluation. In Section 3, we give details of our problem setting, and explain the user model and segmentation systems. Section 4 explains the naïve line-search method for learning segmentation system parameters. In Section 5, we show how the max-margin framework for structured prediction can be extended to handle interactions, and show some basic results. Conclusions are given in Section 6.

2. EVALUATING INTERACTIVE SYSTEMS

Performance evaluation is one of the most important problems in the development of real world systems. There are two choices to be made: (1) The data sets on which the system will be tested, and (2) the quality measure. Traditional computer vision and machine learning systems are evaluated on preselected training and test data sets. For instance, in automatic object recognition, one minimizes the number of misclassified pixels on datasets such as PASCAL VOC [9].

In an interactive system, these choices are much harder to make due to the user in the loop. Users behave differently, prefer different interactions, may have different error tolerances, and may also learn over time. The true objective function of an interactive system – although intuitive – is hard to express analytically: The user wants to achieve a satisfying result easily and quickly. We will now discuss a number of possible solutions, some of which, are well known in the literature (see table 1 for an overview). We rely on the standard assumption that there exists a consistent set of optimal parameters for a set of images.

Method	user in loop	user can learn	inter-action	effort model	parameter learning	time	price
User model	yes	yes	yes	yes	this paper	fast	low
Crowd sourcing	yes	yes	yes	yes	conceivable	slow	a bit
User study	yes	yes	yes	yes	infeasible	slow	very high
Static learning	no	no	no	no	used so far	fast	very low

Table 1: Comparison of methods.

2.1 Static Interactions

A fixed set of user-made interactions (brush strokes) associated with each image of the dataset is most commonly used in interactive image segmentation [6, 22, 8]. These strokes are chosen by the researchers themselves and are encoded using image trimaps. These are pixel assignments with foreground, background, and unknown labels (see figure 2b). The system to be evaluated is given these trimaps as input and their accuracy is measured by computing the Hamming distance between the obtained result and the ground truth. This scheme of evaluation does not consider how users may change their interaction by observing the current segmentation results. Evaluation and learning methods which work with a fixed set of interactions will be referred to as *static* in the rest of the paper.

Although the static evaluation method is easy to use, it suffers from a number of problems: (1) The fixed interactions might be very different from the ones made by actual users of the system. (2) Different systems prefer different type of user hints (interaction strokes) and thus a fixed set of hints might not be a good way of comparing two competing segmentation systems. For instance, geodesic distance based approaches [2, 12, 22] prefer brush strokes equidistant from the segmentation boundary as opposed to graph cuts based approaches [7, 20]. (3) The evaluation does not take into account how the accuracy of the results improves with more user strokes. For instance, one system might only need a single user interaction to reach the ground truth result, while the other might need many interactions to get the same result. Still, both systems will have equal performance under this scheme. These problems of static evaluation make it a poor tool for judging the performance of newly proposed segmentation system.

2.2 User Studies

A user study involves the system being given to a group of participants who are required to use it to solve a set of tasks. The system which is easiest to use and yields the correct segmentation in the least amount of time is considered the best. Examples are [16] where a full user study has been conducted, or [2] where an advanced user has done with each system the optimal job for a few images. However, user studies are very impractical to arrange if thousands of parameters are to be tested.

While overcoming most of the problems of a static evaluation, we have introduced new ones: (1) User studies are expensive and need a large number of participants to be statistically significant. (2) Participants need time to familiarize themselves with the system. For instance, an average driver steering a Formula 1 car for the first time, might be no faster than with a normal car. However, after gaining experience with the car, one would expect the driver to be much faster. (3) Each system has to be evaluated independently by participants, which makes it infeasible to use this scheme in a learning scenario where we are trying to find the optimal parameters of the segmentation system among thousands or millions of possible ones.

2.3 Evaluation using Crowdsourcing

Crowdsourcing has attracted a lot of interest in the machine learning and computer vision communities. This is primarily due to the success of a number of incentive schemes for collecting training data from users on the web. These are either based on money [23], reputation [28], or community efforts [21]. Crowdsourcing has the potential to be an excellent platform for evaluating interactive vision systems such as those for image segmentation. One could ask Mechanical Turk [1] users to cut out different objects in images with different systems. The one who needs the least number of interactions on average might be considered the best. However, this approach too, suffers from a number of problems such as fraud prevention. Furthermore, as in user-studies, it cannot be used for learning in light of thousands or even millions of systems.

2.4 Evaluation with an Active User Model

In this paper we propose a new evaluation methodology which overcomes most of the problems described above. Instead of using a fixed set of interactions, or an army of human participants, our method only needs a model of user interactions. This model is a simple algorithm which – given the current segmentation, and the ground truth – out-

puts the next user interaction. This user model can use simple rules, such as "give a brush stroke in the middle of the largest wrongly labelled region", or alternatively, can be learnt from the interaction logs. We will see that a simple user model exhibits similar behavior as a novice human user. There are many similarities between the problem of learning a user model and the learning of an agent policy in reinforcement learning. Thus, one may exploit reinforcement learning methods for this task. Pros and cons of evaluation schemes are summarized in table 1.

3. SEGMENTATION: PROBLEM SETTING

We use the publicly available GrabCut database of 50 images with known ground truth segmentations[2]. In order to perform large scale testing and comparison, we down-scaled all images to have a maximum size of 241×161, while keeping the original aspect ratio[3]. For each image, we created two different **static** user inputs: (1) A "static trimap" computed by dilating and eroding the ground truth segmentation by 7 pixels[4]. (2) A "static brush" consisting of a few user made brush strokes roughly indicating foreground and background. We used on average about 4 strokes per image. (The magenta and cyan strokes in Fig. 2c give an example). All this data is visualized in figure 1. Note, in Sec. 3.2 we describe a third "dynamic trimap" called the *robot user* simulating the user.

3.1 The Segmentation Systems

We use 4 different interactive segmentation systems in the paper: "GrabCut(GC)", "GC Simple(GCS)", "GC Advanced(GCA)", and "GeodesicDistance (GEO)".

GEO is a very simple system. We first learn Gaussian Mixture Model (GMM) based color models for fg/bg from user made brush strokes. The shortest path in the likelihood ratio yields a segmentation [2].

The systems (GC, GCS, GCA) minimize the energy

$$E(\mathbf{y}) = \sum_{p \in \mathcal{V}} E_p(y_p) + \sum_{(p,q) \in \mathcal{E}} E_{pq}(y_p, y_q) \qquad (1)$$

by graph cuts. Here $(\mathcal{V}, \mathcal{E})$ is an undirected graph whose nodes are pixels p with color x_p and segmentation label $y_p \in \{0, 1\}$, where 0/1 correspond to bg/fg, respectively. We define $(\mathcal{V}, \mathcal{E})$ to be an 8-connected graph.

The unary terms are computed from a probabilistic model for the colors of background ($y_p = 0$) and foreground ($y_p = 1$) pixels using two different GMMs $\Pr(x|0)$ and $\Pr(x|1)$. $E_p(y_p)$ is then computed as $-\log(\Pr(x_p|y_p))$ where x_p contains the three color channels of pixel p. Importantly, GrabCut [20] updates the color models based on the whole segmentation. In practice we use a few iterations only.

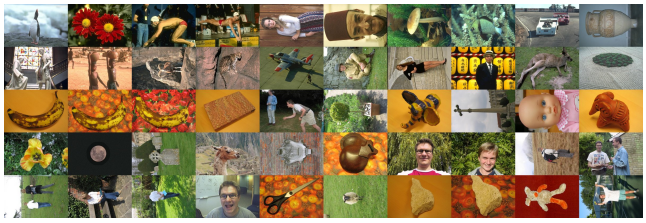
The pairwise term has an Ising and a contrast-dependent component

$$E_{pq}(y_p, y_q) = \frac{|y_q - y_p|}{\text{dist}\,(p,q)} \left(w_i + w_c \exp\left[-\beta \|x_p - x_q\|^2\right]\right)$$

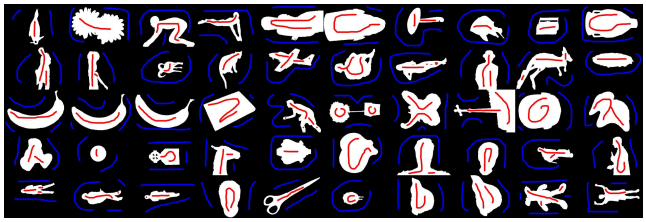
where w_i and w_c are weights for the Ising and contrast-dependent pairwise terms respectively, and β is a parameter with $\beta = 0.5 \cdot w_\beta / \langle \|x_p - x_q\|^2 \rangle$ where $\langle \cdot \rangle$ denotes expectation over an image sample [20]. We can scale β with the parameter w_β.

To summarize, the models have two linear free parameters: w_i, w_c and a single non-linear one: w_β. GC minimizes the energy defined above, and is effectively the original GrabCut system [20]. GCS is a simplified version, where color models (and unary terms) are fixed up front; they are learnt from the initial user brush strokes (see Sec. 3.1) only. GCS will be used in max-margin learning and to check the active user model, but it is not considered as a practical system.

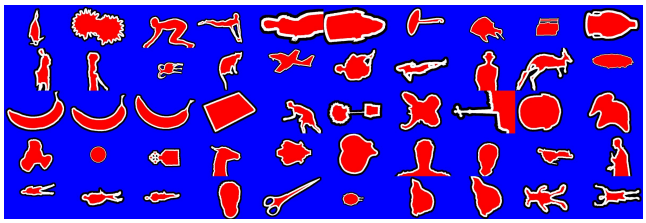
Finally, GCA is an advanced GrabCut system performing considerably better than GC. Inspired by recent work [17], foreground regions are 4-connected to a user made brush stroke to avoid deserted foreground islands. Unfortunately, such a notion of connectivity leads to an NP-hard problem and various solutions have been suggested [27, 19]. However, all these are either very slow and operate on super-pixels [19] or have a very different interaction mechanism [27]. We remove deserted foreground islands in a postprocessing step.



(a) Images $\mathbf{x}^k$



(b) User trimaps $\mathbf{u}^k$



(c) Tight trimaps $\mathbf{u}^k$

Figure 1: We took the 50 GrabCut images (a) with given ground truth segmentations (coded as black/white) and considered two kinds of user inputs (coded as red/blue): User defined strokes (b) and tight trimaps generated by eroding the groundtruth segmentation (c). The user strokes where drawn by looking at the ground truth segmentation $\mathbf{y}^k$ and ignoring the image $\mathbf{x}^k$.

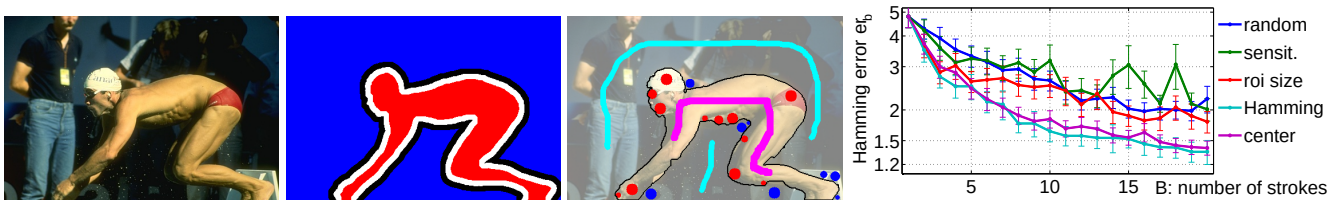
3.2 The Robot User

We start the robot user from an initial fixed set of brush strokes (the "static brush trimap"). The robot user puts

[2]http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm

[3]The quality of the segmentation results is not affected by this down-scaling.

[4]This input is used for both comparison and parameter learning e.g. [6, 22].



(a) Input image (b) Tight trimap (c) Robot user (d) Performance of different robot users

Figure 2: An image from the database (a), tight trimap (b), robot user (red/blue) started from user scribbles (magenta/cyan) with segmentation (black) after B=20 strokes with 1.4% error (c) and performance comparison of different robot users (d).

brushes in the form of dots with a maximum fixed size (here 4 pixel radius). At the boundary, the fixed brush size is scaled down, in order to avoid that the brush straddles the boundary. Fig. 2c shows an example robot user interaction, where red/blue dots are the robot user interactions and cyan/meganta are fixed brushes.

Given the ground truth segmentation $\mathbf{y}^k$ and the current segmentation solution $\mathbf{y}$, the robot user model is a policy $s : (\mathbf{x}^k, \mathbf{y}^k, \mathbf{u}^{k,t}, \mathbf{y}) \mapsto \mathbf{u}^{k,t+1}$ which specifies which brush stroke to place next. Here, $\mathbf{u}^{k,t}$ denotes the user interaction history of image $\mathbf{x}^k$ up to time t. We have investigated various options for this policy: (1) Brush strokes at random image positions. (2) Brush strokes in the middle of the largest, wrongly labelled region (center). For the second strategy, we find the largest connected region of the binary mask, which is given by the absolute difference between the current segmentation and ground truth. We then mark a brush stroke at the pixel which is inside this region and furthest away from the boundary. This is motivated by the observation that users find it hard to mark pixels at the boundary of an object because they have to be very precise.

We also tested user models which took the segmentation algorithm explicitly into account. This is analogous to users who have learnt how the segmentation algorithm works and thus interact with it accordingly. We consider the user model which marks a circular brush stroke at the pixel (1) with the lowest min marginal (sensit.), inspired by [3]. (2) which results in the largest change in labeling (roi size). (3) which decreases the Hamming error by the biggest amount (Hamming). We consider each pixel as the circle center and choose the one where the Hamming error decreases most. This is very expensive, but in some respects is the best solution[5]. "Hamming" acts as a very "perfect user", who knows exactly which interactions (brush strokes) will reduce the error by the largest amount. It is questionable that a user is actually able to find that optimal position.

Fig. 2d shows the performance of 5 different user models (robot users) over a range of 20 brushes. The Hamming error is used to measure the error (see sec. 3.3), which is averaged over all 50 images of our database. Here we used the GCS system, since it is computationally infeasible to apply the (sensit.; roi; Hamming) user models on other interaction systems. GCS allows for efficient computation of solutions by dynamic graph cuts [15]. In the other systems, this is not possible, since unaries change with every brush stroke, and hence we have to treat the system as a black box.

As expected, the random user performs badly. Interestingly the robot users minimizing the energy (roi, sensit.) also perform badly. This is in sharp contrast to [3] where they use the system uncertainty to guide the user scribbles. Our conjecture is that reducing the system uncertainty (active learning) might be less relevant when a user is in the loop. To be precise, a scribble at a position which is certain but wrong might be considerably better than at a position which is uncertain but wrong. Both "Hamming" and "center" are considerably better than the rest. It is interesting to note that "center" is actually only marginally worse than "Hamming". It has to be said that for other systems, e.g. GEO this conclusion might not hold, since e.g. GEO is sensitive to the location of the brush stroke than a system based on graph cut, as [22] has shown.

To summarize, "center" is a user strategy which is motivated from the point of view of a "system-unaware user" (or "novice user") and is computationally feasible. Indeed, in sec. 3.4 we will validate that this strategy correlates quite well with real novice users. We conjecture that the reason is that humans tend to place their strokes in the center of wrongly labeled regions. Also, "center" performed for GCS nearly the same as the optimal strategy "Hamming". Hence, for the rest of the paper we always stick to the user "center" which we call from here onwards our *robot user*.

3.3 The Error Measure

For a static trimap input there are different ways for obtaining an error rate, see [6, 14]. In a static setting, most papers use the number of misclassified pixels (Hamming distance) between the ground truth segmentation and the current result. We call this measure "er_b", i.e. Hamming error for brush b. One could do variations, e.g. [14] weight distances to the boundary differently. Fig. 2d shows how er_b behaves with each interaction.

For learning and evaluation we need an error metric giving us a single score for the whole interaction. One choice is the "weighted" Hamming error averaged over a fixed number of brush strokes B. In particular we choose the error "Er" as: $Er = [\sum_b f(er_b)]/B$. Note, to ensure a fair comparison between systems, B must be the same number for all systems. A simple choice for the weighting function is $f(e) = e$. However, throughout the paper (if not stated differently) we use a quality metric which may match more closely with what the user wants. For this, we use a sigmoid function $f : \mathbb{R}_+ \to [0, c]$, $c = 5$ of the form

$$f(e) = 0, \; e \leq 1.5 \text{ and } f(e) = c - \frac{c}{(e - 0.5)^2}, \; e > 1.5. \quad (2)$$

[5]Note, one could do even better by looking at two or more brushes after each other and then selecting the optimal one. However, the solution grows exponentially with the number look-ahead steps.

Observe that f encodes two facts: all errors below 1.5 are considered negligible and large errors do never weigh more than c. The first reasons of this settings is that visual inspection showed that for most images, an error below 1.5% corresponds to a visually pleasing result. Of course this is highly subjective, e.g. a missing limb from the segmentation of a cow might be an error of 0.5% but is visually unpleasing, or an incorrectly segmented low-contrast area has an error of 2% but is visually not disturbing. The second reason for having a maximum weight of c is that users do not discriminate between two systems giving large errors. Thus errors of 50% and 55% are equally penalized.

Note, ideally we would learn $f(e)$ by a user study.

Due to runtime limitations for parameter learning, we do want to run the robot user for not too many brushes (e.g. maximum of 20 brushes). Thus we start by giving an initial set of brush strokes (cyan/magenta in e.g. fig. 2c) which are used to learn the (initial) colour models. At the same time, we want that most images reach an hamming error level of about 1.5%. A run of the robot user for the GCA system showed that this is possible (i.e. for 68% of images the error is less than 1.5% and for 98% less than 2.5%). We also confirmed that the inital static brush trimap does not affect the learning (see sec. 4) considerably[6].

3.4 Validating the Robot User

We conducted a user study to check our assumption that the robot user is indeed related to a human "novice user" (details of user study are in [18]). We designed an interface which exactly corresponds to the robot user interface, i.e. where the only choice for the human user is to select the position of the circular brush.

We asked 6 people to segment 10 randomly selected images from our database, with each of our 3 systems (GCA, GC, GCS) with reasonable parameters settings (see [18]). For every new image, a system was randomly chosen. We also confirmed that users did not train up for a particular system in the course of the study by asking for multiple segmentations of the same image.

The final error Er (mean $\pm$ std.) averaged over all images and 6 human users is 0.88 ± 0.09 (GCA), 1.20 ± 0.12 (GC), 1.86 ± 0.07 (GCS). It shows a clear correlation with the error of our robot user: 0.88 (GCA), 1.06 (GC), 1.89 (GCS). Fig. 3 depicts the error $f(er_b)$ wrt interaction time.

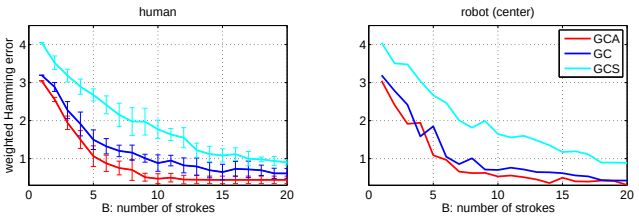


Figure 3: The error rates $f(er_b)$ of 6 humans versus those produced by the robot user.

4. LEARNING BY LINE-SEARCH

Systems with few parameters can be trained by simple line-search. Our systems, GC, GCS, and GCA, have 3 free parameters: w_c, w_i, w_β. Line-search is done by fixing all but one free parameter w_ϕ and simulating the user interaction process for 30 different discrete values $w_{\phi,i}$ of the free parameter w_ϕ over a predefined range. The optimal value w_ϕ^* from the discrete set is chosen to minimize the leave-one-out (LOO) estimate of the test error[7]. Not only do we prevent overfitting but we can also efficiently compute the Jackknife estimator of the variance [29, ch. 8.5.1] – a measure of how certain the optimal parameter is. We run this procedure for all three parameters individually starting from $w_c = 0.1$, $w_i = 0$, $w_\beta = 1$. These initial settings are not very different to the finally learned values, hence we conjecture that initialization is not crucial.[8] One important thing to notice is that our dataset was big enough (and our parameter set small enough) as to not suffer from over-fitting. We see this by observing that training and test error rates are virtually the same for all experiments. In addition to the optimal value we obtain the variance for setting this parameter. In rough words, this variance tells us, how important it is to have this particular value. For instance, a high variance means that parameters different from the selected one, would also perform well. Note, since our error function (Eq. 2) is defined for static and dynamic trimaps, the above procedure can be performed for all three different types of trimaps: "static trimap" (e.g. fig. 1(c)), "static brush" (e.g. fig. 1(b)), "dynamic brush".

In the following we only report results for the two best performing systems, GC and GCA. Table 2 summarizes all the results, and Fig. 4 illustrates some results during training and test (more plots are in [18]). One can observe that the three different trimaps suggest different optimal parameters for each system, and are differently certain about them.

More importantly, we see that the test error is lower when trained dynamically in contrast to static training. This validates our conjecture that an interactive system has to be trained in an interactive way.

Let us look closer at some learnt settings. For system GCA and parameter w_c (see table 2a (first row), and Fig. 4a) we observe that the optimal value in a dynamic setting is lower (0.03) than in any of the static settings. This is surprising, since one would have guessed that the true value of w_c lies somewhere in between the parameters learned with a loose and very tight trimap. This shows that the procedure in [22] is not necessarily correct, where parameter are learned by averaging the performance from two static trimaps. Furthermore, neither the static brush nor the static trimap can be used to guess the settings of all parameters for a dynamic model. For instance, the static "tight trimap" is a quite useful guidance for setting w_c, w_i, but less useful for w_β.[9] To summarize, conclusions about the optimal parameter setting of an interactive system should be drawn by a large set of interaction and cannot be made by looking solely at a few (here two) static trimaps.

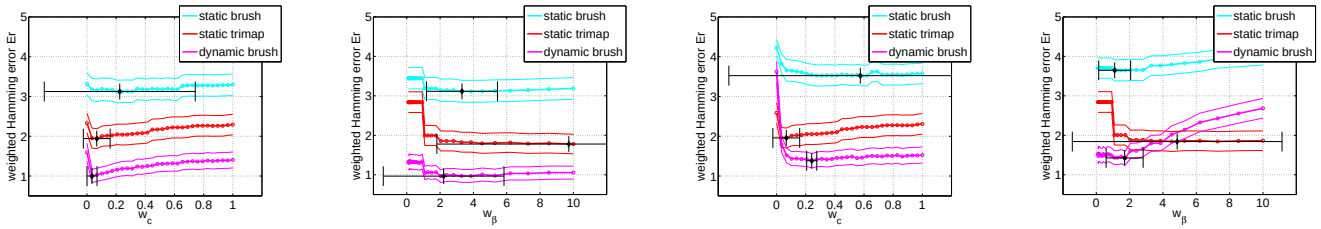
For the sake of completeness, we have the same numbers

[6]We started the learning from no initial brushes and let it run for 60 brush strokes. The learned parameters were similar as with starting from 20 brushes.

[7]This is number-of-data-point-fold cross validation.

[8] However, compared to an exhaustive search over all possible joint settings of the parameters, we are not guaranteed to find the global optimum of the objective function.

[9]Note, the fact that the uncertainty of the "tight trimap" learning is high, gives an indication that this value can not be trusted very much.



(a) GCA, w_c train (b) GCA, w_β train (c) GC, w_c train (d) GC, w_β train

Figure 4: *Line-search. We compare 3 different training procedures for interactive segmentation: Static learning from a fixed set of user brushes, static learning from a tight trimap and dynamic learning with a robot user starting from a fixed set of user brushes. Error Er(± stdev.) for two segmentation systems (GC/GCA) as a function of line-search parameters, here w_c and w_β. The optimal parameter is shown along with its Jackknife variance estimate (black horizontal bar).*

Trimap	w_c	w_i	w_β	Test (Er)
dynamic brush	0.03± 0.03	4.31± 0.17	2.21± 3.62	1.00
static trimap	0.07± 0.09	4.39± 4.40	9.73± 7.92	1.04
static brush	0.22± 0.52	0.47± 8.19	3.31± 2.13	1.19

(a) *System GCA.*

Trimap	w_c	w_i	w_β	Test (Er)
dynamic brush	0.24± 0.03	4.72± 1.16	1.70± 1.11	1.38
static trimap	0.07± 0.09	4.39± 4.40	4.85± 6.29	1.52
static brush	0.57± 0.90	5.00± 0.17	1.10± 0.96	1.46

(b) *System GC.*

Table 2: Optimal parameter values ± stdev. for different systems after line-search for each parameter individually.

for the GC system in table 2b. We see the same conclusions as above. One interesting thing to notice here is that the pairwise terms (esp. w_c) are chosen higher than in GCA. This is expected, since without post-processing a lot of isolated islands may be present which are far away from the true boundary. So post-processing automatically removes these islands. The effect is that in GCA the pairwise terms can now concentrate on modeling the smoothness on the boundary correctly. However, in GC the pairwise terms have to additionally make sure that the isolated regions are removed (by choosing a higher value for the pairwise terms) in order to compensate for the missing post-processing step.

It is interesting to note that for the error metric $f(er_b) = er_b$, we get slightly different values (full results in [18]). For instance, we see that $w_c = 0.07 \pm 0.07$ for GCA with our active user. This is not too surprising, since it says that larger errors are more important (this is what $f(er_b) = er_b$ does). Hence, it is better to choose a larger value of w_c.

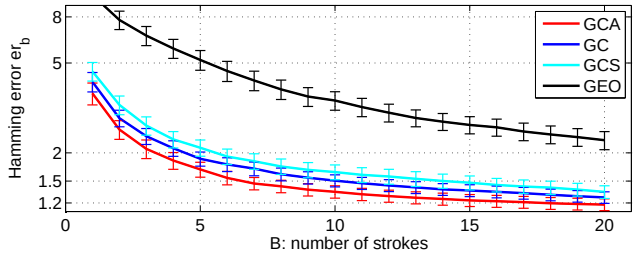


Figure 5: *System comparison*: Segmentation performance of 4 different systems: GCA, GC, GCS and GEO using the robot user. Error er_b averaged over all images.

System Comparison.

Fig. 5 shows the comparison of 4 systems using our robot user. The systems GC, and GCA where trained dynamically. The order of the performances is as expected; GCA is best, followed by GC, then GCS, and GEO. GEO performs badly, since it does no regularization (i.e. smoothing) at the boundary, compared to the other systems. This corresponds with the findings in [13] on a different dataset.

5. MAX-MARGIN LEARNING

The line-search method used in Section 4 can be used for learning models with few parameters only. Max-margin methods deal which models containing large numbers of parameters and have been used extensively in computer vision. However, they work with static training data and cannot be used with an active user model. In this Section, we show how the traditional max-margin parameter learning algorithm can be extended to incorporate an active user.

5.1 Static SVMstruct

Our exposition builds heavily on [24] and the references therein. The SVMstruct framework [26] allows to adjust linear parameters $\mathbf{w}$ of the segmentation energy $E_{\mathbf{w}}(\mathbf{y}, \mathbf{x})$ (Eq. 1) from a given training set $\{\mathbf{x}^k, \mathbf{y}^k\}_{k=1..K}$ of K images $\mathbf{x}^k \in \mathbb{R}^n$ and ground truth segmentations[10] $\mathbf{y} \in \mathcal{Y} := \{0,1\}^n$ by balancing between empirical risk $\sum_k \Delta(\mathbf{y}^k, f(\mathbf{x}^k))$ and regularisation by means of a trade-off parameter C. A (symmetric) loss function[11] $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ measures the degree of fit between two segmentations $\mathbf{y}$ and $\mathbf{y}^*$. The current segmentation is given by $\mathbf{y}^* = \arg\min_{\mathbf{y}} E_{\mathbf{w}}(\mathbf{y}, \mathbf{x})$. We can write the energy function as an inner product between feature functions $\psi_i(\mathbf{y}, \mathbf{x})$ and our parameter vector $\mathbf{w}$: $E_{\mathbf{w}}(\mathbf{y}, \mathbf{x}) = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{y}, \mathbf{x})$. With the two shortcuts $\delta\boldsymbol{\psi}^k_{\mathbf{y}} = \boldsymbol{\psi}(\mathbf{x}^k, \mathbf{y}) - \boldsymbol{\psi}(\mathbf{x}^k, \mathbf{y}^k)$ and $\ell^k_{\mathbf{y}} = \Delta(\mathbf{y}, \mathbf{y}^k)$, the margin rescaled objective [25] reads

[10] We write images of size (n_x, n_y, n_c) as vectors for simplicity. All involved operations respect the 2d grid structure absent in general n-vectors.

[11] We use the Hamming loss $\Delta_H(\mathbf{y}^*, \mathbf{y}^k) = \mathbf{1}^\top |\mathbf{y}^k - \mathbf{y}^*|$.

$$\min_{\boldsymbol{\xi}\geq\mathbf{0},\mathbf{w}} \quad o(\mathbf{w}) := \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{C}{K}\mathbf{1}^\top\boldsymbol{\xi} \qquad (3)$$
$$\text{sb.t.} \quad \min_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}^k}\left\{\mathbf{w}^\top\delta\boldsymbol{\psi}_\mathbf{y}^k - \ell_\mathbf{y}^k\right\} \geq -\xi_k \qquad \forall k.$$

In fact, the convex function $o(\mathbf{w})$ can be rewritten as a sum of a quadratic regulariser and a maximum over an exponentially sized set of linear functions each corresponding to a particular segmentation $\mathbf{y}$. Which energy functions fit under the umbrella of SVMstruct? In principle, in the cutting-planes approach [26] to solve Eq. 3, we only require efficient and exact computation of $\arg\min_\mathbf{y} E_\mathbf{w}(\mathbf{y})$ and $\arg\min_{\mathbf{y}\neq\mathbf{y}^k} E_\mathbf{w}(\mathbf{y}) - \Delta(\mathbf{y},\mathbf{y}^k)$. For the scale of images i.e. $n > 10^5$, submodular energies of the form $E_\mathbf{w}(\mathbf{y}) = \mathbf{y}^\top\mathbf{F}\mathbf{y} + \mathbf{b}^\top\mathbf{y}$, $F_{ij} \geq 0, b_i \in \mathbb{R}$ allow for efficient minimisation by graph cuts. As soon as we include connectivity constraints as in Eq. 1, we can only approximately train the SVMstruct. However some theoretical properties seem to carry over empirically [11].

5.2 Dynamic SVMstruct with "Cheating"

The SVMstruct does not capture the user interaction part. Therefore, we add a third term to the objective that measures the amount of *user interaction* ι where $\mathbf{u}^k \in \{0,1\}^n$ is a binary image indicating whether the user provided the label of the corresponding pixel or not. One can think of $\mathbf{u}^k$ as a partial solution fed into the system by the user brush strokes. In a sense $\mathbf{u}^k$ implements a mechanism for the SVMstruct to *cheat*, because only the unlabeled pixels have to be segmented by our $\arg\min_\mathbf{y} E_\mathbf{w}$ procedure, whereas the labeled pixels stay clamped. In the optimisation problem, we also have to modify the constraints such that the only segmentations $\mathbf{y}$ *compatible* with the interaction $\mathbf{u}^k$ are taken into account. Our modified objective is given by:

$$\min_{\boldsymbol{\xi}\geq\mathbf{0},\mathbf{w},\mathbf{u}^k} \quad o(\mathbf{w},\mathbf{U}) := \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{K}\mathbf{1}^\top\boldsymbol{\xi}+\iota \qquad (4)$$
$$\text{sb.t.} \quad \min_{\mathbf{y}\in\mathcal{Y}|_{\mathbf{u}^k}\setminus\mathbf{y}^k}\left\{\mathbf{w}^\top\delta\boldsymbol{\psi}_\mathbf{y}^k - \ell_\mathbf{y}^k\right\} \geq -\xi_k,\ \iota \geq \mathbf{a}^\top\mathbf{u}^k\ \forall k.$$

For simplicity, we choose the amount of user interaction or cheating ι to be the maximal $\mathbf{a}$-reweighted number of labeled pixels $\iota = \max_k \sum_i a_i|u_i^k|$, with uniform weights $\mathbf{a} = a\cdot\mathbf{1}$. Other formulations based on the *average* rather than on the *maximal* amount of interaction proved feasible but less convenient. We denote the set of all user interactions for all K images $\mathbf{x}^k$ by $\mathbf{U} = [\mathbf{u}^1,..,\mathbf{u}^K]$. The compatible label set $\mathcal{Y}|_{\mathbf{u}^k} = \{0,1\}^n$ is given by $\{\hat{\mathbf{y}} \in \mathcal{Y}|u_i^k = 1 \Rightarrow \hat{y}_i = y_i^k\}$ where $\mathbf{y}^k$ is the ground truth labeling. Note that $o(\mathbf{w},\mathbf{U})$ is convex in $\mathbf{w}$ for all values of $\mathbf{U}$ and efficiently minimisable by the cutting-planes algorithm. However the dependence on $\mathbf{u}^k$ is horribly difficult – we have to find the smallest set of brush strokes leading to a correct segmentation. Geometrically, setting one $u_i^k = 1$ *halves* the number of possible labellings and therefore removes half of the label constraints. The problem (Eq. 5) can be re-interpreted in different ways:
A modified energy $\tilde{E}_{\mathbf{w},\mathbf{v}}(\mathbf{y}) = E_\mathbf{w}(\mathbf{y}) + \sum_{i\in\mathcal{V}} u_i^k\phi_i(y_i,y_i^k)$ with *cheating potentials* $\phi_i(y_i,y_i^k) := \infty$ for $y_i \neq y_i^k$ and 0 otherwise allows to treat the SVMstruct with cheating as an ordinary SVMstruct with modified energy function $\tilde{E}_{\mathbf{w},\mathbf{v}}(\mathbf{y})$ and extended weight vector $\tilde{\mathbf{w}} = [\mathbf{w};\mathbf{u}^1;..;\mathbf{u}^K]$.
A second (but closely related) interpretation starts from the fact that the true label $\mathbf{y}^k$ can be regarded as a *feature vector* of the image $\mathbf{x}^k$[12]. Therefore, it is feature selection in a very particular feature space. There is a direct link to multiple kernel learning – a special kind of feature selection.

5.3 Optimisation with strategies

We explored two approaches to minimise $o(\mathbf{w},\mathbf{U})$. Based on the discrete derivative $\frac{\partial o}{\partial\mathbf{U}}$, we tried coordinate descent. Due to the strong coupling of the variables, only very short steps were possible [13]. Conceptually, the *optimisation* is decoupled from the user interaction, where removal of already known labels from the cheating does not make sense. At every stage of interaction, a user acts according to a *strategy* $s : (\mathbf{x}^k,\mathbf{y}^k,\mathbf{u}^{k,t},\mathbf{y},\mathbf{w}) \mapsto \mathbf{u}^{k,t+1}$. The notion of strategy or policy is also at the core of a robot user. In order to capture the sequential nature of the human interaction and assuming a fixed strategy s, we relax Eq. 4 to

$$\min_{\boldsymbol{\xi}\geq\mathbf{0},\mathbf{w}} \quad o(\mathbf{w},T) := \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{C}{K}\mathbf{1}^\top\boldsymbol{\xi}+\iota \qquad (5)$$
$$\text{sb.t.} \quad \min_{\mathbf{y}\in\mathcal{Y}|_{\mathbf{u}^{k,T}}\setminus\mathbf{y}^k}\left\{\mathbf{w}^\top\delta\boldsymbol{\psi}_\mathbf{y}^k - \ell_\mathbf{y}^k\right\} \geq -\xi_k \qquad \forall k$$
$$\iota \geq \mathbf{a}^\top\mathbf{u}^{k,T},\ \mathbf{u}^{k,T} = s^T(\mathbf{x}^k,\mathbf{y}^k,\mathbf{w}) \qquad \forall k$$

where we denote repeated application of the strategy s by $s^T(\mathbf{x}^k,\mathbf{y}^k,\mathbf{w}) = \bigcirc_{t=0}^{T-1} s(\mathbf{x}^k,\mathbf{y}^k,\mathbf{u}^{k,t},\mathbf{w})$ and by $\bigcirc$ the function concatenation operator. Note that we still cannot properly optimise Eq. 5. However, as a proxy, we develop Eq. 5 forward by starting at $t = 0$ with $\mathbf{u}^{k,0}$. In every step t, we interleave the optimisation of the convex $o(\mathbf{w}^t,t)$ and the inclusion of a new user stroke yielding $\mathbf{w}^T$ as final estimate.

5.4 Experiments

We ran our algorithm on $K = 200$ artificial images of size 40×40 pixels generated by sampling the fg/bg HSV components from independent Gaussian processes with length scales $\ell = 3.5$ pixels. They were combined by a smooth α-map and then squashed through a sigmoid transfer function (see figure 6c). We use a restricted GCS system with three parameters (w_i, w_c, w_u), where w_u is the weight for the unary term[14]. Given the ground truth segmentation we fit a Gaussian mixture model for fore- and background which is not updated during the segmentation process. We used two pairwise potentials (Ising w_i and contrast w_c), a weight on unaries (w_u) and the random robot user with $B = 50$ strokes to train SVMstruct (C=100, 4-neighborhood) on 50 images, keeping 150 test images. Fig. 6b shows, how the weights of the linear parameters varies over time. Edge-blind smoothing (w_i) is switched off, whereas edge-aware smoothing becomes stronger (w_c). The Hamming error on the test set decreases more with dynamically learnt weights.

6. CONCLUSION

This paper showed how robot users can be used to train and evaluate interactive systems. Line-search is used to find good parameters for different interactive segmentation systems under a user interaction model. We also compared the

[12]It is the most informative feature with corresponding predictor given by the identity.
[13]In the end, we can only safely flip a single pixel u_i^k at a time to guarantee descent.
[14]We did not fix w_u to 1, as before, to give the system the freedom to set it to 0.

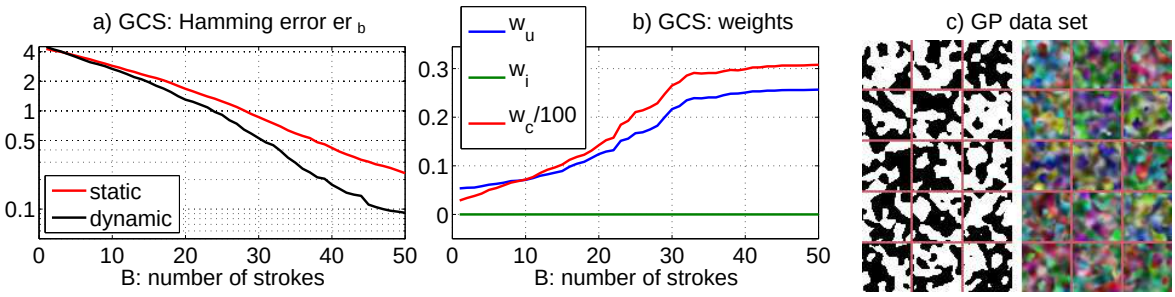


Figure 6: *Max-margin stat/dyn*: a) Segmentation performance using GCS when parameters are either statically or dynamically learnt. b) Evolution of weights $\mathbf{w} = (w_u, w_i, w_c)$ (GMM unary, Ising, contrast) during the optimisation. Static parameters refer to the case where $B = 0$, and dynamically learned parameters are for $B = 50$. c) Sample of the data set used.

performance of the static and dynamic user interaction models. With more parameters, line-search becomes infeasible, leading naturally to the max margin framework.

We introduced an extension to SVMstruct incorporating user interaction models, and showed how to solve the corresponding optimisation. We obtained promising results on a small simulated dataset. The main limitation of the max margin framework is that crucial parts of state-of-the-art segmentation systems (e.g. GCA) cannot be handled. These parts include (1) non-linear parameters, (2) higher-order potentials (e.g. enforcing connectivity) and (3) iterative updates of the unary potentials.

7. REFERENCES

[1] amazon.com. Amazon mechanical turk. `https://www.mturk.com`, 2010.

[2] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, 2007.

[3] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *CVPR*, 2010.

[4] M. Bilgic and L. Getoor. Reflect and correct: A misclassification prediction approach to active inference. *ACM TKDD*, 3(4), 2009.

[5] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[6] A. Blake, C. Rother, M. Brown, P. Perez, and P. Tor. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004.

[7] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001.

[8] O. Duchenne, J.-Y. Audibert, R. Keriven, J. Ponce, and F. Ségonne. Segmentation by transduction. In *CVPR*, 2008.

[9] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. `http://www.pascal-network.org/challenges/VOC`, 2009.

[10] J. A. Fails and J. Dan R. Olsen. Interactive machine learning. In *International Conference on Intelligent User Interfaces*, 2003.

[11] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

[12] L. Grady. Random walks for image segmentation. *PAMI*, 28:1–17, 2006.

[13] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010.

[14] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008.

[15] P. Kohli and P. Torr. Efficiently solving dynamic MRFs using graph cuts. In *ICCV*, 2005.

[16] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *SIGGRAPH*, 23, 2004.

[17] J. Liu, J. Sun, and H.-Y. Shum. Paint selection. In *SIGGRAPH*, 2009.

[18] H. Nickisch, P. Kohli, and C. Rother. Learning an interactive segmentation system. Technical report, `http://arxiv.org/abs/0912.2492`, 2009.

[19] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *CVPR*, 2009.

[20] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 23(3):309–314, 2004.

[21] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77:157–173, 2008.

[22] D. Singaraju, L. Grady, and R. Vidal. P-brush: Continuous valued MRFs with normed pairwise distributions for image segmentation. In *CVPR*, 2009.

[23] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Internet Vision Workshop at CVPR*, 2008.

[24] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.

[25] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML*, 2004.

[26] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *ICML*, 2004.

[27] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008.

[28] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI*, pages 319–326, 2004.

[29] L. Wasserman. *All of Statistics*. Springer, 2004.