**Software Requirements Specification (SRS) Document**

|  |  |
| --- | --- |
|  | **Object Detection and Tracking against 3D Mesh Model**  **Team 35**  **AadilMehdi J Sanchawala**  **Antony Martin**  **Priyank Modi**  **Rohan Chacko** |

Note: This is a “living document”, meaning its content will change with the implementation of the project. Use it to capture key project requirements and make sure that your product features match the requirements exactly – if you wish to add any features, they must be added first to the requirements. Changes in the document must be approved by the customer (mentor) and the instructor or TA Mentor.

*Remove this text and the descriptive paragraphs in each section stating what to do before you turn it in*.

# Brief problem statement

The project deals with detection and tracking of an inventory of objects against their 3D Mesh models in real time video(RGB) stream.

Replace this text and the instructions below with your statement in black.  
(2-5 lines describing the problem being addressed. Note that even if you are simply restating what is already in the project description, you must rephrase it in your words. This gives an opportunity for the customer to identify and provide feedback on differences in interpretation, if any).

# System requirements

Opencv

C++

Cython

obj/stl format for 3D models

C++ libraries associated with fast matrix operations, image processing and computer vision

Replace this text and the instructions below with your statement in black.  
(Identify the system requirements for your solution. If you require particular technologies, languages and libraries, list them as well).

# Users profile

The Medical Personnel interacts with the system to keep track of the surgical instruments and organs of the patients during the operation procedures. The said person would be interacting with the system in the user mode(application level). The user requires a basic working knowledge of computers.

# Feature requirements (described using use cases)

**Read the instructions below and fill in the table. Delete all the blue text turning it in.**

(This is a numbered list of use cases that are the features of the system to be implemented. Each use case is an operation that the user can perform on/with the system. For each use case, provide a description (2-3 sentences) so you know what to build and so you can write a test case to demonstrate that your system provides that feature. For each use case, you will identify (during release planning) the release in which it will be implemented: R1 or R2. Typically, your project will have 10-15 use cases, but feel free to add or delete table rows if you decide to use finer-grain or coarse-grain use cases).

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **User Case Name** | **Description** | **Release** |
| 1 | Load and Register the model. | Load the models into the system, registering the model by identifying the point features. | R1 |
| 2 | Registering the input streaming device. | Registering the input device. | R1 |
| 3 | Detect when the object enters the frame. | Detect the position of entry of the object. Note the time and position of entry of the object. | R1 |
| 4 | Track the object in real time. | Note the position (coordinates) of the object with respect to the frame based on the confidence score above a threshold. | R1 |
| 5 | Note the time at which the object leaves the frame | Note the time at which the object leaves the frame and notify the user about the last seen time and position of the object. | R1 |
| 6 | Extend the above use cases to multiple objects. | Handle occlusions and multiple copies of the same model.  Detect and track the inventory of objects based on the confidence score above a threshold. | R2 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Use case diagram**

**Read the instructions below and fill in the table. Delete all the blue text before adding this to your repository or turning it in to your instructor.**

Draw the UML use case diagram for the system. Make sure the use cases shown in the diagram correspond to the use cases described in the previous section.

**Use case description**

**Delete all the blue text and fill-in the template before adding this to your repository or turning it in to your instructor.**

|  |  |
| --- | --- |
| **Use Case Number:** | UC-01 |
| **Use Case Name:** | Load and Register the Model. |
| **Overview:** | Load the models into the system, registering the model by identifying the point features. |
| **Actors:** | Technical person with working knowledge of computers. |
| **Pre condition:** | Have the mesh model of the required object in STL/OBJ format. |
| **Flow:** | Main (success) Flow:   1. Specify path of the 3D mesh model to be loaded 2. Display a message stating whether the model has been loaded. 3. Identify feature points of the 3D model. 4. Display status message if the model has been registered correctly. |
|  | Alternate Flows:   1. If the model has not been loaded into the system correctly, display appropriate error message. 2. If the feature points have not identified correctly, a corresponding error message is displayed. |
| **Post Condition:** | Model loaded and registered correctly |

|  |  |
| --- | --- |
| **Use Case Number:** | UC-02 |
| **Use Case Name:** | Registering the input streaming device. |
| **Overview:** | Detect the position of entry of the object. Note the time and position of entry of the object. |
| **Actors:** | Technical person with working knowledge of computers. |
| **Pre condition:** | 3D mesh models have to be loaded and registered correctly. |
| **Flow:** | Main (success) Flow:   1. Specify the device index of the streaming device to be used. 2. Verify if the video stream is being fed into the system correctly. 3. Display a message if the streaming device is ready to be used. |
|  | Alternate Flows:  1) Display appropriate error message. |
| **Post Condition:** | Streaming device appropriately calibrated with the system. |

|  |  |
| --- | --- |
| **Use Case Number:** | UC-03 |
| **Use Case Name:** | Detect when the object enters the frame. |
| **Overview:** | Detect the position of entry of the object. Note the time and position of entry of the object. |
| **Actors:** | Software |
| **Pre condition:** | Previous use cases must be satisfied. |
| **Flow:** | Main (success) Flow:   1. Detect the position of entry of the object into the frame based on a confidence score. 2. Initialise the object of the respective class. 3. Log the position & time of entry. |
|  | Alternate Flows:  None |
| **Post Condition:** | Object detected. |

|  |  |
| --- | --- |
| **Use Case Number:** | UC-04 |
| **Use Case Name:** | Log the position of the object at any given time. |
| **Overview:** | Note the position (coordinates) of the object with respect to the frame based on the confidence score above a threshold value. |
| **Actors:** | Software |
| **Pre condition:** | Previous use cases must be satisfied. |
| **Flow:** | Main (success) Flow:   1. Detect the object in every frame based on a confidence score. 2. Log the position of the object in every frame. |
|  | Alternate Flows: None |
| **Post Condition:** | Object detected in every frame. |

|  |  |
| --- | --- |
| **Use Case Number:** | UC-05 |
| **Use Case Name:** | Note the time at which the object leaves the frame |
| **Overview:** | Note the time at which the object leaves the frame and notify the user about the last seen time and position of the object. |
| **Actors:** | Software |
| **Pre condition:** | Previous use cases must be satisfied. |
| **Flow:** | Main (success) Flow:   1. Detect the position of departure of the object from the frame based on a confidence score. 2. Log the position & time of exit. |
|  | Alternate Flows:  None |
| **Post Condition:** | Last seen position of the object notified to the user. |

|  |  |
| --- | --- |
| **Use Case Number:** | UC-06 |
| **Use Case Name:** | Extend the above use cases to multiple objects. |
| **Overview:** | Handle occlusions and multiple copies of the same model.  Detect and track the inventory of objects based on the confidence score above a threshold. |
| **Actors:** | Software |
| **Pre condition:** | Previous use cases must be satisfied for a single object. |
| **Flow:** | Main (success) Flow:   1. Various objects will be detected when it enters the frame. 2. Objects will be initialised for the respective classes. 3. Position and time of entry of each object will be logged. |
|  | Alternate Flows: None |
| **Post Condition:** | All objects detected in every frame based on a confidence score. |