**CHAPTER 1**

**INTRODUCTION**

* 1. **Problem Definition**

The study of this problem is based on the field of image processing. In this project, we intend to provide an approach to the problem of Multiple Object Tracking, which has a wide range of applications such as video editing, video surveillance, security etc. The main focus is to track multiple objects in a scene individually in the presence of Occlusion. Since the objects in the world exhibit complex interactions, when captured in a video sequence, some interactions manifest themselves as occlusions. Occlusion results in mistaken match when finding the most similar candidate in a tracking-by-detection strategy.

**1.2 Purpose**

The purpose of this document is to present a detailed description of the object tracking software. This document will explain the purpose and features of the tracking system software, the interfaces of the software, what the software will do, the constraints under which it must operate and how it will respond to user inputs which in this case is a video input.

* 1. **Image Processing**

In [electrical engineering](http://en.wikipedia.org/wiki/Electrical_engineering) and [computer science](http://en.wikipedia.org/wiki/Computer_science), image processing is any form of [signal processing](http://en.wikipedia.org/wiki/Signal_processing) for which the input is an image, such as a [photograph](http://en.wikipedia.org/wiki/Photograph) or [video frame](http://en.wikipedia.org/wiki/Video_frame); the [output](http://en.wikipedia.org/wiki/Output) of image processing may be either an image or, a set of characteristics or [parameters](http://en.wikipedia.org/wiki/Parameter) related to the image. Most image-processing techniques involve treating the image as a [two-dimensional](http://en.wikipedia.org/wiki/Two-dimensional) [signal](http://en.wikipedia.org/wiki/Signal_(electrical_engineering)) and applying standard signal-processing techniques to it.

Image processing usually refers to [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing), but [optical](http://en.wikipedia.org/wiki/Optical_engineering) and [analog image processing](http://en.wikipedia.org/wiki/Analog_image_processing) also are possible. Image processing is about general techniques that apply to all of them. The acquisition of images (producing the input image in the first place) is referred to as [imaging](http://en.wikipedia.org/wiki/Imaging).

Digital image processing is the use of computer [algorithms](http://en.wikipedia.org/wiki/Algorithm) to perform [image processing](http://en.wikipedia.org/wiki/Image_processing) on [digital images](http://en.wikipedia.org/wiki/Digital_image). As a subcategory or field of [digital signal processing](http://en.wikipedia.org/wiki/Digital_signal_processing), digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of [Multidimensional Systems](http://en.wikipedia.org/wiki/Multidimensional_Systems).

* 1. **WORKING ENVIRONMENT**

**1.4.1 MATLAB ENVIRONMENT**

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB (matrix laboratory) is a [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems). These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations.

MATLAB provides many functions for image processing and other tasks. Most of these functions are written in the MATLAB language and are publicly readable as plain text files. Thus the implementation details of these functions are accessible and open to scrutiny. The defense can examine the processing used in complete detail, and any challenges raised can be responded to in an informed way by the prosecution. This makes MATLAB very different from applications, such as Photoshop.

It should be noted that some MATLAB functions cannot be viewed. These are generally lower level functions that are computationally expensive and are hence provided as 'built in' functions running as native code. These functions are heavily used and tested and can be relied on with considerable confidence.

MATLAB also has easy to use graphics commands that make the visualization of results immediately available. Special applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

* 1. **Applications**

Object tracking has many applications in today's diverse range of embedded systems and is an important task within the field of computer vision. The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms. There are three key steps in video analysis: Detection of interesting moving objects, tracking of such objects from frame to frame and the analysis of object tracks to recognize their behavior. Therefore, the use of object tracking is pertinent in the tasks of:

* **Motion-Based Recognition**-Human identification based on gait, automatic object Detection.
* **Automated Surveillance** - Monitoring a scene to detect suspicious activities or unlikely events.
* **Human-Computer Interaction**- Gesture recognition, eye gaze tracking for data input to computers.
* **Facial Motion Capture**-This technology tracks the motion of the person/persons when they sit in front of the camera. Traditional marker based systems apply up to 350 markers to the actors face and track the marker movement with high resolution cameras. Next generation systems such as CaptiveMotion utilize offshoots of the traditional marker based system with higher levels of details.
* **Tracking Objects in Flickering Lighting**- Sometimes, the lighting in the videos flicker, thereby making different objects difficult to recognize. And also when the objects are moving very fast in the videos, it becomes difficult to analyse the motion of the objects. Object tracking can be used to analyse the motion of the objects in fast moving conditions and in bad lighting conditions.

**1.6** **Graphical User Interface**

**1.6.1** **Subject Object Tracker GUI**

The *Subject Object Tracker* software offers a Graphical User Interface which is animated, 3D based and has a point and click interface.

The GUI was designed using HTML, CSS and WebKit. The softwares, Photoshop and Maya were used to incorporate 3D effects.

Extensive use of 3D objects, animations and transitions are incorporated to completely eliminate typing work.

The Subject Object Tracker interface consists of 3 Stages

* **The Introduction Stage**- This part of the GUI provides a brief description about object tracking and the problem encountered in the same.
* **The Selection Stage**- This part of the GUI provides two options namely a “WEBCAM” option and an “UPLOAD FILE” option. The WEBCAM option is for streaming live input from the webcam whereas the UPLOAD FILE option is to upload a file from the disk or memory.
* **The Output Stage**- This part of the GUI outputs the Tracked Video Output which is displayed to the user.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 SURVEY OF PAPERS**

**2.1.1 Segmentation and Tracking Multiple Objects Under Occlusion From Multiview Video (2011)**

The approach used by our tracking software in tracking multiple objects is based on the above IEEE paper. Qian Zhang et al[1] presented a multi-view approach to segment the foreground objects consisting of a group of people into individual human objects and track them across the video sequence. Depth and occlusion information recovered from multiple views of the scene is integrated into the object detection, segmentation, and tracking processes. Multiple cues are employed to segment individual human objects from the group. To propagate the segmentation through video, each object region is independently tracked by motion compensation and uncertainty refinement, and the motion occlusion is tackled as layer transition.

**Drawbacks**: The whole system cannot be realized in real time since the offline operations are quite time consuming

**2.1.2 Multi-view video based multiple objects segmentation using graph cut and spatiotemporal projections (2009)**

In this paper, Qian Zhang and King Ngi Ngan[2] present an automatic algorithm to segment multiple objects from multi-view video. The approach is that Initial Interested Objects (IIOs) are automatically extracted in the key view of the initial frame based on the saliency model. Multiple objects segmentation is decomposed into several sub-segmentation problems, and solved by minimizing the energy function using binary label graph cut. The experiments are implemented on a couple of multi-view videos with real and complex scenes. Excellent subjective results have shown the robustness and efficiency of the algorithm that they proposed.

**2.1.3 Multi View Image Surveillance and Tracking (2002-2003)**

The author, James Black et al[3] provided a framework for object tracking using multiple camera views. The thesis applies a novel framework for performance evaluation of video tracking algorithms. This framework is a novel contribution for performance evaluation, since it is possible to automatically generate large volumes of testing datasets without the need to perform exhaustive manual ground truth generation. The framework allows any video tracking algorithm to be evaluated over a variety of datasets, which vary in perceptual complexity and represent a number of different tracking scenarios. Using the proposed method, it is possible to resolve both dynamic and static object occlusions.

**Drawbacks**: The generated video sequences will be biased towards the motion detection algorithm used to capture the original ground truth tracks. A few ground truth tracks will be observed in regions where tracking or detection performance is poor.

**2.1.4 An Efficient Object Tracking Algorithm with Adaptive Prediction of Initial Searching Point (2006-2007)**

Jiyan Pan, Bo Hu, and Jian Qiu Zhang[4] proposed an algorithm which adaptively predicts possible coordinate transform parameters for the next frame and selects them as the initial searching point when looking for the real transform parameters. By doing so, tracking algorithms have less risk of being distracted by local maxima resulting from interferences, and tracking performance is thus improved. This paper uses an adaptive Kalman filter to achieve this purpose, but instead of directly filtering the values of transform parameters, we apply the Kalman filter on the changing rate of those parameters to effectively predict their future values. The paper also quantitatively analyzes the cause of the model noises in the Kalman filter and also derives their analytical expressions. However the paper is applicable only for single object tracking and not multiple objects. It also does not consider the problem of occlusion.

**Drawbacks**: It cannot handle multiple objects i.e it only handles single object. It also does not handle the problem of occlusion.

**2.1.5 Robust People Tracking with Global Trajectory Optimization (2006-2007)**

Jerome Berclaz et al[5] presented an algorithm that can reliably track multiple persons in a complex environment and provide metrically accurate position estimates. This is achieved through global optimization of their trajectories over 100-frame batches. This introduces a 4 second delay between image acquisition and output of the results, which the authors believe to be compatible with many surveillance applications given the robustness increase it offers. However there are many possible extensions of this work. The most obvious ones are improvements of the stochastic model. The color model could be refined by splitting bodies into several uniform parts. Similarly, the motion model could take into account consistency of speed and direction. Modeling the avoidance strategies between people would also help.

**Drawbacks**: The stochastic model proposed can be further improved. The motion model does not take into account consistency of speed and direction.

**2.1.6 Robust Occlusion Handling in Object Tracking(2007-2008).**

In this paper, Jiyan Pan and Bo Hu[6] proposed an object tracking algorithm which demonstrates high robustness against occlusions. This is achieved by effectively analyzing the occlusion situation to generate proper template mask and rectifying initial erroneous target location caused by occlusions. It utilizes the spatiotemporal context to analyze the occlusion situation of the target. The occlusion analysis is performed in a progressive manner, so that we high reliability and high resolution is achieved simultaneously. The proposed algorithm also performs variant-mask template matching VMTM) to remove the error of the target location introduced by performing template matching when the template mask reflecting the current occlusion situation is unavailable yet.

**Drawbacks**: When complete occlusions occur, the algorithm will fail. If a part of an occluder has exactly the same appearance as a nearby part of the target, the algorithm will not be able to detect it. This is applicable only for single object and not multiple objects.

**2.1.7 A Linear Programming Approach for Multiple Object Tracking (2007-2008)**

Hao Jiang et al[7] presented a novel framework for optimizing multiple object tracking that can be solved efficiently based on a linear programming relaxation. The system models object tracking as a multi-path searching problem. It also explicitly models track interaction, such as object spatial layout consistency or mutual occlusion, and optimizes multiple object tracks simultaneously. The proposed scheme explicitly models track interaction such as the spatial layout constraint and object mutual occlusion. Experiments show that the proposed scheme works robustly in tracking objects with complex interactions in long video sequences. The linear program relaxation can also be solved more efficiently than previous methods such as extended dynamic programming.

**Drawbacks**: The system does not implicitly propose a method to handle complete occlusion.

**2.1.8 Robust and Accurate Object Tracking under Various Types of Occlusions (2008-2009)**

Jiyan Pan et al[8] proposed an object tracking solution that contains a set of algorithms aiming at enhancing the robustness and accuracy of object tracking when various types of occlusions occur. In the proposed solution, the content-adaptive progressive occlusion analysis (CAPOA) algorithm effectively acquires the current occlusion situation by: scanning the ROI progressively, using the joint information provided by three sources and making decisions according to content-adaptive thresholds. The variant mask template matching (VMTM) is performed to rectify the target location that initially might be inaccurate due to the influence of occlusions. The drift-inhibitive masked Kalman appearance filter (DIMKAF) significantly reduces template drift by correctly evaluating the noise models and obtaining the optimal Kalman gains for all the template pixels at every frame. The local best match authentication (LBMA) method reliably detects the end of complete occlusions by authenticating the genuineness of local best matches. However the system is not designed to track multiple objects.

**Drawbacks:** When a human head rotates from a frontal view to a rear view, the dark hair that gradually becomes visible from one side of the face sometimes fails to be recognized as part of the target. During the period of complete occlusion, the motion of the target is so irregular that the target is out of the searching range when it reemerges. And the system is also not designed to track multiple objects.

**2.1.9 Robust Multi-Person Tracking from a Mobile Platform (2009-2010)**

Andreas Ess et al[9]presented an integrated system for multi-person tracking from a mobile platform. The approach jointly estimates camera position, stereo depth, object detection, and tracking.

The interplay between those components is represented by a graphical model. The different modules (appearance-based object detection, depth estimation, tracking, and visual odometry) were integrated in a graphical model and exchanged information using a set of feedback channels. This close coupling proved to be a key factor in improving system performance.

A set of failure prevention, detection, and recovery mechanisms were also proposed. The resulting system can handle very challenging scenes and track many interacting pedestrians simultaneously and over long time frames.

**Drawbacks**: In this method ignoring odometry failures can lead to erratic tracking behavior, since tracking relies on correct 3D world coordinates. Small drift is produced when the visual odometry is restarted from the scratch.

**2.1.10 Tracking Multiple Occluding People by Localizing on Multiple Scene Planes (2009-2010)**

The authors, Saad M. Khan and Mubarak Shah[10]presented an algorithm that can reliably track multiple people in a complex environment. This is achieved by resolving occlusions and localizing people on multiple scene planes using a planar homographic occupancy constraint. By combining foreground likelihood information from multiple views and obtaining the global optimum of space-time scene occupancies over a window of frames, we segment out the individual trajectories of the people. Unlike other multi-view approaches that require fully calibrated views, our approach is purely image-based and uses only 2D constructs. Detection and tracking are performed simultaneously by graph cuts segmentation of tracks in the space-time occupancy likelihood data

**Drawbacks**: If a person is not part of the high foreground likelihood regions in the views, it may cause a missed detection (false negative).

**2.1.11 Joint Pose Estimation of Multiple Persons (2010-2011)**

Marcin Eichner and Vittorio Ferrari[11]presented a novel multi-person pose estimation framework, which extends pictorial structures (PS) to explicitly model interactions between people and to estimate their poses jointly. Interactions are modeled as occlusions between people. They first proposed an occlusion probability predictor, based on the location of persons automatically detected in the image, and incorporated the predictions as occlusion priors into their multi-person PS model. Their model includes an inter-people exclusion penalty, preventing body parts from different people from occupying the same image region.

**Drawbacks**: The algorithm produced does not handle occlusion among people. It mainly concentrates on images.

**2.1.12 Efficient Block Division Model For Robust Multiple Object Tracking (2010-2011)**

The authors, Wenhan Luo et al[12]proposed a framework for occlusion handling in multiple object tracking. spatial information is introduced to avoid the mistaken match between object and candidate Based on block division appearance , the proposed framework can efficiently and accurately track objects through occlusion, obtaining correct occlusion relationship. Occlusion is deduced by monitoring the variation of each block.

**CHAPTER 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Minimum Hardware Requirement**

|  |  |
| --- | --- |
| **CATEGORY** | **REQUIREMENT** |
| CPU | Intel Core2™ Duo, minimum 1.6 GHz |
| RAM | Minimum 512 MB (1 GB recommended for large views, 1 GB recommended on Microsoft Windows Vista). |
| Network | Ethernet, 1000 Mbit or higher recommended. |
| Graphics Adapter | AGP or PCI-Express, minimum 800X600,  16 bit colors |
| Camera | Two HD Web Cameras |

* 1. **Software Specification**

1. Software Specification For Deployment
   1. Chrome Browser
   2. Drivers for the Web Cameras
2. Software Specification For Development
   * 1. Matlab 2011
     2. NetBeans IDE
     3. GlassFish Application Server

**Matlab 2011**

MATLAB (matrix laboratory) is a [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). Developed by [MathWorks](http://en.wikipedia.org/wiki/MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including C,C++, [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and [Fortran](http://en.wikipedia.org/wiki/Fortran).

MATLAB can be regarded as a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

MATLAB is one of a few languages in which each variable is a matrix (broadly construed) and "knows" how big it is. Moreover, the fundamental operators (e.g. addition, multiplication) are programmed to deal with matrices when required.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](http://en.wikipedia.org/wiki/MuPAD) [symbolic engine](http://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](http://en.wikipedia.org/wiki/Symbolic_computing) capabilities. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems).

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of [engineering](http://en.wikipedia.org/wiki/Engineering), [science](http://en.wikipedia.org/wiki/Science), and [economics](http://en.wikipedia.org/wiki/Economics). MATLAB is widely used in academic and research institutions as well as industrial enterprises.

In addition to the MATLAB system itself, Mathworks offers a variety of Toolboxes which help in extending the capability of the computing environment. The *Image Processing toolbox* for instance supports a wide range of Image Processing operations like Image analysis, segmentation, morphology, feature extraction, and measurement. It also supports operations on Image sequence and video display and so on. *Computer Vision System Toolbox* can be used to compute the depth map between two rectified stereo images.

Another important toolbox which is of relevance to our project is the Image Acquisition Toolbox which allows us to connect the hardware (which in this case are the webcams to generate stereo vision) and set acquisition parameters, and preview and acquire image data. We can also log the data to MATLAB in several formats, and also generate an AVI file, right from the tool.

**Netbeans IDE**

NetBeans IDE is a free, open-source, cross-platform IDE with built-in-support for Java Programming Language. The NetBeans IDE is written in Java and can run on Windows, Mac OS, Linux, Solaris and other platforms supporting a compatible [JVM](http://en.wikipedia.org/wiki/Java_Virtual_Machine). A pre-existing JVM or a [JDK](http://en.wikipedia.org/wiki/Java_Development_Kit) is not required.

The NetBeans platform allows applications to be developed from a set of modular [software components](http://en.wikipedia.org/wiki/Software_component) called ***modules***. Modularity is achieved because all the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the [Java language](http://en.wikipedia.org/wiki/Java_(programming_language)), editing etc.

NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules.

The NetBeans IDE has many features and tools for each of the Java platforms. Those in the following list are not limited to the Java SE platform but are useful for building, debugging, and deploying applications and applets:

**Source Code Editor**

* Syntax highlighting for Java, JavaScript, XML, HTML, CSS, JSP, IDL
* Customizable fonts, colors, and keyboard shortcuts
* Live parsing and error marking
* Pop-up Javadoc for quick access to documentation
* Advanced code completion
* Automatic indentation, which is customizable
* Word matching with the same initial prefixes
* Navigation of current class and commonly used features
* Macros and abbreviations
* Goto declaration and Goto class
* Matching brace highlighting
* JumpList allows you to return the cursor to previous modification

The NetBeans IDE also provides full-featured refactoring tools, which allow you to rename and move classes, fields, and methods, as well as change method parameters. It also provides a debugger and a GUI Builder.

NetBeans allows you to see all your objects in a project represented as nodes of a tree, each having its own icon to represent the type of object the node represents. Within the Files tab, you can easily view the trees and representative nodes. If you double-click on a node, it opens up into a subtree that contains more detail. You can collapse or expand trees as necessary. Right-clicking on any node provides easy access to specific functions that you can perform and tools that you can use on that object. Expand the subtrees of the project node that you just created, and you will notice that the fields, constructors, methods, and bean patterns appear as node branches.

NetBeans IDE provides full support for Java EE 6. We can develop components including web pages, servlets and web services. Finally the developed application can be deployed to an application server such as GlassFish, WebLogic, Tomcat or JBoss server.

**GlassFish**

GlassFish is an [open-source](http://en.wikipedia.org/wiki/Open-source_software) [application server](http://en.wikipedia.org/wiki/Application_server)  project started by [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) for the [Java EE](http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) platform. GlassFish is the [Reference implementation](http://en.wikipedia.org/wiki/Reference_implementation) of Java EE and as such supports [Enterprise JavaBeans](http://en.wikipedia.org/wiki/Enterprise_JavaBeans), [JavaServer Pages](http://en.wikipedia.org/wiki/JavaServer_Pages), servlets, etc. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. Optional components can also be installed for additional services.

GlassFish uses a derivative of [Apache Tomcat](http://en.wikipedia.org/wiki/Apache_Tomcat) as the [servlet](http://en.wikipedia.org/wiki/Servlet) container for serving Web content, with an added component called [Grizzly](http://en.wikipedia.org/w/index.php?title=Grizzly_(software)&action=edit&redlink=1) which uses Java [New I/O](http://en.wikipedia.org/wiki/New_I/O) (NIO) for scalability and speed.

GlassFish Server 3.1 is also the fastest open source application server offering advanced features such as application versioning, application-scoped resources, and great development tool support from NetBeans 7.0, Eclipse and other popular IDEs.

* 1. **Requirements Analysis**
     1. **Scope**

The object tracking software will be an useful and efficient software used for Video Surveillance. We present an approach for tracking varying number of objects through both temporally and spatially significant occlusions. To this end, tracking is performed at both the region level and the object level.

Frame Difference technique is used for motion estimation. At the object level, each object is located based on spatial distributions and inter-occlusion relationships. We propose an architecture that is capable of tracking objects even in the presence of long periods of partial or full occlusions.

We demonstrate the viability of this approach by experimenting on several videos .The software will be deployed as a web application to address the portability issue. The user interface is designed to function on all possible web browsers.

**3.3.2 Functional Requirements**

The FunctionalRequirements document (also called Functional Specifications or Functional Requirement Specifications), defines the capabilities and functions that a [System](http://www.ofnisystems.com/Validation/Validation_Terminology.htm#System) must be able to perform successfully.

1. The software must generate a tracking report and must generate depth and occlusion maps.
2. The software must handle both Partial and Complete occlusions.
3. The software must handle both inter-object and intra-object occlusions.
4. The Software must adopt a multi-view approach for object tracking.
5. The software must perform Region Based Tracking.

**3.3.3** **Non Functional Requirements**

1. The software shall work both Online and Offline.
2. The Software shall be capable of supporting avi video format.
3. The Software shall have a well designed User Interface incorporating 3D objects and transitions.
4. The Software shall work on all platforms thereby achieving Portability.

**CHAPTER 4**

**System Design**

* 1. **Architecture Diagram**

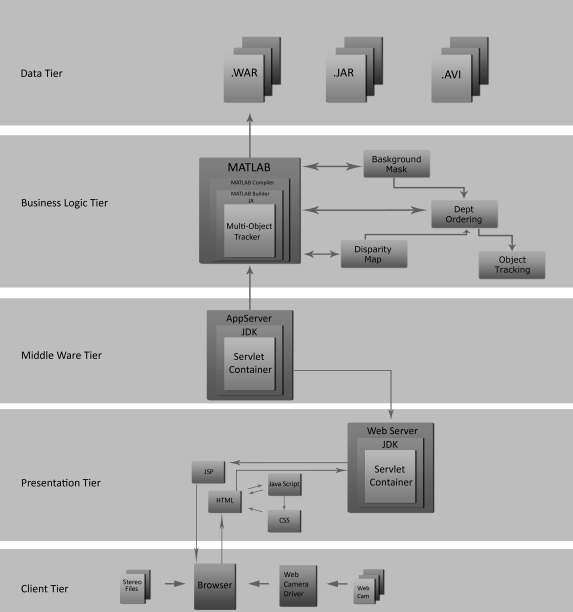
****

Figure 4.1 Interaction view of the architecture

As seen in figure 4.1, the interaction view of the architecture is modeled as a n-tier architecture. The components and their relationship with the components in the other tiers are clearly illustrated .

**4.2 METHODOLOGY**

****

Figure 4.2 Block diagram of proposed methodology

Our project focuses on the development of a software that can be used to track moving objects continuosly in a video sequence in the presence of occlusion. The input which is the video sequence is captured by two web cameras which are positioned and aligned suitably to achieve stereo vision.

Stereo Vision helps in the extraction of the 3D information or the depth information. This can inturn be used to compute disparity between the images obtained by stereo. It is however important to note that disparity and the depth are inversely proportional.

We propose to segment and track multiple objects correctly and consistently when they overlap with each other under occlusion in a complex scene using a model consisting of three stages. First the process of segmentation, wherein the foreground objects are separated individually so that they can be tracked separately. The phase prior to the segmentation is the Background Subtraction wherein the foreground objects are separated from the background.

Depth, occlusion, motion, and color information are used as cues to achieve segmentation and object detection. Occlusion is reasoned using the depth and occlusion data cues. Finally, tracking multiple objects simultaneously is achieved while handling inter-object occlusions.

The two cameras collaboratively work to compute depth, accurate object silhouette and occlusion information of a specific view from the other view of the scene, which are then integrated into the object detection, segmentation, and tracking processes.

**4.3 OTHER UML DIAGRAMS**

**4.3.1 CLASS DIAGRAM**

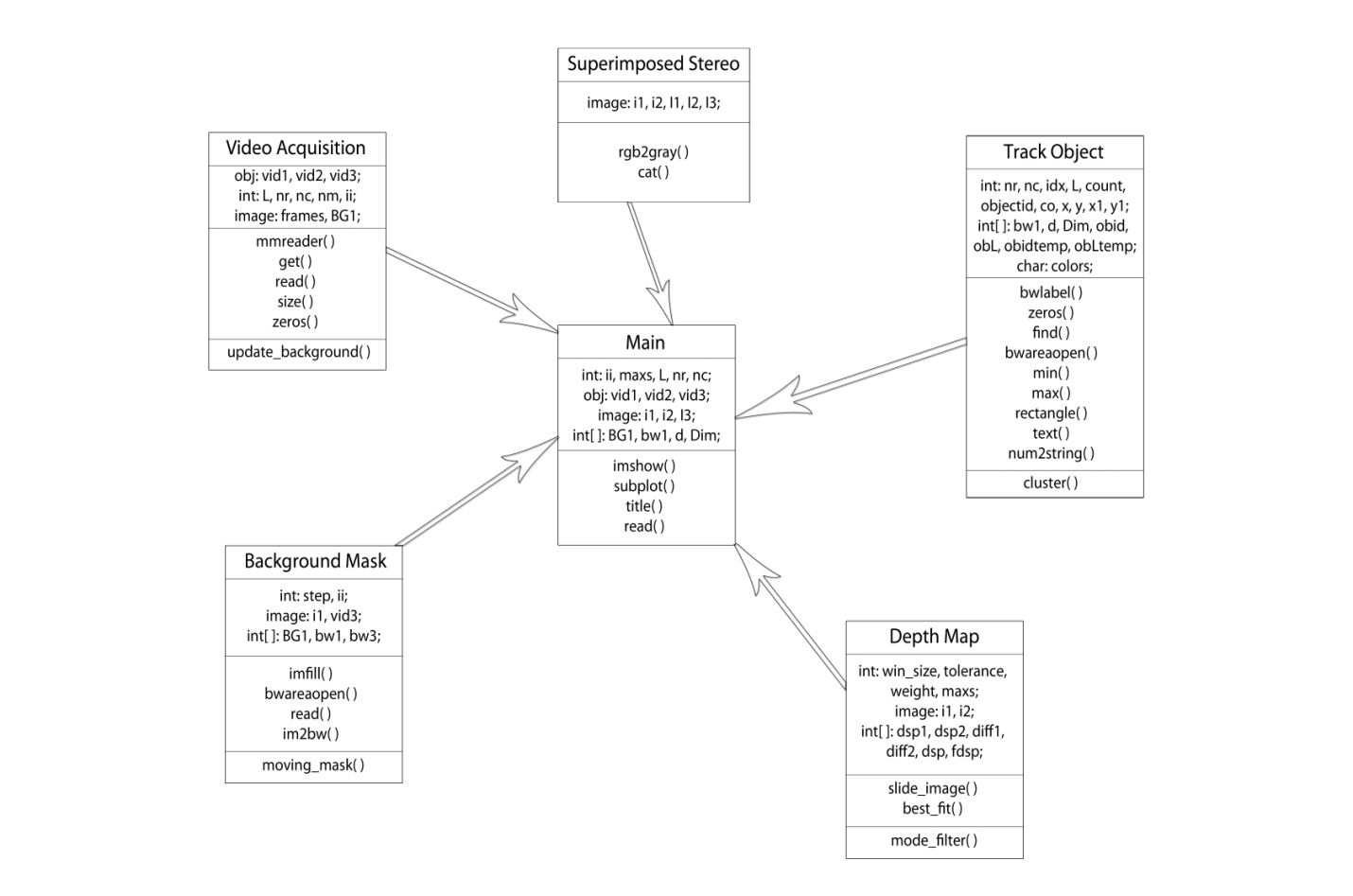
****

Figure 4.3 Class Diagram

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 OVERVIEW**

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component or other computer system through programming and deployment. There is a possibility of having more than one implementation for a given specification or standard.

The main aim of the implementation phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design.

Implementation of any software is preceded by important decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as real environment in which the system works, the speed that is required, the security concerns and other implementation specific details etc.

The purpose of System Implementation is to ensure that the upcoming system deployment and transition occurs smoothly, efficiently, and flawlessly. To implement a system successfully, a large number of inter-related tasks need to be carried out in an appropriate sequence.

There are certain implementation decisions that are taken before implementing a project successfully. Some of the implementation decisions that we considered in our project were , the selection of the programming language for development of the dissertation work, selection of appropriate toolboxes for Video acquisition, stereo vision etc.

**5.2 FUNCTIONAL FLOW**

****

Figure 5.1 Hierarchical structure diagram

* **Main-** Indicates the function which performs multiple object tracking with occlusion reasoning
* **Superimposed Stereo-** Function that combines the left and right view of the stereo by superimposing them. Stereo helps in computing the disparity which happens to be the difference between the left and right image in the video sequence.
* **Mask Background-** Function which separates the foreground and the background. It is based on simple subtraction( frame difference)
* **Video Acquisition-** Function which receives the video input. The video sequence from the webcam is captured using an Image Acquisition Toolbox.
* **Update Background-** This function updates the background by considering the median of frames for the moving mask.
* **Track Object**- This function effectively tracks the object which has been clustered or ordered based on depth.
* **Mode Filter-** Used to get filtered disparity in the output.

**5.3 ACTIVITY DIAGRAM**

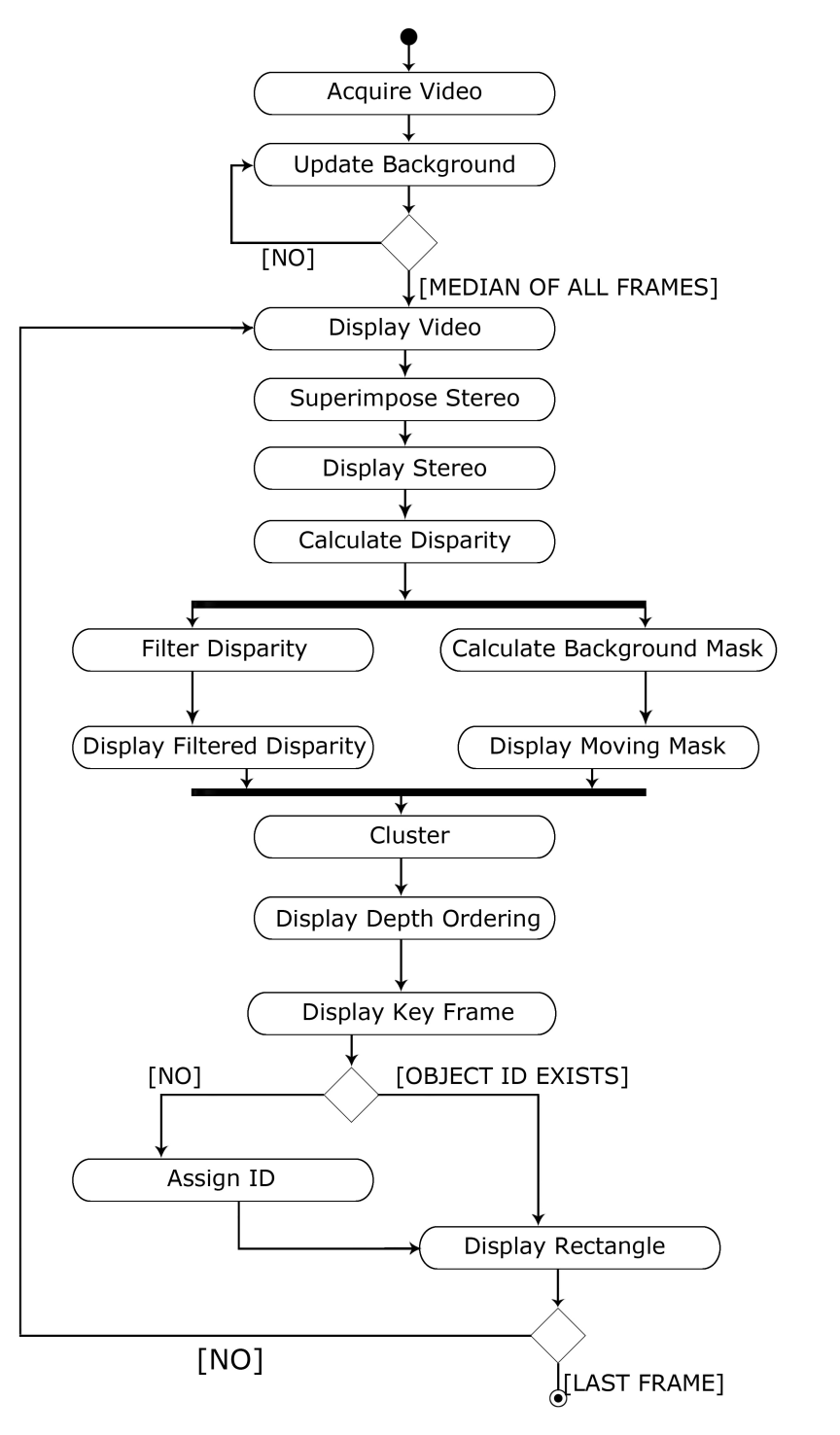
****

Figure 5.2 Activity diagram

**CHAPTER 6**

**VERIFICATION AND VALIDATION**

**6.1 TESTING**

Testing provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Software testing is the process used to help identify the correctness, completeness, security and quality of the developed computer software. Testing is a process of technical investigation performed on behalf of stakeholders i.e. intended to reveal quality related information about the product with respect to the context in which it is intended operate.

During testing, software engineering produces a series of test cases that are used to rip apart the software we have produced.

**6.2 TESTING OBJECTIVES**

The two objectives of testing are

1. To demonstrate to the developer and customer that the software meets its requirements.
2. To discover faults or defects in the software where the behavior of the software is incorrect or does not conform to the behavior to its specification.

**6.3 TESTING TYPES**

* COMPONENT TESTING
* INTEGRATION TESTING

**6.3.1 COMPONENT TESTING**

We verify and validate the result by comparing the output obtained by each component or module of our tracking software with the output of the base paper. Given below are snapshots of the outputs of Background mask and stereo disparity components or modules. We can clearly see that the component outputs matches that of the base paper output.

**EXPECTED OR DESIRED OUTPUT**

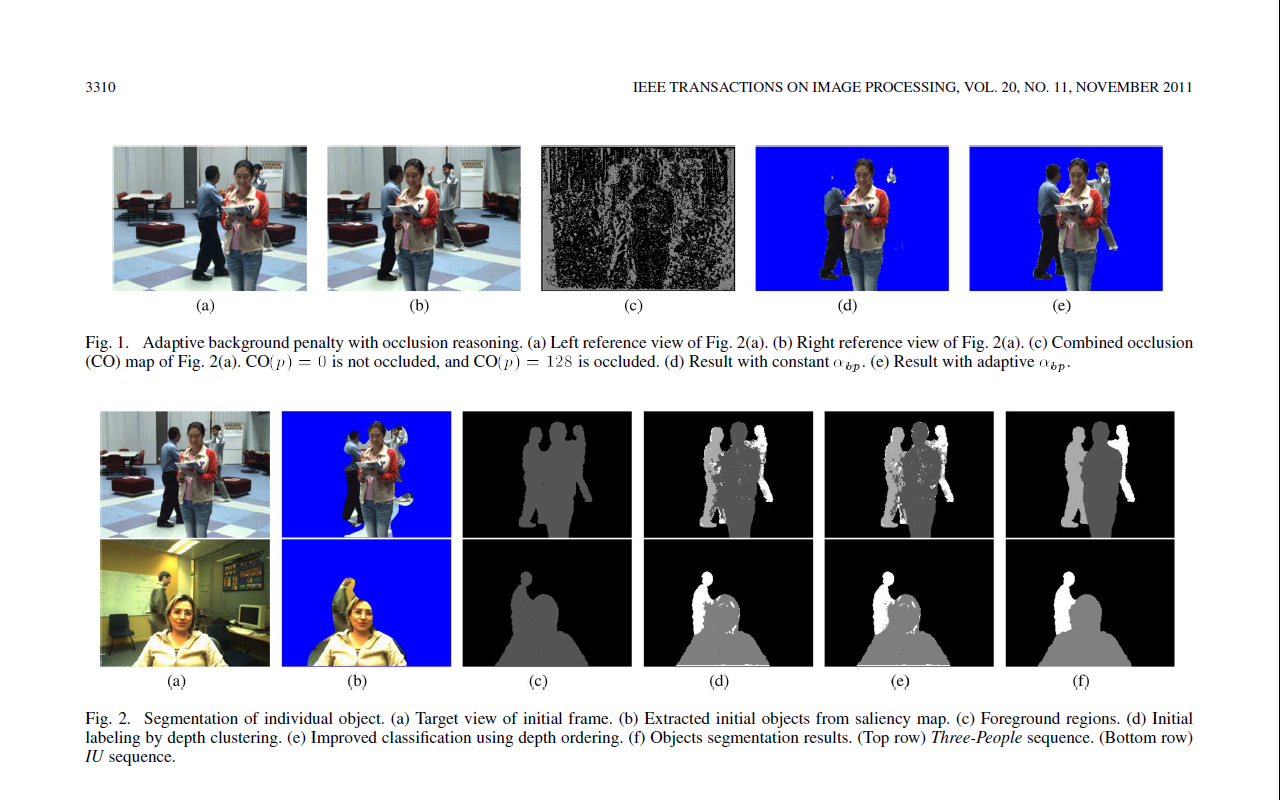


Figure 6.1 Segmented output of base paper

**INDIVIDUAL COMPONENT OUTPUT OBTAINED BY OUR TRACKING SOFTWARE**

**a. Background Mask**

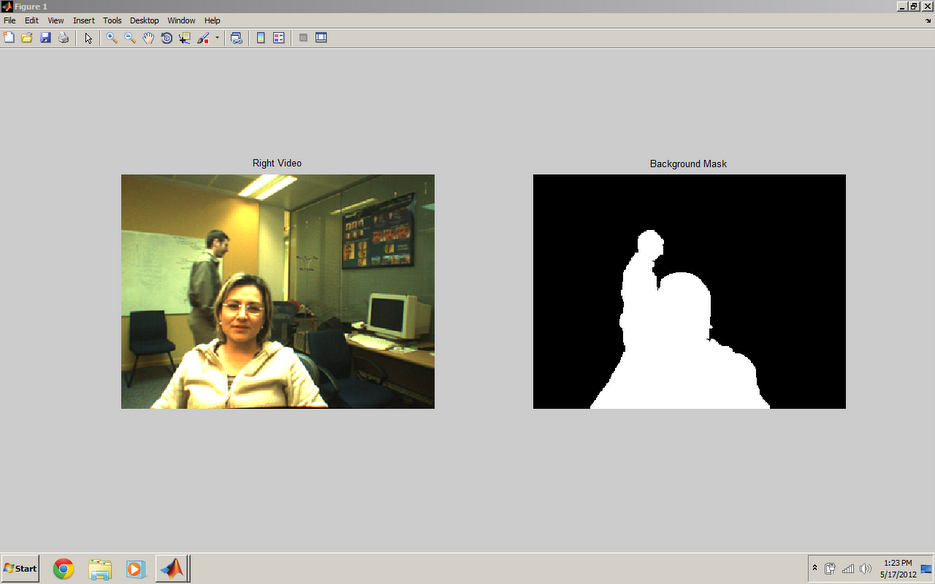
****

Figure 6.2 Background mask output

As seen in the above figure,

* The objects of the foreground are separated from the background.
* This step then aids in depth ordering based on the depth of the object from the point of reference.

**b. Disparity**

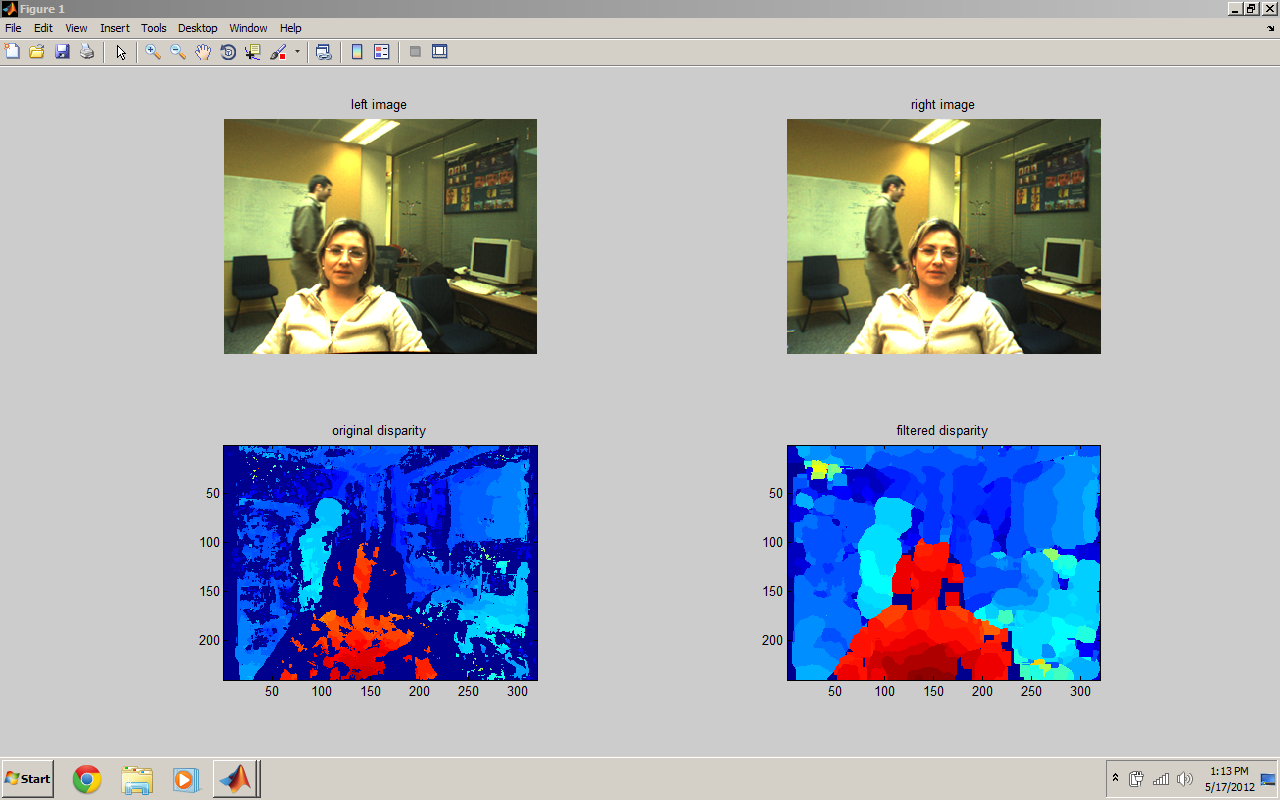
****

Figure 6.3 Disparity output

As seen by the above snapshots, we find that

* The individual component outputs namely the Background mask and the Disparity exactly match that of the expected base paper output

**6.3.2 INTEGRATION TESTING**

We performed integration testing by checking if all the modules of the software were able to collectively work together to achieve the tracked output. As seen in the snapshot below, all the components collectively work together to give an output which is similar to the expected or desired output in the base paper.

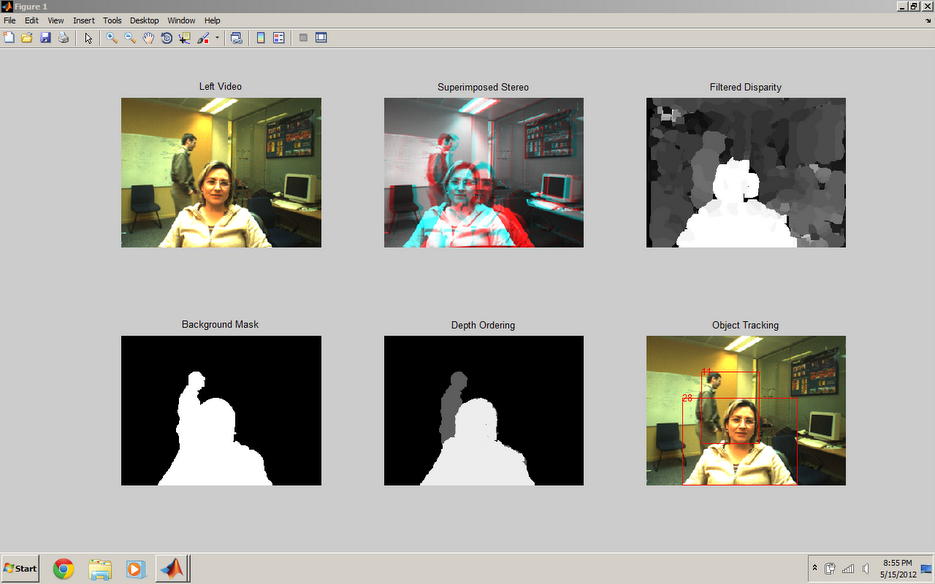


Figure 6.4 Integration testing output

**CHAPTER 7**

**EXPERIMENTAL RESULTS**

The results obtained by the Object Tracking Software are represented as snapshots which are given below. The object tracking software was tested with a video sequence consisting of multiple objects in the presence of occlusion. As seen in the subsequent snapshots, the video sequence initially consists of a single object. Multiple objects which introduce occlusion are later introduced into the scene.

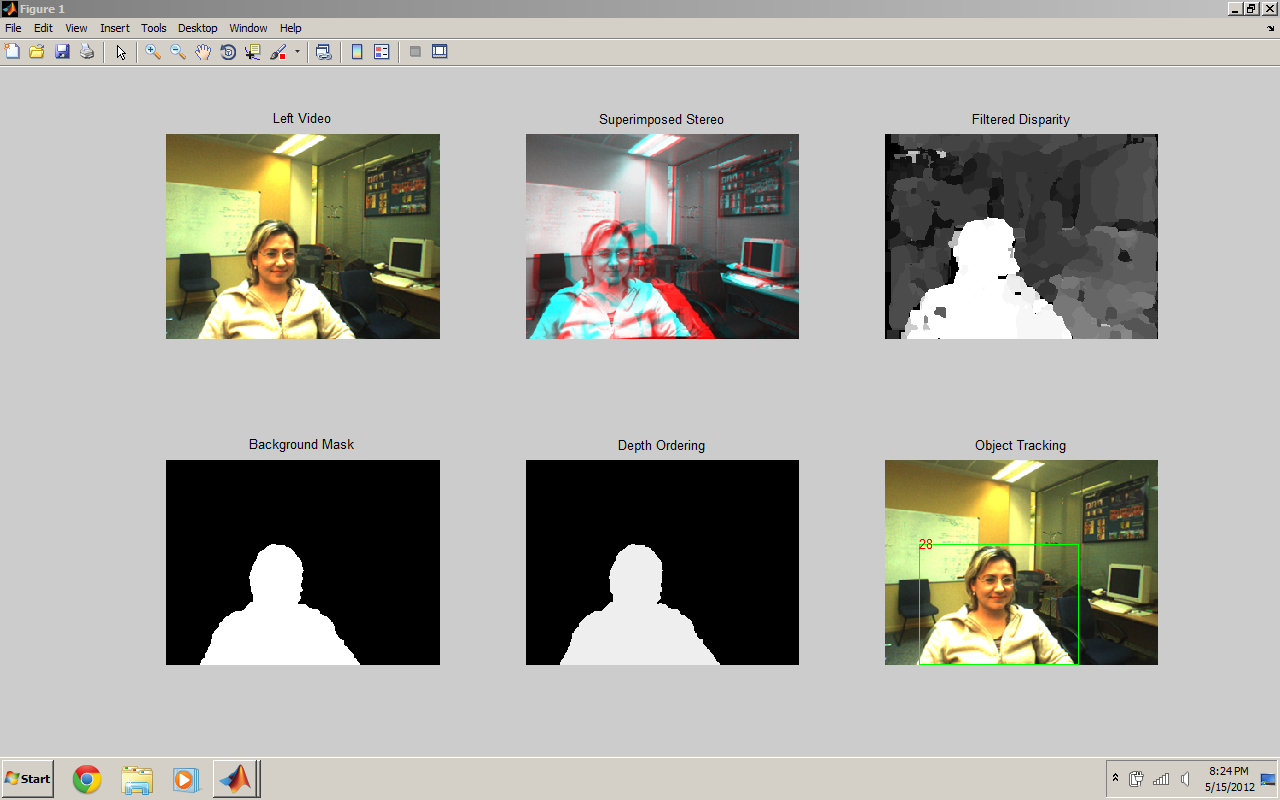
****

Figure 7.1 Single object tracked output

As seen in the above snapshot,

* The first box in the output window indicates the Left Video of the Stereo.
* The Second box gives the superimposed stereo result which is obtained by the two HD Web Cams.
* The Third box gives us the filtered disparity result.

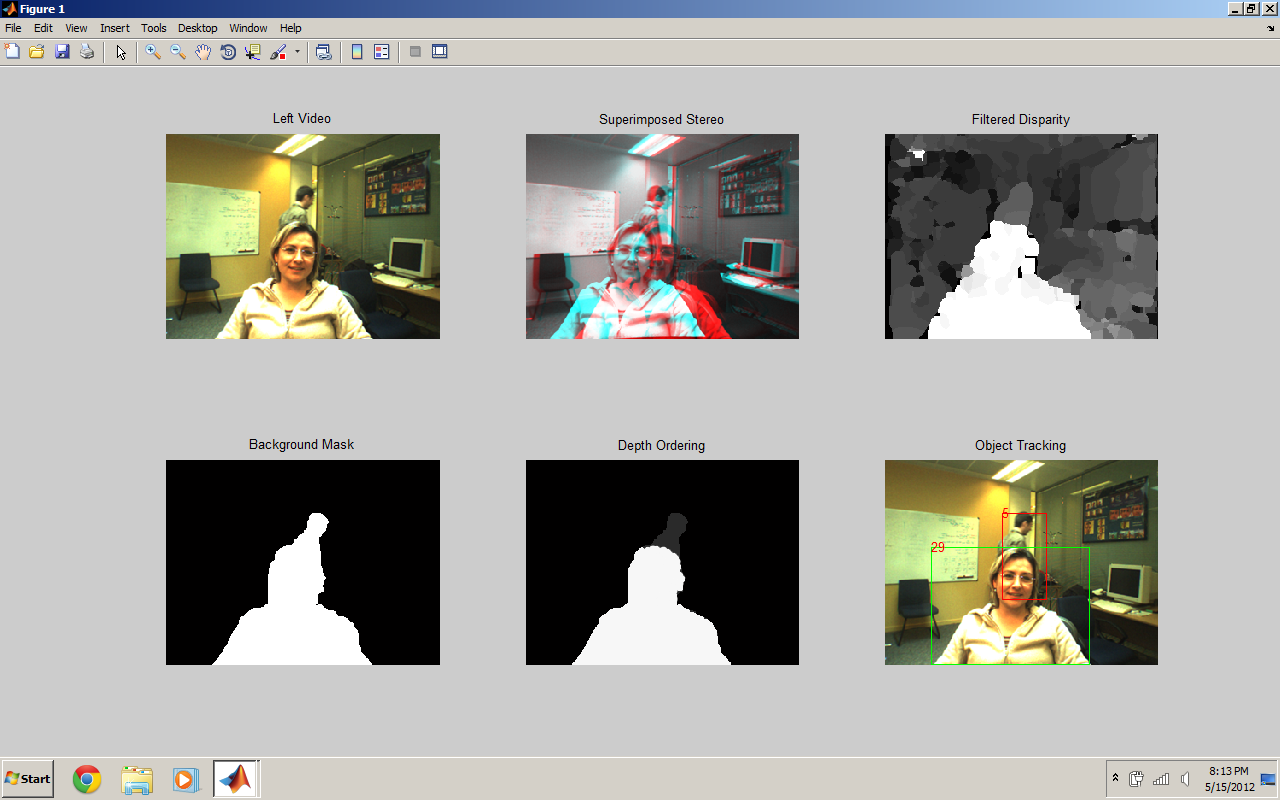
****

Figure 7.2 Multiple objects tracked output

From the above snapshot, we see that

* The fourth box in the window indicates how the Foreground objects are separated from the Background. This is done by simple subtraction using the concept of Moving Mask.
* Coloring the objects based on their Depth (Depth Ordering) is then performed which is indicated by the fifth box.
* The Final Box indicates the Tracked objects in the video sequence. The objects are surrounded by rectangular colored boxes as shown in the snapshot above.

****

Figure 7.3 Web User Interface home page

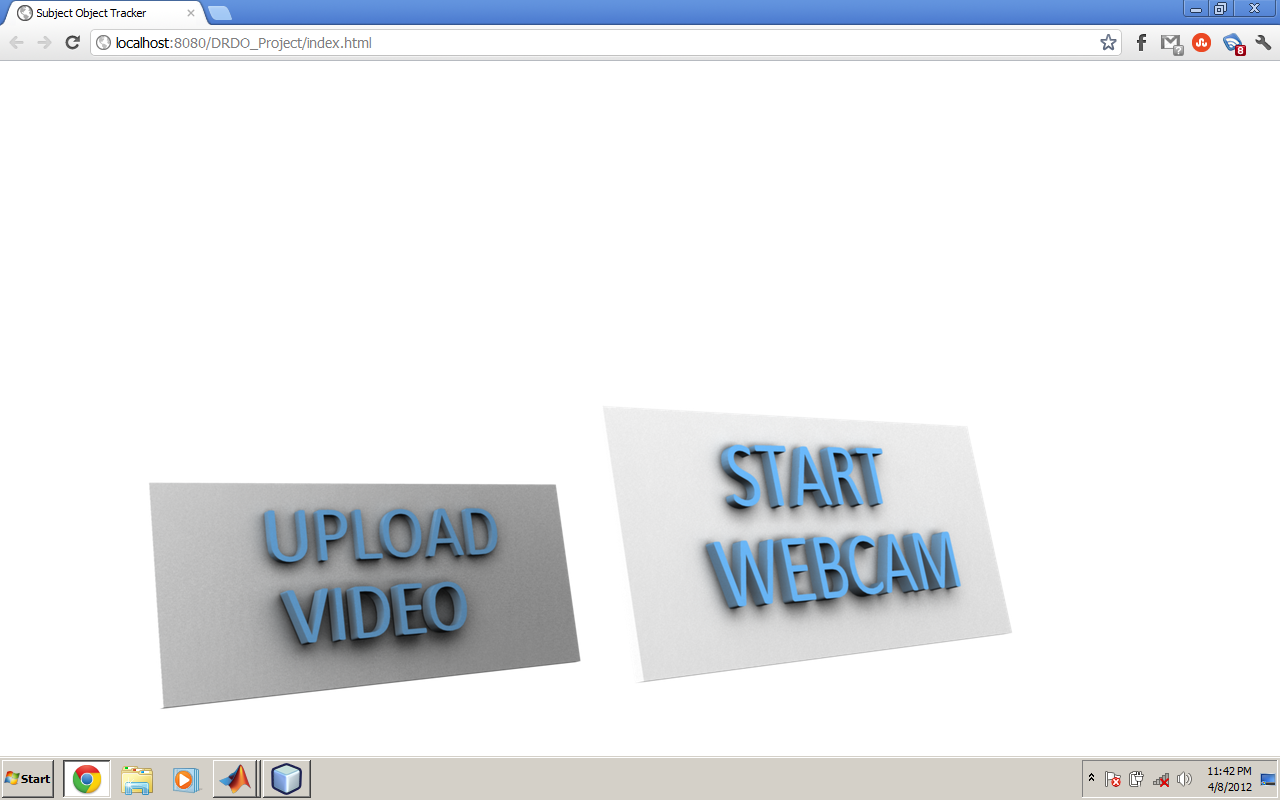
****

Figure 7.4 Web User Interface options page

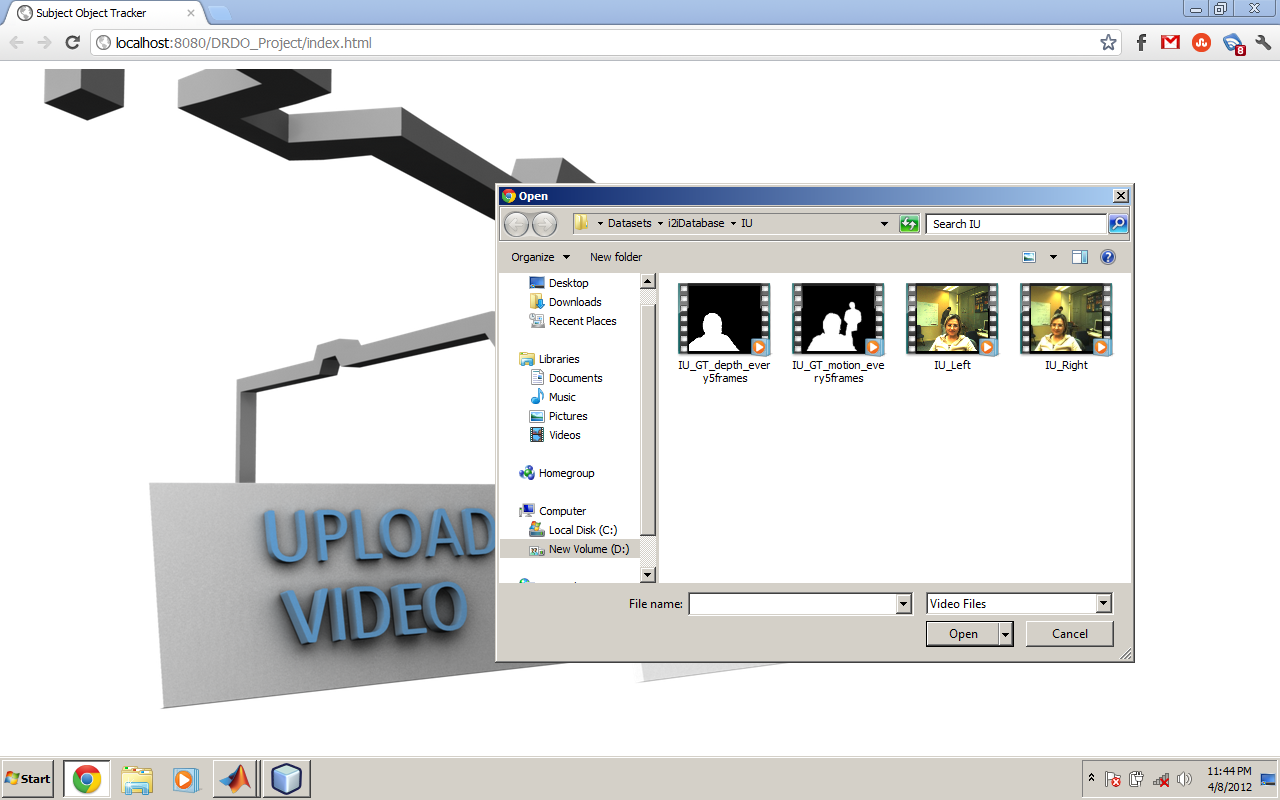
****

Figure 7.5 Web User Interface options upload box

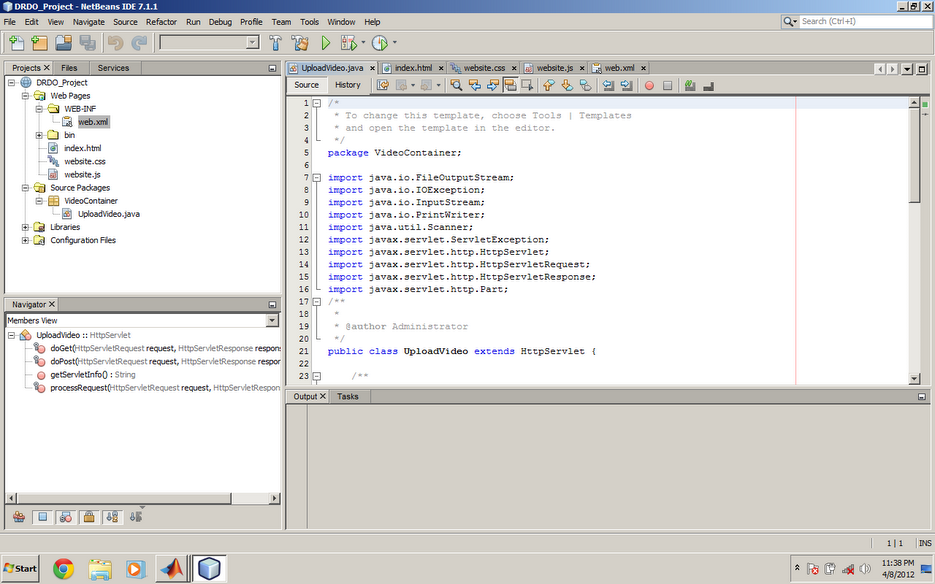
****

Figure 7.6 Project web code in NetBeans IDE

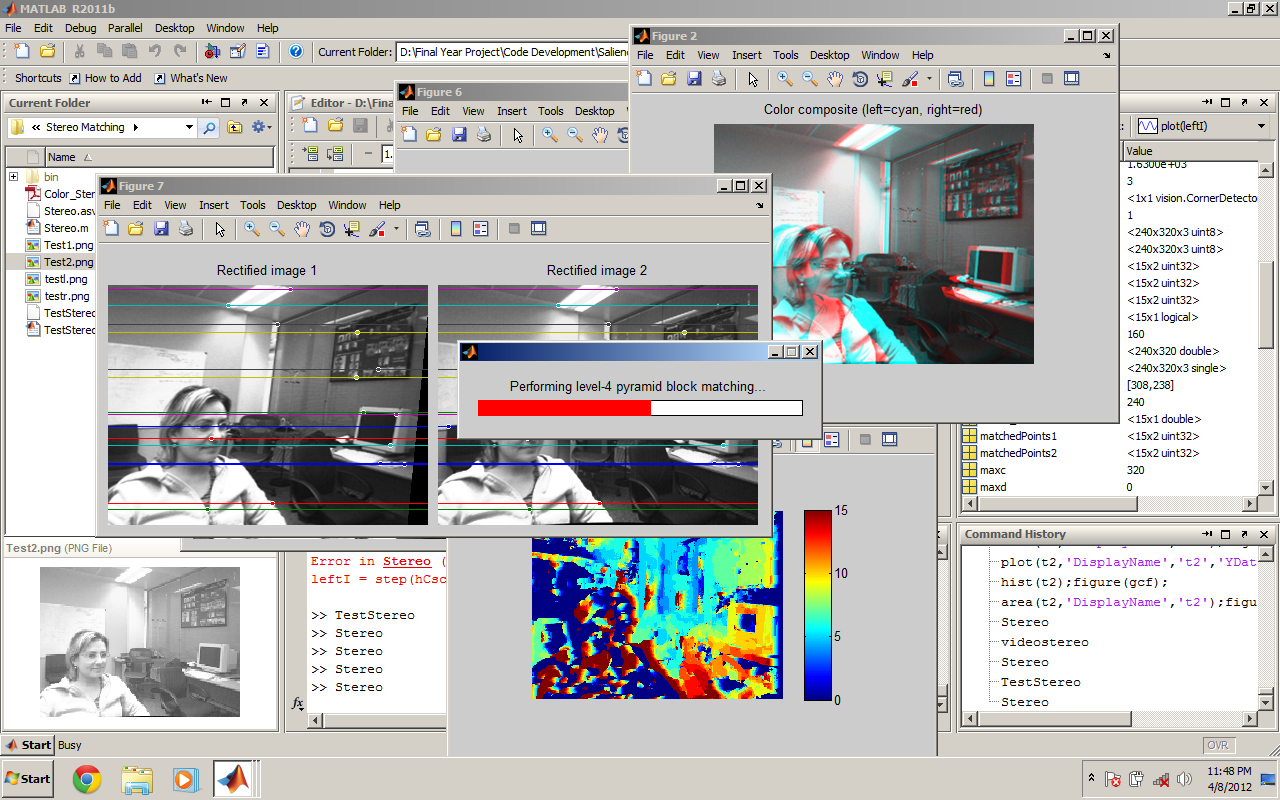
****

Figure 7.7 MATLAB environment

**CONCLUSION AND FUTURE WORK**

We have used a novel multiview(stereo) approach, which segments a group of people into individual human object and track them across the video sequence with high accuracy. Segmentation of individual objects is realized using the depth, occlusion, color, and motion. We are able to verify the tracked result by comparing it with the expected result on several sets of videos.

One challenge in tracking is to develop algorithms for tracking objects in unconstrained videos, for example, videos obtained from broadcast news networks or home videos. These videos are noisy, compressed, unstructured, and typically contain edited clips acquired by moving cameras from multiple views. Another related video domain is of formal and informal meetings. These videos usually contain multiple people in a small field of view. Thus, there is severe occlusion, and people are only partially visible.

One interesting solution in this context is to employ audio in addition to video for object tracking. There are some methods being developed for estimating the point of location of audio source , for example, a person’s mouth, based on four or six microphones. This audio-based localization of the speaker provides additional information which then can be used in conjunction with a video-based tracker to solve problems like severe occlusion.

The use of a particular feature set for tracking can also greatly affect the performance. Generally, the features that best discriminate between multiple objects and, between the object and background are also best for tracking the object. Many tracking algorithms use a weighted combination of multiple features assuming that a combination of preselected features will be discriminative. A wide range of feature selection algorithms have been investigated in the machine learning and pattern recognition communities. Such information is not always available. Moreover, as the object appearance or background varies, the discriminative features also vary.

Tracking and associated problems of feature selection, object representation, dynamic shape, andMotion estimation are very active areas of research and new solutions are continuously being proposed. The use of a particular feature set for tracking can also greatly affect the performance.

Generally, the features that best discriminate between multiple objects and, between the object and background are also best for tracking the object. Many tracking algorithms use a weighted combination of multiple features assuming that a combination of preselected features will be discriminative. Another challenge would be to optimize the tracking process by producing an efficient saliency map.

**REFERENCES**

[1] Qian Zhang, Student Member, IEEE, and King Ngi Ngan, Fellow, IEEE “Segmentation and Tracking Multiple Objects Under Occlusion From Multiview Video**”** IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 20, NO. 11, NOVEMBER 2011.

[2] Q. Zhang and K. N. Ngan, “Multi-view video segmentation using graph cut and spatiotemporal projections,” J. Vis. Commun. Image Represent, vol. 21, no. 5/6, pp. 453–461, Jul./Aug. 2010.

[3] J. Black, T. Ellis, and P. Rosin.”Multiview image surveillance and tracking”. In Motion&Video Computing Workshop, 2002.

[4] J. Pan, B. Hu, and J.Q. Zhang, “An Efficient Object Tracking Algorithm with Adaptive Prediction of Initial Searching Point,” Lecture Notes in Computer Science, vol. 4319/2006, pp. 1113-1122, 2006.

[5] J. Berclaz, F. Fleuret, and P. Fua. “Robust people tracking with global trajectory optimization”. In CVPR, 2006.

[6] Jiyan Pan, Bo Hu, “Robust Occlusion Handling in Object Tracking”, Computer Vision and Pattern Recognition, CVPR '07, JUNE 2007.

[7] H. Jiang, S. Fels, and J. J. Little. “A linear programming approach for multiple object tracking”. In CVPR, 2007.

[8] Jiyan PAN, Bo HU, Member, IEEE, and Jian Qiu ZHANG, Senior Member, IEEE.” Robust and Accurate Object Tracking under Various Types of Occlusions”IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY 2008

[9] A. Ess, B. Leibe, K. Schindler, and L. Van Gool.” Robust multi-person tracking from a mobile platform”. IEEE TPAMI,31(10):1831–1846, 2009.

[10] S. M. Khan and M. Shah, “Tracking multiple occluding people by localizing on multiple scene planes,” IEEE Trans. Pattern Anal. Mach Intell, vol. 31, no. 3, pp. 505–519, Mar. 2009.

[11] M. Eichner and V. Ferrari. “ Joint pose estimation of multiple persons”. In ECCV, September 2010.

[12] Wenhan Luo; Xiaoqin Zhang; Yang Liu; Xi Li; Weiming H; Wei Li. “Efficient Block Division Model For Robust Multiple Object Tracking”. Acoustic, speech and signal processing(ICASSP) 2011

**Source Code**

**acquire\_video.m**

function [vid1, vid2, vid3, L, nr, nc, BG1] = acquire\_video()

vid1 = mmreader(‘IU\_Left.avi');

vid2 = mmreader('IU\_Right.avi');

vid3 = mmreader('IU\_GT\_motion\_every5frames.avi');

L = get(vid1, 'NumberOfFrames');

im=read(vid1,1);

[nr nc nm] = size(im);

frames = zeros(nr,nc,nm,L);

for ii = 1:L

ii

im=read(vid1,ii);

frames(:,:,:,ii) = im;

end

BG1 = update\_background(frames);

end

**cluster.m**

function [D level]= cluster(D1,bw1)

D1(~bw1)=31;

[a b] = hist(D1(:),1:2:31);

D1(~bw1)=0;

a(end) = 0;

a(a<max(a)\*0.05)=0;

[L count] = bwlabel(a);

count

if count>0

D = zeros(size(D1,1),size(D1,2),count);

for ii = 1:count

v(ii) = mean(b(L ==ii));

D(:,:,ii) = abs(D1-v(ii));

end

[val ix] = min(D,[],3);

D = v(ix).\*bw1;

level = v;

else

D1(bw1) = mean2(D1(bw1));

D = D1;

level = mean2(D1(bw1));

end

end

**main.m**

Function [vid1, vid2, vid3, L, nr, nc, BG1] = acquire\_video();

maxs = 30;

for ii = 5:5:L-5

ii

i1=read(vid1,ii);

i2=read(vid2,ii);

subplot(231)

imshow(i1);

title('Left Video');

axis image;

I3 = superimposed\_stereo(i1,i2);

subplot(232)

imshow(I3);

title('Superimposed Stereo');

axis image;

bw1 = mask\_background(BG1,vid3,i1,ii);

subplot(234)

imshow(bw1);

title('Background Mask');

axis image;

[d] = stereo(i1,i2, maxs);

subplot(233)

imshow(d,[0 maxs]);

title('Filtered Disparity');

axis image;

subplot(236)

imshow(i1);

title('Object Tracking');

axis image;

Dim = track\_objects(d,bw1,nr,nc);

subplot(235)

imshow(Dim,[0 maxs]);

title('Depth Ordering');

axis image;

pause(0.0000001)

end

**superimposed\_stereo.m**

function I3 = superimposed\_stereo(i1,i2)

I1 = rgb2gray(i1);

I2 = rgb2gray(i2);

I3 = cat(3,I1,I2,I2);

end

mask\_background.m

function bw1 = mask\_background(BG1,vid3,i1,ii)

Step = 5;

bw1 = moving\_mask(i1,BG1);

bw1 = imfill(bw1,'holes');

bw1 = bwareaopen(bw1,400);

if mod(ii,5) ==0

bw3 = read(vid3,1+floor(ii/Step));

bw1 = im2bw(bw3,0.5);

end

end

**moving\_mask.m**

function bwcl = moving\_mask(im,BG)

s = [1 2 1; 0 0 0; -1 -2 -1];

Rbg = BG(:,:,1);

Gbg = BG(:,:,2);

Bbg = BG(:,:,3);

s = [1 2 1; 0 0 0; -1 -2 -1];

HRdens = conv2(Rbg,s);

VRdens = conv2(Rbg,s');

HGdens = conv2(Gbg,s);

VGdens = conv2(Gbg,s');

HBdens = conv2(Bbg,s);

VBdens = conv2(Bbg,s');

DensRf = (HRdens+VRdens)./9;

DensGf = (HGdens+VGdens)./9;

DensBf = (HBdens+VBdens)./9;

Densbg = cat(3,DensRf,DensGf,DensBf);

im = double(im);

diff = abs(BG-im);

diff = max(diff,[],3);

bw = (diff>30);

R1 = im(:,:,1);

G1 = im(:,:,2);

B1 = im(:,:,3);

HRdens1 = conv2(R1,s);

VRdens1 = conv2(R1,s');

HGdens1 = conv2(G1,s);

VGdens1 = conv2(G1,s');

HBdens1 = conv2(B1,s);

VBdens1 = conv2(B1,s');

DensRf1 = (HRdens1+VRdens1)./9;

DensGf1 = (HGdens1+VGdens1)./9;

DensBf1 = (HBdens1+VBdens1)./9;

Dens1 = cat(3,DensRf1,DensGf1,DensBf1);

diff1 = abs(Dens1-Densbg);

diff1 = max(diff1,[],3);

bw1 = (diff1>30);

bw1 = bw1(2:end-1,2:end-1);

BW = or(bw1,bw);

bwcl = bwmorph(BW,'close');

bwcl = bwareaopen(bwcl,150);

se = strel('line',5,90);

bwcl = imdilate(bwcl,se);

end

**stereo.m**

function [fdsp dsp] = stereo(i1,i2, maxs)

win\_size = 7;

tolerance = 2;

weight = 5;

[dsp1, diff1] = slide\_images(i1,i2, 1, maxs, win\_size, weight);

[dsp2, diff2] = slide\_images(i2,i1, -1, -maxs, win\_size, weight);

dsp = winner\_take\_all(dsp1,diff1,dsp2,diff2,tolerance);

fdsp = modefilt2(dsp,[win\_size,win\_size],2);

end

**slide\_images.m**

function [disparity mindiff] = slide\_images(i1,i2,mins,maxs,win\_size,weight)

[dimy,dimx,c] = size(i1);

disparity = zeros(dimy,dimx);

mindiff = inf(dimy,dimx);

h = ones(win\_size)/win\_size.^2;

[g1x g1y g1z] = gradient(double(i1));

[g2x g2y g2z] = gradient(double(i2));

step = sign(maxs-mins);

for(i=mins:step:maxs)

s = shift\_image(i2,i);

sx = shift\_image(g2x,i);

sy = shift\_image(g2y,i);

sz = shift\_image(g2z,i);

diffs = sum(abs(i1-s),3);

gdiffx = sum(abs(g1x-sx),3);

gdiffy = sum(abs(g1y-sy),3);

gdiffz = sum(abs(g1z-sz),3);

gdiff = gdiffx+gdiffy+gdiffz;

CSAD = imfilter(diffs,h);

CGRAD = imfilter(gdiff,h);

d = CSAD+weight\*CGRAD;

idx = find(d<mindiff);

disparity(idx) = abs(i);

mindiff(idx) = d(idx);

end

end

**shift\_image.m**

function I = shift\_image(I,shift)

dimx = size(I,2);

if(shift > 0)

I(:,shift:dimx,:) = I(:,1:dimx-shift+1,:);

I(:,1:shift-1,:) = 0;

else

if(shift<0)

I(:,1:dimx+shift+1,:) = I(:,-shift:dimx,:);

I(:,dimx+shift+1:dimx,:) = 0;

end

end

end

**modefilt2.m**

function f = modefilt2(img,win,ignore)

if(~exist('ignore'))

ignore = 0;

end

if(~exist('win'))

win = [5 5];

end

if(~isfloat(img))

img = double(img);

end

img = abs(round(img));

f = modefilt2\_mex(img,win,ignore);

end

**track\_objects.m**

function Dim = track\_objects(d,bw1,nr,nc)

[L count] = bwlabel(bw1);

Dim = zeros(nr,nc);

idx = 1;

colors = {'r','g','b','y','c','m','k'};

obid = [];

obL = [];

for nn = 1:count

ob = L ==nn;

[x y] = find(ob);

[Dim levels] = cluster(d,bw1);

for kk = 1:length(levels)

if ~isempty(obid)

match = 0;

for oo = 1:length(obid)

diff = abs(levels(kk)-obid(oo));

if diff<=4

objectid = obL(oo);

match = 1;

obidtemp(kk) = obid(oo);

obLtemp(kk) = objectid;

break;

end

end

if match~=1

obid = [obid levels(kk)];

for ii1 = 1:7

if ~ismember(ii1,obL)

obL = [obL ii1];

break;

end

end

objectid = obL(end);

obidtemp(kk) = levels(kk);

obLtemp(kk) = objectid;

end

else

obid = levels(kk);

objectid = 1;

obidtemp(kk) = levels(kk);

obLtemp(kk) = objectid;

end

ob1 = Dim ==levels(kk);

ob1 = bwareaopen(ob1,200);

[x1 y1] = find(ob1);

co = [min(y1) min(x1) max(y1)-min(y1) max(x1)-min(x1)];

rectangle('position',co,'edgecolor',colors{objectid})

text(co(1), co(2),num2str(levels(kk)),'color','r')

idx = idx+1;

end

obid = obidtemp;

obL = obLtemp;

end

end

**update\_background.m**

function BG = update\_background(frames)

[nr,nc,nm,nf] = size(frames);

frames = double(frames);

BG = zeros(nr,nc,nm);

tic

R1 = frames(:,:,1,:);

G1 = frames(:,:,2,:);

B1 = frames(:,:,3,:);

BG(:,:,1) = median(R1,4);

BG(:,:,2) = median(G1,4);

BG(:,:,3) = median(B1,4);

end

**winner\_take\_all.m**

function pd = winner\_take\_all(d1,m1,d2,m2,tolerance,maxs)

pixel\_dsp = zeros(size(d1));

idx1 = find(abs(d1-d2)<tolerance & m1<m2);

idx2 = find(abs(d1-d2)<tolerance & m2<m1);

pixel\_dsp(idx1) = d1(idx1);

pixel\_dsp(idx2) = d2(idx2);

pd = shift\_image(pixel\_dsp,5);

end

**FUNCTIONS**

* imread

A = imread(FILENAME) reads a grayscale or color image from the file specified by the string FILENAME. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname. The return value A is an array containing the image data. If the file contains a grayscale image, A is an M-by-N array. If the file contains a truecolor image, A is an M-by-N-by-3 array.

* subplot

H = subplot(m,n,p), or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle. The axes are counted along the top row of the Figure window, then the second row, etc.

* disp

disp(X) displays an array, without printing the array name. If X contains a text string, the string is displayed.

Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable.

* imshow

imshow(I) displays the image I which can be either RGB, gray or binary image.

* find

ind = find(X) locates all nonzero elements of array X, and returns the [linear indices](jar:file:///C:/Program%20Files/MATLAB/R2010a/help/techdoc/help.jar%21/math/f1-85462.html#f1-85511) of those elements in vector ind. If X is a row vector, then ind is a row vector; otherwise, ind is a column vector. If X contains no nonzero elements or is an empty array, then ind is an empty array.

* mmreader

Used with the read method to read video data from a multimedia file into the MATLAB workspace.

* read

Read video data from file. i.e [video](jar:file:///C:/Program%20Files/MATLAB/R2011b/help/techdoc/help.jar%21/ref/videoreader.read.html#outputarg_video) = read([obj](jar:file:///C:/Program%20Files/MATLAB/R2011b/help/techdoc/help.jar%21/ref/videoreader.read.html#inputarg_obj)) reads in all video frames from the file associated with obj.

* size

Deals with array dimensions. i.e d = size(X) returns the sizes of each dimension of array X in a vector d with ndims(X) elements. If X is a scalar, which MATLAB software regards as a 1-by-1 array, size(X) returns the vector [1 1].

* Zeros

Creates array of all zeros. i.e B = zeros(n) returns an n-by-n matrix of zeros. An error message appears if n is not a scalar.

* Title

Used to add title to the current axes. i.e title('string') outputs the string at the top and in the center of the current axes.