Visual Motion Based Behavior Learning Using Hierarchical Discriminant Regression

Changjiang Yang and Juyang Weng
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
Email: { yangcha1, weng}@pilot.msu.edu

Abstract

This paper presents a new technique which incrementally builds a hierarchical discriminant regression (IHDR) tree for generation of motion based robot reactions. The robot learned the desired reactions from motion change images, without using other pre-defined features. The generation from training cases is accomplished through the automatically constructed IHDR tree, which automatically derives features that are most related to outputs and disregards subspaces that are not related, or less related, to outputs. The real-time speed is achieved through combination of feature space partition and a coarse-to-fine sample search, which results in a logarithmic time complexity in the number of nodes. The experiments showed that the IHDR method can interpolate the mapping between high dimensional input space and the output control signal space from a variety of objects of various shapes with different motion patterns, based on the size-dependent negative logarithmic distance measures in the hierarchical feature space. The trained robot is able to aim to its camera towards moving object and move toward or away according to the size of moving object.

key words: Motion detection, discriminant analysis, decision trees, incremental learning.

1 Introduction

Information about visual motion is useful for many vision-based applications, such as video surveillance [4, 10, 6], traffic monitoring [3], human detection and tracking for video teleconferencing or human-machine interface [13, 8, 2], lip tracking and gesture recognition [11, 9]. Detecting moving objects from the background is commonly realized by either by comparing with a known background model or subtracting consecutive video frames. Once candidate pixels are extracted from the background, a pre-designed object model is applied to them for further processing. There are many successful applications that utilize this approach. These object model based methods require that specific information about object in the scene is known a priori, such as a lip, a head or a human body. The motion detection is further complicated when moving objects occlude some of its components, such as arms, and occlusions occur between multiple objects [7].

In the work presented here, we are interested in looking into possibility of relaxing the requirement for knowing the type of objects in the scene and the availability of an object model. This type of scenario typically involves tasks that do not require detailed analysis of objects, but only require global processing of motion cues available. For example, a mobile robot needs to detect moving object in the scene. If the object is constantly moving, the robot needs to continuously adjust the aim direction of its camera so that the moving object stays roughly at the center of the image frames. The robot also needs to adjust its distance to the moving object so that the size of moving object is neither too small nor too large. These requirements arises from our intended application where the robot needs to detect moving objects, adjust the aim and distance, and then take a picture for further analysis by humans at a later time. In this application, we do not know what objects may appear. Therefore, we cannot require a pre-designed object model.

In this paper, we present a learning method for this type of applications. Once the moving objects are detected, the robot aims camera towards the moving objects and adjusts its distance to the objects in real-time. One advantage of this method is that it can deal with unknown and complex moving shape. Because its generalization is based on statistics using supervised learning, the method can learn arbitrary behaviors without programming for each individually. With the tree structure, the method can generate real-time control signals and is scalable to a large number of training cases. We found that the effort spent for training is minimal compared with straight programming. An overhead associated with learning is the space to store experience as IHDR tree, but it is well within the capacity if current PCs. In our experiments, a short time context is used to smooth control signals in time, while longer context is necessary for other applications such as hand sign recognition [1].

In this method, rich information in the motion images can be preserved by treating motion pixel images as a high dimensional input vector, where each pixel corresponds to a dimension of the vector. Due to the large number of deformation variations that exist in the motion pixels, building a classifier corresponds to build an image information database. Therefore, the classifier must address three issues: fast retrieval, accurate performance, and the capability to incrementally update representation.

To achieve fast retrieval in the database of the motion images, the representation and the organization of the image database are the key components. The other issue is how to organize the database. Linear search is very time-consuming which makes it impractical. One way to solve this problem is to use a decision tree. Decision trees organize the data in a hierarchical structure so that retrieval time can be logarithmic, which is essential for motion detection.

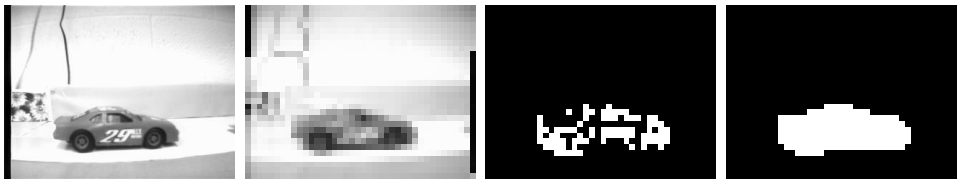
We present an incremental way of constructing a decision tree that uses discriminant analysis at each internal node to detect the moving objects. It is well known that the performance of a classification problem depends on the training data set. It is desirable to incorporate the incremental learning capability to a classifier. The classifier can adapt to new training data without re-computing the statistics from the whole image database. Another advantage of incremental training is that an incremental method allows the trainer to interleave training and testing, which enables the human trainer to train the system with "weak cases", potentially reducing the database size and cost of training process.

This paper is organized as follows. In Section 2, we describe our method to detect the moving objects. The IHDR used in our method is outlined in Section 3. Experimental results of simulated as well as real image sequences and real time implementation are presented in Section 4. Section 5 contains the conclusion and discussion.

2 The Method

The task to be performed by the robot is closely related to visual attention selection. Moving patterns in the scene may be very complex. For example, a human body may involve several individually moving limbs, while each limb may not necessarily be completely detected at all time. In order to keep shape information, we do not simply compute the center of the moving pixels. Instead, a fast filling process is used to fill the pixels from the leftmost moving pixels to the rightmost one in each pixel row, where each moving pixel is identified if its intensity difference between two consecutive frames is sufficiently large by passing a threshold. The filling process is necessary since internal region of a moving object often do not exhibit large intensity changes.

The filled image is a binary image. A pixel takes a value 255 if the intensity change is large originally or it is between two such pixels in the same row (or column, but we chose not to do columnwise filling to save the processing time). Otherwise, the pixel takes zero value. This filled motion image contains shape information about the moving patterns. If multiple moving components are presented, such as the case of a moving human body, the filled motion image tends to give a single blob, but there is no guarantee that this blob is connected even if both rowwise and columnwise fillings are carried out. This can be seen by considering two blobs, one at the upper left corner and one at the lower right corner. Fig. 1 shows an example which illustrates the preprocessing procedure of motion images.



(a) (b) (c) (d)

Figure 1: An example illustrating the preprocessing procedure of motion detection. (a) the photo of the moving car, (b) one frame from the image sequence, (c) the difference binary image by subtracting two consecutive frames and thresholding, (d) the filled motion image.

The next question is how to use this filled motion image. We avoid doing explicit analysis of the shape and position of the blobs, since such an analysis requires some parametric shape models, which cannot apply to all unpredictable shapes that may appear. Instead, we apply statistical methods directly to the filled motion image. A vector representation of image with r rows and c columns is a vector of dimension rc, each component in the vector is equal to the intensity of the corresponding pixel value. In other words, every pixel is mapped to a unique component in the vector. The actual mapping is immaterial but is fixed once it is selected. We use rowwise mapping: from left to right then from top to bottom. This currently popular representation allows us to effectively consider not only statistics of nearby pixels, but also far away pixels. A feature vector in this vector space can be represented by an image, which characterizes position, shape and intensity of the the feature pattern, no matter how complex it is. An example such a feature is the well known eigenface in face recognition [12]. This feature representation is not translational invariant. In our application, translational invariance is not what we want since the position of a detected object is essential for determining the aim direction.

Therefore, the problem of robot action generation from images is converted into action generation from filled motion images. Each filled motion image is itself an image. A major

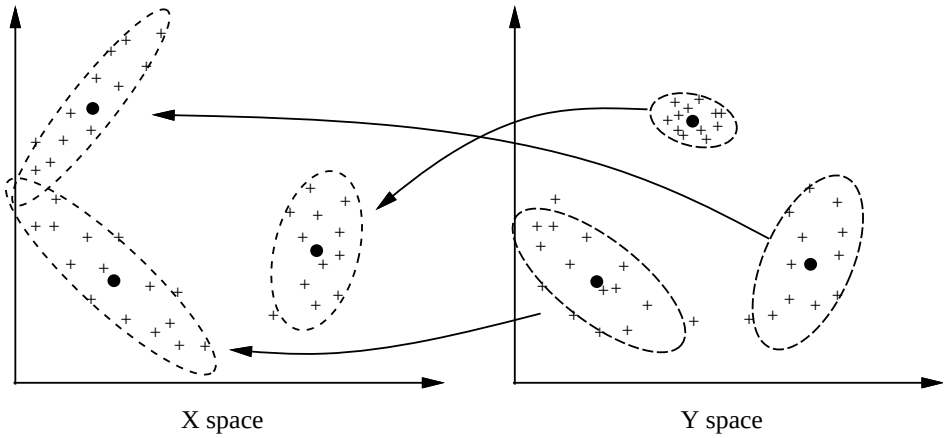


Figure 2: Y-clusters in space $\mathcal{Y}$ and the corresponding x-clusters in space $\mathcal{X}$. The first and the second order statistics are updated for each cluster. By default, the normalized Mahalanobis distance is used for x-cluster and the Euclidean distance is used for distance to y-cluster.

difference between a filled motion image and an original electro-optical image is that the former disregard some intensity information that is not directly related to object motion.

3 Mapping

Our problem is converted into the following one: approximating a mapping $h : \mathcal{X} \mapsto \mathcal{Y}$ from a set of training samples $\{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$, where y_i is a numerical output. We outline our hierarchical discriminating regression method here. More detail is available in [5].

Our goal is to automatically generate discriminant features although no class label is available (other than the numerical vectors in space $\mathcal{Y}$). We must process each sample (x_i, y_i) to update the HDR tree using only a minimal amount of computation.

Two types of clusters are formed at each node of the HDR tree — y-clusters and x-clusters. as shown in Fig. 2. The y-clusters are clusters in the output space $\mathcal{Y}$ and x-clusters are those in the input space $\mathcal{X}$. There are a maximum of q (e.g., $q = 6$) clusters of each type at each node. The q y-clusters determine the virtual class label of each sample (x, y) based on its y part. The virtual class label is used to determine which x-cluster the input sample (x, y) should update using its x part. Each x-cluster approximates the sample population in $\mathcal{X}$ space for the samples that belong to it. It may spawn a child node from the current node if finer approximation is required. For computational efficiency, none of the x-clusters and y-clusters keep actual input samples, unlike the traditional clustering methods. Only the first orders of

statistics are used to represent the clusters. For example, each y-cluster keeps the mean vector and each x-cluster keeps the mean vector and the full covariance matrix in an efficient form. The algorithm recursively builds an HDR tree from a set of training samples. The deeper a node in the tree, the smaller the variances of its x-clusters are. The following is the outline of the algorithm for tree building and retrieval.

Procedure 1 `BuildSubtree`*: Given a node N and a subset S' of the training samples in $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$, recursively build an subtree which roots from the node N. At most q clusters are allowed in one node.*

1. *Let p be the number of the clusters in node N.*
   - *Call* `Clustering-Y` *(procedure 2) to obtain p y-clusters.*
   - *If y_i belongs to i-th y-cluster, then x_i belongs to i-th x-cluster.*
2. *Compute the mean and covariance matrix of each x-cluster.*
3. *For every x_i of (x_i, y_i) in S':*
   - *Find the nearest x-cluster j according to the probability-based distances.*
   - *Place sample (x_i, y_i) in the cluster j.*
4. *For each cluster j, we have a portion of samples S_j that belong to the x-cluster. If the largest Euclidean distance among y_i's in the x-cluster is larger than threshold δ_y, a child node N_j of N is created from the x-cluster and this procedure is recursively repeated with input samples S_j and node N_j. The number δ_y represents the sensitivity of the HDR tree in $\mathcal{Y}$ space.*

Procedure 2 `Clustering-Y`*: Given a set of output vectors $Y = (y_1, y_2, \ldots, y_n)$, return p y-clusters. $p \leq q$, where q represents the maximum clusters allowed in one node.*

1. *Let the mean Y_1 of y-cluster 1 be y_1. Set $p = 1$ and $i = 2$.*
2. *For i from 2 to n do*
   (a) *Find the nearest y-cluster j for y_i.*
   (b) *Compute the Euclidean distance $d = dist(y_i, Y_j)$.*
   (c) *If $d \geq \delta_y$ and $p < q$, let the mean Y_{p+1} of y-cluster $p + 1$ be y_i. Set $p = p + 1$.*
   (d) *Otherwise, update y-cluster j by using new member y_i.*

The procedure to create an HDR tree just calls procedure `BuildSubtree` with root R and all the training samples $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$. The procedure for query the HDR tree for an unknown sample x is described in below.

Procedure 3 `Retrieval`*: Given an HDR tree T and sample x, estimate the corresponding output vector y. Parameter k specifies the upper bound in the width of parallel tree search.*

1. *From the root of the tree, compute the probability-based distance to every cluster in the node. Select at most top k x-clusters which have the smallest probability-based distances to x. These x-clusters are called active x-clusters.*

2. *For every active cluster received, check if it points to a child node. If it does, mark it inactive and explore its child node by computing the probability-based distances of x-clusters in the child node. At most k^2 active x-clusters can be returned.*

3. *Mark at most k active x-clusters according to the smallest probability-based distances.*

4. *Do the above steps 2 through 3 recursively until all the resulting active x-clusters are all terminal.*

5. *Let the cluster c have the shortest distance among all reached leaf nodes. Output the corresponding mean of its y-cluster as the estimated output y for x.*

As proved in [5], the time for updating or query the tree is $O(\log m)$ where m is the number of leaf nodes in the tree, assuming that the number of samples in each leaf node is bounded above by a constant.

In the above algorithm, we need to estimate the distance for an input vector x to belong to an x-cluster. Each x-cluster is represented by its mean as its center and the covariance matrix. However, since the dimensionality of the space $\mathcal{X}$ is typically very high, it is not practical to directly keep the covariance matrix.

As explained above, each internal node keeps up to q x-clusters. The centers of these q x-clusters are denoted by

$$C = \{c_1, c_2, ..., c_q \mid c_i \in \mathcal{X}, i = 1, 2, ..., q\}. \tag{1}$$

The locations of these q centers tell us the subspace $\mathcal{D}$ in which these q centers lie. $\mathcal{D}$ is a discriminant space since the clusters are formed based on the clusters in space $\mathcal{Y}$. Once we find this discriminant space $\mathcal{D}$, we can compute the probability for $\mathcal{X}$ to belong to each x-cluster, using Gaussian model. Sample size must be used so that when covariance matrix

is not reliable with few samples, simpler distance metrics is used, as explained in [5]. The incrementally built HDR tree is called IHDR tree.

In summary, the complete motion detection algorithm is as follows:

- Step 1: Capture the image sequence. Perform temporal differentiation to get the difference images.
- Step 2: Threshold the output of Step 1 and generate the binary motion images. Fill the gaps between the pixels along rows to get the filled motion images.
- Step 3: Input the filled binary motion images and corresponding action vector to update the IHDR tree.
- Step 4: Given each new filled motion image, perform the motion detection by retrieve the action from IHDR tree.

Since the IHDR is generated incrementally, we can feed training samples incrementally, until the current tree performs well for all the new cases.

4 Experimental Results

A Khepera robot was used in the experiment. A microcamera was mounted on top of the robot. The filled motion image contains the position, shape and size of the moving object that has been observed. Our visual motion based algorithm accepts an image X at a time and outputs the retrieved control signal C to update the heading direction and at a proper distance so that robot can take a picture of the moving object. In other words, the robot tracks the moving object by turning its body in real time and when object moves away or moves closer, the robot moves ahead or back up accordingly. Human trainer supplies the desired control signals based on the current image captured by the robot. The training process was realized by a approximation function from filled motion image to control signals. This is a very challenging task since the function to be approximated is for a very high dimensional input space and the real application requires the robot to perform in real time.

We applied our algorithm to this challenging problem. A subset of samples are shown in Fig 3. The original images are grayscale intensity images with resolution 30×40. Our method uses the entire images directly and the feature vectors of subspaces are automatically derived by IHDR online instead of manually designed by human off-line. For each moving object, we collected 5000 images with human supplied actions as one of the four primitive actions: move-left, move-right, move-forward, and move-backward. For performance evaluation, these

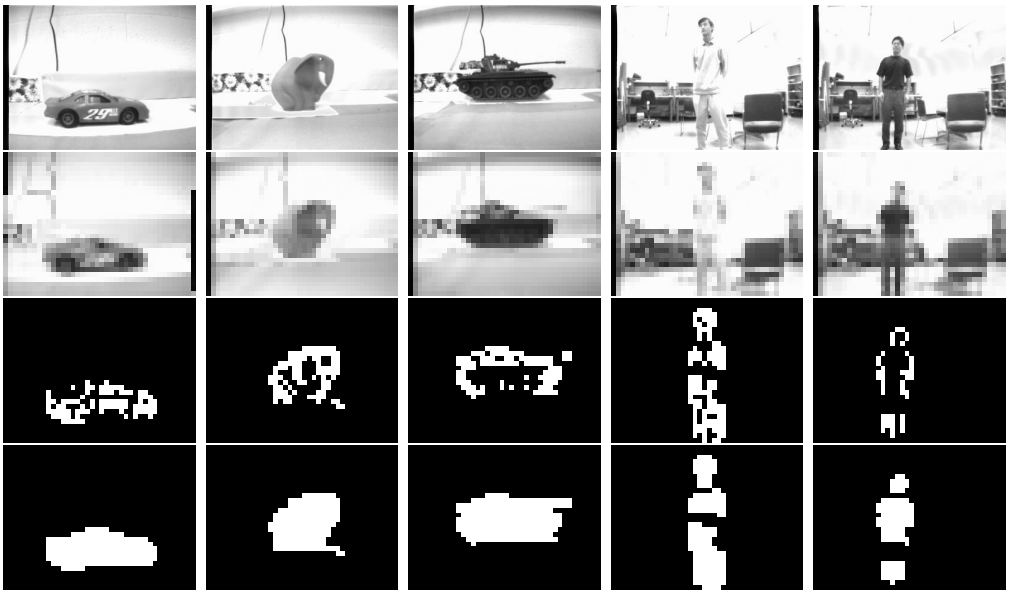


Figure 3: A subset of images used in motion detection problem. The images in the first row show the original views. The images in the second row show the reduced resolution view. The images in the third row show the motion pixel images. The last row shows the filled motion pixel images.

5000 images were divided into 5 groups of equal size for leave-one-out tests. Each test leaves one group out as test images and remaining groups are used as training images. The error rates are average of the leaving-one batch out cross validation error rates. To take into account the possibility of than different moving objects may be presented, we chose 5 objects that are representing of the object shapes that we are interested in, one toy elephant, two vehicles, and two humans. These five objects resulted in many different shapes in the filled motion images, due to the variation in lighting, viewing angle, viewing distance and object deformation. Table 1 shows the error rates for motion detection using five moving objects. Each row represents an IHDR tree trained by the corresponding object. Each column represents the object tested by each of the IHDR tree.

We also performed leave one class out cross-validation to verify our method, which used the four of the five dataset as the training dataset, and the rest one as the test dataset. The performance is shown in Table 2. All our test on Khepera robot has shown that a system of such an error level can control the robot very reliably when judged visually. This is largely because the robot does not have to make correct action every single time in order to display reliable accumulative long term behaviors.

Table 1: Error rates of actions for five moving objects

Training \ Test	car	elephant	tank	human 1	human2
car	0.6%	1.2%	10.4%	11.1%	11.2%
elephant	4.0%	0.7%	9.1%	14.9%	12.4%
tank	15.7%	8.2%	0.5%	10.2%	13.3%
human 1	0.4%	0.7%	2.6%	0.5%	2.5%
human 2	1.2%	4.2%	1.8%	2.3%	0.4%

Table 2: Leave one class out error rates for motion detection

Test dataset	car	elephant	tank	human 1	human2
Error rate	10.0%	16.1%	17.5%	4.9%	8.0%

5 Conclusions

We have presented a learning based method for training a robot to react according to the moving objects. Unlike traditional approaches to motion-based image analysis, we do not assume a known type of objects. Once filled motion images is given, the IHDR method automatically derive features that are important to actions. The highly distributed coarse-to-fine features stored in the tree correspond to an automatically constructed decision system. Our experimental data showed that the IHDR tree can generalize reasonably beyond the object that it has learned, to other different objects.

References

[1] Y. Cui and J. Weng. Hand sign recognition from intensity image sequences with complex backgrounds. In *Proc. 2nd International Conference on Automatic Face- and Gesture-Recognition*, pages 259–264, Killington, Vermont, October 1996.

[2] J. Davis and A. Bobick. The representation and recognition of action using temporal templates. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, 1997.

[3] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1997.

[4] I. Hartitaoglu, D. Harwood, and L. S. Davis. W4: Who? when? where? what? a real-time system for detecting and tracking people. In *Proc of the third IEEE Int'l Conf. Automatic Face and Gesture Recognition*, pages 222–227, 1998.

[5] W. Hwang and J. Weng. Hierarchical discriminating regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, Nov. 2000.

[6] T. Kanade. Advances in cooperative multi-sensor video surveillance. In *Proc. IUW*, pages 3–24, 1998.

[7] S. Khan and M. Shah. Tracking people in presence of occlusion. In *Proc. Asian Conf. Comp. Vision*, pages 1132–1137, 2000.

[8] J. Ohya and et al. Virtual metamorphosis. *IEEE Multimedia*, 6(2):29–39, 1999.

[9] N. Oliver and A. Pentland. Lafter: Lips and face tracking. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, 1997.

[10] P. L. Rosin. Thresholding for change detection. In *Proc. 6th Int'l Conf. Computer Vision*, pages 274–279, 1998.

[11] Y. L. Tian, T. Kanade, and J. F. Cohn. Robust lip tracking by combining shape, color and motion. In *Proc. Asian Conf. Comp. Vision*, pages 1040–1045, Taipei, Taiwan, Jan. 2000.

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

[13] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1998.