ISSN Print: 2394-7500
ISSN Online: 2394-5869
Impact Factor: 8.4
IJAR 2021; SP7: 73-80

Vivek Kumar
Assistant Professor, KSVCEM, Bijnor, Uttar Pradesh, India

Nidhi Bishnoi
Associate Professor
KSVCEM, Bijnor, Uttar Pradesh, India

Dr. Shueb Ali Khan
Professor, KSVCEM, Bijnor, Uttar Pradesh, India

Correspondence
Vivek Kumar
Assistant Professor, KSVCEM, Bijnor, Uttar Pradesh, India

(*Special Issue*)
"Twenty-First Century: Cultural and Economic Globalization"

Automated System for Identification of Moving Object Using Dip

Vivek Kumar, Nidhi Bishnoi and Dr. Shueb Ali Khan

DOI: https://doi.org/10.22271/allresearch.2021.v7.i7Sb.8679

Abstract
This project deals with the tracking and following of a single object in a sequence of frames and the velocity of the object is determined. Algorithms are developed for improving the image quality, segmentation, feature extraction and for deterring the velocity. Segmentation is performed to detect the object after reducing the noise from that scene. The object is tracked by plotting a rectangular bounding box around it in each frame. The velocity of the object is determined by calculating the distance that the object moved in a sequence of frames with respect to the frame rate that the video is recorded. The algorithms developed can also be used for other applications (real-time, object classification, etc.).

In this project, we use an inbuilt AVI file and then we show it after decompress it with the help of VIRTUALDUB1.9.10 software. After that, we apply all the above algorithms for tracking the object.

Keywords: Segmentation, AVI file, pan-tilt mechanism, dynamic template matching, etc.

Introduction

1.1 Motivation

The motivation behind this project is to develop software for tracking, the major application in security, surveillance and vision analysis. The developed software must be capable of tracking any single object moving in the frame and to implement on hardware that is capable of onboard calculations with high performance and low power consumption. This system might be useful for extending real-time surveillance or object classification.

1.2 Goal

The goal of this project is to develop an algorithm for the recognition of an object and tracking the moving object in a sequence of frames.

1.3 Outline of the project

The implementation of object tracking and its velocity determination is explained step by step in this report. In chapter 2, few approaches are implemented in real-time for object tracking and velocity determination are explained. In chapter 3, algorithms that are developed to accomplish this project are explained. In chapter 4, the implementation part which includes the logical approach of tracking done for the project is described.

Object tracking is central to any task related to vision systems. Tracking objects can be complex [10] due to loss of information caused by the projection of the 3D world on a 2D image, noise in images, complex object motion, non-rigid or articulated nature of objects [14], partial and full object occlusions, complex object shapes, scene illumination changes, and real-time processing requirements. Tracking is made simple by 10 imposing constraints on the motion and/or appearance of objects. In our application we have tried to minimize the number of constrains on the motion and appearance of the object. The only constraint on the motion of the object is that it should not make sudden change in the direction of motion

while moving out of the viewing range of the camera. Unlike other algorithms [13] the present algorithm is capable of handling the entry and exit of an object. Also [14], [15] no colour information is required for tracking an object. There is no major constraint on the appearance of the object though an object which is a little brighter than the background gives better tracking results. There are three key steps in implementation of our object tracking system:

- Detection of interesting moving objects,
- Tracking of such objects from frame to frame,
- Analysis of object tracks to automate the pan-tilt mechanism

1.3.1 Frame differencing

The system first analyses the images, being grabbed by the camera, for detection of any moving object. The *Frame Differencing* algorithm [11], [16] gives as output the position of the moving object in the image. This information is then used to extract a square image template (of fixed size) from that region of the image. The templates are generated as and when the appearance of the object changes significantly.

1.3.2 Dynamic template matching

The newly generated template is then passed on to the tracking module, which starts tracking the object taking the template as the reference input. The module uses template-matching [12], [15] to search for the input template in the scene grabbed by the camera. If the object is lost while tracking (signifying that the object has changed its appearance) a new template is generated and used. Since the image templates, being used for matching, are generated dynamically the process is called Dynamic Template Matching.

1.3.3 Pan-tilt mechanism

The movement of the object is analyzed for automation of the Pan-Tilt mechanism. Depending upon the movement of the object the pan-tilt mechanism is operated to keep the object in the camera's view. 11 In adaptive background subtraction method, a reference background is initialized at the start of the system with the first few frames of video and updated to adapt to short and long term dynamic scene changes during the operational period. At each new frame, foreground pixels are detected by subtracting the intensity values from the background and filtering the absolute value of the differences with a dynamic threshold per pixel. The reference background and the threshold values are updated by using the foreground pixel information. The detected foreground pixels usually contain noise due to image acquisition errors, small movements like tree leaves, reflections and foreground objects with textures colored similar to the background. These isolated pixels are filtered by the use of a sequence of morphological operations dilation and erosion. After this step, the individual pixels are grouped and labeled by using a two pass component labeling algorithm to create connected moving regions. These regions are further processed to group disconnected blobs and to eliminate relatively small sized regions. After grouping, each detected foreground object is represented with its bounding box, area, center of mass and color histogram which will be used in later steps.

After segmenting moving pixels from the static background of the scene, connected regions are classified into predetermined object categories human, human group and vehicle. The classification algorithm depends on the comparison of the silhouettes of the detected objects with pre-labeled (classified) templates in an object silhouette database. The template database is created by collecting sample object silhouettes from sample videos and labeling them manually with appropriate categories. The silhouettes of the objects are extracted from the connected foreground regions by using a contour tracing algorithm [19]. Next, the distance between each boundary pixel and the center of mass point is calculated to create a distance signal starting from the top pixel and continuing clock-wise until reaching the same pixel. The distance signals are first normalized to be of the same length, then smoothed and finally normalized again to cover the same area. The comparison metric used in matching the templates with the detected objects are the L1 distance of normalized silhouette distance signals. The class of the template silhouette with minimum distance from the detected object's silhouette is assigned to the object's class. Temporal tracking information is used to support classification decision. 12 As the final step in the presented system, the tracking algorithm tracks the detected objects in successive frames by using a correspondence-based matching scheme. It also handles multi-occlusion cases where some objects might be fully occluded by others. It uses 2D object features such as position and size to match corresponding objects from frame to frame. It keeps color histograms of detected objects in order resolve object identities after a split of an occlusion group. The output of the tracking step supports both motion segmentation and object classification steps.

1.4 Tools used

MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is a numerical computing environment and programming language. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN. The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics,

engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB system

The MATLAB system consists of five main parts:

1. Development Environment

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

2. The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms. So, this library is helpful for object tracking process.

3. The MATLAB Language

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

4. Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

5. The MATLAB Application Program Interface (API)

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

MATLAB image processing toolbox

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment. The toolbox supports a wide range of image processing operations, including

- Spatial image transformations
- Morphological operations
- Neighborhood and block operations
- Linear filtering and filter design
- Transforms
- Image analysis and enhancement
- Image registration
- De blurring Region of interest operations.

Many of the toolbox functions are MATLAB M-files, a series of MATLAB statements that implement specialized image processing algorithms.

1.5 Main applications

Object tracking is an important task within the field of computer vision. The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms. Object tracking from video sequence is a very important topic and has various applications in video compression, surveillance, robot technology, etc. In many applications, the focus is on tracking moving objects. The misdetections in the object detection process make this process a difficult one. Object tracking is required in many vision applications such as human-computer interfaces, video communication/compression, road traffic control, security, and surveillance system. Often the goal is to obtain a record of the trajectory of moving single or multiple targets over time and space. Object tracking from video sequence is a challenging task because of the large amount of data used and the common requirement for real time computation.

The use of object tracking is pertinent in the tasks of:

- motion-based recognition, that is, human identification based on gait, automatic object detection, etc.
- automated surveillance, that is, monitoring a scene to detect suspicious activities or unlikely events;
- video indexing, that is, automatic annotation and retrieval of the videos in multimedia databases;
- human-computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc.;
- Traffic monitoring, that is, real-time gathering of traffic statistics to direct traffic flow.
- Vehicle navigation that is, video-based path planning and obstacle avoidance capabilities.

2.1 Moving object detection

Each application that benefit from smart video processing has different needs, thus requires different treatment. However, they have something in common: moving objects. Thus, detecting regions that correspond to moving objects such as people and vehicles in video is the first basic step of almost every vision system since it provides a focus of attention and simplifies the processing on subsequent analysis steps. Due to dynamic changes in natural scenes such as sudden illumination and weather changes, repetitive motions that cause clutter (tree leaves moving in blowing wind), motion detection is a difficult problem to process reliably. Frequently used techniques for moving object detection are background subtraction, statistical methods, temporal differencing and optical flow whose description are given below.

2.2 Background subtraction

Background subtraction is particularly a commonly used technique for motion segmentation in static scenes. It

attempts to detect moving regions by subtracting the current image pixel-by-pixel from a reference background image that is created by averaging images over time in an initialization period. The pixels where the difference is above a threshold are classified as foreground. After creating a foreground pixel map, some morphological post processing operations such as erosion, dilation and closing are performed to reduce the effects of noise and enhance the detected regions. The reference background is updated with new images over time to adapt to dynamic scene changes. There are different approaches to this basic scheme of background subtraction in terms of foreground region detection, background maintenance and post processing. In Heikkila and Silven uses the simple version of this scheme where a pixel at location (x, y) in the current image It is marked as foreground if $|It(x, y) - Bt(x, y)| > T$ is satisfied where _ is a predefined threshold. The background image BT is updated by the use of an Infinite Impulse Response (IIR) filter.

The foreground pixel map creation is followed by morphological closing and the elimination of small-sized regions. Although background subtraction techniques perform well at extracting most of the relevant pixels of moving regions even they stop, they are usually sensitive to dynamic changes when, for instance, stationary objects uncover the background (e.g. a parked car moves out of the parking lot) or sudden illumination changes occurs.

2.2.1 Statistical methods

More advanced methods that make use of the statistical characteristics of individual pixels have been developed to overcome the shortcomings of basic background subtraction methods. These statistical methods are mainly inspired by the background subtraction methods in terms of keeping and dynamically updating statistics of the pixels that belong to the background image process. Foreground pixels are identified by comparing each pixel's statistics with that of the background model. This approach is becoming more popular due to its reliability in scenes that contain noise, illumination changes and shadow The W4 system uses a statistical background model where each pixel is represented with its minimum and maximum intensity values and maximum intensity difference between any consecutive frames observed during initial training period where the scene contains no moving objects.

After thresholding, a single iteration of morphological erosion is applied to the detected foreground pixels to remove one-pixel thick noise. In order to grow the eroded regions to their original sizes, a sequence of erosion and dilation is performed on the foreground pixel map. Also, small-sized regions are eliminated after applying connected component labeling to find the regions. The statistics of the background pixels that belong to the non-moving regions of current

2.3 Object classification

Moving regions detected in video may correspond to different objects in real-world such as pedestrians, vehicles, clutter, etc. It is very important to recognize the type of a detected object in order to track it reliably and analyze its activities correctly. Currently, there are two major approaches towards moving object classification which are shape-based and motion-based methods. Shape-based methods make use of the objects‟ 2D spatial information whereas motion-based methods use temporally tracked features of objects for the classification solution.

2.3.1 Motion-based classification

Some of the methods in the literature use only temporal motion features of objects in order to recognize their classes. In general, they are used to distinguish non-rigid objects (e.g. human) from rigid objects (e.g. vehicles). The method proposed in [8] is based on the temporal self-similarity of a moving object. As an object that exhibits periodic motion evolves, its self-similarity measure also shows a periodic motion. The method exploits this clue to categorize moving objects using periodicity. Optical flow analysis is also useful to distinguish rigid and non-rigid objects. A. J. Lipton proposed a method that makes use of the local optical flow analysis of the detected object regions. It is expected that non-rigid objects such as humans will present high average residual flow whereas rigid objects such as vehicles will present little residual flow. Also, the residual flow generated by human motion will have a periodicity. By using this cue, human motion, thus humans, can be distinguished from other objects such as vehicles.

2.4 Object tracking

Tracking is a significant and difficult problem that arouses interest among computer vision researchers. The objective of tracking is to establish correspondence of objects and object parts between consecutive frames of video. It is a significant task in most of the surveillance applications since it provides cohesive temporal data about moving objects which are used both to enhance lower level processing such as motion segmentation and to enable higher level data extraction such as activity analysis and behavior recognition. Tracking has been a difficult task to apply in congested situations due to inaccurate segmentation of objects. Common problems of erroneous segmentation are long shadows, partial and full occlusion of objects with each other and with stationary items in the scene. Thus, dealing with shadows at motion detection level and coping with occlusions both at segmentation level and at tracking level is important for robust tracking. Tracking in video can be categorized according to the needs of the applications it is used in or according to the methods used for its solution. Whole body tracking is generally adequate for outdoor video surveillance whereas objects‟ part tracking is necessary for some indoor surveillance and higher level behavior understanding applications.

Object detection and tracking

The overview of our real time video object detection, classification and tracking system is shown in Figure 3.1. The proposed system is able to distinguish transitory and stopped foreground objects from static background objects in dynamic scenes; detect and distinguish left and removed objects; classify detected objects into different groups such as human, human group and vehicle; track objects and generate trajectory information even in multi-occlusion cases and detect fire in video imagery. In this and following chapters we describe the computational models employed in our approach to reach the goals specified above.

Our system is assumed to work real time as a part of a video-based surveillance system. The computational complexity and even the constant factors of the algorithms we use are important for real time performance. Hence, our

decisions on selecting the computer vision algorithms for various problems are affected by their computational run time performance as well as quality. Furthermore, our system‟s use is limited only to stationary cameras and video inputs from Pan/Tilt/Zoom cameras where the view frustum may change arbitrarily are not supported. The system is initialized by feeding video imagery from a static camera monitoring a site. Most of the methods are able to work on both color and monochrome video imagery. The first step of our approach is distinguishing foreground objects.

5.1.1 Algorithm for object detection module

1. Grab the ith frame *fi* (8 bit gray-scale)
2. Retrieve the (i-3)th frame *fi-3* from the image buffer. The image buffer is an array of image variables which are used for temporary storage of frames. The array is programmed to behave as a queue with only three elements at any given point of execution.
3. Perform *Frame Differencing Operation* on the ith frame and the (i-3)th frame where the resultant image is represented as
   fr (8 bit gray-scale)
   fr = fi – fi-3 (2)

Here, it is noticeable that instead of subtracting the ith frame from the (i-1) the frame we are subtracting the ith frame from the (i-3)th frame. This has been done taking into consideration that even slow moving objects should detected. It has been observed that image subtraction on consecutive frames detects only fast moving objects (objects whose position changes noticeably from one frame to other). Such a method fails to detect slow moving objects. Therefore to remove this limitation we subtract ith frame from the (i-3)th frame to ensure a detection which is independent of speed.

4. Perform the *Binary Thresh-holding Operation* on *fr* separating the pixels corresponding to the moving object from the background. This operation also nullifies any inaccuracies introduced due to the camera flickering. The result of this operation is a binary image, *fb,* wherein only those pixels are set as „1‟ which correspond to the moved object. In the thresh-holding technique a parameter called the brightness threshold (T) is chosen and applied to the image f[m,n] as follows:
   IF f[m,n]≥T fb[m,n]=object=1 ELSE fb[m,n]=background=0 This version of the algorithm assumes that we are interested in light objects ona dark background. For dark objects on a light background we would use:
   IF f[m,n]≤ T fb[m,n]=object=1
   ELSE fb[m,n]=background=0
   While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives. In our case we are using fixed threshold (a threshold that is chosen independently of the image data). As our main objective is to separate the object pixels from that of the background this approach gives fairly good results.
5. Perform an *Iterative Mathematical Morphological Erosion Operation* on *fb* to remove really small particles from the binary image. The result of this step is again a binary image, *fbb*. This step ensures that small insignificant movements in the background are ignored ensuring better object detection.
6. Calculate the center of gravity (COG) of the binary image *fbb*. The result of this operation is a set of two integers C(cog_x, cog_y) which determines the position of the moving object in the given scene. The COG is calculated by:
   cog_x =cog_y + x (3a)
   cog_y = cog_y + y (3b)
   Total = Total + 1 (3c)
   For each pixel where x, y is the current pixel location. The resulting COG is then divided by the Total value:
   cog_x = cog_x/Total (3d)
   cog_y = cog_y/Total (3e)
   To result in the final x, y location of the COG.
7. Transfer the positional information C(cog_x, cog_y) to the object tracking module.
8. Store the frame *fi* in the image buffer and discard the frame *fi-3*
9. Increment *i* by 1
10. Goto (Step 1)

Algorithm used

This chapter explains the algorithms to track the single object and to estimate the velocity of moving object. The sequential approach to track the moving objects and the velocity of objects are as follow.

- Noise removal: To improve the image quality
- Segmentation: To separate multiple regions in image
- Feature Extraction: To analyze the regions in image
- Tracking: Analyzing the position, velocity and moving direction of object

6.1 Noise removal

A noise removal algorithm is developed by using the median filter as explained in section 2.1.3. Algorithm for noise removal is explained as follow

1. Read the input image
2. for (present position=initial position: final position)

a) Scan all the surrounding elements
b) Use bubble sort technique to sort all the values
c) Calculate the median of it

3. Assign the value to the 'present position' of the input image

- Initial position = 2nd row, 2nd column of an image resolution
- Final position = (n-1) row, (n-1) column of an image resolution
- Present position = the pixel value and it varies from initial to final position

Values are taken from 2nd row-2nd column to (n-1) row-(n-1) column because we need to scan all the surrounding elements of each pixel.

6.2 Segmentation

To perform the segmentation operation, frame difference algorithm is implemented as it takes less processing time. Frame difference algorithm performs separation of two sequential frames and it is explained in detail in section 2.2.3[8]. Algorithm for the segmentation is explained as follow

1. Read the input images
2. for (present position=initial position: final position)

a) Difference between the pixels values at present position of two images is calculated
b) Calculate the absolute value
c) Store the difference in new image at same pixel position that is at present position.

6.3 Feature extraction

Every object has a specific feature which is used to visualize the object and used for tracking. After performing the segmentation, a rectangular bounding box is plotted with the dimensions of the object produced in the residual image. Section 2.3.2 explains a clear view on bounding box. Algorithm for the bounding box is as followed

1. Read the image difference
2. for (present position=initial value: final value) of Y resolution
3. for (present position=initial value: final value) of X resolution
a) calculate the sharp change in intensity of image from top and bottom
b) store the values in an array
4. Height of the bounding box is = bottom value- top value
5. for (present position=initial value: final value) of X resolution
6. for (present position=initial value: final value) of Y resolution
a) calculate the sharp change in intensity of image from left and right
b) store the values in an array
7. Width of the bound box = right value - left value
8. Using the dimensions, draw boundary to the image Initial value: the starting position of the pixel in an image. Final value: the ending position of the pixel in an image.
Height = (bottom: value - top: value)/2
Width = (right: value - left: value)/2
9. Add the height value with the top value and store it in a variable like mid top
10. Add the width value to the left value and store it in a variable like mid left
11. Assign the max intensity to the pixel at pixel value at (mid top, mid left)

6.3.1 Distance

The distance travelled by the object is determined by using the centroid. It is calculated by using the Euclidean distance formula. The variables for this are the pixel positions of the moving object at initial stage to the final stage. Algorithm for calculating distance is explained as follow

1. Read the centroid position of each image.
2. Calculate the distance between two centroid images
3. for (present position=initial value: final value) of X resolution
4. for (present position=initial value: final value) of Y resolution
5. Calculate change in distance by

Distance= ((X2 - X1)2 + (Y 2 - Y 1)2)1/2

Where X1=previous pixel position and X2=present pixel position in width

Y1=previous pixel position and Y2=present pixel position in height

6. Store all the distance values in an Array.

6.3.2 Velocity

Velocity of moving object is determined using the distance travelled by the centroid to the frame rate of the video explains the velocity in 2-dimension. Algorithm for calculating velocity is explained as follow

1. Read the distance travelled by the object
2. Velocity = distance_travelled/frame_rate
3. Save the value in an array
4. The velocity of moving object in the sequence frames is defined in pixels / second

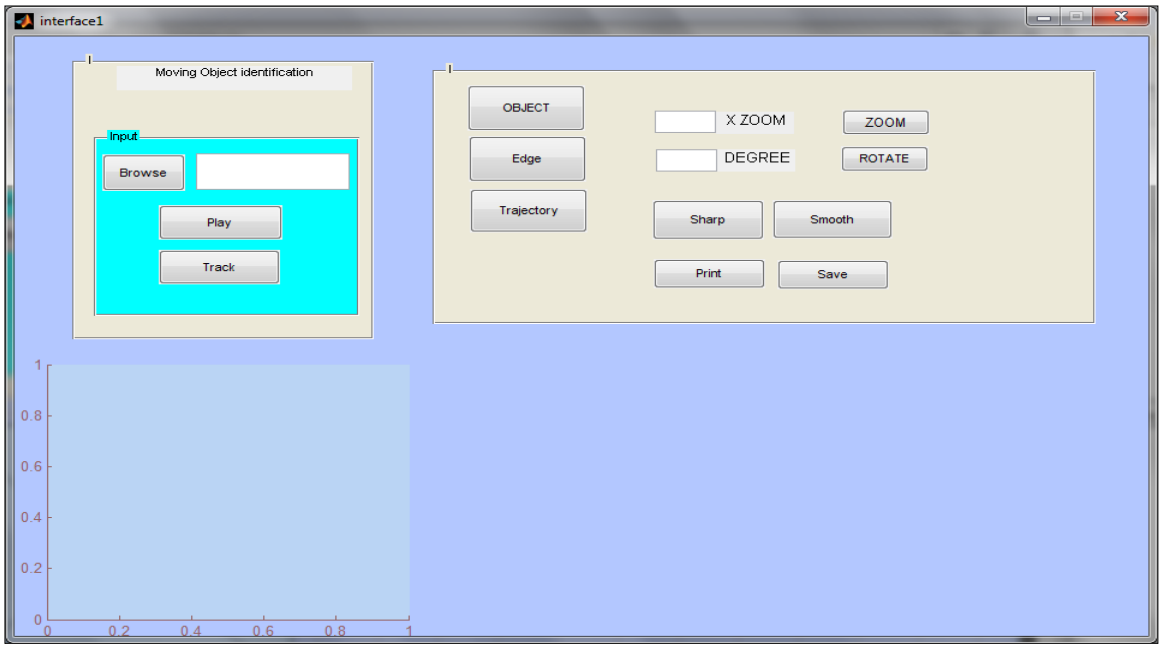

Fig 1: Output screen

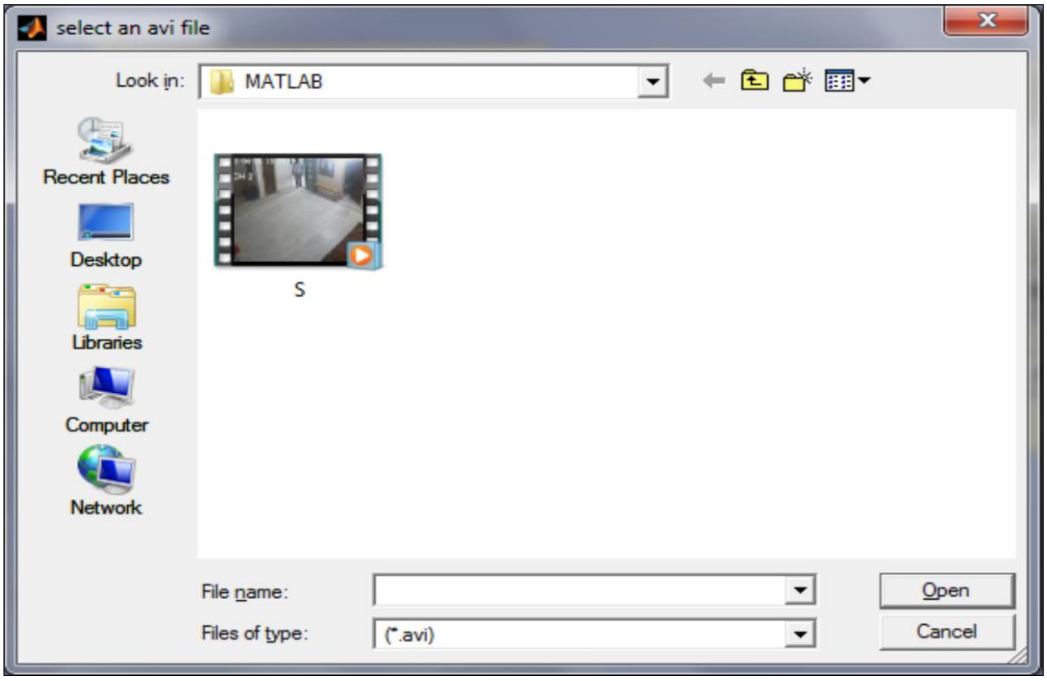

Fig 2: Browse Window (only AVI)

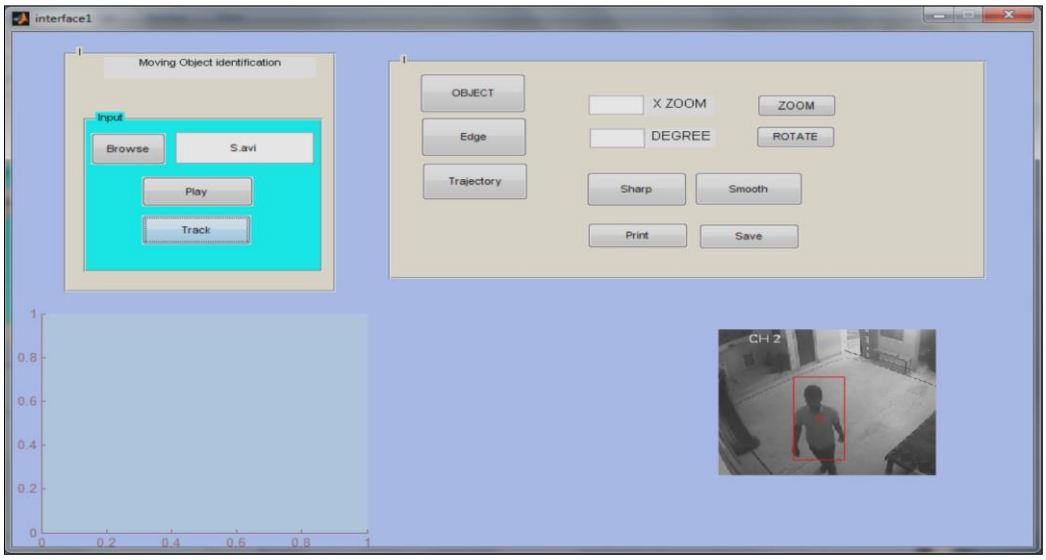

Fig 3: Tracking

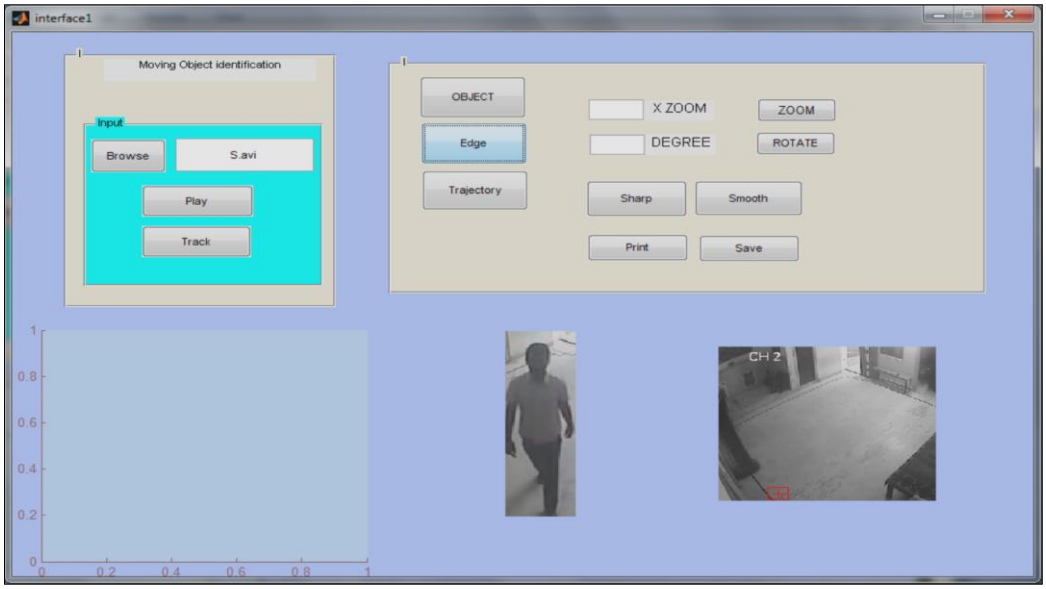

Fig 4: Object Detection

Conclusion

In this report, we propose an efficient algorithm for detecting a moving object using background elimination technique. Initially we compute the frame differences (FD) between frames Fi and Fi+k. The frame differences obtained are then compared with one another which help in identifying the stationary background image. The moving object is then isolated from the background. In the post processing step, the noise and shadow regions present in the moving object are eliminated using a morphological gradient operation that uses median filter without disturbing the object shape. This could be used in real time applications involving multimedia communication systems.
Other image processing technique or mechanism can be incorporated to increase the robustness and performance of this project. Object tracking can be incorporated in the algorithm to increase the robustness of the detection process. The tracking algorithm can be done using motion segmentation, which segments moving objects from the stationary background. A discrete feature-based approach is applied to compute 43 displacement vectors between consecutive frames. With this, the moving object can be detected as well as their associated tracks.

References

1. Alper Yilmaz, Omar Javed, Mubarak Shah. Object tracking: A survey. ACM Comput. Surv. 2006;38(4):13.
2. Ashley Walker Robert Fisher, Simon Perkins, Erik Wolfart. Gaussian smoothing in world wide web. http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm, 2003.
3. Ashley Walker Robert Fisher, Simon Perkins and Erik Wolfart. Mean _lter in world wide web. http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm, 2003.
4. Bar-Shalom Y, Foreman T. Tracking and Data Association. Academic Press Inc, 1988.
5. Berthold KP Horn, Brian G Schunck. Determining optical _ow. ARTIFICIAL INTELLIGENCE, 1992, 389-407.
6. Kamath C, Gyaourova A, Cheung SC. Block matching for object tracking. https: // computation. llnl. gov/ casc/ sapphire/ pubs/ UCRL-TR-200271. pdf, 2003.
7. Comaniciu D, Ramesh V, Meer P. Kernel-based object tracking. IEEE Trans. Patt. Analy. Mach. Intell. 2003;25:564-575.
8. Murray D, Basu A. Motion tracking with an active camera, IEEE Trans. Pattern Anal. Machine Intell. 1994;16:449–459.
9. David S. Bright. Removing shot noise from face image. http://www.nist.gov/lispix/imlab/noise/shotfc.html, 2004.
10. Mike Spann. Image segmentation in world wide web. http://www.eee. bham.ac.uk/spannm/Teaching%20docs/Computer%20Vision%20Course/Image% 20Segmentation.ppt, Course resources for 2008.
11. Edward R. Dougherty. Mathematical Morphology in Image Processing, 1993.
12. Fieguth P, Terzopoulos D. Color-based tracking of heads and other mobile objects at video frame rates. In IEEE Conference on, 1997.
13. Jain R. Nagel H. On the analysis of accumulative difference pictures from image sequences of real world scenes. IEEE Trans. Patt. Analy. Mach. Intell. 1979;1(2):206–214.
14. Martin Mangard. Motion detection on embedded smart cameras. Master's thesis, Technische Universität Graz, 2006.
15. Rafael Gonzalez C, Richard Woods E. Digital Image processing, Second Edition. Prentice Hall International, 2002.
16. Richard YD, John XU, Allen G, Jesse Jin S. Robust realtime tracking of non-rigid objects, Proceedings of the Pan-Sydney area workshop on Visual information processing, 2004, 95-98.
17. Zhou YT, Venkateswar V, Chellappa R. Edge detection and linear feature extraction using a 2-d random _eld model. IEEE Trans. Pattern Anal. Mach. Intell. 1989;11(1):84-95.
18. [19] YILMAZ, A., JAVED, O., AND SHAH, M. 2006. Object