Texture-Based Segmentation of Natural Images Using Neural Networks

Banavar Sridhar, Anil Phatak and Gano Chatterji
NASA Ames Research Center
Moffett Field, CA 94035

Abstract The optical flow algorithms provide a sparse set of ranges as a function of azimuth and elevation. A natural way to enhance the range map is by interpolation. However, this should be undertaken with care since interpolation assumes continuity of range. The range is continuous in certain parts of the image and may jump at object boundaries. In such situations, the ability to detect homogeneous object regions by scene segmentation may be used to determine regions in the range map which can be enhanced by interpolation. This paper describes an image segmentation method based on scalar texture measures. A neural net approach for image segmentation based on scalar texture measures is discussed. The generalization of the network approach to subsequent images in the sequence is examined. It is shown that a cascade of neural networks, where each neural network is trained on a single scalar texture measure, can be used for image segmentation. Finally, the segmentation achieved by the various methods is shown for an image sequence.

I. INTRODUCTION

Images from electro-optical sensors provide a covert way of detecting objects in the flight path of a low-flying helicopter. A sequence of images is processed using techniques based on optical flow and recursive estimation to compute object location information, referred to as range map, in the field of view of the sensor[1, 2]. The range map can be used either for the guidance of the helicopter in an automatic system or for display to a pilot. For guidance and display purposes, these discrete set of ranges need to be grouped further into sets which correspond to objects in the real world. The optical flow algorithms provide a sparse set of ranges as a function of azimuth and elevation. Figure 1 represents the first image in a sequence of 80 images. Figure 2 shows a range map computed using the optical flow approach where the white squares represent regions in the image where range information is available. It may be noted that this sparse set corresponds mostly to object boundaries.

A natural way to enhance the range map is by interpolation. However, this should be undertaken with care since interpolation assumes continuity of range. The range is continuous in certain parts of the image and may jump at object boundaries. In such situations, the ability to detect homogeneous object regions by scene segmentation followed by selective probing by an active sensor can be used to fill gaps in the range map.

In the published literature several methods have been proposed for scene segmentation. A brief review of some of the methods is available in [3]. Several scene segmentation methods use matrix or scalar texture features. Research on the use of texture for image segmentation is motivated by the classification problems associated with

Manuscript received August 1, 1992.
neural network to identify transient operating conditions in nuclear power plants. Franklin, Sutton, and Anderson [17] from GTE have used an one-layer neural network, to monitor fluorescent bulb manufacturing. Very little research works have been done in using neural network to monitor the degradation of machines.

**METHODOLOGY**

The goal of this research was to develop an innovative, proactive, maintenance implementation technique -- an integrated learning, monitoring, and recognition methodology that can be used to monitor system behavior adaptively and provide an early warning of possible faults. Figure 1 shows the system concept of the proposed methodology, an integrated learning, monitoring, and recognition technique, for a proactive maintenance system. The cerebellar model articulation controller (CMAC) is used to adaptively learn machine behavior and a pattern discrimination model (PDM) based on the CMAC is used to monitor, recognize, and quantify behavioral changes. The PDM serves as a watchdog to monitor the behavior of the machine by using a confidence value that represents the conditional probability of degradation. The fault then can be detected by comparing the confidence value of the PDM output with a threshold confidence value.

The author proposes to use the CMAC as a real-time data acquisition and learning tool for the behavior of the actuators and sensors when a system is in operation. The unique part of this approach is the use of CMAC to learn the good or normal patterns under different working conditions rather than to learn the bad or wrong patterns. In many cases, it is very difficult to create the bad and wrong conditions for a machine during training.

**CMAC**

The CMAC [18-21] is essentially a clever adaptive table lookup technique for representing complex, nonlinear functions over multi-dimensional, discrete input spaces. It reduces the size of the lookup table through hashing-coding (many-into-few mapping); provides for response generalization and interpolation through a distributed, topographic representation of the inputs; and learns the appropriate nonlinear function through a supervised learning process that adjusts the content or weight of each address in the lookup table.

In the CMAC architecture, each discrete input \( S \) maps to many locations in the memory (one "location" contains a vector of the same dimension as \( P \) and the function \( F(S) \) is computed by summing the values at all of the locations mapped to by \( S \). The set \( S \) of input vectors is mapped onto the random table locations, which are then mapped by hashing onto a much smaller set weight table. This mapping scheme has the advantage of providing automatic interpolation (generalization) between input states in the space \( S \) (similar inputs produce similar outputs).

Briefly speaking, the CMAC can be described as a computing device which accepts an input vector \( S=(S_1, S_2, S_3, \ldots, S_n) \) and produces an output vector \( P=F(S) \). To compute the output vector \( P \) for a given input state \( S \), pair mapping is performed, namely

\[
f: \quad S \rightarrow A
\]

\[
g: \quad A \rightarrow P
\]

where \( A \) is called the association cell vector, which is actually a large table of memory addresses. Given an input vector \( S=(S_1, S_2, S_3, \ldots, S_n) \), the mapping function \( f \) points to some memory addresses (location in the \( A \) table); these locations are among the association cells \( A \) and are referred to as the active association cells. The number of active association cells for any given input is a fixed parameter of the CMAC module selected by the user. \( Weight(s) \), or tabulated values that contribute to the output response, are attached to each association cell.

In CMAC, the values of a set of inputs act as pointers to a table of random numbers. These numbers are hashed to form a set of addresses in a weight table. The values in the weight table pointed to by these addresses are summed to yield an output. Training consists of adjusting the values in the weight table based on the error between their present summation value and the desired output for the particular values of the input that resulted in these addresses.

\[
win = wio + \left[ \left( d - \sum_{j=1}^{n} w_j \right) / n \right]
\]

where
\[ w_{o} = \text{old value of weight at address location} \]
\[ d = \text{desired output value for present input vector} \]
\[ n = \text{number of addressed locations in the weight table} \]
\[ w_{j} = \text{value of weight at address location} \]
\[ m = \text{percentage of correction} \]
\[ w_{n} = \text{new value of weights at address location} \]

**PATTERN DISCRIMINATION MODEL (PDM)**

To perform pattern recognition and discrimination tasks, a pattern discrimination model, based on the confidence checking of the CMAC output, is developed. In Figure 2, a confidence table is used in parallel with the weight table. During the CMAC training, the data stored in the weight table will activate a "1" value in the confidence table during training. These stored memory locations represent the desired or normal behavior of the machine. During the actual operations, the identical behavior will map to the same memory location in a weight table. A confidence value can be obtained by summing the total "1"s and dividing by the number \( N \), which is the size of the address table. It derives a 100 percent confidence value since the same number of "1"s were used for both training and operating conditions. If there is a behavioral change, the change will cause different mapping locations in the weight table. Then, a "0" value will be placed in the confidence table when there is memory location change in the weight table. A confidence value can be obtained by summing the total "1"s and "0"s and dividing by the size of the address table.

Typically, if the parameters are selected properly, the confidence value is 100% during the normal training and learning process. In case of input pattern changes, the data will be mapped in the locations which are not exactly the same as the ones previously trained or learned. Therefore, a confidence value is calculated to represent the level of change (or level of degradation) of the behavior. A high percent (%) confidence value shows a high possibility of a good pattern. A low confidence value represents the degraded or fault pattern. The confidence value is calculated based on the following formula:

\[ \text{Confidence Value} = 100 \times (\text{Repeatabile-Weight-Count) / (Address-Array-Size}) \]

For example, in Figure 2 a good pattern is learned during the training. The data are mapped on four locations in the weight table. The confidence value is 100 percent [since \( 100\times(1+1+1+1)/4 = 100 \) percent]. During operations, a fault pattern is observed by the CMAC, and data are mapped on different locations in the weight table. If two locations are different, two "0"s will be placed on two "1"s positions. Then, a 50 percent [\( 100\times(1+1+0+0)/4 = 50 \) percent] confidence value is obtained.

The confidence value presented here represents a conditional probability of behavioral degradation. If these values can be monitored in an incremental time interval, then the rate of degradation can be calculated. The rate of changes will suggest the urgency of the maintenance.

**EXPERIMENTATION**

The new machine behavioral learning, degradation monitoring, and fault detection technique was tested and implemented in monitoring degradation in the accuracy of a robot. To examine the developed CMAC-PDM and to investigate its capability for monitoring degradation and detecting faults, seven different degradation test patterns were generated by adjusting the backlash screws of the robot arms and wrist. In this experiment, the position accuracy and path straightness were used as evaluation criteria for measuring robot performance [22-24]. A PUMA 560 robot was used to study machine degradation recognition and monitoring of the developed technique. Figure 3 shows the experiment setup. The HP 5518A (two-frequency He-Ne laser) model is used to measure the robot position accuracy and path straightness.

Before adjusting the backlash of the robot arm, a set of path straightness and position accuracy data was generated to train the CMAC by programming a robot to follow the taught path pattern. There are three different movements for both +/- direction at each of five taught positions. There are 13 inputs (1 for position number and 12 for position/straightness data) for CMAC training for each position.

**TEST RESULTS**

A degraded test pattern was generated by introducing different levels of backlash adjustment.
Seven adjustments were made in this experiment, with each level of adjustment representing a different degree of degradation (see Figure 4). These sets of patterns were used to test the CMAC-PDM and to investigate its capability in monitoring degradation. Figure 5 shows the confidence plot for each level of adjustment. The graph indicates that smaller address array sizes (20 or 30) can recognize each level of degradation more effectively. Test result shows that the CMAC-PDM can monitor the degradation of robot performance adaptively and quantitatively.

CONCLUSION

This paper presents a concept for using neural networks to monitor machine degradation and detect faults. Since degradations generally occur before failures, monitoring the trends of machine degradation allows the degraded behavior or faults to be corrected before they cause failure and machine breakdowns. A novel proactive implementation technique, namely, a pattern discrimination model (PDM), based on the cerebellar model articulation controller (CMAC) has been developed and refined. Experimentation has been used to demonstrate the feasibility of this technique.

In conclusion, the developed methodology has demonstrated an adaptive capability in providing an active and quantitative performance indicator that enable the maintenance personnel to perform early fault diagnosis and effective maintenance.

REFERENCES

FIGURE 1 PROPOSED MACHINE DEGRADATION MONITORING METHODOLOGY

FIGURE 2 CMAC PATTERN DISCRIMINATION MODEL
FIGURE 3

ROBOT POSITION ACCURACY DEGRADATION
MEASUREMENT USING LASER INTERFEROMETER

<table>
<thead>
<tr>
<th>BACKLASH ADJUSTMENT CONDITION</th>
<th>TRAVEL DISTANCE 1</th>
<th>TRAVEL DISTANCE 2</th>
<th>TRAVEL DISTANCE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>POSITION ACCURACY</td>
<td>STRAIGHTNESS</td>
<td>POSITION ACCURACY</td>
</tr>
<tr>
<td>ADL #1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #2</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #5</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #6</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ADL #7</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

FIGURE 4

TESTING DATA SETS FOR DIFFERENT BACKLASH ADJUSTMENT CONDITIONS

FIGURE 5

Confidence (Percentage)

Address Array Size

732