Connected Component Analysis and Change Detection for Images

Prasad S.Halgaonkar
Department of Computer Engg,
MITCOE – Pune University, India

***Abstract* – Detection of the region of change in images of a particular scene or object taken at different time gap is of widespread interest due to a large number of applications in various disciplines. Some of the important applications of change detection include Video surveillance, medical diagnosis and treatment, civil infrastructure, underwater sensing etc.**

In this paper, described is a two-scan labeling algorithm whereby, unlike the classical approach, equivalences are processed during the first pass in order to determine the correct state of equivalence classes at each time of the scan. This is obtained by merging classes as soon as a new equivalence is found, the data structure used to support the merging being a simple 1D array. This approach allows the check for a conflict to be carried out on class identifiers rather than on labels, as instead it is mandatory with the classical algorithm. It has been show that this significantly improves the efficiency of the labeling process.

***Keywords* – Digital Image Processing, Change Detection, Connected Component Analysis.**

I. Introduction

a) Change Detection

The basic problem definition of change detection is that, we are given a set of images of the same scene taken at several different times. Our goal is to identify the set of pixels that are significantly different from the last image of the sequence i.e. current image and the previous image, these pixels are called the change pixels or change pixel mask. This change mask may result from a combination of various factors. It includes appearance or disappearance of objects, motion of objects relative to the background, or shape change of objects.

The main thing is that the change mask should not contain unimportant or noise forms of change, such as those induced by illumination variation, camera motion, sensor noise etc. This can be removed by using an appropriate filter before change detection step of an image.

To make change detection problem more specific, let $\{I_1, I_2, I_3,\ldots..,I_M\}$ be an image sequence in which each image maps a pixel co-ordinate$X \in R^l$ to an intensity or color$I(X) \in R^k$. Typically, k = 1 for grayscale images or k = 3 for RGB color images. A basic change detection algorithm takes the image sequence as input and generates a binary image

$\rightarrow$

$B:R^1$ [0, 1] called a change mask or change detected output image that identifies the changed region in the last image according to the following rule.

$$B(X) = 1, \quad \text{if there is a significant change at pixel X of } I_M$$
$$= 0, \quad \text{otherwise.} \qquad \ldots\ldots\ldots (1)$$

b) Connected Component Analysis

The output of the change detection module is the binary image that contains only two labels, i.e., ‘0’ and ‘255’, representing as ‘background’ and ‘foreground’ pixels respectively, with some noise. The goal of the connected component analysis is to detect the large sized connected foreground region or object. This is one of the important operations in motion detection. The pixels that are collectively connected can be clustered into changing or moving objects by analysing their connectivity [1].

In binary image analysis, the object is extracted using the connected component labelling operation, which consist of assigning a unique label to each maximally connected foreground region of pixels.

To perform such type of operation various approaches are presented in the literature. They differ in the data structure used to register the equivalence label and are application specific. One of the important labeling approaches is “classical sequential labeling algorithm” [2, 3]. It is based on two raster scan of binary image. The first scan performs the temporary labeling to each foreground region pixels by checking their connectivity of the scanned image. When a foreground pixel with two or more than two foreground neighboring pixels carrying the same label is found, the labels associated with those pixels are registered as being equivalent. That means these regions are from the same object. The handling of equivalent labels and merging thereafter is the most complex task.

II. Change Detection AND Connected Component Analysis

1) Change Detection

1.1 Types of Change Detection

Based on the above information there are various types of change detection methods available as explained in the survey on change detection by Richard J. Rodake et. al.[4]. Here, implemented is a simple change detection algorithm based on differencing.

1.1.1 Differencing of images

This change detection method is based on the signed difference image $D(X) = I_1(X) – I_2(X)$ and such approaches are commonly used. The most obvious algorithm is to simply threshold the difference image i.e., the change mask $B(X)$ is generated according to the following rule.

$$B(X) = \begin{cases} 1 & \text{if } |D(X)| > \tau \\ 0 & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

The threshold τ is chosen by observation. This technique is sensitive to noise and variation in illumination. The goal of a change detection algorithm is to detect significant changes while rejecting unimportant ones. Some preprocessing steps require an image for detecting important and unimportant changes in the image. These steps include some geometric and illumination (intensity) adjustment. There are several methods that are closely related to simple differencing.

1.1.2 Significance and hypothesis test

The decision rule in many change detection algorithms is based on the statistical hypothesis test. The decision such as whether or not a change has occurred at a given pixel in an image correspond to one of two hypothesis. The null hypothesis H_0 or the alternative hypothesis H_1, corresponding to no-change and change decision respectively.

The image pair $(I_1(X), I_2(X))$ is viewed as a random vector. Knowledge of the conditional joint probability density functions P $\left(I1(X)\ ,\ {}^{I2(X)}/_{H0}\right)$ and P $\left(I1(X)\ ,\ {}^{I2(X)}/_{H1}\right)$ allows us to choose the hypothesis that best describes the intensity changes at *X*.

1.1.3 Predictive Models

The results of the more sophisticated change detection algorithms are based on the close relationship between the nearby pixels both in space and time. In this, the pixels for which predictor performs poorly are classified as changed. The goal is to distinguish unusual changes from the expected changes, but this process somewhat is uncertain and it’s unable to correspond directly to the concept of foreground and background.

1.1.4 The shading model

Several change detection techniques are based on the shading model. The observed image intensity at a pixel *X* can be modelled as the product of two components, the illumination $I_l(X)$ from the light source (s) in the scene and the reflectance $I_0(X)$ of the object surface to which *X* belongs, as follows

$$I(X) = I_l(X)\, I_0(X)\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

This is called the shading model. Here, only the reflectance component $I_0(X)$ contains the information about the object in the scene. Based on a type of illumination-invariant change detection can be performed by first filtering out the illumination component from the image. Such algorithms generally compare the ratio of image intensities to a threshold determined by observation.

$$R(X) = \frac{I2(X)}{I1(X)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (4)$$

1.1.5 Background Modelling

Background modelling is required in the area of video surveillances. The goal is to determine the number of background pixels (stationary pixels) from the large number of image sequences that are separated by seconds rather than months before determining the foreground pixels (changed pixels). Furthermore, there is frequently an important requirement for outdoor surveillance is to update the background model after some time. Here, most background modelling approaches assume the camera to be fixed, i.e., the images are already registered. There are several background maintenance algorithms present in now-a- day’s video surveillance systems.

Many approaches fall into the mixture-of-Gaussians category. The mean and covariance of each background pixel are usually initialized by observing several seconds of video of an empty scene after applying the Median filtering to determine the stationary pixels and moving pixels. This approach is presented by Ismail Haritaoglu et. al. [5].

1.2 Implemented Algorithm

The steps for the implemented simplified change detection algorithm are presented below:

1. Read first pnm image frame in the sequence.
2. Convert RGB image to **Grayscale** image
   For every pixel in the image co-ordinates
   - Extract the (R, G, B) component of this pixel
   - Compute the Grayscale using $\frac{(R+G+B)}{3}$
   - Mark this pixel in the output image with gray scale.
3. Do Median filtering on grayscale image
4. Store this image as a background
5. Read next frame
6. Convert RGB to Grayscale image
7. Do Median filtering on grayscale image
8. Subtract the current frames from the background

9. Do binary operation on differenced image
10. Do (optional) Median filtering on binary image.
11. Repeat steps 5 to 10 for all image frames.

This change detection method is based on the signed difference image $D(X) = I_1(X) – I_2(X)$. The first frame is considered in the sequence as a background image and subtracted the next consecutive frames from the background image. The binary conversion of an image is to simply threshold the difference image, i.e., the change mask *B(X)* is generated according to the following.

$$B(X) = 255 \quad \text{if } |D(X)| > \tau \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$
$$= 0 \quad \text{otherwise}$$

Threshold τ is chosen manually. Pixels labelled as '0' are called **background** pixels and those labelled with '255' are called **foreground** pixels. This technique is sensitive to noise and variation in illumination. To remove the noise, the Median filtering is applied once again on the change mask or binary image. This results in only the large objects to be recognizable. The goal of this procedure is to detect significant changes while rejecting unimportant ones.

2) Connected Component Analysis

The simplified approach of the "simple and efficient connected component labelling algorithm" [2] is used for this purpose. It is the classical sequential labelling approach which is shown in figure 1 as a simplified illustration. This method mostly relies on the first processing step or the first scan which detects the moving parts. At the end of this phase it is necessary to identify each moving objects or blobs, assigning a unique label to all the pixels belonging to it.

2.1 Implementation

The first scan gives temporary labels to the foreground pixels according to their connectivity. The connectivity check can be done with the help of either a 4-connectivity or 8-connectivity approach. **8-connectivity approach** is used. Here, the idea is to label the whole blob at a time to avoid the label redundancies.

Let a change is detected in the binary image *I* where F and *B* are the foreground and background pixels of a binary image. A connected component of *I*, is the subset of F of maximal size such that in this subset all the pixels are connected. Two pixels *P* and *Q* are connected if there exists a path between the pixels $(p_0, p_1, p_2 \ldots p_n)$ such that, $p_0 = P$, $p_n = Q$ and, $\forall\ 1 \le i \le n$, p_{i-1} and p_i are neighbours. That means the connected component depends on the pixel's neighbourhood.

1. The labelling operation scans the image moving along the row until it comes to the point P, for which S = {255}. When this is true, it checks the four neighbours of which have already been scanned as shown in the following figure 2.

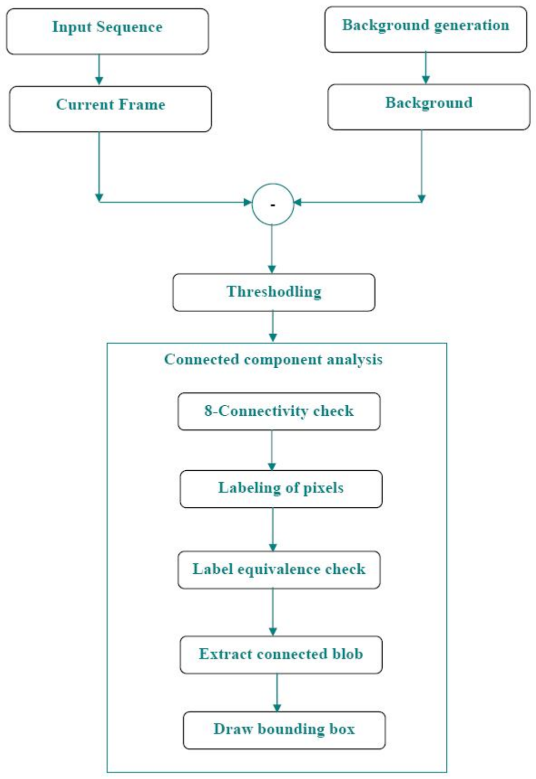

Figure 1: Flow Diagram of Connected component analysis

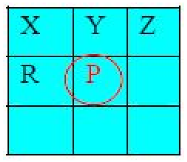

Figure 2: 8-Connectivity check

Based on that information, the labelling of *P* occurs as follows,

If all four neighbours are '0' assign a new label to P, and increment the label,
Else
If only one neighbour has S = {255} assigns its label to *P*
Else (i.e., more than one of the neighbours has S = {255})
Assign one of the labels to P.

Here, note that the relation between the pixels that are expressed through a "label value" in the labelled image depends on the value of the label. That means the two pixels from background, labelled as l_B are not necessarily to be connected, but the two pixels labelled l_P from the foreground region are to be connected.

As a result of the first scan, a temporary label is assigned to pixel belonging to the different components, but different labels may be associated with the same component. As illustrated in the following figure 3 it shows the labelled image.

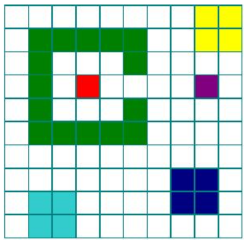
Figure 3: Output of the first scan

Let the red component be the part of green component and is equivalent but this is shown as a different label. Our job to find out the equivalent blobs from the image. All closest neighbouring blobs are found by extracting the starting co-ordinates of the corresponding rows and columns and observing the minimum and maximum distance between them. This is done by putting some thresholds, obtained manually and extracting all closest distance blobs and merging them as one single component or blob.

During the second scan, the unique label '255' is assigned to only maximally connected foreground region and remaining other regions or blobs are considered as a part of background by assigning background label '0'. The output image of the second scan contains only the unique labelled maximally connected foreground region or object. This is illustrated in the following figure 5.4. This output image is passed on to the object detection step which assigns a bounding box around the object to be tracked.

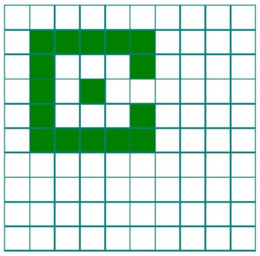
Figure 4: Output of the second scan

2.1.1 Object Identification and the Bounding Box

The minimum and maximum co-ordinates of rows and columns of the identified object obtained from connected component analysis are extracted and a rectangular bounding box is drawn around it using these co-ordinates.

III. EXPERIMENTAL RESULTS

The ***Change Detection and Connected Component Analysis*** routines were implemented in C language [6] in Linux OS [7] with the help of GCC compiler.

1) *Change Detection*

The following figure 5, 6, 7 and 8 shows the output binary images of change detection. It contains the large sized object with some noisy part. As shown in figures, there is effective detection of changes in images.

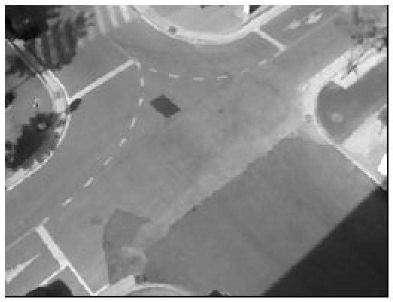
Figure 5: Background grayscale image

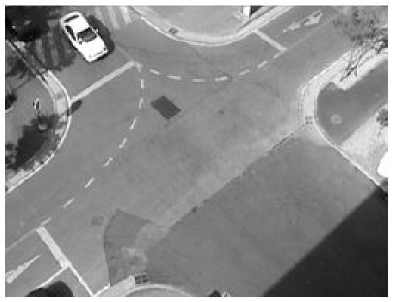
Figure 6: Current Frame

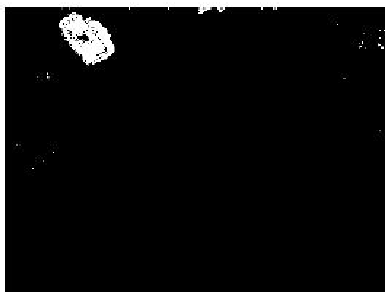
Figure 7: Output Image without Median filtering

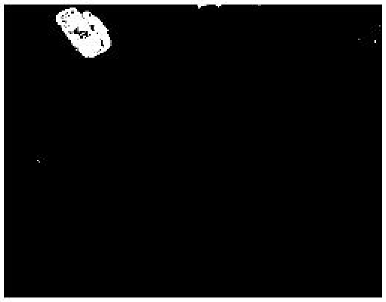
Figure 8: Output Image with Median filtering

2) *Connected Component Analysis*

The following figure 9, 10, 11 and 12 shows the input binary image, output images and output color image of connected component analysis and the bounding box around the detected object.

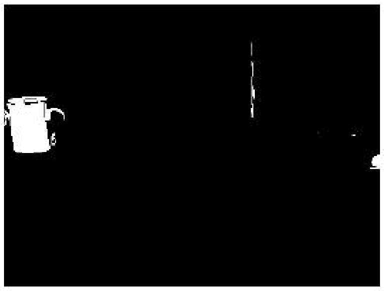
Figure 9: Input image

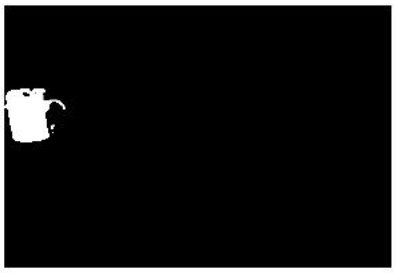
Figure 10: Output image

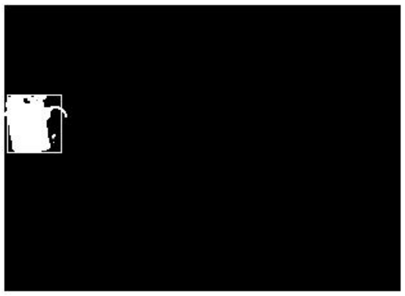
Figure 11: Detected object with the bounding box

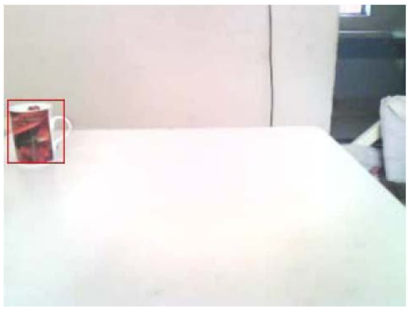
Figure 12: Color image of output

IV. CONCLUSION

Change Detection and Connected Component Analysis are preliminary but necessary steps to achieve our final objective of object tracking. Our future work focuses on developing the algorithm for object tracking to track the identified object in motion, surrounded by the bounding box.

REFERENCES

[1] Allen Bovik, "*The Essential Guide to Video Processing*," Academic Press- 2nd Edition 2009.

[2] Luigi Di Stefano, Andrea Bulgarelli, "*A Simple and Efficient Connected Components Labeling Algorithm*," ICIAP, 10th International Conference on Image Analysis and Processing, pp.322, 1999.

[3] J. Bigun and T. Gustavsson (Eds), "*A Single-Scan Algorithm for Connected Components Labeling in a Traffic Monitoring Application*," *SCIA 2003*, LNCS 2749, pp.677-684, 2003.

[4] Richard J. Radake, Srinivas Andra, "*Image Change Detection Algorithms: A Systematic Survey*," IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 22, no. 3, August 2000.

[5] Ismile Haritaoglu, David Harwood, "*Real-Time Surveillance of People and Their Activity*," IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 22, no. 8, August 2000.

[6] Yeshwant Kanetkar, "*Let Us C*," Allied Publisher, 3rd Edition 1998.

[7] J. Purcell, "*Inc Linux Complete Command Reference*," Red Hat Software, 1997.