University of California
Los Angeles

Window Mask for Possible Object Location Generation

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Yang Xiang

2013

Abstract of the Thesis

Window Mask for Possible Object Location Generation

by

Yang Xiang

Master of Science in Computer Science

University of California, Los Angeles, 2013

Professor Stefano Soatto, Chair

Possible object location generation is an important pre-process for most object detection algorithms. In this thesis, we design a window sampling algorithm to address possible object location generation problem in two steps. First, we use a two-phase feature space partition method to achieve local descriptor classification and find interest points on image which have high probability to be on object of interest. Then we introduced a way to learn the relationship between object bounding window and bag-of-words representation of local image region, with which we can sample windows that are highly possible to contain an object. We implement the algorithm in MATLAB and test it on Graz-02 dataset, which has three object categories: car, bike, human. The algorithm achieves state-of-the-art performance according to coverage, window quality, number of windows and running time. The MATLAB scripts are merged into one file called "WindowMaskCode.pdf", which can be found in supplementary files.

The thesis of Yang Xiang is approved.

Songchun Zhu

Adnan Darwiche

Stefano Soatto, Committee Chair

University of California, Los Angeles

2013

To everyone I treasure in the world.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my appreciation to the staff in computer science department, graduate division and registrar office in UCLA for the help I received from them. I would also like to thank Vasiliy Karasev for the help on using lab servers and to thank Chaohui Wang for the discussion.

Special thanks should be given to Professor Stefano Soatto, my thesis advisor for the constructive suggestions, valuable support and inspirations he gave me during my research.

Finally, I wish to thank my parents for their support and encouragement throughout my study and research.

CHAPTER 1

Introduction

Object detection has been one of the most popular topics in computer vision field. Generally, it is a task to solve problems about "where" and "what". In practice, usually researchers specify this task as locating objects from a set of interested categories using a bounding box. While there is still debating on the mechanism of visual recognition[26], a common idea right now is that object is delineated before it is recognized, which means knowing "where" prior to "what". This indicates a general process which can separates and distinguish every object from the rest of an image. In computer vision field, this give rise to the study on segmentation algorithms[22, 23, 24]. Some context based segmentation algorithms[25, 12] can even achieve scene labeling with a very limited number of object categories at the same time.

However, segmentation task is poorly defined. Scene and object are both hierarchical. In the right image of figure 1.1, the whole human body can be considered as an object, while human face alone can be also considered as an object. Furthermore, glasses on eyes, helmet, nose and mouth can also be considered as separate objects. In upper left image of figure 1.1, it is a question whether the person inside the monitor should be taken out as a separate object. In bottom left image of figure 1.1, people inside the car can be considered as part of the car if only car is interested. All of these images contain visual phrase[14] like "man riding bike", "woman in TV" and "man driving car", which combines several objects to form a more complex and meaningful entity. It is not feasible to do segmentation

without defining object of interest if this segmentation process is going to be the pre-process of an object detection algorithm.

Figure 1.1: Examples showing that objects are hierarchical.

Thus, most object detection algorithms use a completely contrary strategy called sliding window strategy, which is simpler and more general than using a segmentation algorithm. The idea of sliding window strategy is to search every possible location and try to find objects using a trained classifier on every size and aspect ratio, which resulted in an extremely high computational cost. In practice, researchers usually sample windows with fixed step, sizes and aspect ratios to reduce the number of windows. on a $640 * 480$ image, there will be around $100K$ windows for analysis, which is still a very large number and keeps researchers from use complicated object classifier. During the past decade, the most well-known and widely used detectors[2, 27, 11] are all using sliding window strategy with a very simple feature and classifier model. Note that in the deformable part based model[11], Felzenszwalb et al still use a linear classifier based on the simple HoG

feature for each part and the root filter. The scores for each part and the root filter are also computed in a sliding window manner.

Recently, attention has been placed on the generation of possible object locations for object detection. The weakness of sliding window strategy is that the sampling is in dependent of images. It will visit locations that apparently contain no object of interest. To address this, different algorithms have been designed to select only a few regions of interest for further analysis. For example, a weak classifier, either generic[6] or category specific[27], is first used to scan every possible location and only windows with high scores are selected. In [7], over segmented regions are grouped hierarchically to produce possible image regions with high intra-coherence.

In [6, 27], the window selection is based on the analysis of the property of a whole window. In [7], the window generation is based on the relationship between over segmented image regions. There is another perspective which might be helpful: the relationship between an over segmented image region and a window that contains an object. Actually, in [5], Marszalek et al make use of such a relationship to do pixel-wise object detection. The relationship they use is the relative location between pixel-wise object shape mask and local descriptor and the affine transformation between local descriptors. It is good at predicting rigid object. But for objects like pedestrians, which have deformable parts, the performance is not well. There are two reasons for it. First, pixel-wise object shape mask is too specific to particular viewpoint, scale and sub-category, etc. Second, single local descriptor is not stable enough to predict the presence and shape of an object. Although a lot of post processes are added which can combine results from different local descriptors to get more reliable prediction, the overall performance is still not well.

In this thesis, we attempt to address the problem of generating possible object location for object detection by making use of the relationship between local image

region and object bounding box. While pixel-wise object shape mask is too strict, bounding box will be more general and can be shared within and even between categories. It allows us to design a generic window sampling algorithm which can sampling windows for the detection of a set of particular object categories. To represent image region, we use bag-of-words[21] representation of selected local descriptor from a particular set of selected visual words in a dictionary, which is more stable and representative than using a single local descriptor.

We have two main contributions in this thesis. First, we proposed a local descriptor selection algorithm using unsupervised clustering and supervised linear classification which can select local descriptors that have high possibility to be on an object of interest. Second, we proposed the idea of window mask to store the relationship between object bounding box and image region that has similar bag-of-words representation.

The sampling algorithm is tested on Graz-02[5] dataset, which contains bicycle, car and human categories with large intra-category variation. The dataset is split evenly into training set and test set use the same method in [5]. Figure 1.2 shows different categories of sample images from Graz-02. There are images containing different number of objects with different viewpoint, scale and visibility. On Graz-02, our algorithm achieves state-of-the-art performance on coverage, window quality, number of windows used per image and running time.

1.1 Related Work

In this section, I will briefly discuss the related work in two topics. The first is local descriptor selection, the second is window sampling algorithms.

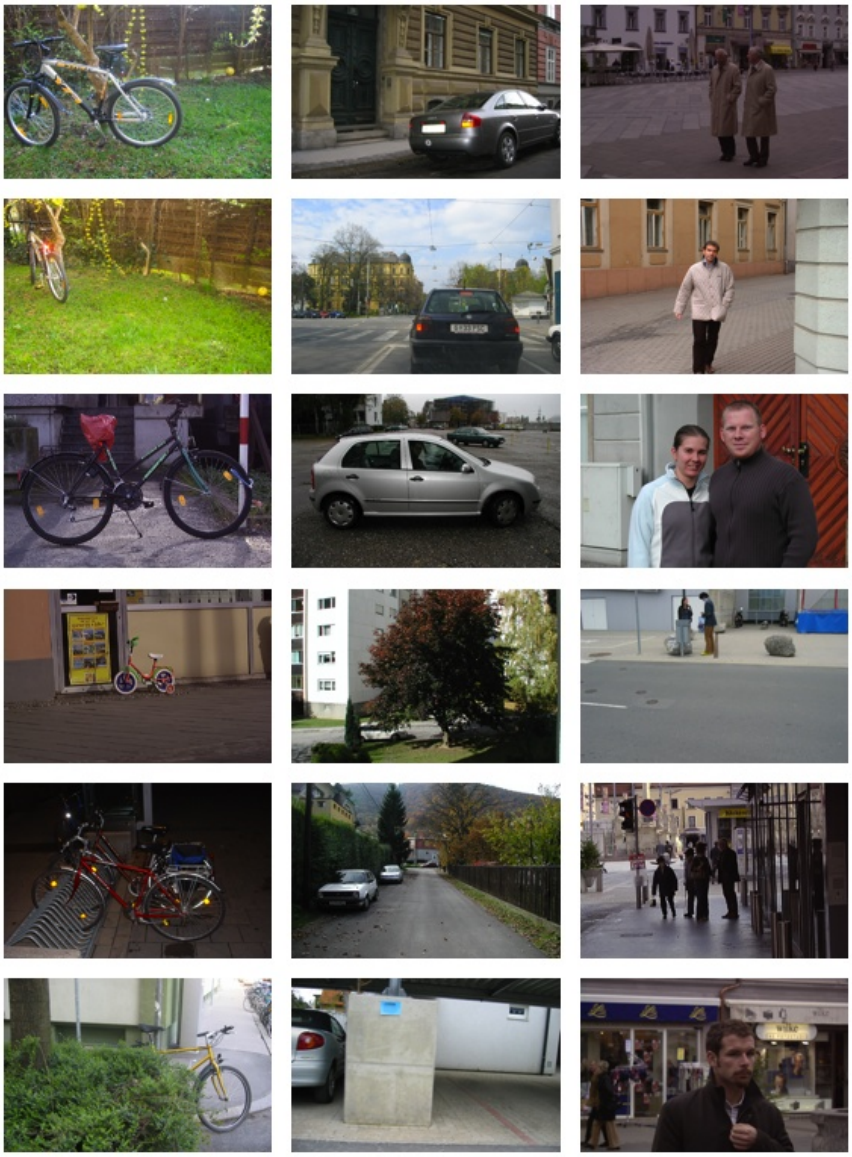


Figure 1.2: Sample images from Graz-02.

1.1.1 Local Descriptor Selection

The study on local descriptor can be dated back to decades ago. The most recent progress in the study of local descriptor should be the invention of SIFT[1]. Since then, countless researches have been made to solve different kinds of problems in computer vision field by making use of local descriptors. For example, in image matching[28, 29, 31], local descriptors are used to match points from two images which indicate the same thing. In image classification[32, 30], a local descriptor is usually assign to one of the visual words[21] in a pre-constructed dictionary to indicate the appearance of a particular visual property.

Usually, there are two steps in obtaining local descriptors. First, use an interest point detector to find locations and scales that local descriptors are going to be extracted. Second, extractor is applied to the particular locations and scales to obtain local descriptors. There are a lot of researches that study the performance of different kinds of detectors and extractors. The study on extractors usually focuses on the discriminative power, invariance and the trade-off between them[20]. The detector is used to decide the positions to apply an extractor. It reduces the number of local descriptors to be computed, thus reduces the computational cost. For a comprehensive survey on extractor and detector, refer to [15].

In some fields like image matching, interest point detectors such as Harris-Affine[33] and Difference of Gaussians[28] are still used to reduce the number of positions to be considered. In image classification researches that are based on Bag-of-Words[21], it has been proved that dense sampling of local descriptors is better than using interest point detectors[34].

The other way to reduce the number of local descriptors is to select local descriptors after they are densely extracted. Using this strategy, the selected local descriptors will usually have semantic meanings. In [17], classifiers are trained to select local descriptors that are on object of interest categories. Actually these

classifiers are some kinds of part detectors. It is to some extent the same as [35, 11], which directly train part detectors. Our research is very similar to these researches, except that we don't consider specific object categories. The local descriptor classifiers we train are thus less attached to specific object parts and have much lower semantic, which will be shared by more than one object categories.

The idea of local descriptor selection should be compared with researches related to mid-level visual concept[16, 18, 19]. In [16], binary classifiers are trained for each object category to select local patches that are highly possible to fall on desired objects. These classifiers serve as object detectors or part detectors similar to [17, 35]. In [18, 19], attributes like furry, smooth and spotted are detected by image level attribute classifiers. Note that most of the attributes defined in [18, 19] are still local feature. We can detect them by particular kinds of extractor and corresponding local descriptor classifier.

1.1.2 Window Sampling Algorithm

In the past decade, part-based model[11, 35] has a great success on object detection. However, the results are still far from satisfactory. One reason is that most object detectors [11, 2, 10] use simple linear feature and classifier to compromise with the computation cost of sliding window strategy. To improve the performance, researchers are trying to use more powerful features and classifiers, which requires a method to reduce the number of windows to be investigated for each image. This gives rise to the study of window sampling algorithms.

In [13], Efros and Hebert study the probability of a window to contain an object from a particular category set (car, pedestrian, etc.) by considering surface geometry and camera view. The result shows that simple image context can be used to significantly reduce number of windows to be considered. In [9, 5], the relationship between location of object and location of parts are studied and used

for object detection. Our window sampling algorithm is constructed based on the observation of [5].

Objectness[6] and Selective Search[7] are by far the most successful and widely used window sampling algorithms. The idea of [6] it to build a Bayesian classifier based on color contrast, edge distribution, saliency, aspect ratio and location on image to determine the probability of a window to contain an object. The idea of [7] is to combine over-segmented regions using a variety of similarity measurements (based on segmentation scale, color histogram, region size, texture and shape complexity) to produce windows with higher intra-coherence compared to other close windows with similar size. [36] combines the ideas of [6, 7]. It uses algorithm similar to Selective Search to produce windows and uses classifier similar to Objectness to score and reduce the number of windows.

CHAPTER 2

Local Descriptor Selection for Object Recognition

The purpose of local descriptor selection is to decide if a given local descriptor has a high possibility to be on an object. The basic idea of selection is a two-phase space partition.

The first phase is unsupervised clustering, which separate the feature space into small sub-spaces. Each visual word in the resulted dictionary is the center of the sub-space it represents. Local descriptors assigned to the same visual word have high visual similarity. However, this intra-visual-word similarity cannot guarantee that local descriptors assigned to the same visual word have the same label, i.e. whether or not they belong to an object of an object of interest category. It is because objects from different categories sometimes share similar visual appearance. For example, the framework of a bicycle and the railing are locally similar on their slender steel pole framework. If we consider bicycle as an interested category and railing as part of background clutter, we cannot separate local descriptors found on them apart just by considering which visual words they are assigned to.

The second phase is supervised local descriptor classification for visual word. Within each word, local descriptors in the whole training set are labeled as either inlier or outlier and a classifier is trained to make classification. Since the training and classification are done in a sub-space, the accuracy is expected to be better than the result if the classification is done without the first phase. Actually, ideas

with similar philosophy have been used a lot in object recognition and very good performance has been achieved. For example, SVM-KNN[8] trains a SVM using the top K nearest neighbors of a query and make decision with the SVM in the sub-space defined by the query and its K nearest neighbors.

During training, we define the separability score S for visual word, which measures how well inlier and outlier local descriptors can be separated. Visual words with high S are chosen and used in the window sampling stage.

There is another obvious choice for local descriptor selection. Instead of constructing a dictionary and making local descriptor classification within each visual word, we can simply train a classifier to classify all the local descriptors [16]. However, there are several drawbacks if such a strategy is used. First, the number of training instances will be extremely large, which forces us to only use linear classifier. Second, the limited discriminative power of local descriptors (sometime it is caused by the invariance power we require local descriptors to have) make the decision boundary between foreground and background local descriptors very complicated. The main reason for the complicated decision boundary is that the local descriptors we use are only good at catching low level features, which might be shared by different categories of objects at the same time. For local features belong to both foreground objects and objects in background, the decision boundary will be very complicated and we cannot separate them well only using a linear classifier. To conclude, we face a dead end if we want to find a single decision boundary to separate local descriptors from foreground and background: computational costs forces us to use linear model while the complicated decision boundary asks for a much more complicated model. It can be expected that the linear model will have a good precision and a very low recall on classifying local descriptors. Besides, it can be expected that even if a powerful non-linear model can be trained for all the local descriptors, the model will highly overfit.

2.1 Why Proposed Local Descriptor Selection Algorithm Works

The way our local descriptor selection algorithm solves the decision boundary problem discussed above can be considered as divide-select-conquer: the whole decision boundary is divided into small local decision boundaries and only those easy-to-solve ones are selected for further use. The local decision boundary problem will be much simpler and the computational cost will be much lower since the number of local descriptors within a particular visual word is only a very small fraction of the whole local descriptor pool. Under this situation, linear classifier works well enough. The resulted decision boundaries are in fact locally better than the decision boundary solved on the whole space. Similar idea has been used in SVM-KNN[8], which achieved state-of-the-art performance at that time.

Figure 2.1 is an illustration for the discussed problem. The black circles with labels A to H are visual words in the dictionary. Red and blue circles are foreground and background local descriptors, respectively. During local descriptor selection, visual words A, C, D, E, F, H are selected and orange lines indicate the decision boundary within each visual word. Green line indicates the decision boundary learned by a linear classifier for the whole feature space. The advantages of the two-phase local descriptor selection are obvious: we can focus only on those easy-to-solve sub-spaces, where we can obtain high precision and recall on local descriptor classification.

Table 2.1 shows the precision and recall using the two methods discussed above. Here we denote the proposed local descriptor selection algorithm as Two-phase Selection and the other one as Direct Selection. The classifier for Direct Selection are trained using 3% of all the local descriptors in Graz-02 dataset (The maximum percentage that can be reached on the server I use is 6%. As the percentage increases from 3% to 6%, the performance slightly drops). There are two ways

Method	PT	RT	PA	RA
Two-phase Selection	0.7532	0.4436	0.6539	0.4080
Direct Selection	0.1471	0.2327	0.0815	0.2269

Table 2.1: Comparison of two local descriptor selection methods.

to measure precision and recall. The first one is to consider the precision and recall over whole dataset, which we denote as "Total Precision" (PT) and "Total Recall" (RT). The second one is to consider the precision and recall on each image and then get the average precision and recall over all images, which we denote as "Average Precision" (PA) and "Average Recall" (RA). The measures starting with "Total" consider the capability of space partitioning. The measures starting with "Average" consider the performance stability. The result shows that by focusing only on sub-spaces, we can obtain much better precision and recall on local descriptor selection. Since the precision and recall for Direct Selection are much smaller than 0.5, we can reverse the output label of the classifier and get better results. But since the ratio of positive and negative instances is very small, only recall is improved (to about 0.6) and precision remains almost unchanged.

The sacrifice of local descriptor selection is that we miss about 5% objects in Graz-02. This means there are 5% objects are not reached by any selected local descriptors. Usually these are very small objects. During window sampling stage, this shortage can be to some extent covered by using windows generated directly from segmentation algorithm. This will be discussed in next chapter.

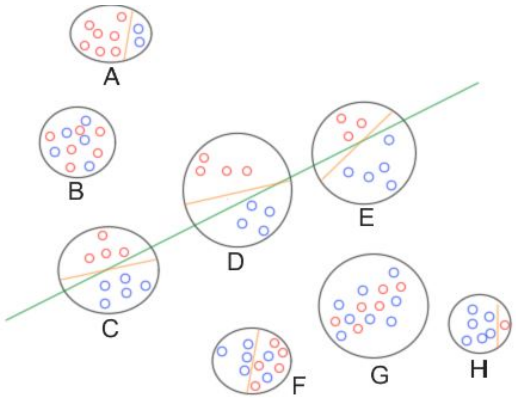


Figure 2.1: Illustration for Local Descriptor Selection. Red circles are local descriptors on object of interest and blue circles are local descriptors are background local descriptors. Black circles labeled from A to H are visual words. Green line is decision boundary when only one linear classifier is used for local descriptor selection. Orange lines are decision boundaries in each visual word.

2.2 Algorithm Description

First, a dictionary of M visual words is constructed from the training set by clustering. Note it as $\Psi = \{\psi_m | m = 1...M\}$. Within each visual word, we have $\{d_{mi}, f_{mi}, l_{mi}^{gt} | m = 1..M, i = 1, ..., C_m\}$, where d_{mi} is the i-th local descriptor, f_{mi} is the location of the i-th local descriptor on the image where it is computed, l_{mi}^{gt} is the ground truth 0-1 label of the i-th local descriptor indicating whether it is on an object. Then, for visual word m, train a binary SVM to minimize the objective function below:

$$\min_{w_m, b_m, \xi_{mi}} \quad \frac{1}{2}\|w_m\|_2^2 + C\sum_i \xi_{mi} \tag{2.1}$$

$$s.t.\ l_{mi}^{gt}(wd_{mi} + b) \geqslant 1 - \xi_{mi}, i = 1, ..., C_m \tag{2.2}$$

$$\xi_{mi} \geqslant 0, i = 1, ..., C_m \tag{2.3}$$

On the validation set, use the learned visual word classifiers to label every local descriptor in every visual word with predicted label l_{mi}^{pred}. Given the predicted

labels and ground truth labels, the separability score for visual word m can be computed by the expression below:

$$P_m = \sum_i 1_{l_{mi}^{gt}=1} \tag{2.4}$$

$$N_m = \sum_i 1_{l_{mi}^{gt}=0} \tag{2.5}$$

$$R_m^P = \frac{\sum_i 1_{l_{mi}^{gt} \& l_{mi}^{pred}=1}}{P_m} \tag{2.6}$$

$$R_m^N = \frac{\sum_i 1_{l_{mi}^{gt} \| l_{mi}^{pred}=0}}{N_m} \tag{2.7}$$

$$S_m = (R_m^P)^{k_1+k_2 P_m} (R_m^N)^{1+k_3 N_m} \tag{2.8}$$

$1_{x=y}$ is 1 when $x = y$ is true and is 0 otherwise. k_1,k_2,k_3 are parameters to balance between the stress on positive local descriptors (precision) and negative local descriptors (recall). The visual words are sorted by the score S and the top 150 are chosen. In practice, it seems these parameters won't affect the result much. Figure.1 shows local descriptors assigned to the top 150 visual words on images of bicycle, car, person category, respectively. Most of the selected local descriptors fall on object of interest while a small number of outliers are mis-classified.

2.3 Heuristic Visual Words Selection

The size of dictionary is much larger than the set of chosen visual words. There are lots of visual words mainly related to background clutter, which are not worth to compute the separability score. One heuristic way is to make use of the weight vector in a Bag-of-Words image classifier. In practice, we train for a SVM for each category as a Bag-of-Words image classifier. Then for each category, choose the visual words that have top 100 weights in the norm vector of corresponding SVM. For Graz-02 dataset, there are 300 visual words chosen (actually there are only 222 unique ones, which shows the share of low level features between object categories).

(a) bicycle (b) car (c) human

Figure 2.2: Selected local descriptors on images of different categories. Note that these local descriptors are from the same set of visual words (top 150 ranking) in dictionary. There are 10 different kinds of marks and each 15 visual words share one mark.

CHAPTER 3

Window Mask Sampling

Local descriptor selection has two advantages to object detection. First, it significantly reduces the number of locations needed to be investigated. In Figure 2.1, we can see that less than 1% local descriptors are chosen. Second, the pattern of local descriptors within a small region and the ground truth bounding box of an object is highly correlated. For example, in Figure 2.1(a), we can find that local descriptors indicated by blue and green circles usually appear on wheels of bicycle. In Figure 2.1(b), local descriptors indicated by green diamonds usually appear on the edge of car tires. In Figure 2.1(c), a single descriptor indicated by light blue cross usually appears on shoes. Such a correlation can be used to generate possible object locations as a pre-processing for object detection. In the rest of this thesis, the bag-of-words representation of a small image region is referred as *local descriptor pattern* and the relative location of the image region and an object bounding box is referred as *relative location pattern*.

From the training set, the relative location pattern can be learned. Since these patterns are not the same between different object categories, the learning process might be done once for each category. However, in practice, the number of local descriptors pattern is limited since the size of dictionary is limited. Furthermore, the relative location pattern is a general relationship that can be shared between categories. In fact, sometimes in different categories there are similar local descriptors patterns appearing on similar positions of object bounding boxes. For reason above, only one generic model is trained for all object categories.

Though the coordinate based on pixel is discrete, the possible number of relative location pattern is still very large. It is reasonable to expect that every relative location pattern found in the training set is unique. However, in a training set with limited number of object of interest categories, the relative location patterns lie in particular sub-spaces of the whole possible relative location pattern space. So during training, related relative location patterns are clustered into a small number of center patterns. For the same reason, nontrivial local descriptor patterns within the training set are also clustered into a small number of center patterns. Thus, the one-to-one mapping between local descriptor pattern and relative location pattern is transferred into a one-to-many mapping between center local descriptor pattern and center relative location patterns. We call a center local descriptor pattern and all the center relative location patterens assigned to it as *window mask*.

3.1 Obtaining segments with coherent semantics

To make use of relative location pattern during training and test, segmentation algorithm should be applied to an image to obtain segments with coherent semantics. Actually, window mask sampling algorithm only needs rectangle boxes to represent local regions. So it is possible to just use rectangle regions obtained by slicing the image regularly. However, this strategy will usually separate a coherent region into several pieces. This will add noises to local descriptor patterns, which will further influence the performance of window mask learning and window sampling.

The better choice is to use a segmentation algorithm to obtain locally coherent regions. The regions obtained from segmentation algorithm are usually coherent in texture and color. So it is more likely that the regions are either inside or outside an object bounding box, which is a desired feature for window mask sampling

algorithm. In practice, we use Simple Linear Iterative Clustering (SLIC)[3] as the segmentation algorithm. For each segment, its bounding box is used for window mask sampling. Note that there will be no selected local descriptors in some of the segments. These segments have a trivial local descriptor pattern and will not be used during window mask learning and window mask sampling. However, their bounding boxes are included in tht output sampling windows to increase the coverage.

3.1.1 Simple Linear Iterative Clustering

In this section, we briefly describe the SLIC algorithm. First, image is transferred to *lab* color space. Each pixel is represented by a 5-D vector $[l, a, b, x, y]$. The first three numbers are for lab color and the last two are coordinates of pixels. K cluster centers are sampled on a regular dense grid and then moved to the nearby positions that have smallest gradients within 3*3 neighborhood. For an image of N pixels, the sample step S, as well as the search radius of each cluster center, is $S = \sqrt{(N/K)}$. Denote each center as $C_k = [l_k, a_k, b_k, x_k, y_k]^T, k = 1, ..., K$. The distance measure D_s is defined as:

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{3.1}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{3.2}$$

$$D_s = d_{lab} + \frac{m}{S} d_{xy} \tag{3.3}$$

where $m = 0.1$ is used in this thesis.

During iterative clustering, first each pixel is assigned to the nearest cluster center whose search region covers this pixel. Then for each cluster, a new cluster center is computed by average *labxy* vector of all pixels within the cluster. This process is repeated until convergence.

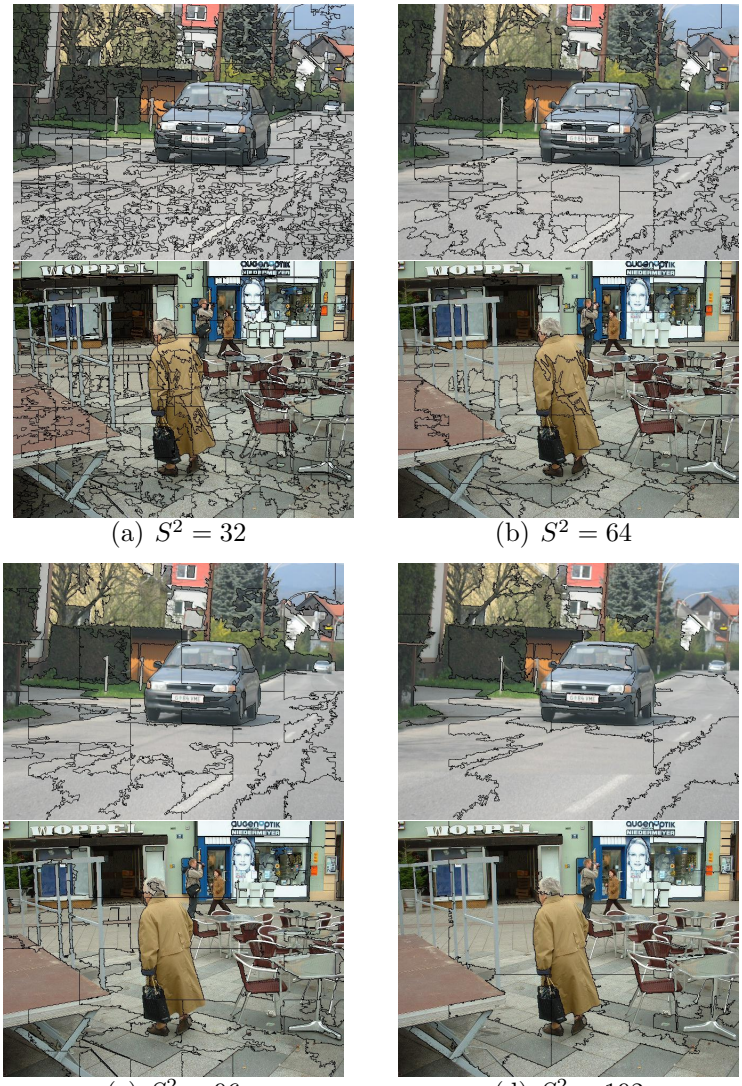
Since the segmentation result will vary a lot when using different S, in practice,

we use $S^2 = 32, 64, 96, 192$ to obtain different kinds of segments at the same location. Figure 3.1 shows the different performance of SLIC using different S.

3.2 Learning the Window Mask

If an image segment lies on an object, we can learn one relative location pattern from it. The relative location pattern defines the relative location of a segment and its corresponding object bounding box. Given a new image segment and one relative location pattern, a series of potential object locations can be computed. That's the main process of sampling. To determine which relative location patterns to be used, local descriptor patterns for each segment is computed. The matching between segment and relative location patterns is then transferred to the matching of local descriptor of segment in test image and local descriptor patterns in training set in a one-to-many manner. This process is the same as the vector quantization phase during computing Bag-of-Words representation. The relative location patterns included in one cluster form a window mask.

The way to store relative location pattern is very important since it directly determines the generated windows. There are two things that should be considered when designing the relative location pattern. First, we can recover the object bounding boxes from which relative location patterns are computed. Second, relative location pattern should be as general as possible, which allows us to combine similar patterns and reduce the number of possible object locations to be generated. This requires that relative location pattern should be as less related to a particular pair of segment and object bounding box as possible. In practice, only three numbers are stored in the relative location pattern: the relative coordinates of the left top corner of segment in the object bounding box, which are in the range of $[0, 1]$, and the ratio of width and height.



(a) $S^2 = 32$ (b) $S^2 = 64$

(c) $S^2 = 96$ (d) $S^2 = 192$

Figure 3.1: The segmentation results of SLIC with different covering radius S are different. In the car image, the segments obtained under larger S are more semantic and coherent on the closer car. But the other car, which is far away and thus small, is not separated from the background. However, when S is small, the small car has been distinguished out from the background. It is similar regarding the pedestrian image. Small S catches segments for small objects in distance, while large S catches segments for large and close objects.

3.2.1 Algorithm Description

For image I_i, we first obtain a set of over-segmented regions $\{\alpha_{in}|n = 1, ..., N_i\}$, where N_i is the number of segments obtained from the i-th image. Let $\{F_{in}|n = 1, ..., N_i\}$ denotes the Bag-of-Words representations of segments $\{\alpha_{in}|n = 1, ..., N_i\}$. These representations are computed from the local descriptors from the selected 150 visual words, which are classified as inliers by the SVMs for the corresponding visual words. Regions that have no selected visual words inside are abandoned. Let $\{W_{in}|n = 1, ..., N_i\}$ denotes the bounding box for each segment. Let $\Theta = \{\theta_{ik}|k = 1, ..., D_i\}$ denotes the ground truth bounding box for the i-th image. Only segments that intersect with a ground truth bounding box and have nontrivial local descriptor pattern are kept. The remaining 3-Tuple list $\{F_{z_t^i z_t^n}, W_{z_t^i z_t^n}, \theta_{z_t^i, z_t^k}|t = 1, ..., T\}$ stores the information about relative location and size of a ground truth bounding window to a segment bounding box with particular representation. Here T is the total number of remained segments, z_t^i, z_t^n, z_t^k are sub-lists of i, n, k, respectively. To make the relative relationship more explicit, let $W = [x_1^w, y_1^w, x_2^w, y_2^w]$ and $\theta = [x_1^\theta, y_1^\theta, x_2^\theta, y_2^\theta]$, the relative relationship between W and θ is defined as:

$$G = [\frac{x_1^\theta - x_1^w}{x_2^\theta - x_1^\theta}, \frac{y_1^\theta - y_1^w}{y_2^\theta - y_1^\theta}, \frac{x_3^\theta - x_1^\theta}{y_3^\theta - y_1^\theta}] \tag{3.4}$$

$\{F_{z_t^i z_t^n}, G_{z_t^i z_t^n}|t = 1, ..., T\}$ is clustered according to F to construct the second dictionary in this algorithm, which is denoted as $\Phi = \{\phi_m|m = 1, ..., C\}$. During clustering, ϕ_m is assigned with a set of F and their corresponding G. G assigned to the same U_m are then clustered into $\{O_{ml}|l = 1, ..., L_m\}$. The final window masks is $\{\phi_m, \{O_{ml}|l = 1, ..., L_m\}|m = 1, ..., C\}$

3.3 Sampling with Window Mask

Given a pair of relative location pattern and image segment, we could obtain one object bounding box using the inverse operation of how we compute relative location pattern. As is discussed in last section, if the way of representing relative location pattern is designed properly, we can recover the ground truth object bounding box using an image segment and the relative location pattern which is computed by the image segment and ground truth object bounding box. However, it cannot be satisfied because the way relative location pattern is represented omits the ratio between size of object bounding box and size of segment and only stores the shape of the object bounding box. So only given a pair of segment and relative location pattern, we cannot determine the size of object bounding box. To address this problem, we multiply the width of the segment bounding box with a scale vector to generate a set of possible width of object bounding box. In practice, it works well on recovering object bounding box in the training phase. Besides, even though we get more than one windows from one pair of segment and relative location pattern, the number of total windows sampled on one image are still small because the representation of relative location pattern allows us to merge them by clustering.

3.3.1 Algorithm Description

First, an image is segmented into small segments and the list of pairs $\{F_n, W_n | n = 1, ..., N\}$ is computed. The W are first included as possible object locations to increase the coverage. For $\{F, W\}$, the top K ϕ with shortest Euclidean distance are retrieved from Φ. Note all the window masks from the top K ϕ as $\{\phi_{z_t^m}, \{O_{z_t^m l}\} | t = 1, ..., K, l = 1, ..., L_{z_t^m}\}$. Windows Ω can be computed from $\{W, O_{z_t^m l} | t = 1, ..., K, l = 1, ..., L_{z_t^m}\}$ as below. Let $W = [x_1^w, y_1^w, x_2^w, y_2^w]$ and

$O_{z_t^m l} = [o_1, o_2, R]$,

$$S_j = \frac{y_2^w - y_1^w}{2} j, j \subset \mathbb{R}_+ \tag{3.5}$$

$$\omega_{z_t^m lj} = [x_1^w - o_1 S_j R, y_1^w - o_2 S_j, x_1^w + S_j(1 - o_1 R), y_1^w + S_j(1 - o_2)] \tag{3.6}$$

$$t = 1, ..., K, l = 1, ..., L_{z_t^m}, j \subset \mathbb{R}_+$$

In practice, all the window masks extracted by the same segment will be clustered into $\frac{1}{4}$ size to reduce the number of sampling windows.

CHAPTER 4

Evaluation

During evaluation, we use concatenated SIFT[1], HoG[2] and color histogram as local descriptor. In detail, local descriptors are extracted on a dense grid with a step of 6 pixels. First, Canny edge map are computed. At each location on the dense grid, if there are edges within 5*5 template, image patches of size 32*32,48*48,64*64 are extracted. Then for each patch, SIFT[1] and color histogram (25 bins per channel) are computed. Then these patches are all resized to 64*64 and HoG[2] features are computed. Finally these features are concatenated to form the final descriptor. The segmentation algorithm used in the learning of window mask and window sampling is Simple Linear Iterative Clustering[3]. All other clustering processes in this thesis are K-Means Clustering[4]. For fast nearest neighbor search, kd-tree [37] is used. The size of dictionary Ψ for local descriptors is 3000. The size of dictionary Φ for local descriptor pattern is 2000. Within each visual word, the maximum number of window masks is controlled to be 10 by clustering. During window sampling in the experiment, for each segment and its local descriptor pattern, top $K = 10, 20, 40$ nearest neighbors in Φ are extracted and window masks corresponded to these visual words are used to generate windows. For each pair of segment and relative location pattern, scale vector $j = [1, 2, 4, 6, 8]$ is used to generate possible window width. Among the parameters discussed above, K and j are the ones that have biggest influence on algorithm performance regarding coverage, window quality and number of windows per image. During implementation, we use VL-Feat[38] for SIFT and HoG

extraction, K-Means clustering, SLIC and kd-tree. We use Lib-Linear[39] as linear classifier in our thesis.

In the following sub-sections, we will first study the influence of parameters K and j. And then we will study the performance gain by adding segment bounding box W into the set of sampling windows. Finally we will compare Window Mask Sampling with Naive Sampling method, Objectness[6] and Selective Search[7]. The performance is studied on coverage, window quality, number of windows and running time per image (only when comparing with other algorithms). Averaged results in five runs are recorded. When considering whether a window is positive or not, we use the following the standard: note sampling window as W_S and ground truth object bounding box as W_{GT}, the sampled window is considered positive when

$$Q = \frac{W_S \cap W_{GT}}{W_S \cup W_{GT}} > 0.5. \tag{4.1}$$

The coverage is the percentage of positive objects in the data set. An object is considered positive if it is covered with at least one positive window. Window quality is the average Q for all positive objects.

Figure 4.1 shows some results of window mask sampling using $K = 10$ and $j = 1, 2, 4, 6, 8$. For clearness, for each object, only window with largest Q is shown. Blue window is the sampled window using window mask, red window is its corresponding segment bounding box. If only red window is shown for an object, that means segment bounding box gives the best Q.

4.1 The Number of Windows Masks Used for Each Segment

For each segment, K window masks are chosen by search nearest neighbors in Φ. As K increases, the number of windows sampled for each segment will increase. So the coverage and quality will increase as K increases. However, it is more likely

Figure 4.1: Windows that have largest overlap with objects. Blue windows and some of the red windows (if no corresponding blue boxes are shown) are the output sampling windows. Red windows are corresponding segments bounding windows of blue boxes.

K	j	Coverage	Quality	Windows
$K = 10$	[1,2,4,6,8]	0.924	0.783	$11K$
$K = 20$	[1,2,4,6,8]	0.934	0.805	$22K$
$K = 40$	[1,2,4,6,8]	0.952	0.824	$45K$
$K = 60$	[1,2,4,6,8]	0.955	0.833	$70K$

Table 4.1: Performance comparison under different K

that as K increases, the window masks involved are more irrelevant with the segment since the distances between their corresponding local descriptor patterns will become larger. The choice of K is a trade-off between coverage, quality and number of windows. Table 4.1 shows the performance of window mask sampling under different K. As K becomes larger, the increase in performance becomes smaller. While window quality still increases, the coverage has converged when $K = 60$.

4.2 The Scale Vector for Window Sampling

The scale vector has a direct influence on the performance of window mask sampling. As the length of scale vector becoming larger and the step between numbers in the vector becoming smaller, the coverage and window quality will increase. Similar to the parameter K, such increase will slow down when the scale vector becomes large enough. Table 4.2 shows the performance of window mask sampling under different scale vector j. As is shown in the table, both enlarging the scale range and decreasing the scale step will increase coverage and quality.

K	j	Coverage	Quality	Windows
$K = 10$	[1,4,7]	0.904	0.768	$7K$
$K = 10$	[1,3,5,7]	0.906	0.778	$9K$
$K = 10$	[1,2,4,6,8]	0.924	0.783	$11K$
$K = 10$	[1,3,5,7,9]	0.914	0.789	$11K$
$K = 10$	[1,2,4,6,8,10]	0.925	0.789	$13K$

Table 4.2: Performance comparison under different Scale Vector j

4.3 Performance Gain by Using Segment Bounding Box

One shortage for Window Mask Sampling is it will sometimes miss very small objects in distance. The main reason for this shortage is: when segment is very small, the sampled windows are very sensitive to the scale vector. Since the scale vector is short and step is large, small objects are often missed with a quality score between 0.4 and 0.5. However, segmentation algorithm can easily separate such small objects out from the background because objects are more visually uniform when it is in distance. So adding segment bounding box in the set of generated windows will increase the coverage.

Figure 4.2 shows some examples that segment bounding boxes give the best Q. From these images we can find that usually small objects with unique visual appearances different from background can be covered by segment bounding boxes.

Table 4.3 shows performance comparison using window mask sampling before

Figure 4.2: Segment bounding boxes that give best Q

Algorithm	Coverage	Quality	$Windows$
With Segment Bounding Box	0.924	0.783	$11K$
Without Segment Bounding Box	0.887	0.782	$10.5K$
Segment Bounding Box Only	0.606	0.632	$0.6K$

Table 4.3: Performance Gain by Adding Segment Bounding Box

and after adding segment bounding box, and the coverage and quality when only segent bounding box is used. Here $K = 10$ and $j = [1, 2, 4, 6, 8]$.

4.4 Comparison with State-of-The-Art Algorithms

The proposed algorithm is compared with Objectness[6] and Selective Search[7] and a naive sliding window sampling method. The naive sampling is the simplest way to generate possible object locations: visit each position on a dense grid of the image and sample fixed number of windows at each position. Here we sample windows at a step of 10 pixels on both X and Y coordinates of an image. At each position we generate windows of size $[45 : 45 : 405] \times [45 : 45 : 405]$, where $\times$ means Cartesian product and $[45 : 45 : 405]$ is short for the vector which starts from 45 and ends at 405 with a step of 45. we use the MATLAB packages of Objectness[6] and Selective Search[7] provided online by their authors. All the algorithms are tested on a Macbook Pro with 2.2 GHz Intel Core i7 processer and 8GB 1333MHz DDR3 memory.

The results on Graz-02 are shown in Table 4.4. Naive Sampling algorithm is the fastest one, taking about 0.1s per image. It can reach very good coverage and

quality score if using small step size and a dense sampling window size, which will significantly increase the number of windows generated for each image. Windows sampled using naive sampling algorithm are independent to images. It is a good property since the performance won't change drastically when it is applied to different source of data. On the contrary, it is the reason why the number of windows per image cannot be reduced while we maintain the coverage and window quality. Using $K = 20$, window mask sampling algorithm can get better coverage and same quality using only $\frac{1}{4}$ of the windows used by naive sampling algorithm.

Objectness[6] can achieve very high coverage with a small number of windows because it uses a general classifier to determine the possibility that a window contains an object and only selects top ones from $100K$ candidate windows using non maximum suppression. However, it sacrifices the window quality. And it is very hard for objectness to increase window quality just by allowing more window candidates to be kept during non-maximum suppression because high quality windows might have been omitted when the low quality windows are selected. In fact, since window mask sampling algorithm has included the computing of local descriptor pattern (in practice, a query of the Bag-of-Words representation of any window is O(1) time complexity), it is easy to train a window classifier like Objectness[6]. It is not considered since window mask is already data driven. This algorithm might overfit to a particular data set if another classifier is trained on it.

Selective Search[7] has almost the best coverage and quality scores using in average $37K$ windows per image. However, it takes in average $35s$ per image. Even though the number of windows can be reduced with very limit coverage and quality drop if we discard some recommended feature in [7], the running time per image won't change much. The slowness is mainly caused by the merge phase in which the algorithm has to decide which two segments to merge.

Window mask sampling algorithm has achieved the second best coverage using

only $11K$ windows per image. Consider the results of naive sampling and window mask sampling when $K = 20$: window mask sampling achieve better coverage and same window quality with only $\frac{1}{4}$ of the windows used by naive sampling. Consider the results of Objectness[6] and window mask sampling when $K = 10$: window mask sampling achieves better coverage and better window quality with 5 times of the windows used by Objectness under same processing time. But we have to note that in the MATLAB package of [6] and [7], the most time consuming part has been written in MEX file, while the time consuming part of window mask sampling algorithm are written in pure MATLAB code.

In conclusion, window mask sampling algorithm achieves state-of-the-art performance and surpasses other algorithms in different kinds of comparisons. Even though it is not the best on all the parameters considered in this section, its overall performance makes it the best choice in front of the trade-off between coverage, window quality, number of windows and running time.

Algorithm	Coverage	Quality	Windows	Time/Image
Proposed ($K = 40$)	0.952	0.824	$45K$	$4s$
Proposed ($K = 20$)	0.934	0.805	$22K$	$4s$
Proposed ($K = 10$)	0.924	0.783	$11K$	$4s$
Naive Sampling	0.912	0.805	$87K$	$0.1s$
Objectness[6]	0.899	0.705	$2K$	$4s$
Selective Search[7]	0.989	0.889	$37K$	$35s$

Table 4.4: Performance comparison on Graz-02 dataset

CHAPTER 5

Discussion

The performance evaluation of window sampling algorithms is complicated since there are several measurements to compare. Since no algorithm exceeds the rest on all measurements, we cannot simply decide which one is the best. In practice, we have to select sampling algorithm based on specific needs. The importance of window quality should be further analyzed. A possible way is to compare the object detection results using same detector and different sampling algorithms. It is better to evaluate Window mask sampling algorithm on a larger dataset such as PASCAL VOC in the future to see the generalization capability of window mask and if generic local descriptor selection still work well when number of object categories becomes larger. Optimistic expectation can be made on both of the questions. The first reason is that relative location pattern is a general relationship and can be shared between categories. The second reason is that local descriptor selection is on sub-spaces, whose property won't change much when size of dataset and number of categories become larger. Besides, different kinds of local descriptors and different local descriptor classifiers should be tested in the future.

CHAPTER 6

Conclusion

In this thesis, we study the window sampling algorithm which is used to generate possible object locations for object detection. We first study local descriptor selection method from which we can choose local descriptors with high probability to be on an object. The two phase selection algorithm we design is proved to be much better than using one simple classifier for local descriptor classification. The result shows that by constraining the problem into small sub-spaces and solving them separately, the local decision boundary is simpler and can be well learned by simple linear classifier. Then we design a window sampling method based on local descriptor selection. On Graz-02[5] dataset, our algorithm achieves state-of-the-art performance. Its performance is comparable to Objectness[6], Selective Search[7] on coverage, window quality, number of windows per image and running time.

References

[1] D.Lowe. *Object Recognition from Local Scale-Invariant Features.* In ICCV 1999.

[2] N.Dalal and B.Triggs. *Histograms of Oriented Gradients for Human Detection.* In CVPR 2005.

[3] R.Achanta, A.Shaji, K.Smith, A.Lucchi, P.Fua and S.Süsstrunk. *SLIC Superpixels.* École Polytechnique Fédéral de Lausssanne (EPFL) Technical Report 2010.

[4] D.Arthur and S.Vassilvitskii. *k-means++: The Advantages of Careful Seeding.* In Proceedings of the eighteenth annual ACM-SIAM symposium (2008) on Discrete algorithms.

[5] M.Marszatek and C.Schmid. *Accurate Object Localization with Shape Masks.* In CVPR 2007.

[6] B.Alexe, T. Deselaers and V.Ferrari. *Mearusing the Objectness of Image Windows.* IEEE PAMI, 2012.

[7] J.R.R.Uijlings, K.E.A van de Sande, T.Gevers and A.W.M.Smeulders. *Selective Search for Object Recognition.* IJCV 2013.

[8] H.Zhang, A.Berg, M.Maire and J.Malik. *SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition.* In CVPR 2006.

[9] O.Chum and A.Zisserman. *An Exemplar Model for Learning Object Classes.* In CVPR 2007.

[10] T.Malisiewicz, A.Gupta and A.Efros. *Ensemble of Exemplar-SVMs for Object Detection and Beyond.* In ICCV 2011.

[11] P.Felzenszwalb, R.Girshick, D.McAllester and D.Ramanan. *Object Detection with Discriminatively Trained Part-based Models.* PAMI 2010.

[12] R.G.Cinbis, J.Verbeek and C.Schmid. *Segmentation Driven Object Detection with Fisher Vectors.* In ICCV 2013.

[13] D.Hoiem, A.Efros and M.Hebert. *Putting Objects in Perspective.* IJCV 2008.

[14] M.A.Sadeghi and A.Farhadi. *Recognition Using Visual Phrases.* In CVPR 2011.

[15] K.Mikolajczyk and C.Schmid. *A Performance Evaluation of Local Descriptors.* IEEE PAMI 2005.

[16] Q.Li, J.Wu and Z.Tu. *Harvesting Mid-level Visual Concepts from Large-scale Internet Images.* In CVPR 2013.

[17] G.Dorkó and C.Schmid. *Selection of Scale-invariant Parts for Object Class Recognition.* In ICCV 2003.

[18] V.Ferrari and A.Zisserman. *Learning Visual Attributes.* In NIPS 2008.

[19] S.J.Hwang, F.Sha and K.Grauman. *Sharing Features Between Objects and Their Attributes.* In CVPR 2011.

[20] M.Varma and D.Ray. *Learning The Discriminative Power-Invariance Trade-Off.* In ICCV 2007.

[21] G.Csurka, C.Dance, L.Fan, J.Willamowski and C.Bray. *Visual Categorization with Bags of Keypoints.* In ECCV 2004.

[22] D.Freedman and T.Zhang. *Interactive Graph Cut Based Segmentation with Shape Priors.* In CVPR 2005.

[23] J.Shi and J.Malik. *Normalized Cuts and Image Segmentation.* IEEE PAMI 2000.

[24] P.Felzenszwalb and D.Huttenlocher. *Efficient Graph-based Image Segmentation.* IJCV 2004.

[25] C.Galleguillos, B.McFee, S.Belongie and G.R.G.Lanckriet. *Multi-Class Object Localization by Combining Local Contextual Interactions.* In CVPR 2010.

[26] K. Grill-Spector and N.Kanwisher. *Visual Recognition: as soon as you see it you know what it is. Psychology Science 2005.*

[27] P.Viola, M.Jones. *Robust Real-time Face Detection.* IJCV 2004.

[28] D.Lowe. *Distinctive Image Features From Scale-invariant Keypoints.* IJCV 2004.

[29] K.Grauman and T.Darrell. *Efficient Image Matching With Distributions of Local Invariant Features.* In CVPR 2005.

[30] S.Lazebnik, C.Schmid and J.Ponce. *Beyond Bag of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.* CVPR 2006.

[31] M.Ozuysai, M.Calonder, V.Lepetit and P.Fua. *Fast Keypoint Recognition using Random Ferns.* IEEE PAMI 2010.

[32] A.Bosch, A.Zisserman and X.Munoz. *Image Classification using Random Forests and Ferns.* In ICCV 2007.

[33] K.Mikolajczyk and C.Schmid. *Scale & Affine Invariant Interest Point Detectors.* IJCV 2004.

[34] Fei-Fei.Li and P.Perona. *A Bayesian Hierarchical Model for Learning Natural Scene Categories.* In CVPR 2005.

[35] L.Bourdev, S.Maji, T.Brox and J.Malik. *Detecting People Using Mutually Consistent Poselet Activations.* In ECCV 2010.

[36] I.Endres and D.Hoiem. *Category Independent Object Proposals.* In ECCV 2010.

[37] J.L.Bentley. *Multidimensional Binary Search Trees Used for Associative Searching.* Communications of the ACM 1975.

[38] A.Vedaldi and B.Fulkerson. *VLFeat Library.* http://www.vlfeat.org/

[39] R.Fan, K.Chang, C.Hsieh, X.Wang, and C.Lin. *LIBLINEAR: A Library for Large Linear Classification.* Journal of Machine Learning Research 2008.