Efficient and robust image descriptor for GUI object classification

Anastasia Dubrovina[1]
Computer Science, Technion
nastyad@cs.technion.ac.il

Pavel Kisilev[1], Daniel Freedman[1]
IBM Labs Haifa, Israel
{*pavel.prvt,d.e.freedman*}*@gmail.com*

Sagi Schein, Ruth Bergman
HP Labs Israel
{*sagi.schein, ruth.bergman*}*@hp.com*

Abstract

Graphical User Interface (GUI) object classification is essential for image-based software automation tools. The challenges posed by GUI object classification are significantly different from those in natural image classification. In this paper we present a novel image descriptor developed specifically for GUI objects; it is robust to various changes in the appearance of GUI objects, such as various screen resolution, "skin", as well as various operating system related. We use this image descriptor with Support Vector Machine classifier, and experimentally show the descriptor robustness to the above transformations, and its superior performance compared to existing image descriptors.

1 Introduction

In this paper we address the problem of the Graphical User Interface (GUI) object classification. Examples of such GUI objects are presented in Figure 2. The GUI object classification is essential for software automation tools based on record/replay paradigm , where both the record and the replay stages are image-based, thus mimicking the human-computer interaction.

The main challenge in GUI object classification is the large variation in object's appearance within a given class, even if the same operating system (OS) or browser are considered. For instance, the push buttons shown in Figure 2(e) exhibit variety of sizes, colors, texts or images inside the objects. We assume that the GUI objects differ by *geometric* and *"skin"-related* transformations. The *geometric transformations* may be of two types: (1) scaling (potentially anisotropic) due to changes in screen resolutions, as well as naturally occurring differences such as edit box sizes, which can vary considerably in their aspect ratio; and (2) translation. We assume that the GUI objects in question were found (or segmented) using some segmentation algorithm, and that the object may not be centered within the subwindow provided by the segmentation. The "skin" changes are related to the variety of themes and colors provided by today's operating systems, and these changes are somewhat harder to quantify. Our approach is to capture the look-and-feel information that remains relatively unchanged under "skin" transformations, and not color information that may change greatly.

In order to cope with the above challenges in GUI object classification task, we propose a novel type of image descriptor developed specifically for the GUI object classification task. The proposed descriptor is based on the 1D version of the Fourier-Mellin Transform. In addition, we incorporate information about object image gradients and the percentage of the white color (we explain the details in the next section). We show experimentally that the proposed descriptor is robust to various geometric and skin-related transformations of GUI objects. Using the Support Vector Machine (SVM) classifier [8] along with the proposed descriptor yields high correct classification rates for different GUI objects (see Section 3).

Note that in this paper we do not use the context of the GUI object, such as the surrounding objects, or the way the object changes its appearance in time, in order to perform the classification. Such context information can be successfully employed together with the proposed image descriptor to boost the classification accuracy, as proposed in [4].

1.1 Previous work

In the context of image-based *software automation* products, such as Project Sikuli [11], Prefab [3], EggPlant (`http://www.testplant.com/products/eggplant/`), or related application for data masking - Magen[7], the object recognition is

[1]This work was performed while the author was with the HP Labs Israel.

based on some variant of simple image template matching, or text matching, and therefore, cannot cope with the variability of intra-class transformations we discussed above. Project Sikuli [11] also employs SIFT features [5], but these may not be informative for certain types of GUI objects, as we show in Section 3.

Many state-of-the-art *object classification and recognition* methods use standard features, such as SIFT, SURF [1], ASIFT [12], MSER [6], designed for natural images. Further, as opposed to other object classification frameworks, for instance, PASCAL VOC Challenge (`http://pascallin.ecs.soton.ac.uk/challenges/VOC/`), we deal with very small objects (size of a typical radio button is 12×12 pixels), of artificial nature. We have found that our image descriptor tends to work much better than the above features within the framework of relatively simple objects such as screen-shots and GUI objects.

The rest of the paper is organized as follows: we describe the details of the proposed image descriptor in Section 2. We provide the results of classification of different GUI objects, experimental evaluation of the descriptor robustness, and compare it to the SIFT descriptor in Section 3. We conclude the paper with Section 4.

2 Image descriptor

For each GUI object we calculate the proposed image descriptor (described in details below), using the grayscale image of the object. In order to perform the classification, we first use a dataset of labeled GUI objects with their descriptors to train an SVM classifier, and then, given a new object, we calculate its descriptor and feed to the classifier to detect the object class.

In the following we describe the details of constructing the proposed image descriptor. It contains three different feature types.

1. Fourier-Mellin transform based feature. Given a grayscale image I, we first calculate the derivatives of its projections on x and y axes, as follows

$$\begin{aligned} I^x(x) &= \frac{\partial}{\partial x}\left(\sum_y I(x,y)\right), \\ I^y(y) &= \frac{\partial}{\partial y}\left(\sum_x I(x,y)\right). \end{aligned} \tag{1}$$

Then we apply a 1D Fourier-Mellin transform to it. Schematically, it can be represented as follows (shown here for I^x, but is the same for I^y):

$$\begin{aligned} &I^x\left(\tfrac{x}{a} - x_0\right) \xrightarrow{\mathcal{F}} \mathcal{F}\{I^x\}(au)e^{iux_0} \xrightarrow{|\cdot|} \\ &|\mathcal{F}\{I^x\}(au)| \xrightarrow{u\to\tilde{u}=\log u} |\mathcal{F}\{I^x\}(\tilde{u}+\log a)| \xrightarrow{\mathcal{F}} \\ &\mathcal{F}\left\{|\mathcal{F}\{I^x\}|\right\}(v)e^{-iv\log a} \xrightarrow{|\cdot|} \left|\mathcal{F}\left\{|\mathcal{F}\{I^x\}|\right\}(v)\right|, \end{aligned} \tag{2}$$

where $\mathcal{F}$ denotes the Fourier transform, a is the scaling factor and x_0 is the horizontal shift.

First, we take the Fourier Transform of the 1D signal, that is of the derivatives of the image projections. Then we take its absolute value; this operation eliminate the horizontal and the vertical shifts in the spatial domain. We then use only half of the spectrum, that is the positive frequencies, as it is symmetric anyway. Next, we take the log scale in the frequency domain; this operation converts the scale into a new shift. Similarly, taking the second Fourier Transform and the absolute value eliminates the scale dependence. (Again, we use only positive frequencies). At the end, we get a shift and scale invariant representation, and it's length is $\frac{1}{4}$ of the length of the original signal (since we halved the length by using twice only positive frequencies). In order to obtain a fixed-length representation for image of arbitrary size, we zero-pad I^x to have 1024 samples (or downsample it to have 1024 samples, if I^x is larger) before we calculate the first Fourier transform. Thus the result of this 1D Fourier-Mellin transform is a vector of 256 samples. After we concatenate transformed I^x and I^y signals, we have a 512-dimensional descriptor.

2. Histogram of gradient directions (angles) based feature We calculate the angles of gradients in the grayscale image, wrap them to be in the range $[0, \pi))$, and calculate their 4 bin histogram. We weigh the gradient angles by their magnitude, and also multiply them by triangle window weights according to how close they are to the bin centers, as proposed by David Lowe in his article on SIFT. This descriptor helps mainly to separate radially symmetric objects such as radio buttons from the rest of the object classes.

3. Percentage of white pixels This descriptor calculates the percentage of the pixels with gray values larger than a predefined threshold (we used 245 for this). This descriptor helps to separate edit boxes and push buttons or other rectangular objects which are not empty (as edit boxes are). (In general, the above threshold can be learn from the database, though we did not do it).

Concatenating the above three types of features results in a 518-dimensional vector descriptor. Figure 1 illustrates the process of descriptor calculation. Note that the proposed descriptor size is independent of the original object image size.

Finally, we would like to use the above image descriptor to perform the classification of GUI objects using the SVM classifier. We use use the non-linear version of SVM, namely Kernel SVM, where the kernel is taken to be the commonly used Gaussian kernel between the descriptor vectors. That is, given two images of GUI objects, I_1 and I_2, and their descriptors f_1 and f_2, respectively, we measure the similarity between I_1 and I_2 by

$$K(f_1, f_2) = \exp(-\gamma \left\| f_1 - f_2 \right\|_2^2) \quad , \quad \gamma > 0. \tag{3}$$

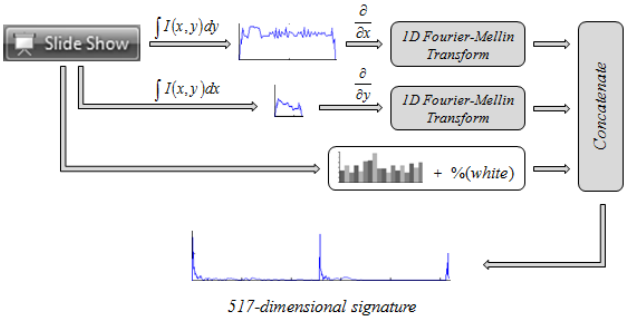

Figure 1. An illustration of the descriptor (signature) computation.

The Gaussian bandwidth γ is chosen by a cross-validation procedure.

3 Experimental results

In this section we show the results of the GUI object classification obtained with the proposed image descriptor. To perform the descriptor evaluation we created a dataset of 258 segmented objects belonging to five GUI object classes: check box, edit box, list box opener, push button and radio button. Examples of objects from these five classes are shown in Figure 2. We used this dataset both to train the Kernel SVM classifier (details in the previous section), and to perform the cross-validation procedure to find the optimal SVM parameters. In our code, we used the LibSVM Kernel SVM implementation [2].

3.1 Classification with the proposed image descriptor

In order to evaluate the quality of the classification with the proposed descriptor we randomly divided the above dataset into non-overlapping training and testing sets. We used training sets of different sizes - $40\% - 90\%$ of the whole dataset size. We performed 10 experiments for each one of these training set sizes - each time we randomly drew the necessary number of the training examples from the dataset. We then calculated the average number of the correctly classified objects from all five classes, normalized by the test set size, as a function of the the training set size. The results are shown in Figure 3, by a solid line with diamond-shaped markers.

3.2 Descriptor persistence

In order to evaluate the robustness of the proposed descriptor to geometric transformations we assembled a set of shifted, scaled and incorrectly segmented versions

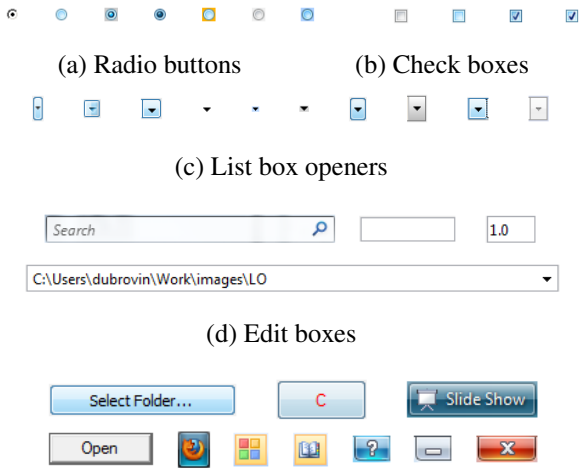

Figure 2. Examples of different objects from our dataset; all snapshots were taken from applications running on Windows Vista operating system.

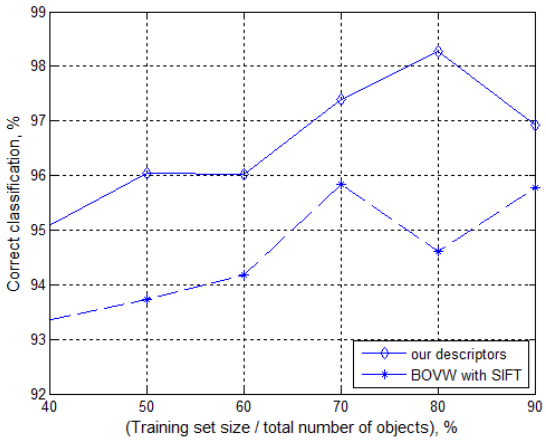

Figure 3. Percentage of correctly classified objects as a function of a training set size.

of an object from our dataset (see Figure 4). We then trained the SVM classifier using the objects from our dataset, excluding the transformed push buttons, and classified the transformed versions of our object. All of them were correctly classified as push buttons. Note that since we are interested in GUI objects, their segmentation can be made relatively accurately, therefore examples of incorrect segmentation in Figure 4 may not happen in practice. Nevertheless, we saw that we could perform their classification even for these extreme cases.

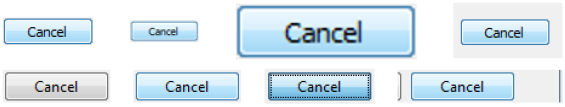

Figure 4. Examples of possible transformations of a GUI object (upper leftmost): shift, scaling, object state (before or after the mouse click), erroneous segmentation.

Figure 5. Examples of push buttons with the detected SIFT features.

3.3 Comparison with SIFT descriptors

In this section we compare the proposed descriptor to the state-of-the-art SIFT descriptor [5]. In order to perform the classification we adopted the Bag Of (Visual) Words (BOVW) approach used in [9]. As the GUI objects in question are very small, we cannot rely on SIFT feature detector. Also, the high inter-class variability in object appearance causes the SIFT feature detector find inconsistent features along the same class. Figure 5 shows an example of several push buttons with SIFT features calculated using Vedaldi's implementation of SIFT algorithm [10] for Matlab©. The detected SIFT features are marked with blue circles, where the circle size represents the feature scale. Smaller objects have no features detected.

Therefore, instead of searching for SIFT features, we used the image pixels with high gradient magnitude value (higher than a predefined threshold of 10) as feature points. For these feature points we calculated modified SIFT descriptors: we used fixed windows of size 8×8 around each feature point, and calculated gradient histograms in its four subwindows of size 4×4, thus obtaining descriptors of length 16. We then performed vector quantization of collection of the descriptors obtained for all the objects, into 500 classes, and used the quantized vectors as visual words. Finally, the object descriptors were calculated as histograms calculating the number of appearances of the above visual words in each one of the object images.

We performed the classification as described in Section 3.1, using the SIFT descriptors. The graph of the correct classification percentage is shown in Figure 3 (a dashed line), together with the correct classification percentage obtained using the proposed descriptors (a solid line). Though the BOVW with SIFT descriptors performs quite well, the proposed descriptor yields better classification.

4 Conclusion

We presented a novel image descriptor designed for GUI object classification. We showed how it could be coupled with the Support Vector Machine classifier, to achieve robust classification of GUI objects from several selected GUI object classes, despite high inter-class object image variability.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] M. Dixon and J. Fogarty. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1525–1534. ACM, 2010.

[4] D. Lehavi, O. Barkol, and S. Schein. Visibly push-down languages for a gui parsing application with probabilistic lexer. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 296 – 299, 2011.

[5] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[7] S. Porat, B. Carmeli, T. Domany, T. Drory, K. Kveler, A. Melament, and H. Nelken. Masking gateway for enterprises. In O. Grumberg, M. Kaminski, S. Katz, and S. Wintner, editors, *Languages: From Formal to Natural*, volume 5533 of *Lecture Notes in Computer Science*, pages 177–191. Springer Berlin / Heidelberg, 2009.

[8] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

[9] J. Sivic and A. Zisserman. Video Google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 127–144. Springer, 2006.

[10] A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.

[11] T. Yeh, T.-H. Chang, and R. C. Miller. Sikuli: using gui screenshots for search and automation. In *UIST'09*, pages 183–192, 2009.

[12] G. Yu and J.-M. Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line*, 2011.