Chapter 4

SYSTEM DESIGN

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

4.1 System Architecture

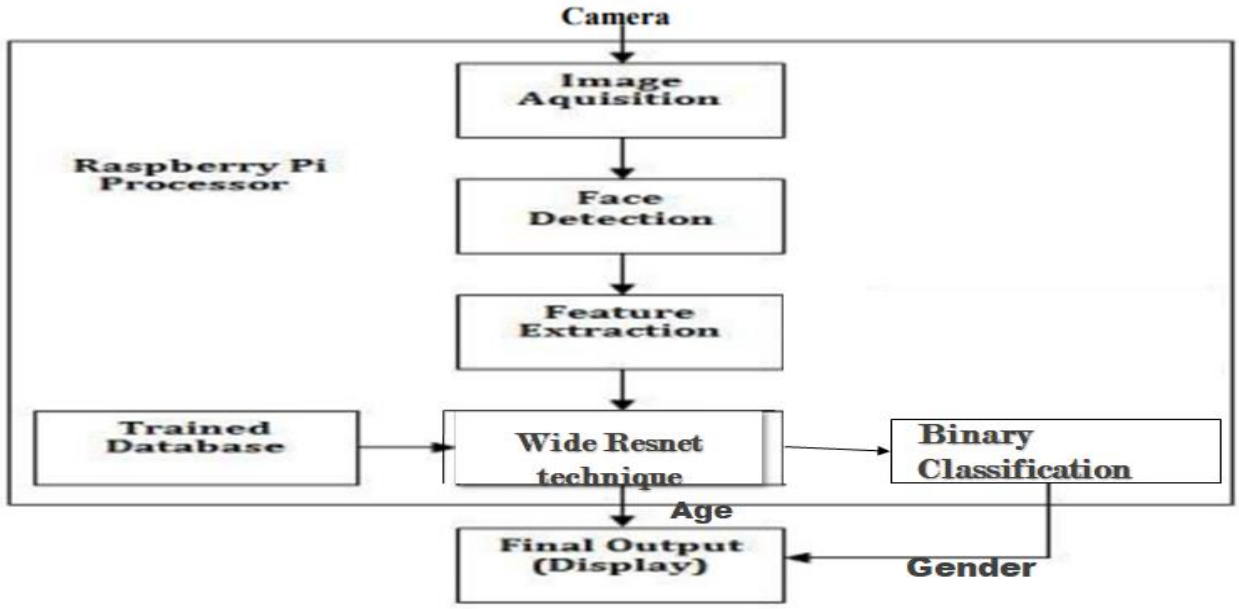

Figure 4.1 Flowchart of overall system design

As shown in figure 4.1, the image will be captured using a camera as an input. Image acquisition is defined as the action of retrieving an image from some source, usually a hardware-based source for processing. It is the first step in the workflow sequence. After image acquisition, the face will be detected. Once the face is detected, features are extracted using a feature extraction method with convolution neural network. Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Along with trained dataset using Support Vector Machine,

the final output will be displayed. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

4.2 Detailed Design

Face Detection and Tracking:

The face detection and tracking component is very important in designing the recognition system. The properties of the detection algorithm will directly affect properties of the overall recognition system. It is clear that undetected faces will not be recognized. Also considering the goal of real time functionality on embedded devices where limited resources are available, spending most of the early on will reduce the application of the other blocks in the figure 4.2.

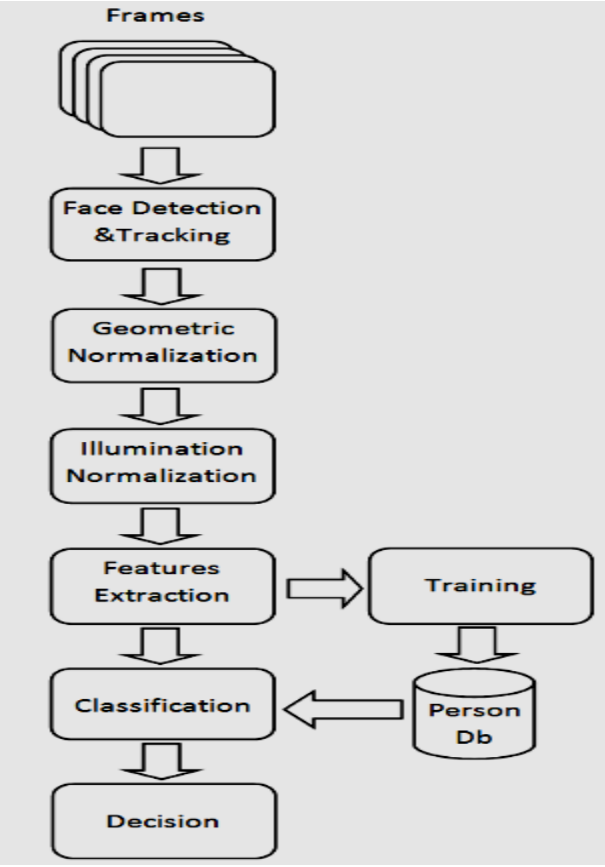

Figure 4.2 System Architecture

Geometric Normalization:

It is very important to detect and track the faces in all conditions and variations. When comparing local regions between faces, an image registration step must be performed so corresponding facial features are synchronised. Simple geometric normalization usually involves bringing the faces to a standard size and rotating them in-plane in order to bring the eyes on the same horizontal line. Figure 4.3 shows some face samples before and after applying the geometric normalization.

Figure 4.3 Geometric Normalization

Illumination Normalization:

If we can control the image capturing environment and impose strict requirements regarding lighting conditions (i.e. control access), recognition accuracy can be improved. In most scenarios where video face recognition is employed, the variations in lighting conditions when the faces are captured can range between dark and bright extremes. The profile face samples for each person to be recognized are captured in very different conditions with still images than those used in video imagery. A pre-processing algorithm should be used to minimize the effect of the lighting conditions when capturing the video images.

Feature Extraction:

Together with the useful information that can be used to differentiate between individuals, the face images described by the pixel values contain redundant information and information that

can be ignored in the classification stage. By extracting only the useful information in this step we improve the accuracy of the recognition and also lower the storage requirement for each face.

Classification:

In the case of still image recognition, the system makes a decision if the test face belongs to one of the people in the database and if so, which one (based on comparing the features computed in the previous step for test faces and a database of people).

In the case of video image recognition, the system compares the series of test faces with those in the sample database. Most commonly this is implemented as a series of still images derived comparisons and at each frame the confidence of our decision is modified based on the history of previous comparisons. Simple classification algorithms like distance between feature vectors are preferred because of their simplicity and speed. More complex learning algorithms can be used if there are enough computation resources.

4.3 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

Figure 4.4 shows the data flow diagram of recognizing the face.

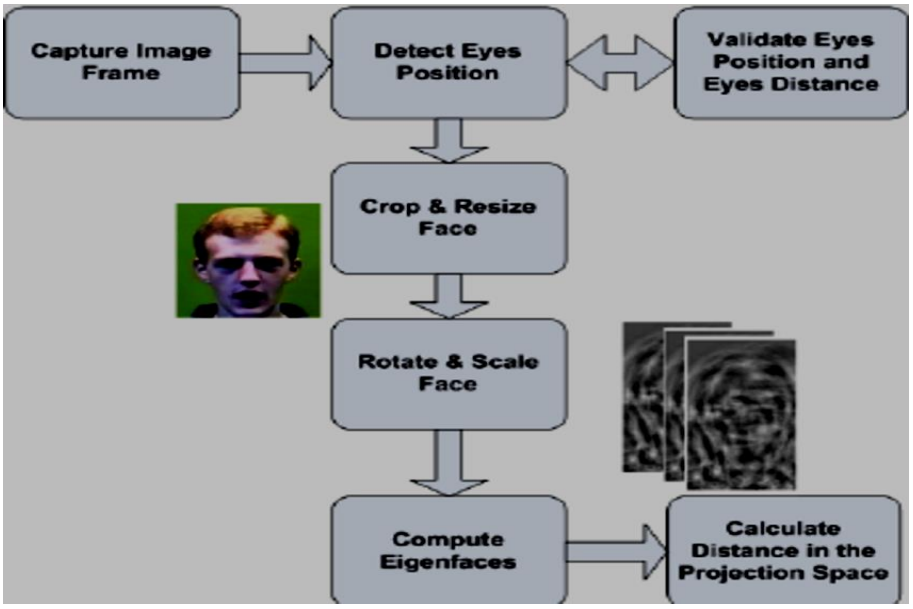

Figure 4.4 Data Flow diagram

4.4 Use case Diagram

Figure 4.5 shows the Use Case diagram.

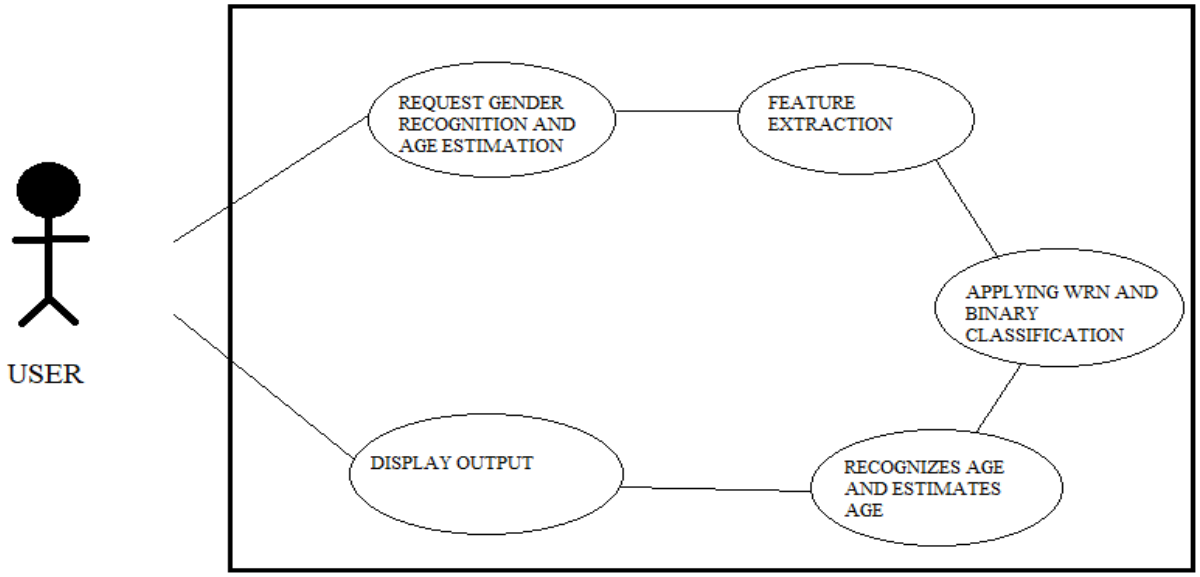

Figure 4.5 Use Case Diagram

Use cases specify the expected behavior, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

4.5 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Figure 4.6 shows sequence diagram.

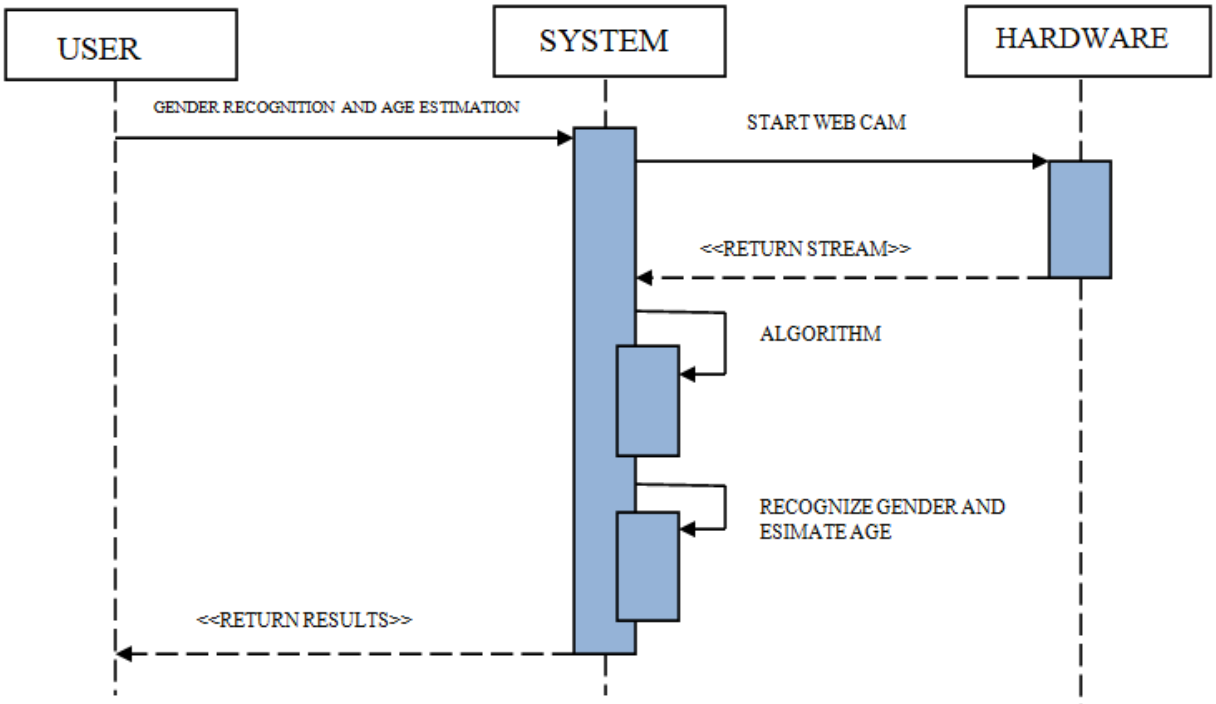

Figure 4.6 Sequence Diagram

4.6 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes,

their attributes, operations (or methods), and the relationships among objects. Figure 4.7 shows the class diagram.

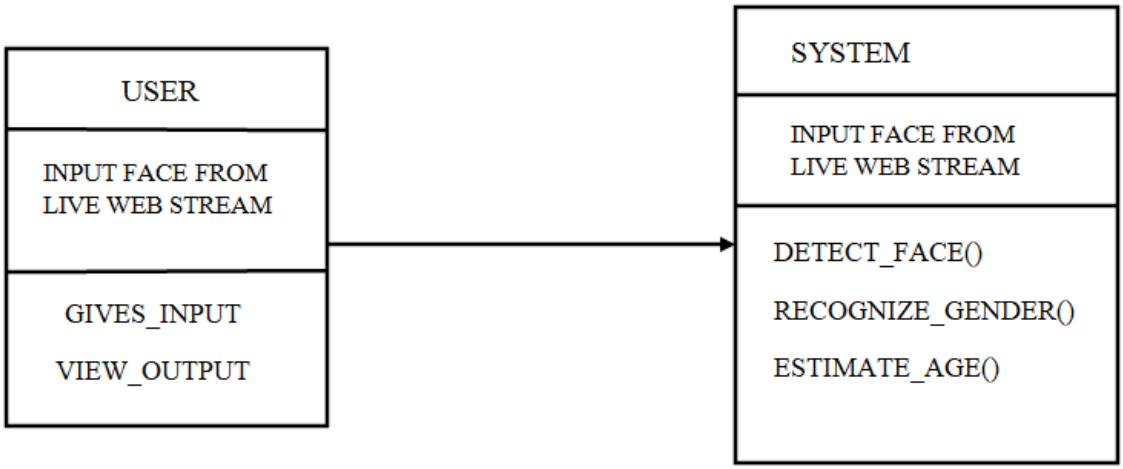

Figure 4.7 Class diagram