Vision Based Object Recognition and Localization

Yogesh Girdhar and Daniel Pomerantz

December 16, 2007

Abstract

The goal of this project is to be able to match unaligned images corresponding to different viewpoints of the same object or location. An application of this matching localizing a robot based on what it sees and compare it with previously seen images of the world. We do this by improving upon standard PCA matching algorithms. Instead of directly comparing images, we first extract sub-windows using an interest operator, convert them to polar coordinates, and then compare the amplitude spectrum of the Fourier transforms. This allows us to achieve rotation invariance. Sub-windows are selected using the Harris corner detection algorithm.
This work is based of work done by Deeptiman Jugessur and Gregory Dudek in [3], [2], and [4].

1 Introduction

Being able to locate oneself based on vision is one of the fundamental tasks that humans are able to do. However, it is very difficult to design a robotic system that can localize itself using vision. Many algorithms use a training set of images that are labeled by a human and seek to match query images with one of the previously identified images. However, these algorithms are very sensitive to differences between the query and training images. While humans are quite good at detecting these differences, vision algorithms are very sensitive to rotation, scaling, shift, occlusion, and background clutter. We seek to design a vision system that is more robust to rotation, shift, and occlusion.

2 Principal Component Analysis

The goal of an image recognition system is to be able to match a given image to a training image of the same object successfully. One approach to doing this is to consider the different intensity values at every single pixels and run a nearest neighbor algorithm on the query image to determine which training image it most closely matches. For an N pixel * N pixel image, with a training set of size T, this requires a run time of $O(N^2T)$, which is computationally intractable.

In PCA, we project the N * N images onto a dimension of size much smaller than the original image. This subspace ideally would strike a balance between retaining as much of the original information as possible and reducing the dimension. That is, after projecting onto this subspace, we would like to be able to reproduce the original image as closely as possible. PCA does precisely this by computing the eigenvectors of the covariance matrix[1] of our training images. The eigenvectors corresponding to the largest eigenvalues are the principal components of the trained images. Normally, we choose at most $|T| << N^2$ eigenvectors, where $|T|$ is the number of images in the training set. When presented with a query image, we can then run nearest neighbor approaches on the projected space, which is now feasible.

This approach works fairly well, however it is very susceptible to small changes in the query image such as partial object occlusion, image rotation, facial expressions, translation, scaling, and different angles of lighting. Our algorithm addresses these issues of partial occlusion, translation and rotation.

3 Occlusion and Shift Invariance using Local Matching

Rather than run PCA on the entire image, we would like to break both our training and test images into several sub-windows. Now we can apply PCA and run our nearest neighbor algorithms on each of the sub-windows. For each sub-window of the query image, we can find the closest matching training sub-windows. We then can combine all of the sub-windows using a voting scheme.

3.1 Choosing Sub-windows

We need a suitable technique for choosing what sub-windows to look at. We could naively divide our original images along a grid of suitable size, but this approach is computationally intractable as we would end up increasing the dimension too substantially. We also could divide our images along a grid and simply deterministically attempt to space out the windows, but this would be very susceptible to translations as we would essentially be ignoring certain parts of the image in this event. So if the query image was the same as the training image, except translated a few pixels, nothing would match.

What we will do instead is run an interest operator on the entire image. The interest operator will detect points of interest and we will base our windows around these. Ideally the interest operator should be able to handle changes such as rotations,

[1]This is computed by first taking all of our N*N image, converting it to a row vector via row stacking and then putting the row vectors into a vector. We then subtract the mean intensity of each point to get a matrix X. The covariance matrix is then XX^T

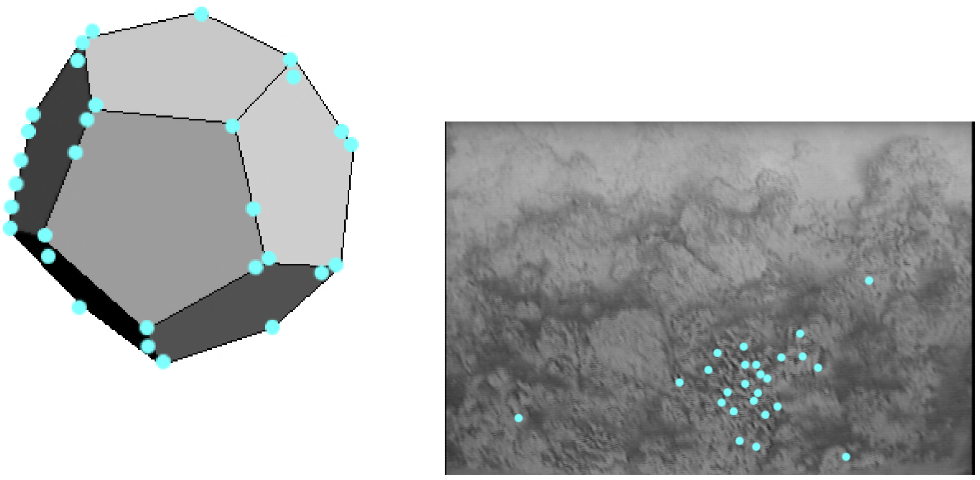

Figure 1: Interest points chosen by Harris interest operator.

translations, and scaling well as then it would be able to detect the same point in both the training phase and the query phase. We also would like to be able to space our interest points out as much as possible. This is so that our algorithm can still work when there is partial occlusion of the image. If we choose all of our windows too close to each other, then a partial occlusion may block all or most of the windows that would have been generated.

3.2 Harris Corner Detection

We chose the Harris corner detector as an interest operator. Harris is designed for detecting corners in an image but suits our purposes very well for several reasons: First of all, it is translation and (planar) rotation invariant as the corner detection is insensitive to these. Secondly, the algorithm can be run quickly and is relatively easy to implement. Additionally, Harris has performed well compared with other operators such as symmetry operators in previous studies [5].It should be remembered that the interest operator could be replaced by other operators such as Sift in the algorithm. The only heuristic is that the operator should be as insensitive to noise (i.e. rotations, translations, etc) as possible.

To compute the Harris measurement values first compute a matrix G at every point defined as follows. Note that $I(x,y)$ refers to the image intensity (in grey-scale) at point (x,y)

$$G := \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix}$$

We then produce a matrix H with is G convolved with a Gaussian in order to smooth G[2]. The idea behind the construction of H is that at a corner there is a sharp change in intensity in both the x direction and the y direction. It turns out, that a good measure of the size of the entries of a matrix is to compute the size of the eigenvalues of the matrix. Rather than explicitly compute the eigenvalues at every step, we will use the fact that for any square matrix, the sum of its eigenvalues is equal to the sum of its trace and the product of its eigenvalues is equal to the determinant of the matrix. Using this fact, we have a good heuristic for measuring the size of the eigenvalues, specifically:

$$h(M) = det(M) - \alpha tr(M)$$

where α is a constant normally chosen less than .1. This defines a function that lets us measure the "cornerness" of a point.

See figure 1 for an example illustrating Harris.

3.3 Selecting the windows

After running the Harris operator on our image, we need to select the proper windows. In order to avoid redundancy in the operator and to better space our windows, we select the top N pixels as given by the Harris operator, but with the requirement that they not intersect. After selecting these points, we then construct a circular window around each of these points by cropping the image around these points in a circular fashion with fixed radius. Note that we also require that our top N points to be towards the interior of the image so that we do not run into image boundary issues. After applying this procedure, we have extracted N sub-windows from the original image.

4 Rotation Invariance

We could immediately run PCA on our sub-windows and could hope for an improvement due to the local nature of our new data set. However, we also wish to improve on the sensitivity to planar rotations. Recall that a rotation of an image in Cartesian coordinates corresponds to a shift in the image in Polar coordinates. See figure 2. We also note that the amplitude spectrum of the Fourier transform is shift invariant. That is, given two images that differ only by translation, the amplitude spectrum of the Fourier transform is identical. Using these two ideas, we can achieve rotation invariance by first converting our images to polar coordinates, and then comparing them in their amplitude spectrum in the frequency domain via applying Fourier transform on each one of them.

[2]We used Karl Rohr's recommended method, which uses $\alpha = 0$

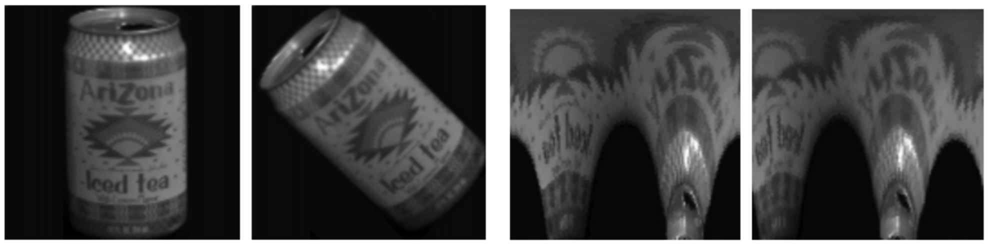

Figure 2: A sample image and its rotated version and their corresponding images in polar coordinates. We see that rotation corresponds to shift in polar coordinates.[Images taken from [3]]

5 Matching Images

We now have a set of sub-windows, each of which is "owned" by an original image. For the time being, we treat these sub-windows entirely independently. That is, we ignore the fact that some of the sub-windows came from the same image and treat them as entirely separate images. We then run PCA on all of these images and project to the eigenspace generated by the training data. For every sub-window of a query image, we are then able to compute the distance in the eigenspace to every other sub-window.

For each query sub-window, we now have the distance to the closest training sub-windows. We now take the results of PCA for each of the sub-windows and combine this into a voting scheme. Each original image gets a vote if it contains a sub-window that matches one of the query sub-windows. We weight the votes according to an exponential. That is, if x is the query sub-window and y is the closest training sub-window, we give a vote of size w_i where w_i is

$$w_i = e^{-d(x,y)^2}$$

Additionally, rather than taking simply the top match, we experiment with taking the top N closest sub-windows and allowing a contributing vote. Since the votes are already weighted according to their distance, this could allow better accuracy since this allows for the case where the closest image has sub-windows that frequently appear as second or third closest matches but rarely as the closest. This captures the idea that it may be more important to count the number of times that each sub-window is a "reasonable" candidate rather than being the best candidate.

6 Overall Algorithm

Combining the ideas developed in sections 2, 3.2, and 4 above, we now sketch our full training algorithm.

- For each training image:

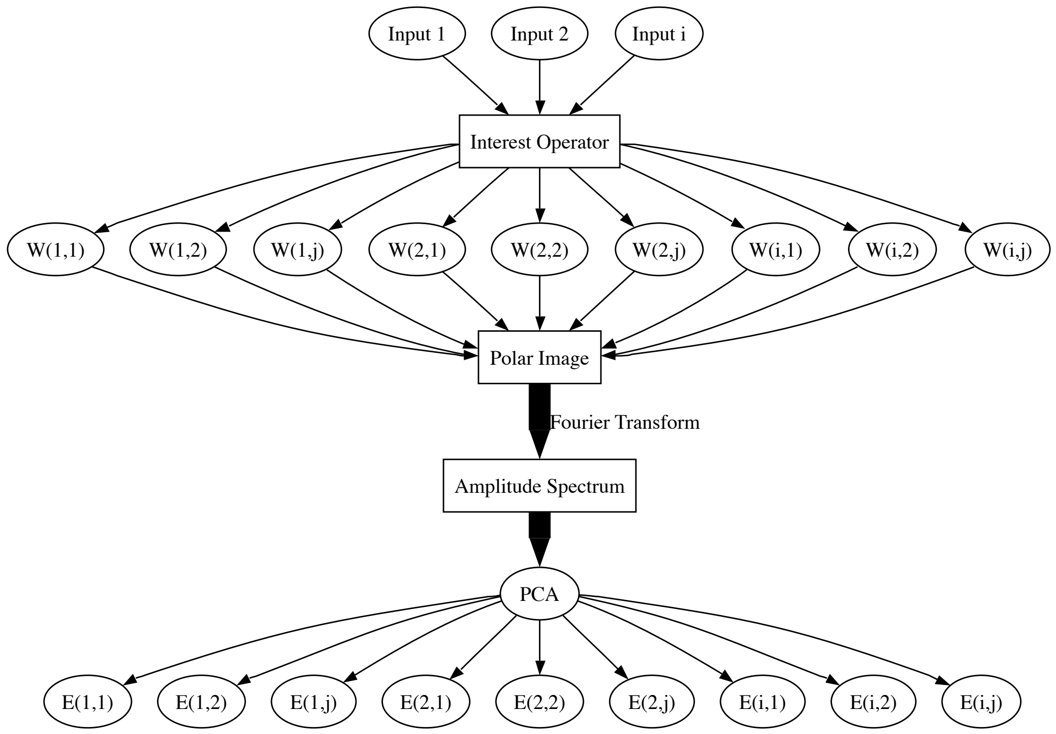

Figure 3: Pipeline for computing a low dimension representation for our input training images.

  - Use an interest operator (Harris) to compute N_w sub-windows of a given size.
  - Convert the sub-windows to polar coordinates.
  - Compute their amplitude spectrum in the frequency domain by taking their Fourier transforms

- Use PCA to compute a low dimensional eigenspace using all the sub-windows of all the training images.
- Project each window onto this space to get a low dimensional vector representative of the sub-window. This forms our training database.

The above is visually summarized in figure 3

We then in query phase, perform the same pipeline process and match the query image to a training image using the voting scheme described in 5.

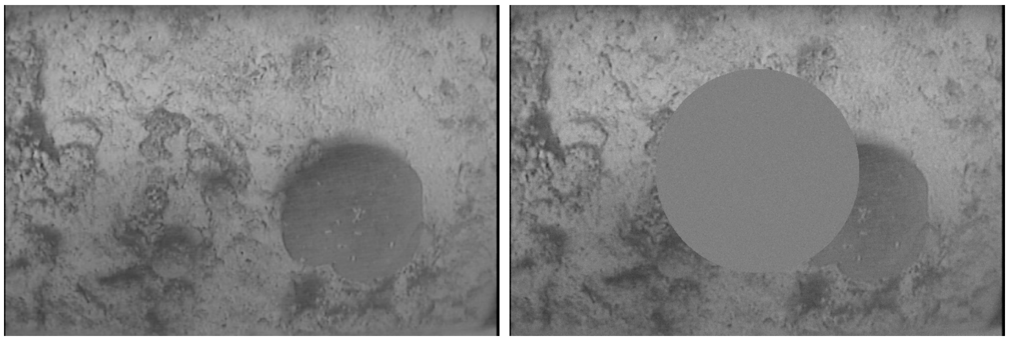

Figure 4: Example of an original query image and after adding 20% occlusion.

7 Our Experiment

We tested the previous ideas on two data sets: an object data set, which was compiled by Jugessur in [4] and a data set consisting of footage taken by the Aqua robot [1]. We experimented with different window sizes as well as occlusion in the test images. We did this by adding a uniform circle to the middle of the test image (see figure 4). We compared our algorithm to the baseline matching algorithm which does not use sub-windows or Fourier transforms.

In order to do these comparisons, we did leave one out cross-validation. In the case of the object set, we have three pictures of each object, and if the algorithm outputs one of the two remaining pictures, we count it as correct. For the Aqua robot training set, we extracted several frames from a video clip. They are numbered sequentially, and we consider the distance between the test image number and the number of the image it matches with. With this methodology, the best possible result is when it is off by only one, but it has two ways to do this. We studied the median distance rather than the mean distance because the mean distance is too susceptible to cases where the algorithm makes large mistakes.

8 Results and Observations

The comparison of the modified algorithm with the regular PCA algorithm is summarized in figure 5. On the object data set, the improved algorithm performs slightly worse than the regular PCA algorithm. However, after adding occlusions to the data set, the regular, global PCA algorithm witnesses a serious decrease in performance. However, the modified algorithm does not suffer nearly as much. In the case of the Aqua data set, our algorithm immediately performs better than the regular one as the algorithm outputs a match that is much closer in reality to the frame that is being queried. The median distance of the match from the query frame for the cases with occlusion 10% or less was two. This is good since the lowest possible matching distance is one.

The differences in the two results is mostly due to the fact that the object data set

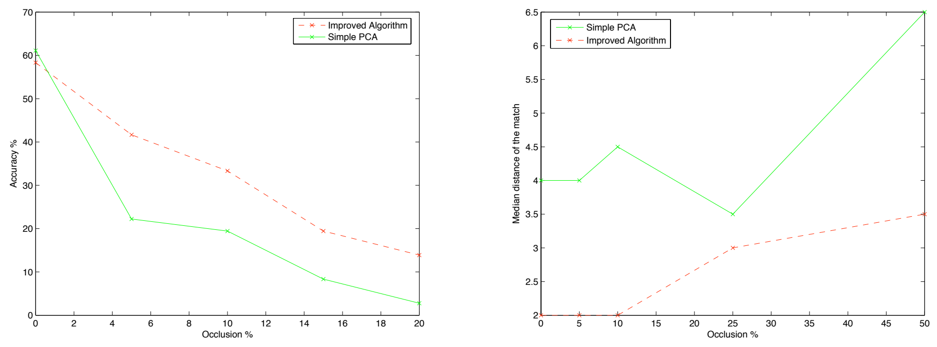

Figure 5: Two graphs comparing the baseline algorithm (green) to our algorithm (red dashes) when occlusion is added. Adding sub-window extraction greatly improves the resiliency to occlusion. In the first graph, we compare percent accuracy in the object data set. In the second we compare median distance in the Aqua data set.

is a much more fixed set up. The pictures are all taken centered around the object. However, for the Aqua robot set, the images are already translated and rotated, even before adding a manual occlusion. This means that for the object set, achieving shift invariance is irrelevant since there is no shift. However, for the Aqua set, even before adding occlusion manually, the global matching does not work due to the rotations and shifting that occur from frame to frame. This indicates that the improved algorithm is significantly better in practice, since it is less sensitive to translations and rotations.

It is important to note that while our algorithm managed to achieve similar success to the baseline PCA algorithm, it was doing so on a much smaller amount of data. For the experiments on the object set, we used the interest operator to select 20 30 pixel x 30 pixel windows. This means for each image, we are storing $20 * 30 * 30 = 18,000$ pixels of information. The object set images were $640 * 480 = 307,200$ pixels of information and the Aqua set images were $720 * 480 = 345,600$ pixels of information. This suggests we could increase the window size or number of eigenvectors used and still have significantly better running speed than ordinary PCA matching

We compared using a "soft" voting scheme vs a "hard" voting scheme on the object data set. This refers to whether or not to count just the closest matching sub-window or the top N matching sub-windows. We very quickly concluded that counting votes from the top N matching sub-windows was the way to go. These results are summarized in table 1. The results show that it doesn't matter whether you include every single sub-windows vote or simply a few, but that it is very important to count more than just one. This is due to the fact that the sub-windows are small enough that the closest matching sub-window will often come from the incorrect image due to noise. However, the image that is actually closest in the training set will almost always have a sub-window that is in the top five. However, we don't need to make N the total number of sub-windows as at a certain point the distance between the training sub-window and the

Number of Votes Counted	Accuracy (%)
1	41.67
5	52.78
10	50
100	58.333

Table 1: Matching accuracy on the object data set as we increase the number of votes counted.

query sub-window is so small that the vote goes towards zero anyway. Based on these preliminary results, we only used 100 votes counted for the rest of the experiments.

Another issue relates to the certainty of matching. In the majority of cases, the strength of the matching (the total amount of weighted votes acquired by the top image) was more than twice as high when the matching was correct than when it was not. This suggests that we could design an algorithm to output "not sure" in certain cases. This would be useful as we could try other techniques to match images in this case.

9 Summary

We presented an algorithm to match images in a way that is more robust to shift, occlusion, and rotation. Compared with simple PCA matching, our algorithm performs competitively in a controlled environment, but significantly better as the occlusion and shifts in the scene are increased. Future improvements could include using a better interest operator like SIFT or another scale-invariant operator and experimenting more with parameters such as the number of windows per image, the number of eigenvectors, and the size of the windows. Additionally, all of our Aqua data was extracted from one video sequence, and it would be useful to experiment on multiple sequences.

References

[1] Gregory Dudek, Philippe Giguere, Chris Prahacs, Shane Saunderson, Junaed Sattar, Luz-Abril Torres-Mendez, Michael Jenkin, Andrew German, Andrew Hogue, Arlene Ripsman, Jim Zacher, Evangelos Milios, Hui Liu, Pifu Zhang, Marti Buehler, and Christina Georgiades. Aqua: An amphibious autonomous robot. *Computer*, 40(1):46–53, 2007.

[2] Gregory Dudek and Deeptiman Jugessur. Robust place recognition using local appearance based methods. *IEEE International Conference on Robotics and Automation (ICRA)*, april 2000.

[3] Deeptiman Jugessur. Robust object recognition using local appearance based methods. Master's thesis, McGill, 2000.

[4] Deeptiman Jugessur and Gregory Dudek. Local appearance for robust object recognition. *IEEE Computer Vision and Pattern Recognition (CVPR)*, june 2000.

[5] R. Sim, S. Polifroni, and G. Dudek. Comparing attention operators for learning landmarks robert.