Forests and Trees: distinguishing many from few with a video camera

Ronald Kuivila
Music Department
Wesleyan University
Middletown, CT 06459 USA
rkuivila@wesleyan.edu

ABSTRACT

Algorithms for real time video motion typically examine 'zones' of pixels. While there may be multiple zones and the zones may move to track motion, the sizes and shapes of the zones typically remain fixed during the analysis of a single frame of video. An algorithm that allows the shape of zones to be dynamically adapted to the video image during analysis is presented, some of its possibilities of application are discussed, and an implementation is demonstrated.

1. INTRODUCTION

There are a number of systems that derive control signals from the analysis of a video stream, such as David Rokeby's Very Nervous System, STEIM's Big Eye, and Eric Singer's Videolina. The algorithms used in these systems are quite similar; all involve the following steps:

• The difference between the luminance of the current frame of video and a reference frame (either the most recent preceding frame of video or a fixed reference frame) is computed.

• A "difference map" is computed. This is a bitmap where pixels in the difference frame exhibiting 'significant change' (i.e. exceeding a threshold determined by the noise floor of the video system) are set to 1 and all other pixels are set to 0.

• The difference map is broken into distinct analysis zones and control signals are derived from the change found in each zone. Triggers (as opposed to continuous control signals) may also be generated based on the patterns of change detected in each zone.

While analysis zones may move and change shape between successive frames of video, the sizes and shapes of the zones typically remain fixed during the analysis of any one frame. This is due in part to the efficiency requirements of a real time system. Changing the shape of the analysis zone would require rereading the difference frame and recomputing the difference map.

Here we present a representation of the difference map that allows any arbitrary subrectangle to be efficiently read as an analysis zone. This facilitates a hierarchical approach to the analysis of the video image, making it possible to efficiently extract both global and local features. The current implementation is as an external object in MAX. However, the approach taken is well suited to implementation on a co-processor with a shared memory interface to a host computer, as is commonly found in current video display devices.

2. DERIVING CONTROL SIGNALS FROM ANALYSIS ZONES

We begin with a brief review of the derivation of control signals from analysis zones within the difference map. The simplest approach is to provide a control signal proportional to the total amount of change in the zone. It is also straightforward to determine extremal information (i.e., the highest, lowest, leftmost, or rightmost changed pixel within a zone). These locations are immediately apparent to users and are, consequently, often a good choice as a control parameter. Such data can also be useful in estimating distances. For example, if a camera is mounted overhead but at one end of a space, the changed pixel lowest on the screen will indicate the closest moving object.

It is also possible to compute the 'centroid of change' within a zone. To define this concept precisely, let \( p(x,y) \) be a function that returns 1 whenever the pixel at point \((x,y)\) has changed 'significantly' and 0 otherwise. Then, the total number of changed pixels (with the summation over all pixels in the analysis zone) is:

\[
\sum p(x,y)
\]

Assuming that some pixels have changed (so that \(\sum p(x,y)\)) is non-zero), the centroid of change, \((cx, cy)\), is defined to be:
This position can be used to recenter the analysis zone, allowing it to track an object as it moves across the entire video image. Of course, the object must move slowly enough that it remains within the analysis zone from frame to frame.

3. AN ALTERNATIVE DIFFERENCE MAP

For each frame of video data, three matrices, P, X, and Y, are computed. Each matrix has dimensions identical to that of the video image. Let $p(x,y)$ be a function that returns 1 whenever pixel $x,y$ has changed significantly and 0 otherwise (i.e., it is simply the difference map expressed as a function). The first matrix is defined as:

$$P(m,n) = \sum p(i,j) \quad \text{where} \quad i < m \text{ and } j < n.$$  

So $P(x,y)$ is the total number of changed pixels in the rectangle formed between $(0,0)$ and $(x,y)$.

However, it is also possible to compute the total number of changed pixels in any arbitrary subrectangle $[(x_1,y_1)(x_2,y_2)]$ as:

$$P = P(x_1,y_1)+P(x_2,y_2)-P(x_1,y_2)-P(x_2,y_1)$$

The second and third matrices, X and Y, represent the sum of the x and y coordinates of changed pixels in the rectangle formed between $(0,0)$ and $(m,n)$.

$$X(m,n) = \sum (i* p(i,j)) \quad \text{where} \quad i < m \text{ and } j < n$$

and

$$Y(m,n) = \sum (j* p(i,j)) \quad \text{where} \quad i < m \text{ and } j < n$$

The centroid of change, $(cx, cy)$, of the rectangle formed by the origin and a point $(x, y)$ can be computed as:

$$(cx, cy) = (X(x,y)/P(x,y), Y(x,y)/P(x,y))$$

The centroid of change in an arbitrary subrectangle $(x_1,y_1,x_2,y_2)$ as:

$$cx = (X(x_1,y_1) + X(x_2,y_2) - X(x_1,y_2) - X(x_2,y_1)) / p$$

$$cy = (Y(x_1,y_1) + Y(x_2,y_2) - Y(x_1,y_2) - Y(x_2,y_1)) / p$$

A simple recursive algorithm can be used to determine the extremal points of change within the analysis zone. For example, assuming there are changed pixels, the following function will find the leftmost zone:

```java
int findLeftmost (int x0, int y0, int w, int y1) {  
  if (w <= 0) { return(x0); }  
  if (totalChangeInRectangle(x0, y0, x0 + w, y1) != 0) {  
    return(findLeftMost(x0, y0, w/2, y1));  
  } else {  
    return(findLeftMost(x0 + w/2, y0, w/2, y1));  
  }
}
```

4. HIERARCHICAL ANALYSIS

If we assume that the maximum size of people (or objects) moving within the field of view of the camera is a relatively small subrectangle of the video image, estimates of the number of people moving within view of the camera and their relative positions can be made using the same recursive approach:

```java
int countContents (int x0, int y0, int w, int h, int res) {  
  if (totalChange(x0, y0, w, h) != 0) {  
    addToActiveList(res,x0,y0);  
    if (w > limit) {  
      countContents (x0, y0, w/2, h/2, res + 1);  
      countContents (x0 + w/2, y0, w/2, h/2, res + 1);  
      countContents (x0, y0 + h/2, w/2, h/2, res + 1);  
      countContents (x0 + w/2, y0 + h/2, w/2, h/2, res + 1);  
    }  
  }
}
```

But this function actually tells us more. At a given resolution, the ratio of the number of active zones to the total number of zones gives an indication of how varied the distribution of motion is at that resolution. These ratios decrease with increasing resolution. For example, if one zone out of four has motion at the coarsest resolution, then only four out of sixteen can possibly have motion at the next finest resolution. If, at some resolution, the ratios stop decreasing, then motion is uniformly distributed at any finer resolution. Thus, it will be impossible to distinguish individual ‘trees’ in the ‘forest’ of motion.

Installations centered on ‘interactive control’ generally suffer with many participants. With too much activity, it becomes impossible to discern the nature of response to any individual action. The resolution at which the ratios stop decreasing is a natural level at which to base interactive control. This may be the activity of individuals or of groups, depending on the total level of activity in the installation.
5. FUTURE WORK: DERIVING TRIGGERS FROM ANALYSIS ZONES AS A BASIS FOR DEPTH ESTIMATES

The simplest way to derive a trigger from an analysis zone is to provide a trigger whenever the total number of changed pixels first exceeds a threshold. However, the low to moderate resolution of video can make the result of this approach unsatisfactory. The problem is to accurately detect the onset of a gesture. For example, a very slow continuous movement may not result in significant change between all frames of video. This will result in an erratic pattern of changed pixels and a 'bursty' trigger signal that does not accurately reflect the patterns of motion. An elegant solution provided by David Rokeby is to integrate the changed pixel signal over a series of increasing time intervals. If the value of a shorter interval integral is larger than that of its longer interval successor, a gesture is in process.

Here is an example of this approach written in C:

```c
filter = float[0,0,0,0,0,0,0];

updateActivityFilter (integer totalChange) {
  for(i=0, k=.7; i<6; i++, k*=.50)
    filter[i] = (1-k)*filter[i] + totalChange*k;
}

motionDetector( float *filter ) {
  for(i=0; i<5; i++)
    if( filter[i] > filter[i+1] ) {return(true); }
  return(false);
}
```

The matrix technique described above can be applied to these filters as well, making it possible to determine the onset of gestures in arbitrary zones of the image. Depth estimates are derived from stereoscopic images by examining the horizontal displacement of correlating points in the two images. The basic computational problem is to identify which points in the two images actually correlate. Gestural onset data could be applied to making these correlations, creating a three dimensional gestural input system.

6. ACKNOWLEDGEMENTS

The work described here was developed during my tenure in the Artist in Berlin program of the DAAD. I would like thank Ingrid Beirer and the Berlin DAAD for their support. I would also like to thank Ioannis Zannos and Staatliches Institut für Musikforschung for providing me with an office, technological support, and a congenial environment during this time. Last, but by no means least, I would like to thank Eric Singer for generously providing me the source code for his VideoIn external object. This saved enormous time in developing the software described in this paper.