Object Recognition using Template Matching

Nikhil Gupta, Rahul Gupta, Amardeep Singh, Matt Wytock

December 12, 2008

1 Introduction

Object Recognition is inherently a hard problem in computer vision. Current standard object recognition techniques require small training data sets of images and apply sophisticated algorithms. These methods tend to perform poorly because the small data set does not reflect the true distribution (selection bias).

Recently, Torralba et al [1] have proposed to develop a large data set of images (80 million images) and apply simple algorithms for object recognition. Their method performs relatively well for some certain classes of objects. Nevertheless, their data sets require very large storage and are noisy.

In this project, we develop precise 3D models of objects and use these to apply simple learning algorithms for object recognition. Three dimensional models have the advantage that they are more compressed than individual images. Thus this allows training large scale object recognition algorithms. We hope to prove that by applying a simple learning algorithm, namely template matching, we are able to achieve strong performance in recognizing objects.

In order to focus our efforts, we are concerned only with the classification task: given an image does it or does it not contain the target object? This task is the focus of much research in computer vision and in this report we will compare our results to the state of the art from the PASCAL Visual Object Classes (VOC) Challenge, 2007 [3]. Further, we chose cars as objects to detect as detecting cars in images is a general problem, and we could make accurate 3D models of the cars ourselves using scaled diecast models.

This report is organized as follows. Section 2 details how we created the 3D car models. Section 3 provides details of basic approach we followed. Section 4 and 5 give experimental results along with some other variations of our approach. Section 6 details an alternate K-means based approach to the problem. Finally, section 7 concludes the report.

2 Building 3D models

An important aspect of our project is the accurate construction of a 3D model of our target object. To this end, we acquired 6 high quality diecast car models [7]. These models are accurate replicas of real cars at an 18:1 ratio. We chose cars representative of the general distribution of car models on the road. 3D models of the car models were built using the NextEngine 3D laser scanner [8] in the Stanford AI lab. This process turned out to be more challenging and time consuming than initially anticipated. Often multiple scans had to be taken for the cars as they have very shiny surface, often requiring a coat with powder so as to make their surfaces less reflective. Also, some areas were still missed in the built model, and required multiple scans to be able to build an accurate 3D model.

3 Template matching with 3D models

Once we have acquired a 3D model, we need a method to match it to a target images. We build on the simple template matching techniques described by Le et al [2]. This is our method of matching a 3D model to a target image I:

1. Using the 3D model, generate a 2D projection at some pose (ϕ, ψ, θ) and scale (z). This will be the template image, T.
2. Convert T and I to grayscale, if necessary.
3. Apply a Gaussian blurring filter to T and I.
4. Apply Canny edge detection to T and I.
5. Apply another Gaussian filter to T and I.
6. Find best match in I for T using template matching and normalized sum of squares difference difference metric.

The goal of the gaussian filters and the edge detection is to provide invariance to changes in color and slight changes in structure and rotation while maintaining a general representation of the object structure from the 3D model. The parameters of the gaussian filters and canny edge detection are parameters of the method that we tuned by hand to give the best performance on our datasets.

In Step (1) there is an infinite space of 2D projections that can be generated from the 3D model. In fact, it is a strength of our approach that our 3D model is a complete representation of the object. However, in practice, naively iterating over all poses and scales and generating a 2D projection in ths fashion is computationally expensive. Therefore, for the purposes of this project, we restrict our attention to a limited set of poses.

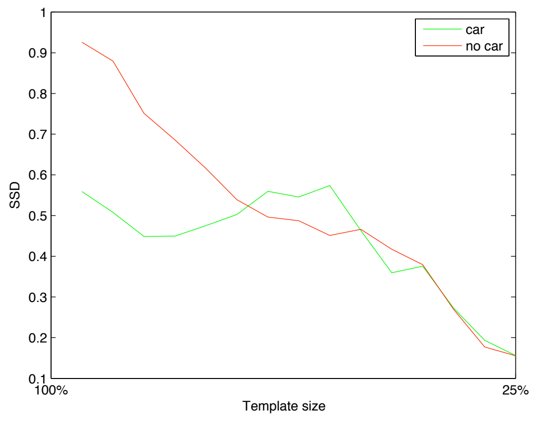


Figure 1: The same template T, scaled to different sizes and compared to image I.

3.1 Template score dependence on size

One phenemenon that we observed when comparing scores from template matching is that there is a strong correlation between the score of the template and its size. Considering the normalized sum of square differences scoring function that we are using to compare the template T and the image I:

$$\frac{\sum_i \sum_j (T(i,j) - I(x+i, y+j))^2}{(\sum_i \sum_j T(i,j)^2)(\sum_i \sum_j I(x+i,y+j)^2)}$$

When template size approaches 0, the score also decreases toward 0 as the minimization problem becomes easier. For example, a template of size 1 is very likely to find its exact match in the image. We observed this empirically, see figure 1. Note that there are other scoring functions used for image comparison when template matching, such as cross correlation, but they all have this same property.

An interesting question that arises, therefore, is to define the optimal loss function for comparing templates of different sizes. We did not derive such a function as part of this project, however it is our intuition that the loss function should be based on the premise that a random template of size M and a random template of size N ($M >> N$) should have approximately the same score when template matching against a given image I.

3.2 Parametric model based on template matching

As discussed above, its unclear how to compare template matches of different sizes. Furthermore, we want to match templates showing the target object at different 2D rotations. Therefore, we pose a supervised learning problem in which the output space $Y = \{0, 1\}$ represents “car” or “not car” and the input feature vector $X = \{x_1, ..., x_n\}$ is composed of our scores from template matching. The dimensionality of the feature vector will be equal to M x N x K, where M is the number of models, N is the number of poses and K is the number of scales.

Based on figure 1, it is our intuitive sense that positive examples should be differentiated from negative examples by an inflection point (local minimum) that represents a good match at a given scale. Whereas negative examples should have scores that decrease monotonically as the template size decreases.

4 Easy data set: 2 poses, single scale

The first data set that we attempted to classify was from the UIUC Image Database [4] for car detection. It contains 1,050 labeled examples of car side views (550 positive, 500 negative) at a single scale. We partitioned these examples randomly into a training set (90%) and a test set (10%).

In order to match our 3D model to this set, we used 1 model, with 2 poses and 15 scales to generate 30 features. We then performed logistic regression on these features to find their optimal weights. Our results on this simplistic data set were good, achieving an average precision (AP) of 0.93. However, a simpler approach using nearest neighbor achieved a better AP of 0.99.

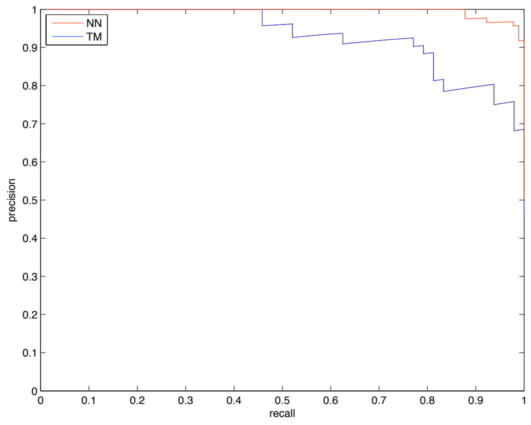


Figure 2: Easy data set: (a) example image (b) PR curves for nearest neighbor and template matching

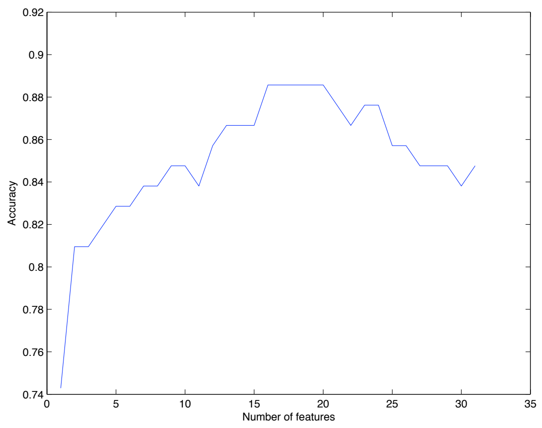
4.1 Feature analysis

To better understand how our features represent our data, we use the forward search feature selection algorithm to determine the most important features (see figure 3). The top 10 features chosen by forward search are: 4, 7, 1, 2, 11, 6, 10, 3, 30, 13.

By visual check, feature 4 corresponds to the scale of the model most closely matching that of the cars in the target images. Therefore, it makes sense that it is the most discriminative single feature since images which match the properly scaled model template well are most likely car images.

In this example, features 1-15 come from pose 1 and features 16-30 come from pose 2. It is somewhat surprising that the first pose 2 feature is selected at 9th in the list. This leads us to believe that adding additional poses does not help as much as we initially anticipated. One possible reason for this is that after running blurring and edge detection the most dominant remaining feature of both poses is the wheels which happen to look roughly the same since the two poses are both side views.

It is also interesting to note that the optimal weighting found by logistic regression corresponds to our intuitive notion of applying a differential function on the feature vectors.



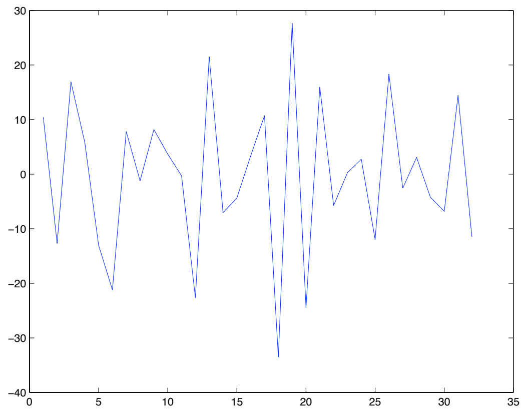


Figure 3: (a) Number of features vs. accuracy (b) optimal weight vector

4.2 Comparison

We implemented the nearest neighbor algorithm for classification by computing the sum of squares distance (SSD) between the test image and all training images. Then, each of the 10 closest images contributes one vote as to the class of the target image ("car" or "not car"). Normalizing these votes gives us a score between 0 and 1, which we can use to compute the precision and recall at different thresholds.

Nearest neighbor works well on this set because the input images are homogeneous in terms of object position color, position and scale. In this scenario, a 3D model that can represent all of the different poses and scales of a model provides little additional value. Furthermore, our template matching routines throw away significant information when doing edge detection and blurring that is clearly used effectively by nearest neighbor. Finally, our car model has a slightly different structure than the older cars present in the images, giving it a slight disadvantage.

5 Difficult data set: Many poses, many scales

The next data set that we worked on was that from the VOC 2007 Challenge [3]. This data set has cars at all scales and poses as well as occluded cars and non-traditional cars, such as the Hummer. They have also pre-segmented the data into a training set and a recommended validation set.

This dataset has labeled car examples split into test, validation and training sets (50/25/25). Unlike the previous data set, this set is skewed 80/20 toward negative examples which means that accuracy is not a good measure of performance. Instead we use average precision (AP) which is the same measure used in the challenge.

The best result from the challenge is AP of 0.78 while nearest neighbor gets AP of 0.20. Our template matching approach scores 0.31.

5.1 Optimizing for average precision

Logistic regression minimizes the error between the hypothesis and the training set. However, for the VOC Challenge, the comparison metric is average precision. AP is computed by iterating over recall in 0.1 intervals and averaging the precision values.

We used cross validation with a forward search algorithm and optimized for AP instead of generalization error. The best AP achieved by forward search on the validation set was 0.34. However, when we used the feature set on the test set it hurt AP slightly. This is not too surprising as the forward search algorithm simplify observed variations in the feature set that randomly improved performance on the validation set. These improvements did not generalize well to the test set.

A related technique would to be to use cross validation with a regularization parameter but we could not try this.

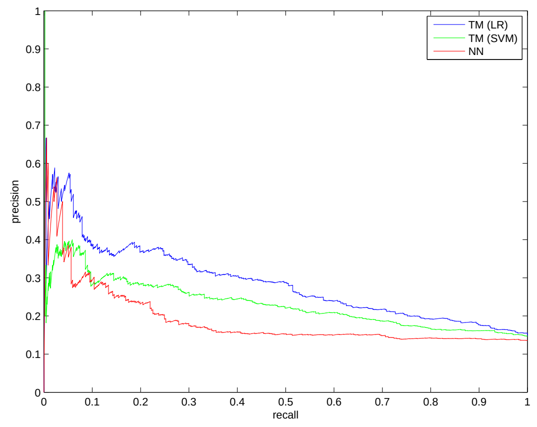


Figure 4: Difficult data set: (a) example image (b) PR curves for template matching with logistic regression (AP = 0.31), template matching with SVM (AP = 0.29) and nearest neighbor (AP = 0.20)

5.2 SVM

We also ran a SVM classifier over our feature set, to compare the results we got by logistic regression.

We used the VOC dataset, and SVM Light [5], with linear, polynominal and radial kernel. The parameters for the SVM classifiers were hand tuned for good results.

With a radial kernel, we got an AP of 0.29, which was comparable, but lower, than what we got from logistic regression. The linear and polynomial kernels, however, were not able to classifly any of the positive test images correctly. This is because the optimal function computed by SVM in these cases was such that $h(x^{(i)}) < 0$ for all i. This is optimal in terms of accuracy but not optimal for the metrics we care about: precision and recall.

6 Alternate approach: K-means clustering

One of the issues with template matching approach is the set of templates to provide to do the match. Here, we used TA Graz-02 database which also provides pixel masks for labelled cars. We ran k-means on edge images of these extracted cars to divide them into a small no. of clusters. Our hope was that this would divide the set of cars into a representative set of poses good for template matching.

We ran k-means to get a set of 28 representative images. Then, template matching was run on a set of cars and non-cars images from the same database. This database is complex with cars in many orientations and sizes. With this approach, we got an AP of 0.74.

For comparison, we also ran nearest neighbor (AP = 0.69) and template matching with 3D model images (AP = 0.73).

There were several difficulties we faced in k-means approach, which require further work. One was a good way to find distance between two images, especially as their aspect ratios differ. We used sum of squares distance on the images resized to fit in a 400x400 square, maintaing their original aspect ratio with empty parts padded with black pixels.

7 Conclusions

The problems we encountered can be categorized into two areas: using the knowledge of the 3D model to extract features from target images, and training the parameters of the system to optimize for precision and recall.

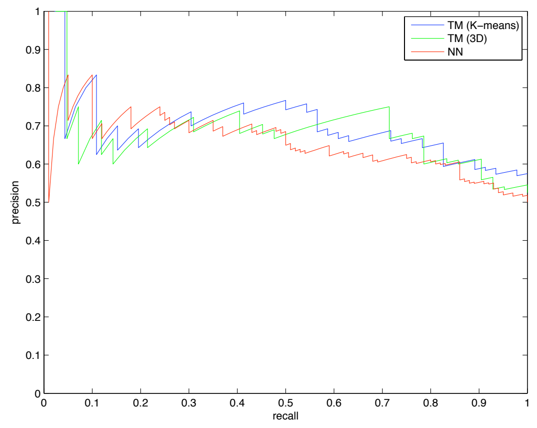


Figure 5: PR curves for nearest neighbor (AP = 0.69), template matching with K-means (AP = 0.74) and template matching with 3D model images (AP = 0.73)

We used our 3D models in a very simple way – by generating 2D projections of them. We then matched the projections to target images. This raised numerous questions, such as, how do you compare scores from the matching of different templates? How effective is template matching edge images at recognizing the target?

We showed that using the scores directly as features in logistic regresison works reasonably well. We also showed that even for difficult data sets template matching with a small number of poses can be effective in recognizing cars. It remains to be seen if simply using more poses will improve the accuracy of the system. Adding more poses will cause the feature vectors to grow considerably. Which in turn may also require more thought as to what is the optimal way to compare scores for template matching of different poses and scales – a question we touched on only briefly. Finally, adding more poses will push the limits of what is computationally reasonable – a question we did not consider as part of this project.

Optimizing for precision and recall, as opposed to accuracy, has been studied before in machine learning. Some examples include, SVMPerf [6] and BHRM. We did not try these as part of this project, but using such techniques would most likely improve our scores on precision and recall significantly.

8 Acknowledgements

Quoc Le provided much guidance on this project, including the idea of using 3D models plus template matching to do object recognition. Quoc also recommended comparing our results to nearest neighbor to better understand the performance of our approach.

References

[1] A. Torralba, R. Fergus, W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. In press, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[2] Quoc Le, Morgan Quigley and Andrew Y. Ng. Visual Servoing by Template matching. (Unpublished)

[3] The PASCAL Visual Object Classes Challenge 2007

[4] `http://l2r.cs.uiuc.edu/~cogcomp/Data/Car`

[5] `http://svmlight.joachims.org`

[6] `http://svmlight.joachims.org/svm_perf.html`

[7] `http://www.diecastmodelswholesale.com`

[8] `https://www.nextengine.com`