VigilCam – a Video Surveillance System

By

Deepak Roy Chittajallu
cdeepakroy@yahoo.com

Department of computer science and Information technology
Jawaharlal Nehru Technological University
Hyderabad, India

Abstract:

Security and Surveillance are two vital issues to be addressed in the commercial and defense arenas. Deploying human resources for these tasks is becoming expensive and better alternative strategies are in need. An autonomous video surveillance system would be such an alternative.

The aim of this thesis is to develop an autonomous video surveillance system that continuously perceives the action involved in the live video imagery captured by the surveillance camera and alerts the human operator of any suspicious activity, thus minimizing the requirement of human resources. This report presents the basic framework and methodologies adopted by us in the development of the ‘VigilCam video surveillance system project’. It deals with the technical approaches that can be adopted for Moving Object Detection, Object Tracking and Activity Perception. A novel method devised for Multiple Object Extraction is also explained. We also show the experimental results of the VigilCam video surveillance system.
Acknowledgements

We gratefully acknowledge the guidance provided by Dr. Kumareswaran, Head of the department Computer Science, right from the conception of the project to the successful completion of it. His constant support and encouragement has been invaluable.

We also acknowledge the help, guidance and inspiration provided by Mr. Chaitanya, Associate professor Computer Science. He, along with Dr. Kumareswaran, has been the “light at the end of the tunnel” for most of our problems.

We would also like to extend our thanks to Prof. Uday Kumar, Head of the department Computer Science and Information technology, for his encouragement and inspiration.

Further, we take this opportunity to thank our Principal, Prof. P. Narasimha Reddy and the college management for providing us with the right atmosphere, facilities and inspiration to put in our best efforts.
# Table of Contents

1. Overview ........................................................................................................................... 5

2. Acquaintance with Image Processing Techniques .......................................................... 8  
   2.1 Introduction .................................................................................................................. 8  
   2.2 Accessing the bits in an Image ..................................................................................... 9  
   2.3 A snapshot of the Image Processing Tutor Application ............................................. 10  
   2.4 Image Processing Algorithms .................................................................................... 10  
      2.4.1 Image Enhancement ............................................................................................. 11  
      2.4.2 Image Segmentation .......................................................................................... 17  

3. Interfacing with Image Capture Devices ......................................................................... 21  
   3.1 Introduction ................................................................................................................ 21  
   3.2 TWAIN ....................................................................................................................... 21  
      3.2.1 Benefits of using TWAIN .................................................................................. 23  
      3.2.2 TWAIN Architecture ....................................................................................... 24  
      3.2.3 Communication between elements of TWAIN ................................................. 26  
      3.2.4 Custom TWAIN Wrapper class ........................................................................ 29  
   3.3 Microsoft® Video For Windows (VFW) .................................................................... 30  
   3.4 TWAIN vs. VFW ......................................................................................................... 31  

4. Moving Object Detection and Average motion tracking .................................................. 32  
   4.1 Introduction ................................................................................................................ 32  
   4.2 Approach adopted for object detection and average motion tracking ....................... 32  
   4.3 Localized segmentation based approach to detect small and distant objects ............ 34  
   4.4 Working Images of Moving Object Detection and Tracking Approach .................... 36  

5. Multiple Object Detection ............................................................................................... 37  
   5.1 Introduction ................................................................................................................ 37  
   5.2 Multiple Object Extraction Algorithm ...................................................................... 37  
   5.3 An Example .............................................................................................................. 39  
   5.4 Advantages of Multiple Object Extraction Algorithm .............................................. 42  
   5.5 Limitations of Multiple Object Extraction Algorithm ............................................. 43  
   5.6 Working image examples of Multiple Object Extraction Algorithm ....................... 43
6. Activity Perception – Restricted Area Surveillance ................................. 45
   6.1 Introduction .............................................................................................. 45
   6.2 Technical Approach adopted for restricted area surveillance ............... 46
   6.3 Integrating Restricted area surveillance into video surveillance system .... 48
7. Future Scope .................................................................................................. 49
Appendix A - References .................................................................................. 51
Appendix B - Glossary ...................................................................................... 52
The tremendous technological advancements made in the field of digital video capture and the accompanying reduction in their prices has made it economical to incorporate them in video surveillance systems. These advanced devices coupled with robust software can be effectively used to work as autonomous surveillance systems, thus weeding out the need for expensive human resources. Most of the surveillance cameras currently used in the commercial, law enforcement and military sectors are being used as a “passive” investigation tool. This begets the need for an autonomous proactive video surveillance system. i.e. computer based systems that continuously monitor the live video imagery captured by surveillance cameras for any suspicious activity and alert the human operator accordingly.

This report puts forth the framework and methodologies adopted in the development of an autonomous video surveillance system called VigilCam. The basic framework or the block diagram of the VigilCam video surveillance system is shown in Fig 1.1.
The VigilCam Video Surveillance System was developed linearly in a sequence of stages, which are as listed in the figure given below:

Fig 1.2 Development stages

This report is organized into a set of chapters wherein each chapter describes the work done and the methodologies used in one or more development stages described above.

**Chapter 2**: Describes the various image processing techniques implemented for getting acquainted with image manipulation.

**Chapter 3**: Describes the two avenues – Twain and Video For Windows (VfW) emphasized for interfacing with image capturing devices.
Chapter 4: Illustrates the approaches used for Moving object detection, average motion tracking and minute change detection

Chapter 5: Describes the novel *Multiple Object Extraction Algorithm*

Chapter 6: Describes how restricted area surveillance is implemented.

Chapter 7: Deals with the limitations of the existing video surveillance system and ideas that can be applied to remove those limitations. It also mentions the possible future extensions to the present system.

![VigilCam video Surveillance System Setup for Road Surveillance](image)

*Fig 1.3 VigilCam video Surveillance System Setup for Road Surveillance*
CHAPTER 2

Acquaintance with Image Processing Techniques

2.1 Introduction

Development of any system in field of computer vision, especially in the area of video surveillance and monitoring, often involves the application of a variety of image processing algorithms. The field of Digital Image Processing refers to processing digital images by means of a digital computer. A digital image consists of a finite number of small elements called pixels, where in each pixel has three defining attributes namely – the Spatial Coordinates (x, y) indicating the pixel location and the intensity indicating the perceptual color of the pixel. It is these small bits, which are manipulated by all image-processing algorithms. Before taking up the major task of developing a video surveillance system that involves the application of complex image processing techniques, the first thing to look into is how to access the information in each frame and how implement the image processing algorithms. So the first phase in the development of the VigilCam Video Surveillance System Project was to get acquainted with various image-processing techniques. This chapter describes the various image processing techniques that were implemented in the Image Processing Tutor Application that was created in the initial stages of the project development process. The Transformed images shown in the next few sections have been processed by the Image Processing Tutor Application created in the first stage of the Vigil Cam video surveillance system development process.

2.2 Accessing The Bits in an Image:
One of the most common image formats is the BMP format or the Device Independent Bitmap. The sequence of images captured by the surveillance camera are normally available in the form of a compressed DIB. The format of the bitmap file or Device Independent Bitmap (DIB) is given in the following picture.

Fig 2.1 Anatomy of a DIB

Freeimage is a widely used free and open source graphics library, which provides bitmap-loading support into the required application. It supports access to a variety of image formats currently existing. It provides functions for many features like loading the bitmap image, getting access to the bits, reading and writing files of various formats etc. It is available in the form of a Dynamic Linked Library to which an application can dynamically link to in order to use its routines. For further details and documentation refer the site [http://freeimage.sourceforge.net/intro.html](http://freeimage.sourceforge.net/intro.html). The Image Processing Tutor Application uses this library to access the bits in image files of various formats.
2.3 A snapshot of the Image Processing Tutor Application

Shown below is the interface of Image Processing Tutor Application that demonstrates the effects of various image-processing techniques on images acquired from files or from an image acquisition device.

Fig 2.2. User Interface of Image Processing Tutor Application

2.4 Image Processing Algorithms

The various image-processing algorithms implemented in the Image Processing Tutor Application are described briefly in the next few sections. Each algorithm is also accompanied by the images transformed using the Image Processing Tutor Application.

2.4.1 Image Enhancement
**Image negative:** Reverses the intensity levels of the image producing the equivalent of a photographic negative. The negative of an image with gray levels in the range \([0,L-1]\) is given by the expression

\[ S = L - 1 - r \]

Where \( S \) is the output gray level, \( r \) is the input gray level.

![Fig 2.3. Original Image](image1)

![Fig 2.4. Negative](image2)

**Log Transformations:** Maps a narrow range of low gray-level values in the input image into a wider range of output levels. The opposite is true of the higher levels of input values. Used to expand the values of dark pixels in an image.

General Form: \( s = c \log(1 + r) \) where \( c \) is a constant and \( r \geq 0 \).

![Fig 2.5. Original Image](image3)

![Fig 2.6. Log transformed image with \( c = 1 \)](image4)
**Power-Law Transformation:** Power-law transformations have the basic form

\[ S = c \cdot \text{power}(r, \gamma) \]

where \( c, \gamma \) are positive constants.

Similar to the Log Transformation, power-law curves with fractional \( \gamma \) map a narrow range of dark input values to a wider range of output values, with the opposite being true for higher values of input levels.

![Original Image](image1.png)  ![Power law transformed image with \( \gamma = 0.25, c = 1 \)](image2.png)

**Spatial Domain filters:**

Filter operations on images are done using a subimage called the filter mask or window which is a fixed dimensional matrix. The general process of applying the filter is as follows:

The filter mask is moved from point to point in an image and at each point the response of the filter is calculated in a predefined manner. The final output of the processing is obtained after applying the filter at every pixel in the image.
Smoothing Spatial filters: These are used for noise reduction and blurring. Blurring is generally used in pre-processing steps such as removal of small details from an image prior to object extraction or bridging of small gaps in lines and curves.

The output of a linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters are called averaging filters.

Mathematically given by:

\[
    g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)}
\]

where \(x = 0, 1, 2, 3 \ldots M-1\), \(y = 0, 1, 2 \ldots N-1\)

image dimensions \(M \times N\)

filter dimensions \(m \times n\)
**Order-statistic filters**: These are non-linear spatial filters whose response is based on ranking the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result.

**Median filter**: Replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel. In order to perform median filtering at a point in an image, first sort the values of the pixel in question and its neighbors determine their median and
assign this value to that pixel. Thus the principal function of median filters is to force points with distinct gray levels to be more like their neighbors.

Minimum and maximum filters: The median represents the $50^{th}$ percentile of a ranked set of numbers. Using the $100^{th}$ percentile results in the max–filter, useful in finding the brightest points in the image. The $0^{th}$ percentile filter is the min-filter is for the less bright pixels.
**Sharpening filters:** The principal objective of sharpening is to highlight the areas of high gray-level discontinuities like edges or lines, in order to enhance the detail that has been blurred.

The approach basically consists of defining a discrete formulation of the second-order derivative, and then constructing the filter mask based on that formulation.

The simplest derivative operator is the laplacian

\[
\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}
\]

\[
\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)
\]

\[
\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)
\]

the digital implementation of the laplacian is obtained by summing the above two components. The laplacian mask is shown in the figure below from which it is evident that the laplacian is an isotropic filter, i.e. it is rotation invariant.

\[
\begin{array}{ccc}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{array}
\]

fig: 2.20 filter mask used to implement the digital Laplacian
2.4.2 Image Segmentation

**Point detection**: The detection of isolated points in an image is straightforward: a point is said to be detected at a location on which the mask is centered if

\[ |R| \geq T \]

where \( R \) denotes the result of the mask applied on the image and \( T \) non-negative threshold

i.e. an isolated point is significantly different from its background and will be significantly different from its surroundings and will easily be detected.

**Line Detection**:  
Masks for line detection:

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
2 & 2 & 2 \\
-1 & -1 & -1
\end{array} \quad \begin{array}{ccc}
-1 & 2 & -1 \\
-1 & 2 & -1 \\
2 & -1 & -1
\end{array} \quad \begin{array}{ccc}
-1 & -1 & 2 \\
-1 & 2 & -1 \\
2 & -1 & -1
\end{array} \quad \begin{array}{ccc}
2 & -1 & -1 \\
-1 & 2 & -1 \\
-1 & -1 & 2
\end{array}
\]

- Horizontal lines
- Vertical Lines
- +45° lines
- -45° lines
Edge Detection:

An edge is the set of connected pixels that lie on the boundary between two regions. First or Second order derivatives are used to detect edges.

Sobel Gradient operators:

\[
\begin{array}{ccc}
-1 & 2 & -1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\quad
\begin{array}{ccc}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
1 & 1 & 1 \\
\end{array}
\]
Prewitt Gradient operators:

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\quad
\begin{array}{ccc}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{array}
\]

The sobel gradient masks have slightly superior noise-suppression characteristics than prewitt masks. The prewitt masks are simpler to implement than the sobel masks.

The coefficients in all the masks shown sum to zero indicating they give a response of Zero in the areas of constant gray level.

Prewitt masks for diagonal edges

\[
\begin{array}{ccc}
0 & 1 & 1 \\
-1 & 0 & 1 \\
-1 & -1 & 0 \\
\end{array}
\quad
\begin{array}{ccc}
-1 & -1 & 0 \\
-1 & 0 & 1 \\
0 & 1 & 1 \\
\end{array}
\]

Sobel Gradient operators for diagonal edges

\[
\begin{array}{ccc}
0 & 1 & 2 \\
-1 & 0 & 1 \\
-2 & -1 & 0 \\
\end{array}
\quad
\begin{array}{ccc}
-2 & -1 & 0 \\
-1 & 0 & 1 \\
0 & 1 & 2 \\
\end{array}
\]
3.1 Introduction

The primary input of a video surveillance system is video acquired from video capturing devices. The first and foremost task before developing a video surveillance system hence, is to grab frames from a capture device. The advent of various video capture devices by different companies has made it necessary for developers of both the image acquisition devices and the software applications to recognize the need for a standard communication between the image devices and the applications. A standard would benefit both groups as well as the users of their products. It would allow the device vendors’ products to be accessed by more applications and application vendors could access data from those devices without concern for which type of device, or particular device, provided it.

TWAIN and Video For Windows are two such widely used standards. This chapter deals with interfacing with video capture devices using these two aforementioned standards.

3.2 TWAIN

TWAIN defines a standard software protocol and API (application programming interface) for communication between software applications and image acquisition devices (the source of the data).
The three key elements in TWAIN are:

• **The Application software**
  - An application must be modified to use TWAIN.

• **The Source Manager software**
  - This software manages the interactions between the Application and the Source. This code is provided in the TWAIN Developer’s Toolkit and should be shipped for free with each TWAIN application and Source.

• **The Source software**
  - This software controls the image acquisition device and is written by the device developer to comply with TWAIN specifications. Traditional device drivers are now included with the Source software and do not need to be shipped by applications.

![Fig 3.1 Elements of TWAIN](image-url)
3.2.1 The Benefits of Using TWAIN

For the Application Developer:

- Allows you to offer users of your application a simple way to incorporate images from any compatible raster device without leaving your application.
- Saves time and dollars. If you currently provide low-level device drivers for scanners, etc., you no longer need to write, support, or ship these drivers. The TWAIN-compliant image acquisition devices will provide Source software modules that eliminate the need for you to create and ship device drivers.
- Permits your application to access data from any TWAIN-compliant image peripheral simply by modifying your application code once using the high-level TWAIN application programming interface. No customization by product is necessary. TWAIN image peripherals can include desktop scanners, hand scanners, digital cameras, frame grabbers, image databases, or any other raster image source that complies to the TWAIN protocol and API.
- Allows you to determine the features and capabilities that an image acquisition device can provide. Your application can then restrict the Source to offer only those capabilities that are compatible with your application’s needs and abilities.
- Eliminates the need for your application to provide a user interface to control the image acquisition process. There is a software user interface module shipped with every TWAIN-compliant Source device to handle that process. Of course, you may provide your own user interface for acquisition, if desired.

For the Source Developer:

- Increases the use and support of your product. More applications will become image consumers as a result of the ease of implementation and breadth of device integration that TWAIN provides.
- Allows you to provide a proprietary user interface for your device. This lets you present the newest features to the user without waiting for the applications to incorporate them into their interfaces.
• Saves money by reducing your implementation costs. Rather than create and support various versions of your device control software to integrate with various applications, you create just a single TWAIN-compliant Source.

For the End User:
• Gives users a simple way to incorporate images into their documents. They can access the image in fewer steps because they never need to leave your application.

3.2.2 The TWAIN Architecture

The transfer of data is made possible by three software elements that work together in TWAIN:

The application, the Source Manager, and the Source.

These elements use the architecture of TWAIN to communicate. The TWAIN architecture consists of four layers:

• Application
• Protocol
• Acquisition
• Device

The TWAIN software elements occupy the layers as illustrated below. Each layer is described in the sections that follow.

Fig 3.2 TWAIN Software Elements
**Application**

The user’s software application executes in this layer. TWAIN describes user interface guidelines for the application developer regarding how users access TWAIN functionality and how a particular Source is selected. TWAIN is not concerned with how the application is implemented. TWAIN has no effect on any inter-application communication scheme that the application may use.

**Protocol**

The protocol is the “language” spoken and syntax used by TWAIN. It implements precise Instructions and communications required for the transfer of data.

The protocol layer includes:

- The portion of application software that provides the interface between the application and TWAIN
- The TWAIN Source Manager provided by TWAIN
- The software included with the Source device to receive instructions from the Source Manager and transfer back data and Return Codes

**Acquisition**

Acquisition devices may be physical (like a scanner or digital camera) or logical (like an image database). The software elements written to control acquisitions are called Sources and reside primarily in this layer. The Source transfers data for the application. It uses the format and transfer mechanism agreed upon by the Source and application. The Source always provides a built-in user interface that controls the device(s) the Source was written to drive. An application can override this and present its own user interface for Acquisition, if desired.
Device

This is the location of traditional low-level device drivers. They convert device-specific Commands into hardware commands and actions specific to the particular device the driver was written to accompany. Applications that use TWAIN no longer need to ship device drivers because they are part of the Source. TWAIN is not concerned with the device layer at all. The Source hides the device layer from the application. The Source provides the translation from TWAIN operations and interactions with the Source’s user interface into the equivalent commands for the device driver that cause the device to behave as desired.

3.2.3 Communication Between the Elements of TWAIN

Communication between elements of TWAIN is possible through two entry points. They are called DSM_Entry( ) and DS_Entry( ). DSM means Data Source Manager and DS means Data Source.

![Diagram](image)

Fig 3.3 Entry points for Communication between elements
The Application

The goal of the application is to acquire data from a Source. However, applications cannot contact the Source directly. All requests for data, capability information, error information, etc. must be handled through the Source Manager. TWAIN defines approximately 140 operations. The application sends them to the Source Manager for transmission. The application specifies which element, Source Manager or Source, is the final destination for each requested operation. The application communicates to the Source Manager through the Source Manager’s only entry point, the **DSM_Entry( )** function.

The parameter list of the DSM_Entry function contains:

- An identifier structure providing information about the application that originated the function call

- The destination of this request (Source Manager or Source)

- A triplet that describes the requested operation. The triplet specifies:
  - Data Group for the Operation (DG_)
  - Data Argument Type for the Operation (DAT_)
  - Message for the Operation (MSG_)

- A pointer field to allow the transfer of data
The Source Manager

The Source Manager provides the communication path between the application and the Source, supports the user’s selection of a Source, and loads the Source for access by the application. Communications from application to Source Manager arrive in the DSM_Entry( ) entry point.

- **If the destination in the DSM_Entry call is the Source Manager**
  - The Source Manager processes the operation itself.

- **If the destination in the DSM_Entry call is the Source**
  - The Source Manager translates the parameter list of information, removes the destination parameter and calls the appropriate Source. To reach the Source, the Source Manager calls the Source’s DS_Entry( ) function. TWAIN requires each Source to have this entry point.
3.2.4 Custom TWAIN Wrapper Class:

The class diagram for the TWAIN Wrapper, a custom C++ wrapper class that enables applications to easily communicate with image acquisition devices, is as shown below:

```cpp
CTwainWrapper
m_hTwainDll : HINSTANCE
m_hMessageWnd : HWND
m_blnSourceSelected : BOOL
m_blnSourceEnabled : BOOL
m_blnDSMOpen : BOOL
m_blnDSOpen : BOOL
m_bSourceEnabled : BOOL

CTwainWrapper()
<<virtual>> ~CTwainWrapper()
InitTwain()
ReleaseTwain()
<<virtual>> GetIdentity()
<<virtual>> SelectDataSource()
<<virtual>> OpenDataSource()
ProcessMessage()
SelectDefaultDataSource()
<<const>> IsDSMLOaded()
<<const>> IsSourceSelected()
<<const>> IsDSMOpen()
<<const>> IsDSOpen()
<<const>> IsSourceEnabled()
<<const>> GetReturnCode()
<<const>> GetStatus()
SetImageCount()
CallDSMEntry()
CloseDSM()
CloseDS()
GetCapability()
GetCapability()
SetCapability()
SetCapability()
EnableDataSource()
<<virtual>> DisableDataSource()
GetImageInfo()
TranslateMessage()
TransferImage()
EndTransfer()
CancelTransfer()
GetImage()
<<abstract>> CopyImage()
```

Fig 3.4 Class diagram of Custom Twain Wrapper Class
3.3 Video For Windows:

Microsoft® Video for Windows® (VFW) provides functions that enable an application to process video data. VFW was introduced in 16-bit Windows. Video For Windows provides applications with a simple, message-based interface to access video and waveform-audio acquisition hardware and to control the process of streaming video capture to disk.

Video For Windows is distributed in the form of a dynamic linked library(dll). It includes various functionalities such as:

- AVI(Audio Video Interleaved) File Functions and Macros
- Video Compression Manager
- Video Capture
- Custom File and Stream Handlers
- DrawDib

Video for Windows supports streaming video capture and single-frame capture in real-time. In addition, it provides control of video sources that are Media Control Interface (MCI) devices so the user can control (through an application) the start and stop positions of a video source, and augment the capture operation to include step frame capture.

The capture window you create by using the VFW can perform the following tasks:

- Capture audio and video streams to an audio-video interleaved (AVI) file.
- Connect and disconnect video and audio input devices dynamically.
- View a live incoming video signal by using the overlay or preview methods.
- Specify a file to use when capturing and copy the contents of the capture file to another file.
- Set the capture rate.
- Display dialog boxes that control the video source and format.
- Create, save, and load palettes.
- Copy images and palettes to the clipboard.
• Capture and save a single image as a device-independent bitmap (DIB).

For more information about Video For Windows refer to the Microsoft Developers Network (MSDN) site http://msdn.microsoft.com/

3.4 TWAIN vs. VFW

Certain Twain-compatible image acquisition devices do not allow the applications to override the Data source user interface. Overriding the user interface is mandatory in video surveillance systems that continuously grab and understand frames without any user intervention. Hence we used Video for windows for interacting with image capture devices in the VigilCam Video surveillance system.
4.1 Introduction
The most important aspect of any video surveillance system is motion detection or intrusion detection. It is the first step towards building any surveillance system. This chapter deals with our approach for motion detection and tracking of a single object. It is assumed that there is only one object, which is the cause of motion. This chapter also introduces the segmentation approach devised by us to detect the motion of small and distant objects.

4.2 Approach adopted for object detection and average motion tracking
Detection of moving objects in video streams is known to be a significant, and difficult, research problem. Aside from the intrinsic usefulness of being able to segment video streams into moving and background components, detecting moving blobs provides a focus of attention for recognition, classification, and activity analysis, making these later processes more efficient since only “moving” pixels need be considered.

Moving object detection is done using an approach called Background Subtraction. This involves subtracting out the stationary components from each frame in the video sequence. Even though Background Subtraction is simple to implement, consumes less processing time and provides more feature data compared to other conventional approaches like temporal differencing and optical flow, it is extremely sensitive to changes caused by environmental noise. A threshold-based approach is used to overcome the disturbances due to environmental noise.
For this, two threshold levels are defined viz:

- **Local Tolerance Level**: defines the minimum percentage by which the gray level of a pixel has to change so as to be detected as a changed pixel.

- **Global Tolerance Level**: defines the minimum percentage of changed pixels that should be present in a frame to detect any motion in it.

The average motion is tracked by continuously evaluating the centroid of each frame and plotting these centroids across frames. If it is assumed that there is only a single object responsible for the motion, this approach tracks the direction of motion of the object by plotting the centroid of it in each frame of the video sequence. This track can later be fed as input to other modules like behavior inference modules for higher-level analysis.

The centroid of an object is the weighted average of its pixel values. It can be calculated by using the following formulae.

\[
X = \frac{\sum x \cdot f(x, y)}{\sum f(x, y)}
\]

\[
Y = \frac{\sum y \cdot f(x, y)}{\sum f(x, y)}
\]

Where \(f(x, y)\) is the pixel gray level at position \((x, y)\), \(X\) is the x-coordinate of the centroid and \(Y\) is the y-coordinate of the centroid.

**Disadvantages:**

- One of the major drawbacks of the *Background Subtraction approach* is that it makes no considerations for stationary objects in the scene that start to move. Although these are detected, they leave behind “holes” [see figure 2.1.1] where the newly exposed background imagery differs from the known background. This anomaly can be removed by observing the object for a few frames, if it is a hole then update it to the reference frame.
This method fails to detect objects that are far away from the camera since the change recorded due to these objects is very small relative to the entire frame. These small changes fall below the global tolerance level defined and hence are not detected as a change. To overcome this, a localized segmentation based approach is used that is described in the next section.

4.3 Localized segmentation based approach to detect small and distant Objects

As described above, the Moving Object Detection approach presented in the previous section fails to detect small or distant moving objects. The motion of such objects amounts to small changes relative to the entire frame, which fall below the global Tolerance level and hence move undetected. To overcome this limitation, the image is divided into a number of small segments of a pre-defined size and the approach described in the previous section is localized to each segment. For localization of the approach a new threshold level is defined namely –

- **Segment Tolerance Level**: defines the minimum percentage of changed pixels that should be present in the segment to detect any motion in it.
The above mentioned threshold level is used as follows:

- IF percentile number of changed pixels in a segment $\geq$ Segment Tolerance Level
  Then
  Genuine object has caused the motion detected.
  Else
  Environmental noise has induced the motion detected.

As is evident from the above discussion, the moving object detection approach described in the previous section is a special case of the localized approach explained above wherein there is only one segment of size equal to the size of the entire frame. The smaller the segment size to which the localized approach is applied, the greater is the robustness of the system in the detection of small and distant objects. At the same time very small segment sizes i.e. excessive segmentation leads to increasing sensitivity to environmental disturbances.

All the detection parameters used in the moving object detection module of VigilCam Video surveillance system are shown in fig 3.2.

![Detection parameters](image)

Fig 4.2 Detection parameters used by moving object detection module of VigilCam video-surveillance system
4.4 Working Images of Moving Object Detection and Tracking Approach

Fig. 4.3 A moving person detected by the VigilCam Video-surveillance systems

Fig. 4.4 The moving person tracked by the VigilCam Video-surveillance systems
5.1 Introduction
In real world, video surveillance systems are set up to monitor busy areas where there is often a need to analyze the activity of multiple objects appearing in the view of the capture devices. The most primitive of this activity perception is to isolate all the objects in each frame grabbed by the capture devices. This chapter discusses a novel Multiple Object Extraction Algorithm devised by us, the application of the algorithm to an example case, the advantages and limitations of the algorithm. This chapter also includes the working images of our Vigil Cam application in relation to multiple object detection.

5.2 Multiple Object Extraction Algorithm

Input:
An image frame divided into fixed rectangular segments and the status of each segment (changed/unchanged). Refer to chapter 3 for a detailed description of the segmentation methodology.

Output:
A set of regions where in each region represents an object in the input frame. These regions provide the approximate shape and the exact position of the object in the frame.
Procedure:

Step 1:
Start at the top-left segment of the frame. Initially the object region list consists of no regions.

Step 2:
Execute a left-to-right segment wise pass until a changed segment is detected or until all segments have been traversed.

If a changed segment is detected, then go to step 3.

If all the segments have been traversed, then go to step 4.

Step 3:
Check the segments in the 8-neighbourhood of the newly found segment.

Case 1:
If none of the segments in its 8-neighbourhood are already part of an object region, then create a new object region and add it to the list of object regions. Place this segment into the newly created object region.

Case 2:
If all the changed segments in its 8-neighbourhood are part of a single object region created previously, then place this segment into that region.
Case 3:
If the changed segments in its 8-neighbourhood belong to different object regions then merge all those object regions into a single region and add the merged region into the list of object regions. Delete the child regions from the object region list. Place this segment into the merged region.
Go to step 2.

Step 4:
All the objects in the frame have been extracted and placed in the object region list. These regions can now be fed as input to other modules viz. Object classification and recognition, Behavior analysis modules for further analysis.

5.3 An Example
This section includes the application of the Multiple Object Extraction algorithm [section 5.2] to the frame shown below. As can be seen the frame consists of two objects. The changed segments have been shaded and numbered for convenience of explanation.

Fig 5.1 A Frame grabbed by the capture device
Applying the object extraction algorithm to the frame shown above we go through the following steps:

- In a left-to-right segment wise traversal starting from the top-left segment, the first changed segment detected would be segment-1.

- As is evident none of the changed segments in the 8-neighbourhood of segment-1 (Segments 2, 7, 8) are already a part of an object region. Hence a new Object region A is created and segment-1 is added to this region.

- The next changed segment detected would be segment-2. Since segment-1 in the 8-neighbourhood of segment-2 is already a part of object region A, segment-2 is added to region A. Similarly segment-3 is also added to region A. The object region list after processing segment-3 is shown below.

**Object Region List**

<table>
<thead>
<tr>
<th>Region A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
</tr>
</tbody>
</table>

Fig 5.2 Object Region List after segment 3 is processed

- The next changed segment would be segment-4. Since none of the changed segments in its 8-neighborhood (segments 5, 10, 11) are part of an object region a new object region B is created and segment-4 is added to this region.

- In this manner when segment-50 is detected there would be 3 regions A, B and C as shown in Fig 5.3
Since segment - 49 and segment – 40 in the 8-neighbourhood of segment-50 belong to different object regions – Region A and B respectively, regions A and B are merged into a single region AB and region AB is added to the list of object regions. Regions A and B are deleted from the Object region List and segment-50 is added to the merged region AB.

Proceeding in this fashion when all the segments have been processed there would be two object regions – AB and C. These two regions represent two objects as expected. The two objects extracted by applying the object extraction algorithm to the frame shown in Fig 5.1 are shown below.
These objects can now be fed as input to object classification and behavior inference modules for higher-level analysis.

### 5.4 Advantages of Multiple Object Extraction Algorithm

The Object Extraction algorithm described in section 5.1 extracts all the objects in a single segment-wise pass through the frame. The algorithm can be applied continuously after each segment is checked for any change. Hence the algorithm saves a lot of time and it works very fast. The Algorithm also makes use of the fixed rectangular segmentation of the frame [refer chapter 4] that was done to detect minute changes caused by minute objects and objects at a farther distance.
5.5 Limitations of Multiple Object Extraction Algorithm

- Since the object extraction algorithm takes advantage of a fixed rectangular segmentation of the frame [refer chapter 3], the exact shape of the object is not preserved.
- When two objects overlap then the algorithm identifies them as a single object. This problem stems from the monocular vision of a single camera, because monocular vision does not facilitate proper depth perception.

5.6 Working image examples of the Object Extraction Algorithm

The results of integrating the Multiple Object Detection Algorithm into the VigilCam video Surveillance system are shown in the following images.

![Fig 5.5 VigilCam Video surveillance system setup for road surveillance](image_url)
Fig 5.6 VigilCam Video surveillance system setup for road surveillance
6.1 Introduction

Live video, the primary input of a video surveillance system, is fundamentally different than a collection of images because video sequences contain action. Representations and methods need to be developed to explicitly consider the interpretation of action. Ironically, the interpretation of live video is often simpler than static images because the inherent continuous motion in the video imagery can be beneficially exploited to find out the active moving objects, thus making activity understanding methods simpler by narrowing down their requirements to perceive only active regions. Unlike the processing of static imagery, where the primary question to be answered is “What is in the scene”, the interpretation of video sequences focus on answering the question “What is happening in the scene”. There are numerous real-world scenarios that would benefit from an automatic video surveillance system capable of understanding actions taking place in a live video viz.

- **Aberrant behavior**: Identify vehicles moving in an unusual manner e.g. drivers, who are lost, don’t know where they are going, under the influence of drugs.

- **Indoor vigilance or commercial shop vigilance**: recognition of unusual activities such as people stealing products put on a display rack in commercial shops, frequent meet between people who r not co-workers e.t.c.
- **Restricted area surveillance** – “Trespassers will be prosecuted!”: Do not allow any sort of activity in certain high-security areas e.g. restrict any activity within a perimeter around certain exhibits in museums; restrict any object motion on certain delicate lawns in gardens.

This chapter depicts our approach for the *restricted area surveillance* scenario and how this action understanding method can be integrated into a video surveillance system. Restricted area surveillance is of significant importance in civilian areas such as museums, parks, and proprietary lands e.t.c.

### 6.2 Technical approach adopted for restricted area surveillance

As elucidated above restricted area surveillance involves strictly restricting even a small degree of activity within the perimeter of certain highly secured areas. We developed a drawing interface where in the user marks out restricted and permitted areas on the reference frame which is virtually a site map. The restricted areas defined by the user are then used by the activity perception module for restricting any sort of activity in those areas. In order to be ensured that any sort of complex area can be defined, we defined two types of regions that the user can define:

- **Permitted Region**: the region in which activity is permitted i.e it is considered as common behavior.
- **Clip Region**: this region is used to cut out a certain part of a permitted region. It helps in defining complex regions.

Using the two aforementioned regions any complex area can be defined. A few images of the custom drawing interface illustrating the definition of complex areas are given below.
Fig 6.1 custom drawing interface that allows the user to specify restricted/permitted areas on reference Frame or site map.

Fig 6.2 an example area defined using the custom drawing interface; Green – permitted region, red - clip region
6.3 Integrating restricted area surveillance into video surveillance System

The integration of the restricted area surveillance described in previous section into the existing framework of a video surveillance system is shown in the block diagram below.

As is evident from the above block diagram, the moving object detection module finds out the centroid of the objects in each frame and if these centroids lie inside the restricted region i.e. if the object is found in the restricted area the human operator is alerted through an alarm.
In the Future the existing framework for detection, tracking and behavior inference can be extended in a variety of directions:

- The current version of this software uses *Background Subtraction* for moving object detection. This creates holes when stationary objects move. This anomaly can be removed by developing a model for updating the reference frame. Updating the Reference frame refers to observing the object for a few frames, if it does not move then update the object region to reference image.

- The current version of the software is too sensitive to environmental changes like wind, rustling of leaves, change in illumination etc, which can be eliminated. For example, motion due to rustling of leaves can be removed by detecting the to and fro motion of the leaves across some frames. These regions can then be ignored.

- We can extend our tracking methods to incorporate multiple cameras. This will require the coordination between the cameras to ensure that the same object is being tracked in each camera. Using multiple cameras also facilitates *depth perception*, which can be used to detect overlapped objects .It can also be used for finding the distance of objects.

- We can extend our object isolation algorithm to not only detect multiple objects but also to track them. This can be done by correlating the object regions based on certain criterion across successive frames.
• Better behavior inference can be done by developing techniques for classifying complex activities such as interaction between objects or recognizing particular instances of people or vehicles and detecting common activities between objects.

• The current system can be integrated into an object classification and recognition module. This facilitates in recognition of objects in a frame. Object classification can be done using neural networks and also by other conventional classification techniques.
References


- TWAIN – Standard for Image Acquisition Devices. Twain 1.9 specification.
**Background Subtraction**

Background subtraction is a technique used for detecting moving objects in a live video captured frame from a video capture source. It involves subtracting out the background or stationary components from each frame in a live video sequence.

**Digital Image Processing**

Digital Image Processing is a field of computer science that involves the processing of digital images by means of a digital computer. A digital image is composed of a finite number of small elements called pixel. Each pixel has a particular location and also an intensity attribute that specifies its perceptual color.

**Reference Frame**

Reference frame is a snapshot of the background or more precisely a snapshot of all the stationary components present in front of the surveillance camera. It can be used to detect moving objects by subtracting out the stationary components or the reference frame from each frame procured by the camera.
Twain

Twain is an image capture Application Program Interface (API) for Microsoft windows and Apple Macintosh Operating systems. The standard was first released in 1992 and is currently ratified at version 1.9 as of January 2000. It is typically used as an interface between image processing and a scanner or digital camera.

Local Tolerance Level

Local Tolerance Level is one of the detection parameters of a threshold-based technique adopted by the VigilCam video surveillance system to negate the effect due to environmental disturbances. It defines the minimum percentile by which the gray level of a pixel should change to be detected as a changed pixel.

Global Tolerance Level

Global Tolerance Level is one of the detection parameters of a threshold-based technique adopted by the VigilCam video surveillance system to negate the effect due to environmental disturbances. It defines the minimum percentage of changed pixels that should be present in a frame to detect any motion in it.

Segment Tolerance Level

Segment Tolerance Level is one of the detection parameters of a threshold-based technique adopted by the VigilCam video surveillance system to negate the effect due to environmental disturbances. It defines the minimum percentage of changed pixels that should be present in the segment to detect any motion in it.
FreeImage

FreeImage is an Open Source library project for developers who would like to support popular graphics image formats like PNG, BMP, JPEG, TIFF and others as needed by today's multimedia applications. The library comes in two versions: a binary distribution that can be linked against any 32-bit C/C++ compiler and a source distribution. Workspace files for Microsoft Visual C++ 6 are provided, as well as makefiles for Linux. For further details refer to the Freeimage site http://freeimage.sourceforge.net/intro.html.

Video For Windows

Microsoft® Video for Windows® (VFW) provides applications with a simple, message-based interface to access video and waveform-audio acquisition hardware and to control the process of streaming video capture to disk. VFW was introduced in 16-bit Windows. In addition, it provides control of video sources that are Media Control Interface (MCI) devices so the user can control (through an application) the start and stop positions of a video source, and augment the capture operation to include step frame capture. For further details refer the MSDN library http://msdn.microsoft.com/.