SANDIA REPORT

SAND98–0503 • UC–330
Unlimited Release
Printed February 1998

CONF-960672--3
CONF-970679--2

Sensing and Compressing 3-D Models

John Krumm

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND98-0503
Unlimited Release
Printed February 1998

Distribution
Category UC-330

Sensing and Compressing 3-D Models

John Krumm
Intelligent System Sensors and Controls Department
Sandia National Laboratories
P. O. Box 5800
Albuquerque, New Mexico 87185-1003

Abstract

The goal of this research project was to create a passive and robust computer vision system for producing 3-D computer models of arbitrary scenes. Although we were unsuccessful in achieving our overall goal, several components of this research have shown significant potential. Of particular interest is the application of parametric eigenspace methods for planar pose measurement of partially occluded objects in gray-level images. The techniques presented provide a simple, accurate, and robust solution to the planar pose measurement problem. In addition, the representational efficiency of eigenspace methods used with gray-level features were successfully extended to binary features, which are less sensitive to illumination changes. The results of this research are presented in two papers that were written during the course of this project. The papers are included in sections 2 and 3. The first section of this report summarizes our 3-D modeling efforts.

Contents

CONF-960672--3

CONF-970679--1

Figures

2. Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects

3. Object Detection With Vector Quantized Binary Features

1. Summary

The goal of this research was to create a passive means of producing computerized, 3-D models of arbitrary scenes. Current methods for solving this problem suffer from restrictive assumptions about scene formation and the physical world. For instance, conventional binocular stereo vision assumes that both cameras see the same set of 3-D scene points, which, due to occlusions, is hardly ever 100 percent true. False physical assumptions like these have lead to poor performance. Our approach was different in that we aimed to exploit the rich models of computer graphics in order to make a more robust computer vision system for building 3-D scene models.

Computer graphics has been very successful at accounting for the detailed physical phenomena involved in scene formation. The major difference between computer graphics and computer vision is that computer graphics solves the problem of making an image from a scene description, while computer vision tries to go in the opposite direction. Given the multitude of physical phenomena to account for, using the models of computer graphics for computer vision results in a complex inversion problem for which there is no closed form solution.

Our initial approach to the problem starts with a pair of stereo images as input. The algorithm performs an exhaustive random search for the 3-D scene that best accounts for the two input images. To evaluate each candidate scene, the algorithm uses computer graphics techniques to compute the scene's appearance from the point of view of the two cameras that provided the input images. In this approach, the 3-D scenes are represented by voxels (volume elements). The voxel representation facilitates a systematic search and a very general 3-D representation. This representation allows the complex physical models of computer graphics to be used for image construction and interpretation for passive 3-D model building.

Although our initial results obtained on a low-resolution 16x16x16 voxel space were very successful, we were unable to obtain comparable results at more realistic levels of resolution. The extremely complex surface reflectance properties of objects contained in representative real-world scenes proved to be a major hurdle that time and budget constraints prevented us from crossing. Rather than abandon our efforts at this point in our research, however, we chose to investigate a more traditional approach for 3-D model building using binocular stereo vision.

In stereo vision, a correspondence between image features acquired from two different camera views is used to establish 3-D model points. Although the approach is conceptually simple, the correspondence of image features can be problematic. In fact, the inadequacy of solutions to the correspondence problem has limited the use of binocular stereo vision in computer vision applications. The basic problems with feature correspondence in stereo imaging have to do with the fact that things can look significantly different from different points of view. It is possible for two stereo views to be sufficiently different that, even though an image feature is common to both views, they can not be matched correctly. Even worse, it is possible that a feature present in one view can be obscured in the other view, making it impossible to match. More robust solutions

to the correspondence problem are necessary if binocular stereo vision is to become a truly practical means for creating 3-D computer models of arbitrary scenes.

Our investigations in developing robust solutions to the correspondence problem involved application of "parametric eigenspace" methods. Recently, Murase and Nayer have presented the parametric eigenspace for object recognition and pose measurement based on training images. Although their system is easy to use, it has potential problems with background clutter and partial occlusions. We have extended their approach to use several small features on the object rather than a monolithic template. These "eigenfeatures" are matched using a median statistic, giving the system robustness in the face of background clutter and partial occlusions. We have demonstrated the applicability of the approach on the more general problem of planar pose measurement of partially occluded objects. The pose measurement accuracy has been demonstrated with a controlled test and the algorithm's robustness is demonstrated on cluttered images with objects of interest partially occluded.

In addition to the eigenspace methods developed for gray-level features, we have extended our methods to include binary features, which are less sensitive to illumination changes. Binary features, contained within square sub-templates, are automatically chosen on each training image. Using features rather than whole templates makes the algorithm more robust to background clutter and partial occlusions. Instead of representing the features with real-valued eigenvector principle components, we use binary vector quantization to avoid floating point computations. The object is detected in the image using a simple geometric hash table and Hough transform. On a test of 1000 images, the algorithm works on 99.3 percent. We present a theoretical analysis of the algorithm in terms of the receiver operating characteristic, which consists of the probabilities of detection and false alarm. We verify this analysis with the results of our 1000-image test, and we use the analysis as a principled way to select some of the algorithm's important operating parameters.

The details of our research involving parameteric eigenspace methods for planar pose measurement are presented in sections 2 and 3. This work constitutes a significant portion of our research and has proven extremely successful.

IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 1996, San Francisco

Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects

John Krumm
Intelligent Systems and Robotics Center
Sandia National Laboratories
MS 0949, P.O. Box 5800
Albuquerque, NM 87185
jckrumm@sandia.gov

Abstract

Planar pose measurement from images is an important problem for automated assembly and inspection. In addition to accuracy and robustness, ease of use is very important for real world applications. Recently, Murase and Nayar have presented the "parametric eigenspace" for object recognition and pose measurement based on training images. Although their system is easy to use, it has potential problems with background clutter and partial occlusions. We present an algorithm that is robust in these terms. It uses several small features on the object rather than a monolithic template. These "eigenfeatures" are matched using a median statistic, giving the system robustness in the face of background clutter and partial occlusions. We demonstrate our algorithm's pose measurement accuracy with a controlled test, and we demonstrate its detection robustness on cluttered images with the objects of interest partially occluded.

1. Why Work on Planar Pose Measurement?

Planar pose measurement is an important part of automated assembly and inspection. In this paper, we are envisioning a workcell of the type illustrated in Figure 1. An overhead camera is pointed down at objects resting on a flat surface. The task is to measure the pose, (x, y, θ), of a given object in the image.

Although there are several solutions available to the pose measurement problem, both commercially and academically, none of the solutions have yet to win widespread appeal. One of the main barriers to increased use of computer vision in automated manufacturing is that the vision systems are difficult to tune. Pose measurement is intended to function as part of an automated manufacturing line. But this advantage is lost when the vision system requires reprogramming from a skilled operator to account for changes in illumination, optics, and objects. Current commercially available solutions typically require a training phase in which an operator manually helps the vision system identify important features of the objects of interest. These features must be carefully chosen based on their consistency and their ability to indicate the pose of the object. This requirement leads to heuristic rules for the operator to follow such as "specular highlights

This work was performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

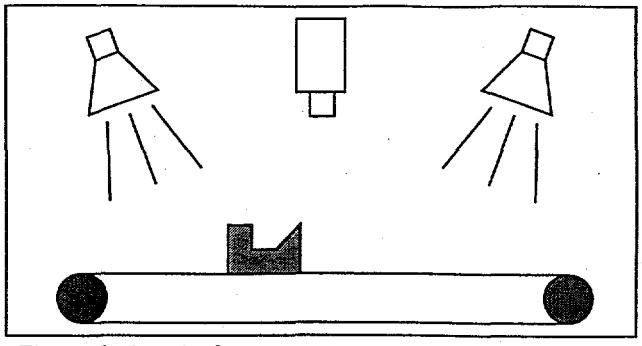


Figure 1: A typical pose measurement station with an overhead camera and lights pointed down at a conveyor belt carrying objects to be assembled or inspected.

are bad features because they shift based on the orientation of the object, lights, and camera," and "line segment features are not good for localizing the object along the line." Given the cost of computer vision experts, it often appears less expensive to find less flexible or more labor intensive solutions to the pose measurement problem.

In this paper, we present a new solution to the pose measurement problem. It is based on Murase and Nayar's "parametric eigenspace" idea[2], which uses principle component templates based on training images. We show how to apply this idea to multiple, automatically detected features on the object. We match features using a median distance measure, which gives the algorithm robustness. Using features instead of monolithic templates, our algorithm overcomes problems of segmentation, background clutter, and partial occlusions, while retaining the automatic programming advantage of the original system.

2. Parametric Eigenspace

A new technique for pose measurement, called "parametric eigenspace", has been developed by Murase and Nayar [2]. Their work is related to earlier eigenface research by Turk and Pentland [7]. The method is used to recognize objects and measure their orientation based on training images of the objects in different orientations. This solution is attractive because it requires no expert human assistance for picking features. A brief explanation of parametric eigenspace follows.

In its full form, as explained in [2], the parametric eigenspace method works on a presegmented image to identify an object and give its orientation around one axis under a few

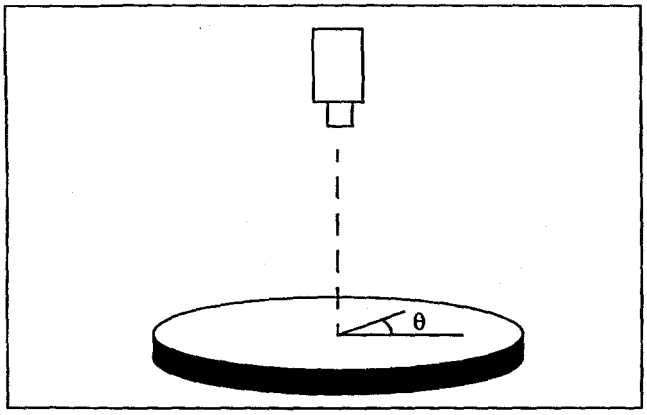


Figure 2: Training images are taken with an overhead camera looking down at the object which rests on a motorized rotation table.

different lighting conditions. The method is based on a series of training images of the object taken at different orientations. In our implementation, we use 90 or 180 training images of the object (four degrees or two degrees apart) taken from a camera directly overhead while the object is rotated on a motorized table below. This setup is illustrated in Figure 2.

For training, the object is first segmented from each training image into a rectangular image patch. Segmentation in the training phase is normally easy because the scene can be carefully controlled. We use a backlit table to make a binary mask of the object. The segmented patches must be the same size for each training image. This equal size requirement means the patches must include significant parts of the background if the object is elongated or has significant concavities, as illustrated in Figure 3.

Each segmented training patch is normalized in intensity by dividing each pixel by the sum of the squares of the pixels in the unnormalized patch. While Murase and Nayar also normalize for size, we do not since we assume the camera will always be the same distance from the object. Finally, the mean of the entire normalized set of training patches is computed and subtracted from each normalized patch. Each of these processed patches is scanned in raster order to form a column vector containing all the pixels in the patch. All the column vectors are placed side by side into a matrix X. The sample covariance matrix is formed as

$$Q = XX^T.$$

The eigenvectors of Q form an orthogonal basis set for the normalized, zero mean training patches. These eigenvectors are called eigenimages when they are scanned from column vectors back into images. The normalized, zero mean training patches can be expressed as a weighted linear sum of the eigenimages. Moreover, the patches can be accurately approximated by only the first few eigenimages. In equation form, this is

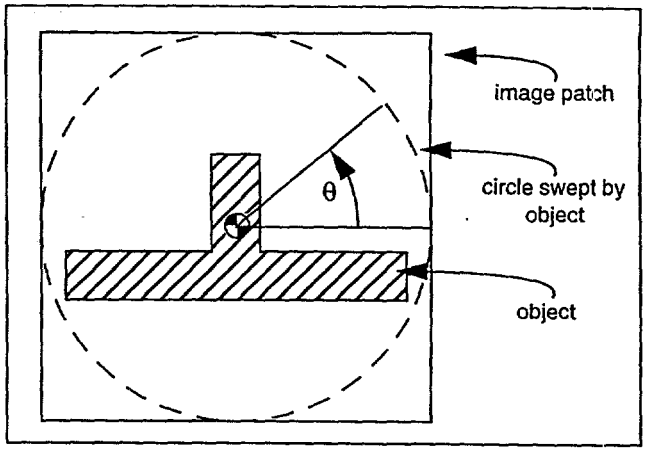


Figure 3: In previous approach, segmented image patch must be large enough to contain all trained orientations of object, which could include large parts of background.

$$\bar{p}_i \approx \sum_{j=0}^{l-1} c_{ij}\bar{e}_j \qquad (1)$$

where

$\bar{p}_i = i^{th}$ normalized, zero mean training patch

$\bar{e}_j = j^{th}$ eigenvector of Q

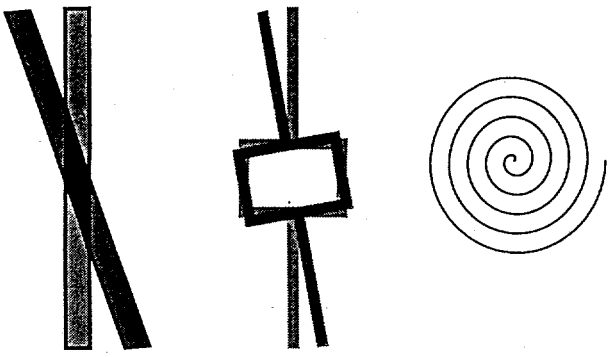
c_{ij} = weighting coefficient $= \bar{p}_i \cdot \bar{e}_j$

l = number of eigenimages used

The simple computation of the weighting coefficients comes from the fact that the eigenvectors are orthogonal. For our experiments, we used $l = 10$.

Equation (1) implies that each training patch $\bar{p}_i$ can be represented by l coefficients $(c_{i,0}, c_{i,1}, ..., c_{i,l-1})$. These coefficients are a point in an l-dimensional space, and each training patch projects to such a point. Since the training images represent an ordered progression of angles (θ in Figure 2), the coefficients plotted in l-dimensional space normally fall on a smooth curve. Each point represents a different training image and thus represents a different orientation of the object. This curves resides in "parametric eigenspace", because it can be parameterized by the angle θ.

Pose measurement in Murase and Nayar's formulation consists of first segmenting an input image by some means to find the object. This rectangular image patch is then normalized and the mean of the training patches is subtracted (same processing as training images). This processed patch is projected into the parametric eigenspace by taking dot products with the l previously computed eigenimages. To find the angle θ, the parametric curve is interpolated to find the closest point. More details can be found in [2], which also discusses a similar approach to object recognition and



small overlapping region | bounding box of overlap includes background | possibly no overlap

Figure 4: It is difficult to make good templates from the overlapping regions of these objects' silhouettes.

training under different illumination.

The advantage to the eigenspace approach is that the training is automatic. One disadvantage is that the object must first be segmented in the run-time image, which can be difficult in any image without a uniform background. Another disadvantage is that the basic parametric eigenspace method works on rectangular image patches that contain the whole object. Thus, for optimum matching, none of the object can be occluded, and the background of the object to be detected in an image cannot be different from the background in the training images. Rectangular image patches become a problem when the object does not have a generally rectangular shape, because other objects can intrude in the background. Our algorithm addresses all these problems.

Recently, Murase and Nayar have modified their approach to address the problems of segmentation and background clutter[3]. Their "image spotting" algorithm scans the image for the object, which solves the segmentation problem. (This assumes that the object's appearance is invariant with respect to translation in the image, which is approximately true when using a long focal length lens.) The algorithm uses training images that are reduced in size to the area of common overlap between the object's silhouettes in the training images at different angles, thus eliminating the background from the training images. Some objects do not have much overlap in their silhouettes, so the image spotting algorithm can sometimes split the training images into subsets that have sufficient overlap. Thus, the reduced size training templates solve the problems of background clutter for some objects, but not all. Some objects defy this splitting, as their silhouettes have virtually no overlap that could be contained in a bounding box without background as the object rotates in discrete increments. Three illustrations of such shapes are given in Figure 4. The problem of partial occlusions still remains, too.

3. Eigenfeatures for Pose Measurement

We have developed a new way of using parametric eigenspace that avoids the problem of segmentation, background clutter, and partial occlusions. We avoid the segmentation requirement by applying our method at every offset in the image, using the fast Fourier transform to speed up the projections into eigenspace. We solve the problem of background clutter and partial occlusions by using several small, rectangular image patches on each object rather than one large patch. In addition, using features means our algorithm does not use large, uniform regions of the object for matching, which often leads to false matches. Our method retains the advantages of the original algorithm in that it works entirely based on training images and requires no programming from a skilled operator. The remainder of this section explains our method in detail.

3.1 Gathering Training Features

We gather training images using an overhead camera pointed down at a backlit table mounted on a motorized rotator. We take images every two or four degrees using overhead lighting, giving a total of 180 or 90 training images per object. We take another set of images at the same set of angles with the overhead lights off and the backlit table on. The backlit images are thresholded to form masks for segmenting the training images. This backlit segmentation is used only for training and is not part of the on-line system.

Features consist of small, rectangular image patches. We use a feature size of 15 x 15 pixels along with a feature-finder developed by Shi and Tomasi[6]. Given an image, their algorithm produces a list of rectangular image patches of a prespecified size that is ranked based on the features' ability to be tracked through image sequences. We find their features to be good for pose measurement, too, because they are good at localizing position, *e.g.* points and corners. Shi and Tomasi's recipe for finding good features is to first compute partial derivatives at every pixel in the image. If the image is $I(x, y)$, then the partial derivatives are $I_x(x, y)$ and $I_y(x, y)$. For the numerical derivatives, we use Savitzky-Golay filters as described in *Numerical Recipes* [4] with the derivative filter size equal to the window width (15 x 15). At each pixel, a matrix is formed whose elements are sums of the products of the partial derivatives taken in the feature windows. The matrix is

$$Z = \begin{bmatrix} \sum I_x^2(x, y) & \sum I_x(x, y) I_y(x, y) \\ \sum I_x(x, y) I_y(x, y) & \sum I_y^2(x, y) \end{bmatrix},$$

where the sums are taken over the pixels in the feature window. The second eigenvalue of this matrix is used to rank the feature - the higher the better. The top 30 features of a wire connector at different angles are shown in Figure 5.

Shi and Tomasi's algorithm is especially convenient, since the only parameters it takes are the size and number of fea-

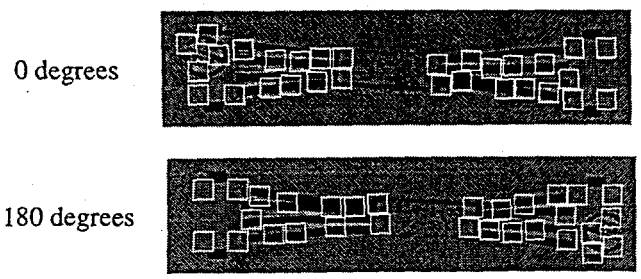


Figure 5: Features automatically selected on wire connector training images.

tures. We constrain the features to have no overlap, and to be centered somewhere within the mask computed from the backlit images.

Every training image has its own set of features - there is no tracking of features from image to image. Thus, features on certain parts of the object, such as specular highlights, could come and go as the object rotated. In a more general 3D orientation problem, rotating the object could cause parts of the object to come into and out of view, making feature correspondence impossible. Since we do not require feature continuity from angle to angle, this would not be a problem for our algorithm.

3.2 Training

The training for our method begins in the same way as described for the image patches in Section 2, except it uses small features instead of image patches. For n training images and m feature per image, we have mn total features. Since we do not track features through the training images, we must have some other way to account for the changing appearance of a given feature due to rotation. We do this by noting how the pixels in each feature window change as the object rotates slightly beneath them. For a given feature in training image i, we take out a feature in the same (x, y) location in training images $i-1$ and $i+1$ (with circular wrap-around of the indices on the first and last training images). This gives a total of $3mn$ features for each object. Each of these features is processed as described for the image patches in Section 2 (normalizing and subtracting aggregate mean).

The $3mn$ processed features are scanned in raster order into column vectors. We designate the pixels from the k th feature from image i to be $\bar{v}_{ik}$, where $i \in [0, 1, ..., n-1]$ indexes the n training images (and therefore the orientations θ_i), and $k \in [0, 1, ..., m-1]$ indexes the m features. The two features in the same location from images $i-1$ and $i+1$ are designated $\bar{v}^-_{ik}$ and $\bar{v}^+_{ik}$ respectively. If the dimensions of the features is w x w pixels, then each feature vector will be w^2 x 1.

The processed feature vectors are assembled into the columns of matrices:

$$X = [\bar{v}_{00} \mid \bar{v}_{01} \mid ... \mid \bar{v}_{ij} \mid ... \mid \bar{v}_{(n-1)(m-1)}]$$

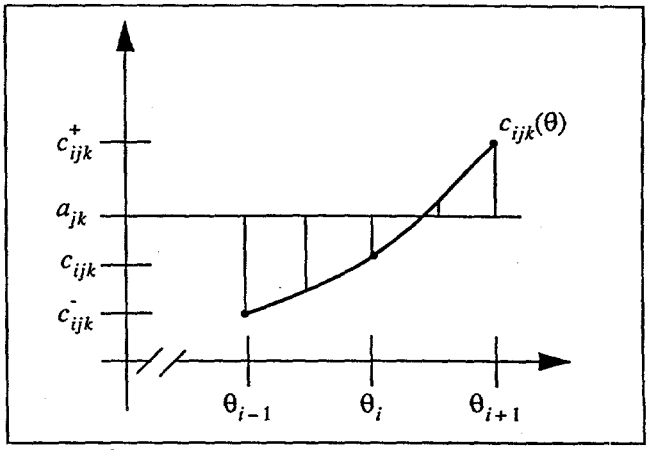


Figure 6: Parametric eigenspace for one eigenimage of one feature at one angle. We compute the distances between the parametric curves and actual eigenvector weights a_{jk} to find the best angle.

$$X^- = [\bar{v}^-_{00} \mid \bar{v}^-_{01} \mid ... \mid \bar{v}^-_{ij} \mid ... \mid \bar{v}^-_{(n-1)(m-1)}]$$

$$X^+ = [\bar{v}^+_{00} \mid \bar{v}^+_{01} \mid ... \mid \bar{v}^+_{ij} \mid ... \mid \bar{v}^+_{(n-1)(m-1)}] .$$

The order of the vectors in these matrices is irrelevant. (The bars in the matrices above represent matrix partitioning and not absolute value.) The dimensions of X, X^-, and X^+ are w^2 x mn. The sample covariance matrix of all $3mn$ features is formed as

$$Q = [X \mid X^- \mid X^+] [X \mid X^- \mid X^+]^T ,$$

whose dimensions are w^2 x w^2. In our implementation, $w = 15$, so the size of Q is a relatively modest 225 x 225.[1] We compute the eigenvectors of Q and designate them as $\bar{e}_j$, where $j \in [0, 1, ..., l-1]$ indexes the l eigenvectors that we use. In our implementation we use 10 eigenvectors. We call the $\bar{e}_j$ "eigenfeatures".

Each feature can be approximated as a weighted linear sum of the first few eigenvectors. These weights are $c_{ijk} = \bar{v}_{ik} \cdot \bar{e}_j$, $c^-_{ijk} = \bar{v}^-_{ik} \cdot \bar{e}_j$, and $c^+_{ijk} = \bar{v}^+_{ik} \cdot \bar{e}_j$. With these coefficients, we form small parametric eigenspaces as quadratics fit through sets of three parameterized points $(\theta_{i-1}, c^-_{ijk})$, (θ_i, c_{ijk}), and $(\theta_{i+1}, c^+_{ijk})$ for all images i, eigenvectors j, and features k. Figure 6 shows a sample quadratic for one eigenvector of one feature in one image. We call this function $c_{ijk}(\theta)$.

3.3 Detection and Pose Estimation

Our algorithm scans the input image in order to find the features on the object. An image to be analyzed must be processed in the same way as the features in the training image.

1. There is a technique outlined in [1] that shows how to compute the eigenvectors of XX^T by instead computing the eigenvectors of X^TX. If X is tall and narrow, this leads to a smaller eigenvector problem. In our case, however, X tends to be wider than it is tall, so we compute the eigenvalues of XX^T directly.

In order to normalize each feature-sized patch, we compute the local power at every point in the image in w x w windows by convolving a w x w, unit-height rectangle function with an image where each pixel has been squared. Each pixel in the image is considered as the center of a feature. Overlapping w x w, feature-sized windows in the image are projected onto the eigenimages. The projections are computed as correlations between the normalized input image and the eigenvectors. Both the convolution and correlations are done in the Fourier domain for speed. We also subtract the mean of all the training features as appropriate. This processing leaves us with an image where each pixel contains l eigenvector coefficients a_j.

To find the object and estimate its pose, we scan through the image in small increments (increments of one to five pixels) looking for the appropriate features in the appropriate spatial configuration. At each point we check all n training angles. For a given image point and a given training angle θ_i, we consider the point to be centered on the first feature ($k = 0$) of the m features for that angle. The centers of the other features are picked up in the image at the appropriate offsets with respect to the first feature. The eigenvector coefficients at these feature points are called a_{jk} where $j \in [0, 1, ..., l-1]$ indexes the l eigenvectors and $k \in [0, 1, ..., m-1]$ indexes the m features. To get a rough estimate of quality of the match at θ_i, we compare the image features to the trained features with no interpolation between angles. We compute a squared distance for each feature as

$$d_{ik} = \sum_{j=0}^{l-1} [c_{ijk} - a_{jk}]^2 ,$$

where the sum is taken over the l eigenvectors. If we have the approximate correct location in the image and the approximate correct angle index i, then all the d_{ik} will be small. If we have the approximate pose but the object is partially occluded, then only some of the d_{ik} will be small, because only some of the features will be visible. Therefore, we use the median to combine the d_{ik} into a single distance measure:

$$d_i = \text{median}_k (d_{ik}) .$$

The rough pose estimate is the position and angle that give the minimum d_i. The resolution of this estimate is limited to the pixel resolution of the image scan in location and the angle increment of the training images in orientation. We designate the best angle index as i^*.

Given the rough pose estimate, we refine the position and orientation with a gradient descent search. We form a distance function $d_{i^*}(\theta)$ that combines the distances between the lm eigenvector coefficients and the corresponding parametric eigenspace quadratics for the rough training angle estimate θ_{i^*}:

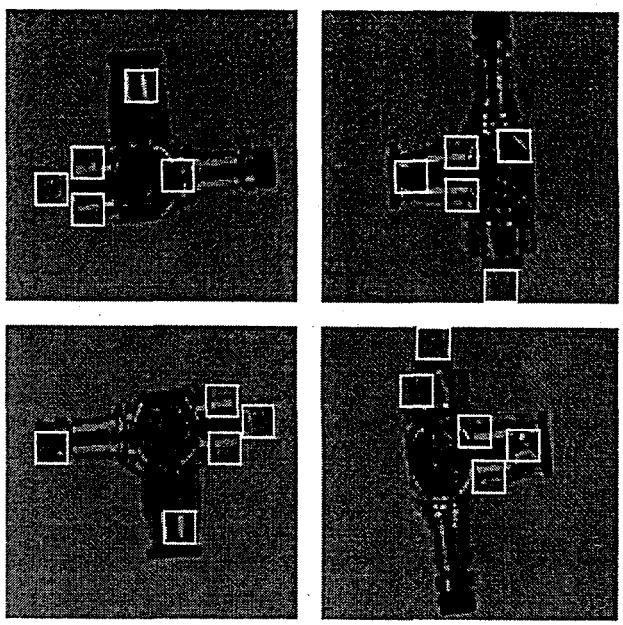


Figure 7: Shiny metal valve used for testing. Automatically found features are shown in rectangles. Specular highlights change depending on the valve's angle, meaning that consistent features are difficult to find.

$$d_{i^*}(\theta) = \sum_{k=0}^{m-1} \sum_{j=0}^{l-1} [c_{i^* jk}(\theta) - a_{jk}]^2 ,$$

where the sums are taken over the m features and l eigenvectors. One of the addends in the sum is illustrated in Figure 6. Since $c_{i^* jk}(\theta)$ is a quadratic in θ, $d_{i^*}(\theta)$ is a quartic, and

$$\frac{d d_{i^*}(\theta)}{d\theta} = 0$$

is a cubic. One of the solutions to the cubic minimizes the sum of squared distances to give the best angle θ. We wrap this closed form minimization in a gradient descent over pixel location to give the final subpixel pose estimate. We do not use a detection threshold since we assume the object is present somewhere in the image.

We were recently made aware of a similar approach to this problem developed by Ohba and Ikeuchi. Like us, they use a principle component analysis of features taken from the object. The major differences are that they employ a step to eliminate similar-looking features, and they use a voting scheme to find the object rather than the image scanning that we use.

4. Results

We tested our algorithm for both accuracy and robustness. For accuracy, we used a shiny, metal valve as the object, shown in Figure 7. The mirror finish on this object meant that the features consisted mostly of specular highlights,

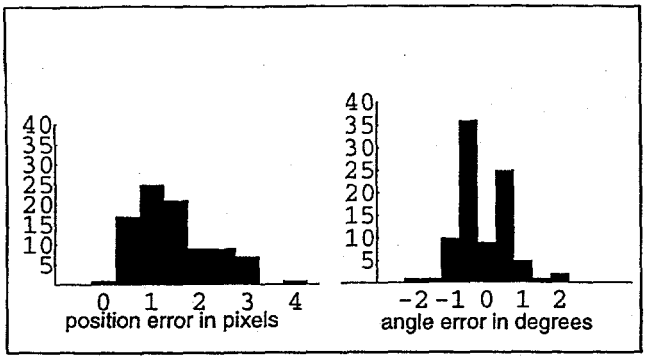


Figure 8: Histograms of position and angle errors for 90 test images of valve.

which changed dramatically as the object rotated. This would confuse any algorithm that depended on tracking features. We took 90 training images four degrees apart and used five features. We tested the algorithm on 90 test images taken at orientations halfway between each training image. We scanned the image in one-pixel increments. Our algorithm found the object in every image, with an average error in position of 1.4 pixels with a standard deviation of 0.8 pixels, and an average absolute value error in angle of 0.6 degrees with a standard deviation of 0.3 degrees. Histograms of the errors are in Figure 8.

Our test for robustness used two different objects - a long, thin wire connector and a shiny, metal pipe connector shaped like a "T". The wire connector has very little overlap between its silhouettes as it rotates, making a single template nearly impossible to use for this object. The "T" connector's shininess makes it difficult to track features on it. For both these objects, we used 180 training images taken two degrees apart and 30 features. We scanned the image first in increments of three pixels and then increased the search resolution to one pixel centered around the best result from the first pass. We tested the algorithm on images with background clutter and partial occlusions. The algorithm correctly found the object in about 80% of the test images. Successful results are shown in Figure 9.

5. Conclusion

We have shown how to use eigenfeatures for pose measurement in the plane. The use of training images to find good features makes the algorithm work without a skilled operator. The eigenvalue decomposition makes the algorithm more efficient than raw pixel matching. By scanning the image for the object, we avoid the problem of segmentation. The innovative use of features rather than monolithic templates allows the algorithm to work in spite of background clutter and partial occlusions. The combination of eigenspace analysis and features provides for a simple, accurate, and robust solution to the planar pose measurement problem.

References

[1] Murakami, Hiroyasa and B.V.K. Vijaya Kumar. "Efficient Calculation of Primary Images from a Set of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(5), September 1982, 511-515.

[2] Murase, Hiroshi and Shree K. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance", *International Journal of Computer Vision*, 14(1), 1995, 5-24.

[3] Murase, Hiroshi and Shree K. Nayar, "Image Spotting of 3D Objects using Parametric Eigenspace Representation", *9th Scandinavian Conference on Image Analysis*, June 1995, 325-332.

[4] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (Second Edition)*, Cambridge University Press, 1992.

[5] Ohba, Kohtaro and Katsushi Ikeuchi. "Recognition of the Multi Specularity Objects using the Eigen-Window". Carnegie Mellon University School of Computer Science Technical Report CMU-CS-96-105, February 29, 1996.

[6] Shi, Jianbo and Carlo Tomasi. "Good Features to Track", *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, 593-600.

[7] Turk, Matthew and Alex Pentland. "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, 3(1), 1991, 71-86.

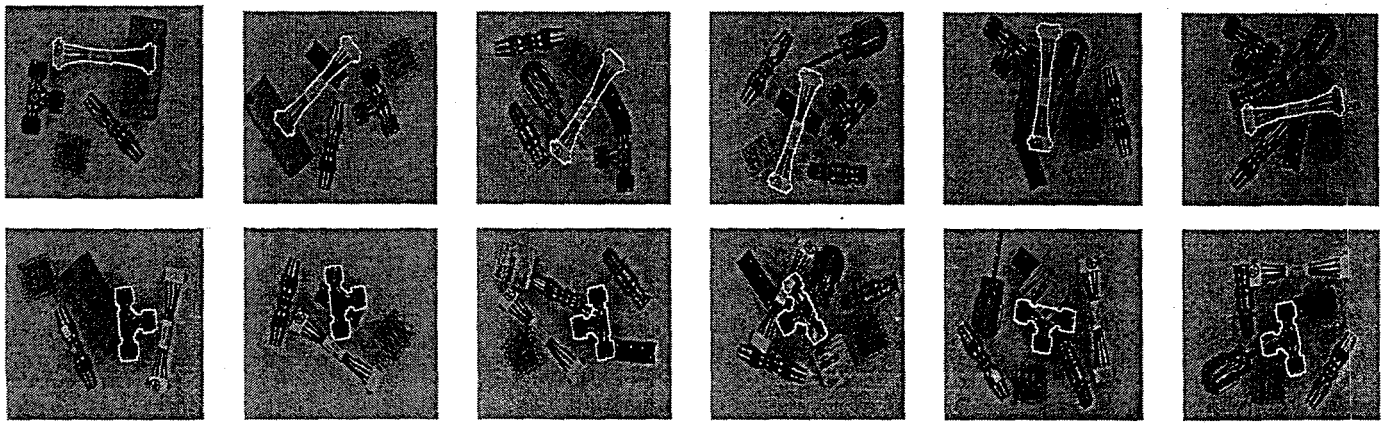


Figure 9: Results of object detection on images of partially occluded wire connector (top row) and "T" connector.

CONF-970679--2

Object Detection with Vector Quantized Binary Features

John Krumm
Intelligent Systems & Robotics Center
Sandia National Laboratories
Albuquerque, NM 87185
jckrumm@sandia.gov

Abstract

This paper presents a new algorithm for detecting objects in images, one of the fundamental tasks of computer vision. The algorithm extends the representational efficiency of eigenimage methods to binary features, which are less sensitive to illumination changes than gray-level values normally used with eigenimages. Binary features (square subtemplates) are automatically chosen on each training image. Using features rather than whole templates makes the algorithm more robust to background clutter and partial occlusions. Instead of representing the features with real-valued eigenvector principle components, we use binary vector quantization to avoid floating point computations. The object is detected in the image using a simple geometric hash table and Hough transform. On a test of 1000 images, the algorithm works on 99.3%. We present a theoretical analysis of the algorithm in terms of the receiver operating characteristic, which consists of the probabilities of detection and false alarm. We verify this analysis with the results of our 1000-image test, and we use the analysis as a principled way to select some of the algorithm's important operating parameters.

1. Overview and Context

Detecting objects in images and measuring their location is a fundamental task of computer vision, with applications in manufacturing, inspection, world modeling, and target recognition. Often the scene is inherently cluttered, the object may be partially occluded, and illumination may change. In this case, the algorithm must look at features internal to the objects' silhouette, and look at them in such a way that missing features and changing illumination are tolerated.

Researchers have responded to this need in many ways, including fairly recent, elegant object detection algorithms based on principle components of training images of the object[6][10]. In particular, Murase and Nayar[6] extract templates from training images of the object in different orientations, compute eigenvector principle components of these templates, and recover the object's orientation in new images by projecting them onto the principle components. They address the problem of illumination changes by taking training images under different lighting conditions. The whole-object template idea was improved by algorithms that look at only part[7] or parts[4][8] of the training templates. This helped to reduce or eliminate the effects of background clutter and partial occlusions.

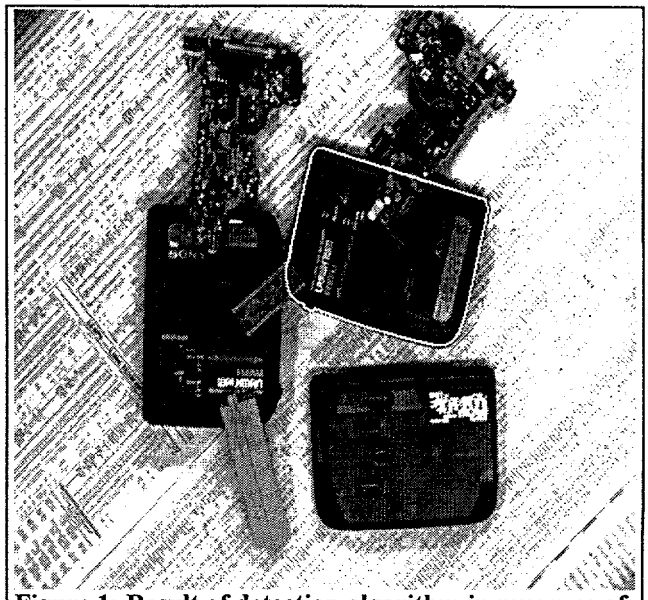


Figure 1: Result of detection algorithm in presence of background clutter and partial occlusions.

This paper presents and analyzes a new algorithm for object detection based on binary subtemplates (features) of the training images. Binary features are more robust to illumination changes than the gray-level features of previous methods. We replace the eigenvector principle components with binary vector quantization, a common method for image compression. This avoids any floating point processing after edge detection. An object is represented with a separate, distinct model for each pose in the training set, which avoids problems caused by self-occlusion and the movement of specular highlights. A model consists of vector quantized binary features and their relative spatial offsets with respect to each other. An example of detection results in an image with background clutter and partial occlusions is shown in Figure 1. On a test of 1000 cluttered images containing the test

This work was performed at Sandia National Laboratories and supported by the U.S. Department of Energy under contract DE-AC04-94AL85000.

object, the algorithm correctly detected the object in 99.3%. In order to determine the best values of the algorithm's important parameters, we derive and plot receiver operating characteristic. This shows the tradeoff between the probabilities of detection and false alarms.

2. Appearance Modeling with Binary Features

An object's pose has six degrees of freedom in general, and a general version of our algorithm would have a separate model for each discretized pose in this six-dimensional space. Such a high-dimensional model space implicitly accounts for the normally confounding effects of parallax, self-occlusion, and the movement of specular highlights. Our detection problem is less general in that we have a camera pointed down at objects resting on a plane which is perpendicular to the camera's optical axis. This eliminates all but one continuos degree of freedom. For now, we will describe the algorithm as having n_m models for a single object, with the models spread over the applicable degrees of freedom. In general, each model M_i, $i \in [1,2,3,\ldots,n_m]$, models one pose of the object to be detected.

A model consists of a set of binary feature vectors (square subtemplates) arranged with respect to each other in image coordinates. Each model comes from a training image of the object in a known pose. We get binary edge images using dilated zero crossings of Laplacian of Gaussian versions of the training images. The Gaussian filter has $\sigma = 2$, and we dilate with an $n_d \times n_d$ structuring element of 1's with $n_d = 3$. The idea to dilate the edges comes from [3], and we do so to help ensure some overlap between training features and possibly corrupted actual features. We designate the edge training images as $e_i(x,y)$, where $i \in [1,2,3,\ldots,n_m]$ indexes the model and $x \in [0,1,2,3,\ldots,n_x - 1]$ and $y \in [0,1,2,3,\ldots,n_y - 1]$ give the pixel coordinates.

In order to eliminate the background from the training images, we made binary masks of the training images by backlighting the rotation table used in training. These backlit images were thresholded to form the object masks

$$b_i(x,y) = \begin{cases} 1 \text{ if } (x,y) \text{ is on object in } v_i(x,y) \\ 0 \text{ otherwise} \end{cases} . \quad (1)$$

A model M_i represents features that are square patches of dilated edge points from $e_i(x,y)$. Using an idea from [9], we pick patches that are easy to localize in that they do not correlate well with their surroundings.

We introduce a windowing operator that extracts a square region of size $(2b+1) \times (2b+1)$ pixels centered around (x,y) and scans the pixels into a column vector:

$$\overline{\Omega}\{f(x,y);x_0,y_0,b\} = [f(x+x_0-b, y+y_0-b), \ \ldots \ f(x+x_0+b, y+y_0-b), f(x+x_0-b, y+y_0-b+1), \ \ldots \ f(x+x_0+b, y+y_0-b+1), \ \vdots \ \vdots \ \vdots \ f(x+x_0-b, y+y_0+b), \ \ldots \ f(x+x_0+b, y+y_0+b)]^T . \quad (2)$$

The windowed neighborhood around each pixel in $e_i(x,y)$ is rated as a feature based on

$$r_i(x,y) = b_i(x,y) \min_{\substack{-d \le d_x \le d \\ -d \le d_y \le d}} \{D_H[\overline{\Omega}\{e_i(x',y');x,y,b\}, \overline{\Omega}\{e_i(x',y');x+d_x,y+d_y,b\}]\} . \quad (3)$$

$D_H(\bar{r},\bar{s})$ is the Hamming distance between binary vectors $\bar{r}$ and $\bar{s}$. The Hamming distance simply counts the number of unequal elements in corresponding positions of its two arguments. In words, $r_i(x,y)$ is computed by taking a square of dimension $(2b+1) \times (2b+1)$ pixels centered on (x,y) in $e_i(x,y)$ and computing its Hamming distance with equal sized squares of pixels centered in the surrounding $d \times d$ neighborhood of $e_i(x,y)$. The minimum of these Hamming distances is the rating of the feature. The feature will rate highly if it is dissimilar to its surroundings. For our program, we chose $b = 7$ pixels to give binary features of size 15x15. We chose $d = 3$ pixels.

The best feature is taken as the $(2b+1) \times (2b+1)$ square surrounding the maximum value in $r_i(x,y)$. Subsequent features are chosen as squares surrounding the next highest value in $r_i(x,y)$ that does not overlap any previous features. Nominally, we chose $n_f = 40$ features based on the analysis in Section 5. Because features can-

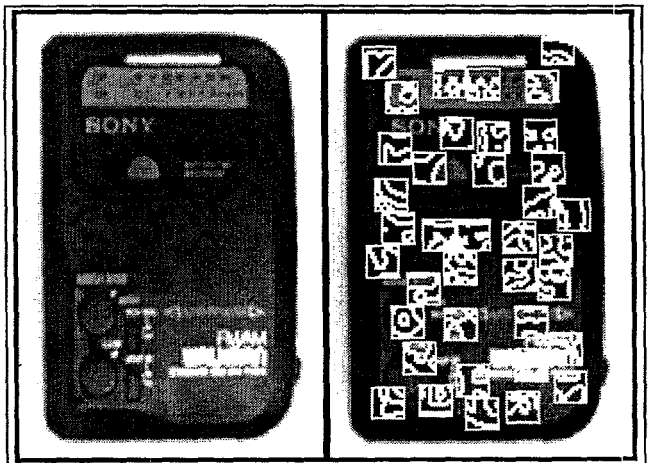


Figure 2: Binary features automatically selected for modeling object

not overlap, a liberal upper bound on n_f can be computed by dividing the area of the smallest object mask $b_i(x,y)$ by the area of a feature. A future step is to allow the number of features to vary from model to model. The features chosen for one model of one of our objects of interest are shown in Figure 2.

The binary features are scanned into column vectors called $\bar{f}_{ij}$ with $i \in [1,2,3,\ldots,n_m]$ indexing the models and $j \in [1,2,3,\ldots,n_f]$ indexing the features within the models. The corresponding locations of the features in the training images are $\bar{x}'_{ij} = \left(x'_{ij}, y'_{ij}\right)$. Since the absolute location of the object in the training image is unimportant, we take the feature locations as offsets with respect to the first feature in each model: $\bar{x}_{ij} = \bar{x}'_{ij} - \bar{x}'_{i1} = \left(x'_{ij} - x'_{i1}, y'_{ij} - y'_{i1}\right)$.

By picking an independent set of features from each training image, we are not forced to depend on tracking training features from pose to pose. This is difficult if features appear and disappear due to specular highlights or self-occlusions.

As we mentioned above, our problem is one of detecting objects in stable poses resting on a plane with a camera looking down from above (*e.g.* conveyor belt). We model translation of the object parallel to the resting plane as proportional translation in the image, which assumes away effects of parallax and image distortion, *i.e.* the image of the object is shift-invariant. The degree of freedom that we model, then, is rotation around an axis perpendicular to the resting plane. Each model M_i models the object at angle θ_i.

We can compute the number of models n_m we need in this situation by considering the feature size. The training images of the object are separated by $\Delta\theta$ in angle. This angle should be small enough that a binary feature should not change appearance over the range $[\theta_i - \Delta\theta/2, \theta_i + \Delta\theta/2]$. Measured from the center of a $(2b+1)$ x $(2b+1)$ feature, the distance to the center of the farthest pixel is $\sqrt{2}b$. This pixel will move along an arc of length $\sqrt{2}b\Delta\theta$ between training images. We endeavor to make this arc length much less than one pixel to ensure that the feature will not change over the $\Delta\theta$ range in angle. If we set $\sqrt{2}b(\Delta\theta) = 0.2$ pixels, then $b = 7$ gives $\Delta\theta = 1.16°$. We set $\Delta\theta = 1.0°$, giving $n_m = 360$.

3. Encoding Binary Features

The recent work in subspace methods for object recognition [3, 4, 6-8, 10], as well as the standard principle component analysis of pattern recognition, can be thought of as applications of data compression. A set of high-dimensional training vectors are projected into a lower-dimensional space to serve as efficient models of the items in the training set. When new data is to be classified, it is compressed in the same way as the training data and then compared to the compressed version of the training data. The advantage of this approach is not only efficiency, but that the compression process groups similar features, thereby tolerating the inevitable variations in features from image to image. We verify this assertion at the end of this section.

Looking to standard methods in image compression, we found no close analogue of eigenvectors for binary data like our training features $\bar{f}_{ij}$. Huttenlocher *et al.*[3] use binary whole-object templates compressed with real-valued eigenvectors for matching. The eigenvectors serve as floating point principle components onto which the binary edge images are projected, giving real-valued compressed versions of binary training templates. One of the goals of our work was to avoid using any floating point processing on binary data. Thus, instead of eigenvectors, we chose to use binary vector quantization, which preserves the binary nature of the data in its compressed form.

Traditionally, the goal of vector quantization has been image compression[1]. The image to be compressed (binary, gray level, or color) is split into rectangular blocks. Each block is compared to a relatively small set of previously determined code blocks of the same size. The image block is represented by the index of the most similar code block. This index is transmitted or stored, and the image can be approximately reconstructed from the indices and code blocks. In our case, we have a set of $n_m n_f$ training features $\bar{f}_{ij}$ that we want to represent with a much smaller set of n_c code features $\bar{F}_a$, $a \in [1,2,3,\ldots,n_c]$. Each code feature is the same size as the training features, *i.e.* $(2b+1)^2$.

We use the standard method for computing code features, the Lloyd algorithm, described in [1]. This is the same as the k-means algorithm from pattern recognition. The first iteration of the Lloyd algorithm uses code features randomly selected from the training set. Each of the training features is matched with its closest code feature. To measure distance between features, we use the Hamming distance, described above. The new set of code features is taken as the centroids of the clusters of matched features in the previous iteration. The iterations continue until the code features stop changing, or until the total dissimilarity is deemed small enough, or until the program has run too long. In our case, we found that 10 iterations were enough to produce a good set of code features.

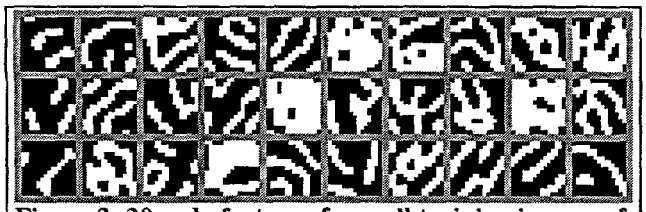


Figure 3: 30 code features from all training images of object in Figure 2.

To compute the centroid of a set of binary vectors based on the Hamming distance, we have corresponding elements of each vector vote for "0" or "1". For determining each element of the centroid, the majority rules, with ties broken randomly.

The result of the Lloyd algorithm is a mapping from any training feature $\bar{f}_{ij}$ to the index of its corresponding code feature $\overline{F}_{c_{ij}}$. We designate this mapping as $N(\bar{f}_{ij}) = c_{ij}$. 30 code features for all 360 training images of the object in Figure 2 are shown in Figure 3.

We can assess the quality of the mapping by computing the probability that a feature will be miscoded. This helps to select the number of codes, n_c, that we will choose, as described in Section 5. We could derive the probability of miscoding by assuming some probability distribution on the features, or we could approximate the probability with an experiment. We chose the later approach.

In order to approximate the probability of miscoding, we took 360 test images of the object in Figure 2 at angles halfway between the $n_m = 360$ training images. For each feature $\bar{f}_{ij}$, we extracted a test feature at the same relative location as the training feature in the two adjacent test images. We coded these test features, and took the probability of miscoding as the fraction of these features that were *not* mapped to the same code as their respective training feature. We repeated the experiment for different numbers of code features, n_c, rerunning the Lloyd algorithm each time we changed n_c. The result is plotted in Figure 4. Of course, this experiment does not account for all anticipated variations of the features (*e.g.* illumination effects), but it does give an approximate idea of miscoding probabilities as well as the general behavior of the detection algorithm with variations in n_c.

As we expect, a small number of code features leads to a small probability of miscoding. This is because the nearest neighborhood of each code feature is large, so the chance of straying to another neighborhood is small. This supports our assertion at the beginning of this section that that one advantage of subspace methods is that small appearance variations are tolerated by the many-to-one

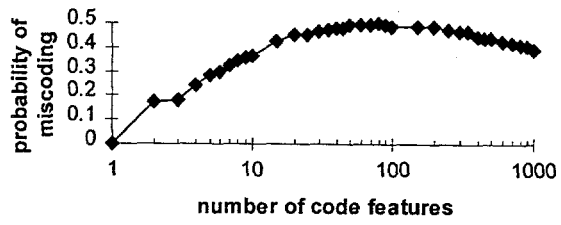


Figure 4: An experiment shows that the probability of miscoding a feature rises and then falls with the number of code features chosen.

mapping. Based on this data, and the analysis in Section 5, we chose $n_c = 30$ codes, which gives a probability of miscoding of $p_b = 0.464$.

Figure 4 shows that the probability of miscoding peaks at about $n_c = 80$ and then starts to drop. We did not expect this behavior, and we are still speculating on the cause. It may be that for large numbers of code features, the code features represent training features that really belong together in some sense, while for smaller numbers of code features, there is a "forced marriage" between features that are really not very similar. An interesting extension of this work would be to explore the implications of large numbers of code features.

4. Detecting an Object

An object model M_i consists of a set of feature codes and locations for one pose of the object:

$$M_i = \left\{(\bar{x}_{i1}, c_{i1}), (\bar{x}_{i2}, c_{i2}), \ldots, (\bar{x}_{in_f}, c_{in_f})\right\}. \quad (4)$$

We search for these features in a preprocessed image in order to detect the object.

The input image is processed the same way as the training images to give $e(x,y)$, an image of dilated Laplacian of Gaussian zero-crossings. We then code the $(2b+1)\,\mathrm{x}\,(2b+1)$ neighborhood around each pixel with the code features $\overline{F}_a$ computed from the Lloyd algorithm. The corresponding image of code indices is

$$c(x,y) = \arg\min_{a \in \{1,2,\ldots,n_c\}} \left\{ D_H\left[\overline{\Omega}\left\{e(x',y'); x, y, b\right\}, \overline{F}_a\right]\right\} \quad (5)$$

To detect an object, we search the image for all the models M_i of the object. Since we assume that the object's appearance is space-invariant, we search over all translations of all models M_i. We keep track of the search with a three-dimensional Hough transform $H_{ix'y'}$, whose three indices correspond to the three search parameters:

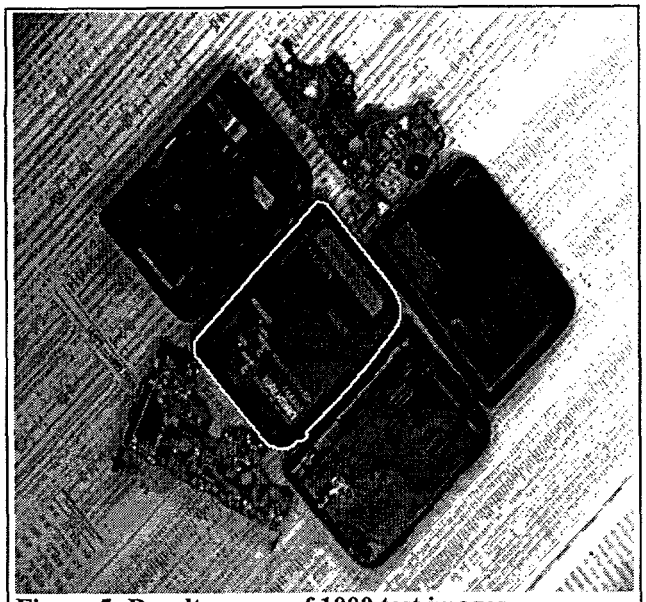


Figure 5: Result on one of 1000 test images.

$$\begin{aligned} &i \in [1,2,3,\ldots,n_m] \text{ indexes models} \\ &x \in [0,1,2,\ldots,n_x] \text{ indexes image column} \\ &y \in [0,1,2,\ldots,n_y] \text{ indexes image row} \end{aligned} \quad (6)$$

The bins of the Hough transform keep track of how many features were found for each model M_i at each location in the image.

The Hough transform is filled by consulting a simple geometric hash table upon encountering each code index in $c(x,y)$. Each code index lends evidence to several different model and translation possibilities. Upon encountering a particular code index, the algorithm indexes into the hash table to find models that contain that code. Each occurrence of an equivalent code in a model causes one bin of the Hough transform to be incremented. The incremented bins correspond to the model index i and the position of the first feature on that model translated by the position of the code index in $c(x,y)$. A standard geometric hash table must be indexed by pairs of points (or more)[5]. Since we have a distinct model for each pose of the object, we can index on single points.

Each bin in the filled Hough transform contains a count of the number of features found that support a given model at a given location. We declare a detection of model M_{i_0} at location (x_0, y_0) if $H_{i_0 x_0 y_0} \geq k_T$. The integer k_T is a detection threshold that specifies the number of features that must be found for the object be considered detected. Our nominal value is $k_T = 13$, based on our analysis in Section 5.

We tested the program on 1000 images of the unoccluded object in a cluttered background, one of which is shown in Figure 5. Of these 1000, the object was correctly detected on 99.3% with at least $k_T = 13$ features and with no other detections exceeding the number of features correctly found. We also tested the algorithm on images with partial occlusion as in Figure 1. The algorithm works in such cases, but we have not performed a statistically significant test.

After offline training, the program takes 5.5 minutes to compute the pose of an object in a new image, running on a 50 MHz Sun Sparc 10. The bulk of the time is devoted to encoding the image ($\approx$ 2.5 minutes) and filling the Hough transform ($\approx$ 3.0 minutes).

5. Receiver Operating Characteristics

The "receiver operating characteristic" is a way of measuring the performance of a signal detection algorithm[11]. The receiver operating characteristic graphically shows the tradeoff between the probability of a detection and the probability of a false alarm. A false alarm occurs when the algorithm signals a detection, but the object was not really present. The tradeoff in our case is controlled by the detection threshold k_T. If k_T is low, then the algorithm is more likely to find the object, even if many of the features are not found. But, a low k_T will also increase the chance of a false alarm. Conversely, a high k_T will decrease the chance of a false alarm, but also decrease the chance of a valid detection. The receiver operating characteristic is useful for assessing and optimizing the performance of the algorithm. Typical signal detection algorithms are less procedural than ours in that they normally consist of a set of measurements plugged into an equation. While our algorithm is less straightforward, it still admits to a mathematical analysis of probabilities.

Traditionally, the receiver operating characteristic considers the case of a single (possibly multidimensional) measurement. For our case, it is more useful to modify the receiver operating characteristic to show what we are most interested in for a computer vision application. We will base our probability calculations on the algorithm's behavior on a search through the entire image rather than just a single measurement. The two probabilities that we calculate are:

$P(\text{detection})$ = probability that algorithm will find correct model in correct location given that unoccluded object appears somewhere in image

$P(\text{fase alarm})$ = probability that algorithm will find any model anywhere in image given that object does not appear anywhere in image

This analysis is similar in spirit to that of Grimson and Huttenlocher[2], who analyze the Hough transform for pose measurement. However, they analyze a situation

where each feature fills a swath of bins in the Hough transform, while our formulation only fills several disconnected bins. They warn that the probability of false positives can be very high, but our theory and experiment show that it is easy to bring the probability of a false positive arbitrarily low with our algorithm.

5.1 Probability of False Alarm

We will begin by computing the probability of a false alarm, since this is easier, and the same techniques will be used in the next section. Our algorithm reports a detection if it finds at least one model M_i somewhere in the image supported by at least k_T features. For lack of a reason to believe otherwise, we will assume that the feature codes in an image with no object are uniformly distributed. Given this, the probability of observing a given feature index at a given location in $c(x,y)$ is n_c^{-1}, where n_c is the number of codes we choose to use. When the algorithm considers the existence of a given model at a given location, it looks for n_f code indices in a certain geometrical arrangement in $c(x,y)$. The probability of finding any subset $m \subset M_i$ of exactly l features at l different locations and not finding specific features at the remaining $n_f - l$ locations in $c(x,y)$ is given by a binomial distribution:

$$b\left(l; n_f, n_c^{-1}\right) = \binom{n_f}{l} n_c^{-l}\left(1-n_c^{-1}\right)^{n_f-l} \tag{7}$$

The probability of falsely reporting a detection of a given model *at a single location* is the sum of the probabilities of finding between k_T and n_f features:

$$p_f = \sum_{l=k_T}^{n_f} b\left(l; n_f, n_c^{-1}\right). \tag{8}$$

It is clear from this equation that a lower value of the detection threshold k_T increases the chance of a false detection.

The probability of a false alarm, as we have defined it, is the probability that a model with at least k_T features will be found somewhere in the image. We calculate this probability as follows:

$$\begin{aligned} &P(\text{false alarm}) \\ &= P(\text{finding } \geq k_T \text{ features somewhere in image}) \\ &= 1 - P(\text{not finding } \geq k_T \text{ features anywhere in image}) \quad (9) \\ &= 1 - b\left(0; n_m n_x n_y, p_f\right) \\ &= 1 - \left(1-p_f\right)^{n_m n_x n_y} \end{aligned}$$

where we calculate the probability of not finding $\geq k_T$ features anywhere in image as a binomial probability with zero successes out of $n_m n_x n_y$ tries with a probability of success on any given trial of p_f. As we expect, the probability of a false alarm rises with the size of the image ($n_x n_y$) and the number of models searched for (n_m).

5.2 Probability of Detection

A successful detection occurs when the algorithm finds at least k_T features on the object at the correct pose and no greater number of features from a single model anywhere else. Since we assume that the problem is translation invariant, the object's position in the image makes no difference, and we start with the following probability of a successful detection:

$$P(\text{detection}) = \sum_{i=1}^{n_m} P\left(\text{finding model } M_i \middle| M_i\right) P(M_i). \tag{10}$$

By the event "M_i" we mean that the object is at the orientation represented by M_i. We will assume that instances of the model in the images are uniformly distributed, so that $P(M_i) = n_m^{-1}$.

We further specify the probability of a detection as

$$n_m^{-1} \sum_{i=1}^{n_m} P\left(\begin{array}{l} \text{finding } m \subset M_i \text{ such that } S(m) \geq k_T \\ \text{at correct location } \textit{and} \text{ not finding} \\ \text{any } \tilde{m} \subset M_j \text{, j} \in [1,2,3,\ldots,\text{n}_\text{m}] \text{ such} \\ \text{that } S(\tilde{m}) \geq S(m) \text{ at any other location} \\ \text{in image} \mid M_i \end{array}\right). \tag{11}$$

$S(m)$ is the number of elements in the set m. The event specified in this equation is that of finding at least k_T features out of the set of features that indicates the model and not finding a larger set of features from the models anywhere else. We split this term into mutually exclusive events based on the number of features found on the correct model, noting also that the two statements connected by "*and*" are independent events:

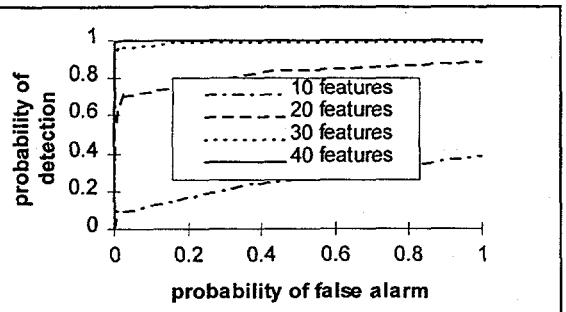


Figure 6: Theoretical receiver operating characteristic as detection threshold varies for different number of features and constant 30 codes. 40 features are adequate. The bend in the curve for 40 features occurs at a detection threshold of 13 features.

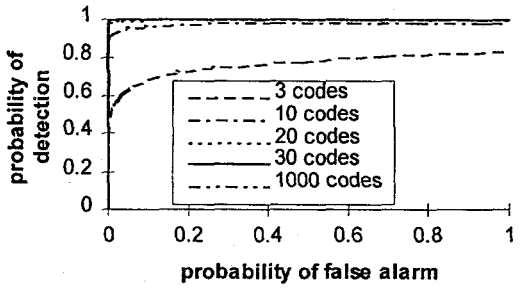


Figure 7: Theoretical receiver operating characteristic as detection threshold vaires for different number of codes and constant 40 features. 30 codes are adequate. The bend in the curve for 30 codes occurs at a detection threshold of 13 features.

$$n_m^{-1}\sum_{i=1}^{n_m}\sum_{k=k_T}^{n_f}\left\{P\left(\text{finding } m\subset M_i \text{ such that } S(m)=k \middle| M_i\right)\left[1-P\left(\begin{array}{l}\text{finding any } \tilde{m}\subset M_j, j\in[1,2,3,\ldots,n_m]\\ \text{such that } S(\tilde{m})\geq k \text{ at any other}\\ \text{location in image}\end{array}\right)\right]\right\} \tag{12}$$

The correct way to compute the first term in the product above is to consider all possible subsets $m\subset M_i$. Since there are so many possibilities, this is prohibitively expensive in computer time. Instead, we assume that each feature has the same probability of being miscoded, p_b, as computed at the end of Section 3. Then the probability of finding exactly k features and miscoding n_m-k features for all possible sets m such that $S(m)=k$ is

$$b(k;n_f,1-p_b)=\binom{n_f}{k}(1-p_b)^k(p_b)^{n_f-k}. \tag{13}$$

For the second term in the product in Equation 12, we assume again that the feature codes in the background of the image are uniformly distributed. The probability of finding any specific combination ($\tilde{m}\subset M_j$) of k or greater features at a single location is

$$p_m=\sum_{l=k}^{n_f}b\left(l;n_f,n_c^{-1}\right). \tag{14}$$

Omitting the correct pose, there are $n_m n_x n_y-1$ opportunities to find k or greater features from any model. The second term in the product in Equation 12 becomes

$$\begin{aligned}&1-P(\text{finding } \geq k \text{ features somewhere in image})\\&=1-\sum_{i=1}^{n_m n_x n_y-1}b\left(i;n_m n_x n_y-1,p_m\right)\\&=1-\left[1-b\left(0;n_m n_x n_y-1,p_m\right)\right]\\&=\left(1-p_m\right)^{n_m n_x n_y-1}.\end{aligned} \tag{15}$$

Noting that we have assumed away any dependence on the particular model M_i, the probability of detection is

$$P(\text{detection})=\sum_{k=k_T}^{n_f}\left\{b\left(k;n_f,1-p_b\right)\left[1-\sum_{l=k}^{n_f}b\left(l;n_f,n_c^{-1}\right)\right]^{n_m n_x n_y-1}\right\}. \tag{16}$$

5.3 Receiver Operating Characteristic Curves

Receiver operating characteristics based on the probabilities of false alarm and detection derived above are plotted in Figure 6 and Figure 7.

Receiver operating characteristic curves are generally parameterized by a detection threshold. In our case, the detection threshold is k_T, which gives the number of features that must be found to consider the object present. A good receiver operating characteristic curve will look like the symbol Γ, with the vertical segment coincident with the vertical axis, and the corner being at point (0,1). Such a curve means that the false alarm rate stays low even for high probabilities of detection. Our goal is to adjust parameters in an attempt to reach an ideal curve shape and then take k_T to be the value at the upper left corner of the curve. Figure 6 shows the effect of varying the number of features with the number of codes kept constant at $n_c=30$. It shows that 40 features give a good curve. The sharp bend occurs at $k_T=13$.

Figure 7 shows the effect of varying the number of codes with the number of features kept constant at

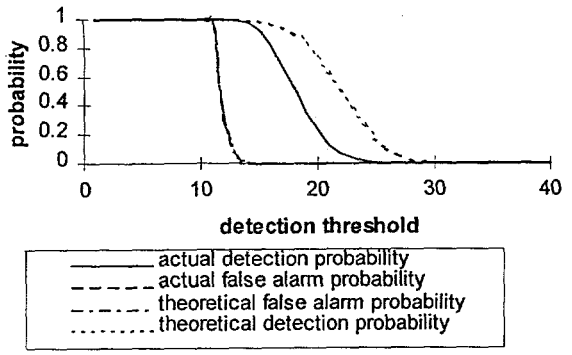


Figure 8: Actual and theoretical receiver operating characteristic.

$n_f = 40$. It shows that 30 codes give a good curve, with the sharp bend in the curve occurring at $k_T = 13$. This is how we chose the number of features, number of codes, and detection threshold.

In our test of 1000 images, we kept track of the number of features found for the top 10 detections for each image. Using this data, we plotted an empirical receiver operating characteristic curve, as show in Figure 8. We also plotted the theoretical receiver operating characteristic curve ($n_f = 40$ and $n_c = 30$). As shown, the false alarm rate is very well predicted by Equation 9. Equation 16 tends to overestimate the probability of detection slightly, which could be due to a higher probability of miscoding than what our experiment in Section 3 showed.

Nearly every computer vision algorithm comes with parameters that must be adjusted by the user - so-called "magic numbers". We list the magic numbers used by our program in Table 1. The ideal computer vision algorithm has no magic numbers, which means it does not have to be adjusted for different situations. The next best alternative is to provide a principled method to choose the parameters. As shown in Table 1, we were able to do this for about half the adjustable parameters, with the receiver operating characteristic accounting for three of the most important.

6. Summary

This paper presents a new algorithm for detecting objects in images. It uses models based on training images of the object, with each model representing one pose. Since each pose is modeled uniquely, this helps reduce the confounding effects of specular highlights, and eliminates the need to track features during training. Objects are modeled in terms of square binary edge patches that are automatically selected from the training images based on their dissimilarity with their surroundings. Internal features means the algorithm is robust in the face of background clutter. The features are compressed using binary vector quantization, which gives an efficient representation of the models. The detection algorithm fills a 3D Hough transform. We derive the probabilities of detection and false alarm (receiver operating characteristics) and use this analysis to determine some of the important operating parameters of the program.

parameter	value	how set
image size (n_x, n_y)	(512,480)	preset
smoothing σ	2 pixels	by eye
dilation n_d	3 pixels	experience
feature size b	7 pixels	experience
feature correlation distance d	3 pixels	experience
number of models n_m	360	subpixel feature change (Section 2)
probability of miscoding p_b	0.464 (for $n_c = 30$)	miscoding experiment (Figure 4)
number of features n_f	40	receiver operating characteristic
number of codes n_c	30	receiver operating characteristic
detection threshold k_T	13 features	receiver operating characteristic

Table 1: Settings of parameters (magic numbers)

References

[1] Gray, Robert M., "Vector Quantization", *IEEE ASSP Magazine*, April 1984, pp. 4-29.

[2] Grimson, W. Eric L. and Huttenlocher, Daniel P. "On the Sensitivity of the Hough Transform for Object Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3), March 1990, pp. 255-274.

[3] Huttenlocher, Daniel .P., Lilien, Ryan H., and Olson, Clark F. , "Object Recognition Using Subspace Methods", *Proceedings of the Fourth European Conference on Computer Vision*, 1996, pp. 536-45.

[4] Krumm, John C., "Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects", *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, June 1996, pp. 55-60.

[5] Lamdan, Yehezkel and Wolfson, Haim J., "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", *Proceedings of the Second International Conference on Computer Vision*, December 1988, pp. 238-249.

[6] Murase, Hiroshi and Nayar, Shree K., "Visual Learning and Recognition of 3D Objects from Appearance", *International Journal of Computer Vision*, 14(1), 1995, pp. 5-24.

[7] Murase, Hiroshi and Nayar, Shree K., "Image Spotting of 3D Objects using Parametric Eigenspace Representation", *9th Scandinavian Conference on Image Analysis*, June 1995, 325-332.

[8] Ohba, Kohtaro and Ikeuchi, Katsushi, *Recognition of the Multi Specularity Objects using the Eigen-Window*, Carnegie Mellon University School of Computer Science Technical Report CMU-CS-96-105, February 1996.

[9] Shi, Jianbo and Tomasi, Carlo, "Good Features to Track", *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, June 1994, pp. 593-600.

[10] Turk, Matthew and Pentland, Alex, Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, 3(1), 1991, 71-86.

[11] Van Trees, Harry L., *Detection, Estimation, and Modulation Theory, Part I, Detection, Estimation, and Linear Modulation Theory*, John Wiley & Sons, New York, 1968.