International Journal of Computer Applications (0975 – 8887)
Volume 139 – No.6, April 2016

Clustering of Human Beings in an Image and Comparing the Techniques on the basis of Accuracy and Time

Dipen Saini
Assistant Professor
D.A.V. College, Jalandhar

Ramandeep Kaur
Assistant Professor
D.A.V. College, Jalandhar

ABSTRACT

Data clustering is a method in which we make cluster of objects that are somehow similar in characteristics. The criterion for checking the similarity is implementation dependent. If data has some meaning and it corresponds to a human being, than how we can group it in an image or a video. In this research work, we have used three types of algorithms for group detection in an image. In this paper three types of algorithms are used in group detection which detects groups in an image linear dimensionally.

Keywords

Data clustering, Digital image, group detection, features extraction.

1. INTRODUCTION

In this paper, we present methods for tracking groups of people in an image to recognize abnormal behaviours. [1] In this paper we have used three types of algorithms to detect groups in an image. First algorithm is used to detect groups on the basis of distance, second algorithm detects groups on the basis of template matching and the third one detects groups on the basis of feature extraction. Suppose we have an image of a public place in which we have identified a terrorist, with the help of group detection we can detect various groups present in an image and from that we can extract faces and after face extraction we can identify all other terrorist present in that image.

The remainder of this paper is organized as follows. Section 2 briefly reviews some related work. Section 3, gives system architecture and description in which introduction to various algorithm with practical implementation are given. Comparisons of various algorithms with practical results are given in Section 4. Conclusion and future work is mentioned in section 5.

2. RELATED WORK

Systems for tracking people have usually employed some form of background subtraction: Haritaoglu et al. used greylevel subtraction [2, 3].

The car tracking system of Koller et al. [4] used an adaptive background model based on monochromatic images filtered with Gaussian and Gaussian derivative (vertical and horizontal) kernels. Background 'subtraction' using these filter outputs yielded results superior to the use of raw grey levels. Some systems for surveillance and monitoring of wide area sites have tracked people essentially assuming that each connected component obtained from background subtraction (and some further processing) corresponds to a moving object [5, 6]. Trackers based on 2D active shape models have been used but can only cope with moderate levels of occlusion [7, 8]. Color 'blobs' were used in Pfinder to build a person model in a controlled indoor environment [9]. Intille et al.'s closed world tracking was used (for example) to track players on an American football field [10]. McKenna and Gong used a combination of motion, skin color and face detection to track people and their faces [11]. Lipton et al. combined temporal differencing with template matching to track people and cars in wide area scenes [12]. Darrell et al. combined depth from stereo with skin color and face detection [13]. Bregler proposed an ambitious probabilistic framework for tracking at multiple levels of abstraction [14]. Many approaches have been proposed for tracking the human body.

3. SYSTEM ARCHITECTURE AND DESCRIPTION

This chapter discusses the three types of algorithms used to detect the group in an image. In this, we explain each of the steps of all the three algorithms, involved in our solution to group detection i.e. one or more groups present in an image in detail. Group detection is done on the basis of three things first is the distance i.e. distance between the persons, second one is template matching in which template matching is done on face and the last one is feature extraction which is used to calculate face orientation.

3.1 Group Detection on the Basis of Distance

Fig. 3.1 shows the block diagram of the first algorithm and its constituent modules. In this algorithm groups are detected on basis of distance. An image captured by a still camera is the input to the system. After receiving an image as input to the system, we first use a Haar cascade algorithm of upper body for finding the number of people in an image. Being Haar cascade algorithm detects people randomly, after that sorting of people has to be done from left to right. After that distance is calculated between first person and second person, if it is less than 100 pixels, then that is considered to be one group, next distance is calculated between second person and third person, if the distance between them is less than 100 pixels then the group becomes from first person to third. Now if the distance between third and fourth is greater than 100 pixels then a new group formation will start from fourth person and previous group will end at third person and so on till the last person. Thus, by performing all these steps we find all the groups present in an image.

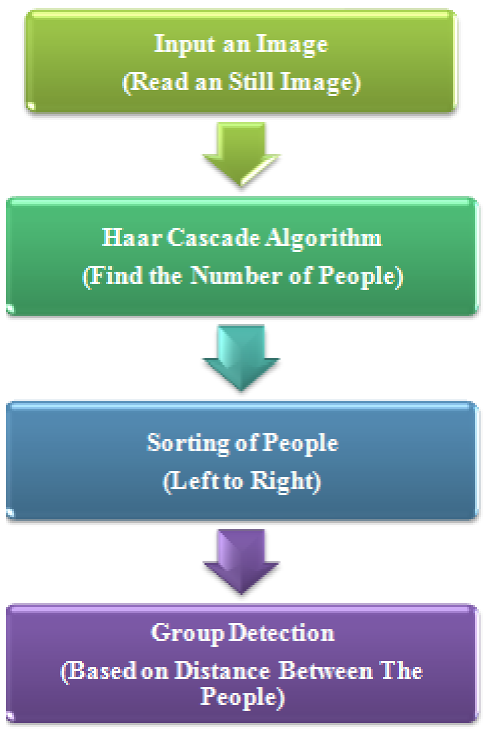


Figure 3-1 First Block diagram of the system

3.1.1 *Input to the System*

The basic and important thing for any system is its input data which is used to verify the functioning of the system by seeing the results. We need input data so that we can apply different types of algorithms on it and see the results whether they are same as desired or not. In this case, an image is an input to the system. An image is nothing more than a two dimensional signal. It is defined by the mathematical function f(x, y) where x and y are the two co-ordinates horizontally and vertically. The value of f(x, y) at any point is gives the pixel value at that point of an image. A digital image is a numeric representation (normally binary) of a two-dimensional image. Depending on whether the image resolution is fixed, it may be of vector or raster type. By itself, the term "digital image" usually refers to raster images or bitmapped images.

Figure 3-2 Digital image

The above figure 3.2 is an example of digital image that you are now viewing on your computer screen. But actually, this image is nothing but a two dimensional array of numbers ranging between 0 and 255.

128	30	123
232	123	124
123	77	89
80	255	255

Each number represents the value of the function f(x, y) at any point. In this case the value 128, 230, 123 each represents an individual pixel value. The dimensions of the picture is actually the dimensions of this two dimensional array.

3.1.2 *Haar Cascade Algorithm*

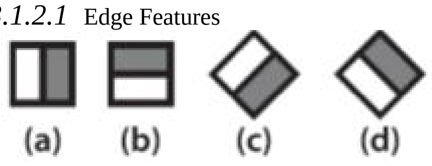
Haar cascade is an object detecting technique based on Viola-Jones algorithm which is used to detect objects present in a digital image. Haar cascade algorithm is used to detect person present in an image based on various features like face, eyes, nose, upper body, lower body. In Haar cascade algorithm, we have Haar cascade classifier which basically tells what to look for in images. Most of the time, when one is about to create a classifier, we suddenly have to decide which features to consider. A feature is a characteristic, something which will hopefully bring enough information in the decision process so the classifier can cast its decision. For example, suppose we are trying to create a classifier for distinguishing whether a person is overweight. A direct choice of features would be the person's height and weight. Hair color, for example, would not be a much informative feature in this case.

The core origin for Haar classifier object detection is the Haar-like features. These features, rather than using the intensity values of a pixel, use the alteration in contrast values between adjacent rectangular groups of pixels. The contrast variances between the pixel groups are used to determine relative light and dark areas. Two or three adjacent groups with a relative contrast variance form a Haar like feature.[15]

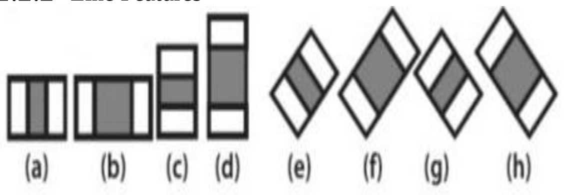
The object detector based on Haar classifiers uses rectangular areas (Haar features) to make the decision if the region of the image looks like the object of interest or not.

$$f(x) = \begin{cases} -1, & v_i < t_i \\ +1, & v_i \geq t_i \end{cases}$$

3.1.2.1 Edge Features



3.1.2.2 Line Features



3.1.2.3 Center-Surround Features

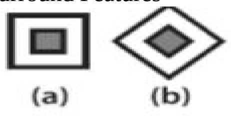


Figure 3-3 Type of Haar like features used in Object detection

Figure 3.3 shows different types of Haar features used. The Haar detector uses a form of AdaBoost but organizes it as a rejection cascade of nodes, where each node is a multitree AdaBoosted classifier designed to have high (say, 99.9%) detection rate (low false negatives) at the cost of a low (near 50%) rejection rate (high false positives). For each node, a "not in class" result at any stage of the cascade terminates the computation, and the algorithm then declares that no object exists at that location.

In this algorithm, we used Haar cascade algorithm for upper body. Haar cascade algorithm is used to detect the upper-body region, which is defined as the head and shoulders area.

3.1.2.1 Upper Body Classification Problem

The Classification Model property controls the type of object to detect. By default, the detector is configured to detect faces. The upper mode classification model uses Haar features to encode the details of the head and shoulder region. Because it uses more features around the head, this model is more robust against pose changes, e.g. head rotations/tilts.

$\text{detector} = \text{vision.CascadeObjectDetector(Model)}$

creates a System object, detector, that detects objects using the Viola-Jones algorithm.

The $\text{vision.CascadeObjectDetector}$ System object comes with several pretrained classifiers for detecting frontal faces, profile faces, noses, eyes, and the upper body. However, these classifiers are not always sufficient for a particular application.

The $\text{vision.CascadeObjectDetector}$ System object detects objects in images by sliding a window over the image. The detector then uses a cascade classifier to decide whether the window contains the object of interest. The size of the window varies to detect objects at different scales, but its aspect ratio remains fixed. The detector is very sensitive to out-of-plane rotation, because the aspect ratio changes for most 3-D objects. Thus, you need to train a detector for each orientation of the object. Training a single detector to handle all orientations will not work. The cascade classifier consists of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners. Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive[16].

3.1.2.2 Detecting upper body in an image using Upper Classification Model

Detection of upper body is done using these three steps:

(a) Create a detector object and set properties.

(b) Read input image and detect upper body.

(c) Annotate detected upper bodies.

(a) Create a detector object and set properties.

$$\text{bodyDetector} = \text{vision.CascadeObjectDetector(Model)};$$

$$\text{bodyDetector.MinSize} = [\,];\ \text{(Default Value)}$$

$$\text{bodyDetector.MergeThreshold)} = 4;\ \text{(Default Value 4)}$$

$$\text{detector} = \text{vision.CascadeObjectDetector(MODEL)}$$

creates a System object, detector, configured to detect objects defined by the input string, MODEL. The MODEL input describes the type of object to detect. There are several valid MODEL strings, such as 'FrontalFaceCART', 'UpperBody', and 'ProfileFace'.

Minsize: Size of smallest detectable object, specified as a comma-separated pair consisting of 'MinSize' and a two-element [height width] vector. Set this property in pixels for the minimum size region containing an object. It must be greater than or equal to the image size used to train the model. Use this property to reduce computation time when you know the minimum object size prior to processing the image. When you do not specify a value for this property, the detector sets it to the size of the image used to train the classification model. This property is tunable.

Detection threshold, specified as a comma-separated pair consisting of 'MergeDetections' and a scalar integer. This value defines the criteria needed to declare a final detection in an area where there are multiple detections around an object. Groups of colocated detections that meet the threshold are merged to produce one bounding box around the target object. Increasing this threshold may help suppress false detections by requiring that the target object be detected multiple times during the multiscale detection phase. When you set this property to 0, all detections are returned without performing thresholding or merging operation. This property is tunable.

(b) Read input image and detect upper body.

$$\text{I} = \text{imread(Name of the Image)};$$

$$\text{bboxBody} = \text{step(bodyDetector, I)};$$

$\text{bboxBody} = \text{step(bodyDetector, I)};$ returns BBOX, an M-by-4 matrix defining M bounding boxes containing the detected objects. This method performs multiscale object detection on the input image, I. Each row of the output matrix, BBOX, contains a four-element vector, [x y width height], that specifies in pixels, the upper-left corner and size of a bounding box. The input image I, must be a grayscale or truecolor (RGB) image.

(c) Annotate detected upper bodies

$$\text{IBody} = \text{insertObjectAnnotation(I,}'\text{rectangle}', \text{bboxBody,}'\text{UpperBody}');$$

$$\text{figure, imshow(IBody), title(}'\text{Detected upper bodies}');$$

3.1.2.2.1 Example

$$\text{pplDetector} = \text{vision.CascadeObjectDetector(}'\text{Upper Body}');$$

$$\text{bodyDetectorMinSize} = [60\ 60];$$

$$\text{bodyDetectorMergeThreshold} = 10;$$

$$\text{i} = \text{imread(}'\text{4.jpg}');;$$

$$\text{bboxBody} = \text{set(pplDetector, i)};$$

$$\text{out} = \text{insertObjectAnnotation(i,}'\text{rectangle}', \text{bboxBody,}'\text{UpperBody}');$$

$$\text{figure, imshow(out), title('Detected upper bodies');}$$

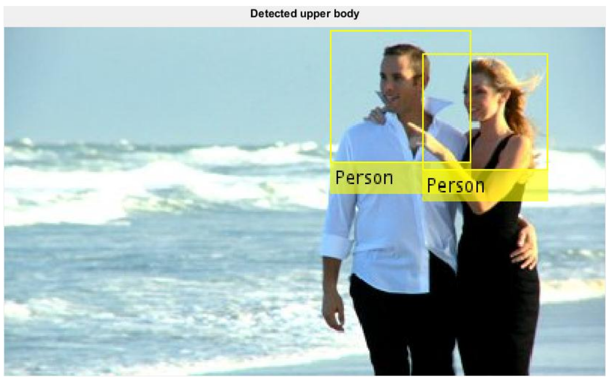


Figure 3-4 Detection of upper body

3.1.3 Sorting of Person

Sorting is any process of arranging items according to a certain sequence or in different sets, and therefore, it has two common, yet distinct meanings: ordering: arranging items of the same kind, class or nature, in some ordered sequence. Many techniques are present for sorting of elements like insertion sort, bubble sort, merge sort etc .But we have used bubble sort for sorting of persons. Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list.

3.1.4 Group Detection

After sorting of people group detection is done. Group detection is done by calculating the distance between the persons. First of all the distance is calculated between first person and second person, if it is less than 100 pixels, then that is considered to be one group, next distance is calculated between second person and third person, if the distance between them is less than 100 pixels then the group becomes from first person to third. Now if the distance between third and fourth is greater than 100 pixels then a new group formation will start from fourth person and previous group will end at third person and so on till the last person.

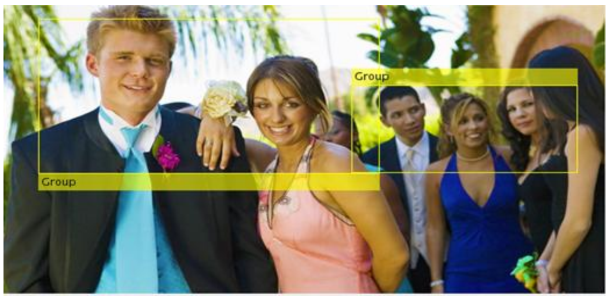


Figure 3-5 Group detection using distance

3.2 Group detection on the basis of Template matching

Fig. 3.6 shows the block diagram of the second algorithm and its constituent modules. In this algorithm, first three steps are same as that of previous algorithm. This algorithm differs at the fourth step. Group is detected using template matching. First of all an image captured by a still camera is the input to the system. After receiving an image as input to the system, we first use a Haar cascade algorithm of upper body for finding the number of people in an image. Being Haar cascade algorithm detects people randomly, after that sorting of people has to be done from left to right. After Template matching is done on its face. Thirteen templates are created for defining face orientation from -90 to 90 degree. Each person is matched with thirteen templates and the best possible template is picked and the face orientation is decided. After calculating face orientation of all the persons in an image, then grouping algorithm is applied.

Grouping Algorithm:

If the person is right facing, then orientation is positive, if the person is front facing, then orientation is zero, if the person is left facing, then face orientation is negative. Groups are cut out when the face orientation of let's say ith person in n number of people is negative and face orientation of (i+1) th person is positive. The only thing which is to be taken care of is the face angle of first two people. If face angle of person1 and person2 is same then count (the number of groups) should be started from 1 and if both are different then count should be started from 2.

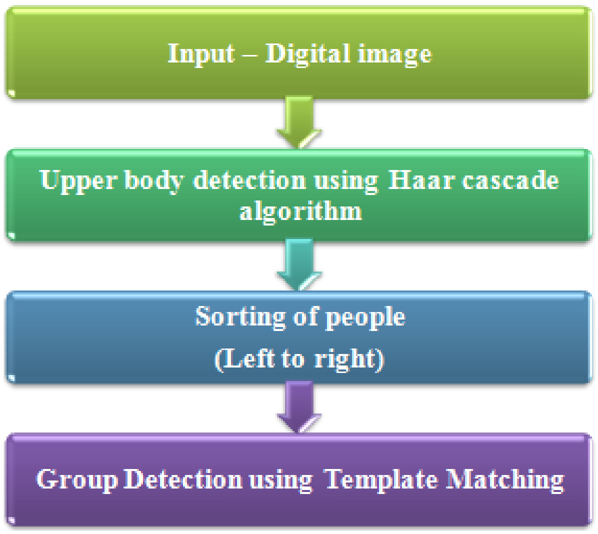

Input – Digital image
Upper body detection using Haar cascade algorithm
Sorting of people (Left to right)
Group Detection using Template Matching

Figure 3-6 Steps of algorithm2

3.2.1 Template Matching and Grouping of persons

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch).

We need two primary components:

(a) Source image (I): The image in which we expect to find a match to the template image

(b) Template image (T): The patch image which will be compared to the template image

A basic method of template matching uses a convolution mask (template), tailored to a specific feature of the search image, which we want to detect. This technique can be easily performed on grey images or edge images. The convolution output will be highest at places where the image structure matches the mask structure, where large image values get multiplied by large mask values.

This method is normally implemented by first picking out a part of the search image to use as a template: We will call the search image S(x, y), where (x, y) represent the coordinates of each pixel in the search image. We will call the template T(x t, y t), where (xt, yt) represent the coordinates of each pixel in the template. We then simply move the center (or the origin) of the template T(x t, y t) over each (x, y) point in the search image and calculate the sum of products between the coefficients in S(x, y) and T(xt, yt) over the whole area spanned by the template. As all possible positions of the template with respect to the search image are considered, the position with the highest score is the best position. This method is sometimes referred to as 'Linear Spatial Filtering' and the template is called a filter mask.

In this algorithm, we have used thirteen templates of face outline having orientation from -90 to 90 degree stored in the data base. The faces present in the image given as input are matched with the face outlines of the templates. We match four faces of the image with the thirteen templates. Each person face is matched with thirteen templates and the best possible template is picked and the face orientation is decided. After calculating face orientation of all the persons in an image, then grouping algorithm is applied. In grouping algorithm if the person is right facing, then orientation is positive, if the person is front facing, then orientation is zero, if the person is left facing, then face orientation is negative. Groups are cut out when the face orientation of let’s say ith person in n number of people is negative and face orientation of (i+1) th person is positive. The only thing which is to be taken care of is the face angle of first two people. If face angle of person1 and person2 is same then count (the number of groups) should be started from 1 and if both are different then count should be started from 2.

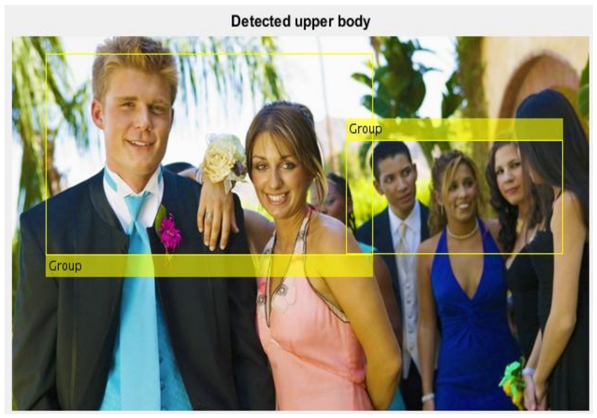


Figure 3-7 Group detection using template matching

3.3 Group detection on the basis of feature extraction

Fig. 3.8 shows the block diagram of the second algorithm and its constituent modules. In this algorithm, first three steps are same as that of previous algorithm. This algorithm differs at the fourth step. Group is detected using Feature Extraction. First of all an image captured by a still camera is the input to the system. After receiving an image as input to the system, we first use a Haar cascade algorithm of upper body for finding the number of people in an image. Being Haar cascade algorithm detects people randomly, after that sorting of people has to be done from left to right. After that each person is extracted from image. Then face orientation is calculated using feature extraction. If front face is detected in an image, face orientation is considered to be zero .Side face or right ear is detected then face orientation is taken to be negative. For the positive orientation of face ,As we have already taken out upper body that means there will be face there, but as first two did not detect that face, that means this orientation will be positive. After calculating face orientation of all the persons in an image, then grouping algorithm is applied.

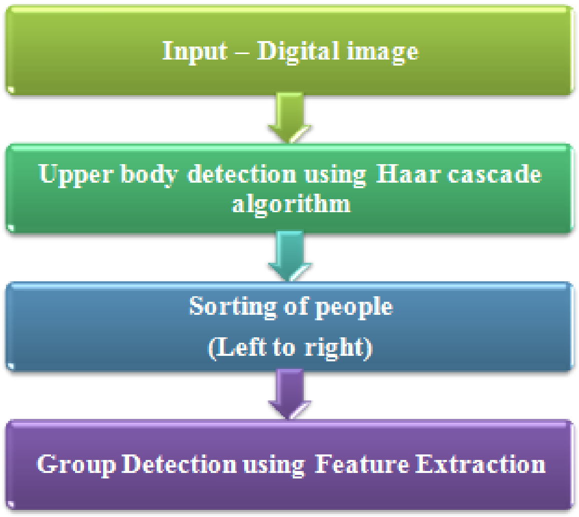

Input – Digital image
Upper body detection using Haar cascade algorithm
Sorting of people (Left to right)
Group Detection using Feature Extraction

Figure 3-8 Steps for algorithm3

3.3.1 *Feature extraction and grouping of persons*

Feature extraction is the process of generating features to be used in the selection and classification tasks. Feature extraction is most critical because the particular features made available for discrimination directly influence the efficacy of the classification task. The end result of the extraction task is a set of features, commonly called a feature vector, which constitutes a representation of the image. There are different types of features which extracted from an image. In case of an image having persons features extracted are like eyes, nose, and ear in case of face detection.

In this algorithm, after sorting of persons, features of the persons are extracted. Features extracted like front face, side face or ear and upper body which include shoulder and chest. Then face orientation is calculated using feature extraction. If front face is detected in an image, face orientation is considered to be zero .Side face or right ear is detected then face orientation is taken to be negative. For the positive orientation of face ,As we have already taken out upper body that means there will be face there, but as first two did not detect that face, that means this orientation will be positive. After calculating face orientation of all the persons in an image, then grouping algorithm is applied. In grouping algorithm, If the person is right facing, then orientation is positive, if the person is front facing, then orientation is zero, if the person is left facing, then face orientation is negative. Groups are cut out when the face orientation of let’s say ith person in n number of people is negative and face orientation of (i+1) th person is positive. The only thing which is to be taken care of is the face angle of first two people. If face angle of person1 and person2 is same then count (the number of groups) should be started from 1 and if both are different then count should be started from 2.

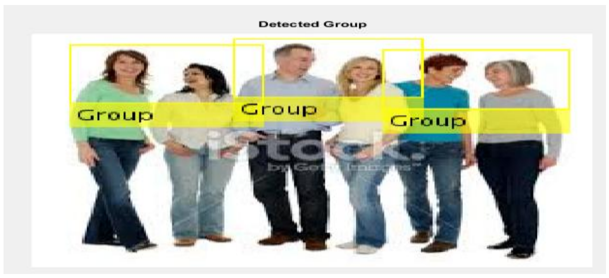


Figure 3-9 Group detection using feature extraction

4. BASIC SYSTEM AND FURTHER IMPROVEMENTS

This basic system proved to be a good starting point for tackling practical challenges in group detection in an image. This work can be used as a base for further research in group detection in an image. We have used three types of algorithms for group detection in an image.

The three types of algorithms used in group detection:

(a) Group detection based on distance

(b) Group detection based on template matching

(c) Group detection based on feature extraction

Group detection done with the help of these algorithms detects groups in an image linear dimensionally. From the comparison of these algorithms it would be clear which algorithm is efficient and better to use.

Comparison of algorithms

- Group detection based on distance uses the distance between the persons in image and distance is calculated on the basis of pixels i.e. pixel distance is calculated. By implementing this algorithm we come to know that accuracy is less than 50%.Accuracy is the degree to which the result of a measurement, calculation, or specification conforms to the correct value or a standard i.e. how to correctly we detect a group in an image and its execution time is good, it takes 5 seconds.

- Group detection based on template matching uses the templates stored in database. In this algorithm, template matching is done on the face of persons. Thirteen templates are created for defining face orientation from -90 to 90 degree. Each person is matched with thirteen templates and the best possible template is picked and the face orientation is decided. After calculating face orientation of all the persons in an image, then grouping of person's is done. By implementing this algorithm we come to know that it is more accurate than the previous one as its accuracy rate is more than 50% but this algorithm takes more execution time than the previous algorithm it takes nearly 1 minute.

- Group detection using feature extraction uses the concept of feature extraction from the objects present in an image. After each person's features like front face, shoulder, chest, side face or ear etc. extracted from image then face orientation is calculated using feature extraction. If front face is detected in an image, face orientation is considered to be zero .Side face or right ear is detected then face orientation is taken to be negative. For the positive orientation of face ,As we have already taken out upper body that means there will be face there, but as first two did not detect that face, that means this orientation will be positive. After calculating face orientation of all the persons in an image, then grouping of person's is done. . By implementing this algorithm we come to know that it has same accuracy rate as that of previous one i.e. more than 50% but this algorithm takes less execution time than the previous algorithm it takes nearly 5 sec.

What makes for a GOOD algorithm?

- It must work.
- It must complete its task in a finite (reasonable) amount of time.

Two or more algorithms that solve the same problem can be very different and still satisfy these two criteria. Therefore, the next step is to determine which algorithm is "best." There are generally two criteria used to determine whether one algorithm is "better" than another.

- Space requirements (i.e. how much memory is needed to complete the task).
- Time requirements (i.e. how much time will it take to complete the task).

Table 4-1 Comparison of algorithms

Type of algorithm	**Accuracy rate**	**Execution time**
Group detection based on distance	Less than 50%	5 sec
Group detection based on template matching	More than 50%	1 min
Group detection based on feature extraction	More than 50%	5 sec

From the above table, it is clear that group detection using feature extraction is best as its accuracy rate is good as compare to other algorithms and consumes less time(which is one of the factor to be an algorithm good) as compare to other algorithms.

5. CONCLUSION AND FUTURE WORK

We have used three different types of algorithm for detecting groups in an image and compared them on the basis of accuracy rate and execution time and come to know that group detection using feature extraction is best as its accuracy rate is good as compare to other algorithms and consumes less time (which is one of the factor to be an algorithm good) as compare to other algorithms. Group detection using these three algorithms detects groups in an image linear dimensionally. We can further improve our research work by doing group detection in an image multi- dimensionally and then compare with the previous algorithms and see the results which are good in case of accuracy, time and space.

6. REFERENCES

[1] M. Thonnat and N. Rota, "Image understanding for visual surveillance application," Third international workshop on cooperative distributed vision CDV-WS'99, Kyoto, Japan, pp. 51–82, Nov. 1999.

[2] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who, when, where, what: A realtime system for detecting and tracking people", IEEE International Conference on

Automatic Face and Gesture Recognition, pp. 222–227, April 1998.

[3] I. Haritaoglu, D. Harwood, and L.S. Davis, “Hydra: Multiple people detection and tracking using silhouettes”, IEEE International Workshop on Visual Surveillance, pp. 6–13, June 1999.

[4] D. Koller, J. Weber, and J. Malik, “Robust multiple car tracking with occlusion reasoning”, European Conference on Computer Vision, pp. 189–196, May 1994.

[5] N. Oliver, B. Rosario, and A. Pentland, “A Bayesian computer vision system for modeling human interactions”. International Conference on Vision Systems, January 1999.

[6] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for humancomputer interaction: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, pp. 677–695, 1997.

[7] A. Baumberg and D. Hogg, “An efficient method for contour tracking using active shape models”, IEEE Workshop on Motion of Nonrigid and Articulated Objects, pp.194–199, November 1994.

[8] N. Johnson and D. Hogg, “Learning the distribution of object trajectories for event recognition”, Image and Vision Computing, 14, pp. 609–615, 1996.

[9] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Realtime tracking of the human body”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, pp. 780–785, 1997.

[10] S.S. Intille, J.W. Davis, and A.F. Bobick, “Realtime closed world tracking”, IEEE Conference on Computer Vision and Pattern Recognition, pp. 697–703, June 1997.

[11] S.J. McKenna and S. Gong. Recognizing moving faces. In H. Wechsler, P. J. Phillips, V. Bruce, and F. Fogelman Soulie, editors, “Face Recognition: From Theory to Applications”, NATO ASI Series F, vol. 163, 1998.

[12] A.J. Lipton, H. Fujiyoshi, and R.S. Patil, “Moving target classification and tracking from realtime video”, DARPA Image Understanding Workshop, pp. 129–136, November 1998.

[13] T. Darrell, G. Gordon, M. Harville, and J.Woodfill, “Integrated person tracking using stereo, color, and pattern detection”, IEEE Conference on Computer Vision and Pattern Recognition, pp. 601–609, June 1998.

[14] C. Bregler, “Learning and recognizing human dynamics in video sequences”, IEEE Conference on Computer Vision and Pattern Recognition, pp. 568–574, June 1997.

[15] P.I. Wilson and J.Fernandez, “Facial feature detection using Haar classifiers”. Journal of Computing Sciences in Colleges archive, pp. 127-133, April 2006.