Breaking it Down: The World as Legos

Benjamin Savage, Eric Chu

To devise a general formalization for identifying objects via image processing, we suggest a two-pronged approach of identifying principal parts and model fitting to encode spatial relationship. We begin with a relatively simple object, the cube. Additional motivation for this choice was the fact that much of the man made world is composed of cubic shaped structures, so cube recognition could serve as an input to later, more complex object recognition tasks. Our recognition formalization begins with training SVMs to identify the principle parts of an object. For our project we used an SVM to pick out cube corners like those appearing in man-made objects. In the second stage of our formalism, information from the detected locations of the principle object components is used to fit a pre-defined model to the location data. We then used the minimal least-squares affine transformation to fit an arrangement of seven corners—our "cube-model"—to various permutations of the detected corners. This model contains information about the spatial relationships between the constituent corners and successfully distinguishes between possible cubes and impossible cubes.

Introduction

Current object recognition algorithms face the challenge of generalization. It is possible for a machine to identify a teacup if the teacup is presented to the machine in the same size and orientation; otherwise, the machine falters in its recognition. In an attempt to model human vision and human spatial cognition, the purpose of this project is to take a step towards making object detection and recognition rotationally invariant, scale invariant, and partially occlusion invariant. By constructing object models and discovering a least-squares mapping from a suspect object to its model, we hope to achieve that end.

The Overview

After a cursory observation of the human visual system, we generated a list of three major elements that contribute to the flexibility of human object recognition:

1. Using parts and their relationships to identify the whole
2. Using context to identify the object
3. Being able to mentally manipulate / rotate the object

We have successfully implemented very basic elements of each of these three components in our attempt to construct a better object recognition algorithm. Figure 1 diagrams the general flow chart devised for this particular scheme.

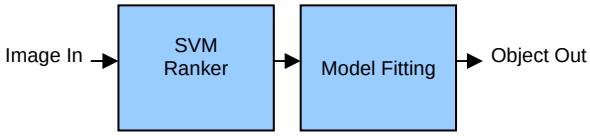


Figure 1 Flow chart of algorithm.

The SVM step performs the operation of decoding contextual information as well as identifying fundamental parts of an object. The Model Fitting step provides the computer with a mathematical framework with which to rotate, scale, translate, and skew the object. It also encodes the spatial relationships between the fundamental parts of an object. Using these steps in conjunction with each other, we are able to identify very basic cubes.

From now on, we shall discuss how the algorithm applies to identifying cube-like objects and conclude by elaborating on how it can be generalized to a broader framework.

The SVM

Since the simple building blocks of a cube are its corners, the main purpose of the SVM / machine learning algorithm is to reinforce or discourage certain classifications of pixels as "cube-corners". These corners will be used later in object mappings.

In early incarnations of this project, we attempted to use the Harris corner detector to detect "cube-corner" candidates. This approach was unsuccessful because it only used local pixel data and did not use contextual clues to make predictions. We had much more success with the SVM classifier by using large image patches and geometrical information to encode contextual knowledge into our classifier.

We first converted our color image to an intensity image, and then took the gradient of that image. To recognize "cube-corners" in this image we extracted 51x51 pixel image patches to make sure that some amount of context was observed. Experimentation with training an SVM to *directly* classify image patches of this kind showed little promise for many reasons. Therefore, to both endow the SVM with a sense of the spatial ties between the various pixels in this patch, and to substantially reduce the dimension of the space, we found the projection of this 2601-vector (51x51 = 2601) into a 277 dimensional space. The projection "basis" chosen was a collection of geometric masks that were blurred significantly. Below (figure 2) are artistic renditions of 3 such spaces.

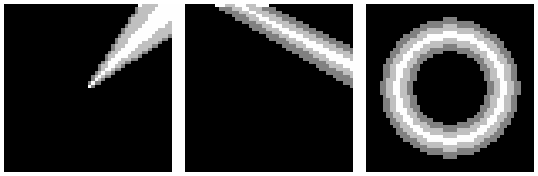


Figure 2 Artistic renditions of our basis masks applied on to 51x51 image patches.

These subspaces found trends in various dimensions, while remaining invariant to a good deal of noise. Furthermore, they allowed us to highlight significant characteristics of a corner in a 51x51 patch (such as an edge emanating from the center pixel). Using this technique gave superior results to using PCA to create a basis of "eigen-corners", and far superior results to using the direct 51x51 cutout.

In the final step, the 277 vector given by projecting into this subspace is scaled to unit length to eliminate the variation caused by differences in lighting. This 277 element vector was used as training and test vectors in an SVM. Using a Gaussian kernel in our SVM with a γ of 5, we were able to obtain less than 13% error on a leave-one-out cross-validation test.

As another form of validation we ran the SVM on every single 51x51 image patch in an image not used in our training set. Every pixel is labeled with its margin from the support vectors. More "confident" points are highlighted in red. The results are shown below. These results were even more encouraging than the LOOCV test error. Not only did the SVM find most cube-corners in the image, but the confidence of the prediction seems to be well correlated with how well centered the corner is in the cut-out. We described this image as a "corner-heatmap". (Note: our SVM was further improved after using it to create this image, but as the image took 30 minutes to generate we decided not to update this image.)

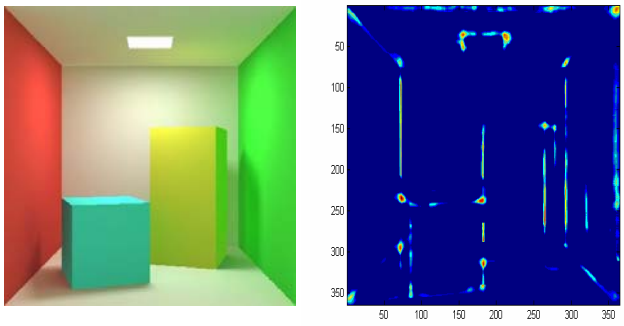


Figure 3 Original photo (left)—the Cornell Box—and corner-heatmap (right).

Cube Regression

We developed mathematical machinery to grapple with the task of identifying cubes from our detected corner data. The general idea is laid out in figure 4. We modeled a perspective of a cube (or even more generally, any object with highly distinguishable "principle" image components such as corners) as a set of 3-D homogeneous points X = {x(1), x(2), … , x(m)}. When appearing in an image, this set of points may be rotated, scaled, and translated. On top of this there may be random noise, or perhaps distortion due to the fact that the cube is being viewed from a slightly different perspective.

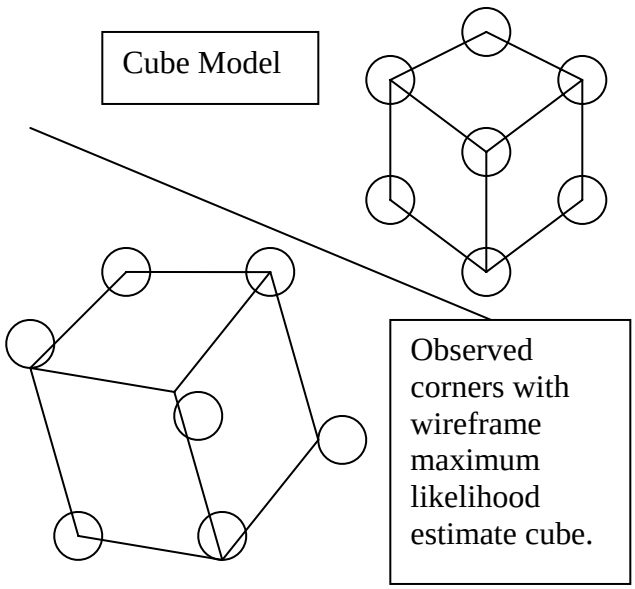


Figure 4. Diagram of cube regression in action.

Thus, we will model the positions of all of the corners of the cube in the image as an affine transformation of the x(j), added to a 2-D Gaussian error term, as in equation 1.

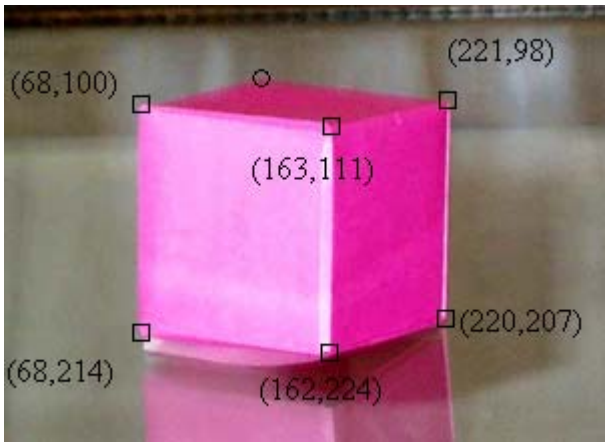
$$z^{(i)} = Ax^{(i)} + \varepsilon \qquad (1)$$

Note that x is in homogenous coordinates (so that A contains translation information). This model allows us to find the Likelihood of this arrangement of the corners given the parameters of the affine transformation. We even briefly toyed with the idea of viewing this problem in a Bayesian framework where there was some prior distribution on the transformation A, although we didn't have time to explore this idea in depth. We found the maximum likelihood estimate of the affine transformation to be as follows.

$$W = S \cdot X^T \left(X \cdot S \cdot X^T \right)^{-1} \qquad (2)$$

$$A_i = Z^{(i)} \cdot W \qquad (3)$$

In equation (2), $Z^{(i)}$ is a 3 x *m* matrix of the observed corner locations (of image *i*) in homogenous coordinates, (see figure 5), X is also a 3 x *m* matrix but this time of the model points, and S is a diagonal matrix of weightings used to weight the error of each point relative to one another. (It is the inverse covariance matrix when the distribution of noise around each point is assumed to be a circular Gaussian).



$$Z^{(i)} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 68 & 0 & 221 & 163 & 68 & 162 & 220 \\ 100 & 0 & 98 & 111 & 214 & 214 & 207 \end{bmatrix}$$

Figure 5. (Top) An example training image. (Bottom) The $Z^{(i)}$-matrix used in equation 3. Note that the 2nd point is missing (and hence all zeros).

We were pleased with the resemblance of this weighted regression formula to the normal equations. In our tests we found that this worked quite well. This formula allowed us to quickly find the likelihood that a set of points was generated by a given model. As desired it is completely invariant to translation, rotation in the plane, scaling, and is mildly robust to minor distortion. We even found a method of fitting less than the full *m* points by setting ones to zeros in the homogenous coordinate. If the SVM did not observe a corner (either it was not in the image, or the SVM simply made an error), A was still calculated, using the remaining data from the other points (with the zero-ed column). This allowed us to realize the dream of making a model robust to partial occlusion.

To generate the "cube-model" we derived a form of the EM algorithm. By finding the ordered locations of all visible corners in hundreds of observed cubes {Z_1, Z_2, …, Z_n}, we generated a training set. By assuming each training example was generated by the same model under an affine transformation A_i, for i = 1 to n, and treating the A_i as the latent variables, we were able to find the best "cube-model". We were also able to find the variance of error for each individual point to create the weighting matrix S. Experimentation showed that using this S gave superior results to the non-weighted case of simply using the identity matrix for S. The EM algorithm was extremely efficient and converged in around 10 steps. Figure 6 shows the results of this algorithm after convergence.

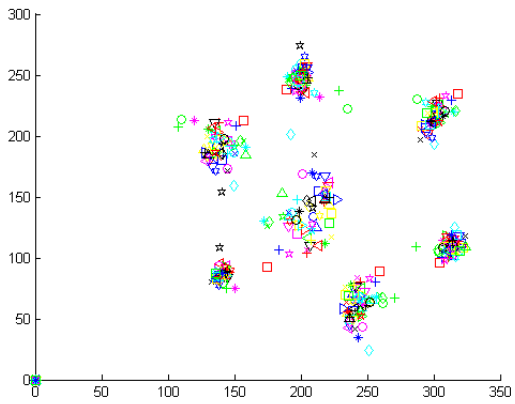


Figure 6. Note the seven major clusters corresponding to the seven prominent corners of a cube. Also, note that the "center" corner has the most variance.

This mathematical side of our project was far more satisfying than the image processing portion. We were able to apply the techniques taught in class (Linear regression, EM algorithm, multivariate gaussians, maximum likelihood estimation), to achieve the goal of storing the spatial relationships of parts of an object in a mathematical model, and to create an affine transformation invariant object detection framework. We encountered one truly difficult challenge however when it came time to implement this methodology: factorial blowup. Unfortunately, the order of the corner points is important, and so there was a factorial blow up. When 150 cube corners were detected in an image, say, we had to try all

$[({}_7C_0)* ({}_{150}P_7)] + [({}_7C_1)* ({}_{150}P_6)] + [({}_7C_2)* ({}_{150}P_5)] + [({}_7C_3)* ({}_{150}P_4)]$
possible permutations (each bracketed term corresponds to a certain number of un-observed corners).

There is still hope that one might solve this problem. It makes sense to use the confidence of the SVM prediction to try the "most confident" corners first. There are other optimizations and heuristics to try here for ordering that search, but we didn't have time to implement any of these methods.

It may also be possible that this algorithm is useful just not in this particular context. It might be more suited to eliminating options rather than to find the optimal one.

Using the simple heuristic (ordering by SVM confidence), we were able to identify a cube in simple cube images (see figure 7).

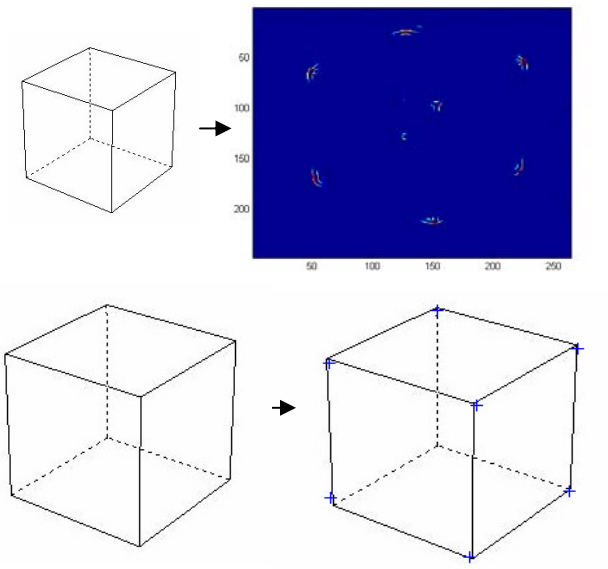


Figure 7. (Top row) Shows the result of running simple heuristics on the given cube before finally choosing the top 7. (Bottom row). Shows the results of model fitting to the top 6 or 7 points.

The Future

One idea we had and experimented with was to train multiple SVMs to identify the different corners of a cube. This would have to be done in the general object case. (For example, using this framework to detect human faces, one would train a nose SVM, a right eye SVM, a mouth SVM, etc…). This drastically reduces the number of permutations of image locations to try in the model fitting step of the paradigm. We lacked enough training data to create robust SVMs for each type of corner, we would have needed several hundred more images of cubes. The other problem, (particular to the cube) is that every "principle-part" is the same as every other one, just rotated in 3 dimensions.

"Tiering" is another idea to for future research to pursue. Once the method in this paper is perfected to recognize small objects, we treat collections of objects in the same fashion to recognize more complex, compound objects, such as cities, freeways, or airplane formations. By constructing, say, a model of a collection of cubes to represent a city.

Finally, the weakness of the SVM is that it is a binary classifier. Ultimately, to create a machine capable of identifying all objects, one would like to not enter image patches into thousands of various SVMs. Ideally one could create an n-object-classifier, whose running time did not vary with n. This might improve detection of every kind of object, cube corners would be better defined in contrast to all other objects than with simply "cube-corners" and "non-cube-corners".

Acknowledgements

We'd like to acknowledge Geremy Heitz for initial guidance, Carl Erickson for mathematical assistance, and Thorsten Joachims for SVM_light. And, of course, Andrew Ng for wonderful instruction that made this possible.

References

Cornell Box, The. *Cornell University Program of Computer Graphics.* 15 Dec, 2006. <http://www.graphics.cornell.edu/online/box/>

Corner Detection. *Wikipedia.* 15 Dec, 2006. <http://en.wikipedia.org/wiki/Corner_detection>

Derpanis, Konstantinos G. The Harris Corner Detector. 15 Dec, 2006. <www.cse.yorku.ca/~kosta/CompVis_Notes/harris_detector.pdf>