

Fast Object Class Recognition

by

Maryam Mahdavian

AN HONOURS THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Science

in

THE FACULTY OF SCIENCE

(Department of Computer Science)

The University of British Columbia

May 2005

© Maryam Mahdavian, 2005

Abstract

We propose a novel approach for object class recognition using scale invariant features and Gaussian Processes as our kernel-based classifier. We measure the performance of this approach in two stages: predicting the presence of a class of objects in images and localizing them. Our object class recognition method is comparable to other state-of-the-art approaches. Furthermore, we propose sophisticated numerical methods for reducing the computational cost of Gaussian Processes. Speed-ups are achieved using Krylov Subspace and Dual-Tree methods. We also reduced the storage requirement from $O(M^2)$ to $O(M)$. We trained and tested our algorithms on the PASCAL collection of images, which is a common database in this research area.

Contents

Abstract	ii
Contents	iii
1 Introduction	1
2 Object Class Recognition	3
2.1 Object Recognition with Scale Invariant Local Features	4
2.2 Object Vocabulary Construction	4
3 Classification with Gaussian Processes	6
3.1 Krylov subspace iteration	8
3.2 Fast N-Body Problems: Dual-tree Method	10
4 Experiments	12
4.1 Part 1 : Image Classification, the "Presence" Problem	13
4.2 Part 2 : Binary Classification, the "Localization" Problem	14
4.3 Part 3 : Computational Cost: Comparison of Three Approaches	15
5 Results	17
6 Conclusion	20
7 Acknowledgements	21
Bibliography	22

Chapter 1

Introduction

Object class recognition has been a challenging problem in Computer Vision for a long time. We propose an approach for solving this problem using Lowe's scale invariant SIFT features [8]. We train the classifier with the most frequently seen SIFT features of object classes and predict the labels of features in test images. Most state-of-the-art techniques for object class recognition take advantage of Machine Learning methods for classification. In our approach, we use Gaussian Processes to classify SIFT features into pre-specified classes.

Second problem that we address in this thesis is the complexity of classification task with Gaussian Processes. Although these processes perform very well in classification, their high computational cost make them impractical for many applications. We apply an instance of Krylov Subspace methods, Conjugate Gradient to reduce the computational time of classification from $O(M^3)$ to $O(M^2)$. We then apply Dual-Tree method which in the family of N-Body fast methods to reduce the computational time from $O(M^2)$ to $O(M \log M)$ and reduce the storage requirement from $O(M^2)$ to $O(M)$ where M is the often large number of SIFT features in the training set.

In section 2, we address the problem of object class recognition and explain our

solution. In section 3, we discuss classification with Gaussian Processes. In Section 4 and 5 we introduce our methods for speeding up the classification task. In sections 6 and 7, we outline the experiments and describe the performance in terms of performance and efficiency.

Chapter 2

Object Class Recognition

Recognizing object categories is one of the oldest yet most challenging problems in Computer Vision. The problem is conceptually difficult due to the undefined nature of object similarity. Building a similarity matrix apriori is very challenging. The objects in different classes sometimes can have more visual features in common compared to some of the objects in the same class. The key problem is recognizing the visual features of objects by which we identify them as members of particular classes[12]. There are a number of on-going research projects for solving this problem [9]. Many state-of-the-art approaches use SIFT features [8][9] alone or in combination of other visual features as the primary sets of features. In this work we extract 128-dimensional SIFT features from training images of an object class and cluster them. Based on the frequency of features, we choose a subset of them as examples of that specific object class. We call this subset of representative features the object class vocabulary. In this section we describe SIFT features briefly and explain the construction of the object class vocabulary in more details.

2.1 Object Recognition with Scale Invariant Local Features

For feature extraction, we rely on Lowe’s “Scale Invariant Feature Transform” (SIFT) features [8] which have been proved to be one of the most successful approaches in object recognition. In this approach keypoints are identified by finding peaks of a Difference-of-Gaussian (DOG) function convolved with different scales of an image. Keypoints of an image are located in regions and scales where there is a high amount of variation after convolutions. This means that it is likely for these locations to contain useful information for matching. In addition, since these keypoints are in the peak of DOG, minor changes in their surroundings do not greatly affect their locations. For a SIFT feature of an image, we know its location, scale and a local key descriptor of the region around it. Rather than performing PCA on features or representing the local region as the pixels values, we represent SIFT features in the way of Lowe’s. Following his approach, a local region is divided into K smaller subregions and each subregion is described by a histogram of size Q of image gradients. These appearance descriptors are invariant to small changes in position of keypoints and brightness. The descriptor is formed by a vector containing the values of orientation histogram entries. The best results are achieved with a 4 array of histograms with 8 orientation bins [8]. In total, we have a $4 \times 8 = 128$ element feature vector for each keypoint [1]. As a result the appearance of each feature from an image lies in a 128 dimensional space where each dimension is a number between 0 and 255.

2.2 Object Vocabulary Construction

Considering the nature of SIFT, we think that it is likely for objects of the same class to have similar local features. Therefore, to construct a vocabulary for a particular object class, we are looking for features that are most common in the training images of objects. For training images we use loosely segmented images provided

by the PASCAL database, which can be downloaded from (<http://www.pascal-network.org/>). In these images a boundary rectangular box has been specified around the object. We crop regions inside these boxes out of the training images and extract their SIFT features. We choose a selection of cluster means of SIFT features through Vocabulary Construction that is outlined in Figure 2.1.

- 1) For each object class c , initialize F_c , that is the set of feature clusters for object class c
- 2) Sequentially cluster features of F_c based on their Normalized Euclidian distance: For each new SIFT feature, f_i from interest regions, compare f_i with all mean-features in F_c .
 - 2.1) If the distance between features f_i and mean-feature f_j is less than t we add f_i to the corresponding cluster and recompute the mean.
 - 2.2) Otherwise (i.e. if there is no cluster of features close enough to f_i we form a new cluster for f_i and set the mean-feature of this cluster to its only feature.
- 3) sort feature clusters by their sizes.
- 4) pick S the most-commonly seen feature clusters, that is the S largest clusters.

Figure 2.1: *Object Vocabulary Construction*

There are two parameters involved this process: t , the bound of feature clusters, and S , the number of feature clusters we use for training. By increasing S and decreasing t we will have more features in the vocabulary. We chose $t = 0.05$ and experimented with different values of S .

Chapter 3

Classification with Gaussian Processes

Gaussian Processes are a generalization of a finite Gaussian distributions and an instance of stochastic process. Similar to multivariate Gaussian distributions, they operate on functions instead of vectors. These processes are classical nonparametric techniques. In recent years they have attracted interest in machine learning [4]. A Gaussian Process is a distribution:

$$p(\mathbf{t}|\mathbf{K}, \{x_n\}) = \frac{1}{\mathbf{Z}} \exp\left(-\frac{1}{2}(\mathbf{t} - \mu)\mathbf{K}^{-1}(\mathbf{t} - \mu)\right),$$

where \mathbf{t} is a set of random functions indexed by $\{x_n\}$, $\mathbf{t} = \{t(x_1), t(x_2), \dots\}$, and matrix \mathbf{K} is the covariance matrix of these functions. With a Gaussian Process model we can predict the value of t_{N+1} given x_{N+1} and a set of observations on variables. Gaussian Processes is a promising non-linear interpolation tool, but they can be modified to produce efficient classifiers [3]. In the classification task the goal is to predict the labels of test variables based on their similarity to labelled training data. The prediction can be formulated in the form of matrix vector multiplication:

$$\mathbf{y}^{(u)} = \mathbf{K}_{\mathbf{u}\mathbf{l}}\mathbf{K}_{\mathbf{l}\mathbf{l}}^{-1}\mathbf{y}^{(l)}. \quad (3.1)$$

where $\mathbf{y}^{(u)}$ and $\mathbf{y}^{(l)}$ represent vector of labels for test and train variables respectively, $\mathbf{K}_{\mathbf{l}\mathbf{l}}$ is the covariance matrix for labelled variables and $\mathbf{K}_{\mathbf{u}\mathbf{l}}$ is the similarity matrix

between train and test variables. We can measure the similarity between variables with different kernel functions as long as they produce a valid kernel matrix. In our application we use Gaussian Kernel for measuring the similarity. Therefore, the similarity between x_i and x_j is measured by kernel function k such that

$$k(x_i, x_j) = e^{-\frac{1}{\sigma}\|x_i - x_j\|^2}. \quad (3.2)$$

where x_i and x_j can be train or test variables and σ is a hyper-parameter. In our application we assigned the value of σ by cross-validation. Note that entries in matrix \mathbf{K}_{ll} represent the similarity for all pairs of train variables $x_i^{(l)}$ and $x_j^{(l)}$. As a result matrix \mathbf{K}_{ll} is a valid kernel matrix and consequently symmetric positive definite. Entries of matrix \mathbf{K}_{ul} , $k(x_i^{(u)}, x_j^{(l)})$, represent the similarity between unlabeled variables $x_i^{(u)}$'s and labeled variables $x_j^{(l)}$'s. Therefore, \mathbf{K}_{ul} and \mathbf{K}_{ll} are $N \times M$ and $M \times M$ matrices where N is the size of unlabeled test variables and M represents the size of the training set. With our choice of kernel function, higher similarity values will be assigned to points that are closer to each other in feature space. We can extend binary classification with Gaussian Process to multi-class classification, by performing binary classification for each class, where background features are considered to be a separate class. In other words we can introduce a matrix of labels $\mathbf{Y}^{(u)}$ and $\mathbf{Y}^{(l)}$ instead of vector of labels. Consequently, equation (3.1) can be written as:

$$\mathbf{Y}^{(u)} = \mathbf{K}_{\text{ul}}\mathbf{K}_{\text{ll}}^{-1}\mathbf{Y}^{(l)}. \quad (3.3)$$

where now $\mathbf{Y}^{(u)} \in \mathbf{R}^{N \times T}$, $\mathbf{Y}^{(l)} \in \mathbf{R}^{M \times T}$ and T is the number of classes including the background class.

The disadvantage of Gaussian Processes is in their high computational cost. The cost of inverting matrix \mathbf{K}_{ll} is $O(M^3)$ where M is the size of training set and usually much larger than N the size of test data sets since we are performing supervised learning. Although this can be thought as a one-time cost, in practice we might want to add features to our training set and this high computational cost can be problematic especially for large data sets.

Instead of inverting matrix \mathbf{K}_{Π} in computing $x := \mathbf{K}_{\Pi}^{-1}\mathbf{y}^{(l)}$ we can solve the linear system of equations, $\mathbf{K}_{\Pi}x = \mathbf{y}^{(l)}$ for vector of variables x [11]. Although the naive solution to this system of equations is also $O(M^3)$ as well, we can take advantage of numerical methods such as Krylov subspace methods to decrease the cost of computation to $O(M^2)$, and subsequently of Dual-Tree recursion to reduce the cost to $O(M \log M)$.

3.1 Krylov subspace iteration

The intuition behind Krylov subspace methods is to use the history of what we have already learned. We formulate this intuition in terms of projecting an M -dimensional problem into a lower dimensional subspace. Given a matrix \mathbf{K}_{Π} and a vector $b := \mathbf{y}_l$, the associated Krylov matrix is:

$$\mathbf{P} = [\mathbf{b} \quad \mathbf{K}_{\Pi}\mathbf{b} \quad \mathbf{K}_{\Pi}^2\mathbf{b} \quad \mathbf{K}_{\Pi}^3\mathbf{b} \quad \dots].$$

The *Krylov subspaces* are the spaces spanned by the column vectors of this matrix. In each iteration we expand the subspace by one column and find a new estimate of $x_t = \mathbf{K}_{\Pi}^{-1}\mathbf{b}$ in the Krylov subspace. One way of solving this system of equations is applying GMRES algorithm or its special instance for symmetric matrices, the MINRES algorithm. This means that in step t , the solution is approximated by a vector $x_t \in \mathbf{P}_t$ where \mathbf{P}_t is the krylov subspace spanned in iteration t . A similar problem in another object recognition task has been solved before using MINRES [10]. However in this application we can take advantage of a simpler Krylov Subspace method, the Conjugate Gradients algorithm. The Conjugate Gradients (CG) method is applicable to Symmetric Positive Definite (SPD) matrices. Since \mathbf{K}_{Π} is a valid kernel matrix, by definition it is a SPD matrix. Eigenvalues of a SPD matrix are all positive, or equivalently $x^T\mathbf{K}_{\Pi}x > 0$ for every nonzero $x \in \mathbf{R}^M$ where M is the number of SIFT features in the training set. Under this assumption we can define function $\|\cdot\|_{\mathbf{K}_{\Pi}}$ by: $\|x\|_{\mathbf{K}_{\Pi}} = \sqrt{x^T\mathbf{K}_{\Pi}x}$ which is a norm on \mathbf{R}^m and is called \mathbf{K}_{Π} -norm. We care about the \mathbf{K}_{Π} -norm

of vector $e_t = x_* - x_t$ which measures the error in approximating x_* at step n . The CG iteration can be described as a system of recurrence formulas that generates the unique sequence of iterates $\{x_t \in \mathbf{P}_t$ with the property that at step t , $\|e_t\|$ is minimized [2]. The algorithm is shown in Figure 3.1.

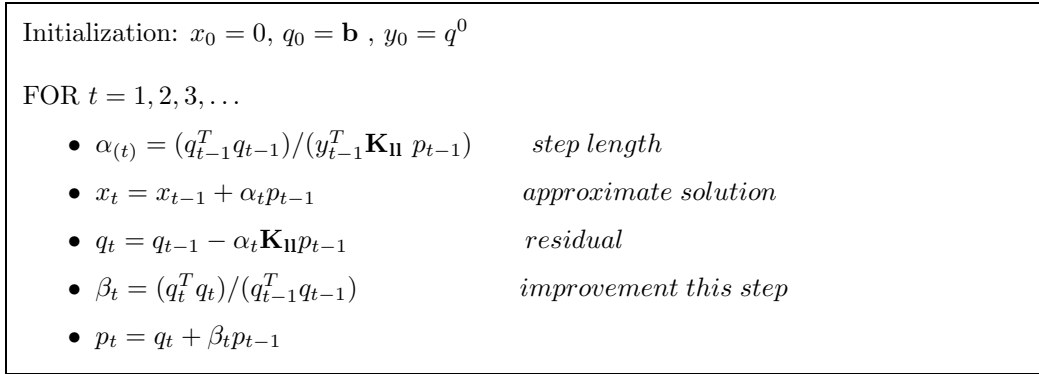


Figure 3.1: *The Conjugate Gradients Algorithm.*

The CG iteration is extraordinary simple and programmable in a few lines. Since it deals only with m -vectors, it is simpler, for example, than Gaussian elimination with pivoting. The only complication is the choice of convergence. For matrices such as Kernel matrices, the eigen-vectors converge quickly and not many iterations are needed. We can also set a threshold for convergence instead of specifying the exact number of iterations [6]. At each step, the CG algorithm involves several vector manipulations and one matrix-vector product, the computation of $\mathbf{K}_{\Pi} p_{t-1}$ which appears twice but can be computed only once. If matrix \mathbf{K}_{Π} is dense and unstructured the $O(M^2)$ vector-matrix multiplication dominates the cost and the cost of CG will be $O(M^2)$. But \mathbf{K}_{Π} is well-structured since it is a kernel matrix in which all entries are the similarities of pairs of points computed by Gaussian Kernel. Therefore, by exploiting the structure we can reduce the complexity to $O(M \log M)$.

3.2 Fast N-Body Problems: Dual-tree Method

The expensive step in the Conjugate Gradient algorithm is the operation $g = \mathbf{K}_{\Pi} y_{t-1}$ in the first step which requires solving an $O(M^2)$ kernel estimates:

$$g_i = \sum_{j=1}^M y_j e^{-\frac{1}{\sigma} \|x_i - x_j\|^2}$$

for $i = 1, 2, \dots, M$. This expression can be written in the generic form:

$$f_i = \sum_{j=1}^M f_j e^{-\frac{1}{\sigma} \|x_i - x_j\|^2} \quad i = 1, 2, \dots, M$$

It is possible to reduce the cost to $O(M \log M)$ at the expense of a small specified error ϵ using fast methods. The main idea in using these fast methods is that for certain kinds of kernel functions this matrix-vector multiplications exhibit some geometric structure. Therefore instead of viewing this operations in a linear-algebraic fashion, we approach them as N-body problems. N-body problems are generalized from the computational physics problem of efficiently computing all pair wise potentials of a set of points. These methods can be applied to efficiently compute sum of products and maximum of products [7]. The operation of matrix vector multiplication is considered as a sum-product problem. To efficiently compute this operation, one can use fast multi-pole or Dual-tree methods. The author has experimented these methods and has observed that in high-dimensional spaces such as in the 128-dimension space of SIFT features, the current variations of Fast Multiple Methods do not perform well. On the other hand *Dual-Tree* (DT) algorithm provides a computationally efficient way for our specific matrix-vector multiplication. Dual-Tree provides simple yet effective space-partitioning trees, where each node in the tree defines a distinct region in the data space using a bounding hyper-rectangle. Each split is made along a single coordinate which has the largest variance [4]. The tree is often built all the way down to some predefined number of points ρ at the leaves. In this project we are using the software provided by Dustin Lang [7] which is based on the work of Alex Gray and Andrew Moore [5]. This algorithm works for any non-increasing kernel function, and non-negative or negative particle weights.

The approach is based on a dual-tree recursion where only those frontier nodes of the tree that have useful information are expanded. In other words we keep partitioning the regions of space that contain useful information. The algorithm allows the influence on each target point to be estimated within an error bound. In this work we set the error bound, ϵ , to 0.00001. By applying the Dual-Tree method, we also reduce the storage requirement from $O(M^2)$ to $O(M)$ since we do not need to explicitly store entries of $\mathbf{K}_{\mathbf{ll}}$ and $\mathbf{K}_{\mathbf{ul}}$ in $M \times M$ and $N \times M$ matrices.

Chapter 4

Experiments

We designed three sets of experiments. In the first part, we are tackling the problem of predicting the presence of a general object, for example 'a' car, in images. In other words, given a number of classes of objects, such as cars and motor-bikes, and a test image, the system decides if an object (from the specified object classes) is present in the image. We call the problem of this set of experiments the "Presence" problem.

The second set of the experiments deals with localizing a specific object class in the image. For example knowing that there is a car in an image we would like to be able to localize the car in the image. We limited ourselves to the problem of finding the centre of the object in a test image. We call this problem the "Localization" problem. In both parts, for training and test, we used images from PASCAL database, which can be downloaded from the PASCAL project web site, (<http://www.pascal-network.org/>).

In the third set of experiments, we compare the computational cost of three approaches: CG-DT, which is applying Conjugate Gradient and Dual-Tree in Gaussian Processes, CG where we are not using Dual-Tree and NAIVE which is the naive $O(M^3)$ approach.

Our approach performs poorly in many cases of Localization problem. However, we think that the problem is mainly in our insufficient local features and lack of

any geometry rather than in our choice of classifier, Gaussian Processes. Therefore, we think that the application of fast methods in Gaussian Processes for classifying SIFT features still will be useful for future research in this area. In section 7, we report the experimental results and discuss the possible ways that we can improve our approach.

4.1 Part 1 : Image Classification, the "Presence" Problem

In this part, we focus on 4 different classes of images: images that contain cars (CAR), images that contain motor bikes (MBIKE), images that contain people (PEOPLE) and those that contain none of them (NONE). We will perform multi-class classification with Gaussian Processes for this purpose. For training, we use the set of images provided by PASCAL database. We extract SIFT features of objects and construct a vocabulary for these four classes (CAR, MBIKE, PEOPLE and NONE) as we explained in section 2. For class NONE we use the features extracted from the background of training images. After Constructing the vocabulary for each object class, we build the covariance matrix for training features of all object classes as we explained in section 3. For each test image we also extract SIFT features. Then, we measure the similarities of pairs of features from the test image and the vocabularies of all classes using Gaussian Kernel, $k(h_i, f_j) = e^{-\frac{1}{\sigma} \|h_i - f_j\|^2}$, where h_i and f_j represent two features from the test image and the vocabulary. With these values, we form the similarity matrix \mathbf{K} . Then using Gaussian Processes we make a prediction for each feature of the test image. Once we get the matrix of response from computing equation (3.3), we have a set of responses for each test feature corresponding to each object class. In this application, we apply Gaussian Processes for classifying features into three object classes, CAR, MBIKE and people. If the responses of all classes are lower than a threshold for a test feature, we consider this feature to be in class NONE. We assigned the value of this threshold by cross-validation on a separate subset of training images. If at least one of the responses is greater than the threshold, we will consider the feature to be in the class for

which it has the highest value. By this operation we can label a group of features corresponding to classes of CAR, MBIKE, PEOPLE. The rest of labels will be considered in class NONE. The class with highest number of positive labels for features is the object class present in the image. If none of the features are labelled as CAR, MBIKE or PEOPLE, the image does not contain any of object classes. We train our classifier on 120 images from PASCAL database, 40 for each object class. We then performed the classification on 40 test images, 10 from each classes CAR, MBIKE, PEOPLE or NONE. The performance can be varied by choosing different values for S , the variable that we introduce in object vocabulary construction in section 2. In vocabulary construction, we choose S largest cluster of features. By increasing S more features will be represented as defining features of an object class.



Figure 4.1: *Images from different classes of CAR, MBIKE and PEOPLE.*

4.2 Part 2 : Binary Classification, the "Localization" Problem

In this part, we are interested in labelling SIFT features to binary classes. We apply equation (3.1) of section 3 to predict if features belong to a particular object class or to the background. For this binary classification task, we focused on class of cars. We trained our classifier with 60 images of cars. We extracted SIFT features from inside the boundary box (which is loosely segmenting the object from the background) and SIFT features from the background, that is outside the box. We constructed the

vocabulary for class object car and class object background. We obtained different sizes of vocabulary for different values of S . We labelled features from the vocabulary of cars with value 1 and we gave label 0 to the training features of the background. We then labelled the SIFT features of test images. From equation (3.1), we get a vector of responses correspond to the likelihood that a feature belongs to the object class. If the response for a feature is higher than certain threshold we considered that feature as a member of features of class of object car. Similar to the previous part precision and recall changes with different values of S . After labelling positive features, we averaged their positions in the image and marked this point in the image. Figure (4.2) shows this process. We considered the “Localization” task successful if this centre of features is located inside the object. We evaluated our approach on 20 random test images from ETHZ and TUGRAZ subsets of PASCAL database.

4.3 Part 3 : Computational Cost: Comparison of Three Approaches

In this set of experiments we implemented three approaches, NAIVE, CG and CG-DT in matlab and compared the computational time required for each of these approaches. This computation time included the training and testing time. However the time for vocabulary construction is not included since that part is fixed for all three approaches. We obtained different sizes of set of features, by different values for variable S and as a result different number of training features. We only classified the features of one image in this set of experiments in order to provide a fair comparison. This test image contains 810 SIFT features.



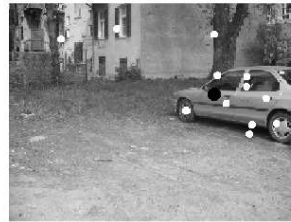
(a)



(b)



(c)



(d)

Figure 4.2: *Picture (a) shows a test image from the PASCAL data base. Picture (b) illustrates the SIFT features, extracted from the image. In (c), features have been labelled and positive features in the image have been marked with white circles. Picture (d) illustrates the centre of these positive features marked by a black circle.*

Chapter 5

Results

We evaluated the performance of our system in terms of accuracy and efficiency. For the first part of the experiment, that is the “Presence” problem, we measured the false positives, the images that wrongly classified in an object class and true positives, the images that have been correctly classified, for each class of CAR, MBIKE and POEOPLE. We then calculated the ratio of these numbers to the size of positive test set. We present the results in a plot in figure (4.2). Number of true positives and false positives increase by increasing the value of S , that is by including less frequent features into the object vocabulary.

In the second part of the experiment, we evaluated the performance by counting the number of images in which the centre of features has been marked inside the object. Out of 20 objects, only for 8 of them we achieved this goal. We think that there are several reasons behind this poor performance. One is that a number of background local features, such as local features of window in buildings, are very similar to local features of a car. To solve this problem we should find the set of discriminative features, that is the set of features that are seen in objects and not in backgrounds, and construct the object vocabulary based on them. Also we think that not considering any geometry in constructing our model and in classification task is another weakness of our approach.

As explained in section 6.3, we compared the computational cost of three approaches

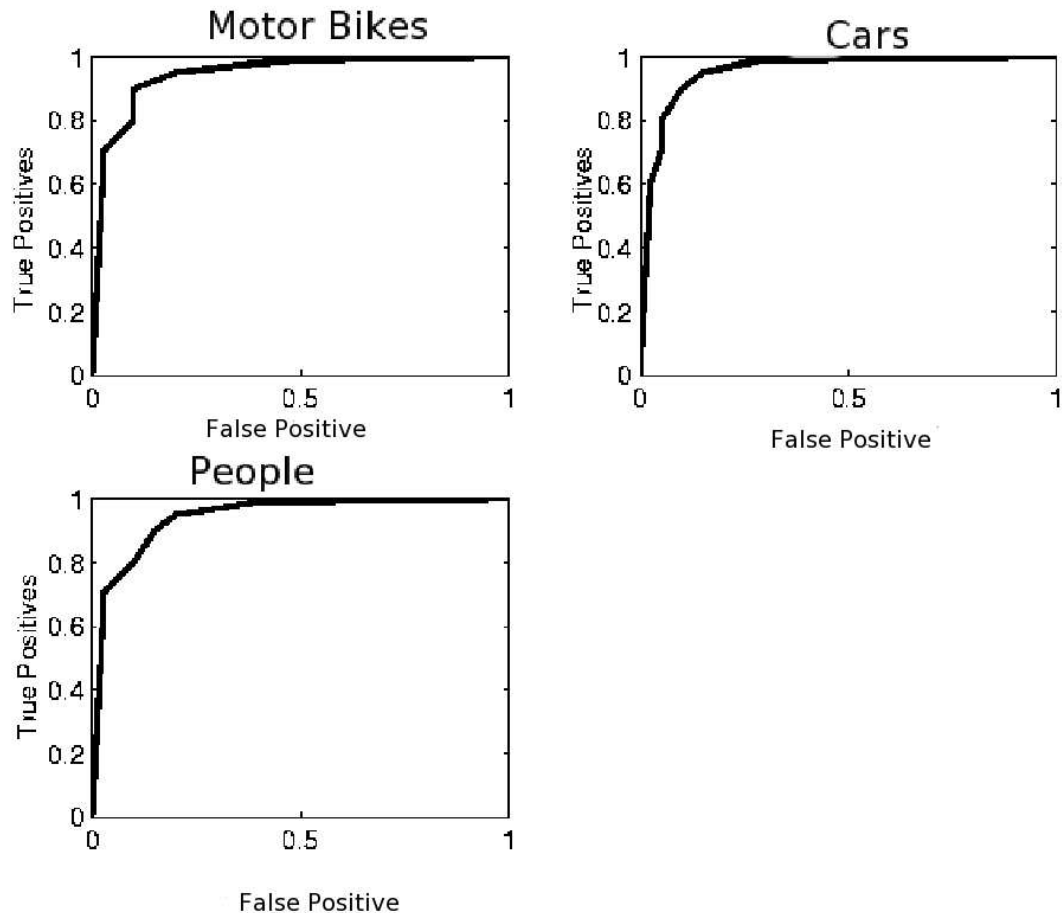


Figure 5.1: *The performance of our approach for the “Presence” problem is illustrated by measuring true positives and false positives for different classes and plotting ROC curves.*

(NAIVE, CG, CG-DT) on a single test image with different sizes for training set. This comparison is demonstrated in plot of figure (5). By increasing the size of set of test features to more than 2000, the difference between NAIVE and CG as well as CG-DT becomes obvious. By increasing the size of data set to more than 3000, CG-DT outperforms other approaches significantly. Note that since we are dealing with high dimensional data such as in 128-dimensional SIFT features, the constant cost of CG-DT is high. As a result the difference in performance becomes obvious in relatively large datasets. On the other hand, since the problem of object class

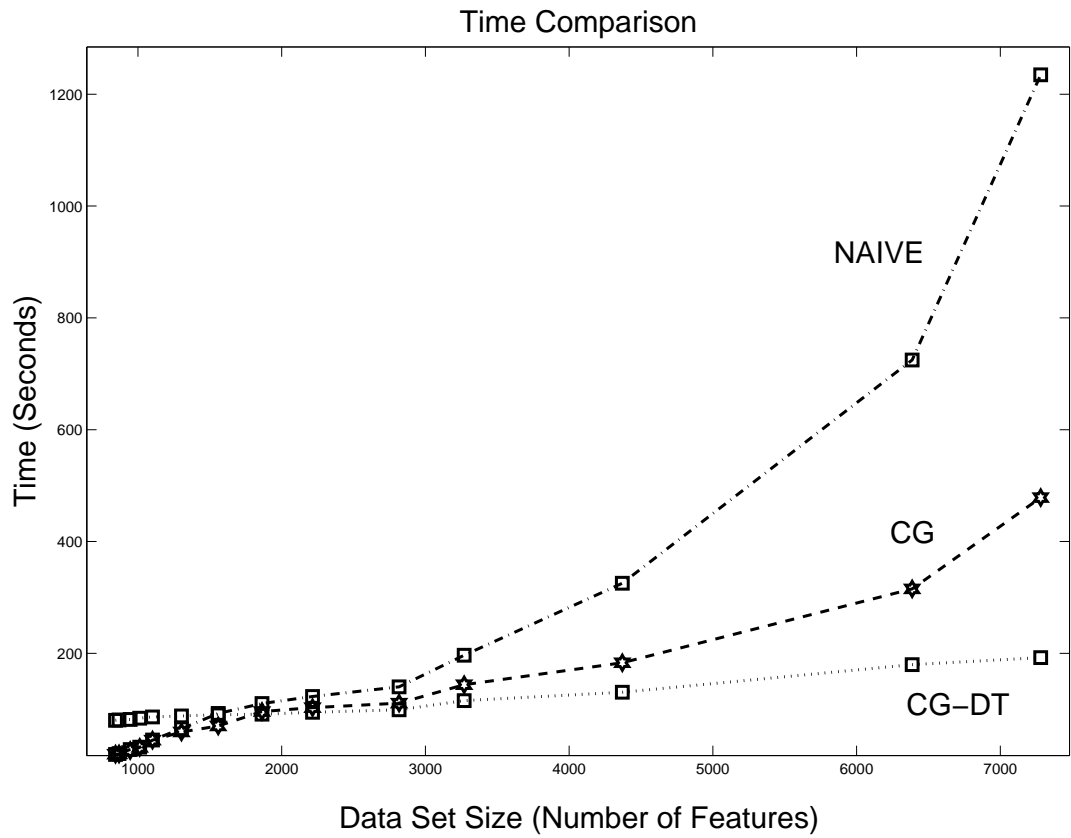


Figure 5.2: *CG-DT* provides a dramatic improvement in computational time as the data set increases in size.

recognition is complex, we need to have a large set of training features. Therefore, not only Conjugate Gradient but dual-tree are essential for practical performance.

Chapter 6

Conclusion

We have presented a novel approach for solving the problem of class of object detection using SIFT features and Gaussian Processes as our kernel-based classifier. Although the results show that our approach is not out-performing the state-of-the-art methods, considering that we only use bag of SIFT features, our classification method performs relatively well. We applied fast computational methods, Conjugate Gradients and Dual-Tree, and managed to reduce the computational time from $O(M^3)$ to $O(M \log M)$ and reduce the storage requirement from $O(M^2)$ to $O(M)$. As a result, Gaussian Processes becomes practical for classification of high-dimensional large datasets. We believe that by taking other features as well as the geometry of parts into consideration we can come up with a powerful object-class detector. Fortunately, since the computational cost is almost linear for large datasets now, the high computational cost of Gaussian Processes is not a concern anymore.

Chapter 7

Acknowledgements

I would like to thank my supervisor, Dr. Nando de Freitas, from whom I learned the most in my undergraduate studies. I also thank Dr. David Lowe and Dr. Kevin Murphy for their helps and insights.

Bibliography

- [1] M Belkin and P Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation **15** (2003), no. 6, 1373–1396.
- [2] J W Demmel, *Applied numerical linear algebra*, SIAM, 1997.
- [3] M N Gibbs, *Bayesian gaussian processes for regression and classification*, PhD Thesis, University of Cambridge, 1997.
- [4] A G Gray, *Fastkernel matrix-vector multiplication with application to gaussian process learning*, CMU-CS-04-110, 2004.
- [5] A G Gray and A W Moore, *thesis*, 'N-Body' Problems in Statistical Learning, 2004.
- [6] D Bau L Trefethen, *Numerical linear algebra*, SIAM, 1997.
- [7] D Lang, *Fast methods for inference in graphical models and beat tracking the graphical model way*, MSc Thesis, University of British Columbia, 2004.
- [8] D G Lowe, *Object recognition from local scale-invariant features*, ICCV, 1999.
- [9] L V Gool M Everingham, *Pascal visual object class challenge results*, <http://www.pascal-network.org/challenges/VOC/voc/index.html>, 2005.
- [10] B Fraser F Hamze M Mahdavian, N de Freitas, *Fast computational methods for visually guided robots*, IEEE International Conference on Robotice and Automation, Barcelona, Spain, 2004.
- [11] D J MacKay M N Gibbs, *Effient implementation of gaussian processes*, <http://www.inference.phy.cam.ac.uk/mng10/GP/gpros.ps.gz>, 1997.
- [12] D G Lowe S Helmer, *Object recognition with many local features*, Workshop on Generative Model Based Vision 2004 (GMBV), Washington, D.C., 2004.