Generic Object Recognition with Probabilistic Models and Weak Supervision

Yatharth Saraf
Department of Computer Science and Engineering
University of California, San Diego

ysaraf@cs.ucsd.edu

Abstract

Real world images of objects belonging to a particular class typically show large variability in shape, appearance, scale, degree of occlusion, etc. Thus, a major challenge for generic object recognition is to develop object models that are flexible enough to accommodate these large intra-class variabilities. Such powerful models, in turn, require large amounts of training data to be effective and it becomes imperative to reduce the degree of human supervision required to a minimum. In this project, parameterized probabilistic models were used to explicitly model different object attributes and these parameters were estimated by maximizing the likelihood of training data. The training data used was mostly from the Caltech dataset with labels for categories. This project is inspired by the work of Fergus et al. (2003).

1. Introduction

Generic object recognition deals with recognizing the category of a given object as opposed to recognizing specific, individual objects. The approach used here was to use generative models to represent object categories. This is different from discriminative approaches to object categorization such as SVMs.

This project was concerned with recognizing classes of objects based on weakly supervised learning. Training data was used to build representative models for the classes which could be used in conjunction with a Bayesian approach for categorizing a new, previously unseen image into one of the learned classes. The class representations, model learning and recognition methods were selected in a manner similar to the work of Fergus et al. [3].

The classes are represented with a “constellation of parts” model that relied upon an entropy-based feature detector [6] to detect regions of interest within an image. These regions of interest were used to represent the images both during learning and recognition.

One serious drawback of the method seems to be that it entails fairly high computational complexity and does not scale very well with the number of parts detected. The problem is that the set of hypotheses (for the valid assignment of detected features to the “parts” in the object model) is very large and this entire hypothesis space must be explored in order to compute the likelihood function during both learning and recognition. The problem of slow recognition is more serious than that of learning (since learning typically happens off-line). The root of this problem is actually the presence of occlusion and clutter in real-world images. In trying to deal with background clutter, an exhaustive search over an exponentially large hypothesis space is forced during both learning and recognition. In this project, it was decided to ignore the problem of background clutter and assume that all the features arise from the object and not from background. This allowed a significant speedup and made the algorithm much more tractable. However, this means that the images would need to be segmented into foreground and background as a pre-processing step, which is not at all a trivial task. If automatic segmentation is not reliable, then the level of manual supervision here is much higher than in Fergus' [3] approach. Thus, the trade-off is between computational tractability and manual supervision and here the choice is made in favour of tractability.

Some of the applications of such a system could automated image database annotation and retrieval, video surveillance, driver assistance, autonomous robots and cognitive support for the visually impaired.

2. Datasets

The training and testing was primarily done on the Caltech Cars (Rear), Caltech Motorbikes and Caltech faces datasets. These can be obtained from *http://www.robots.ox.ac.uk/∼vgg/data*. The Caltech motorbikes dataset had about 800 motorbikes but only images without background clutter were selected for training and testing. The images were used after grayscaling so colour information in the images was discarded in this approach.

3. Outline of the method

Assuming that the images are free of clutter and occlusion, the basic steps for the training phase are:

1. Detect salient regions in all training images using the Kadir-Brady feature detector.

2. Extract the X and Y coordinates, scale and square intensity patches around detected features (of a size equal to the scale of the feature).

3. Rescale the appearance patches to 11×11 and reduce the dimensionality of the patch from 121 to 16 using PCA.

4. Estimate model parameters. The basic model is essentially a single full Gaussian for the X and Y coordinates of feature locations, and one diagonal Gaussian per object part.

The steps for the testing phase are:

1. Extract features of test images in the same manner as in the training phase.

2. Use the learnt model to estimate the probability of detection of each test image.

3. Classify the test images using Bayes' Decision Rule.

The next few sections explain these steps in detail and show results.

4. Detecting Salient Regions

The feature detector used for detecting locations and scale of salient image features was the one by Kadir and Brady [6, 5]. The motivation for using this particular detector is that it is supposed to be stable across different scales and the number of features detected is easily controllable. A sample run of the feature detector is shown in Figure 1 with the detected salient regions marked by circles in the picture. The first image in Figure 1 probably has too many features detected. The desired number of features should be around 30. However, the number of features detected can be controlled by tweaking the parameters of the feature detector appropriately. With a different setting for the starting scale, stopping scale, threshold on saliency, etc. we can get a reduction in the number of detected features. The new detections are shown in the second image.

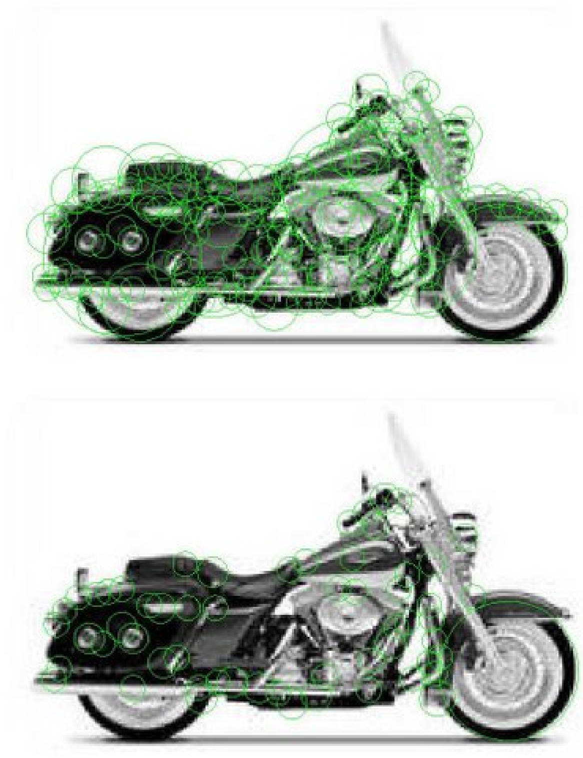

Figure 1. Output of the Kadir-Brady feature detector with different parameter settings. The circles represent the locations and scales of salient regions.

4.1. Characterizing Appearance

The Kadir and Brady feature detector picks out a bunch of salient features from the image and gives their locations and scale as shown in Figure 1. For notational convenience, the locations and scales for all these features are aggregated into the vectors **X** and **S**. The third key source of information is appearance and we need to compute the vector **A** for a given image, which will contain the appearances of all the features.

For computing appearance of a single feature, it is cropped out of the image using a square mask and then scaled down to an 11×11 patch. This patch can be thought of as a single point in a 121-dimensional appearance space. However, 121 dimensions is inconvenient because it will increase the complexity of the model that will be learnt in the learning phase. Thus, the dimensionality of the appearance space needs to be reduced. This is done using PCA [8] and selecting the top 16 components.

During the learning phase, a fixed PCA basis of 16 dimensions is computed. This fixed basis is computed by using patches around all detected regions across all training

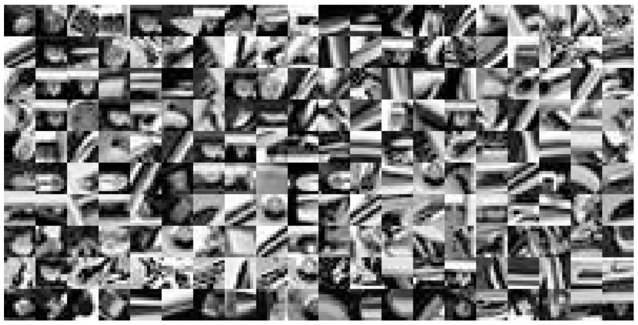
Figure 2. Appearance patches extracted from the 20 most salient features from 10 motorbike images (Caltech dataset). Each row shows the patches extracted from a single training image.

images. The basis is computed per category. An alternate approach is to compute a single basis for all categories by using patches from training images of all classes.

Figure 2 shows the appearance patches extracted from motorbike images in the Caltech dataset. However, the features in Figure 2, are not very encouraging as it is quite difficult even for a human to look at those appearance patches and identify them as belonging to motorbikes. These patches were extracted after a feature detection phase that was similar to the detections in Figure 1 (second image). For comparison, the results of feature detection from Fergus' work [4] are shown in Figure 3. From this, it can be seen that the problem seems to be the scale of the features detected with small, local features firing more strongly than more important larger features.

To get around this problem, I gradually increased the smallest scale admissible for detected features and finally settled on a starting scale of 23 (earlier it was 3). Using this value for starting scale and choosing the top 20 saliency values, the feature detector outputs on various bike images are shown in Figure 4. This looked a lot better and closer to the output of Fergus et. al. (Figure 3). Appearance patches around these new features are extracted, resized and tiled and shown in Figure 5. These appearance patches of the parts seem to provide more information about the image's category as the tyres of the motorbikes can be clearly seen in almost all the input images.

4.2. Extracting appearance features from faces and cars

While the appearance extraction process works well for motorbikes with a starting scale of 23, it wasn't clear if a single scale would work be appropriate for all categories. The detected features for faces with the same starting scale are shown in Figure 6. The corresponding tiled appearance patches are shown in Figure 7. The patches extracted do seem to capture face parts although not in all cases. A smaller starting scale might work better to allow the detection of small but important parts such as the eyes, nose, etc. However, it defeats the whole purpose of the experiment if one must tweak the starting scale for each different type of category because there must be minimal human supervision during both training and testing. This serves to somewhat illustrate the heavy dependence on the feature detector. Similar results for cars are shown in Figures 8 and 9. Car parts don't seem to be characterized well enough at this scale.

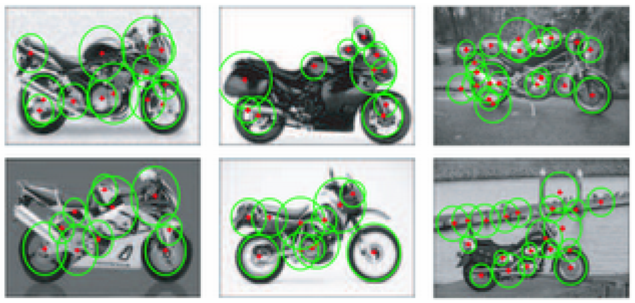
Figure 3. Feature detection results on motorbike images in the work of Fergus et al. [4]

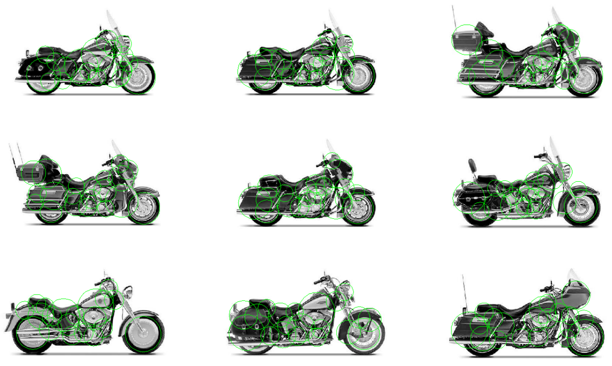
Figure 4. Feature detector outputs on motorbike images using a starting scale of 23.

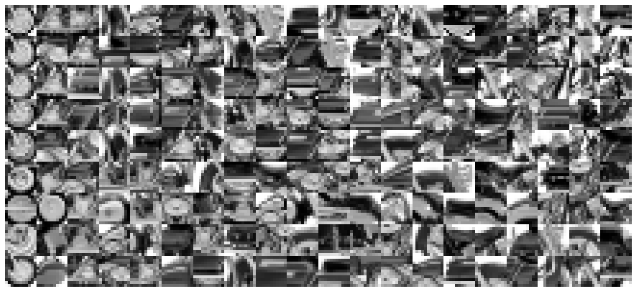
Figure 5. Appearance patches extracted with a starting scale of 23. The 9 rows show the rescaled features (into an 11 x 11 patch) extracted from the 9 motorbikes shown in Figure 4.

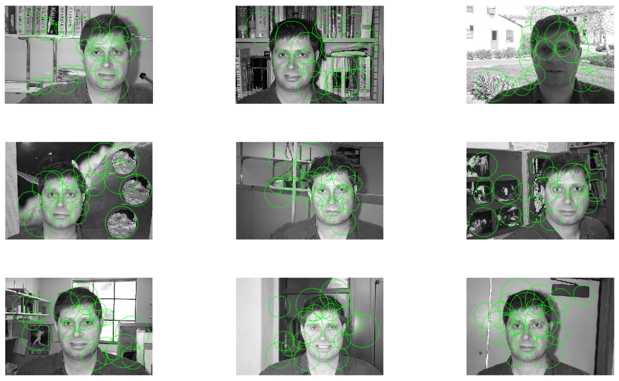
Figure 6. Feature detection for face images with a starting scale of 23.

Figure 7. Appearance patches extracted for face images with a starting scale of 23.

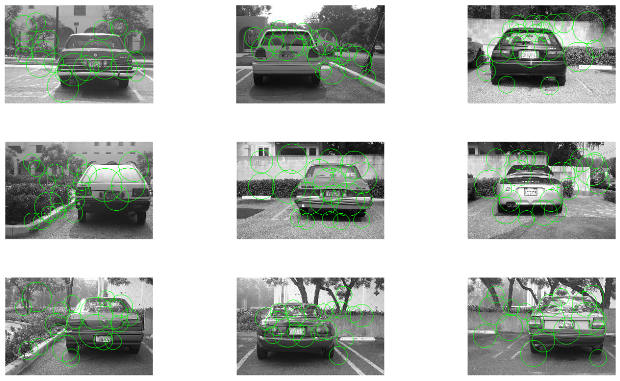
Figure 8. Feature detection for car images with a starting scale of 23.

5. Model learning sans clutter and occlusion

The main complications in this method arise from trying to deal with occlusion and clutter. That's what forces an exhaustive search over an exponentially large hypothesis space during both learning and recognition. To avoid getting into exponential search, it was decided to work with clean data and assume all the features arise from the object and not from background. This is the case with a lot of the images in the Caltech motorbike dataset.

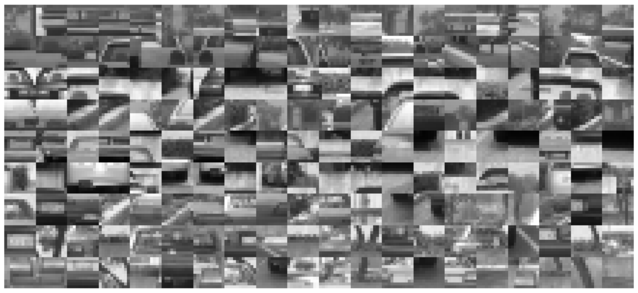
Figure 9. Appearance patches extracted for car images with a starting scale of 23.

Based on this idea, an experiment was run with 20 training images and using the top 10 salient features for learning. Since the data is clean, all features arise from the object and the number of parts is equal to the number of features used. In this formulation, there is no hidden variable and the parameters for the appearance of each part can be directly estimated using Maximum Likelihood (ML) estimation. In addition to appearance, the ML parameters for the joint density of the locations of all the parts are also computed. Then, using these parameters, the recognition procedure was run on the images shown in Figure 10. The first three images were selected from within the training set of 20 images. Thus, the probability of recognition is expected to be high for these. The last image is selected from outside the training set and is deliberately chosen to be quite dissimilar from the training images. While running the code for recognition, there were numerical issues due to the location parameters being ill-conditioned. The covariance matrix of the joint Gaussian density for the locations of the parts was nearly singular. This was probably due to the fact 20 training images is not really enough data. Also, there was no ordering constraint on the X coordinates of the features detected. From this rudimentary test, the log probabilities for recognition from just the appearance models were -50.9192, -54.2892, -57.3182 and -792.5911 for the 4 images respectively. The fourth image would be expected to have a lower matching probability as it is quite different in appearance from the other motorbike images in the training data.

As 20 training images seemed too few, the experiment was re-run with 47 clean training images of motorbikes. This prevented the problems of numerical ill-conditioning thereby allowing the use of the location model as well as appearance. The recognition procedure was run on 9 different test images consisting of motorbikes, cars and faces (Figure 11). The resulting log-probabilities are shown in Figure 12. The X-axis in the log-probability graphs represents the image index as defined in Figure 11. Therefore, images 1-4 were positive test images, images 5-7 were neg-

Figure 10. Examples of clean motorbike test images used for model learning and recognition

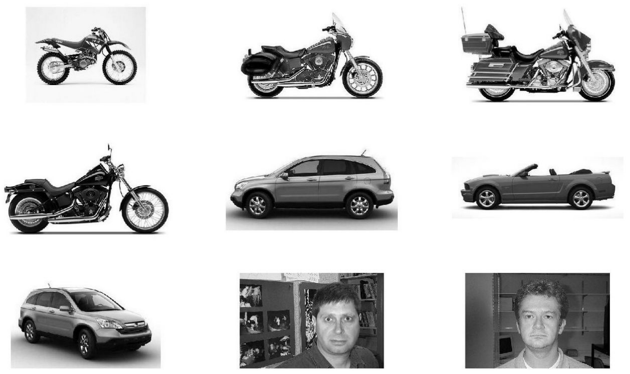

Figure 11. The 9 test images used. The images are indexed in row major order with the images 1-4 as positive motorbike test images and images 5-9 as negative cars and faces test images. These are the image indexes used along the X-axis in all the log-probability graphs that show the test results.

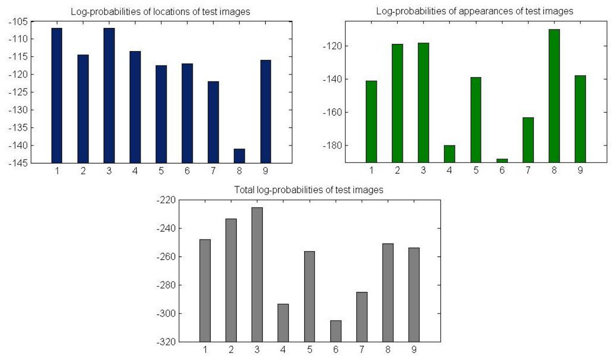

Figure 12. Log probabilities for the 9 test images in Figure 11, with features sorted by saliency, from (a) the location model, (b) the appearance model and (c) sum of the appearance and location models.

ative test images of cars and images 8-9 were negative test images of faces.

From Figure 12 (a), we see that there is a separation between the first 4 motorbike images and the next 5 images. However, the appearance model does not seem to give such a clean separation (Figure 12 (b)). The combined log-probabilities (Figure 12) (c) show a high recognition probability for the first 3 motorbikes although the fourth motorbike has an undesirably low recognition probability.

In all these experiments, the detected features were sorted by saliency. The problem with this is that it isn't clear if there exists any correspondence between detected features based on saliency. As an illustration, Figure 5 shows that the first column (the most salient feature) is consistently the right wheel. However, there is no such regularity with the rest of the columns.

As suggested in Fergus et al. [4], an ordering constraint was imposed by sorting the selected features by X-coordinate rather than saliency. Figure 13 shows the patches extracted from the 47 training images when sorted in this way. The correspondences within columns seems better than in Figure 5. After imposing this ordering constraint, the resulting log-probabilities for the 9 test images in Figure 11 are shown in Figure 14.

From Figure 14 (a), we see a larger separation between location probabilities of motorbikes and negative test images than in Figure 12 (a), except for the fifth image (a car image). This is easily explained by looking at the features detected for that image, shown in Figure 15. The spatial locations of the features are almost exactly the same as for most motorbike images, so a high location probability for this image is not surprising. Also, the ninth image (a face) has the lowest location probability which is similarly explained by Figure 16. The locations of the detected features are quite different from what we would expect for motorbikes.

The appearance probabilities in Figure 14 (b) are also much better separated than in Figure 12 (b), except that the fourth image (a motorbike) is showing an undesirable dip. This is also causing a drop in its total log-probability (Figure 14 (c)). The appearance patches extracted from the test images are shown in Figure 17. The corresponding reconstructed patches in the image domain from the truncated PCA representations of the original patches are shown in Figure 18.

5.1. Throwing in more Gaussians

The variability in the appearance of a single part across different training images in Figure 13 suggests that a single Guassian may not be sufficient in capturing the underlying data. To account for the multi-modality of the appearances of the parts, a mixture of Gaussians was used to model each part with each mixture component assumed to have diagonal covariance matrices (using full covariance matrices was causing numerical problems with 47 training images). The Netlab software [7] turned out to be very useful as it has

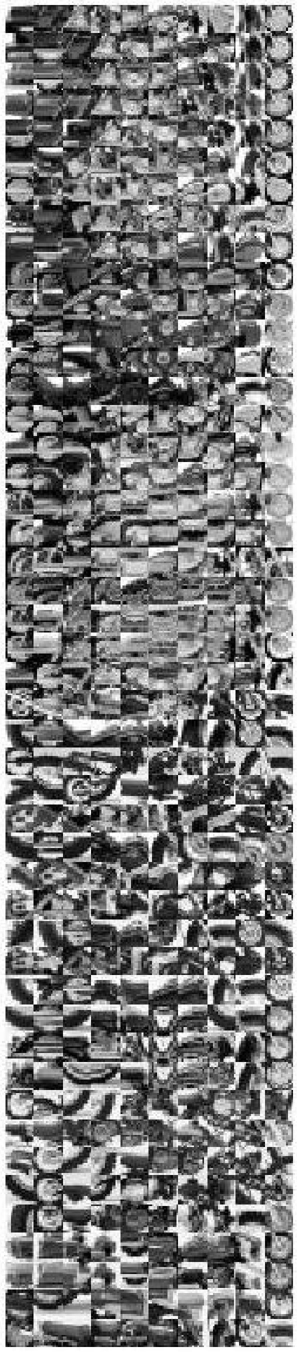
Figure 13. Appearance patches extracted from 47 motorbike training images. Each row represents patches cropped out around the 10 most salient regions in a single training image, sorted by X-coordinate.

in-built routines for learning Gaussian mixture models using the Expectation-Maximization (EM) algorithm. Specifically, the scripts `gmm`, `gmminit`, `gmmem` and `gmmprob` were a big help.

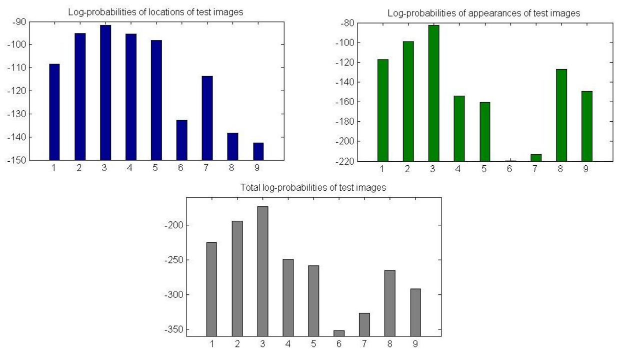

Figure 14. Log probabilities for the 9 test images in Figure 11, with features sorted by X-coordinate, from (a) the location model, (b) the appearance model and (c) sum of the appearance and location models.

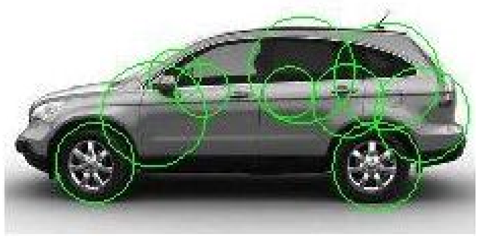
Figure 15. Features detected for the 5th test image. The spatial locations of detected features are similar to motorbike features.

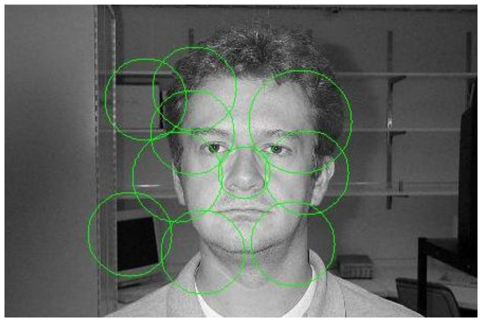
Figure 16. Features detected for the 9th test image. The spatial locations of detected features are quite different from motorbike features.

First, the default EM initialization was used (uniform priors, random means and identity covariances). The resulting log-probabilities, when using 2 mixture components for each part's appearance, are shown in Figure 19. Comparing these appearance probabilities with those in Figure 14, from using a single Gaussian, there is actually a slight degradation in the results and the separation between positive and negative test images is not as clean. This degradation is possibly due to the fact that there isn't enough data.

The `gmminit` script initializes the centers and priors us-

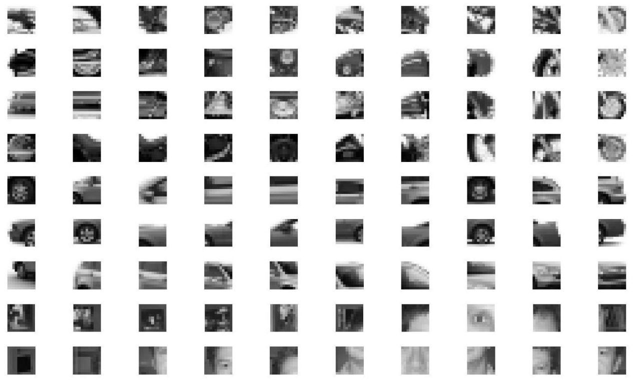

Figure 17. Patches extracted from the 9 test images in Figure 11. Each row shows patches from a single test image. The first four rows are from motorbike test images (positive test cases) while the remaining are from negative test cases (cars and faces).

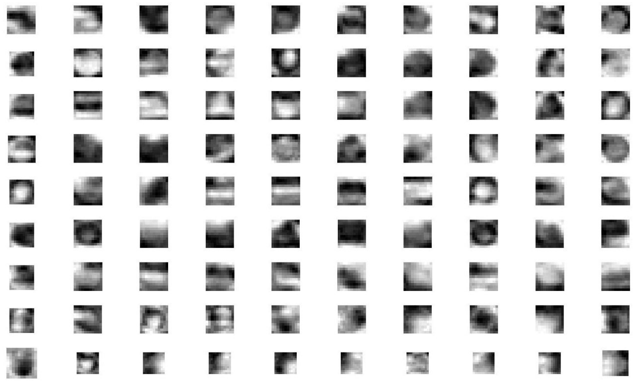

Figure 18. Reconstructed appearance patches in the image domain obtained from truncated PCA representations of the patches shown in Figure 17. These patches have been scaled to use the full colormap for display purposes.

ing k-means on the data. The covariance matrices are calculated as the sample covariances of the points closest to the corresponding centers. When EM was initialized in this way, the resulting log-probabilities obtained are shown in Figure 20. There isn't too much difference from the earlier run with default EM initialization.

Figure 21 shows the results from a run where three components were used instead of two. The results are slightly better with the probabilities of the positive test images having been pushed up a bit relative to the negative test images.

5.2. Reducing dimensionality with random projections instead of PCA

Instead of reducing the dimensionality of the appearance patches using PCA, a random projection matrix [1] was also tried for comparison's sake. The matrix was generated as $G \in \mathbb{R}^{d' \times d}$ with entries $G_{ij} \sim \mathcal{N}(0, 1/d')$. Here the projected appearance vectors had dimension $d' = 16$, reduced from the full dimensionality ($d = 121$). The matrix was generated once during training and the same one was used again during testing.

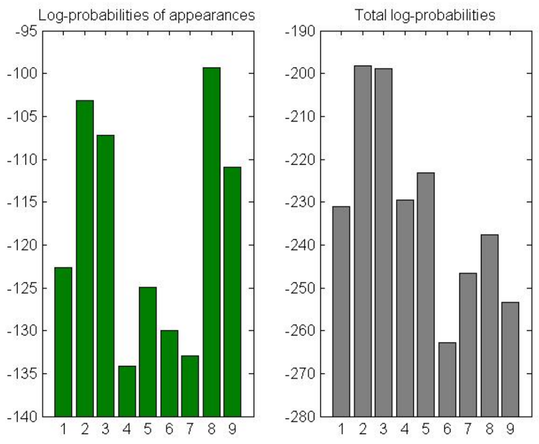

Figure 19. Log probabilities for the 9 test images in Figure 11, with features sorted by X-coordinate, from (a) the appearance model with **two** Gaussians per part and (b) sum of the appearance and location models (The location probabilities are unaffected by the change in the appearance model and so are the same as in Figure 14).

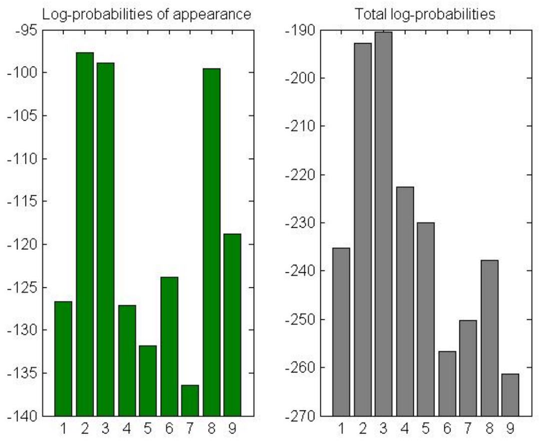

Figure 20. Log probabilities for the 9 test images in Figure 11, with features sorted by X-coordinate, from (a) the appearance model with **two** Gaussians per part and (b) sum of the appearance and location models. Here the EM algorithm was initialized with k-means and sample covariances (The location probabilities are, again, the same as in Figure 14).

The log-probabilities resulting from this approach are shown in Figure 22. Image 1 has taken an undesirable dip and image 4 hasn't been pulled up enough from the other negative test images. This approach does not seem to work

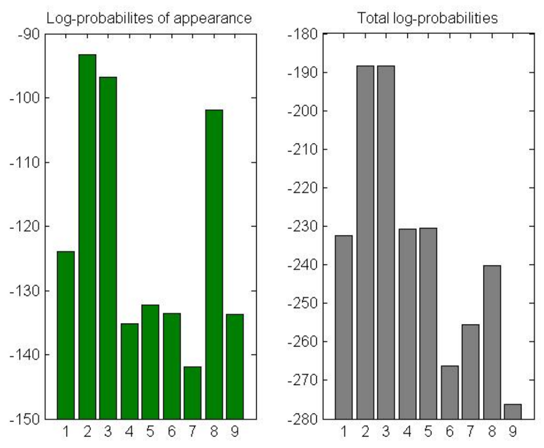

Figure 21. Log probabilities for the 9 test images in Figure 11, with features sorted by X-coordinate, from (a) the appearance model with **three** Gaussians per part and (b) sum of the appearance and location models. Here the EM algorithm was initialized with k-means and sample covariances (The location probabilities are, again, the same as in Figure 14).

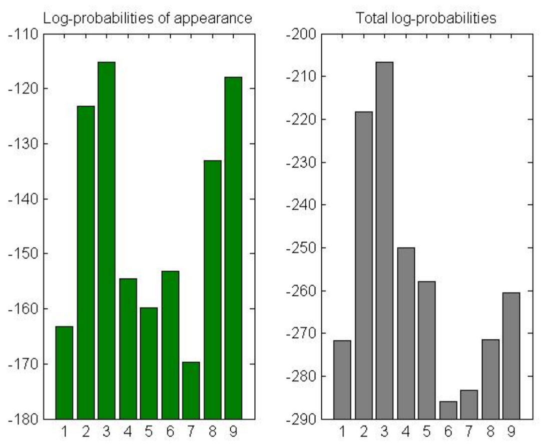

Figure 22. Log probabilities for the 9 test images in Figure 11 from (a) the appearance model with dimensionality reduced by random projections instead of PCA, and (b) sum of the appearance and location models. (The location probabilities are, of course, unchanged from Figure 14).

significantly better than the previous PCA approach. Besides, it was found that there was some variability in the results across different runs when using random projections. PCA, on the other hand, has the added advantage of ensuring repeatability. Thus, it was decided to stick with PCA as the preferred method of dimensionality reduction.

Figure 23 shows the reconstructed patches obtained by projecting the reduced 16-dimensional vectors back to 121 dimensions. This was done by multiplying the reduced dimensionality patches by the pseudo-inverse of the random projection matrix G. There doesn't seem to be any clear statistical regularity for the first four rows compared to the last five rows. By comparison, Figure 18 illustrates that PCA gives a much clearer reconstruction.

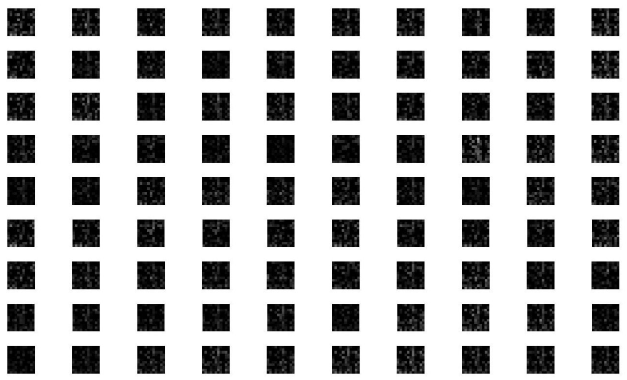

Figure 23. Reduced appearance patches from the test images in Figure 11 projected back to the full 121 dimensions and shown as 11×11 image patches. Again, the first four rows are from motorbike test images (positive test cases) while the remaining are from negative test cases.

6. Nearest-Neighbour Experiments

In an attempt to get more information out of the appearance patches extracted from the images, a nearest-neighbour (NN) experiment was run. The method used to compute distances between the appearances of images is analogous to the computation of Levenshtein (edit) distances [2] between strings. The set of appearance patches extracted by the feature detector from a single image (and ordered by X-coordinate) can be thought of as a string, with each appearance patch acting as a character in the string. Therefore, a single image gives rise to a single string of appearance patches. Appearance patches extracted from training images were stored in their rescaled (but non-dimension-reduced) 11×11 form. Dynamic programming is then used to compute the Levenshtein distance between two such strings extracted from a training and a test image. This distance is used for NN-computations of test images.

For computing the Levenshtein distance between two strings, there is a cost associated with matching characters across strings and also a cost for inserting "gaps" in place of characters in either string. Taking the analogy further, in our case, the cost of matching one appearance patch (character) with another was computed using a straightforward SSD between their intensities. The cost of inserting a "gap" was dynamically computed as the matching cost (SSD) of the patch being matched to a gap and a canonical 11×11 patch having uniform intensity of 0.5 (which can be thought of as a non-informative, "gap"- equivalent for an appear-

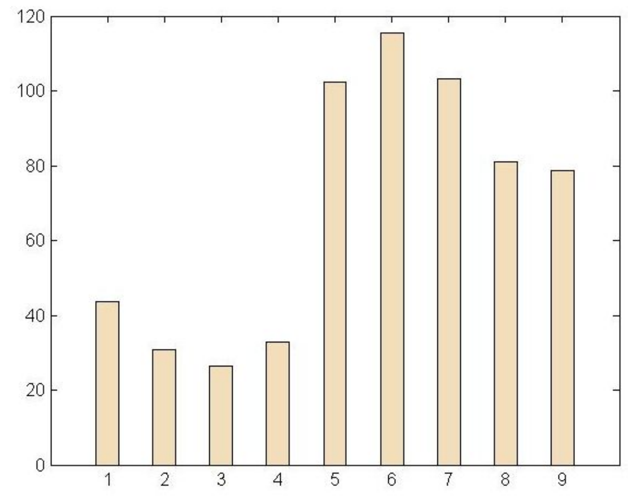

Figure 24. 1-nearest-neighbour distances from the 9 test images in Figure 11 to 47 clean motorbike training images.

ance patch).

Figure 24 shows the resulting 1-nearest-neighbour (1-NN) distances from the 9 test images in Figure 11 to the same 47 motorbike images that were used for training the generative model. A clear separation between the 1-NN distances of the positive and negative test cases can be seen, with the first four motorbike test images showing much lower distances (as expected) than the car and face images.

7. Conclusions and Future Work

The method used here was found to be quite fast for both learning and recognition. This was mainly because the problems of exponential search were avoided by assuming that the training and test data were without background clutter and occlusion. There was a heavy reliance on the performance of the feature detector and its parameters had to be tweaked carefully in order to ensure that meaningful features were obtained. In particular, the starting and stopping scale for the detector had to be set carefully for motorbike images and this setting may not be best suited for every object category. Changing the scale of the images was found to throw off the feature detector and thus, the images were assumed to be approximately scale-normalized. Perhaps a different, multi-scale feature detector might address this issue better.

The appearance model was a bit inconsistent in separating positive and negative test images, at least for the amount of training data used. In terms of future work, experiments with more clean training and test data and with data from multiple object categories might be informative. Also, exponential search could be tried in the manner of [4] in order to deal with clutter and occlusion. However, this would increase the running time dramatically for both learning and recognition. As for the location model, a Gaussian mixture model for the locations of the parts might be useful assuming that enough data is available. In addition, incorporating translation invariance by centering the coordinates, and some kind of scale normalization might be beneficial. It was also seen that sorting the features by X-coordinate rather than saliency helped in better separating the categories. A robust assignment of detected features to model parts may help further. Figure 18 suggests that the appearance model might be improved by using some kind of brightness normalization and using intensity gradients rather than the raw intensity values.

The use of Levenshtein distances in a “bag-of-features” with nearest-neighbour framework was found to be quite promising. This could be an interesting avenue to explore especially with multiple object categories. Again, sensitivity of the Kadir-Brady detector to scale changes might be an issue and various feature detectors may be tried. The use of a multi-scale detector along with jitter distances to compute matching costs between vectorized appearance patches might be beneficial.

Acknowledgements

I’d like to thank Prof. Serge Belongie for having given me the opportunity to work on this project and for extending invaluable help and suggestions throughout its course. Thanks are also due to the CSE 190A Winter 2008 class for useful discussion during the quarter.

References

[1] S. Belongie. http://www.cse.ucsd.edu/classes/fa07/cse252c/hw2.pdf. 7

[2] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw Hill, 2007. 8

[3] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *Proc. of the IEEE Conf on Computer Vision and Pattern Recognition (CVPR)*, 2003. 1

[4] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In H. Nagel, editor, *Cognitive Vision Systems*. Kluwer Academic Publishers, in press – 2006. 3, 5, 9

[5] T. Kadir. http://www.robots.ox.ac.uk/∼timork/salscale.html. 2

[6] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision. 45 (2):83-105*. 1, 2

[7] I. Nabney and C. Bishop. http://www.ncrg.aston.ac.uk/netlab. 5

[8] N. Vasconselos. http://www.svcl.ucsd.edu/courses/ece271a-f06/handouts/dimensionality.pdf. 2