**Unusual Human Activity Detection using Open CV Python with Machine Learning**

**INTRODUCTION**

In this project, we propose a novel method for unusual human activity detection in crowded scenes. Specifically, rather than detecting or segmenting humans, we devised an efficient method, called a motion influence map, for representing human activities. The key feature of the proposed motion influence map is that it effectively reflects the motion characteristics of the movement speed, movement direction, and size of the objects or subjects and their interactions within a frame sequence. Using the proposed motion influence map, we further developed a general framework in which we can detect both global and local unusual activities. Furthermore, thanks to the representational power of the proposed motion influence map, we can localize unusual activities in a simple manner. In our experiments on three public datasets, we compared the performances of the proposed method with that of other state-of-the-art methods, and showed that the proposed method outperforms these competing method. Over a last decade it has been seen the rapid growth and an extraordinary improvement in real-time video analysis. Video surveillance is a prominent area of research which includes recognition of human activities and categorisation of them into usual (normal), unusual (abnormal) or suspicious activities. Due to exponential increase in crime rate, surveillance systems are being put up in malls, stations, schools, airports etc. The face recognition using deep learning and image processing is used to detect the criminal in particular area such as bank, atm, public places etc.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** | i  v  vii  xi  xii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  **4.1 general**  4.1.1 activity diagram  4.1.2 USE CASE DIAGRAM  4.1.3 DATA FLOW DIAGRAM  4.1.4SEQUENCE DIAGRAM  4.1.5 COLLABORATION DIAGRAM  4.1.6CLASS 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 |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 general |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS |  |
| **10** | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Activity Diagram |  |
| 4.3 | Use case Diagram |  |
| 4.4 | Data flow diagram |  |
| 4.5 | Sequence diagram |  |
| 4.6 | Collaboration diagram |  |
| 4.7 | Class diagram |  |
| 4.8 | Architecture Diagram |  |
| 4.9 | State Diagram |  |
| 4.1 | Component Diagram |  |
| 4.12 | E-R Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | 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 | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal 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 | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | Database |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | PWS | Personalized Web Search |
| 5. | UPS | User Personalized Search |
| 6. | JRE | Java Runtime Environment |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Simple activities can be modeled accurately as Markov Chains. However, complex or unfamiliar activities are often difficult to understand and model. For example, a researcher studying activities of daily living for a person with dementia will have a difficult time fitting a model unless she is an expert in dementia and understands its related behavioral science.

* 1. **OBJECTIVE**

The goal of activity recognition is to recognize common human activities in real life settings. Accurate activity recognition is challenging because human activity is complex and highly diverse. Several probability-based algorithms have been used to build activity models.

* 1. **Existing System:**

Numerous attempts have been made in this field to automatize video surveillance but each and every approaches has its own pros and cons. On the basis of prior knowledge and human involvement in the learning process, the research in human activity recognition can be categorized as supervised, unsupervised and semi supervised.

**LITERATURE SURVEY:**

**Title:** Detecting Activities of Daily Living with Smart Meter

**Author:** J. Clement, J. Ploennigs, and K. Kabitzsch.

**Year:** 2013

**Description:** Smart meters provide us new information to visualize, analyze, and op-timize the energy consumption of buildings, to enable demand-response optimiza-tions, and to identify the usage of appliances. They also can be used to help older people to stay longer independent in their homes by detecting their activity and their behavior models to ensure their healthy level. This paper reflects methods that can be used to analyze smart meter data to monitor human behavior in single apartments. Two approaches are explained in detail. The Semi-Markov-Model (SMM) is used to train and detect individual habits by analyzing the SMM to find unique structures representing habits. A distribution of the most possible executed activity (PADL) will be calculated to allow an evaluation of the currently executed activity (ADL) of the inhabitant. The second approach introduces an impulse based method that also allows the detection of ADLs and focuses on temporal analysis of parallel ADLs. Both methods are based on smart meter events describing which home appliance was switched. Thus this paper will also give an overview of popular strategies to detect switching events on electricity consumption data.

**Title:** Incrementally Mining Usage Correlations among Appliances in Smart Homes

**Author:** Y. C. Chen, H. C. Hung, B. Y. Chiang

**Year:** 2015

**Description:** Recently, due to the great advent of sensor technology, residents can collect household appliance usage data easily. However, in general, usage data are generated progressively, visualizing how appliances are used from huge amount of data is challenging. Thus, an algorithm is needed to incrementally discover appliance usage patterns. Prior studies on usage pattern discovery are mainly focused on mining patterns while ignoring the incremental maintenance of mined results. In this paper, a novel method, Dynamic Correlation Miner (DCMiner), is developed to incrementally capture and maintain the usage correlations among appliances in a smart home environment. Furthermore, several optimization techniques are proposed to effectively reduce the search space. Experimental results indicate that the proposed method is efficient in execution time and possesses great scalability. Subsequent application of DCMiner on a real dataset also demonstrates its practicability.

**Title:** The elderly’s independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development

**Author:** Q. Ni, A. B. Garca Hernando

**Year:** 2015

**Description:** Human activity detection within smart homes is one of the basis of unobtrusive wellness monitoring of a rapidly aging population in developed countries. Most works in this area use the concept of "activity" as the building block with which to construct applications such as healthcare monitoring or ambient assisted living. The process of identifying a specific activity encompasses the selection of the appropriate set of sensors, the correct preprocessing of their provided raw data and the learning/reasoning using this information. If the selection of the sensors and the data processing methods are wrongly performed, the whole activity detection process may fail, leading to the consequent failure of the whole application. Related to this, the main contributions of this review are the following: first, we propose a classification of the main activities considered in smart home scenarios which are targeted to older people's independent living, as well as their characterization and formalized context representation; second, we perform a classification of sensors and data processing methods that are suitable for the detection of the aforementioned activities. Our aim is to help researchers and developers in these lower-level technical aspects that are nevertheless fundamental for the success of the complete application.

**Title:** Smart-energy group anomaly based behavioral abnormality detection

**Author:** Alam, Roy, Petruska, and Zemp.

**Year:** 2016

**Description:** Monitoring behavioral abnormality of individuals living independently in their own homes is a key issue for building sustainable healthcare models in smart environments. While most of the efforts have been directed towards building ambient and wearable sensors-assisted activity recognition based behavioral analysis models for remote health monitoring, energy analytics assisted behavioral abnormality prediction have rarely been investigated. In this paper, we propose a data analytic approach that helps detect energy usage anomalies corresponding to the behavioral abnormality of the residents. Our approach relies on detecting everyday appliances usage from smart meter and smart plug data traces in regular activity days and then learning the unique time segment group of each appliance's energy consumption. We focus on detecting behavioral anomalies over a set of energy source data points rather than pinpointing individual odd points. We employ hierarchical probabilistic model-based group anomaly detection [7] to interpret the anomalous behavior and therefore, detect potential tendency towards behavioral abnormality. We apply daily activity logs to evaluate our approach using two realworld energy datasets pertaining to staged functional behaviors, and show that it is possible to detect max. 97% of anomalous days with max. 87% of meaningful micro-behavioral abnormal events generating 1.1% of false alarms. However, we show that our detected abnormality can be meaningfully represented to different stakeholders such as caregivers and family members to understand the nature and severity of abnormal human behavior for sustaining better healthcare.

**Title:** Cloud-supported cyber-physical localization framework for patients monitoring

**Author:** M. S. Hossain

**Year:** 2017

**Description:** The potential of cloud-supported cyber-physical systems (CCPSs) has drawn a great deal of interest from academia and industry. CCPSs facilitate the seamless integration of devices in the physical world (e.g., sensors, cameras, microphones, speakers, and GPS devices) with cyberspace. This enables a range of emerging applications or systems such as patient or health monitoring, which require patient locations to be tracked. These systems integrate a large number of physical devices such as sensors with localization technologies (e.g., GPS and wireless local area networks) to generate, sense, analyze, and share huge quantities of medical and user-location data for complex processing. However, there are a number of challenges regarding these systems in terms of the positioning of patients, ubiquitous access, large-scale computation, and communication. Hence, there is a need for an infrastructure or system that can provide scalability and ubiquity in terms of huge real-time data processing and communications in the cyber or cloud space. To this end, this paper proposes a cloud-supported cyber-physical localization system for patient monitoring using smartphones to acquire voice and electroencephalogram signals in a scalable, real-time, and efficient manner. The proposed approach uses Gaussian mixture modeling for localization and is shown to outperform other similar methods in terms of error estimation.

* 1. **Proposed System:**

Unusual activity recognition systems are developed to make surveillance system more smart and intelligent. Main aim is to detect suspicious or abnormal activities in videos to avoid future happening or to give alert whenever any type of mis happening occur. These anomalous activity recognition system classify normal and abnormal activities of objects. Most of previous research in anomalous or suspicious activity recognition has focused on behavior understanding by training the system manually. Some of work shows unsupervised learning methodologies for activity detection.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Human activity understanding encompasses activity recognition and activity pattern discovery. The first focuses on accurate detection of the human activities based on a predefined activity model. Therefore, an activity recognition researcher builds a high-level conceptual model first, and then implements the model by building a suitable pervasive system.

**2.2 METHODOLOGIES**

**MODULE:**

**Video streaming:**

**Import model**

**Normalization**

**Activity detection**

**Video streaming:**

In this process we are capturing video from video devices and that video stream we are uploading into a Resnet model. Then the processing was started as detecting the activity.

**Import model:**

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. he fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

**Normalization:**

Image Normalization is a process in which we change the range of pixel intensity values to make the image more familiar or normal to the senses, hence the term normalization. Often image normalization is used to increase contrast which aids in improved feature extraction or image segmentation.

**Activity detection:**

Then finally using resnet detecting the activity of the person which passing on the video. Here based on the human behavior system detecting the activities of that person.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

On the other hand, activity pattern discovery is more about finding some unknown patterns directly from low-level sensor data without any predefined models or assumptions. Hence, the researcher of activity pattern discovery builds a pervasive system first and then analyzes the sensor data to discover activity patterns.

**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 shouls what the system do 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**

**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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**Class Diagram**

**EXPLANATION**

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

**Object Diagram**

**EXPLANATION:**

In the above digram 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.

**Component Diagram**

**Deployment Diagram**

**Sequence Diagram**

**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.

**Collaboration Diagram**

**State Diagram**

**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**

**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

Dataset

Preprocessing

Human Activity Prediction

Splitting

Apply Algorithm

Level 1

Training Data, Testing Data

Input Data Set

Preprocessing

Human Activity Prediction

Data, Cleaning Data, Remove Null Values

Apply Algorithm

Splitting

Detection

**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**

Preprocessing

Splitting

Human Activity

User

Algorithm

**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**

Data Gathering

Data Preparation

Data Cleaning

Clustering Data

Segmentation

Feature Extraction

Mining Data

Prediction Human Activity

**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.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding: \*\* In Separate File**

**CHAPTER 7**

**SNAPSHOTS**

**General:**

**SNAPSHOTS**

**\*\* Separate File**

**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.

**8.3Types 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 identity 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 & REFERENCE**

**9.1 CONCLUSION**

Human activity understanding encompasses activity recognition and activity pattern discovery. The system focuses on accurate detection of the human activities based on a predefined activity model. Therefore, an activity recognition researcher builds a high-level conceptual model, and then implements the model by building a suitable pervasive system. On the other hand, activity pattern discovery is more about finding some unknown patterns directly from data without any predefined models or assumptions.

**REFERENCES**

[1] J. Clement, J. Ploennigs, and K. Kabitzsch, Detecting Activities of Daily Living with Smart Meters. Springer, Germany, 11 2013, ch. Advance Technology and Societal Change.

[2] J. Liao, L. Stankovic, and V. Stankovic, “Detecting household activity patterns from smart meter data,” in Intelligent Environments (IE), 2014 International Conference on, 6 2014.

[3] Y. C. Chen, H. C. Hung, B. Y. Chiang, S. Y. Peng, and P. J. Chen, “Incrementally mining usage correlations among appliances in smart homes,” in Network-Based Information Systems (NBiS), 2015 18th International Conference on, 2015.

[4] Q. Ni, A. B. Garca Hernando, and I. P. de la Cruz, “The elderly’s independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development”,2015.

[5] C. Chalmers, W. Hurst, M. Mackay, and P. Fergus, “Smart meter profiling for health applications,” in 2015 International Joint Confer- ence on Neural Networks (IJCNN), July 2015.

[6] K. Gajowniczek and T. Zabkowski, “Data mining techniques for detecting household characteristics based on smart meter data,” Energies, vol. 8, no. 7,2015

[7] Alam, Roy, Petruska, and Zemp, “Smart-energy group anomaly based behavioral abnormality detection,” in 2016 IEEE Wireless Health (WH), Oct 2016.

[8] M. S. Hossain, “Cloud-supported cyber-physical localization framework for patients monitoring,” IEEE Systems Journal, vol. 11, no. 1, pp. 118–127, March 2017.

[9] Marc Wenninger, Jochen Schmidt and Toni Goeller, “Appliance Usage Prediction for the Smart Home with an Application toEnergy Demand Side management and Why Accuracy is not a Good Performance Metric for this Problem”, SMARTGREENS 2017 - 6th International Conference on Smart Cities and Green ICT Systems.

[10]C. Chalmers, W. Hurst, M. Mackay, and P. Fergus, “Smart meter profiling for health applications,” in 2015 International Joint Conference on Neural Networks (IJCNN), July 2015, pp. 1–7.