CHAPTER: 1
INTRODUCTION
1. INTRODUCTION

1.1 About Video Surveillance

As a new technology developed on the basis of image processing, moving object detecting is one of the important research subjects in application fields such as computer vision and video information processing. It has found extensive applications in video compression, target recognition, intelligent monitoring, video retrieval, human computer interaction, traffic monitoring and biomedicine, etc. The procedure of moving object detecting is to decide whether there exist objects moving in video, and to position the target basically. The accuracy rate of object detection exerts great influence on the tracking and recognition in next steps. The application of matrix operation-based computer software MATLAB in image processing will greatly improve the efficiency of moving object detecting. This paper presents the study on the implementation of MATLAB-based moving object detecting algorithm, in which frame difference algorithm was chosen as an example for detect moving object in video from the aspects of video acquisition, image type conversion, image gray-level conversion, image filtering and image segmentation.

1.2 Objective of the Video Surveillance

As video sequence consists of frame sequences which have certain temporal continuity, the detection for moving object in video is conducted in a way that frame sequences are extracted from the video sequence according to a definite cycle. Therefore, moving object detecting has something similar to object detection in still images. Only moving object detecting is more relying on the motion characteristics of objects, i.e. the continuity of time, which is the difference between moving object and object detection in still images. The method frequently used in moving object detecting is video sequence analysis. Two or more frames acquired at different time contain the information about relative motion between an imaging system and a scene. This information is in forms of the gray and color variation between frames, or the location and property variation of marks such as dots, line segments and areas, etc. Therefore, the information about motions can be obtained through analysis and processing of images acquired at different time.
CHAPTER: 2
LITERATURE SURVEY
2. LITERATURE SURVEY

2.1 Existing System

The method frequently used in moving object detecting is video sequence analysis. Two or more frames acquired at different time contain the information about relative motion between an imaging system and a scene. This information is in forms of the gray and color variation between frames, or the location and property variation of marks such as dots, line segments and areas, etc. Therefore, the information about motions can be obtained through analysis and processing of images acquired at different time. Video sequence analysis methods can be classified into three types: optical flow method, background difference method and adjacent frame difference method. Optical flow reflects the image variation caused by motions in a definite time interval. The motion field of images is estimated to incorporate similar motion vectors into moving object. Solving transcendental equations is required in optical flow method, the calculation is both complex and extremely sensitive to noise, the amount of calculation is large, and the real-time performance and the practicability is poor. So this method is difficult to be used in real-time video processing.

Background difference method is a technique for detecting the motion area by using the difference between the current image and the background image. An image is divided into foreground and background in this method. The background is modeled, and the current frame and the background model are compared pixel by pixel. Those pixels according with the background model are labeled as the background, while others are labeled as the foreground. Background subtraction is a common method in moving object detecting, which is used more often in situations with relatively still background. This method has low complexity, however, acquired background images become sensitive to scene changes caused by illumination and external conditions as time goes on. Many fake C dots can emerge, which affects object detection. The mechanism of refreshing background reference frames needs to be added in under uncontrolled environment. Moreover, it doesn't fit camera motions or fit with conditions with large background Grey variation.
Generally an efficient technique must be able to give accurate and reliable results in any conditions. It must be able to handle most of the vital cases and should provide reliability. Generally the background methods contain two steps. They are

- Proper updation of background reference model must be done
- Correct subtraction between present image and background reference model has to performed.

Lucia Maddalena, Alfredo Petrosino presents an overview of some of the motion segmentation techniques. One among them is temporal differencing which takes into account differences in consecutive sequence frames, and it allows to discern static objects which have null as difference from moving objects which have the difference as non-null value. This approach is very adaptive to dynamic environments, but it is strictly dependent on the velocity of moving objects in the scene.

The background subtraction is the most common and efficient method to tackle this problem. It is based on the comparison of the current frame in the video to a reference background, including information on the scene without moving objects in a pixel-by-pixel fashion. It is a popular method for motion segmentation, especially under those situations with a relatively static background. It is simple and does not depend on the velocity of moving objects, but extremely sensitive to changes in dynamic scenes derived from lighting and extraneous events etc. Also the processing of outdoor sequences becomes more difficult in this case. Therefore, it is highly dependent on a good background model.

All the above are the conventional methods that are used for the motion segmentation. The other widely used background subtraction techniques are frame difference, mean filter and mixture of Gaussians. Each method has its own advantages and shortcomings. In the case of Mean Filter, for calculating the image containing only the background, a series of preceding images are averaged. For calculating the background image at the instant $t$,

$$B(x,y) = \sum_{i=1}^{N} V(x, y, t-i)$$

Where $N$ is the number of preceding images taken for averaging.
The drawback is mean and median background models have relatively high memory requirements. The two main methods for background subtraction apart from mean filter are described as

2.1.1 Frame Differencing:

As depicted by the name itself, frame differencing involves taking the difference between two frames and using this difference to detect the object. The current approach is a two step process. First, the object is detected using frame differencing. Then this detected object is compared with the ground truth to learn the reliability of this approach. Two consecutive frames are loaded from a given sequence of video frames. These color frames are converted to gray scale intensity. Nobuyuki Otsu et al. Otsu’s Method is used to determine the threshold value of the gray scale images. The main idea in this method is to find the threshold that minimizes the weighted within-class variance. This turns out to be the same as maximizing the between-class variance. Also, the threshold value is determined such that the pixel values on either side of this value are established to be either a background or a foreground pixel. The algorithm assumes that the image to be threshold contains two classes of pixels or bi-modal histogram (i.e., foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal. Following Otsu’s method, the two consecutive gray scaled images are used to calculate the difference and their absolute difference is used to identify the movement between frames. The noise collected due to differencing is removed by applying the threshold value to the images. Pixels below the threshold are removed from the differenced frame leaving behind required object. The absolute difference between two frames needs to be greater than the threshold for the object to be detected.

\[ |f1-f2| > T \]

Here f1 is the initial frame, f2 is the following frame and T is the threshold value. Here the background is estimated to be the previous frame and the subtraction is performed as follows.
Background subtraction equation then becomes:

\[ B(x, y, t) = I(x, y, t) - I(x, y, t-1) \]

The frame difference equation can be described as

\[ |I(x, y, t) - I(x, y, t-1)| > Th \]

Depending on the object structure, speed, frame rate and global threshold, this approach may or may not be useful and the accuracy of this method depends on the speed of movement in the scene. Faster movements may require higher thresholds. The drawbacks of the various previous approaches. The drawbacks of frame difference are

Limitations:
1. It evidently works only in particular conditions of objects’ speed and frame rate.
2. Very sensitive to the threshold Th and there is one global threshold Th for all pixels in the image.

Hence the above two approaches will not give good results in the following conditions:
1. If the background is bimodal,
2. If the scene contains many, slowly moving objects (mean & median),
3. If the objects are fast and frame rate is slow (frame differencing), and if general lighting conditions in the scene change with time.

The various techniques and their related steps that are used for the foreground detection of moving objects. After the classification of foreground moving objects the next step is to look for the unwanted noises and detections that are obtained. The output of foreground detection contains some noise. The various factors that are responsible for the noise that will be created are clearly mentioned in this paper. Some of them are camera noise, reflectance noise etc., Hence in order to clear off this unwanted noise pixel level processing is needed and the various methods that can be utilized are referred in this paper.

The multiple objects tracking task can be broken down into two subtasks: the detection of objects of interest, and the association between objects and detections along the time. The main challenges in the detection task arise from the fact that there can be missing detections due to occlusions or strong changes in the object appearance.
There are two types of approaches for classifying moving objects.

**Shape-based classification**

Different descriptions of shape information of motion areas such as points, boxes, silhouettes and blobs are available for classifying moving objects. Here it classifies moving object blobs into classes like single human, vehicles, and human groups.

**Motion-based classification**

Residual flow is used to analyze rigidity and periodicity of moving objects. It is expected that rigid objects has little residual flow, whereas a non rigid moving object such as a human being has a higher average residual flow and even display a periodic component. Based on this useful evidence, human motion is distinguished from motion of other objects, such as vehicles. After the removal of unwanted detections the moving objects need to be tracked correctly and has to be counted in real time video surveillance applications.
2.2 Proposed System

The proposed system can process the on-line and offline video.

![Algorithm for Video Surveillance](image)

Figure 2.1 Algorithm for Video Surveillance

2.2.1 Threshold Values

Proper threshold values have to be chosen for background, standard deviation and area of the moving objects. The statistical parameter standard deviation is used in the processing of removing the shadow of the moving object. In this algorithm threshold value of background chosen as 250 pixels, standard deviation is 0.25 and area of the moving object is 8 pixels. 8*8 pixel is taken as one block in this algorithm.
2.2.2 Input Video

The input video format is avi. avi stands for audio video interleave. An AVI file actually stores audio and video data under the RIFF (Resource Interchange File Format) container format. In AVI files, audio data and video data are stored next to each other to allow synchronous audio with-video playback. Audio data is usually stored in AVI files in uncompressed PCM (Pulse-Code Modulation) format with various parameters. Video Data is usually stored in AVI files in compressed format with various codecs and parameters. The avi read, avi info functions are mentioned to read the input video avi format. This Algorithm is tested with input video file having 120 frames.

2.2.3 Extraction

After reading the input video file, extracted the red, green and blue intensities separately to find out the histogram easily. Image(:,:,1),image(:,:,2) and image(:,:,3) functions are used to read the red, blue and green intensities of input video frames.

2.2.4 Gray scale image

Gray scale images are images without color, or achromatic images. The levels of a gray scale range from 0 (black) to 1 (white). After calculating the histogram, images are converted into gray scale images to reduce the complexity while applying the morphological operations.

2.2.5 Subtraction

This proposed algorithm dynamically extracting the background from incoming all video frames, it is subtracted from every subsequent frame and compared with the background threshold. If is greater than the background threshold, it assumed as foreground otherwise it is background. The Background is updated in each and every frame.
2.2.6 Shadow removal

Performing the operation using a function on each frame by 8*8 block wise and result is compared with the variance threshold. If the result is less than the variance threshold, it assumes as shadow and it takes logic 0 otherwise it takes logic 1.

2.2.7 Adjacent Frame Difference Method

Adjacent frame difference method, moving object is extracted according to the differences among two or three continuous frames. The method is the most simple and direct, with which the changing part in video can be quickly detected. As a matter of fact, it only detects objects making relative motions. Moreover, since the time interval between two images is quite short, illumination changes have little influence on difference images, so the detection is effective and stable. The method using frame differences can better adapt to environment in intensive fluctuation, and can easily detect those pixels causing images to change distinctly when the target moves. However, it is inadequate for dots with insignificantly changed pixels. Accordingly, the method is largely used in situations with comparatively simple background and little environmental interference.
CHAPTER: 3
DESIGN ANALYSIS
3. DESIGN ANALYSIS

3.1 Software Requirement Specification

Software Requirements Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. Software specifications are used designing the systems and to manage the time complexity. New tools and techniques are announced in quick succession. This has forced the software engineers and industry to continuously look for new approaches to software design and development. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase).

The SRS phase consists of two basic activities:

1. Problem Analysis:
   The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

2. Specific Requirements:
   Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specification are addressed during this activity. The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

Interface Requirements:

User Interface:
Every user may not be skilled at handling the interfaces. Hence the product that we developed used a simple and easy to use GUI Input from user is via keyboard.
Hardware Interface:

The minimum requirements that are required to interact with a simple GUI are well enough to support this product.

Software Interface:

This product is developed in Windows XP environment using MATLAB. The toolboxes used to develop this product are Image Processing Toolbox and Computer Vision System Toolbox. This project is completely implemented using these two tool boxes.

3.2 Hardware and Software Requirements

Hardware Requirements:

- Processor: Intel® Core™ i3 2.53 GHz / Above
- RAM: RAM 2 GB / Above
- HDD: 120 GB / Above

Software Requirements:

- Operating System: Windows XP and above
- Developing Environment: MATLAB R2011a

3.3 UML Diagrams:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.
**Definition:**

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

**UML is a language:**

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

**UML Specifying:**

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

**UML Visualization:**

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

**UML Constructing:**

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

**Building blocks of UML:**

The vocabulary of the UML encompasses 3 kinds of building blocks

- Things
- Relationships
- Diagrams
Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types Structural Things, Behavioral Things, Grouping Things, An notational Things

Relationships:

Relationships tie the things together. Relationships in the UML are Dependency, Association, Generalization and Specialization.

UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

- Structural Diagrams
- Behavioral Diagrams

Structural Diagrams:

The UML’s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavioral Diagrams:

The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioral diagrams consists of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.
3.3.1 Use Case diagram:

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, associations between actors and users. In general, it shows a set of use cases and actors and their relationships. The creation of a use case model is an excellent vehicle for elicitation of functional requirements.

![Use Case Diagram for Video Surveillance](image)

**Figure 3.1 Use Case Diagram for Video Surveillance**

3.3.2 Sequence diagram

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram has two components are vertical dimension represents time, the horizontal dimension represents different object. The vertical line is called object’s lifeline.
Figure 3.2 Sequence Diagram for Video Surveillance
3.3.3 Collaboration diagram

Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. The collaboration diagram represents interactions among objects in terms of sequenced messages.

![Collaboration Diagram for Video Surveillance](image)

**Figure 3.3 Collaboration diagram for Video Surveillance**

3.3.4 Activity diagram

The purpose of activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. An activity is shown as around box containing the name of the operation.
3.3.5 State diagram

State diagrams are used to give an abstract description of the behavior of the system. This behavior is analyzed and represented in series of events that could occur in one or more possible states.
Figure 3.5 State Diagram for Video Surveillance
4. TECHNOLOGY DESCRIPTION

4.1 Introduction to MATLAB

MATLAB (Matrix laboratory) is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations: solving systems of linear equations, computing Eigen values and eigenvectors, factoring matrices, and so forth. In addition, it has a variety of graphical capabilities, and can be extended through programs written in its own programming language. MATLAB is designed to solve problems numerically, that is, in finite-precision arithmetic. Therefore it produces approximate rather than exact solutions. It is mathematical software that offers an Integrated Development Environment (IDE) with its own programming language, called M and is available for UNIX, Windows Apple Mac OS X platforms. It was created in the seventies by Cleve Moler, chairman of the computer science at the University of New Mexico. He tried to design a new language able to use LINPACK and EISPACK without any knowledge about FORTRON. This software was quickly expanded to other universities and was really welcome by the applied mathematics community. In the eighties Jack Litte, an engineer, joined Moler, because he realized it had a tremendous commercial potential. They wrote MATLAB in C, and founded Math works in 1984. MATLAB then was turned into a language for technical computing of high performance. It integrates visualization, computation and an easy-to-use programming environment where problems and solutions are expressed in familiar mathematical notation.

The typical uses of MATLAB are:

- Math and computation
- Algorithm development
- Modeling, simulation and prototyping
- Data analysis, exploration and visualization
- Scientific and engineering graphics
General purpose commands in MATLAB

- **global**: Declares variables to be global.
- **help**: Searches for a help topic.
- **lookfor**: Searches help entries for a keyword.
- **quit**: Stops MATLAB.
- **who**: Lists current variables.

4.1.1 Introduction to M-Files

Scripts and Functions:

MATLAB can also be used as a programming language. To program in MATLAB you simply create a text file containing MATLAB commands exactly as you would type them interactively in the MATLAB window. The file may have any legal UNIX name, and should end with a .m extension. These files may be placed in the root directory, or a directory named MATLAB. (Any other directories would have to be explicitly added to the MATLAB path.) There are two types of m-files in MATLAB. One is called a script. This is simply a list of MATLAB commands with no header. The other type is a function. Functions have a header line that may look something like: Function y=fun1(x) Functions may be passed arguments, and may return results. To invoke a script or a function simply type the filename (without the .m extension) into the MATLAB window. Also the m-files that are created are included in the help listing. If we perform a help on a specific m-file, help will return any comments which appear before the first line of actual code in the m-file.

Debugging M-files

This section introduces general techniques for ending errors in M-files. Debugging is the process by which you isolate errors in the program or code. Debugging helps to correct two kinds of errors:

1. Syntax errors
   For example omitting a parenthesis or misspelling a function name.

2. Run-time errors
Run-time errors are usually apparent and difficult to track down.

**Correcting an M-file**

To correct errors in an M-file, we can adopt any of the following ways,

- Quit debugging
- Do not make changes to an M-file while MATLAB is in debug mode
- Make changes to the M-file
- Save the M-file
- Clear breakpoints
- Run the M-file again to be sure it produces the expected results.

Thus the MATLAB has many utilities that can be used for efficient tasking. It provides the user with many toolboxes that acts as a guide in many areas.

**4.2 Phases in the project:**

The following is the system block diagram for the proposed system. It depicts the sequence of steps to be followed.

![System block diagram](image)

**Figure 4.1 System block diagram**
4.2.1 Extraction of Foreground Objects

Firstly, an input video file is read into the MATLAB workspace using vision.VideoFileReader. Next in order to separate the moving foreground objects, mixture of Gaussians background subtraction technique is applied. In this method, at each iteration Gaussians are evaluated using a simple heuristic to determine which ones are mostly likely to correspond to the background. Pixels that do not match with the “background Gaussians” are classified as foreground. In MATLAB, for the foreground objects separation the system objects are inbuilt present which are utilized in this context. The system object of vision.ForegroundDetector is created using Gaussian Mixture model to detect the foreground objects. Thus here the background is separated from the foreground objects. Here the video is processed in form of each frame using step function and the foreground objects are detected in each frame. In this step, the shadows of the foreground objects are removed and the shadow free objects are obtained for further processing. Here from each frame after the foreground objects are extracted the threshold value according to the video is applied and pixels having less than the threshold are considered as zero and those with values greater than threshold are assigned one.

4.2.2 Objects Detection and Tracking

Here the blob analysis is used and the system object for vision.BlobAnalysis is created and the required properties are set accordingly and the area and bounding box of the blobs are determined. After that the exclusion of the other objects that are not of interest is done by calculating the ratio of the blobs and bounding box areas. Thus only the required objects are extracted. After the required objects are extracted, the border color of the box can be chosen using vision.ShapeInserter. Here white color is assigned to the CustomBorderColor property in order to get the white bounding boxes around the moving objects. Now the detected objects are enclosed in the bounding objects in each frame and the resultant video is displayed. After the objects are detected, the bounding objects are used to track the objects in respective frames. The particle filtering technique is applied so that the objects can be tracked effectively. A particle filter, also known as a sequential Monte Carlo method (SMC), is a sophisticated model estimation technique based on simulation. Particle filters are usually used to estimate Bayesian models in which the latent variables are connected in a Markov chain.
similar to a Hidden Markov model (HMM), but typically where the state space of the latent variables is continuous rather than discrete. Filtering refers to determining the distribution of a latent variable at a specific time, given all observations up to that time. Particle filters are so named because they allow for approximate "filtering" using a set of "particles" (differently weighted samples of the distribution). There are several types of particle filters present. Some of them are sampling importance resampling filter, auxiliary particle filter, regularized particle filter. The first type of filter can be applied to Bayesian recursive filtering problems whereas the second filter are the variants of standard sampling importance resampling filters. The latter is identical to the sampling importance resampling filter, except for the resampling stage. In this paper, the particle filtering strategy by means of a set of weighted samples and the posterior pdf is approximated accordingly. To avoid degeneracy problem, a resampling stage is performed by means of the sampling importance resampling algorithm.

4.3 Functions used:

1. Vision.VideoFileReader ():-
   VideoFileReader reads video frames and audio samples from video file.
   Syntax:
   
   HVFR = vision.VideoFileReader

   Description:
   It returns a video file reader System object, HVFR, to read video and/or audio from a video file.

2. Vision.ColorSpaceConverter ():-
   ColorSpaceConverter does the image color space conversion.
   Syntax:

   HCSC = vision.ColorSpaceConverter ('Property Name', Property Value ...)

   Description:
   It returns a color space conversion object, HCSC, with each specified property set to the specified value.
3. **Vision.ForegroundDetector ()**

ForegroundDetector detects foreground using Gaussian Mixture Models.

**Syntax:**

\[
H = \text{vision.ForegroundDetector} \left( \text{'Property Name'}, \text{Property Value} \ldots \right)
\]

**Description:**

It returns a foreground detector System object, H, with each specified property set to the specified value.

**ForegroundDetector properties:**

- `NumTrainingFrames` - Number of initial video frames used for training the background model
- `LearningRate` - Learning rate used for parameter updates
- `MinimumBackgroundRatio` - Threshold to determine the Gaussian model in the mixture model that constitutes the background process
- `NumGaussians` - Number of distributions that make up the foreground-background mixture model
- `Initial Variance` - Initial variance to initialize all distributions that compose the foreground-background mixture model

4. **Vision.BlobAnalysis ()**

BlobAnalysis gives the properties of connected regions.

**Syntax:**

\[
HBLOB = \text{vision.BlobAnalysis} \left( \text{'Property Name'}, \text{Property Value} \ldots \right)
\]

**Description:**

It returns a blob analysis object, HBLOB, with each specified property set to the specified value.

**BlobAnalysis properties:**

- `AreaOutputPort` - Enables blob area output
- `CentroidOutputPort` - Enables coordinates of blob centroids output
- `BoundingBoxOutputPort` - Enables coordinates of bounding boxes output
- `OrientationOutputPort` - Enables output vector whose values represent angles between ellipses' major axes and x-axes
PerimeterOutputPort - Enables output vector whose values represent estimates of blob perimeter lengths

MaximumCount - Maximum number of labeled regions in each input image
NumBlobsOutputPort - Enables output of a scalar value that represents actual number of labeled regions in each image
MinimumBlobAreaSource - Source of minimum blob area
MinimumBlobArea - Minimum blob area in pixels
MaximumBlobAreaSource - Source of maximum blob area
MaximumBlobArea - Maximum blob area in pixels

5. Vision.ShapeInserter ():-
ShapeInserter draws rectangles, lines, polygons, or circles on images.

Syntax:
HSHAPEINS = vision.ShapeInserter ('PropertyName', PropertyValue ...)

Description:
It returns a shape inserter System object, HSHAPEINS, with each specified property set to the specified value.

ShapeInserter properties:
Shape - Type of shape(s) to draw
BorderColorSource - Source of border color
BorderColor - Border color of shape
CustomBorderColor - Intensity or color value for shape's border
FillColor - Fill color of shape
CustomFillColor - Intensity or color value for shape's interior
ROIInputPort - Enables defining area for drawing shapes via input
Antialiasing - Enables performing smoothing algorithm on shapes

6. Vision.TextInserter ():-
TextInserter draws text on image or video stream.

Syntax:
HTXTINS = vision.TextInserter (TEXT, 'PropertyName', PropertyValue ...)

Description:
It returns a text inserter object, HTXTINS, with the Text property set to TEXT and other specified properties set to the specified values.

TextInserter properties:
- Text - Text string to draw on image or video stream
- ColorSource - Source of intensity or color of text
- Color - Intensity or color of text
- Location - Top-left corner of text bounding box
- Font - Font face of text
- FontSize - Font size in points
- Antialiasing - Perform smoothing algorithm on text edges

7. Vision.VideoPlayer ():-
VideoPlayer play video or display image

Syntax:
HVP = vision.VideoPlayer ('PropertyName', PropertyValue ...)

Description:
It returns a video player System object, HVP, with each specified property set to the specified value.

VideoPlayer properties:
- Name - Caption to display on video player window
- Position - Scope window position in pixels

8. step ():-
Gives the step response of dynamic systems.

Syntax:
STEP (SYS1, SYS2, ..., T)
Description:

It plots the step response of several systems SYS1, SYS2... on a single plot. The time vector T is optional. You can also specify a color, line style, and marker for each system.

9. int32 ()::-

Convert to signed 32-bit integer.

Syntax:

\[ I = \text{int32}(X) \]

Description:

It converts the elements of the array X into signed 32-bit integers. X can be any numeric object, such as a DOUBLE. The values of an INT32 range from -2,147,483,648 to 2,147,483,647.

10. aviread ()::-

Read AVI file.

Syntax:

\[ \text{MOV} = \text{aviread}(\text{FILENAME}) \]

Description:

It reads the AVI movie FILENAME into the MATLAB movie structure MOV.

11. frame2im ()::-

Return image data associated with movie frame.

Syntax:

\[ [X, \text{MAP}] = \text{frame2im}(F) \]

Description:

It returns the indexed image X and associated color map MAP from the single movie frame F.
12. **imabsdiff ()**:

Absolute difference of two images.

**Syntax**:

\[ Z = \text{imabsdiff}(X, Y) \]

**Description**:

It subtracts each element in array Y from the corresponding element in array X and returns the absolute difference in the corresponding element of the output array Z.

13. **im2double()**:

Convert image to double precision.

**Syntax**:

\[ \text{im2double}(I1) \]

**Description**:

im2double takes an image as input, and returns an image of class double. If the input image is of class double, the output image is identical to it. If the input image is not double, im2double returns the equivalent image of class double, rescaling or offsetting the data as necessary.

14. **rgb2gray ()**:

Convert RGB image or color map to gray scale

**Syntax**:

\[ I = \text{rgb2gray}(RGB) \]

\[ \text{newmap}=\text{rgb2gray} (\text{map}) \]

**Description**:

I=rgb2gray(RGB) converts the true color image to RGB to the gray scale intensity image I. rgb2gray converts RGB images to gray scale by eliminating the hue and saturation information while retaining the luminance.

15. **release ()**

Release a COM interface.
Syntax:
release(object)

Description:
Release (OBJ) releases the interface and all resources used by the interface. Once an interface has been released, it is no longer valid. Subsequent operations on the MatLab object that represents that interface will result in errors.
5. TESTING

5.1 Introduction to Testing:

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements. Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. It represents interesting anomaly for the software.

Testing is the process of detecting errors. It performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also. Generally, the testing phase involves the testing of the developed system using various test data. Preparation of the test data plays a vital role in the system testing. After preparing the test data the system under study was tested using those test data. While testing the system, errors were found and corrected by using the testing steps and corrections are also noted for future use. Thus, a series of testing is performed for the proposed system, before the system was ready for the implementation. Thus the aim of testing is to demonstrate that a program works by showing that it has no errors. The fundamental purpose of testing phase is to detect the errors that may be present in the program. Thus testing allows developers to deliver software that meets expectations, prevents unexpected results, and improves the long term maintenance of the application. Depending upon the purpose of testing and the software requirements, the appropriate methodologies are applied. Wherever possible, testing can also be automated.

5.1.1 Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, altogether,

Testing is a process of executing a program with the intent of finding an error.

A successful test is one that uncovers an as yet undiscovered error.
A good test case is one that has a high probability of finding error, if it exists. The software more or less confirms to the quality and reliable standards.

5.1.2 Levels of testing:

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are

![Levels of Testing Diagram](image)

5.2 Testing Strategies:

5.2.1 Unit testing:

Unit Testing is a level of the software testing process where individual units or components of a software or system are tested. It is also known as component testing, refers to tests that verify the functionality of a specific section of code. The purpose is to validate that each unit of the software performs as designed. All modules must be successful in the
unit test before the start of the integration testing begins. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Unit testing focuses verification effort on the similar unit of software design the form. This is known as form testing. In this testing step, each module is found to be working satisfactorily, as regard to the expected output from the module. Each module has been tested by giving different sets of inputs, when developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user. Each module can be tested using the following two strategies:

**5.2.1.1. White Box Testing:**

White box testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. We tested step wise every piece of code, taking care that every statement in the code is executed at least once. White-box testing can be applied at the unit, integration and system levels of the software testing process. It is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. The white box testing is also called Glass Box Testing. We have generated a list of test cases sample data, which is used to check all possible combinations of execution paths through the code at every module level. This testing has been uses to find in the following categories:

- Execute internal data structures to ensure their validity.
- Guarantee that all independent paths have been executed.
- Execute all logical decisions on their true and false sides.

**5.2.1.2. Black Box Testing:**

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It treats the software as a "black box", examining functionality without any knowledge of internal
implementation. This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level i.e., here the module will be treated as a black box that will take some input and generate output. Output for a given set of input combinations are forwarded to other module. This testing has been uses to find in the following categories:

- Initialization and termination errors.
- Incorrect or missing functions
- Performance errors

5.2.2 Integration Testing:

Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. It works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system. The testing of combined parts of an application to determine if they function correctly together is the main motto of integration testing. There are two methods of doing integration testing. They are Bottom-up integration testing and Top Down integration testing. All modules are combined in the testing step. Then the entire program is tested as a whole.

5.2.3 Validation Testing:

At the culmination of the integration testing, the software is completely assembled as a package, interfacing errors have been uncovered and corrected and final series of software validation testing begins. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements.
5.2.4 System Testing:

System testing is a level of the software testing process where a complete, integrated system or software is tested. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets quality standards. Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. Here entire project has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

5.2.5 Output Testing:

After performing validation testing, the next steps are output testing of the proposed system, since no system could be useful if it does not produce the desired output in the specified format. The output generated are displayed by the system under consideration or tested by asking the user about the format required by them. Here the output format is considered in two ways. One is on the screen and the other is on the printed form.

5.2.6 Acceptance Testing:

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery. Acceptance test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system, the internal logic of program is not emphasized.

In this project we have collected some data and tested whether project is working correctly or not. Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for user acceptance by constantly keeping in touch
with the perspective system users at the time of developing and making changes whenever required. This is done with regard to the following points.

- Input screen design
- Output screen design
- Menu driven system

### 5.3 Test Approach:

Testing can be done in two ways:

- Bottom up approach
- Top down approach

#### 5.3.1 Bottom up Approach:

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

#### 5.3.2 Top down Approach:

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.
5.4 Validation:

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

5.5 Test cases Results:

Testing is the set of activities that can be planned in advance and conducted systematically.

The underlying motivation of program testing is to affirm software quality.

5.6 Sample Screenshots

First of all the screenshots of the input video are shown followed by the screens of the output video that has been obtained in MATLAB.

5.6.1 Screenshots of Input video:

The screenshots of the video that was used in this project are as follows.

Figure 5.2 Person in Motion input video screenshot 1
5.6.2 Screen Shots of Output Video

When the code is executed in MATLAB, the output video with the count is displayed. The screen shots of the output video that was obtained in this project are as follows.
Figure 5.4 Persons in Motion output video screenshot 3

Figure 5.5 Persons in Motion output video screenshot 4
Figure 5.6 Persons in Motion output video screenshot 5

Figure 5.7 Person in Motion output video screenshot 6
Figure 5.8 Persons in Motion output video screenshot 7
CHAPTER: 6
CONCLUSION
6. CONCLUSION

The proposed algorithm extracted the background from the all frames of video and detected the foreground effectively. This algorithm also dynamically updating the background frame by frame. This algorithm also identify the shadow of the moving object and is removed to calculate the area of the object accurately. This algorithm also identifies even small object by adjusting the threshold values. Even the smallest, slowest, fastest, of a moving region is detected accurately by selecting the proper threshold value of the objects. Finally this algorithm works for On-line (Real time) and Off-line (Quasi real time) video processing and its computational complexity is low.

6.1 Future Work

Future work will be directed towards achieving the following issues:

- Object classifications
- Better understanding of human motion not only vehicle, including segmentation and tracking of articulated body parts.
- Improved data logging and retrieval mechanisms to support 24/7 system operations
- Better camera control to enable smooth object tracking at high zoom, incase, video is vibrating Video stabilization algorithm is required.
- Acquisition and selection of “best views” with the eventual goal of recognizing individuals in the scene.
CHAPTER: 7
BIBLIOGRAPHY
7. BIBLIOGRAPHY


