

**An Industry Oriented Mini Project Report  
on**

**Real-time active visual  
tracking system**

*Submitted to the*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
HYDERABAD**

*In partial fulfillment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

*BY*

**PATI AISHWARYA SRI - 17WJ1A05N0**

**Under the Esteemed Guidance of**

**Dr. S. MADHU**

**Professor, CSE Dept.**



**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS  
(AUTONOMOUS)**

**School of Engineering and Technology**

**Ibrahimpattanam R.R. District 501506.**

**2020-21**



# GURU NANAK INSTITUTIONS TECHNICAL CAMPUS



Approved by  
AICTE - New Delhi



Affiliated to  
JNTU - Hyderabad



Accredited by  
National Assessment and  
Accreditation Council



Accredited by  
National Board of  
Accreditation

AUTONOMOUS  
under Section 2 (f) & 12 (b) of  
University Grants Commission Act

---

## Department of Computer Science and Engineering

### CERTIFICATE

This is to certify that this project report entitled “**Real-time active visual tracking system**” by PATI AISHWARYA SRI(17WJ1A05N0) submitted in partial fulfillment of the requirements for the degree of **Bachelor of Technology in Computer Science and Engineering** of the **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2020-21, is a bonafide record of work carried out under our guidance and supervision.

---

**INTERNAL GUIDE**  
**Dr. S. Madhu**

---

**PROJECT CO-ORDINATOR**  
**Mrs. V. Swathi**

---

**HOD CSE**  
**Prof. V. Devasekhar**

---

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I wish to express our sincere thanks to **Dr. Rishi Sayal**, Associate Director, GNITC for providing us the conducive environment for carrying through our academic schedule and project with ease.

I have been truly blessed to have a wonderful adviser **Prof. V.Devasekhar** Associate Professor & HEAD, Dept of CSE, GNITC for guiding us to explore the ramification of our work and we express my sincere gratitude towards him for leading me through the completion of project.

I thank Project Coordinator **Mrs.V.Swathi**, Assistant Professor, CSE, GNITC for providing seamless support and right suggestions that are given in the development of the project.

I especially thank our internal guide **Dr. S. Madhu**, professor ,CSE, GNITC for his constant guidance in every stage of the project. We would also like to thank all our lecturers for helping us in every possible way whenever the need arose.

On a more personal note we thank our beloved parents and friends for their moral support during the course of our project.

**PATI AISHWARYA SRI - 17WJ1A05N0**

## **Real-time active visual tracking system**

# TABLE OF CONTENTS

CONTENT	PAGE NO.
ABSTRACT.....	i
LIST OF FIGURES.....	ii
LIST OF SYMBOLS.....	iii-vi
CHAPTER 1 : INTRODUCTION.....	1-4
1.1 GENERAL	
1.2 OBJECTIVE	
1.3 EXISTING SYSTEM	
1.3.1 EXISTINGSYSTEM DISADVANTAGES	
1.3.2 LITERATURE SURVEY	
1.4 PROPOSED SYSTEM	
1.4.1 PROPOSED SYSTEM ADVANTAGES	
CHAPTER 2 :PROJECT DESCRIPTION.....	5-6
2.1 GENERAL	
2.2 METHODOLOGIES	
2.2.1 MODULES NAME	
2.2.2 MODULES EXPLANATION	
2.2.3 MODULE DIAGRAM	
2.2.4 GIVEN INPUTAND EXPECTED OUTPUT	
2.3 TECHNIQUE OR ALGORITHM	
CHAPTER 3 : REQUIREMENTS.....	7-8
3.1 GENERAL	
3.2 HARDWARE REQUIREMENTS	
3.3 SOFTWARE REQUIREMENTS	
CHAPTER 4 : SYSTEM DESIGN.....	9-21
4.1 GENERAL	
4.1.1 ACTIVITY DIAGRAM	
4.1.2 USE CASE DIAGRAM	
4.1.3 DATA FLOW DIAGRAM	

4.1.4	SEQUENCE DIAGRAM	
4.1.5	COLLABORATION DIAGRAM	
4.1.6	CLASS DIAGRAM	
4.1.7	SYSTEM ARCHITECTURE	
4.1.8	OBJECT DIAGRAM	
4.1.9	STATE DIAGRAM	
4.1.10	COMPONENT DIAGRAM	
4.1.11	E-R DIAGRAM	
4.2	DATABASE DESIGN (ALL LEVEL)	
<b>CHAPTER 5 :</b>	<b>SOFTWARE SPECIFICATION.....</b>	<b>22-24</b>
5.1	GENERAL	
<b>CHAPTER 6 :</b>	<b>IMPLEMENTATION.....</b>	<b>25-32</b>
6.1	GENERAL	
6.2	IMPLEMENTATION	
6.3	DATA BASE TABLE STRUCTURE	
<b>CHAPTER 7 :</b>	<b>SNAPSHOTS.....</b>	<b>33-34</b>
7.1	GENERAL	
7.2	VARIOUS SNAPSHOTS	
<b>CHAPTER 8 :</b>	<b>SOFTWARE TESTING.....</b>	<b>35-37</b>
8.1	GENERAL	
8.2	DEVELOPING METHODOLOGIES	
8.3	TYPES OF TESTING	
<b>CHAPTER 9:</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT.....</b>	<b>38</b>
10.1	CONCLUSION	
10.2	FUTURE ENCHANCEMENTS	
<b>REFERENCES.....</b>		<b>39</b>

# **ABSTRACT**

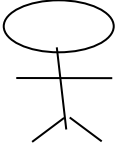
This project describes implementation of a real-time visual tracking system equipped with an active camera. The system is intended for indoor human motion tracking. Real-time tracking is achieved using simple and fast motion detection procedures based on frame differencing and camera motion compensation. Results of on-line person tracking are presented. Based on preliminary results of object detection in each image which may have missing and/or false detection, the multiple object tracking method keeps a graph structure where it maintains multiple hypothesis about the number and the trajectories of the object in the video. The image information drives the process of extending and pruning the graph, and determines the test hypothesis to explain the video. While the image-based object detection makes a local decision, the tracking process confirms and validates the detection through time, therefore, it can be regarded as temporal detection which makes a global decision across time. The multiple object tracking method gives feedbacks which are predictions of object locations to the object detection module. Therefore, the method integrates object detection and tracking tightly. The most possible hypothesis provides the multiple object tracking result. The experimental results are presented.

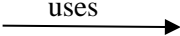
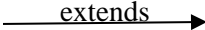

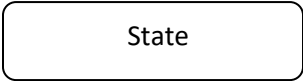
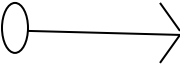
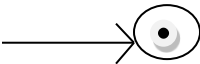
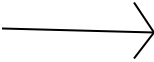
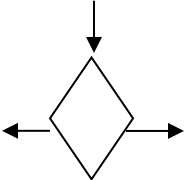
## LIST OF FIGURES

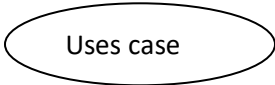
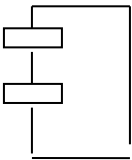
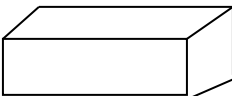
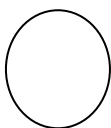


FIGURE NO	NAME OF THE FIGURE	PAGE NO.
4.1	Use case Diagram	9
4.2	Class Diagram	10
4.3	Object Diagram	11
4.4	Component diagram	12
4.5	Deployment diagram	13
4.6	Sequence diagram	14
4.7	Collaboration diagram	15
4.8	State Diagram	16
4.9	Activity Diagram	17
4.10	Data flow Diagram	18
4.11	Data flow Diagram	18
4.12	E-R diagram	19
4.13	System Architecure	20

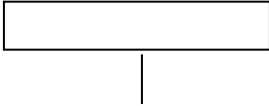
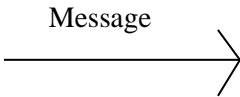


## LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 5px; width: 150px;"> <i>+ public</i>  <i>-private</i>  <i># protected</i> </div> <div style="border: 1px solid black; padding: 5px; width: 150px;"> <i>Class Name</i>  <hr/> <i>-attribute-</i>  <i>attribute</i>  <hr/> <i>+operation</i>  <i>+operation</i>  <i>+operation</i> </div> </div>	Represents a collection of similar entities grouped together.
2.	Association	<div style="display: flex; justify-content: center; align-items: center; margin-bottom: 10px;"> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px; margin: 0 5px;"></div> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class B</div> </div> <div style="display: flex; justify-content: center; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class A</div> <div style="border-bottom: 1px solid black; width: 50px; margin: 0 5px;"></div> <div style="border: 1px solid black; padding: 5px; margin: 0 10px;">Class B</div> </div>	Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Class A</div> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;">Class B</div> <div style="width: 10px; height: 10px; background: black; margin: 0 auto 5px auto; transform: rotate(45deg);"></div> </div> <div style="text-align: center;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">Class A</div> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;">Class B</div> <div style="width: 10px; height: 10px; background: black; margin: 0 auto 5px auto; transform: rotate(45deg);"></div> </div> </div>	Interaction between the system and external environment

5.	Relation (uses)		Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint

13.	Use case		Interact ion between the system and external environment.
14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or acion.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.

19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

# CHAPTER 1

## INTRODUCTION

### 1.1 GENERAL

The multiple object tracking method works on fixed cameras. It starts with an adaptive background modelling module which deals with changing illuminations and does not require objects to be constantly moving.

### 1.2 OBJECTIVE

Our multiple object tracking method is reliable to deal with occlusions, irregular object motions, changing appearances by postponing the decision of object trajectories until sufficient information is accumulated over time.

### 1.2 Existing System:

Multiple object tracking has been a challenging research topic in computer vision. It has to deal with the difficulties existing in single object tracking, such as changing appearances, non-rigid motion; dynamic illumination and occlusion, as well as the problems related to multiple object tracking including inter-object occlusion, multi-object concision. There has been much work on multiple object visual tracking. they use a sampling algorithm for tracking fixed number of objects. Tao et al. [2] present an efficient hierarchical algorithm to track multiple people.

### LITERATURE SURVEY:

**Title:** Towards Computational Baby Learning: A Weakly-Supervised Approach for Object Detection

**Author:** Xiaodan Liang; Si Liu ;Yunchao Wei ;Luoqi Liu.

**Year:** 2015

### Description:

Intuitive observations show that a baby may inherently possess the capability of recognizing a new visual concept (e.g., chair, dog) by learning from only very few positive instances taught by parent(s) or others, and this recognition capability can be gradually further

improved by exploring and/or interacting with the real instances in the physical world. Inspired by these observations, we propose a computational model for weakly-supervised object detection, based on prior knowledge modelling, exemplar learning and learning with video contexts. The prior knowledge is modeled with a pre-trained Convolutional Neural Network (CNN). When very few instances of a new concept are given, an initial concept detector is built by exemplar learning over the deep features the pre-trained CNN. The well-designed tracking solution is then used to discover more diverse instances from the massive online weakly labeled videos. Once a positive instance is detected/identified with high score in each video, more instances possibly from different view-angles and/or different distances are tracked and accumulated. Then the concept detector can be fine-tuned based on these new instances. This process can be repeated again and again till we obtain a very mature concept detector. Extensive experiments on Pascal VOC-07/10/12 object detection datasets [9] well demonstrate the effectiveness of our framework. It can beat the state-of-the-art full-training based performances by learning from very few samples for each object category, along with about 20,000 weakly labeled videos.

**Title:** Track and Transfer: Watching Videos to Simulate Strong Human Supervision for Weakly-Supervised Object Detection

**Author:** Krishna Kumar Singh, Fanyi Xiao.

**Year:** 2016

**Description:**

The status quo approach to training object detectors requires expensive bounding box annotations. Our framework takes a markedly different direction: we transfer tracked object boxes from weakly-labeled videos to weakly-labeled images to automatically generate pseudo ground-truth boxes, which replace manually annotated bounding boxes. We first mine discriminative regions in the weakly-labeled image collection that frequently/rarely appear in the positive/negative images. We then match those regions to videos and retrieve the corresponding tracked object boxes. Finally, we design a hough transform algorithm to vote for the best box to serve as the pseudo GT for each image, and use them to train an object detector. Together, these lead to state-of-the-art weakly-supervised detection results on the PASCAL 2007 and 2010 datasets.

**Title:** Watch and Learn: Semi-Supervised Learning of Object Detectors from Videos

**Author:** Ishan Misra, Abhinav Shrivastava, Martial Hebert

**Year:** 2015

**Description:**

I present a semi-supervised approach that localizes multiple unknown object instances in long videos. We start with a handful of labeled boxes and iteratively learn and label hundreds of thousands of object instances. We propose criteria for reliable object detection and tracking for constraining the semi-supervised learning process and minimizing semantic drift. Our approach does not assume exhaustive labeling of each object instance in any single frame, or any explicit annotation of negative data. Working in such a generic setting allow us to tackle multiple object instances in video, many of which are static. In contrast, existing approaches either do not consider multiple object instances per video, or rely heavily on the motion of the objects present. The experiments demonstrate the effectiveness of our approach by evaluating the automatically labeled data on a variety of metrics like quality, coverage (recall), diversity, and relevance to training an object detector.

**Title:** Weakly supervised localization of novel objects using appearance transfer

**Author:** Mrigank Rochan ; Yang Wang.

**Year:** 2015

**Description:**

I consider the problem of localizing unseen objects in weakly labeled image collections. Given a set of images annotated at the image level, our goal is to localize the object in each image. The novelty of our proposed work is that, in addition to building object appearance model from the weakly labeled data, we also make use of existing detectors of some other object classes (which we call “familiar objects”). We propose a method for transferring the appearance models of the familiar objects to the unseen object. Our experimental results on both image and video datasets demonstrate the effectiveness of our approach.

**Title:** Large Scale Semi-Supervised Object Detection Using Visual and Semantic Knowledge Transfer

**Author:** Yuxing Tang ; Josiah Wang ; Boyang Gao

**Year:** 2016

**Description:**

Deep CNN-based object detection systems have achieved remarkable success on several large-scale object detection benchmarks. However, training such detectors requires a large number of labeled bounding boxes, which are more difficult to obtain than image-level annotations. Previous work addresses this issue by transforming image-level classifiers into object detectors. This is done by modeling the differences between the two on categories with both image level and bounding box annotations, and transferring this information to convert classifiers to detectors for categories without bounding box annotations. We improve this previous work by incorporating knowledge about object similarities from visual and semantic domains during the transfer process. The intuition behind our proposed method is that visually and semantically similar categories should exhibit more common transferable properties than dissimilar categories, e.g. a better detector would result by transforming the differences between a dog classifier and a dog detector onto the cat class, than would by transforming from the violin class. Experimental results on the challenging ILSVRC2013 detection dataset demonstrate that each of our proposed object similarity based knowledge transfer methods outperforms the baseline methods. We found strong evidence that visual similarity and semantic relatedness are complementary for the task, and when combined notably improve detection, achieving state-of-the-art detection performance in a semi-supervised setting.

## **1.1 Proposed System**

The tracking algorithm accepts the probabilities of preliminary object detection and keeps multiple hypotheses of object trajectories in a graph structure, as shown in Figure 2. Each hypothesis consists of the number of objects and their trajectories. The first step in tracking is to extend the graph to include the most recent object detection results, that is, to generate multiple hypotheses about the trajectories.



## **CHAPTER 2**

### **PROJECT DESCRIPTION**

#### **2.1 GENERAL:**

In this paper, two seamlessly integrated solutions, selfpaced learning and multi-modal learning, are used to achieve high precision and recall during training sample generation.

#### **2.2 METHODOLOGIES**

##### **MODULE:**

1. INPUT VIDEO
2. PRE PROCESSING
4. SEGMENTATION
5. CLASSIFICATION

##### **➤ INPUT VIDEO:**

Firstly I take an video for detecting the objects whose presented on the video at the anytime. The multiple object tracking method works on fixed cameras. It starts with an adaptive background modelling module which deals with changing illuminations and does not require objects to be constantly moving.

##### **➤ PRE PROCESSING:**

Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightness) The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, and translation) are also classified among pre-processing methods. Since here similar techniques are used.

### ➤ **SEGMENTATION:**

I can divide or partition the image into various parts called segments. It's not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works.

### ➤ **CLASSIFICATION:**

As a result of the model executed during object detection. In this detection we can detect multiple objects. It is used to filter weak calculations with the Confidence value. Unwanted values were filtered out in this way.

## **2.3 TECHNIQUE USED OR ALGORITHM USED**

Our multiple object tracking method is reliable to deal with occlusions, irregular object motions, changing appearances by postponing the decision of object trajectories until sufficient information is accumulated over time. It makes a global decision. The most possible hypothesis generates the multiple objects tracking result. The trajectories provide information of object identifications, motion histories, timing and object interactions. The information can be applied to detect abnormal behaviors in video surveillance and collect traffic data in traffic control systems.

## **CHAPTER 3**

### **REQUIREMENTS ENGINEERING**

#### **3.1 GENERAL**

We provide a brief discussion on the relationship between FEOD and other types of supervisions, excluding the methods using strong labels. First, strictly speaking, FEOD is a semi-supervised task. But to the best of our knowledge, most works on semi-supervised object detection (SSOD) assume around 50% of all the labeled bounding boxes.

#### **3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should state what the system does and not how it should be implemented.

- PROCESSOR : DUAL CORE 2 DUOS.
- RAM : 4GB DD RAM
- HARD DISK : 250 GB

#### **3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

## **SOFTWARE REQUIREMENTS**

- Operating System : Windows 7/8/10
- Platform : Spyder3
- Programming Language : Python
- Front End : Spyder3

### **3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

### **3.5 NON-FUNCTIONAL REQUIREMENTS**

#### **EFFICIENCY**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

## CHAPTER 4

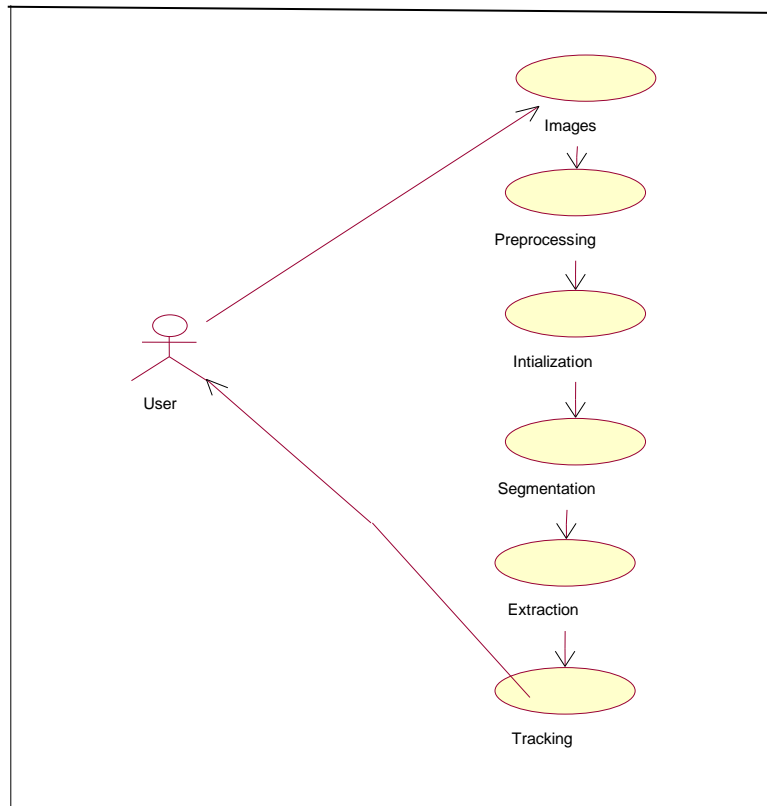
### DESIGN ENGINEERING

#### 4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

#### UML Diagrams

##### Use case diagram

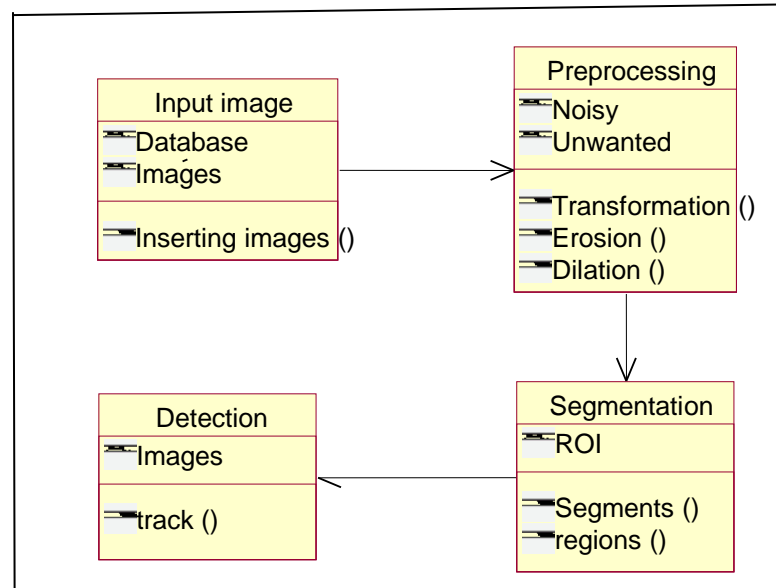


**fig-4.1: Use case diagram for Real-time active visual tracking system**

## EXPLANATION:

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. As shown in the fig-4.1 consists of user as actor. Each will play a certain role to achieve the concept.

## Class Diagram

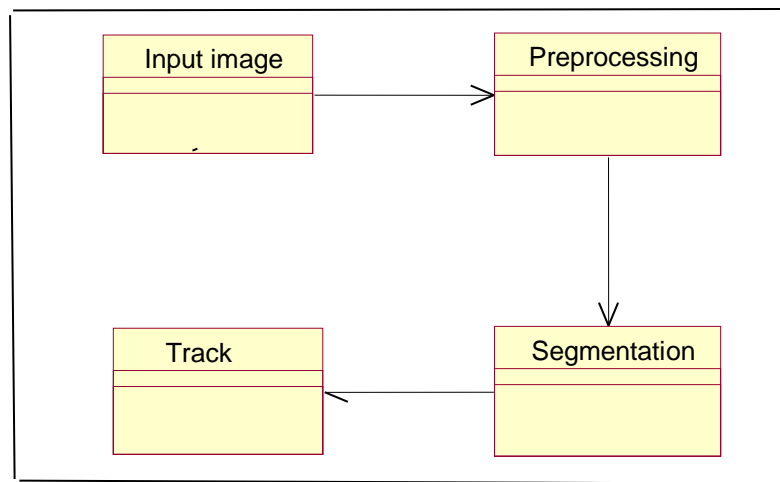


**Fig.4.2: Class diagram for Real-time active visual tracking system**

## EXPLANATION

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. As shown in the fig-4.2 the various classes involved in our project.

## Object Diagram

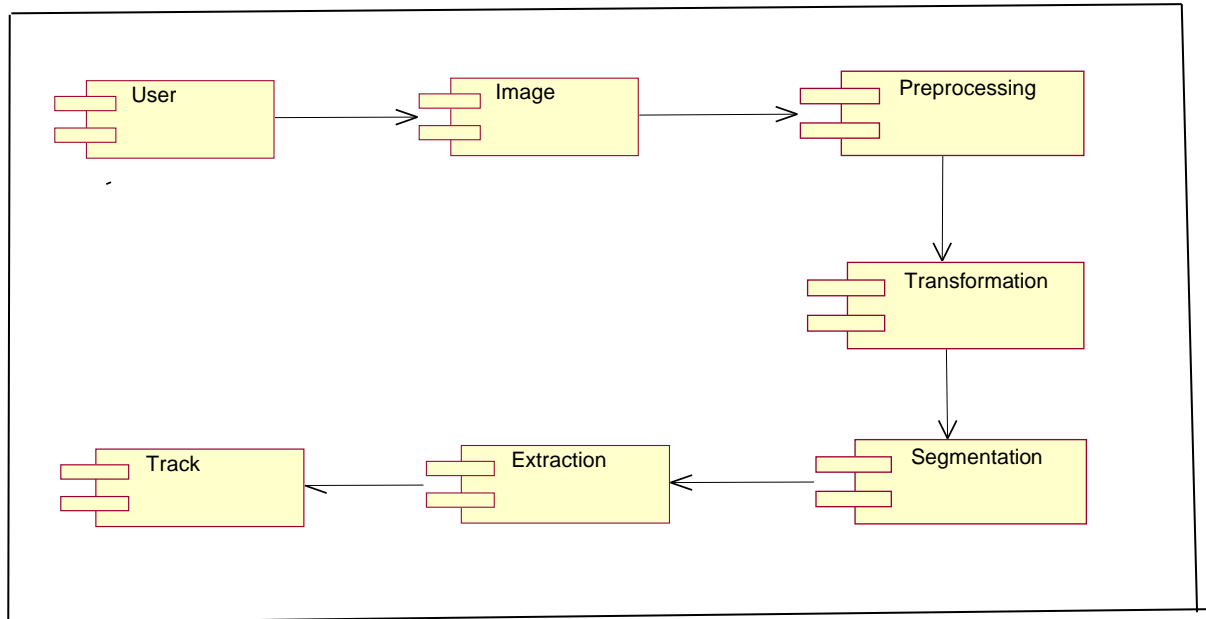


**Fig.4.3: Object diagram for Real-time active visual tracking system**

### EXPLANATION:

As shown in the fig-4.3 tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security. An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, namely to show examples of data structure.

## Component Diagram



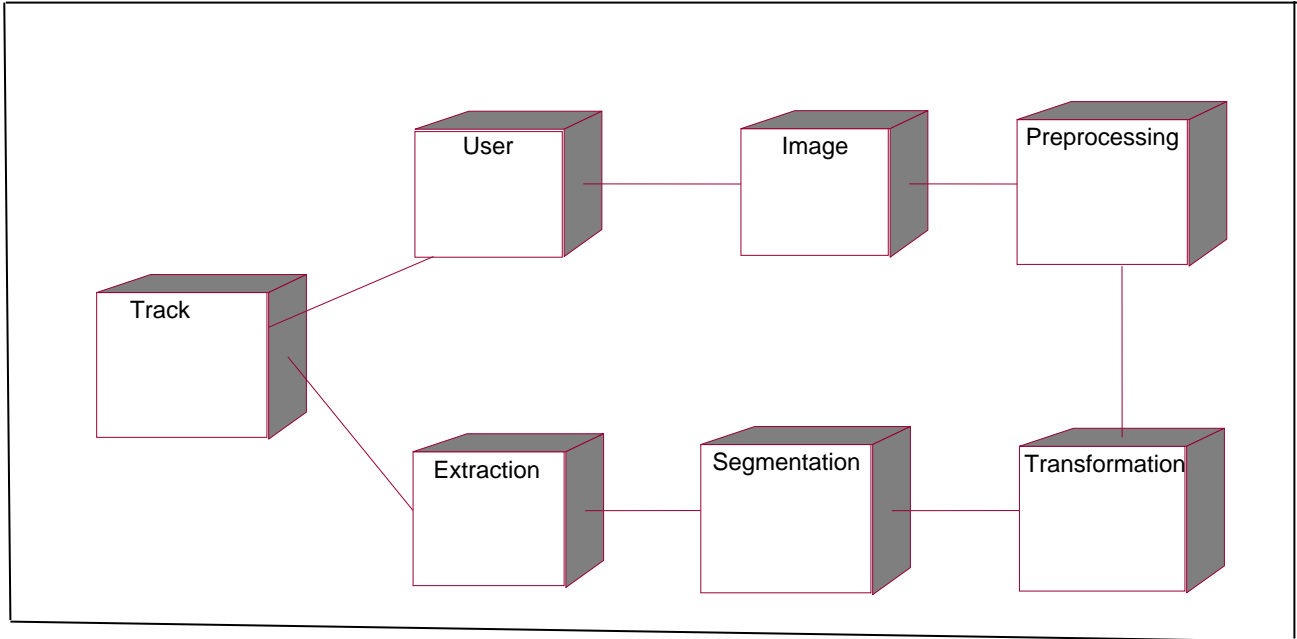
**Fig.4.4: Component diagram for Real-time active visual tracking system**

### EXPLANATION:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. A component is a replaceable and executable piece of a system. A component provides the set of required interfaces that a component realizes or implements. These are the static diagrams of the unified modeling language.



## Deployment Diagram

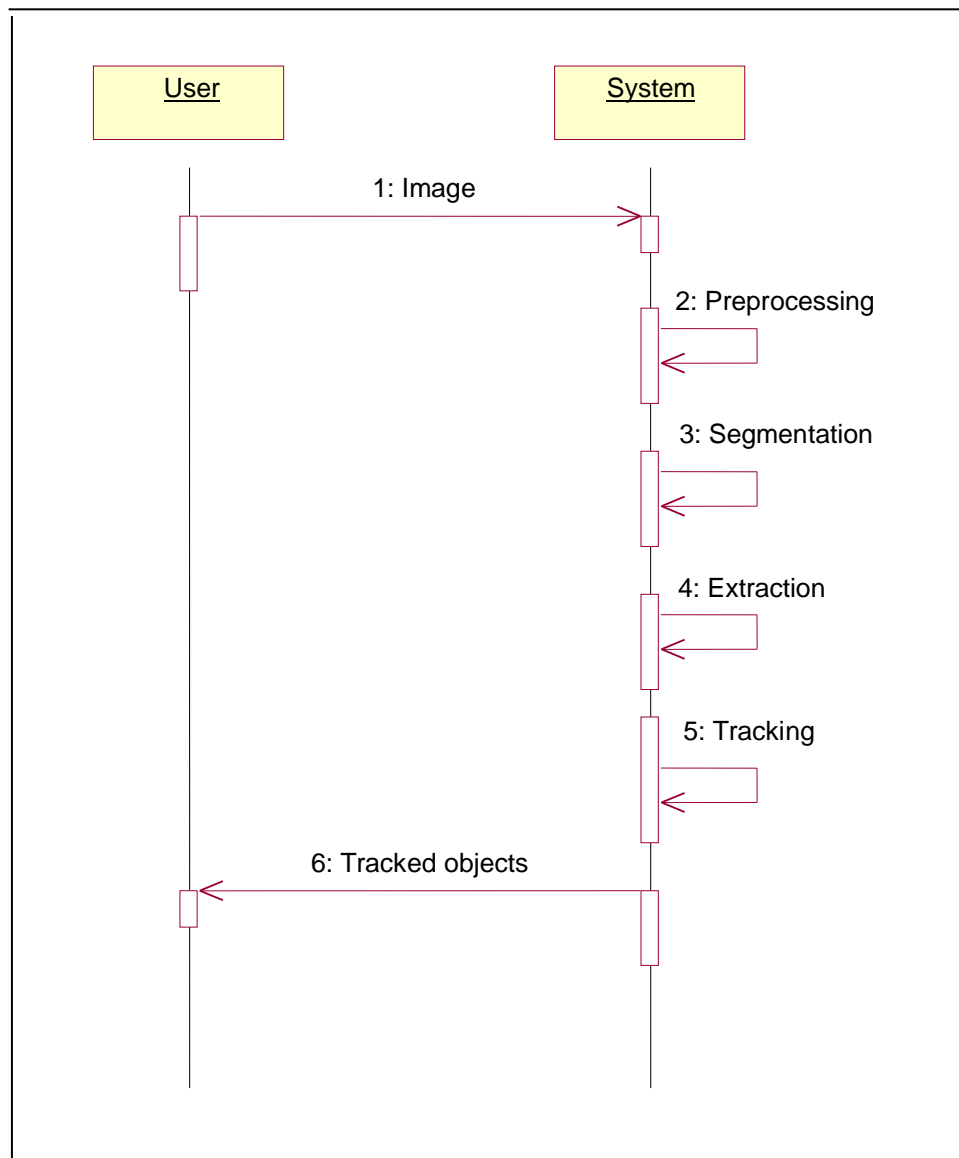


**Fig.4.5 :Deployment diagram for Real-time active visual tracking system**

### EXPLANATION:

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

## Sequence Diagram

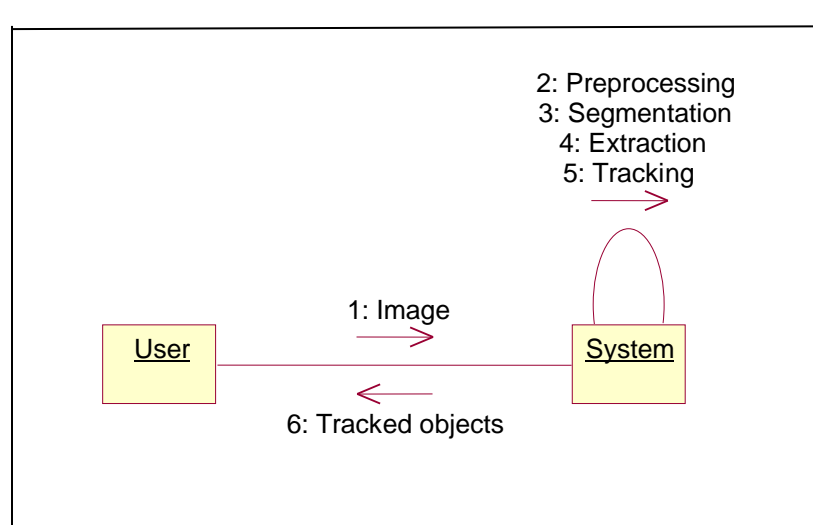


**Fig.4.6 :Sequence diagram for Real-time active visual tracking system**

### EXPLANATION:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

## Collaboration Diagram

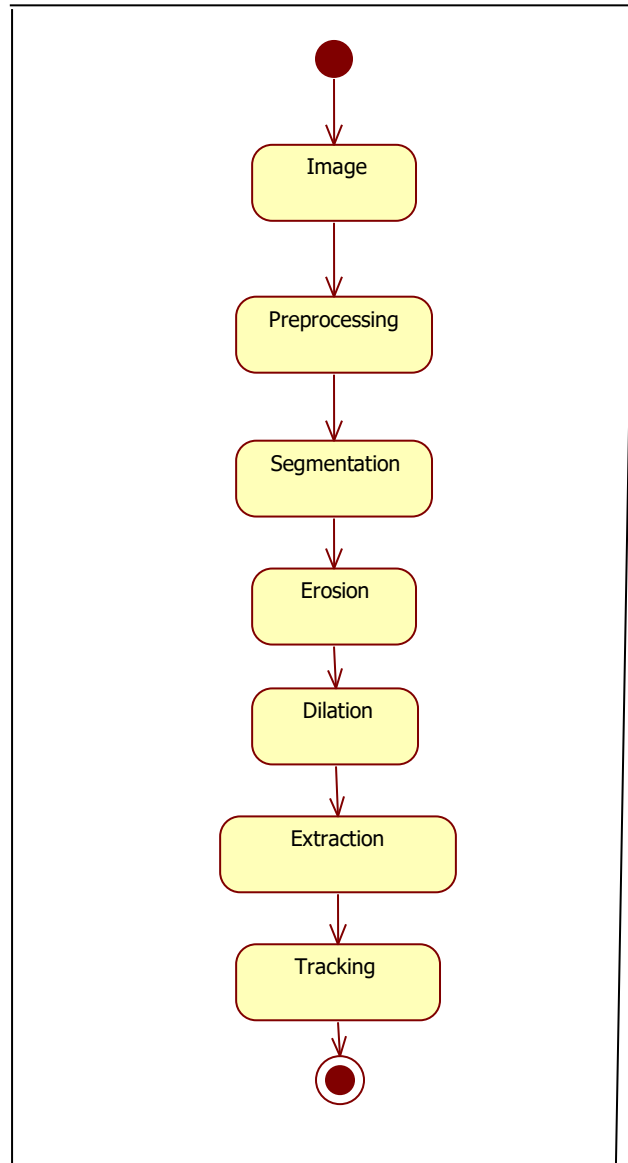


**Fig.4.7: Collaboration diagram for Real-time active visual tracking system**

### EXPLANATION:

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). As shown in the fig-4.7 these diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

## State Diagram



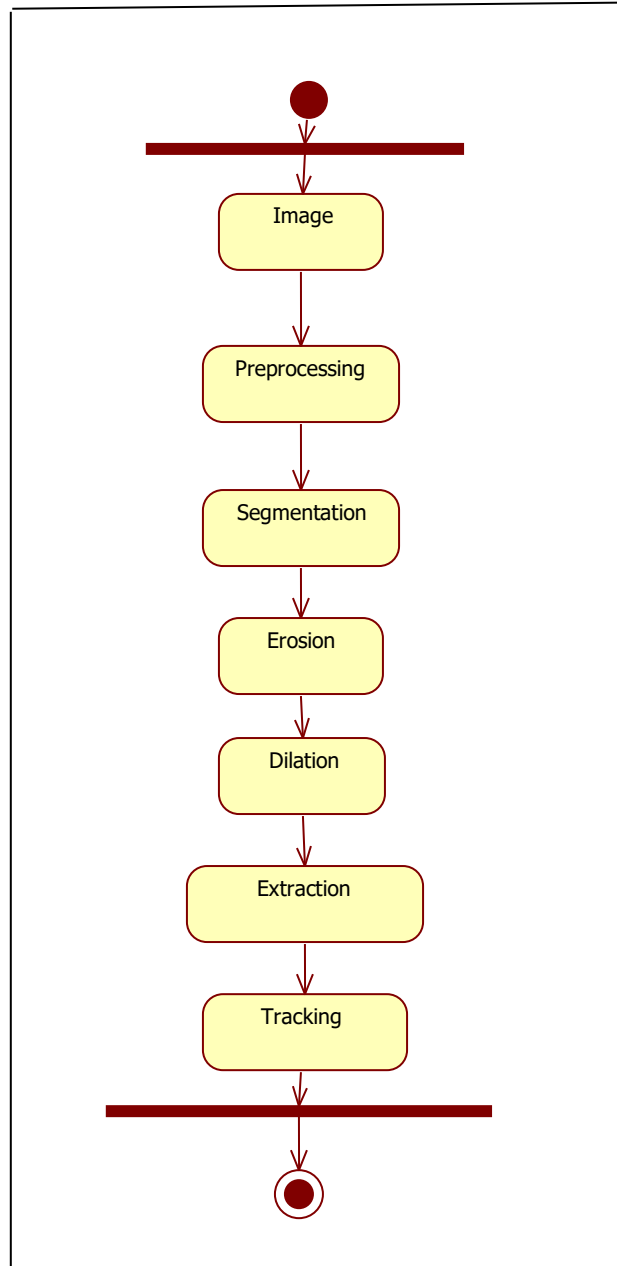
**Fig.4.8:State diagram for Real-time active visual tracking system**

### EXPLANATION:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at

other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

### Activity Diagram



**Fig.4.9 :Activity diagram for Real-time active visual tracking system**

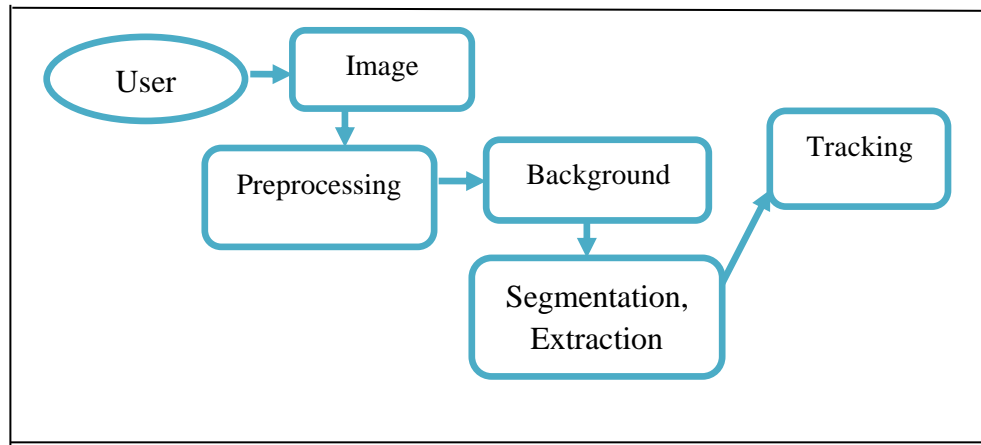
### EXPLANATION:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language,

activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

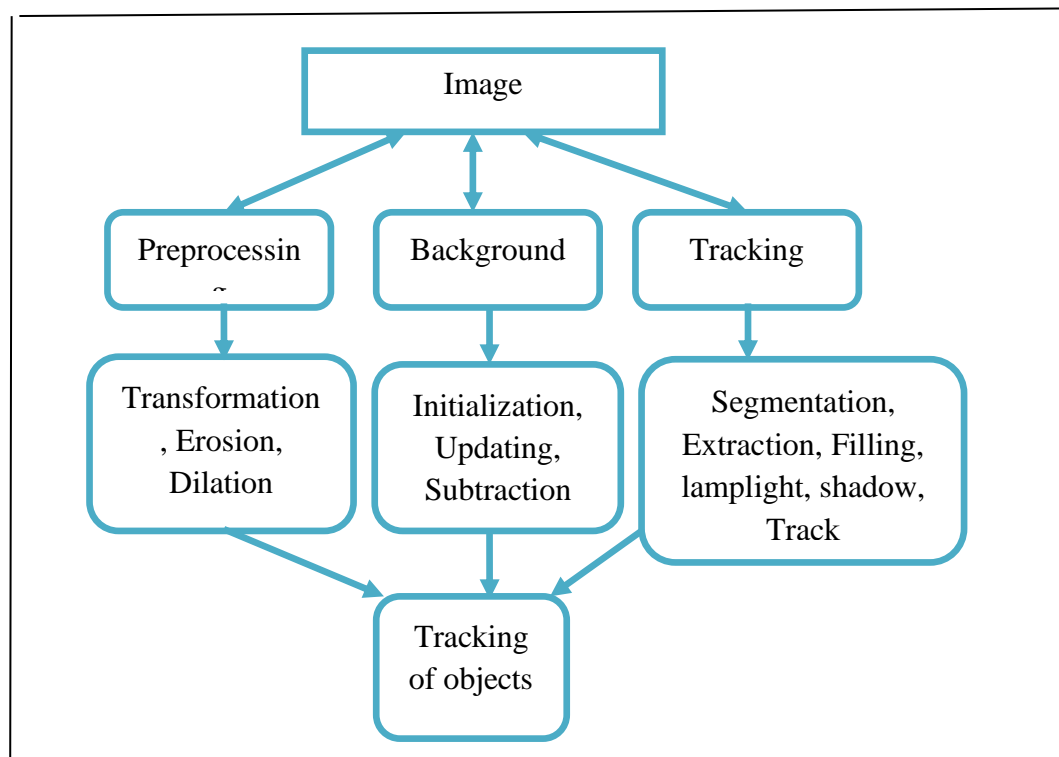
## Data Flow Diagram

### Level 0



**Fig.4.10 :Data flow diagram for Real-time active visual tracking system**

### Level1



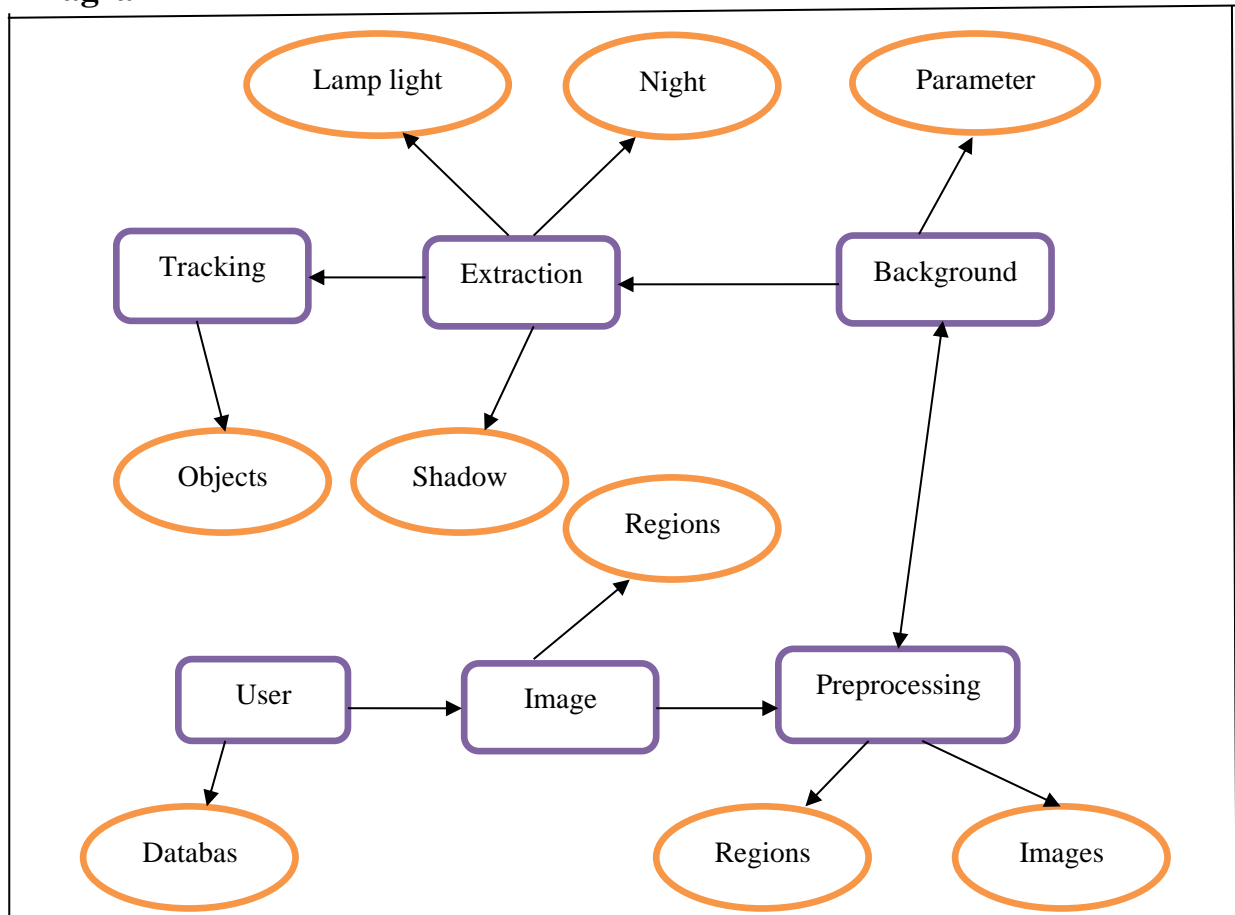
**Fig.4.11:Data flow daigram for Real-time active visual tracking system**

## EXPLANATION:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

## E-R Diagram

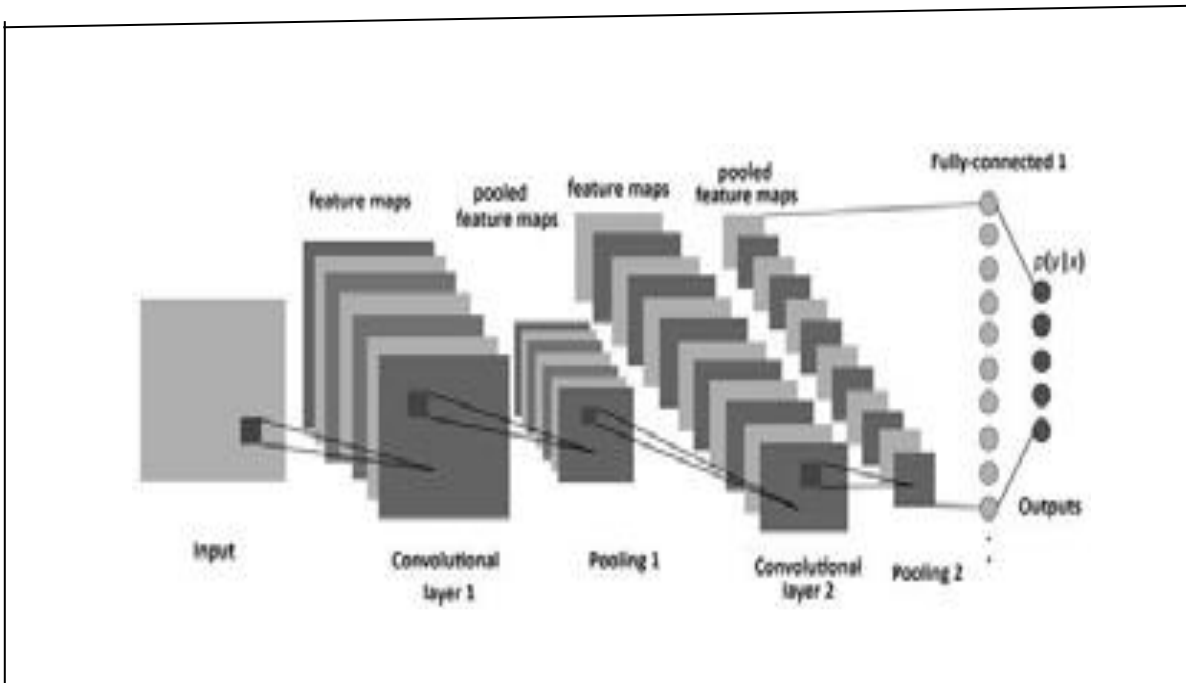


**Fig.4.12: E-R diagram for Real-time active visual tracking system**

### EXPLANATION:

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

### SYSTEM ARCHITECTURE:



**Fig.4.13: System Architecture for Real-time active visual tracking system**

### EXPLANATION:

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of



# CHAPTER 5

## DEVELOPMENT TOOLS

### Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

### History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

### Importance of Python

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## Features of Python

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Libraries used in python:**

- numpy - mainly useful for its N-dimensional array objects.
- pandas - Python data analysis library, including structures such as dataframes.
- matplotlib - 2D plotting library producing publication quality figures.
- scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.

## CHAPTER 6

### IMPLEMENTATION

#### 6.1 GENERAL

**Coding:**

# USAGE

```
# python multi_object_tracking_fast.py --prototxt \  
#      --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --video race.mp4
```

```
# import the necessary packages
```

```
from imutils.video import FPS
```

```
import multiprocessing
```

```
import numpy as np
```

```
import argparse
```

```
import imutils
```

```
import dlib
```

```
import cv2
```

```
import tensorflow as tf
```

```
def start_tracker(box, label, rgb, inputQueue, outputQueue):
```

```
    # construct a dlib rectangle object from the bounding box
```

```
    # coordinates and then start the correlation tracker
```

```
    t = dlib.correlation_tracker()
```

```
    rect = dlib.rectangle(box[0], box[1], box[2], box[3])
```

```

t.start_track(rgb, rect)

# loop indefinitely -- this function will be called as a daemon
# process so we don't need to worry about joining it
while True:

    # attempt to grab the next frame from the input queue
    rgb = inputQueue.get()

    # if there was an entry in our queue, process it
    if rgb is not None:

        # update the tracker and grab the position of the tracked
        # object
        t.update(rgb)
        pos = t.get_position()

        # unpack the position object
        startX = int(pos.left())
        startY = int(pos.top())
        endX = int(pos.right())
        endY = int(pos.bottom())

        # add the label + bounding box coordinates to the output
        # queue

```

```

        outputQueue.put((label, (startX, startY, endX, endY)))

# initialize our list of queues -- both input queue and output queue
# for every object that we will be tracking
inputQueues = []
outputQueues = []

# initialize the list of class labels MobileNet SSD was trained to
# detect
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
            "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
            "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
            "sofa", "train", "tvmonitor"]

# load our serialized model from disk
print("[INFO] loading model...")

net =
cv2.dnn.readNetFromCaffe("mobilenet_ssd/MobileNetSSD_deploy.prototxt",
                        "mobilenet_ssd/MobileNetSSD_deploy.caffemodel")

# initialize the video stream and output video writer
print("[INFO] starting video stream...")

vs = cv2.VideoCapture("race.mp4")

```

```

writer = None

# start the frames per second throughput estimator
fps = FPS().start()

# loop over frames from the video file stream
while True:
    # grab the next frame from the video file
    (grabbed, frame) = vs.read()

    # check to see if we have reached the end of the video file
    if frame is None:
        break

    # resize the frame for faster processing and then convert the
    # frame from BGR to RGB ordering (dlib needs RGB ordering)
    frame = imutils.resize(frame, width=600)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # if our list of queues is empty then we know we have yet to
    # create our first object tracker
    if len(inputQueues) == 0:
        # grab the frame dimensions and convert the frame to a blob

```

```

(h, w) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 0.007843, (w, h), 127.5)

# pass the blob through the network and obtain the detections
# and predictions
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in np.arange(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated
    # with the prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections by requiring a minimum
    # confidence
    if confidence > 0.7:
        # extract the index of the class label from the
        # detections list
        idx = int(detections[0, 0, i, 1])
        label = CLASSES[idx]

        # if the class label is not a person, ignore it

```



```

if CLASSES[idx] != "person":
    continue

# compute the (x, y)-coordinates of the bounding box
# for the object
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
bb = (startX, startY, endX, endY)

# create two brand new input and output queues,
# respectively
iq = multiprocessing.Queue()
oq = multiprocessing.Queue()
inputQueues.append(iq)
outputQueues.append(oq)

# spawn a daemon process for a new object tracker
p = multiprocessing.Process(
    target=start_tracker,
    args=(bb, label, rgb, iq, oq))
p.daemon = True
p.start()

```

```

        # grab the corresponding class label for the detection
        # and draw the bounding box
        cv2.rectangle(frame, (startX, startY), (endX, endY),
                       (0, 255, 0), 2)
        cv2.putText(frame, label, (startX, startY - 15),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255,
0), 2)

```

```

# otherwise, we've already performed detection so let's track
# multiple objects
else:

```

```

    # loop over each of our input queues and add the input RGB
    # frame to it, enabling us to update each of the respective
    # object trackers running in separate processes
    for iq in inputQueues:
        iq.put(rgb)

```

```

# loop over each of the output queues
for oq in outputQueues:

```

```

    # grab the updated bounding box coordinates for the
    # object -- the .get method is a blocking operation so
    # this will pause our execution until the respective
    # process finishes the tracking update

```

```

(label, (startX, startY, endX, endY)) = oq.get()

# draw the bounding box from the correlation object
# tracker
cv2.rectangle(frame, (startX, startY), (endX, endY),
               (0, 255, 0), 2)

cv2.putText(frame, label, (startX, startY - 15),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)

# check to see if we should write the frame to disk
if writer is not None:
    writer.write(frame)

# show the output frame
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop
if key == ord("q"):
    break

# update the FPS counter
fps.update()

```

```
# stop the timer and display FPS information

fps.stop()

print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))


# check to see if we need to release the video writer pointer
if writer is not None:

    writer.release()


# do a bit of cleanup
cv2.destroyAllWindows()
vs.release()
```

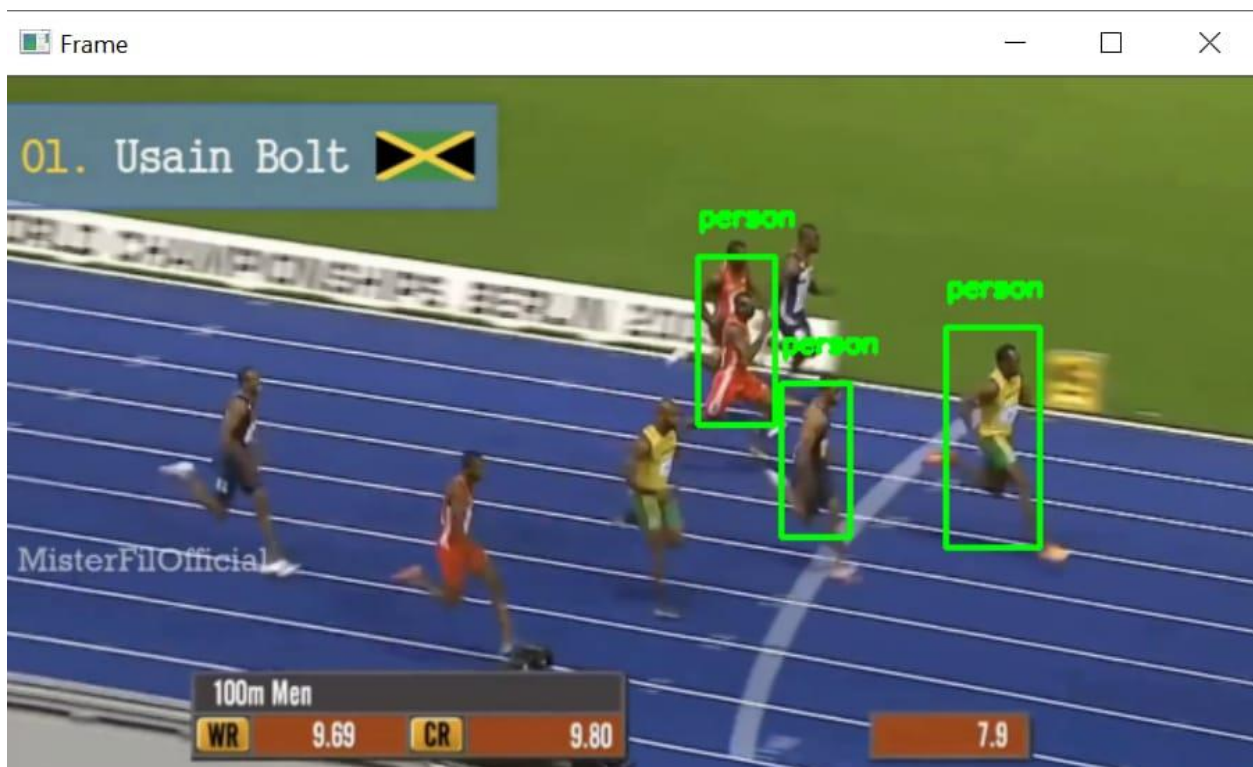
## CHAPTER 7

### SNAPSHOTS

#### General:

This project implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

#### SNAPSHOTS



**Fig.7.1: Output for Real-time active visual tracking system**

#### EXPLANATION:

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of deep learning

## **CHAPTER 8**

### **SOFTWARE TESTING**

#### **8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

#### **8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

#### **8.3 Types of Tests**

##### **8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

### **8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

### **8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

### **8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

#### **Acceptance testing for Data Synchronization:**

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

### **8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identify the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.



## **CHAPTER 9**

### **CONCLUSION & FUTURE ENHANCEMENT**

#### **9.1 CONCLUSION**

The tracking module provides feedbacks to the object detection module to improve the local detection performance. According to the trajectories in the top hypothesis, the multiple object tracking module predicts the most likely locations to detect objects. This interaction tightly integrates the object detection and tracking, and makes both of them more reliable.

#### **9.2 FUTURE ENHANCEMENT**

An image based likelihood is then computed to give a probability to each hypothesis. This computation is based on the object detection probability, appearance similarity, trajectory smoothness and image foreground coverage and compactness. The probabilities are calculated based on a sequence of images, therefore, they are temporally global representations of hypothesis likelihood. The hypothesis are ranked by their probabilities and the unlikely hypotheses are pruned from the graph in the hypothesis-management

## REFERENCES

- [1] J.P. MacCormick and A. Blake, “A probabilistic exclusion principle for tracking multiple objects,” in ICCV99, 1999, pp. 572–578.
- [2] H. Tao, H.S. Sawhney, and R. Kumar, “A sampling algorithm for tracking multiple objects,” in Vision Algorithms 99, 1999.
- [3] M. Isard and J.P. MacCormick, “Bramble: A bayesian multiple-blob tracker,” in ICCV01, 2001, pp. II: 34–41.
- [4] C. Hue, J.P. Le Cadre, and P. Perez, “Tracking multiple objects with particle filtering,” IEEE Trans. on Aerospace and Electronic Systems, vol. 38, no. 3, pp. 791–812, July 2002.
- [5] D.B. Reid, “An algorithm for tracking multiple targets,” AC, vol. 24, no. 6, pp. 843–854, December 1979.
- [6] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe, “Sonar tracking of multiple targets using joint probabilistic data association,” IEEE Journal Oceanic Eng., vol. OE-8, pp. 173–184, July 1983.
- [7] R.L. Streit and T.E. Luginbuhl, “Maximum likelihood method for probabilistic multi-hypothesis tracking,” in Proceedings of SPIE International Symposium, Signal and Data Processing of Small Targets, 1994.
- [8] H. Gauvrit and J.P. Le Cadre, “A formulation of multi target tracking as an incomplete data problem,” IEEE Trans. on Aerospace and Electronic Systems, vol. 33, no. 4, pp. 1242–1257, Oct 1997.
- [9] S. Avidan, “Support vector tracking,” in CVPR01, 2001, pp. I:184–191.
- [10] I. Haritaoglu, D. Harwood, and L.S. Davis, “W4s: A realtime system for detecting and tracking people in 2 1/2-d,” in ECCV98, 1998.