Active Online Learning for Interactive Segmentation Using Sparse Gaussian Processes

Rudolph Triebel, Jan Stühmer, Mohamed Souiai, and Daniel Cremers

Computer Vision Group, Dep. of Computer Science TU Munich,
Boltzmannstrasse 3, 85748 Garching, Germany
{rudolph.triebel,jan.stuehmer,mohamed.souiai}@in.tum.de
cremers@tum.de

Abstract. We present an active learning framework for image segmentation with user interaction. Our system uses a sparse Gaussian Process classifier (GPC) trained on manually labeled image pixels (user scribbles) and refined in every active learning round. As a special feature, our method uses a very efficient online update rule to compute the class predictions in every round. The final segmentation of the image is computed via convex optimization. Results on a standard benchmark data set show that our algorithm is better than a recent state-of-the-art method. We also show that the queries made by the algorithm are more informative compared to randomly increasing the training data, and that our online version is much faster than the standard offline GPC inference.

1 Introduction

Automatic image segmentation is one of the most important problems in computer vision. Its attractiveness stems from its very large range of applications, including medical imaging and robotics. However, in general the image segmentation problem is ill-posed, because a correct segmentation depends strongly on the application. Therefore, we focus on the *interactive* segmentation problem, where the user provides information about the regions to be segmented, e.g. by manually sampling image pixels and assigning them to a predefined region class. These *user scribbles* are used as ground truth information, and the aim is to infer a good segmentation using these scribbles as constraints on the labelling. To do this, many approaches have been presented in the literature with impressive results. However, current methods can reach high classification rates only by requiring comparably many user scribbles, and the number of user scribbles needed usually grows very fast as the segmentation quality approaches 100%.

In this paper, we present a method that asks for user input more intelligently by actively querying pixels to be labeled where the classification was made with high uncertainty. This way, only a few user scribbles are needed to obtain a high quality segmentation. Our method uses an efficient sparse Gaussian Process classifier (GPC) to learn background and foreground models, providing an accurate estimation of the classification uncertainty. We also present a very efficient way to compute the class predictions on every round using an online update rule.

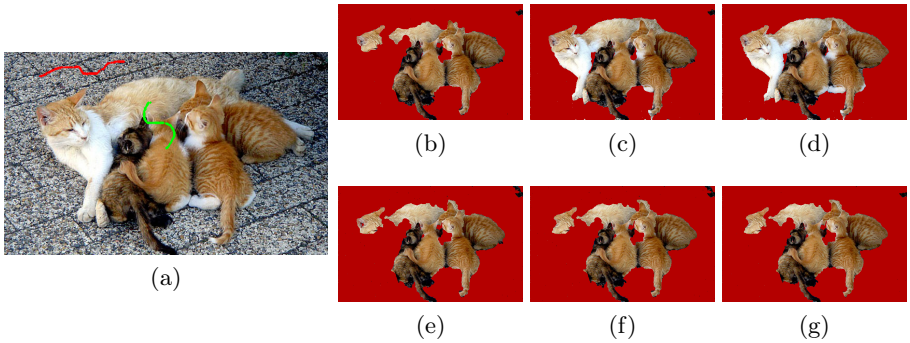
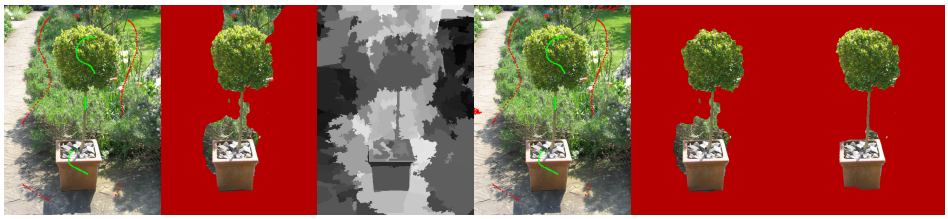


Fig. 1: Comparison between the Parzen window estimator [8] and our sparse GP classifier for foreground classification. From the initial scribble image (a) both approaches learn a model for the foreground. As none of the scribble pixels for the foreground class is white, both approaches fail to classify the white neck of the cat correctly (b, e). However, in the next active learning round, the GP manages to query this part from the user based on its accurate estimation of the predictive uncertainty (c). In contrast, the Parzen window estimator does not query this part, because its uncertainty is low despite its incorrect classification, i.e. it is over-confident (f). After 6 rounds the GP achieves a very good segmentation (d), while the Parzen window estimator still gives a lower-quality segmentation (g).

1.1 Related Work

Many previous works use energy minimization for image segmentation, and since the work of Boykov *et al.* [1], intensive research, e.g. [13, 7], has been done on embedding the input image onto a discrete lattice and computing a segmentation using the min-cut framework. Another line of work [16, 8] models segmentation in the continuous domain and is based on the convex relaxation technique of Nikolova *et al.* [9]. Both discrete and continuous approaches impose spatial consistency as a prior on the image labelling. Our work is related to [8] where the data term describing the pixel class probabilities includes spatial information while estimating the colour distribution using a Parzen window estimator. However, we use an Informative Vector Machine (IVM) [5], a sparse version of the Gaussian Process Classifier, and employ active learning, which improves the segmentation result quickly after only a few training rounds (see Fig. 1). In contrast to the sparse GP algorithm of Csató and Opper [2], the IVM has advantages in the context of active learning, mainly due to the information-theoretic criterion used to select the subset of the training points.

In the field of active learning, Kapoor *et al.*[4] address object categorization using a GP classifier (GPC) where data points possessing large uncertainty (using posterior mean and variance) are queried for labels and used to improve classification. Triebel *et al.* [15] use an IVM to actively learn traffic lights in urban traffic images. Here, we use a similar approach, but with a very efficient online update method for the classification step of the GPC. Vezhnevets *et al.* [17], as well as



(a) (b) (c) (d) (e) (f)

Fig. 2: Example sequence of our proposed active learning framework. The algorithm starts with initial user scribbles as shown in (a). It then learns a sparse GP classifier and segments the image using the GP prediction and a regularization term (b). Then, candidate regions for new, informative user scribbles are computed (c). These are based on the normalized entropy of the GP prediction, i.e. bright regions represent a higher classification uncertainty than darker regions. In this case, a segment at the upper right border is chosen. A label is queried for these pixels (here it is background), and a sub-set of uniformly sampeled pixels together with the class labels is added to the training data (d). In the next round, the classification is improved and the result is refined (e). After a few rounds (here 4 in total), the final segmentation is obtained (f).

Wang *et al.* [18] also use active learning for interactive image segmentation, but either with a CRF+NaiveBayes [17] or a Gaussian Mixture Model (GMM) [18] as an underlying classifier. We use a GPC, because it is non-parametric, i.e. it does not assume a functional model for the data, and it was shown to provide very accurate uncertainty estimates, which is crucial in active learning.

2 Algorithm Overview

Fig. 2 shows an example sequence of our active learning framework for interactive image segmentation. From a set of initial user scribbles from both foreground and background regions (Fig 2a), our algorithm learns a sparse Gaussian Process Classifier (GPC) and classifies the remaining pixels. Then, a segmentation is obtained using regularization (Fig. 2b), and an uncertainty measure is computed from the predictive variance returned by the GPC. We use a GPC, because its uncertainty estimates are more reliable than those produced by other learning methods such as Support Vector Machines, where reliable refers to a strong correlation between uncertain and incorrectly classified samples (see, e.g., [10]). Then, we perform an over-segmentation of the original image based on super-pixels [3] and compute the average classification uncertainty (entropy) for each segment (see Fig. 2c). In the next step, the algorithm selects the segment with the highest uncertainty to query a ground truth label from the user, samples pixels uniformly from the segment, and adds the samples with the obtained labels to the training data set (see Fig. 2d). Note that, due to imperfections in the

segmentation, some segments can contain both foreground and background pixels. In that case, the user can select a "don't know" option, and the next segment is chosen in the order of decreasing entropies. This however, occurs only rarely when the segmentation is done sufficiently fine-grained. The whole learning and classification process is then repeated for a fixed number of times or until an appropriate stopping criterion is met (Fig. 2e and 2f).

3 Gaussian Process Classification

Every round of our active learning algorithm starts by training a Gaussian Process Classifier (GPC) on the current set of user scribbles. If we denote the scribbles as pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, where $\mathbf{x}_i$ are feature vectors[1] and $y_i \in \{-1, 1\}$ are binary labels denoting background or foreground, then the task is to compute a *predictive distribution* $p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*)$. Here, $(\mathbf{x}_*, y_*)$ is an unseen pixel/label pair, $\mathcal{X}$ the set of all training pixels, and $\mathbf{y}$ the training labels. To compute the predictive distribution, the GPC first estimates a distribution $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})$ over the *latent variables* $\mathbf{f} \in \mathbb{R}^N$, approximating it with a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ, i.e.: $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) \approx \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \Sigma)$. This is done using Bayes' rule:

$$p(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathcal{X})}{\int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathcal{X}) d\mathbf{f}}, \tag{3.1}$$

where $p(\mathbf{f} \mid \mathcal{X}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, K)$ is the prior of the latent variables, and

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_i p(y_i \mid f_i) \tag{3.2}$$

are the likelihoods, which are conditionally independent. These likelihoods are determined using a *sigmoid function* Φ, i.e. $p(y_i \mid f_i) = \Phi(y_i f_i)$, which has the effect that Eq. (3.1) cannot be computed in closed form. Here, Expectation Propagation (EP) and Assumed Density Filtering (ADF) are commonly used approximations based on a Gaussian $q(y_i \mid f_i)$ that minimises the Kullback-Leibler (KL) divergence between $q(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathcal{X})$ and the numerator of Eq. (3.1).

Then, for a given new test data point $\mathbf{x}_*$, the GP classifier computes the mean μ_* and the variance σ_*^2 of the latent variable distribution

$$p(f_* \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_* \mid \mathcal{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) d\mathbf{f} \tag{3.3}$$

and uses that to compute the predictive distribution

$$p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \int \Phi(f_*) p(f_* \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) df_*. \tag{3.4}$$

[1] These can be either RGB pixel values or a combination of image coordinates and RGB values of the pixels. In our implementation, we use the latter, because it also provides locality information about background and foreground.

If Φ is the cumulative Gaussian function this can be done in closed form using

$$p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \Phi\left(\frac{\mu_*}{\sqrt{1+\sigma_*^2}}\right). \tag{3.5}$$

3.1 Information-theoretic Sparsification

One problem with the GPC is its huge demand of memory and run time, because it maintains an $N \times N$ covariance matrix, and the number of training samples N can be very large. Therefore, we use a sparsification known as the Informative Vector Machine (IVM) [6]. The main idea here is to only use a sub-set of training points denoted the *active set* $\mathcal{I}_D$, from which an approximation q of the posterior is computed. As above, q is Gaussian, i.e. $q(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \Sigma)$. The IVM computes $\boldsymbol{\mu}$ and Σ incrementally, i.e. in step j a new $\boldsymbol{\mu}_j$ and Σ_j are computed:

$$\boldsymbol{\mu}_j = \boldsymbol{\mu}_{j-1} + \Sigma_{j-1}\mathbf{g}_j \tag{3.6}$$

$$\Sigma_j = \Sigma_{j-1} - \Sigma_{j-1}(\mathbf{g}_j\mathbf{g}_j^T - 2\Gamma_j)\Sigma_{j-1} \tag{3.7}$$

where

$$\mathbf{g}_j = \frac{\partial \log Z_j}{\partial \boldsymbol{\mu}_{j-1}}, \qquad \Gamma_j = \frac{\partial \log Z_j}{\partial \Sigma_{j-1}}, \tag{3.8}$$

and Z_j is the approximation to the normalizer in Eq. (3.1) using the estimate q_j. Initially, $\boldsymbol{\mu}_0 = \mathbf{0}$, and $\Sigma_0 = K$, where K is the prior GP covariance matrix. Then, at iteration j the training point $(\mathbf{x}_k, y_k)$ that maximizes the entropy difference between q_{j-1} and q_j is selected into the active set. The algorithm stops when the active set has reached a desired size D. In our implementation, we choose D as a fixed fraction of N.

Due to a circular dependence between $\mathcal{I}_D$ and the kernel hyper parameters θ, the IVM training algorithm loops a given number of times over two steps: estimation of $\mathcal{I}_D$ from θ and minimizing the *marginal likelihood* Z_D using $\partial Z_D/\partial\theta$, thereby keeping $\mathcal{I}_D$ fixed. Although there are no convergence guarantees, in practice a few iterations are sufficient to find good kernel hyper-parameters.

4 Online Update of the IVM

In addition to its sparsity, the IVM differs from the standard GP also by its ability to compute the posterior distribution $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})$ *incrementally*. Thus, the algorithm loops over all active points and updates mean vector $\boldsymbol{\mu}$ and covariance matrix Σ by increasing their lengths in every iteration. In particular, it keeps the lower triangular matrix L_d of a Cholesky decompositon in memory and updates it using rank-1 Cholesky updates, where L_d is of size $d \times d$ and $d = 1, \ldots, D$. Further details of this procedure are given in Algorithm 1 of Lawrence *et al.*[6]. For our purpose, this incremental scheme is particularly useful, because it avoids the complete re-computation of the GP parameters in every training round and adds only a fixed number of rows and columns to L_d. This decreases the training time substantially, as we show below. For an efficient *class prediction*, we furthermore propose a novel online update rule, as described next.

4.1 Online Computation of the Class Prediction

To predict a class label y_* for a new test data point $\mathbf{x}_*$, the IVM computes the mean μ_* and the covariance σ_* of the approximation to the predictive distribution given in Eq. (3.3), and uses them to obtain the class probability (Eq. (3.5)). With the notation of Rasmussen and Williams [12], this can be expressed as

$$\mu_* = \mathbf{k}_*^T (K + \tilde{\Sigma})^{-1} \tilde{\boldsymbol{\mu}} \tag{4.1}$$

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \tilde{\Sigma})^{-1} \mathbf{k}_*, \tag{4.2}$$

where $\tilde{\boldsymbol{\mu}}$ and $\tilde{\Sigma}$ are the *site parameters* of the approximate Gaussian likelihood $q(\mathbf{y} \mid \mathbf{f})$, K is the prior covariance matrix, i.e. the kernel function k applied to all pairs of training points $\mathbf{x}_1, \ldots, \mathbf{x}_N$, and $\mathbf{k}_* = (k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N))$. Note that $\mathbf{k}_*$, $\tilde{\boldsymbol{\mu}}$, and $B := K + \tilde{\Sigma}$ are only computed for D active points with $D < N$.

In general, Eqs. (4.1) and (4.2) have to be computed completely anew for every new test point $\mathbf{x}_*$, and it is usually unlikely to observe the same test point again. In active learning, this means that the complexity of making predictions increases quadratically with the training rounds, because in every training round the matrix B is larger due to the additional active points in the training data. However, for interactive image segmentation, we can use the fact that class predictions are made on the same pixels (i.e. test points) in every round. This means that $\mathbf{k}_{*,t}$ from round t can be obtained from $\mathbf{k}_{*,t-1}$ of the previous round by appending the covariances $k(\mathbf{x}_*, \mathbf{x}_{D_{t-1}+1}), \ldots, k(\mathbf{x}_*, \mathbf{x}_{D_t})$ between $\mathbf{x}_*$ and the new active points, where D_t is the total number of active points in round t. This can be used to compute $\mu_{*,t}$ and $\sigma_{*,t}$ incrementally from $\mu_{*,t-1}$ and $\sigma_{*,t-1}$. To do this, we note that B_t is given by its Cholesky decomposition $L_t L_t^T$, and

$$L_t := \begin{pmatrix} L_{t-1} & 0 \\ A & L_+ \end{pmatrix}, \tag{4.3}$$

where L_+ is lower-triangular. To compute B_t^{-1}, we use

$$B_t^{-1} = \begin{pmatrix} L_{t-1} L_{t-1}^T & L_{t-1} A^T \\ A L_{t-1}^T & A A^T + L_+ L_+^T \end{pmatrix}^{-1}, \tag{4.4}$$

and compute the Schur complement as

$$S = A A^T + L_+ L_+^T - A L_{t-1}^T (L_{t-1} L_{t-1}^T)^{-1} L_{t-1} A^T = L_+ L_+^T.$$

With this, we obtain

$$B_t^{-1} = \begin{pmatrix} C & -L_{t-1}^{-T} A^T S^{-1} \\ -S^{-1} A L_{t-1}^{-1} & S^{-1} \end{pmatrix}, \tag{4.5}$$

where $C = (L_{t-1} L_{t-1}^T)^{-1} + L_{t-1}^{-T} A^T S^{-1} A L_{t-1}^{-1}$. We now formulate Eq. (4.2) as:

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \begin{pmatrix} \mathbf{k}_{*,t-1} & \mathbf{k}_{*,+} \end{pmatrix} B_t^{-1} \begin{pmatrix} \mathbf{k}_{*,t-1} \\ \mathbf{k}_{*,+} \end{pmatrix}, \tag{4.6}$$

where $\mathbf{k}_{*,+}$ is the vector of newly added covariances in round t. Plugging Eq. (4.5) into Eq. (4.6) we obtain for the rightmost term r of Eq (4.6):

$$r = \hat{\mathbf{k}}_{t-1}^T \hat{\mathbf{k}}_{t-1} + \hat{\mathbf{k}}_{t-1}^T A^T S^{-1} A \hat{\mathbf{k}}_{t-1} - 2\hat{\mathbf{k}}_{t-1}^T A^T S^{-1} \mathbf{k}_{*,+} + \mathbf{k}_{*,+}^T S^{-1} \mathbf{k}_{*,+} \quad (4.7)$$

where $\hat{\mathbf{k}}_{t-1} = L_{t-1}^{-1} \mathbf{k}_{*,t-1}$. It follows that the first term of r in Eq. (4.7) and the first term in Eq. (4.6) define the predictive variance of the previous round $\sigma_{*,t-1}$

$$\sigma_{*,t-1} = k(\mathbf{x}_*, \mathbf{x}_*) - \hat{\mathbf{k}}_{t-1}^T \hat{\mathbf{k}}_{t-1}, \quad (4.8)$$

whereas the remaining terms of r can be subsumed into

$$(L_+^{-1} \mathbf{k}_{*,+} - L_+^{-1} A \hat{\mathbf{k}}_{t-1})^T (L_+^{-1} \mathbf{k}_{*,+} - L_+^{-1} A \hat{\mathbf{k}}_{t-1}), \quad (4.9)$$

which simplifies into

$$(L_+^{-1} \Delta\mathbf{k})^T (L_+^{-1} \Delta\mathbf{k}) \quad (4.10)$$

where $\Delta\mathbf{k} = \mathbf{k}_{*,+} - A\hat{\mathbf{k}}_{t-1}$. This results in an efficient way to compute $\sigma_{*,t}$: We store $\hat{\mathbf{k}}_{t-1}$ from the previous round and compute $\Delta\mathbf{k}$ and $L_+^{-1}\Delta\mathbf{k}$. Then we multiply the result with itself (Eq. (4.10)) and substract it from $\sigma_{*,t-1}$. Similarly, we can compute $\mu_{*,t}$ from $\mu_{*,t-1}$ of the previous round using the difference vector $\Delta\boldsymbol{\mu} := \boldsymbol{\mu}_{*,+} - A\hat{\boldsymbol{\mu}}_{t-1}$, where $\hat{\boldsymbol{\mu}}_{t-1} = L_{t-1}^{-1}\tilde{\boldsymbol{\mu}}_{t-1}$. To summarize, we have

$$\mu_{*,t} = \mu_{*,t-1} + (L_+^{-1}\Delta\boldsymbol{\mu})^T (L_+^{-1}\Delta\mathbf{k}) \quad (4.11)$$

$$\sigma_{*,t} = \sigma_{*,t-1} - (L_+^{-1}\Delta\mathbf{k})^T (L_+^{-1}\Delta\mathbf{k}). \quad (4.12)$$

4.2 The Kernel Hyper-Parameters

As mentioned before, finding optimal hyper parameters for the kernel function involves several iterations over active set determination and gradient-descent on the marginal likelihood. However, doing this in every training round has several disadvantages: first, it requires a large computational effort, and second it makes the formulation of the online computation developed in the previous section invalid. The reason for the latter is that the online formulation relies on the fact that the active set does not change across the learning rounds, because otherwise $\mathbf{k}_*$ would have to be recomputed completely in every round. Fortunately, it turns out that the kernel hyper parameters do not change significantly across the training rounds, and, even when they do change, they only have a minor impact on the classification results of the GPC. This is another strength of the GPC framework, because essentially it represents a non-parametric model. In our implementation, we obtain the kernel hyper-parameters using cross-validation on a hold-out set. Compared to the usual gradient-descent based maximization of the log marginal, this has the advantage that the kernel parameters are optimized *across* a number of images, and not for each individual image. Especially, as we use locality and color features in combination with an Automatic Relevance Determination (ARD) kernel, the obtained length scales represent a general weighting between position and color. This turned out to achieve much better results than a per-image training of the ARD kernel parameters.

5 Segmentation

The class predictions from the IVM are only local estimates, and they disregard global properties of the image I. Therefore, we formulate the segmentation problem on the image domain $\Omega \subset \mathbb{R}^2$ as finding a *foreground region* $\hat{\Omega}_F$ so that

$$\hat{\Omega}_F = \arg\min_{\Omega_F} \lambda \int_{\Omega_F} -\log p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*)\, d\mathbf{x} + \lambda \int_{\Omega \setminus \Omega_F} -\log p(y_* = -1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*)\, d\mathbf{x} + \mathrm{Per}_\alpha(\Omega_F), \quad (5.1)$$

where Per_α is the perimeter of Ω_F, weighted by a local metric $\alpha(\mathbf{x}) = e^{-\gamma|\nabla I|}$ that depends on the image gradient, and λ is the weight of the dataterm. This functional favours spatial regularity by penalizing the boundary length of the foreground region. First, we define an indicator function $u(\mathbf{x})$ that is 1 for $\mathbf{x} \in \Omega_F$ and 0 otherwise. Then, the segmentation problem can be written in a variational formulation:

$$\min_{u \in [0,1]} \int_\Omega \varrho(\mathbf{x}) u(\mathbf{x})\, d\mathbf{x} + \frac{1}{\lambda} \int_\Omega \alpha(\mathbf{x}) |\nabla u(\mathbf{x})|\, d\mathbf{x}, \quad (5.2)$$

where the first term encodes the cost of a pixel to belong to the foreground and $\varrho(\mathbf{x}) = \log p\left(u\left(\mathbf{x}\right) = 0\right) - \log p\left(u\left(\mathbf{x}\right) = 1\right)$. The second term of Eq. (5.2) is the total variation (TV) of the indicator function u which penalizes the perimeter of the foreground region. Since the TV is not differentiable everywhere, we rewrite Eq. (5.2) as a saddle point problem:

$$\min_{u \in [0,1]} \max_{|v| \le \alpha(\mathbf{x})} \int_\Omega \varrho(\mathbf{x}) u(\mathbf{x}) + \int_\Omega u(\mathbf{x}) \operatorname{div} v(\mathbf{x}). \quad (5.3)$$

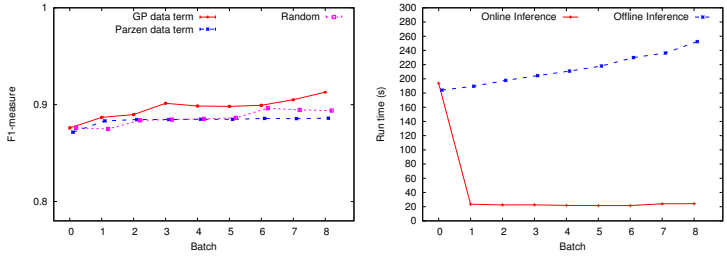
This can be efficiently minimized using a first-order primal-dual method [11].

6 Experimental Results

We evaluate our active learning approach on the benchmark data set from the University of Graz [14]. It consists of images with ground truth segmentations and user scribbles. As our method applies for foreground and background segmentation we chose a subset of 44 images from the dataset which contain only two object classes. As performance measure for this benchmark we use the f_1 measure, which is defined as the harmonic mean of precision and recall.

6.1 Benefits of the GP classifier

We compare our approach with the method of Nieuwenhuis and Cremers [8]. There, the data term is computed using a Parzen window (PW) estimator, and



(a) (b)

Fig. 3: (a) Average f-measure over 8 active learning rounds. The GPC steadily improves the segmentation, because its label queries are more informative for classification. In contrast, the Parzen window only improves slighty and then remains at a lower performance level. We also show GPC results where new user scribbles are chosen randomly and not based on the entropy. This also improves the segmentation, as it increases the training data, but it is worse than the entropy-based method. (b) Run time of online and offline inference, averaged over all images. Note that in batch 0, the online and the offline method take the same time, because they both build up the initial covariance matrix. However, in later steps the online computation time drops down significantly.

the training data consists of color information and positions of user scribbles. We use the same idea, but employ a GPC instead of the PW. Our benefit is the ability to detect misclassifications using the predictive uncertainty, which is more strongly correlated to incorrect classifications than for the PW. As a result, in active learning the GPC generates more informed questions (see Fig. 1). For a quantitative evaluation, we ran active learning with the GPC and the PW on the Graz data set (Fig. 3a). Both approaches perform equally well in the first rounds, but then the GPC (red curve) outperforms the PW (blue curve), because it asks more informed label queries, while the PW tends to be overconfident. We also show the results for randomly selected scribbles (magenta curve) instead of those with the highest uncertainty. We see that random sampling also improves the classification, as it provides more training data in every round, but the improvement is smaller compared to selecting the most uncertain segments. This is because the GP requests the more informative user scribbles.

Some results from the Graz data set are shown in Fig. 4. The left column shows the images with the inital user scribbles. Columns two and three show the uncertainties of the GPC (brighter is more uncertain) and the segmentation after the first learning round. The general segmentation is good, but small miclassifications occur. However, these often correspond to locations of high uncertainty, e.g. the lower right corner of the helicopter image or the third peg on the wardrobe: here the classification is incorrect, but the uncertainty is also high. This enables the classifier to correct the error in subsequent training rounds.

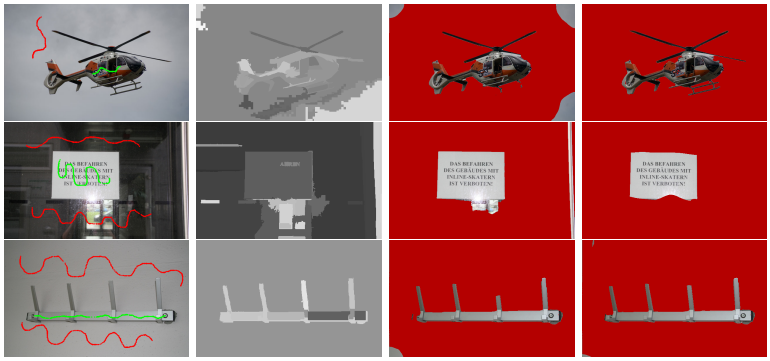


Fig. 4: Examples from the Graz benchmark. **First column:** original images with initial user scribbles. **Second column:** classification uncertainties after the first learning round. **Third column:** resulting segmentation after the first round. Note how the algorithm misclassifies some small areas, but the classification in those same areas is often very uncertain (see, e.g., the third peg on the wardrobe). Thus, the errors can be corrected by querying more useful, i.e. informative user scribbles. **Last column:** final results, obtained after a few further active learning rounds (between 1 and 5). Here, a high-quality segmentation is obtained.

6.2 Advantage of the Online Inference Algorithm

As mentioned in Sec. 4.1, we use a very efficient online class prediction step. Note that this is different from an online *training* step: while the latter is inherently provided by the IVM approach, the former is a novel contribution. In Fig. 3b, we show its benefit over the standard offline technique in every active learning round. Observe that for all but the first learning round the average run time drops from the order of minutes to the order of seconds. Also note that the increase in run time over the learning rounds is super-linear in the offline case, where for the online method it is roughly linear. In the first round, the online and the offline method perform the same steps, because every pixel is compared to all training points. Currently, we compute this in parallel on 8 CPU threads, but we expect a substantial speed-up when using a GPU implementation.

7 Conclusions

We present an efficient active learning approach and show its application to interactive segmentation. Our method learns models for background and foreground adaptively by informed questions based on the classification uncertainty and uses a regularizer that favors regions with smooth contours. To make the classification process efficient, we use an online update method that incrementally estimates the class posteriors. This reduces computation time substantially, without reducing the high segmentation performance of the active learning method.

Acknowledgment This work was partly funded by the EU project SPENCER (ICT-2011-600877).

References

1. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. Trans. on Patt. Analysis and Machine Intell. 23(11), 1222–1239 (2001)
2. Csató, L., Opper, M.: Sparse on-line gaussian processes. Neural Computation 14(3), 641 – 668 (2002)
3. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Intern. Journal of Computer Vision 59(2), 167–181 (2004)
4. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Gaussian processes for object categorization. Intern. Journal of Computer Vision 88(2), 169–188 (2010)
5. Lawrence, N., Seeger, M., Herbrich, R.: Fast sparse gaussian process methods: The informative vector machine. Adv. in neural inf. proc. systems 15, 609–616 (2002)
6. Lawrence, N.D., Platt, J.C., Jordan, M.I.: Extensions of the informative vector machine. In: Proc. of the First Intern. Conf. on Deterministic and Statistical Methods in Machine Learning. pp. 56–87. Springer-Verlag (2004)
7. Lombaert, H., Sun, Y., Grady, L., Xu, C.: A multilevel banded graph cuts method for fast image segmentation. In: Intern. Conf. on Computer Vision (ICCV). pp. 259–265 (2005)
8. Nieuwenhuis, C., Cremers, D.: Spatially varying color distributions for interactive multi-label segmentation. Trans. on Patt. Analysis and Machine Intell. 35(5), 1234–1247 (2013)
9. Nikolova, M., Esedoglu, S., Chan, T.F.: Algorithms for finding global minimizers of image segmentation and denoising models. SIAM Journal on Applied Mathematics 66(5), 1632–1648 (2006)
10. Paul, R., Triebel, R., Rus, D., Newman, P.: Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In: Proc. of the Intern. Conf. on Intelligent Robots and Systems (IROS) (2012)
11. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the piecewise smooth mumford-shah functional. In: Intern. Conf. on Computer Vision (ICCV) (2009)
12. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning (2006)
13. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Trans. on Graphics (TOG) 23(3), 309–314 (2004)
14. Santner, J., Pock, T., Bischof, H.: Interactive multi-label segmentation. In: Asian Conf. on Computer Vision. pp. 397–410. Springer (2011)
15. Triebel, R., Grimmett, H., Paul, R., Posner, I.: Driven learning for driving: How introspection improves semantic mapping. In: Proc of Intern. Symposium on Robotics Research (ISRR) (2013)
16. Unger, M., Pock, T., Trobin, W., Cremers, D., Bischof, H.: TVSeg - interactive total variation based image segmentation. In: British Machine Vision Conf. (2008)
17. Vezhnevets, A., Buhmann, E.J., Ferrari, V.: Active learning for semantic segmentation with expected change. In: Conf. on Comp. Vision and Patt. Recog. (2012)
18. Wang, D., Yan, C., Shan, S., Chen, X.: Active learning for interactive segmentation with expected confidence change. In: Asian Conf. on Computer Vision (2012)