A Framework for Learning Visual Discrimination

Justus H. Piater and **Roderic A. Grupen**

Computer Science Department
University of Massachusetts
Amherst, MA 01003
{piater|grupen}@cs.umass.edu

Abstract

A visual system that interacts with the real world must be able to adapt to unexpected situations and to flexibly discover relevant visual cues. We present a method that allows incremental learning of discriminative features. The feature space includes juxtapositions of k oriented local pieces of edge (edgels) and is parameterized by k and the relative angles and distances between the edgels. Specific features are learned by sampling candidate features from this space, increasing k as needed, and retaining discriminative features. For recognition of an unknown scene or object, features are queried one by one. As a result of each query, zero or more candidate object classes are ruled out that do not exhibit this feature to a sufficient degree. We discuss issues of computational complexity, and present experimental results on two databases of geometric objects.

Introduction

A highly desirable ingredient of an artificial system that interacts with the real world is its ability to handle unexpected situations, adapt to them, and learn to act appropriately. Its visual system should be capable of incrementally learning to recognize and search for new objects and scenes. While visual recognition has received enormous attention in computer vision and artificial intelligence, most work has concentrated on specialized systems designed to solve predefined tasks. This can be attributed to the tremendous difficulty of the general problem. For example, in typical leading feature-based object recognition systems, both the object database and the feature set are predefined (Mel 1997; Schiele & Crowley 1996). Most methods require a complete set of training images to be available at the learning stage (Turk & Pentland 1991; Murase & Nayar 1995).

Our work is motivated by the idea of an artificial sensorimotor system situated in the real world. Like a biological system, it has some built-in capabilities as well as methods for learning and exploration. Through interaction with the world, it perpetually learns and improves visual (and other) capabilities. Hence, we desire a system that can incrementally learn about new objects and visual features. Categories should be developed as a result of their empirical significance, rather than as an artifact of the discriminatory properties of the features employed for recognition. The system should be able to find features that discriminate between categories or individual objects wherever this is important.

These desired properties call for a very large feature space, expressive enough to discriminate at various levels of detail. It is not generally feasible to make optimal use of very large feature sets. Instead, we employ the following general (suboptimal) strategy (Amit & Geman 1997; Amit, Geman, & Wilder 1997):

- Impose a *partial order* on the feature space that categorizes the features into various levels of structural complexity. The underlying assumption is that structurally simple features are easier to discover and have less discriminative potential than complicated features, but are still useful for some aspects of the learning problem.
- Because exhaustive search in feature space is prohibitive, *sample* features from the feature space, beginning at the lowest level of complexity, and consider more sophisticated features as required.

An obvious way to generate such a large and partially ordered feature space is through combinatorics: Simple features can be composed in various ways to yield arbitrarily complex features. The specific features we are currently using are introduced in the following section. In the remainder of the paper, we present the recognition algorithm and the feature learning mechanism, and discuss some experimental results.

Features

Our feature set is formed by combinations of oriented edge segments (Fig. 1), often called *edgels*. A feature consists of at least two such edgels, and is defined by their relative orientations θ and the distances d between them.

This feature set has the desirable property of invariance with respect to in-plane rotation. Furthermore, no explicit contour extraction or segmentation need to be performed, which avoids two general problems that are very difficult to solve robustly. Since our features do not rely on contiguous edges, they are expected to be relatively robust with respect to various kinds of image degradation. On the downside, the features are not scale invariant. One way to achieve scale invariance is the introduction of a multiresolution scheme (Piater & Grupen 1999).

Steerable filters (Freeman & Adelson 1991) are employed to compute the orientation of image points efficiently. First,

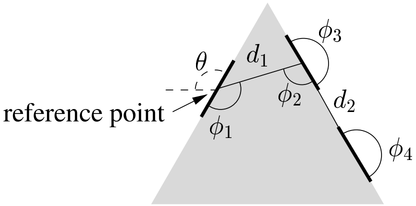

Figure 1: A combinatorial feature of order 3 (composed of three oriented edge segments). The feature is defined by the angles ϕ and the distances d, and the orientation of this specific instance is denoted by θ.

an image $I(i,j)$ is convolved with two basis kernels $G_1^0 = \frac{\partial}{\partial x}G(x,y)$ and $G_1^{\pi/2} = \frac{\partial}{\partial y}G(x,y)$, where $G(x,y)$ is a 2-D Gaussian. These directional derivative-of-Gaussian kernels respond most strongly to vertical and horizontal step edges, respectively. The convolutions yield the edge images $G_1^0(i,j)$ and $G_1^{\pi/2}(i,j)$. By the steerability property of the Gaussian-derivative kernels, the orientation θ of an image point (i,j) is given by

$$\tan\theta(i,j) = \frac{G_1^{\pi/2}(i,j)}{G_1^0(i,j)}, \tag{1}$$

and the corresponding edge *strength* is

$$G_1^\theta(i,j) = \sqrt{G_1^0(i,j)^2 + G_1^{\pi/2}(i,j)^2}. \tag{2}$$

Furthermore, the response (strength) of an oriented first-derivative Gaussian kernel at an arbitrary orientation θ can be computed as

$$G_1^\theta(i,j) = G_1^0(i,j)\cos\theta + G_1^{\pi/2}(i,j)\sin\theta. \tag{3}$$

While we are only using oriented edges, this framework can incorporate any other rotation-invariant spatial primitive (Piater & Grupen 1999). For example, multi-scale steerable vectors of Gaussian derivatives of various orders have been shown to be useful for appearance-based recognition (Rao & Ballard 1995). Other cues can also be included, e.g. three-dimensional features obtained from stereo processing, or temporal signatures such as motion patterns.

Specific features are selected by a learning procedure described below, and are retained for recognition purposes. The precise feature selection and recognition mechanisms are not critical; any of a variety of algorithms can be employed. However, it is important that the features are learned with respect to the specific recognition mechanism with which they are to be used. Our simplified learning/recognition system serves to illustrate the cooperation between these two components. Since the learning procedure builds on the recognition procedure, the latter is described first.

Recognition

The idea underlying our recognition procedure is that individual features provide various degrees of evidence in favor of some classes, and against some others. As a design choice, only the presence of a feature in an unknown image is considered evidence, not its absence. This should serve to increase robustness with respect to partial occlusion. On the other hand, in the absence of other means this implies that the system cannot discern certain classes: Consider a case where all objects of class A possess all features also exhibited by objects of a class B, plus more. Then, given an object that fits the description of class B, the system is never able to rule out class A.

We assume here that the learning procedure has provided a set of features for recognition, and a set of example images which are identical to or a subset of the training images. The recognition procedure queries individual features in sequence, and maintains a list of candidate classes. A query of a feature either serves to rule out one or more candidate classes, or leaves the candidate list unaltered. The goal is to reduce the list of candidates to length one.

To find the best feature to query, we employ the generalized Kolmogorov-Smirnoff distance (KSD) proposed by Utgoff and Clouse (1996) , who demonstrated its competitiveness with the best known decision tree metrics. Given a variable, it returns a cutpoint that maximizes the difference between the class-conditional cumulative distributions of this variable. The variable to be queried is the one that maximizes this metric among all variables.

Here, the variable associated with a feature is the maximum *strength* of this feature in an image. Computation of the maximum strength involves applying the feature operator and measuring its strength, at each location in the image. To apply a feature operator f of order o at location (i_1, j_1), the local orientation $\theta(i_1, j_1)$ is first computed using Eqn. 1. We now have the location and orientation of the reference point of the feature (cf. Fig. 1). Then, the coordinates (i_k, j_k) and orientations θ_k of the other points of this feature are computed for $k = 2, \ldots, o$ using the ϕ and d values of this feature. The strength s_f is then given by the product of the individual Gaussian-derivative responses at the desired locations and orientations:

$$s_f(i,j) = \prod_{k=1}^{o} G_1^{\theta_k}(i_k, j_k)$$

The terms $G_1^{\theta_k}$ are efficiently computed using Eqn. 3. Some tolerance in the relative angles ϕ is granted through the smooth and wide peaks of the G_1^θ (a sum of two sinusoids), and some tolerance in the distance d is provided by the size of the Gaussian kernels which does not require alignment at pixel accuracy.

The computation of $s_f^{\max} = \max_{(i,j)} s_f(i,j)$ is linear in the number of pixels in an image times the order of f. The latter is generally small, can in practice be bounded above, and can thus be assumed to be constant. The efficiency can be increased by a large constant if s_f is computed only for a small subset of *salient* points in the image, rather than exhaustively. Salient points are those with a high edge strength (given by Eqn. 2). The choice of the precise saliency function is not critical. The $s_f^{\max}$ need only be computed once for each image and can then be stored for future use.

Given a subset C of candidate classes, the best feature to query is one that maximizes the KSD among the example images of these classes. Once the best feature f^* and the corresponding cutpoint c^* are identified, the strength $s_{f^*}^{\max}$ is computed for the queried image. If $s_{f^*}^{\max} \leq c^*$, then the feature cannot be asserted, and the list of candidate classes is left unaltered in accordance with the possibility that the feature f^* might actually be present but occluded in this image. In this case, the next-best feature as ranked by the Kolmogorov-Smirnoff metric is queried. If $s_{f^*}^{\max} > c^*$, then we identify all classes for which the majority of all example images have $s_{f^*}^{\max} \leq c^*$, and remove them from the list of candidate classes.

Notice that while up to here our procedure was analogous to conventional decision trees, this last step constitutes a major simplification. In decision trees, candidate classes are allowed to be split across the current cutpoint, whereas we maintain them in their entirety, based on their majority. This assumes that the class-conditional densities are unimodal, which is not generally true in practice. On the other hand, this procedure effectively avoids overfitting which in the case of a potentially infinite, dynamic feature set is much more critical than in typical classification problems involving a fixed set of features: In our problem, almost any two training images can be discerned by some feature, which would have a devastating effect on the generalization properties of the resulting classifier. Rather, we want to force our system to learn better features for which the assumption of unimodality is as close to true as possible.

This procedure is iterated until one of the following situations occurs: (1) There is only one candidate class left, which is then returned as the classification, (2) there is no candidate left, which means the system is totally unable to make any statement about the classification, or (3) the feature set is exhausted. In the latter case, the remaining list of candidate classes is returned as possible classifications. The entire recognition procedure is summarized in Tab. 1.

Note that all KSDs and decision thresholds can in principle be precomputed, which allows the construction of a special type of decision tree. In this case, the expected time taken to recognize an unknown image is logarithmic in the number of classes, and does not directly depend on the number of features or the number of stored example images.

1. $C := \{\text{all classes}\}$
2. Rank features by maximum KSD with respect to C.
3. $f^* := \text{best}$ feature, c^* the corresponding cutpoint.
4. While $s_{f^*}^{\max}(I) \leq c^*$, assign $f^* := \text{next}$ best feature, c^* the corresponding cutpoint. If no features left, return C.
5. $C := C \setminus \{$ all classes k for which the majority of all example images I_i has $s_{f^*}^{\max}(I_i) \leq c^* \}$
6. If $|C| > 1$, go to step 2.
7. Return C (possibly empty).

Table 1: The procedure for recognizing a novel image I.

Feature Learning

The goal of feature learning is to accumulate a set of features useful for discrimination among visual patterns. At the outset, this set is empty. Images are presented to the system one by one. Upon presentation of a novel training image I, the image is first run through the recognition procedure described previously. If it is recognized correctly, this training image results in no change. If it is misclassified, it is added to the set of *example images*, and the $s_f^{\max}(I)$ are computed for all features f. It is then run through the recognition procedure again because some KSDs may have changed to our advantage. If it is again misclassified, it is attempted to learn a new feature.

What are the properties required of the new feature? We note that a classification can fail for one of two reasons: Either the correct class is ruled out at some stage during the recognition process, or the system runs out of suitable features and returns a set of possible class labels which contains the correct one.

In the first case, we want to find a feature f_{new} that gets chosen in place of the feature f_{old} that previously ruled out the true class. Thus, the KSD achieved by f_{new}, $\mathsf{ksd}(f_{\text{new}})$, needs to be greater than $\mathsf{ksd}(f_{\text{old}})$ among the subset C of all classes still under consideration at this stage in the recognition process.

In the second case, C is the set of classes returned by the recognition procedure, and $\mathsf{ksd}(f_{\text{old}})$ is taken to be zero. In both cases, the feature must be present in image I to a degree stronger than the cutpoint associated with f_{new}, i.e. $s_{f_{\text{new}}}^{\max}(I) > c_{\text{new}}$.

It is now attempted to find such an f_{new} by randomly sampling features from image I. This sampling proceeds in stages: First, some number of new order-2 features are generated by randomly choosing pairs of points from among the salient pixels in I, and computing the two angles ϕ_i (using Eqn. 1) and the distance d. The saliency function may be the same as that used in the recognition procedure. To keep the features local, the distance between the two points is limited. Next, all existing features (i.e. those previously learned and those just sampled) are augmented to the next order. This is done by sampling a new reference point (again within a certain distance) and noting the resulting ϕ_i and d with respect to the reference point of the parent feature.

The augmentation step can be repeated several times. The sampling process is terminated once a feature f_{new} achieves $\mathsf{ksd}(f_{\text{new}}) > \mathsf{ksd}(f_{\text{old}})$ and $s_{f_{\text{new}}}^{\max}(I) > c_{\text{new}}$, or after a maximum number of augmentation steps is completed without success.

If a suitable feature is found, it is added to the set, and the current training image is again run through the recognition procedure. The properties of the new feature guarantee that either of the following occurs: (1) The new feature is chosen at the stage that previously failed during the recognition process, and the correct class is not ruled out at this stage; or (2) it is chosen at some earlier stage during the recognition process. If the recognition fails again, the feature sampling process iterates. The feature learning procedure is summarized in Tab. 2.

For a brief look at the time complexity, first note that the feature augmentation process involves iterating over all preexisting features and newly sampled candidate order-2 features, say n_f in total. Computation of $\mathsf{ksd}(f_{\text{new}})$ requires

1. If I is recognized correctly, stop.
2. Add I to the example images and compute the $s_f^{\max}(I)$.
3. If it is recognized correctly, stop.
   Else, note C and $\mathsf{ksd}(f_{\text{old}})$ at the failing recognition step.
4. Generate a candidate f_{new} by sampling or augmentation.
5. If $\mathsf{ksd}(f_{\text{new}}) > \mathsf{ksd}(f_{\text{old}})$ and $s_{f_{\text{new}}}^{\max} > c_{\text{new}}$, add f_{new} to the set and go to step 3.
6. If the maximum number of new sample features is reached, stop; else go to step 4.

Table 2: The procedure for learning a novel image I.

processing each example image in each class under consideration, which on average is proportional to the total number n_I of accumulated example images. Therefore, learning one new feature has a time complexity on the order of $2^a n_f n_I$, where a is a small constant giving the maximum number of augmentation steps. Since the number of pre-existing features is directly related to the number n_I of accumulated example images, finding one new feature takes time proportional to n_I^2. Clearly this is not acceptable for large-scale recognition problems. Feature selection has long been established as a difficult combinatorial problem (Ferri *et al.* 1994). Similarly to other work in feature learning, we need to identify suitable heuristics for reducing both factors n_f and n_I in the complexity term.

Experimental Results

To illustrate the operation of our system, we trained it on two simple supervised object recognition tasks. In each case, the database contained non-occluded, clutter-free example views of simple geometric objects (Fig. 2). In one task, the database consisted of eight synthetic objects, each of which was rendered in high quality at 15 different views. For the other task, low-quality images were taken of real geometric objects. There were 18 views of a sphere, 19 views of a cone in various positions, and 16 views of a cube. The images of the class "sphere" included spheres of two different sizes, and the images of the class "cube" contained two cubes that differed in size.

The learning system was trained on each task as described above. The images of the training set were iteratively presented to the system in random order, until either the system had learned the training set perfectly, or until no feature was found during an entire cycle through the training set even though there were some misclassifications. To learn a new feature, first up to 10 new order-2 features were sampled. Then, the set of all pre-existing and new candidate features was augmented in up to two iterations.

Tables 3 and 4 show the results obtained by a 10-fold cross-validation procedure. In all test cases, the recognition procedure returned a single class label. In the synthetic-object task, the most common error was to label a "cyl6" as a "tub6". This is not surprising since a "cyl6" does not exhibit any edge features that a "tub6" does not possess. As noted earlier, this type of confusion is to be expected by the design of the system. A similar situation occurs in the real-object task: The most common error was to mistake a sphere for a cone. Again, spheres do not exhibit edge features that

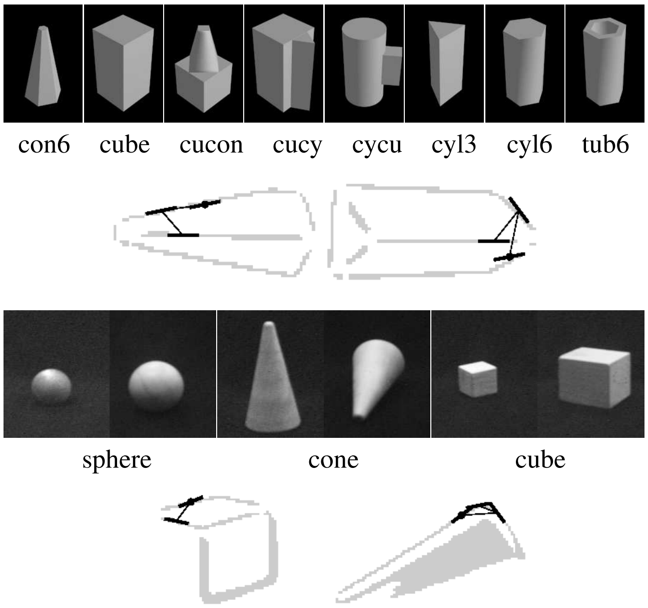

Figure 2: The synthetic and real-object tasks: Example views and examples of features learned.

	classification results:								sums:
	con6	cube	cucon	cucy	cycu	cyl3	cyl6	tub6	
con6	15								15
cube		14		1					15
cucon			15						15
cucy		1	3	11					15
cycu				1	14				15
cyl3						15			15
cyl6						1	10	4	15
tub6								15	15
sums:	15	15	18	13	14	16	10	19	120

Table 3: Confusion matrix summarizing the cross-validated test-set performance on the synthetic-object task. The overall proportion of correct recognitions was 0.91.

distinguish it from a cone lying down, revealing its circular base in profile (see Fig. 2). The real-object task was relatively hard due to the two different sizes of spheres. Without multiscale processing, it is not obvious how to find features that are sufficiently scale-invariant. Nevertheless, both tasks were learned relatively well.

Figure 2 includes some examples of order-2 and order-3 features found during learning. The gray shaded areas indicate the salient points used for sampling new features. The majority of all learned features were of order 2, with an appreciable number of order-3 and occasional order-4 or order-5 features. Due to the randomness of the algorithm and differing characteristics of the training images, the number of features learned, the number of example images retained, and the number of iterations through the training set varied considerably between the individual folds of the cross-validation procedures, as detailed in Tab. 5. In all cases, only about half of the features that were learned at some stage during the training process were in the end actually consulted

	classification results:			sums:
	sphere	cube	cone	
sphere	15		3	18
cube		14	2	16
cone			19	19
sums:	15	14	24	53

Table 4: Confusion matrix summarizing the cross-validated test-set performance on the real-object task. The overall proportion of correct recognitions was 0.91.

Fold:	1	2	3	4	5	6	7	8	9	10
Synthetic Objects:										
# iter.:	3	3	***5***	4	***5***	***4***	***5***	***6***	***4***	***5***
# imgs.:	43	40	47	42	40	41	54	43	38	40
# feats.:	24	19	27	19	23	18	30	22	22	16
Accur.:	.88	1.0	.81	.88	.81	1.0	.88	1.0	1.0	1.0
Real Objects:										
# iter.:	***6***	***6***	4	5	4	3	3	4	10	5
# imgs.:	21	26	24	22	18	26	20	20	32	17
# feats.:	13	12	12	11	9	15	13	11	26	9
Accur.:	1.0	1.0	1.0	1.0	1.0	.83	.80	.60	.75	1.0

Table 5: Characteristics of the synthetic- and real-object tasks. Shown are the numbers of iterations through the training set, the number of misclassified example images stored, the number of features learned, and the test-set accuracy achieved. Emphasized numbers indicate that the training set was not learned perfectly.

on some training or test image. In other words, the other half had been superceded by some better feature at later stages of training.

Conclusions and Future Work

We presented a framework for visual recognition based on a combinatorial feature space of potentially infinite size. While our feature primitives currently consist of oriented edges, the framework can in principle accommodate any type of local feature. The resulting compound features are invariant to in-plane rotations as long as the feature primitives are. Certain texture signatures are one important example. Extension of the framework to achieve scale invariance is straightforward by processing images at multiple resolutions.

A partial simple-to-complex ordering of the feature space permits feature search in polynomial time. While simple-to-complex feature sampling is not generally optimal with respect to any meaningful objective, this heuristic is intuitively pleasing in that it prefers simplicity over complexity. Assuming that most distinctions between object classes can be expressed in terms of low-order features according to our definition, simple-to-complex sampling expends most effort in those areas of the feature space where success is most likely to occur.

Even so, the current feature learning method does not scale well to larger-scale applications because each newly sampled candidate feature must be evaluated on every stored example image. Therefore, one goal of further research is to reduce the number of example images that need to be stored. Moreover, new feature points are currently chosen purely at random. The identification of more focused search methods would lead to tremendous improvements in performance.

The current recognition procedure can operate in time logarithmic in the number of classes. At the cost of increasing the time complexity, it is straightforward to extend the system to recognize multiple objects or to offer several weighted candidate classifications for a single object. One way to do this is to pursue all usable features at each decision, not just the best one, and follow each branch. However, our specific recognition procedure is not an indispensable component of the system. Non-sequential associative mechanisms, for example, constitute an attractive alternative.

Acknowledgments

This research was supported in part by the National Science Foundation under grants CISE/CDA-9703217, IRI-9704530, and IRI-9503687. The database of synthetic objects was provided by Levente Tóth at http://www.cis.plym.ac.uk/cis/levi/UoP_CIS_3D_Archive/8obj_set.tar.

References

Amit, Y., and Geman, D. 1997. Shape quantization and recognition with randomized trees. *Neural Computation* 9:1545–1588.

Amit, Y.; Geman, D.; and Wilder, K. 1997. Joint induction of shape features and tree classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.* 19:1300–1305.

Ferri, J. J.; Pudil, P.; Hatef, M.; and Kittler, J. 1994. Comparative study of techniques for large-scale feature selection. In Gelsema, E. S., and Kanal, L. N., eds., *Pattern Recognition in Practice IV*, 403–413. Elsevier Science B.V.

Freeman, W. T., and Adelson, E. H. 1991. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Machine Intell.* 13(9):891–906.

Mel, B. W. 1997. Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation* 9:777–804.

Murase, H., and Nayar, S. K. 1995. Visual learning and recognition of 3-D objects from appearance. *Int. J. Computer Vision* 14:5–24.

Piater, J. H., and Grupen, R. A. 1999. Toward learning visual discrimination strategies. Computer Science Technical Report 99-01, University of Massachusetts, Amherst, MA.

Rao, R. P. N., and Ballard, D. H. 1995. An active vision architecture based on iconic representations. *Artificial Intelligence* 78:461–505.

Schiele, B., and Crowley, J. L. 1996. Object recognition using multidimensional receptive field histograms. In *ECCV'96, Fourth Europ. Conf. on Computer Vision*.

Turk, M., and Pentland, A. 1991. Eigenfaces for recognition. *Journal of Cognitive Neuroscience* 3(1):71–86.

Utgoff, P. E., and Clouse, J. A. 1996. A Kolmogorov-Smirnoff metric for decision tree induction. Computer Science Technical Report 96-3, University of Massachusetts, Amherst, MA.