USING HIERARCHY OF PARTS FOR IMAGE CLASSIFICATION

Ondřej Nečas, David Bařina
Doctoral Degree Programme (1, 2), FIT BUT
E-mail: {xnecas14, xbarin02}@stud.fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

Abstract: In this paper, we describe a modified approach to extraction of edge sequences from an image. These sequences (further referred as parts) comprise a hierarchical structure. Using a hierarchy is generally more effective than other methods, but could lead to some particular difficulties.

Keywords: feature extraction, image classification, edge, hierarchy

1 INTRODUCTION

The procedure of object classification usually consists of two different tasks: extraction of image features and determining the right class according to these features. We could use raw image data and put it on the input of the classifier but it would be very computation demanding operation and probably with worse results than using some sophisticated features. Someone could confuse object classification and object detection. The main difference between these two is that in object detection we usually need only a few models for representing one object class. On the other hand for object classification we need significantly more object models. More models means more performance demanding task and thus it is suitable to use some hierarchical structure to represent those models of many distinctive classes.

In the following work we will describe a modified approach originally described in [2, 1]. Improved features were later used for classification in [3]. Our approach is based on the same idea but we are using slightly different technique how to accomplish similar results. This method is working on statistical principle. We are searching for some common patterns which can be found in an image. Therefore we are exploiting only geometrical information contained in that image. There are many hierarchical approaches based on similar ground. For example in [5, 4] are building more complex features upon Gabor filter responses on an image.

The purpose of image features is to describe the image by as few as possible values, and not to loose much of an information. Approaches like SIFT [6] or its optimized version SURF [7] describe the image by some carefully chosen keypoints, which are invariant to some common transformations. Every keypoint is then represented by an array of local gradient directions histograms. The method described below extracts greater amount of simpler features. These features are not necessarily simpler as we advance to higher layers in the hierarchy. Therefore we can easily control the difficulty of feature extraction with respect to speed or precision.

2 HIERARCHY OF PARTS

In this chapter will be described the principle of the algorithm starting by extracting edges from the input image and building the hierarchy upon these edges. In the end will be described the detection of the parts in an image.

2.1 Edge extraction

The first phase of this algorithm is edge extraction. Grayscale image is filtered through series of Gabor function convolution kernels. Each kernel is rotated by 45 or 30 degrees so we finally get 4 or 6 maps with responses in corresponding directions. Only the direction with highest response is interesting for further processing. In the original work [2] were the authors using odd and even Gabor filter banks. Computation of so many convolutions is very performance demanding task so we have suggested some optimizations. We are computing convolution only with odd Gabor function and only in two directions aligned with axis x and y. These kernels are also axis separable and therefore more efficient to compute.

$$\alpha = \operatorname{atan2}(g_x, g_y) \tag{1}$$

$$E = \sqrt{g_x^2 + g_y^2} \tag{2}$$

Angle and energy of each edge is acquired using equation (1) and (2), respectively. Where g_x is response in horizontal and g_y in vertical direction. Now we have strength of an edge and corresponding direction. We need to threshold the energy values to get rid of some insignificant edges and noise. However there is still to much pixels to process and we need to suppress non-maximal values by comparing actual pixel with its neighbours.

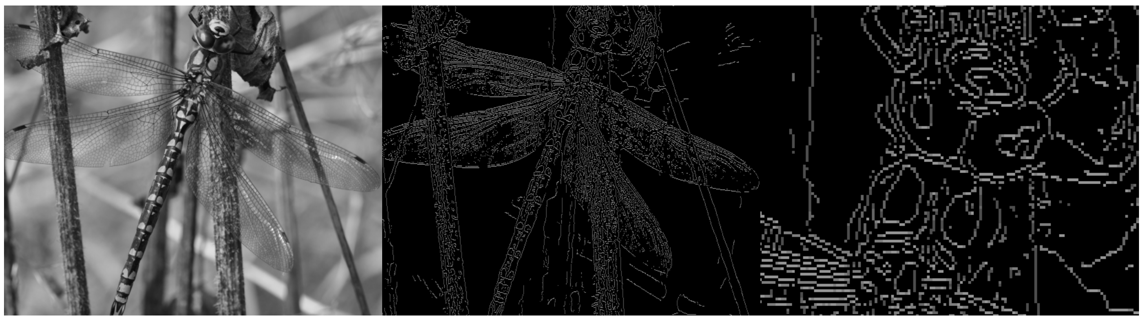
(a) Original image (b) Extracted edges (c) Detail

Figure 1: Example of original image and respective extracted edges.

2.2 Learning the hierarchy

As we have acquired the L_1 parts, we can continue building the hierarchy. The following procedure applies to any layer in the hierarchy. At first we need to construct so-called spatial maps. Spatial map is created for every combination of parts P_i^n and P_j^n. Spatial map represents the occurrence of other parts P_j^n around part P_i^n. We cycle through all parts p_i^n in the image and we increment counter at the relative position of p_j^n in the corresponding map (P_i^n, P_j^n). Also we must avoid adding parts which have greater overlap with central part. For that reason we carry list of corresponding pixels for every part instance.

Afterwards we find maximum in every map. Maps with insignificant maximum are automatically dismissed. Now we have compositions comprised of one central part P_i^n and one subpart P_j^n at relative position from the central part. There could be found large amount of possible compositions, so we push them on the stack sorted by the size of a maximum in spatial map. For each iteration of building more complex compositions we pull out only a few of the best parts. Process of building

PartInstance (p_i^n)	Part (P_i^n)	Subpart
<list> pixels (p_i^1)	<list> links (P_i^{n+1})	part (P_i^{n-1})
position	<list> subparts	relative position
part (P_i^n)	index	

Table 1: Data structures in layer n: P_i^n is part from library of parts, p_i^n is part instance detected in image.

more complex compositions continue in similar manner as above. We need to find all compositions comprised of two subparts in all images and we again build spatial maps around them.

Detection of parts will be described in the next chapter. However we need to point out one important difference in detection of parts in this phase. Parts detected in the ordinary way would share many of the parts from the lower layer. Therefore we are accepting only those parts, which have some subparts not shared among others.

Generally it is practical to use no more than 6 subparts. After finding all significant compositions comprised from up to k subparts there are some parts significantly close to others in regard of described pixels. To reduce the amount of parts which do not contribute by any kind of useful information we are grouping together parts describing the same sections of image. We are comparing the area of pixels shared between different parts and when the area exceed some threshold we declare those parts identical. In the process could be found clusters with more than one identical part, so we choose the best one and throw away the others. Finally those chosen parts are added to the list of links in corresponding part as indicated in table 1.

2.3 DETECTION OF PARTS

Similarly like in the process of learning we need firstly to obtain parts of L_1. As was mentioned earlier L_1 parts are edges detected in an image in several directions. Every initial direction represents one atomic part P_i^1. And again we cycle through all parts p_i^n in the image and for every part we have to check the presence of any part $P_i^n + 1$ from the list of links. The presence of a part $P_i^n + 1$ is verified by checking that all subparts of that part are present at their respective relative positions from the central part.

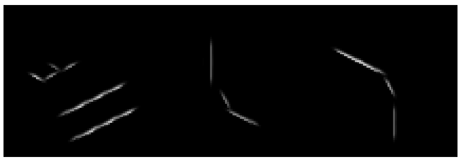

Figure 2: Example of L_4 parts learned on the dataset of keyboards.

In higher layers the radius of neighbourhood would get wider and it could lead to unreasonable memory demands. So we undersample the positions of parts by some factor in the process of building each higher layer. This technique is used even in the learning process. Unlike in [1] we are not applying additional sorting of parts after their detection in picture.

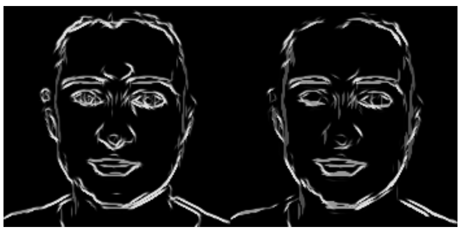
(a) L_2 (1158) (b) L_5 (591)

Figure 3: Visualization of detected parts: number of detected parts in corresponding layers.

3 IMAGE CLASSIFICATION AND RESULTS

For classification across many classes proceeds the learning process in the following way. Parts of layer 2 are learned on all input images simultaneously. Parts of higher layers are learned per category. For every category we choose several of the best parts and after going through all the categories are the parts mixed together. Because in higher layers parts cover larger areas of an image and the number of possible compositions is rapidly growing we decrease the maximum of subparts in compositions in higher layers.

In this phase we can't put on an input of the classifier the list of all parts detected in an image. The simplest way of reducing the number of detected features is to export histogram of parts for every image. To include a little more information we divide the picture in 3×3 regions and generate histograms of parts for every region. Adding overlap between regions yields some significant improvement. As parts in higher layers cover greater area it is not necessary to divide the picture in so many regions. Therefore the final feature vector is composed of divided histograms of L_2 and global histograms of higher layers.

Results were obtained by using linear SVM classifier [8]. Processing times per one image in the following tables were measured on the AMD Phenom II 945 processor (3.0 GHz). As a typical example of usage we chose a subset of scene dataset from [9]. In this dataset of 15 different categories we can assess the picture as a whole, rather than finding the object of interest in a large picture. Results show the relationship between processing time and corresponding score.

	Average precision				Number of parts			
Threshold	L2	L3	L4	L234	L2	L3	L4	t(s)
0.1 / 0.01	67.0	71.7	70.8	74.5	142	912	1592	4.4
0.07 / 0.007	67.2	69.4	66.6	70.7	139	890	1478	5.2
0.2 / 0.1	60.2	63.5	62.7	66.4	135	731	1832	2.7

Table 2: Results measured with different edge thresholds.

Setting the threshold too low can result in the longer processing time and no performance gain. With higher threshold is the processing time significantly shorter, but also the performance is compromised.

Smaller resolution undersampling can give us slightly worse results paid off by shorter times.

4 CONCLUSION

In this paper we have described modified algorithm originally proposed in [2]. Besides optimization of edge extraction procedure we used other significant simplifications that make this algorithm more

	Average precision				Number of parts			
Factor	L2	L3	L4	L234	L2	L3	L4	t(s)
3.0	67.0	71.7	70.8	74.5	142	912	1592	4.4
2.5	67.0	71.5	71.5	74.4	142	824	2014	3.5
2.0	67.0	71.2	71.3	73.8	142	771	2179	2.5

Table 3: Results measured with different factors for undersampling the higher layers.

suitable for implementation on GPGPU and other platforms. Similarly like in [3] we have demonstrated successful use of our features in image classification. Several possibilities of further work come into consideration. For example process the images on different scales could yield some significant improvement over actual results or using Bag of words model instead of simple histogram could improve the results.

ACKNOWLEDGEMENT

This work has been supported by the EU FP7-ARTEMIS project SMECY (Smart Multicore Embedded SYstems), grant no. 100230 and by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

REFERENCES

[1] Fidler, S., Leonardis, A.: Toward Scalable Representations of Object Categories: Learning a Hierarchy of Parts, *Computer Vision and Pattern Recognition*, pp. 1-8, 2007

[2] Fidler, S., Berginc, G., Leonardis, A.: Hierarchical Statistical Learning of Generic Parts of Object Structure, *Computer Vision and Pattern Recognition*, vol. 1, pp. 182-186, 2006

[3] Fidler, S., Boben, M., Leonardis, A.: Similarity-based cross-layered hierarchical representation for object categorization, *Computer Vision and Pattern Recognition*, pp. 1-8, 2008

[4] Mutch, J., Lowe, D. G.: Object Class Recognition and Localization Using Sparse Features with Limited Receptive Fields, *International Journal of Computer Vision*, vol. 80, no. 1, pp. 45-57, 2008

[5] Mutch, J., Lowe, D. G.: Multiclass Object Recognition with Sparse, Localized Features, *Computer Vision and Pattern Recognition*, vol. 1, pp. 11-18, 2006

[6] Lowe, D. G.: Object Recognition from Local Scale-invariant Features, *International Conference on Computer Vision*, vol. 2, pp. 1150, 1999

[7] Bay, H., Tuytelaars, T., Gool, L. V.: SURF: Speeded Up Robust Features, *European Conference on Computer Vision*, vol. 3951, pp. 404-417, 2006

[8] Chapelle, O., Haffner, P., Vapnik, V.: Support vector machines for histogram-based image classification, *Neural Networks*, vol. 10, no. 5, pp. 1055-1064, 1999

[9] Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, *Computer Vision and Pattern Recognition*, vol. 2, pp. 2169-2178, 2006