Int J Comput Vis (2012) 100:261–274
DOI 10.1007/s11263-012-0537-4

User-Centric Learning and Evaluation of Interactive Segmentation Systems

Pushmeet Kohli · Hannes Nickisch · Carsten Rother · Christoph Rhemann

Received: 23 May 2011 / Accepted: 11 May 2012 / Published online: 30 May 2012

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. In this paper, we study the problem of evaluating and learning interactive segmentation systems which are extensively used in the real world. The key questions in this context are how to measure (1) the effort associated with a user interaction, and (2) the quality of the segmentation result as perceived by the user. We conduct a user study to analyze user behavior and answer these questions. Using the insights obtained from these experiments, we propose a framework to evaluate and learn interactive segmentation systems which brings the user in the loop. The framework 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 systems · Image segmentation · Learning

P. Kohli (✉) · C. Rother
Microsoft Research Cambridge, Cambridge, UK
e-mail: pkohli@microsoft.com

C. Rother
e-mail: carrot@microsoft.com

H. Nickisch
MPI for Intelligent Systems, Tübingen, Germany
e-mail: hannnes@nickisch.org

C. Rhemann
Vienna University of Technology, Vienna, Austria
e-mail: rhemann@ims.tuwien.ac.at

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. In the past, these conditions 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. Interactive systems for generating collages and panoramas of images (Rother et al. 2006) and object cut-and-paste (image segmentation) (Rother et al. 2004) have become particularly among users. 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]

The performance of an interactive system 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

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

been little work in computer vision or machine learning devoted to user-centric *learning* of interactive systems. This paper tries to bridge this gap.

We choose to study the learning and evaluation problems in the context of interactive segmentation systems which are extensively used in the real world. 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.[2] Most work on learning and evaluating interactive segmentation systems assume a fixed input, without considering how real world users interact with the system in practice.

The paper addresses the problem of: (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 conduct a user study to analyze user behavior and answer the key questions of how to measure (a) the effort associated with a user interaction, and (b) the quality of the segmentation result as perceived by the user. Using the insights obtained from these experiments, we propose a framework to evaluate and learn interactive segmentation systems which brings the user in the loop. Although we apply our framework to only interactive segmentation systems, it can be applied to general machine intelligence and computer vision problems.

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 analysis of the behavior of users of interactive segmentation systems. (3) The use of a user model for evaluating and learning interactive systems. (4) A comparison of state-of-the-art segmentation algorithms under an explicit user model. (5) A new algorithm for max-margin learning with user in the loop. Two recent articles (Gulshan et al. 2010; Blake et al. 2011) already employ our robot user to learn and compare various different segmentation algorithms, which demonstrates the usefulness of our approach.

A preliminary version of this paper appeared as (Nickisch et al. 2010). This extended version describes a new user study which provides us with insights on how users measure accuracy of solutions and interaction effort in the context of interactive segmentation systems.

[2]We will refer to each user interaction in this scenario as a *brush stroke*.

Organization of the Paper In Sect. 2, we discuss the problem of interactive system evaluation. In Sect. 3, we give details of our problem setting, and explain the different segmentation systems and datasets considered in our study. In Sect. 4, we describe the artificial user model used in our study. In Sect. 5, we describe the results of our user study which provides us with insights on how users perceive segmentation quality and interaction effort. Section 6 explains the naïve line-search method for learning segmentation system parameters. In Sect. 7, we show how the max-margin framework for structured prediction can be extended to handle interactions, and show some basic results. We conclude by listing some ideas for future work in Sect. 8.

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 (Everingham et al. 2009).

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.

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 (Blake et al. 2004; Singaraju et al. 2009; Duchenne et al. 2008). 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 Fig. 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

Table 1 Comparison of methods

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
Crowdsourcing	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

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 types 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 (Bai and Sapiro 2007; Grady 2006; Singaraju et al. 2009) prefer brush strokes equidistant from the segmentation boundary as opposed to graph cuts based approaches (Boykov and Jolly 2001; Rother et al. 2004). (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 systems.

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 (Mortensen and Barrett 1998) and (Li et al. 2004) where a full user study has been conducted, or (Bai and Sapiro 2007) 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 (Sorokin and Forsyth 2008), reputation (von Ahn and Dabbish 2004), or community efforts (Russell et al. 2008). 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 (amazon.com 2010) 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—outputs the next user interaction. This user model can use simple rules, such as "place 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.

Concurrent with our work, McGuinness and O'Connor (2010, 2011) have also proposed the use of user models to evaluate the performance of interactive image segmentation systems. They have introduced a number of deterministic and stochastic strategies to choose brush strokes. They also reason about more sophisticated models for brush stroke generation. However, unlike our work which looks at both learning and evaluation of interactive systems, the main focus of their paper is on evaluation of these systems. One of their high-level insights, which is similar to our work, is that simple strategies such as choosing the center of the erroneous region for placing the brush stroke performs reasonably well in obtaining accurate segmentations compared to random or more energy aware strategies.

Our framework is also vaguely related to the recent and interesting work of Vijayanarasimhan and Grauman (2011b, 2011a) on active learning for object recognition which contains a user based annotation system within it. In a separate paper (Vijayanarasimhan and Grauman 2009), Vijayanarasimhan and Grauman had looked at the problem of predicting the time taken by a user to annotate any given image, which can be seen as the learning of a implicit user model.

3 Interactive Segmentation Systems: Problem Setting

We now describe in detail the segmentation systems and datasets used in our studies on evaluation and learning interactive systems.

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 (Bai and Sapiro 2007).

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

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

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 (Rother et al. 2004) 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 (Rother et al. 2004). 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 (Rother et al. 2004). 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 Sect. 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 (Liu et al. 2009), 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 (Vicente et al. 2008; Nowozin and Lampert 2009). However, all these are either very slow and operate on super-pixels (Nowozin and Lampert 2009) or have a very different interaction mechanism (Vicente et al. 2008). We remove small disconnected foreground regions in a postprocessing step.

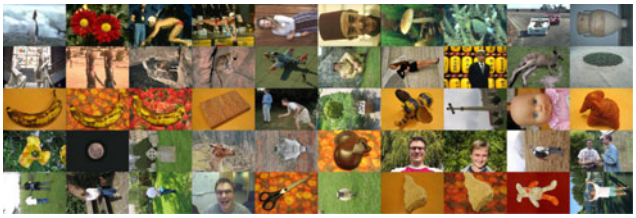
3.2 Datasets for Evaluation

We use the publicly available GrabCut database of 50 images with known ground truth segmentations.[3] 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.[4] 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.[5] (2) A "static brush" consisting of a few user made brush strokes roughly indicating foreground and

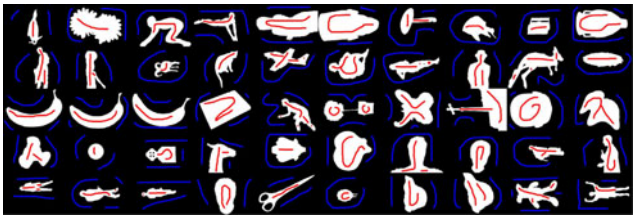
[3] http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm.

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

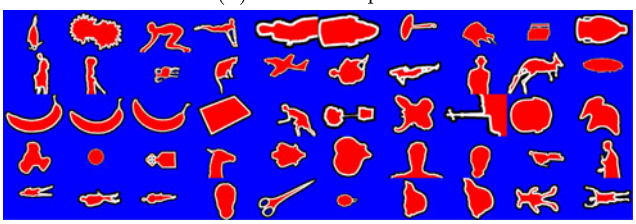
[5] This input is used for both comparison and parameter learning e.g. (Blake et al. 2004; Singaraju et al. 2009).



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



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



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

Fig. 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 $\mathbf{x}^k$

background. We used on average about 4 strokes per image. (The magenta and cyan strokes in Fig. 2(c) give an example.) All this data is visualized in Fig. 1. Note, in Sect. 4 we describe a third "dynamic trimap" called the *robot user* simulating the user.

3.3 The Error Measure

For a static trimap input there are different ways for obtaining an error rate, see (Blake et al. 2004; Kohli et al. 2008). In the static input 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 can also use variations of the Hamming distance measure, e.g. (Kohli et al. 2008) weight distances to the boundary differently. Figure 2(d) 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

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

Observe that f encodes two facts: all errors below 1.5 are considered negligible and large errors 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. A second reason for having a lower limit on the errors considered significant is that for most segmentation problem instances, it is hard to define a single precise ground truth segmentation. This is due to a number of factors including mixed pixels, shadows etc. In their user study for evaluating the performance of the Intelligent Scissors's algorithm for image segmentation, Mortensen and Barrett (1998) compared the results of their algorithm to those of a suite of users (including variation within them) rather than to a fixed "ground truth" that was itself determined by a single user. The 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 that ideally we would learn $f(e)$ by performing 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. 2(c)) which are used to learn the (initial) colour models. At the same time, we want that most images reach a 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 initial static brush trimap does not affect the learning (see Sect. 6) considerably.[6]

4 Evaluation Using a Robot User

We start the robot user from an initial fixed set of brush strokes (the "static brush trimap") such as the one shown

[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.

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

Fig. 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**)

in Fig. 1(b). The robot user puts 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. Figure 2(c) 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 Batra et al. (2010). (2) which results in the largest change in labeling or which maximizes the size of the region of influence (ROI). (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.[7] "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.

Figure 2(d) 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 Sect. 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 (Kohli and Torr 2005). 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 which are guided by the energy (ROI, SENSIT) also perform badly. This is in sharp contrast to (Batra et al. 2010) where they use the system uncertainty to guide the user scribbles. We conjecture that this phenomenon is due to two primary factors. First, a scribble at a position where the labelling is certain but wrong may provide more information to the algorithm than a scribble at a position which is uncertain but wrong. Second, a number of computer vision studies have shown that MRF models used for image segmentation are misspecified i.e. the most probable solutions under these models are not the ground truth solutions (Szeliski et al. 2006).[8] In such cases, providing information that reduces uncertainty of the model might not move it towards the ground truth solution.

Both "Hamming" and "center" strategies for the robot user 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 (Singaraju et al. 2009) 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 Sect. 5 we will validate that this strategy correlates quite well with real novice users. We conjecture that the reason

[7] 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.

[8] This behaviour is also observed in our experiments. Note that after each user interaction we obtain the global optimum of our current energy. Also, note that the energy changes with each user interaction.

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*. Note, that the recent work of (Gulshan et al. 2010) has utilized a very similar type of robot user. We would like to refer the interested reader to their webpage[9] where code and an extend dataset is available online.

5 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 Nickisch et al. 2009). 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.

Our user study had 12 participants out of which 6 participants were familiar with computer vision but had no background knowledge about the tested image segmentation algorithms. The other 6 participants were computer literate but did not have any expertise in computer vision. We asked the participants to segment 10 randomly selected images from our database, with each of our 3 systems (GCA, GC, GCS) with reasonable parameters settings (see Nickisch et al. 2009). 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.

Every user was shown the input image with some initial scribbles (see user interface in Fig. 3) and an initial object outline computed with these scribbles shown as a line of marching ants. Right next to the image we displayed the same image after cutting out the foreground object using the ground truth segmentation. The user was asked to refine the initial segmentation result such that it matches this object outline. To make sure that our analysis was not affected by the choice of initial brush strokes, we created two different set of initial brush strokes for the Grabcut images. The first 6 participants started with a solution obtained by applying the brush stroke set 1, while the next 6 participants started with solutions obtained from the brush stroke set 2.

For refining the object outline, the user could place circular brushes on the image (the radius of the circle was determined as in the robot user). Additionally, we automatically switched between fg and bg (red and blue brush) by using the underlying ground truth segmentation information. Hence, switching between the two brushes was not penalized. The user could place a maximum of 20 brushes per image. If he/she was satisfied with the result before, he/she could press the "Next" button to go to the next image (see Fig. 3).

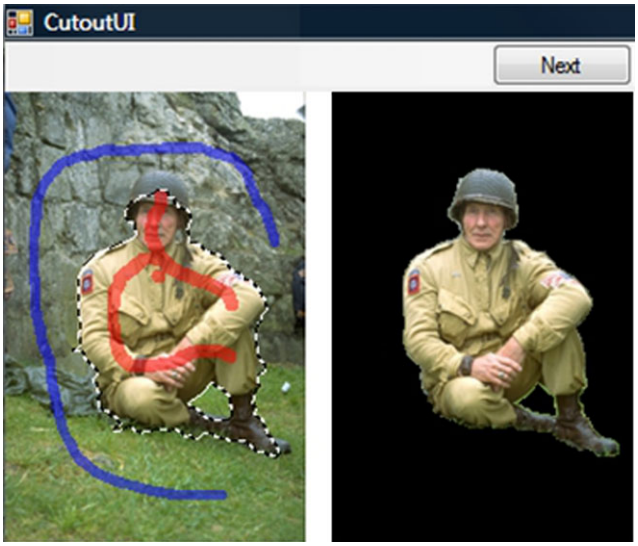


Fig. 3 The user interface for the user study

For each image the segmentation outline was computed with one of the three different segmentation algorithms (GCA, GC or GCS) chosen randomly for each image (to avoid any bias of the user towards an algorithm). We presented every image three times such that every algorithm was applied once per image. Parameter settings for the three algorithms were $w_i = 0$, $w_\beta = 1$, and $w_c = 0.03$ (GCS), $w_c = 0.24$ (GC), $w_c = 0.07$ (GCA). These are reasonable settings and for w_c the same as the learned values in Table 2(a) and 2(b).

Figure 4 depicts the segmentation error $f(er_b)$ with respect to number of interactions. The results suggest a strong similarity between the robot and the human users in that the relative ordering of the performances of the segmentations systems is preserved. Notice that results from both user groups (using brush strokes sets 1 and 2) are similar. We also see that till about 7 brush strokes, the robot user and humans perform quite similarly in terms of absolute error rate. However, after that the robot outperforms the human user and can reach an error rate close to zero. One key difference, which causes this effect, is that humans do stop early when they are satisfied with the results. Hence, the pixel error in segmentations obtained by human subjects flattens out with larger amount of brush strokes and never reaches an error rate of zero.

The final error Er (mean $\pm$ std.) averaged over all images and 6 human users from group 1 is 0.442 ± 0.090 (GCA), 0.610 ± 0.113 (GC), 0.896 ± 0.079 (GCS). It shows a clear correlation with the error of our robot user: 0.00 (GCA), 0.112 (GC), 0.476 (GCS). The corresponding numbers for the group 2 experiment for human participants were:

[9] http://www.robots.ox.ac.uk/~vgg/research/iseg/.

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

(a) *System GCA*.

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

(b) *System GC*.

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

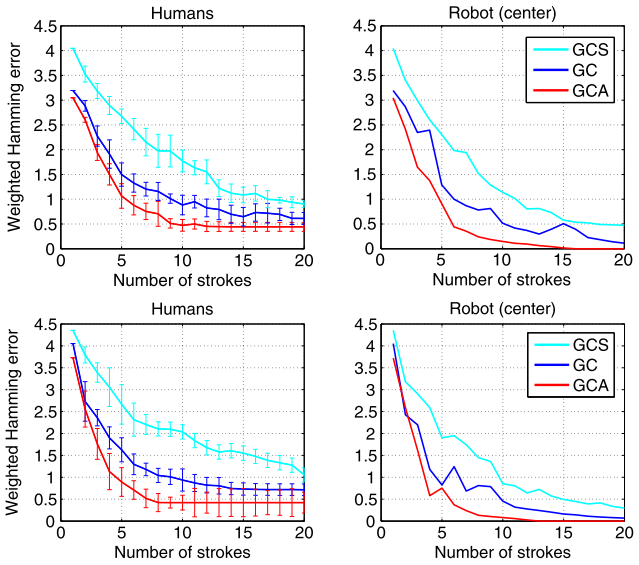


Fig. 4 Comparison of the error rate $f(er_b)$ in segmentation results obtained by the robot user and by 12 human subjects. The subjects were divided into two groups of 6 each which started with segmentations obtained using two different set of initial brush strokes. Results with brush stroke set 1 are shown in the *top row* and with brush stroke set 2 are shown in the *bottom row*. The key point to note is that the relative rankings of the three different segmentation systems is exactly the same under interactions by humans and those made by the robot user. This result supports the case for the use of the robot user to evaluate the performance of the segmentation systems. Also note that the results under the two different sets of initial brush strokes are very similar

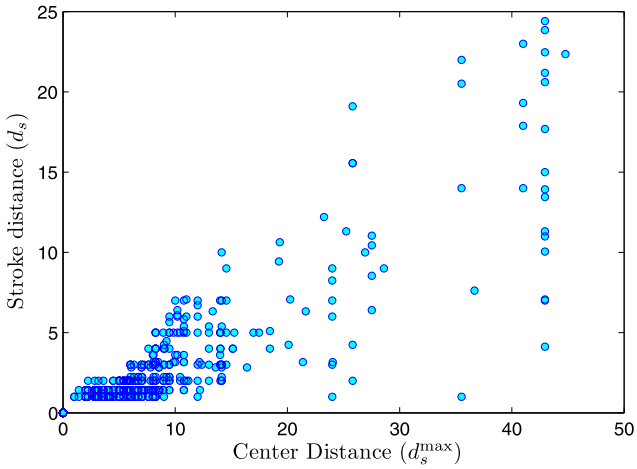


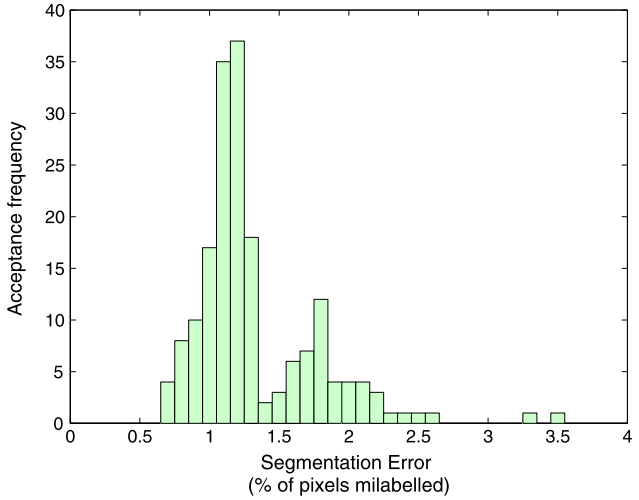
Fig. 5 The figure compares the brush stroke locations of humans with those of the 'center-region' robot. For every brush stroke made by 6 human subjects, we show its minimum distance d_s from the boundary of the erroneous region and its maximum possible value $d_s^{\max}$ which is the minimum distance of the center of the erroneous region from its boundary. Here the erroneous region refers to the erroneous region in the segmentation result before the brush stroke was made

0.422 ± 0.237 (GCA), 0.871 ± 0.132 (GC), 1.055 ± 0.163 (GCS), and for the robot user were: 0.00 (GCA), 0.069 (GC), 0.296 (GCS).

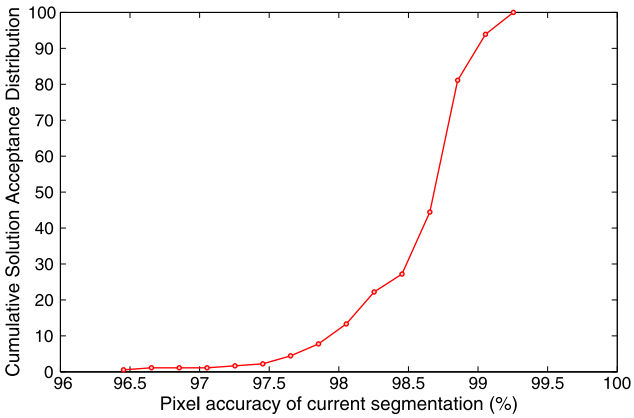
We also analyzed where users place brush strokes in the image and how this compares with the 'center' policy robot user. For this analysis, we looked at all the brush strokes made by the second group of 6 users to segment all images using all segmentation algorithms. In total, there were 1404 brush strokes. Figure 5 shows the distance d_s of each brush stroke s from the boundary of the erroneous region and the maximum distance possible $d_{\max}$ i.e. the distance of the center of the erroneous region from its boundary. Here the erroneous region refers to the erroneous region in the segmentation result before the brush stroke was made. That means if all points are on the diagonal then the user would always place it exactly in the center of the erroneous region. To get an overall sense of the trend, we looked at the average of the ratio of $\frac{d_s}{d_{\max}}$ (0 means at the boundary and 1 in the center of some erroneous region). The average ratios for the different algorithms were: GCS:0.89, GC:0.83, and GCA:0.85.

5.1 Perceptual Accuracy Satisfaction Threshold

One of the most interesting issues in the development of an intelligent interactive segmentation system is the level of segmentation accuracy that satisfies users. To answer this question, and to test the validity of our hypothesis that segmentation errors below a particular threshold do not matter (see Sect. 3.3) we looked at the level of accuracy at which users pressed the "Next" button on the user interface. The results are shown in Fig. 6. The results clearly show that the "next presses" are peaked between 1–1.5 % pixel error, which is consistent with the error metric chosen by us in Sect. 3.3 for our automated scheme for the evaluation of the different interactive systems under different users models.



(a)



(b)

Fig. 6 User satisfaction distribution over different segmentation error rates. (**a**) The histogram shows the number of times a user was satisfied with segmentation of particular pixel error. (**b**) The cumulative distribution of the user satisfaction over solution accuracy

5.2 Measuring Interaction Effort

There has been little work on analyzing the time taken for segmenting objects in images, or making particular brush strokes. A notable exception is the study conducted by Vijayanarasimhan and Grauman (2009) who tried to predict the time taken by users to label complete images.

Our quantitative evaluation of the performance of different interactive systems assumes that all interactions require equal amount of effort. This is not necessarily the case for segmentation systems which uses brush strokes to mark pixels as foreground or background. Brush strokes made near the segmentation boundary require a lot more effort compared to strokes which are made away from the boundary. This is because the user had to make sure that the stroke does not pass through the boundary. To test this hypothesis, we analyzed the time taken by the user to generate any brush stroke. The results of this study can be seen in Fig. 7. It

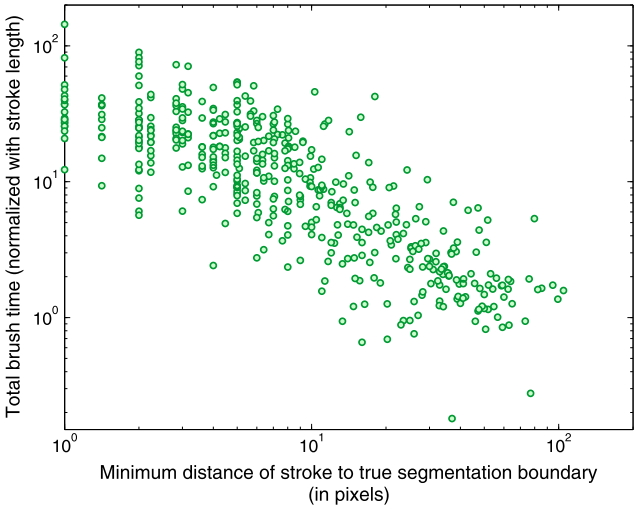


Fig. 7 The figure shows the time taken by the user to generate different brush strokes which are at different distances from the true boundary of the object in the image

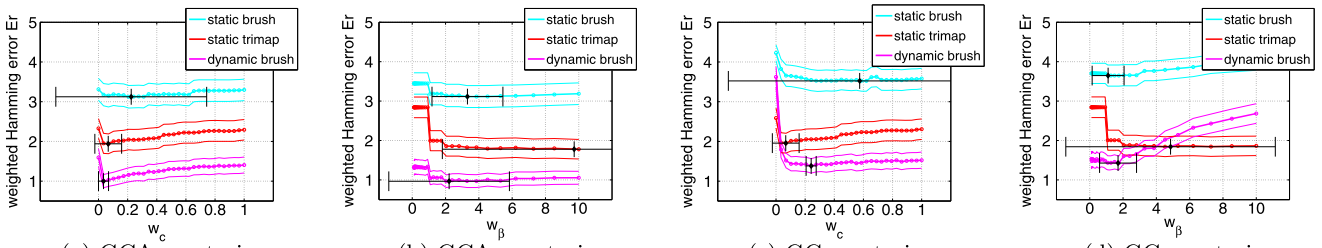
can be seen that strokes made near the segmentation boundary require significant more time labelling pixels on average compared to strokes which were made farther away.

6 Learning by Line-Search

We will now address the problem of learning or estimating the optimal parameters for different interactive segmentation systems. 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.[10] Not only do we prevent overfitting but we can also efficiently compute the Jackknife estimator of the variance (Wasserman 2004, 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.[11] 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

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

[11] 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.



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

Fig. 8 *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 $Er(\pm$ 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*)

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. 8 illustrates some results during training and test (more plots are in Nickisch et al. 2009). 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 2(a) (first row), and Fig. 8(a)) 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 (Singaraju et al. 2009) is not necessarily correct, where parameter are learnt 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.[12] 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.

[12]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.

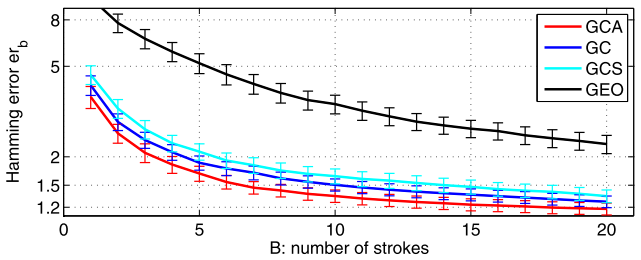


Fig. 9 System comparison: Segmentation performance of 4 different systems: GCA, GC, GCS and GEO using the robot user. Error er_b averaged over all images

For the sake of completeness, we have the same numbers for the GC system in Table 2(b). 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 Nickisch et al. 2009). 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.

System Comparison Figure 9 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 (Gulshan et al. 2010) on a different dataset.

7 Max-Margin Learning

The line-search method used in Sect. 6 can be used for learning models with few parameters only. Max-margin methods (Tsochantaridis et al. 2004; Taskar et al. 2004; Szummer et al. 2008) 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.

The structure of this section is as follows. After reviewing the static case of max-margin learning (Sect. 7.1), we describe the dynamic case (Sect. 7.2) where the user is in the loop. The optimization of the dynamic case is very challenging and we suggest two different heuristic techniques in Sect. 7.3. The latter one, optimization with strategies, is a simple and practical solution which is used in the experimental part in Sect. 7.4.

7.1 Static SVMstruct

Our exposition builds heavily on (Szummer et al. 2008) and the references therein. The SVMstruct framework (Tsochantaridis et al. 2004) 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[13] $\mathbf{y}\in\mathcal{Y}:=\{0,1\}^n$ 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[14] $\Delta:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}_+$ 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 the 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 (Taskar et al. 2004) reads

$$\begin{aligned}&\min_{\boldsymbol{\xi}\geq\mathbf{0},\mathbf{w}}\ o(\mathbf{w}):=\frac{1}{2}\|\mathbf{w}\|^2+\frac{C}{K}\mathbf{1}^\top\boldsymbol{\xi}\\&\text{sb.t.}\ \min_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}^k}\left\{\mathbf{w}^\top\delta\boldsymbol{\psi}^k_{\mathbf{y}}-\ell^k_{\mathbf{y}}\right\}\geq-\xi_k\quad\forall k.\end{aligned}\tag{3}$$

[13] We write images of size $(n_x\times n_y\times n_c)$ as vectors $\in\mathbb{R}^n$, $n=n_xn_yn_z$ for simplicity. All involved operations respect the 2d grid structure absent in general n-vectors.

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

In fact, the *constrained* convex function $o(\mathbf{w})$ can be rewritten as a *unconstrained* function

$$\hat{o}(\mathbf{w})=\frac{1}{2}\|\mathbf{w}\|^2+\max\Big(0,\max_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}^k}\left\{\ell^k_{\mathbf{y}}-\mathbf{w}^\top\delta\boldsymbol{\psi}^k_{\mathbf{y}}\right\}\Big),$$

which is 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? The cutting-planes approach (Tsochantaridis et al. 2004) to solve Eq. (3), only requires 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 carry over empirically (Finley and Joachims 2008).

7.2 Dynamic SVMstruct with "Cheating"

The SVMstruct does not contain the user interaction part a priori. 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 $\arg\min_{\mathbf{y}} E_{\mathbf{w}(\mathbf{y})}$ procedure, whereas the labeled pixels stay clamped. In the optimisation problem, we also have to modify the constraints such that only segmentations $\mathbf{y}$ *compatible* with the interaction $\mathbf{u}^k$ are taken into account. Our modified objective is given by:

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

For simplicity, we choose the amount of user interaction or cheating ι to be the maximal $\mathbf{a}$-reweighted number of labeled pixels $\iota=\max_k\sum_i a_i|u^k_i|$, with uniform weights $\mathbf{a}=a\cdot\mathbf{1}$. In practice we should use different weights for different interactions (as explained in Sect. 5.2).

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 then given by $\{\hat{\mathbf{y}}\in\mathcal{Y}|u^k_i=1\Rightarrow\hat{y}_i=y^k_i\}$ where $\mathbf{y}^k$ is the ground truth labeling. Note that $o(\mathbf{w},\mathbf{U})$ is convex in the weights $\mathbf{w}$ for all values of $\mathbf{U}\in\{0,1\}^{n\times K}$,

hence the global minimiser $\mathbf{w}^*_{\mathbf{U}} = \arg\min_{\mathbf{w}} o(\mathbf{w}, \mathbf{U})$ can efficiently be computed 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 $u^k_i = 1$ *halves* the number of possible labellings and therefore removes half of the label constraints. The problem (Eq. (4)) can be reinterpreted in different ways:

A modified energy $\tilde{E}_{\mathbf{w},\mathbf{v}}(\mathbf{y}) = E_{\mathbf{w}}(\mathbf{y}) + \sum_{i\in\mathcal{V}} u^k_i \phi_i(y_i, y^k_i)$ with *cheating potentials* $\phi_i(y_i, y^k_i) := \zeta |y_i - y^k_i|$ where the constant ζ is sufficiently large $0 \ll \zeta < \infty$ allows to treat the SVMstruct with cheating as an ordinary SVMstruct with a modified energy function $\tilde{E}_{\mathbf{w},\mathbf{v}}(\mathbf{y})$ and an 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*[15] of the image $\mathbf{x}^k$. 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.

7.3 Optimisation—Two Strategies

We explored two approaches to minimise $o(\mathbf{w}, \mathbf{U})$: (i) coordinate descent and (ii) relaxation by *strategies*. Note, that we evaluate experimental (Sect. 7.4) only the latter approach.

The idea of (block) coordinate descent is very simple: minimise one variable (block) at a time; upon convergence, a local minimum is reached. In our case, we interleave running cutting planes $\mathbf{w} \leftarrow \mathbf{w}^*_{\mathbf{U}}$ and label descent[16] $\mathbf{U} \leftarrow \mathbf{U} + \partial o/\partial \mathbf{U}$ using the discrete gradient of the pseudo boolean map $\mathbf{U} \mapsto o(\mathbf{w}, \mathbf{U})$. Even though the gradient $\partial o/\partial \mathbf{U}$ can be evaluated efficiently,[17] we empirically observed that the coupling between the pixels in $\mathbf{U}$ is extremely strong allowing small steps only.[18]

Imitating the user, who incrementally builds up the optimal "cheating" $\mathbf{U}$ by drawing foreground and background brush strokes, we grow $\mathbf{U}$ over time t such that $|\mathbf{U}^t| < |\mathbf{U}^{t+1}|$ that is, we disallow removal of already known labels. At every stage of interaction, a user acts according to a *strategy* $s : (\mathbf{u}^{k,t}, \mathbf{x}^k, \mathbf{y}^k, \mathbf{y}, \mathbf{w}) \mapsto \mathbf{u}^{k,t+1}$. The notion of strategy (or policy) is also at the core the robot user idea. Assuming a fixed strategy s, Eq. (4) becomes

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

where we denote repeated application of the strategy s by $s^T(\mathbf{x}^k, \mathbf{y}^k, \mathbf{w}) = \bigcirc_{t=0}^{T-1} s(\mathbf{u}^{k,t}, \mathbf{x}^k, \mathbf{y}^k, \mathbf{w})$ and by $\bigcirc$ the function concatenation operator. This is a relaxation in two ways: On the one hand $\mathbf{U}$ grows monotonically and on the other hand the globally optimal strategy is replaced by the proxy s—our robot user. The optimisation of Eq. (5) works by starting at $t = 0$ with $\mathbf{u}^{k,0}$ the initial brush strokes. In every step t, we compute $\mathbf{w}^t \leftarrow \mathbf{w}^*_{\mathbf{U}^t}$ using cutting planes and update the cheating using the strategy $\mathbf{u}^{k,t+1} \leftarrow s(\mathbf{u}^{k,t}, ..)$ of the robot user. As a result, we obtain a sequence of weights $\mathbf{w}^t$ with $\mathbf{w}^T$ being the solution to Eq. (5). In summary, the above procedure can be seen as iteratively applying static SVMstruct, where after each iteration the robot user places a new brush stroke, based on the current results.

The overall computational cost is T times the cost of an individual cutting plane optimisation.

7.4 Experiments

We now report the results of our experiments with the dynamic SVMstruct algorithm i.e. the optimization of Eq. (5) using strategies. 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 Fig. 10(c)). 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.[19] 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. Figure 10b 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). Our experiments indicate that the Hamming error on the test set decreases more with dynamically learnt weights.

[15] It is in fact the most informative feature with corresponding predictor given by the identity.

[16] To our knowledge, there is no simple graph cut like algorithm to do the minimisation in $\mathbf{U}$ all at once.

[17] The cost is K runs of dynamic graphcuts of size n, though.

[18] In the end, we can only safely flip a single pixel u^k_i at a time to guarantee descent.

[19] We did not fix w_u to 1, as before, to give the system the freedom to set it to 0.

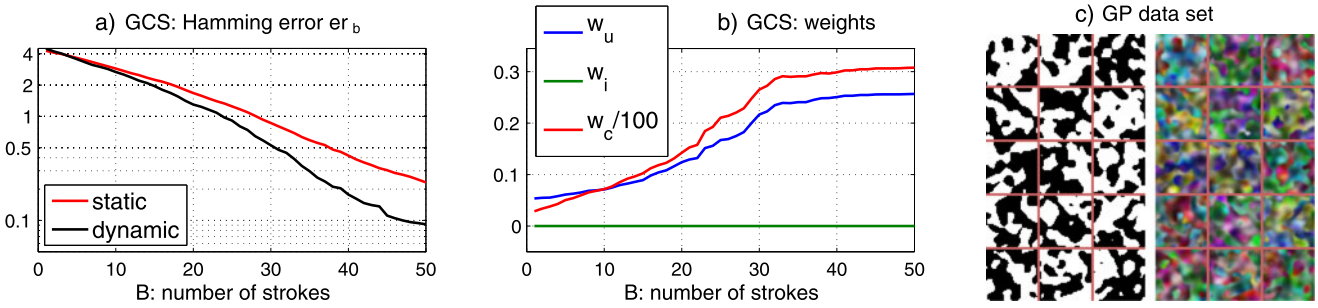


Fig. 10 *Max-margin stat/dyn*: (**a**) Segmentation performance using GCS when parameters are either statically or dynamically learnt. (**b**) Average evolution of weights $\mathbf{w} = (w_u, w_i, w_c)$ (GMM unary, Ising, contrast) during the optimisation. Static parameters refer to the case where $B = 0$, and dynamically learned parameters are for $B = 50$. (**c**) Sample of the data set used

8 Conclusion

This paper addressed the problem of evaluating and learning interactive intelligent systems. We performed a user study for evaluating different interactive segmentation systems which provided us with new insights on how users perceive segmentation accuracy and interaction effort. We showed these insights can be used to build a robot user which can be used to train and evaluate interactive systems.

We showed how a simple line-search algorithm can be used to find good parameters for different interactive segmentation systems under a user interaction model. We also compared the performance of the static and dynamic user interaction models. With more parameters, line-search becomes infeasible, leading naturally to the max margin framework. To overcome this problem, we introduced an extension to SVMstruct which incorporates 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.

The evaluation and learning of intelligent interactive systems has been relatively ignored by the machine learning and computer vision communities. With this paper, our aim is to inspire discussion and new research on this very important problem.

Acknowledgement Christoph Rhemann was supported by the Vienna Science and Technology Fund (WWTF) under project ICT08-019.

References

amazon.com (2010). Amazon mechanical turk. https://www.mturk.com

Bai, X., & Sapiro, G. (2007). A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*.

Batra, D., Kowdle, A., Parikh, D., Luo, J., & Chen, T. (2010). iCoseg: interactive co-segmentation with intelligent scribble guidance. In *CVPR*.

Blake, A., Rother, C., Brown, M., Perez, P., & Tor, P. (2004). Interactive image segmentation using an adaptive GMMRF model. In *ECCV*.

Blake, A., Kohli, P., & Rother, C. (2011). *Markov random fields for vision and image processing*. Cambridge: MIT Press.

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

Duchenne, O., Audibert, J. Y., Keriven, R., Ponce, J., & Ségonne, F. (2008). Segmentation by transduction. In *CVPR*.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., & Zisserman, A. (2009). http://www.pascal-network.org/challenges/VOC

Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *ICML*.

Grady, L. (2006). Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*, 1–17.

Gulshan, V., Rother, C., Criminisi, A., Blake, A., & Zisserman, A. (2010). Geodesic star convexity for interactive image segmentation. In *CVPR*.

Kohli, P., Ladicky, L., & Torr, P. (2008). Robust higher order potentials for enforcing label consistency. In *CVPR*.

Kohli, P., & Torr, P. (2005). Efficiently solving dynamic MRFs using graph cuts. In *ICCV*.

Li, Y., Sun, J., Tang, C. K., & Shum, H. Y. (2004). Lazy snapping. In *SIGGRAPH* (Vol. 23).

Liu, J., Sun, J., & Shum, H. Y. (2009). Paint selection. In *SIGGRAPH*.

McGuinness, K., & O'Connor, N. E. (2010). A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, *43*(2), 434–444.

McGuinness, K., & O'Connor, N. E. (2011). Toward automated evaluation of interactive segmentation. In *CVIU*.

Mortensen, E. N., & Barrett, W. A. (1998). Interactive segmentation with intelligent scissors. In *Graphical models and image processing*.

Nickisch, H., Kohli, P., & Rother, C. (2009). *Learning an interactive segmentation system* (Tech. rep.). http://arxiv.org/abs/0912.2492

Nickisch, H., Rother, C., Kohli, P., & Rhemann, C. (2010). Learning and evaluating interactive segmentation systems. In *ICVGIP*.

Nowozin, S., & Lampert, C. H. (2009). Global connectivity potentials for random field models. In *CVPR*.

Rother, C., Bordeaux, L., Hamadi, Y., & Blake, A. (2006). Autocollage. *ACM Transactions on Graphics*, *25*(3), 847–852.

Rother, C., Kolmogorov, V., & Blake, A. (2004). "GrabCut"—interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*.

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*, 157–173.

Singaraju, D., Grady, L., & Vidal, R. (2009). P-brush: Continuous valued MRFs with normed pairwise distributions for image segmentation. In *CVPR*.

Sorokin, A., & Forsyth, D. (2008). Utility data annotation with amazon mechanical turk. In *Internet vision workshop at CVPR*.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., & Rother, C. (2006). A comparative study of energy minimization methods for Markov random fields. In *ECCV*.

Szummer, M., Kohli, P., & Hoiem, D. (2008). Learning CRFs using graph cuts. In *ECCV*.

Taskar, B., Chatalbashev, V., & Koller, D. (2004). Learning associative Markov networks. In *ICML*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector learning for interdependent and structured output spaces. In *ICML*.

Vicente, S., Kolmogorov, V., & Rother, C. (2008). Graph cut based image segmentation with connectivity priors. In *CVPR*.

Vijayanarasimhan, S., & Grauman, K. (2009). What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *CVPR* (pp. 2262–2269).

Vijayanarasimhan, S., & Grauman, K. (2011a). Cost-SENSITive active visual category learning. *International Journal of Computer Vision*, *91*(1), 24–44.

Vijayanarasimhan, S., & Grauman, K. (2011b). Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*.

von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. In *SIGCHI* (pp. 319–326).

Wasserman, L. (2004). *All of statistics*. Berlin: Springer.