Gopika Mane, Krunank Panchal, Sanchita Sable / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 3, May-Jun 2013, pp.292-297

Analysis Of Various Techniques Used For Implementation Of Video Surveillance System

Gopika Mane*, Krunank Panchal, Sanchita Sable*****
*(Marathwada Mitra Mandal's Institute Of Technology (MMIT), Lohgaon, PUNE University, Pune-47)
** (Department of Information Technology (MMIT), PUNE University, Pune-47)
*** (Department of Information Technology (MMIT), PUNE University, Pune-47)

ABSTRACT

Video Surveillance software's are developed in order to provide the user of the software security from threats to data and other property from burglars. Image and Pixel matching algorithm are required for implementing the Video Surveillance System. This is especially required for computer security applications, where our algorithms are effectively used. These algorithms are useful for tracking an object in motion and classifying it on the bases of motion level, which would help in subsequent motion detection analysis. So that when an activity occurs in absence of user, the application detects it and performs the required action. This algorithm helps to represent graphically the amount of changes occurred in the environment.

Keywords - Cluster Matching, Image Comparisons, Pixel Comparisons, Pixel level Estimation, Similarity Threshold, and Sub-pixel level Estimation.

1. INTRODUCTION

Video surveillance systems produce huge amounts of data for storage and display. Long-term human monitoring of the acquired video is impractical and ineffective. Automatic abnormal motion detection system that can effectively attract operator attention and trigger recording is considered as the key to successful video surveillance in dynamic scenes, such as airport terminals. The video-surveillance architectures are used with limited computing power and available near the camera for compression and communication. The algorithm uses the macro block motion vectors that are generated in any case as part of the video compression process. Motion features are derived from the motion vectors.

The algorithm is modular, in the sense that different feature vectors are suggested and alternative probability density estimation or modeling methods are be used. The input to the algorithm is the set of macro-block motion vectors (such as intra-frame and intra-block flags) that are produced anyway by the compression proces which is an essential part of many modern video surveillance systems. The algorithm is used mainly for triggering video recording for later human analysis and then transmission to a human observer by detecting the 'object' that generated the abnormal motion.

In this paper, we represented an algorithm which works with the idea of building a model of the static scene (i.e. without moving objects) called background, and compares every frame of the sequence to this background with respect to discriminate the regions of abnormal motion, called foreground (the moving objects). Therefore we are interested with the algorithm that must work without or new static objects settling in the scene which means that the background must be kept temporally adaptive, so that there must be a *local* estimation for the background value, and these algorithms do not use much resource like computing power and memory. These used algorithms imply that the statistical measures on the temporal activity must be kept locally available in every pixel, and which needs to constantly updated. This algorithmic approach makes use of the single model like the previous frame or a temporal average for the background, and global threshold for decision. Also there are some background estimation methods that are based on the analysis of the histogram check of the values that are taken by each pixel within a fixed number of past frames, with the features like mean, median or mode of the histogram can be chosen to set the background value, and the foreground that can be discriminated by comparing the difference between the current frame and the background with the histogram check variance. Therefore the video surveillance system requires the recursive methods that do not keep in memory a histogram for each pixel, but rather a fixed number of estimates are computed recursively. This value of the variance is used directly afterward for the detection of moving area.

2. LITERATURE SURVEY OF ALGORITHMS:

2.1 Pixel Matching Algorithm

Pixel matching method takes the input as high dynamic range image that maps it to a limited range of luminance values reproducible with the display device (i.e. video surveillance system). The approach follows functionality of attempting to

construct a sophisticated model. The operation requires to be performed in three steps. In the first step, we need to estimate local adaptation luminance in the image at each point. After that in second step, the values are applied to the simple function so as to compress those values into the required display range. As there is possibility that the details of the important image can be lost during this process, and solution need to re-introduce details over the image in the final pass. If provided with an accurate description of the environment, these methods are capable of computing luminance values at each pixel which closely match those measured at corresponding points in the real scene.

2.1.1 Agglomerative Clustering Algorithm:

Agglomerative clustering builds the solution by initially assigning each point to its own cluster and then repeatedly selecting and merging pairs of clusters. Thus, it builds a hierarchical merging tree from the bottom (leaves) towards the top (root). The key parameter here is the criterion used for selecting clusters to be merged. We focus on the Group Average criterion, which measures the similarity of two candidate clusters as the average pair wise similarity between their members. Thus, the average-link criterion allows specifying the size or compactness of the resulting clusters. This property is very useful in building compact appearance clusters and makes the algorithm robust to outliers. A similarity threshold that produces visually compact Clusters only depend on the employed feature descriptors, thus can be estimated experimentally and used on different data sets. Another advantage of agglomerative methods is that given the clustering trace from a full hierarchical clustering, i.e. the indices of clusters merged in every step and the similarities between them, we can rebuild the clusters for a different similarity threshold at almost no computational cost.

The Algorithm is basic straightforward

1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
4. Closest two clusters are merged.
5. The proximity matrix needs to update
6. **Until** one single cluster remains

The main drawback of the agglomerative clustering algorithm is its $O(N2 \log N)$ run-time and $O(N2)$ space complexity. This comes from the requirement that clusters should be merged in decreasing order of similarity and that the distances must be recomputed after each agglomeration. In order to make agglomerative clustering applicable to large data sets, both complexities have to be reduced. [5][7]

2.1.2 Reciprocal nearest Neighbor Pair Algorithm:

The improved clustering method is based on the construction of reciprocal nearest neighbor pairs (RNN pairs), that is of pairs of points *a* and *b*, such that *a* is *b*'s nearest neighbor and vice versa. RNN is applicable to clustering criteria that fulfill reducibility property

$d(ci,cj) \leq \inf(d(ci, ck),d(cj ,ck)) \Rightarrow \inf(d(ci,ck),d(cj ,ck)) \leq d(ci \cup cj ,ck)$

Algorithm 1 Average-Link algorithm with RNNs for R points.

```
last ← −1
while R ≠ ∅ do
  if last < 0 then            // Initialize a new chain with a random point v ∈ R.
      last ← 0; Chain[last] ← v ∈ R; R ← R\{v}; Sim[last] ← 0;                (1)
  s ← findNearestNeighbor(Chain[last], R); sm ← sim(Chain[last],s)           (2)
  if sm > Sim[last] then      // No RNNs, add s to the chain.
      last ← last+1; Chain[last] ← s; R ← R\{s}; Sim[last] ← sm;              (3)
  else                        // Found RNNs → agglomerate the last two points in the chain
    if Sim[last] > SimThreshold then
      s ← agglomerate(Chain[last],Chain[last − 1]); R ← R∪{s}; last ← last − 2;  (4)
    else last ← −1            // Discard the current chain.
```

An amortized analysis shows that this algorithm has a computational complexity of $O(N2d)$ with only linear space requirements. The time complexity is high when N is large. Here we present a strategy to further improve also the run-time efficiency.[5][7]

2.1.3 Sub Pixel Mapping Algorithm:

This algorithm creates problem of accuracy when it undergoes mixed pixel in remote sensing classification. This algorithm consists of two stages:

1. Pixel level Corresponding Estimation:

In Pixel level Estimation, We detect corresponding point q0 with pixel level accuracy.

2. Sub Pixel level Corresponding Estimation:

In Sub Pixel level Corresponding Estimation, We recursively improve the sub pixel accuracy by adjusting location of sub pixel window alignment.[9][10]

Procedure for sub pixel level Estimation*:*

Input: Images I0(n1,n2) and J0(n1,n2), Reference point p0 in I0(n1,n2) given in pixel level accuracy, corresponding point q0 in J0(n1,n2) given in pixel level accuracy,

Output: Reference point p-1 in I0(n1,n2) given with pixel level accuracy. Corresponding point q0 in J0(n1,n2) given with pixel level accuracy.

Procedure for pixel level Estimation:

Input: Images I0(n1,n2) and J0(n1,n2), Reference point p0 in I0(n1,n2)

Output: corresponding point q0 in J0 (n1,n2)

2.1.4 Patch Matching Corresponding Algorithm:

Patch Match is one of the fast algorithms that is used to compute the approximate dense K-nearest neighbor correspondences between blotches of two image region. Patch Match is evaluated in three ways: (1) to find k nearest neighbors,(2) to search across rotations and scales, just for addition translations, and (3) to match distances using arbitrary descriptors, and not just with sum-of-squared-differences on patch colors. The Patch Match algorithm finds dense, global correspondences an order of magnitude faster than previous approaches, such as dimensionality reduction (e.g. PCA) combined with tree structures like kd-trees, VP-trees, and TSVQ.The algorithm finds an approximate nearest-neighbor in an image for every small (e.g. 9x9) rectangular patch in another image, using a randomized cooperative hill climbing strategy. Here, the basic algorithm finds only a single nearest-neighbor, at the same scale and rotation. The core algorithm is generalized and extended.

First, for problems such as object detection, denoising, and symmetry detection, we can detect multiple candidate matches for each query patch. Thus we extend the core matching algorithm to find k-nearest neighbors (k-NN) instead of only 1-NN. Second, for problems such as super-resolution, object detection, image classification, and tracking (at re-initialization), the inputs may be at different scales and rotations, therefore, it should extend the matching algorithm to search across these dimensions. Third, for problems such as object recognition, patches are insufficiently robust to changes in appearance and geometry, so we show that arbitrary image descriptors can be matched instead[4][5]

2.1.5 Tone Mapping Algorithm:

Tone mapping techniques that is used for image processing and computer graphics needs to map one set colors to another color set so as to order approximate the appearance of high dynamic range images approximate in the medium that has a dynamic limited range. All Print-outs, CRT or LCD monitors, and projectors have the dynamic limited range which is inadequate and reproduce the full range of intensities light which is present in natural scenes. Tone mapping suffers the problem with the reduction strong contrast received from the radiance scene to the range displayable that is preserving the details of image and appearance of the color that is important to appreciate the content of the original scene.[11]

Fig 1 Tone mapping in digital photography

2.1.6 K-Means Algorithm:

It is Partitioned clustering approach where each cluster gets associated with a center point (centroid). Cluster is assigned with each point that is the closest centroid .K needs to be specified as the number of clusters,

The basic algorithm is very simple

1: k points need to be selected as the initial centroids.

2: Repeat loop

3:Create k clusters by assigning points to the closest centroid.

4: Recomputed the centroid of each cluster.

5: Until the centroids don't change.

Mathematical module for K-Means

K-means is an algorithm used for partitioning (or clustering) *N* data points into *K* disjoint subsets *Sj* that contains *Nj* data points which are required to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S} | x_n - \mu_j |^2$$

Where x_n is a vector representing the *nth* data point and m_j is the geometric centroid of the data points in S_j

Complexity is O(n * K * I * d)

n=number of points,

K = number of clusters,

I = number of iterations,

d = number of attributes

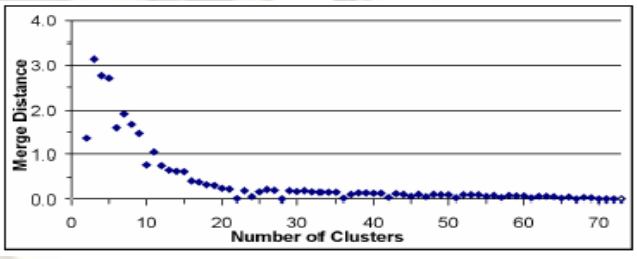

Fig 2 K-Means Algorithm

2.2 Image Matching Algorithm:

Image matching is an important fundamental task in a variety of image processing applications as motion analysis, image sequence analysis, stereo vision. These matching methods can estimate the displacement between two images and employs (1) an analytical function fitting technique to estimate the position of the correlation peak, (2) a windowing technique to eliminate the effect of periodicity, (3) a spectrum weighting

technique to reduce the effect of aliasing and noise.

2.2.1 Classic image check:

The classic image check is used to compare the value of color of every single expected pixel and actual pixel. As one of the expected pixels gets differed from the actual pixel then the check fails. The Tolerance option is required for checking images and then defining the tolerance required for comparing pixel values. This pixel based check is used and suitable if we want an exact match image that has tolerances minimal or any deviations. If the application renders as the component is not fully deterministic, then this algorithm is not appropriate.
Example-The classic image check does not alter the image, and therefore the result looks identical as per the original image

Fig 3 Classic image check algorithm

The classic image check is used for the 'Algorithms image comparison' attribute when empty.

2.2.2 Pixel based identity check:

This algorithm is similar to the classic image check algorithm, but it requires accepting the amount of pixels unexpected. Every pixel is split in to its three sub-pixels red, green and blue. After that it requires to checks every actual color value compared to the expected color value. The final result contains the amount of pixel identical and divided with the total pixel amount. The output calculated is checked with the expected value. When images are not rendered as fully deterministic but we need to accept unexpected pixels with certain percentage, then this algorithm may be used. It is not used, when the actual images are used to have distortions or shifts.
Example-The resultant image of the typical algorithm
Algorithm=identity; Expected=0.95; looks same as per the unique image as this algorithm does not consider image manipulation.

Fig 4 Pixel based identity check algorithm

Parameters
Algorithm=identity
Should use the 'Pixel based identity check' .
Expected
Expected probability needs to be defined which is valid between values 0.0 and 1.0.
Resize (optional)
Needs to be defining when the actual image needs to be resized before calculation to match the expected image size. Valid values may be "true" and "false".
Find (optional)
Required when an image-in-image search.

2.2.3 Pixel based similarity check:

This algorithm needs to be split in every pixel through its three sub-pixels which are red, green and blue. After that it checks every actual value color with the expected value color to calculate the similarity percentile. All the deviations percentile are added up and used to compute the ultimate result with the average deviation against all color values and all pixels values. The calculated result is compared with an expected value. When images are not rendered deterministic fully but we need to accept a certain deviation, then this algorithm can be used to fulfill this requirement. When we accept deviations for some pixels, the deviation average for an image is small, and this algorithm can be used to fulfill the purpose. It is not used, when the actual images are needed to shifts or distortions.
Example-The resultant image of the algorithm exemplary:
Algorithm=similarity; Expected=0.95; Looks identical as per the original image as this algorithm does not follow image manipulation.

Fig 5 Pixel based similarity check algorithm

Parameters
Algorithm=similarity
Should use the 'Pixel based similarity check'
Expected
Expected probability needs to be defined and valid values are between (0.0 and 1.0).
Resize (optional)
Required when the actual image needs to be resized before calculation to the size match of the expected image with valid values as ("true" and "false").
Find (optional)
Declares an image-in-image search.

2.2.4 Block based identity check:

This algorithm is used to partition the image into quadratic blocks as a selectable size. Each color value of these quadratic blocks is evaluated with the color values average pixels which the block contains. If the breadth or height of the image is not a multiple block size, then these blocks are at the right and bottom edge that will be cropped and can be weighted accordingly. With the concluding part the actual blocks are evaluated with

the expected blocks. The final results contain the amount of blocks identical and are divided with the total block amount. The algorithm purpose is to evaluate the image which differs for only some parts and still is identical for the remaining parts.
Example-The algorithm exemplary Algorithm=block; Size=10; Expected=0.95 Ravages in the following image:

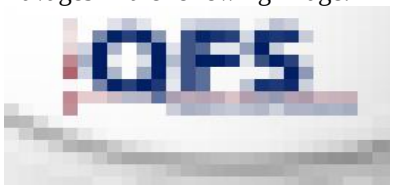

Fig 6 Block based identity check algorithm

Parameters
Algorithm=block
Should use the 'Block-based identity check' algorithm
Size
Size of each block is defined with valid values between 1 and the image size.
Expected
The minimum match possibility for the check to succeed is defined with valid values between 0.0 and 1.0.
Resize (optional)
When the actual image needs to be resized before calculation to the size match of the expected image is defined with valid values between ("true" and "false").
Find (optional)
Required when an image-in-image search

2.2.5 Block based similarity check:

This algorithm is used to partition the image as per the quadratic blocks with a selectable size. Every value of color blocks is evaluated with the color average values of the pixels that block contains. When the evaluated breadth or height of the image is not the multiple with the block size, then blocks contained at the right and bottom edge can be cropped and may be weighted as per it. Each expected block with color value is evaluated with the actual (original) block. These color values will be analyzed for similarity percentile. The final resultant contains all similarity average blocks with their weight taken into consideration. This algorithm is used for evaluating the images with similar variances color.
Example-The algorithm exemplary Algorithm=block similarity; Size=5; Expected=0.95
Ravages in the following image:

Fig 7 Block based similarity check algorithm

Parameters
Algorithm = block similarity
Should use the 'Block-based similarity check' algorithm
Size
Size of each block needs to be defined between valid values (1 and the image size).
Expected
The minimal match probability for the check to succeed should be defined as per the valid values between (0.0 and 1.0).
Resize (optional)
When the actual image needs to be resized before evaluation to match the size of the expected image as per the valid values between ("true" and "false").
Find (optional)
Should define an image-in-image search.

2.2.6 Histogram check:

The need to create a histogram check requires the image to be first broken down into its three build colors red, green and blue. After wards the values of the color for each pixel will be analyzed and partition them into a defined amount of categories (that is referred as buckets when we are talking about histograms). The actual level fill of each bucket can be compared with the expected level. The result of this algorithm will be compared as the relative frequencies of color categories. Histograms are used for many different scenarios. For example if the need to check tendencies color or increase brightness. At this situation histograms may not be used for evaluating rather than plain-colored images.
Example-The algorithm exemplary Algorithm=histogram; Buckets=64; Expected=0.95
Ravages in the following image:

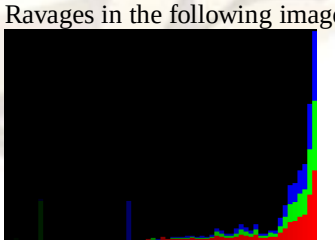

Fig 8 Histogram Check algorithm

Parameters
Algorithm=histogram
Image check should use a 'Histogram' algorithm.
Buckets
Need to define how many buckets are required between valid values as the power of (2 between 2 and 256).

Expected
The minimum match possibility for the check to succeed should be defined between valid values (0.0 and 1.0).
Resize (optional)
When an actual image needs to be resized before evaluation to match the size of the predictable image with thevalid values as ("true" and "false").
Find (optional)
When an image-in-image search is defined.

2.3 Cluster Matching Algorithm:

A new method for estimating displacements in computer imagery through cluster matching is presented. Without reliance on any object model, the algorithm clusters two successive frames of an image sequence based on position and intensity. After clustering, displacement estimates are obtained by matching the cluster centers between the two frames using cluster features such as position, intensity, shape and average gray-scale difference. The performance of the algorithm was compared to that of a gradient method and a block matching method. The cluster matching approach showed the best performance over a broad range of motion, illumination change and object deformation. In this study, clustering is used as a means to group pixels with similar feature‘s in each image frame; the displacements are estimated by matching clusters between two frames in the image sequence. The method should also be applicable to motion estimation for images containing deformable objects with varying brightness. First, pixels in each image frame are clustered the two successive frames are treated independently. Each of the resulting clusters tends to be spatially connected because of the position constraint and contains pixels of similar gray-scale values because of the intensity constraint. Second, clusters are matched between the two consecutive frames. In addition to the position and the intensity, the shape of each cluster and the average gray-scale difference between two clusters are used as matching criteria. The Displacement between the two matching cluster centers is assigned to every pixel in the cluster. This results in a dense set of displacement estimates.[10]

Mathematical module for Cluster Matching Algorithm

$$U = \sum_{s \in \mathbb{S}} [U_m(e(s)) + U_a(e(s), o(s))]$$

$U_m(e(s))$ is the called model energy and is designed to provide spatiotemporal regularity in the motion

$U_a(e(s), o(s)$ is called fittness energy and is designed to ensure a certain level of attachment to the input data, i.e. the observation.

3. CONCLUSION

This paper discuss various pixel matching algorithms and Image matching algorithms such as K-Means, Histogram Check and Cluster Matching Algorithm. These are required to implement different functionalities of Video Surveillance System Application.

ACKNOWLEDGEMENTS

We are thankful to the staff and students of Marathwada Mitra Mandal’s Institute of Technology, Lohagaon, Pune College for the support and encouragement from them

REFERENCES

[1] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: “PatchMatch: a randomized correspondence algorithm for structural image editing.” ACM Transactions on Graphics (Proc. SIGGRAPH) 28 (2009) 24
[2] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: “A database and evaluation methodology for optical ow”. In: Proc. ICCV. Volume 5. (2007)
[3] Buades, A., Coll, B., Morel, J.: “A non-local algorithm for image denoising.” In: Proc. CVPR. (2005) II: 60
[4] O. Boiman and M. Irani, “Detecting irregularities in im- ages and in video”, ICCV, 2005.
[5] S.Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, partbased representation.” *PAMI*, 26(11):1475–1490, 2004.
[6] J.K. Aggarwal and Q. Cai, “Human motion analysis: a re- view,” CVIU, Vol. 73, No. 3, pp. 428-440, 1999.
[7] J.Beis and D. Lowe, “Shape Indexing Using Approximate Nearest-Neighbor Search in High- Dimensional Spaces.” In *CVPR*, pages 1000–1006, 1997.
[8] V. Bhaskaran and K. Konstantinides, “*Image and Video Compression Standards: Algorithms and Architectures,*” Kluwer, 1997.
[9] O.D.Faugeras,”Three dimensional Computer Vision,” MIT Press, 1993.
[10] L.G Brown,”A Survey of image registration techniques,” ACM computing surveys ,vol.24,no.4,pp.325-376,Dec-1992
[11] J.L. Bentley, “Multidimensional binary search trees used for associative searching.” In *Communications of the ACM*, 18(9):509–517,1975