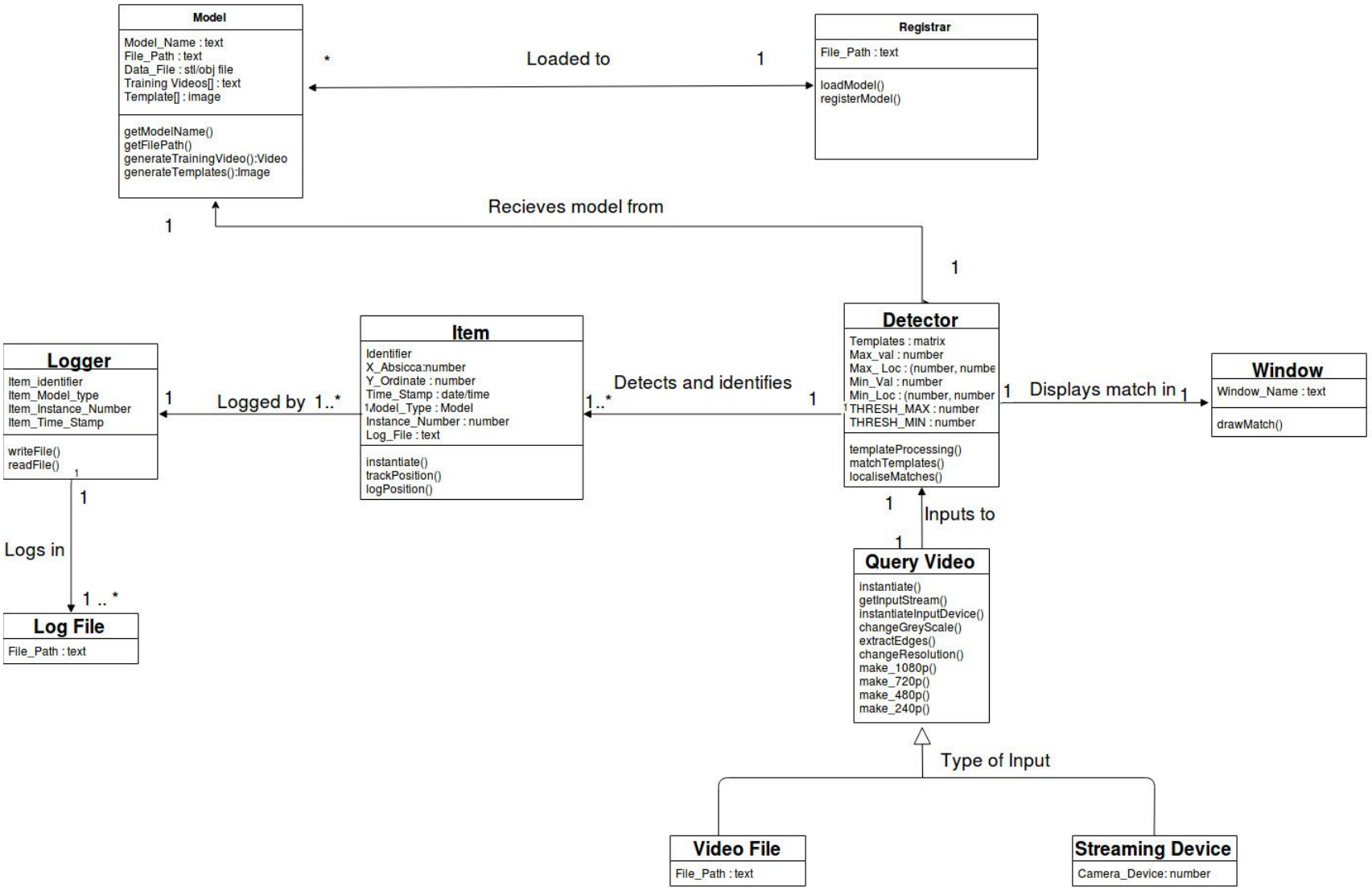
Product Design
Team 35

Members

- **AadilMehdi J Sanchawala**
- **Rohan Chacko**
- **Antony Martin**
- **Priyank Modi**

Object Detector using 3D Mesh Models



<table>
<tr><th>Class Number</th><th>Class Name</th><th>State / Behavior</th></tr>
<tr><td rowspan="4">1</td><td rowspan="4">Model</td><td>Model State</td></tr>
<tr><td>● Model_Name : text
● File_Path : text
● Data_File : stl/obj file of the model
● Training_Videos : list of paths to the training videos extracted from the Data_File
● Templates : Img files</td></tr>
<tr><td>Model Behavior</td></tr>
<tr><td>● getModelName()
○ Register the name of the model.
● getFilePath()
○ Gets the file path from the user.
● generateTrainingVideo()
○ Generate a set of edge template training videos from predefined orientations of the mesh model.
● generateTemplates()
○ Get all the edge templates of the model from the Training_Videos, of the mesh model.</td></tr>
<tr><td rowspan="4">2</td><td rowspan="4">Registrar</td><td>Registrar State</td></tr>
<tr><td>● File_Path : text</td></tr>
<tr><td>Registrar Behavior</td></tr>
<tr><td>● loadModel()
○ Load the mesh model into the system.
● registerModel()
○ Register the mesh model into the database of the system and label it according to the identifier (Model_Name) provided by the Model class.</td></tr>
</table>

<table>
<tr><td rowspan="3">3</td><td rowspan="3">Item</td><td>Item State</td></tr>
<tr><td>

- Identifier
- X_Abscissa : number
- Y_Ordinate : number
- Time_Stamp : date/time
- Model_Type : Model
- Instance_Number : number
- Log_File : text

</td></tr>
<tr><td>Item Behavior</td></tr>
<tr><td colspan="2">

- instantiate()
  - Constructor for initialize the object detected in the video screen, and labeling it under a Model_Type and giving it an instance number.
- trackPosition()
  - Modify the X_Abscissa and Y_Ordinate values for the object with respect to a constant time frame.
- logPosition()
  - Write the position of the object into the Log_File of the Item and mention the time stamp as well.

</td></tr>
<tr><td rowspan="3">4</td><td rowspan="3">Logger</td><td>Logger State</td></tr>
<tr><td>

- Item_Identifier
- Item_Model_Type
- Item_Instance_Number
- Item_Time_Stamp

</td></tr>
<tr><td>Logger Behavior</td></tr>
<tr><td colspan="2">

- writeFile()
  - Get the values returned by logPosition method from the Item class, create appropriate directory structure and log file if not already present and write into the file.
- readFile()
  - Read the position and time stamp values for a particular

</td></tr>
</table>

<table>
<tr><td></td><td></td><td>item of a model type and instance number.</td></tr>
<tr><td rowspan="4">5</td><td rowspan="4">QueryVideo</td><td>QueryVideo State</td></tr>
<tr><td>● File_Path : text
● Camera_Device : number</td></tr>
<tr><td>QueryVideo Behavior</td></tr>
<tr><td>● instantiate()
○ Constructor for the input video stream. Initialize the File_Path of the video stream and Camera_Device index.
● getInputStream()
○ Get the query input stream from the Video_Path or the Camera_Device.
● instantiateInputDevice()
○ Initialize and Register the Camera_Device if any.
● changeGreyScale()
○ Change the input video stream to grey scale.
● extractEdges()
○ Extract the edges from the input video stream per frame of the video.
● changeResolution()
○ Change the resolution of the input video stream into a custom resolution.
● make_1080p()
○ Change the resolution of input video stream to 1080p.
● make_720p()
○ Change the resolution of input video stream to 720p.
● make_420p()
○ Change the resolution of input video stream to 420p.
● make_240()
○ Change the resolution of input video stream to 240p.</td></tr>
</table>

<table>
<tr><td rowspan="4">6</td><td rowspan="4">Detector</td><td>Detector State</td></tr>
<tr><td>

Templates : matrix
Max_Val : number
Max_Loc: (number,number)
Min_Val : number
Min_Loc : (number,number)
THRESH_MAX : number
THRESH_MIN : number
tH: number
tW: number

</td></tr>
<tr><td>Detector Behavior</td></tr>
<tr><td>

templateProcessing()

Retrieve templates from the corresponding directory. Convert image to grayscale. Apply Canny edge detector on each template. Display the result.

matchTemplates()

Match edge templates with each frame of the video stream. Get the brightest matching pixel and its corresponding location

localiseMatches()

Calculate bounding box on the region of interest specified by matchTemplates()

drawMatch()

Draw bounding box based on the coordinates specified by localiseMatches()

</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2">Window</td><td>Window State</td></tr>
<tr><td>

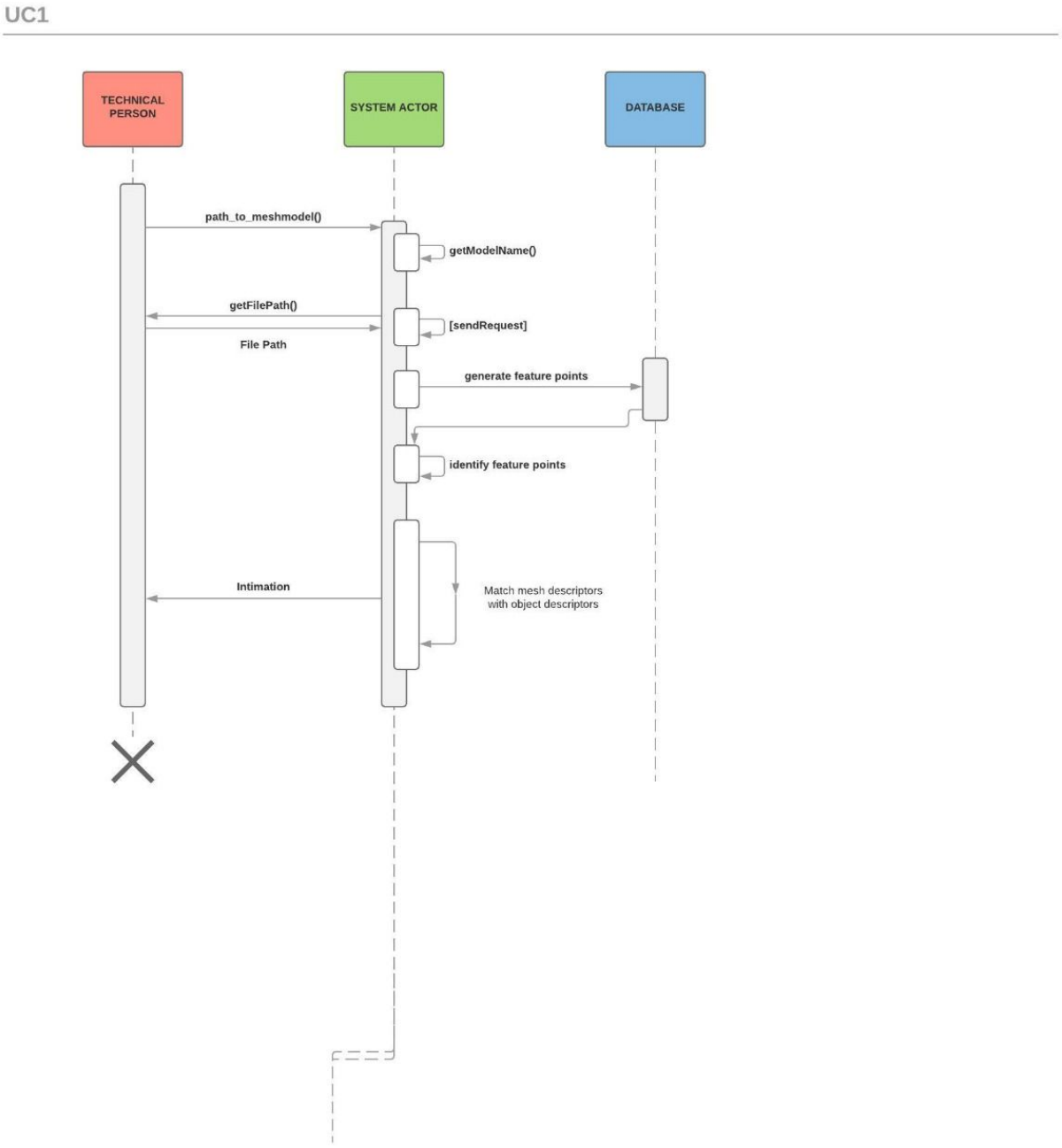
Name : text
Dimension : (number, number)
Match_Coordinates : (number, number)

</td></tr>
</table>

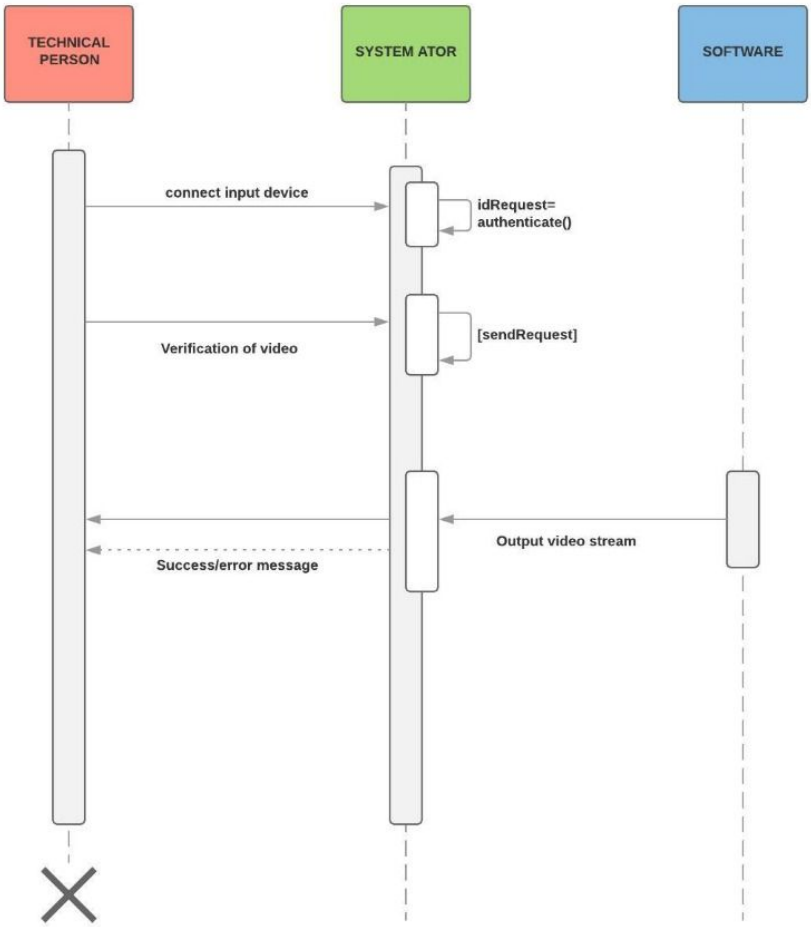
		Window Behavior
		● displayWindow() ○ Display the window on to the screen as per the Dimension. ● drawMatch() ○ Draw the item matches found in the video input stream per frame and draw a bounding box for the item.

.

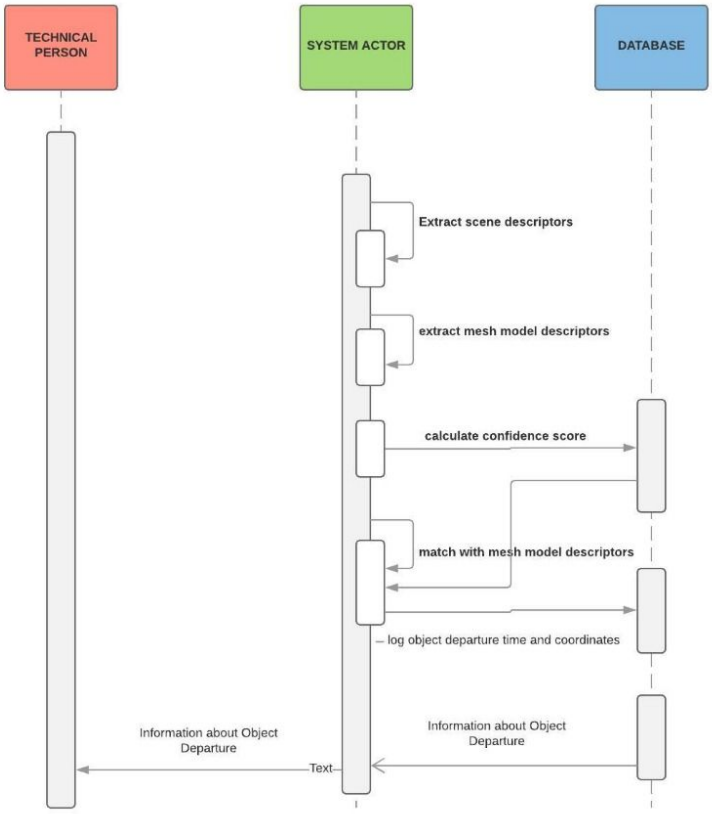
Sequence Diagram(s)



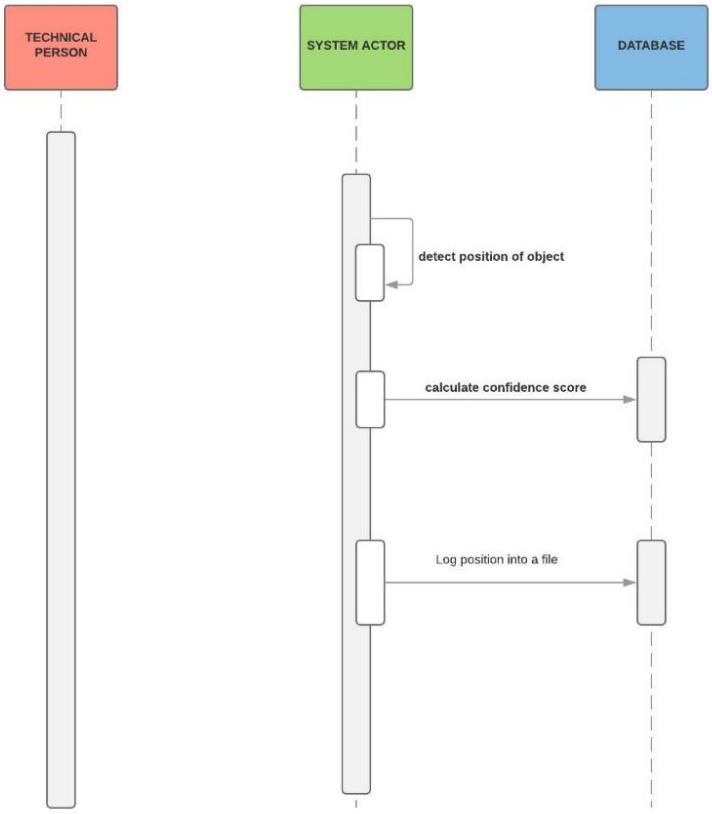
UC-2


TECHNICAL PERSON
SYSTEM ATOR
SOFTWARE
connect input device
idRequest= authenticate()
Verification of video
[sendRequest]
Output video stream
Success/error message

UC-3


TECHNICAL PERSON
SYSTEM ACTOR
DATABASE
Extract scene descriptors
extract mesh model descriptors
calculate confidence score
match with mesh model descriptors
log object departure time and coordinates
Information about Object Departure
Information about Object Departure
Text

UC-4


TECHNICAL PERSON
SYSTEM ACTOR
DATABASE
detect position of object
calculate confidence score
Log position into a file

Design Rationale

First Design

Used a feature based keypoint detector and extract feature descriptors using SIFT/SURF/ORB. This information is extracted for the 3D Mesh Model and for each frame of the video stream. An appropriate matching algorithm is used to match the keypoints of the frame and the most appropriate model identified out of the set of mesh models.

Advantage

- Fast, as the algorithms used were highly optimised for OpenCV.
- Scaled well for multiple objects
- Invariant to rotation, lighting, blur, scale changes of the objects.

Disadvantage

- Time taken for the whole process was slower than what was required.
- Detection and extraction required the objects to have distinct textures and the mesh models does not have a texture. (Major drawback)

The main issue with the first design was that it required the objects to have texture. But mesh models do not have any texture associated with it. Hence, we had to opt for a different approach.

Second Design

After doing appropriate research, we chose multi-scale template matching to proceed with our project. This method used used the projection of the 3D mesh model onto a 2D plane at a particular orientation as a template. This template was used to match the models seen in the frame with the 3D models we had.

Advantage

- Scale invariant
- Did not require the 3D models to have textures.
- Moderate tolerance to blur and lighting effects.

Disadvantage

- Method does not scale well for multiple objects
- As multiple objects are introduced, the software becomes slow.
- Algorithm is inherently rotation invariant.

Since our project required our final design to track objects in any orientation, we had to make our software invariant to rotation effects. For this, we projected the 3D mesh model onto a 2D plane in different orientations. Each of these projections formed a set for that particular model. The set of projections were used to identify and track the respective model in each frame of the video. Since this method is computationally expensive, the software becomes slow on including multiple objects to track. Hence, we will be using a design which is optimised for multi-object tracking.