CS229 Project: Object(Keyboard) Identification in Images

By: Mark Mao

Abstract

In this project, an object detection algorithm is implemented. The algorithm is capable of locating keyboards in images. Main component of the algorithm is based on GMM models which can be applied to objects other than keyboards as well, while parts of the algorithm is heuristics based on the characteristics of the keyboard object. The algorithm achieved 80% detection rate for this task.

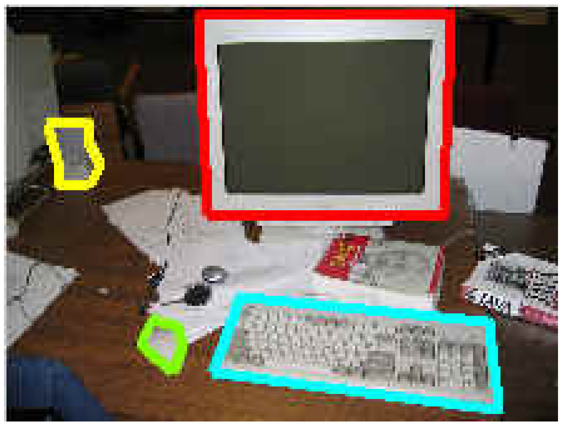
1. Introduction

Identifying objects in images is a trivial task for human brains, while at the same time, an extremely challenging task for computers. In this project, a specific object – keyboard is selected to limit the scope and the difficulty of the task, the goal of the project is to implement an algorithm that can locate the position of keyboard in images.

Many algorithms have been developed for object identification and image segmentation. Some algorithms only use local features while other algorithms try to combine local and non-local features.

GMM has been shown to be a robust technique for classification in high dimensional feature space. The algorithm in this project starts with local block based features and build GMM models for these features, it then tries to incorporate some non-block base features.

The training and testing images are real photos of offices, desks and computers from the MIT-CSAIL image library, an example image is shown below. The images are manually annotated, 33 images are used for training and another 30 images are used for testing.



2. Algorithm Summary

The algorithm consists of six main steps:

1. Feature extraction: extracts DCT features on each block of the training and testing images.
2. GMM training: creates 2 GMM models from training images, one for the keyboard, one for the background using feature vectors obtained in step 1.
3. GMM classification: In testing images, calculates the probabilities of each pixel block belonging to the keyboard class based on the GMM model.
4. Edge detection: detects the edges in the images.
5. Combining: combines the probability score of the GMM model in step 3 and the edge features in step 4 to create a final score for each pixel block.
6. Locate keyboard: searches through images and locate the keyboard object using the score obtained from step 5.

3. Feature Extraction

All training and testing images are first normalized into 640×480 gray scale images. DCT is then calculated for all 32×32 blocks in images, shifting by 8 pixels between each block. Let the DCT coefficients for each blocks be $D_{ij}, i = 1..32, j = 1..32$. A 10 dimension feature vector $F = (f_0, f_1, ...f_9)$ is calculated as follow:

$$f_0 = \sum_{i=1}^{4}\sum_{j=1}^{4} D_{ij}/16 \quad (1)$$

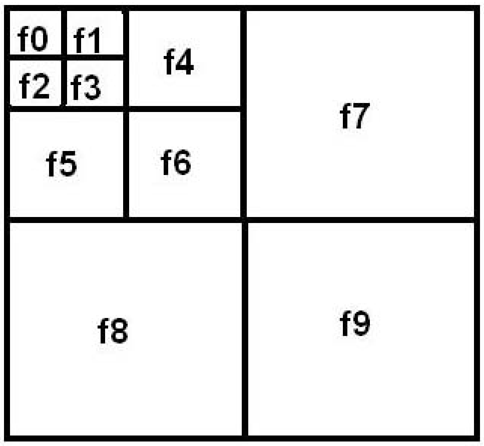
$$f_1 = \sum_{i=1}^{4}\sum_{j=5}^{8} D_{ij}/16 \quad (2)$$

$$\ldots \quad (3)$$

$$f_6 = \sum_{i=9}^{16}\sum_{j=9}^{16} D_{ij}/64 \quad (4)$$

$$\ldots \quad (5)$$

$$f_9 = \sum_{i=17}^{32}\sum_{j=17}^{32} D_{ij}/256 \quad (6)$$



4. GMM

Once feature vectors are extracted, two models are trained, one for the keyboard, one for the background. Gaussian Mixture Model is used to represent the models in the 10 dimensional feature space. 32 Gaussians are used for each model. And to simplify the model, diagonal covariance matrix is used for each Gaussian.

4.1. Training with EM

For each class, the Gaussian Mixture Model is trained with the standard EM algorithm. Parameters of the Gaussians are calculated with the following equation in each iteration until convergence.

$$\begin{aligned} \omega_j^{(i)} &= p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) && (7) \\ \phi_j &= \frac{1}{m} \sum_{i=1}^{m} \omega_j^{(i)}, && (8) \\ \mu_j &= \frac{\sum_{i=1}^{m} \omega_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} \omega_j^{(i)}}, && (9) \\ \Sigma_j &= \frac{\sum_{i=1}^{m} \omega_j^{(i)} I[(x^{(i)} - \mu_j) \cdot (x^{(i)} - \mu_j)]}{\sum_{i=1}^{m} \omega_j^{(i)}} && (10) \end{aligned}$$

4.2. Training Simplification with VQ

A simplified version of Gaussian Mixture Model with hard assignment of data points of VQ instead of the soft decision with EM is also implemented to save computation time. The resulting models showed no significant changes in terms of performance. In the simplified model, the Gaussian parameters are updated with the following equation in each iteration until convergence.

$$\begin{aligned} \phi_j &= \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}, && (11) \\ \mu_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, && (12) \\ \Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} I[(x^{(i)} - \mu_j) \cdot (x^{(i)} - \mu_j)]}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}} && (13) \end{aligned}$$

4.3. Classification with GMM

Once the GMM is created with training images, it is used on testing images to classify each block as keyboard or background. The probability is calculated as follow:

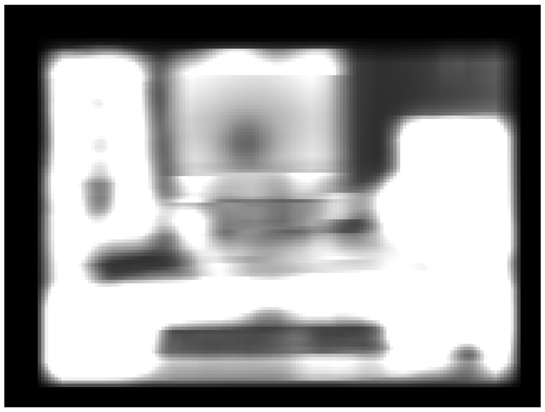
$$p(x|k) = \frac{1}{32} \sum_{j=1}^{32} \frac{\phi_j^{(k)}}{(2\pi)^{10/2} |\Sigma_j^{(k)}|^{1/2}} e^{(-\frac{1}{2}(x-\mu_j^{(k)})^\top \Sigma_j^{(k)-1} (x-\mu_j^{(k)}))} \quad (14)$$

$$p(x|b) = \frac{1}{32} \sum_{j=1}^{32} \frac{\phi_j^{(b)}}{(2\pi)^{10/2} |\Sigma_j^{(b)}|^{1/2}} e^{(-\frac{1}{2}(x-\mu_j^{(b)})^\top \Sigma_j^{(b)-1} (x-\mu_j^{(b)}))} \quad (15)$$

where k is keyboard, b is background and p is the probability of pixel x is part of a keyboard

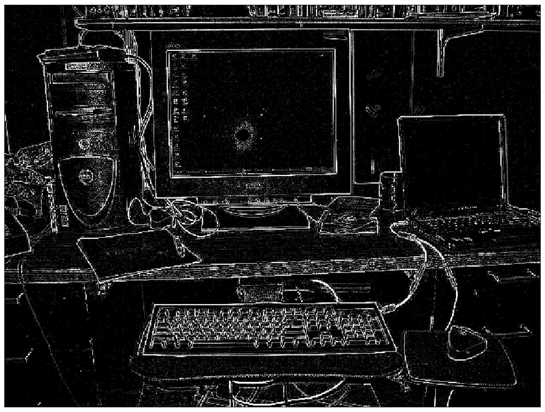
$$p = \frac{p(x|k)}{p(x|k) + p(x|b)} \quad (16)$$

The classification results of one test image is shown below. The first one is the original image. The second image represents likelihood of a pixel being part of a keyboard or background with intensity. The darker pixel blocks are more likely to be keyboard blocks. The classification shows reasonable results as pixels in the keyboard area are generally darker. But it also shows that the model can not distinguish well enough between keyboard and certain areas such as the upper right corner wall, and the notepad to the left.



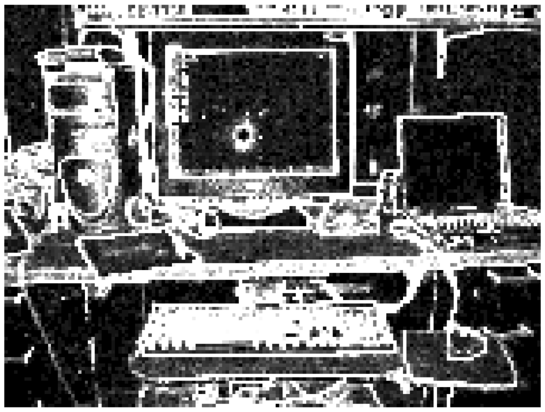
5. Edge Detection and Combined Score

For the specific object we want to detect in this project – keyboards, it is noticed that pixel blocks on the keyboards all have a very high frequency component because of the individual keys. This becomes obvious after running edge detection on the test images, as shown below. The keyboard area has the highest concentration of edges.



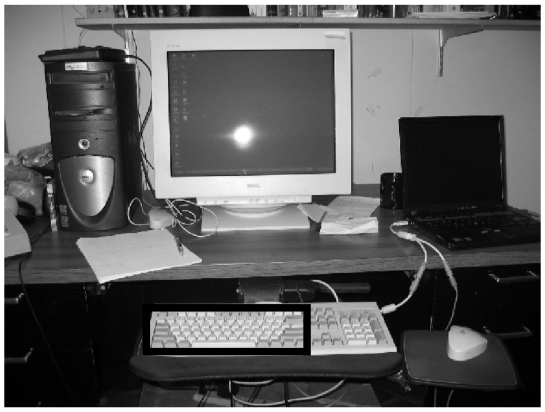
Even though the DCT features being used captures both low and high frequency energy, this specific high frequency component is not well captured because of the low resolution in our features for high frequency components.

To compensate this, a new score is computed for this high frequency component for each block on the edge detected images. Then this new score is combined with the probability scores obtained from the GMM model for all blocks. A new image showing this combined score is shown below, now the lighter area represents pixel more likely to be keyboards. As we can see, now the keyboard blocks standout well compare to the background.



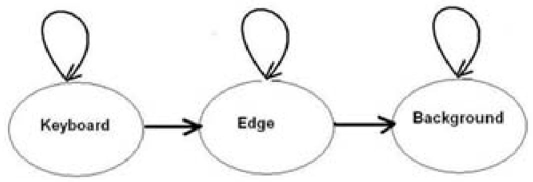
6. Locating Keyboard

A simple algorithm is implemented to search through all pixels in the image, and find a rectangular box such that pixels inside the rectangular box have the maximal average combined score, as shown in the image below.



6.1. A Simple Markov Chain

Anther keyboard locating algorithm that follows a simple 3-state Markov chain is also tried. In this algorithm, for each pixel block, start with a small rectangular box around it, assuming the pixel inside the box are keyboard pixels, then grow the rectangular box larger and larger, and accumulate the probabilities that the pixels on the edge of the box belong to keyboard. At some point, the box will grow pass the edge of the keyboard, then start calculating the probabilities that the pixels on the edge of the box belong to the background. Unfortunately, this algorithm did not perform as well as the simple algorithm above. The reason is that this algorithm is more sensitive to the shape and and the rotation of the keyboard. But if more tuning time is available, I do feel that this algorithm has more potentials.



7. Experiment Results and Summary

The training is done on 33 gray scale images.(640x480) Testing is done on a different set of 30 images. The algorithm correctly identified the location of the keyboard in 24 images, achieving accuracy rate of 80% (or 20% error rate). All testing images(6 errors, 24 correct) are attached in the appendix.

8. Conclusion, Discussion and Future Work

In this project, an algorithm that locates the keyboard in images is implemented, the main component of the algorithm uses GMM model trained based on DCT features. This part of the algorithm can be applied to any objects. Another component of the algorithm uses heuristics that tailored to the characteristics of the keyboard, which is less than universal. Overall, the algorithm achieved good results, with 80% recognition accuracy on the 30 test images.

Through this project, I learned that with enough training data, a decent GMM model can be built for local features. But local features at single resolution level can only go so far in a complicated task such as object identification. Things I would like to try in the future:

- Explore features in hierarchical structure, in the context of keyboards, it means:.

- At lower level, keyboard does have internal structures, if individual keys can be modeled, and also if the structure of groups of keys, e.g. regular keys on the left and the center, arrow and control keys on the center right, and numeric keys to the far right, can be modeled. It will greatly help identifying the whole keyboard.

- At higher level, keyboard almost always show up below the monitor, combined identification of monitor and keyboard should also improve the robustness of the keyboard locating algorithm.

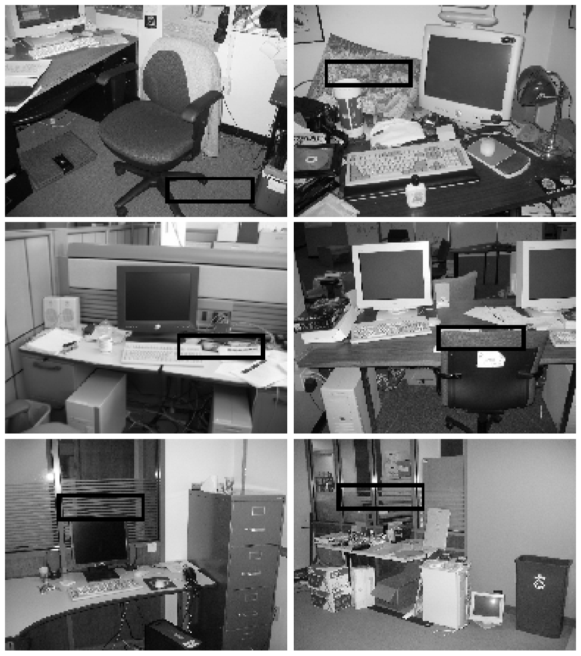
References

[1] Andrew Ng, ”CS 229 Machine Learning Course Notes,” Fall, 2006

Appendix

Images that the algorithm failed

Total 6:



Images that the algorithm succeeded

Total 24: