UNIVERSITY OF
LIVERPOOL

COMP390
2023/24

Computer Vision and AR for Endovascular Intervention

Student Name: [Yulin Huang]

Student ID: [201676465]

Supervisor Name: [Anh Nguyen]

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Anh Nguyen, for his invaluable guidance and support throughout my research. His insights and expertise have been fundamental to the success of this work.

Special thanks also go to Tudor Jianu for his advice and contributions that greatly enhanced this project.

I would like to extend my heartfelt thanks to my family. Their unwavering support and love have been my strength and motivation throughout this journey.

I am also immensely grateful to my girlfriend, Lucy Zhao, for her understanding and encouragement during the course of my thesis. Her support has been a cornerstone of my personal and academic journey.

Furthermore, I wish to acknowledge the critical role played by open-source libraries in the realization of this project. I extend my thanks to the developers of OpenCV[1] and SciKit-Surgery Augmented Reality[2] for providing essential tools and references that significantly contributed to my project.

This work would not have been possible without the collective support and encouragement from each of these individuals and resources. Thank you.

UNIVERSITY OF
LIVERPOOL

COMP390 2023/24

Computer Vision and AR for Endovascular Intervention

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Contents

Abstract

This section should contain a concise summary of the document content.

Statement of Ethical Compliance

Data Category: A
Participant Category: 0
I confirm that I have read the ethical guidelines and will follow them during this project. Further details can be found in the relevant sections of this proposal.

1 Introduction & Background

2 Design & Implementation

2.1 Part1: Real-world Model Interaction and Tracking

In this part, I used OpenCV[1] and SciKit-Surgery Augmented Reality[2] libraries for image processing and model tracking. Qt[3] is used to design the graphical interface, and VTK[4] to build a overlay window[5]. OpenCV and SciKit-Surgery libraries help me process images and track Aruco Marker[6] within video steaming. Qt allows me to create a user interface that ensures user could more easily change multiple settings, which can enhance the user experience. VTK and SciKit-Surgery Augmented Reality library are the management of overlay and multi-layer video rendering in my project. An ArUco Marker Generator is also included, which enables the user to generate different ArUco Markers and save them. This section will detail the system's design, focusing on System Components and Organization, Data Structures and Algorithms, User Interface Design, and Design Notation and Diagrams.

2.1.1 Design

1. **System Components and Organization**
   The project is structured into three primary components, each responsible for distinct functionalities within the system. Here's a detailed breakdown of these components and their organization:

   (a) **Frontend - User Interface**
   The system's user interface is developed using Qt, which is a framework that enables the creation of graphically applications. The main class controlling the UI is *Overlay_and_Tracking.py*, which serves as the central hub for user interactions and display functionalities. This class manages the overlay of models on video streaming and provides interactive buttons for users to control various settings, such as model color, video source, models uploading and changing, or adjusting ArUco marker types and sizes, and more.

   (b) **Backend - Helper Classes**
   The backend is composed of various helper classes, each set to handle specific tasks:

   - *Image Capture:* The *video_source.py* class handles the acquisition of video streams from various sources, including live cameras and recorded media. It is responsible for configuring camera settings,

initializing video capture, and video frame cropping to adapted to screen size. This component ensures the reliability and stability of video feed intake.

- *Model Loading:* Managed by *model_loader.py*, this component is important for the model loading and initializing. It loads ".stl" model files from external files, sets up texture mapping, and prepares the models for real-time overlay. The class also checks for errors in model data to prevent crashes or rendering issues during operation. It also optimizes the structures of model data to enhance rendering efficiency and reduce memory overhead.
- *Model Overlay:* The *overlay_window.py* is central to integrating 3D models with live or recorded video. With the help of VTK, this module could set up a multi-layered rendering environment where each layer can independently handle elements like video backgrounds, 3D models, or some GUI overlays(In the future, maybe.).
- *Transform Management:* The *transform_manager.py* class provides a method for managing 4x4 transformation matrices crucial for spatial adjustments of models in 3D space. It stores and retrieves transformations efficiently. And allow it for dynamic modifications of object orientations and positions.
- *ArUco Marker Tracking:* Functionality provided by *arucotracker.py* includes detecting and decoding ArUco markers from the video stream using OpenCV. This module calculates position and orientation of detected marker, and handle the spatial position data for model tracking.

(c) **Additional Component - ArUco Marker Generator**
An independent component in the system is the ArUco Marker Generator, managed by *Aruco_Generator.py*. This tool allows users to select and visualize different ArUco markers. Users can also save these markers as separate image files.

Organization:

The system's architecture is designed to easy maintenance and scalability. The modular nature of the helper classes allows for isolated development and testing, which enhances the system's robustness and flexibility. This organization simplifies development and testing and enables the integration of additional functionalities in the future with minimal disruption to the existing system.

2. **User Interface Design**

In this section, I will show the user interface design of the two main applications of my Part1. Both applications use the Python-based Qt package for user GUI design. I will also provide a brief description of the Main menu and the ArUco Marker Generator interface. Screenshots of the main layout and main interface will be included in the (a) Main Menu (Overlay and Tracking) and (b) ArUco Generator sections, and screenshots of settings and functionality will be listed separately in the (c) Screen Mockups, Sketches, and Screenshots section.

(a) **Main Menu(Overlay and Tracking)**

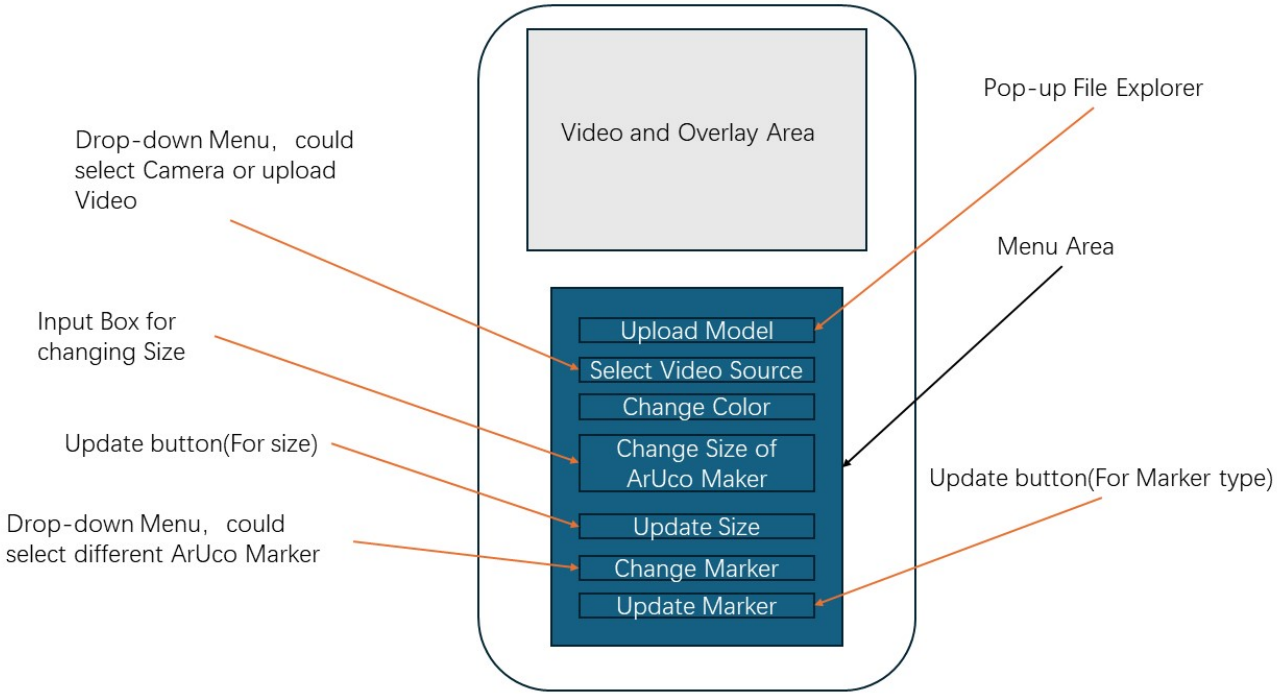

Figure 1: Design of Overlay and Tracking Main Menu

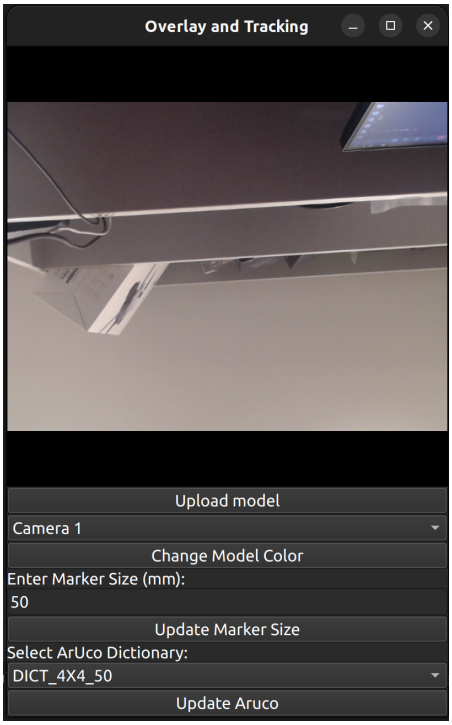

Figure 2: Overlay and Tracking Main Menu(Without Overlay Activated)

The main menu of Overlay and Tracking is the central hub of the application, providing users with access to all the functionalities. The Main Menu consists of the Video and Overlay Area, which displays the video image and overlay of the model, and the Menu Area, which is responsible for setting various parameters. The menu includes buttons for model uploading, video source selection, ArUco marker settings, and model color adjustments.

When you click on the ”Upload Model” button, a file explorer will pop up to select the folder where you want to upload the model. The program will read two files in the folder, one is the model file in ”.stl” format, and the other is a file named ”colours.txt”. the first field in the txt file is the file name of the model, and the last three fields are The first field in the txt file is the file name of the model, and the last three fields are the RGB data of the default colours of the model (the model colours can be changed in the application interface after uploading). After uploading the model will be overlayed onto the video display as Overlay. The second button is the video source selection button, when clicked, a drop-down menu with three options will appear,

"Camera1", "Camera2" and "Upload Video Camera1", "Camera2" and "Upload Video". The app will default to "Camera1" as the video source when it is initially launched. Selecting "Camera1" or "Camera2" will directly switch to the corresponding video source. After selecting the "Upload Video" option, an "Upload Video File" button will appear at the bottom. Clicking on the "Upload Video File" button will start the file explorer, you can select ".mp4", ".avi", ".webm" format video for uploading. After the video has been uploaded the tracking of Overlay and ArUco markers will start automatically if there is already an uploaded model present. Similarly, the video source can be switched back to the Camera input by selecting Camera1. After uploading the model, you can change the colour of the model by clicking the "Change Model Color" button.

To ensure the accuracy of ArUco marker tracking, you need to modify the "ArUco Dictionary" to the corresponding dictionary of the marker you are using and click "Update ArUco "button. You also need to change the "ArUco Marker Size", which represents the size of the real ArUco marker in millimetres, and click the "Update Marker Size" button when you are done.

(b) **ArUco Generator**

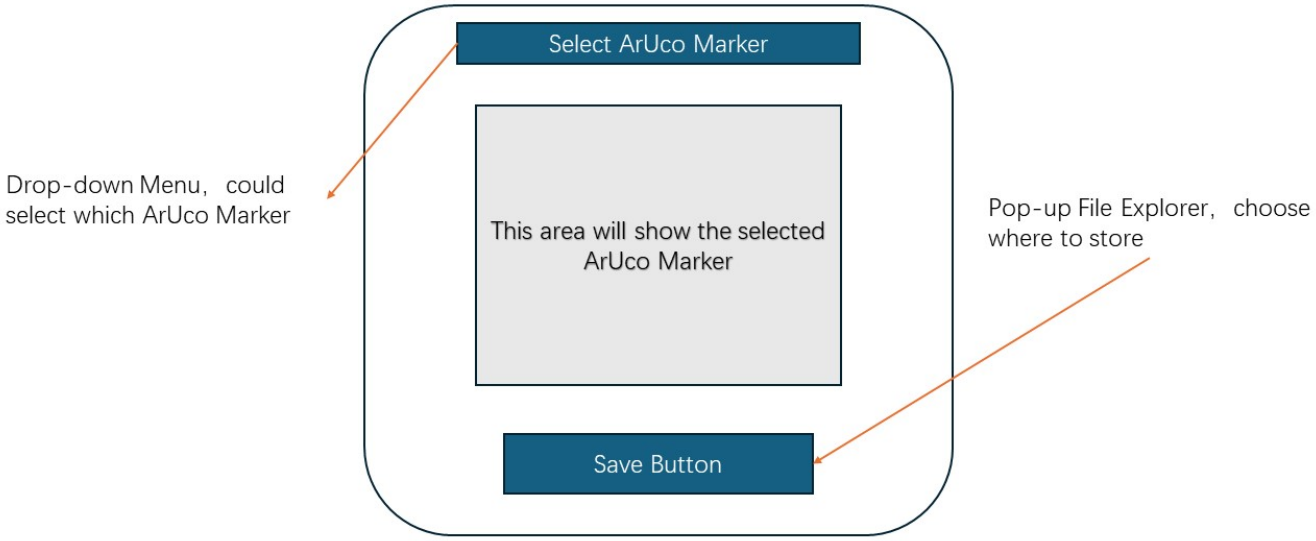

Figure 3: Design of ArUco Generator

Figure 4: ArUco Generator

The ArUco Generator interface allows users to select different ArUco markers from a predefined list and visualize them in real-time. The interface provides a clear preview of the selected marker and allows users to save the marker as an image file for later use. The design is simple and user-friendly, with intuitive controls for easy marker customization.

You can first select the ArUco marker you want to use by launching the drop-down menu via "Select Aruco" and the selected ArUco marker will be displayed on the screen in real time. After that, you can click the "Save Marker" button to start the file explorer to save the marker, the file name will be set to the corresponding dictionary of the ArUco marker by default.

(c) **Screenshots of Settings and Functionality**

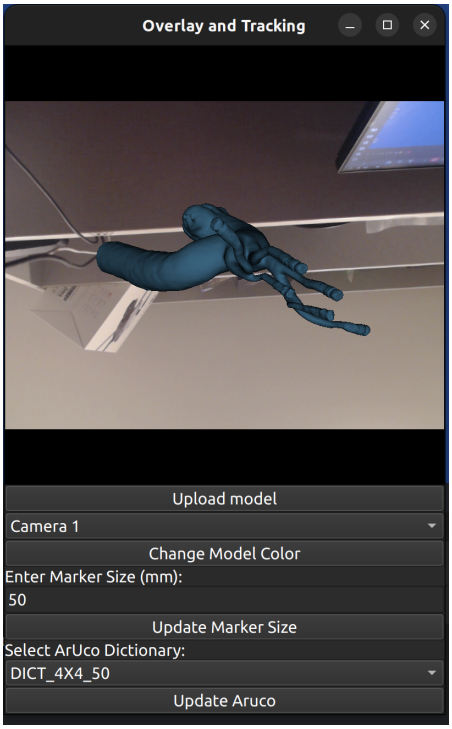

Figure 5: Overlay and Tracking Main Menu(With Overlay Activated)

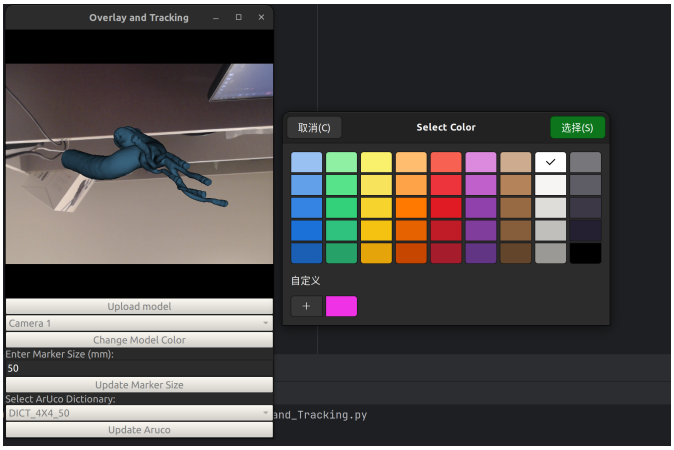

Figure 6: Model Color Change

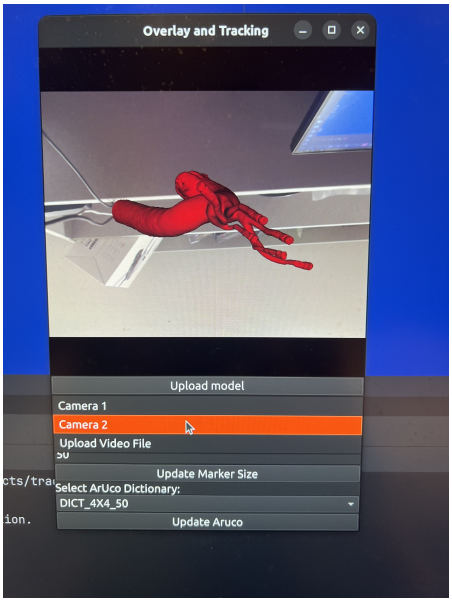

Figure 7: Video Source Switching

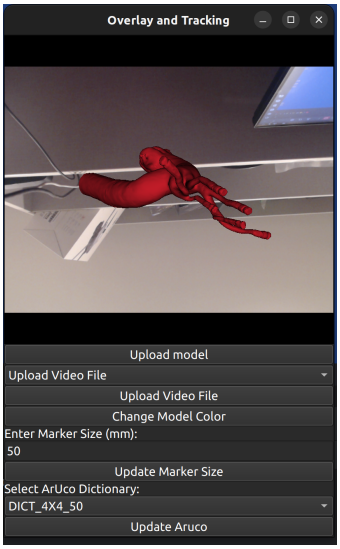

Figure 8: Color Changed and Video Upload Button

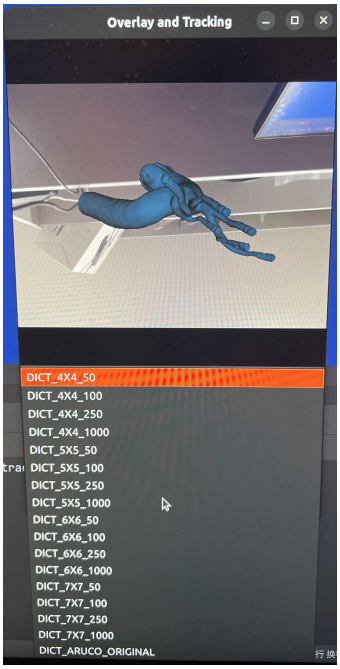

Figure 9: ArUco Dictionary Selection

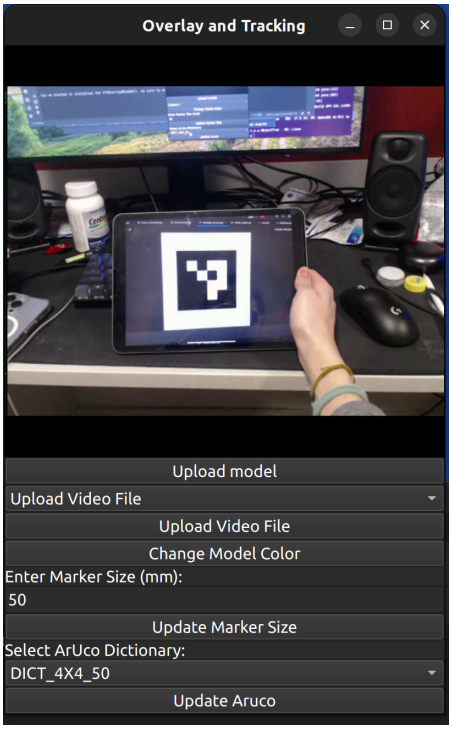

Figure 10: Video Upload

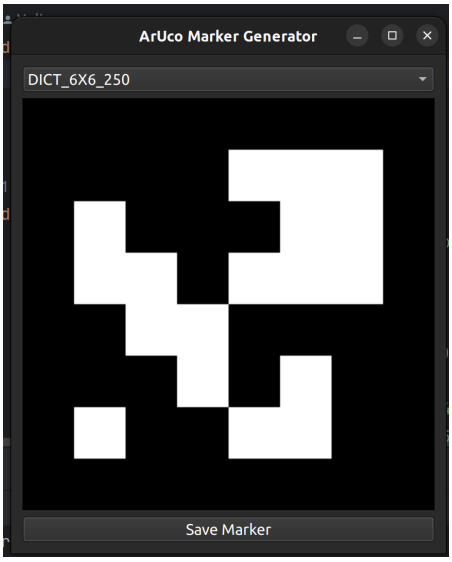

Figure 11: ArUco Dictionary Changed

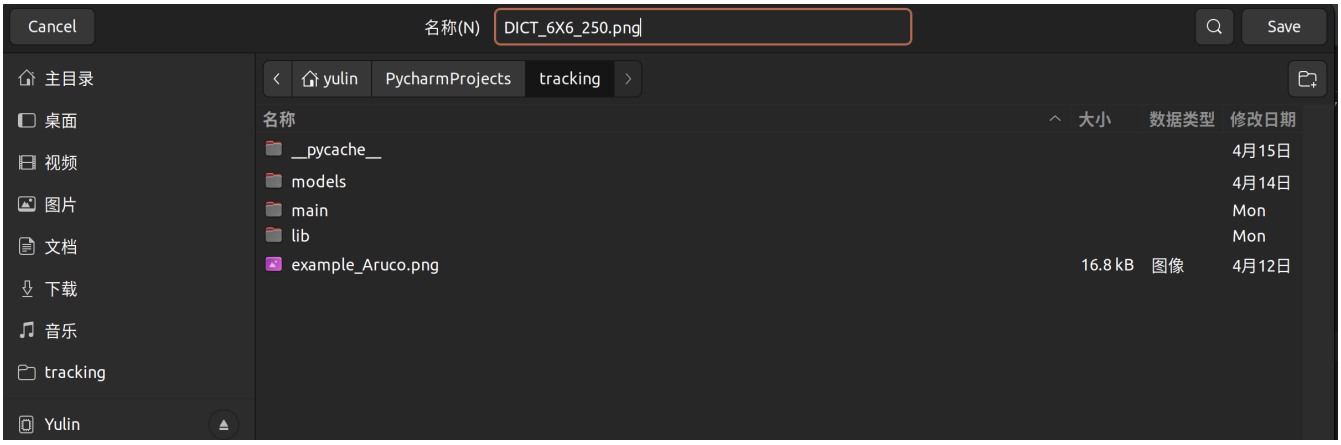

Figure 12: ArUco Marker Saved

3. Design Notation and Diagrams

(a) Use Case Diagrams

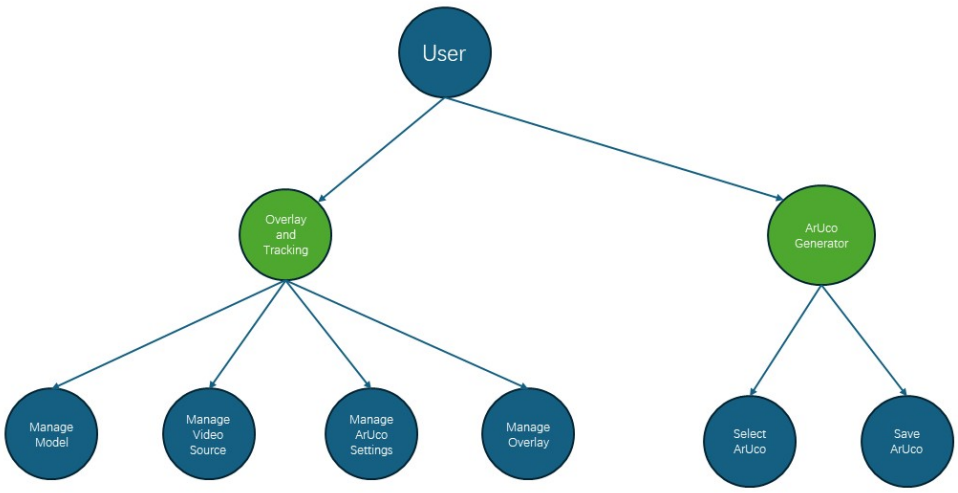

Figure 13: Use Case Diagrams

(b) **Data Flow Diagrams**

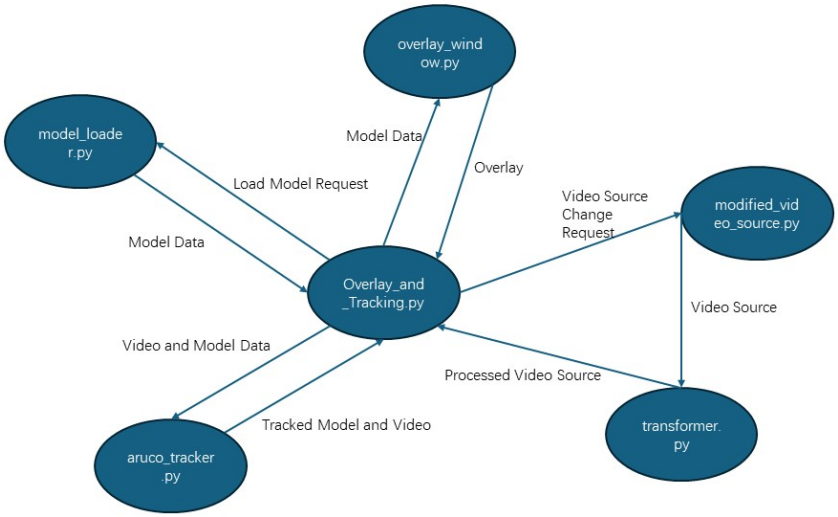

Figure 14: Data Flow Diagrams

2.1.2 Implementation

1. **Backend Implementation**

   (a) **Pre-processing for video capture and upload**

   This part is mainly controlled by *video_source.py* and is combined with a front-end Qt-based user interface to manage the core video processing functions. A robust video source management and processing system is implemented in *video_source.py* through the *TimestampedVideoSource* class and *VideoSourceWrapper* class. The system allows the user to interactively control the selection of video sources through a graphical interface, and it ensures accurate video stream processing.

   - *Video Source Configuration and Management:*
     The *TimestampedVideoSource* class uses the OpenCV library's, emphcv2.VideoCapture method to initialise the video capture from the camera number or file path. It can support basic video capture

functionality or provide the ability to validate the size of the video stream to ensure that the resolution is compatible with the system. If the resolution size provided by the setting is valid, the system sets the resolution accordingly. If the resolution is not valid, the system uses the camera's settings by default. If the requested resolution option is not supported by the camera, the system throws an error, which effectively prevents runtime problems that may result from unsupported video configurations and increases system stability and reliability.

- *Synchronisation and Timestamping:*
  The *TimestampedVideoSource* class also serves to synchronise the video stream with the modelled overlay, embedding a timestamp with the current date and time at each frame captured. Each time a video frame is successfully read, the *datetime.datetime.now()* method gets the current system time. This timestamp is then converted to a string or other format and stored or passed along with the video frame, ensuring that each frame has an exact timestamp. In Augmented Reality (AR) or Virtual Reality (VR) systems, quasi-timestamping this timestamp ensures alignment between the 3D model and the actual environment.With such a timestamping feature, *TimestampedVideoSource* can ensure the synchronisation of the video frames for the subsequent synchronisation of the model overlay.
- *Handles multiple video sources:*
  The *VideoSourceWrapper* class manages multiple video sources, allowing the addition of camera inputs or the uploading of video files, and verifying their existence and compatibility. A centralised method for releasing all video sources is provided in this class to ensure that resources that are no longer needed are managed and cleaned up in a timely manner. Each video source is encapsulated in an instance of the *TimestampedVideoSource* class, which allows each source to be controlled and processed independently. This feature ensures that the system load is at an appropriate level.
- *Retrieve and display video frames:*
  Retrieval of frames is achieved by system calls to the *read()* method of each video source, which is implemented through OpenCV's *cv2.VideoCapture.read()* . This method captures a single frame from the video source and checks to see if it was successfully captured. If successfully captured, the frame is tagged using the current timestamp, ensuring that each frame has an accurate time

reference. When processing multiple video sources, the system ensures synchronisation between frames. Synchronisation of frames is achieved by using the same frame rate setting for all sources and matching frames from different sources with timestamps. This allows for smoothness and synchronisation in the case of multiple video source inputs.

pseudocode for the *video_source.py*:

```
Class TimestampedVideoSource:
    Initialize(source_num_or_file, optional dims):
        Try:
            Open video source with cv2.VideoCapture(source_num_or_file)
        If source not opened:
            Raise RuntimeError("Failed to open Video camera")

        If dimensions provided:
            Validate and set video resolution:
                Check if dimensions are integers and >= 1
                Set resolution using cv2.VideoCapture properties
                If resolution setting fails:
                    Raise ValueError("Requested resolution not supported")

        Initialize empty frame array based on video dimensions
        Initialize timestamp

    Read():
        Capture frame and current time using cv2.VideoCapture.read()
        If frame captured successfully:
            Update timestamp with current datetime
        Return frame, timestamp

    Release():
        Release video source using cv2.VideoCapture.release()

    Update source(new_source):
        Release current source
        Reinitialize with new source

Class VideoSourceWrapper:
    Initialize():
        Create an empty list for storing video sources

    Add camera(camera_number, optional dims):
        Validate camera input (check if camera number is valid)
        Add camera source with optional dimensions to the list

    Add file(filename, optional dims):
```

```
        Validate if filename exists and is a valid file
        Add file source with optional dimensions to the list

    Get next frames():
        For each source in the list:
            If source is opened:
                Read frame from source
            Collect and return all frames

    Release all sources():
        For each source in the list:
            Release source
```

(b) **Image Processing Algorithms in Multi-Layer Video Rendering** In a video rendering system, video frames need to be dynamically managed to create complex visual effects, such as models and real-time video overlays in this project. This also involved real-time adjustments to the Alpha Blending[7] and video. A greyscale image with alpha blending of the RGBA stream was used to precisely control transparency and layering effects to enable the superimposition of a layer's frame (e.g. the model) onto the original frame[8]. Ensuring overlay accuracy and visual fidelity is crucial for applications such as augmented reality[9].

In addition, it is necessary to update and align the video frames to the appropriate layers by adjusting the data range according to the size of the incoming video frames. This incorporation of real-time processing improves the continuity of the video image by preventing visual interruptions caused by frame misalignment[10].

I also introduced Adaptive Exponential Smoothing into the multi-layer video rendering system (*Overlay_and_Tracking.py*). This enhances image processing, such as when processing video streams involving complex dynamic scenes[11]. By dynamically adjusting its smoothing parameter (Alpha), Adaptive Exponential Smoothing (AES) is able to more accurately adapt to changes in content within a video frame, such as lighting adjustments, scene switching, or object movement, and can reduce visual jitter and blurring due to rapid changes [11].

Compared with the traditional Exponential Moving Average, the adaptive feature of AES has a greater advantage. Exponential Moving Average (EMA), although fast in processing and low in computational cost, may not be able to adequately adapt its fixed smoothing parameters to real-time changes in the video content in the face of complex scene

variations, thus affecting the final image quality[12]–[14]. In a multi-layer rendering system, combining AES for real-time video transmission and dynamically adjusting the smoothing parameters according to the content differences between the previous and previous frames can maintain the continuity and naturalness of the visual effects, especially when dealing with moving objects and changing backgrounds. In addition, AES's also better handles scenes with large lighting variations, maintaining the balance of colors and shades of light and dark [15].

I have similarly experimented with Complex Exponential Smoothing, and while it excels in handling data with clear trends and cyclical variations, its application in video rendering systems may not be as straightforward and effective as AES. Because Complex Exponential Smoothing (CES) is designed to provide a more comprehensive understanding of the multiple influences on the data [16], [17], its use in non-predictive applications may lead to overly complex processing and increased computational burden.

Therefore, AES is ultimately used in video rendering systems to respond more directly to real-time changes in video content, reduce visual jitter, and improve the viewing experience.

To summarize, AES can improve image stability and visual quality, as well as enhance the system's responsiveness to environmental changes. By intelligently adjusting processing parameters, AES helps to ensure high efficiency while also adapting to visual jitter or lighting changes that may be encountered with ArUco marker tracking.

Code for the AES:

```
class SmoothedTransform:
    def __init__(self, alpha, min_alpha=0.01, max_alpha=0.9):
        """
        Initializes the adaptive smoothing transform class.

        Parameters:
        alpha (float): The initial smoothing factor.
        min_alpha (float): The minimum allowable value for alpha
        to prevent it from becoming too low.
        max_alpha (float): The maximum allowable value for alpha
        to prevent it from becoming too high.
        """
        self.alpha = alpha
```

```
        self.min_alpha = min_alpha
        self.max_alpha = max_alpha
        self.transform = None
    def adjust_alpha(self, new_transform):
        """
        Adjusts the smoothing factor alpha based on the
        difference between the new transform and the
        current transform to
        better adapt to recent data changes.

        Parameters:
        new_transform (float): The new data point used to
        update the transform.
        """
        if self.transform is not None:
        # Calculate the absolute difference between the current
        # and new transforms
        error = abs(new_transform - self.transform)
        # Dynamically adjust alpha based on the error,
        # inversely scaling it
        self.alpha = max(self.min_alpha, min(self.max_alpha,
        1 / (1 + error)))

    def update(self, new_transform):
        """
        Updates the current transform with a new data point using
        adaptive exponential smoothing.

        Parameters:
        new_transform (float): The new data point to incorporate
        into the smoothed data.

        Returns:
        float: The updated transform value.
        """
        if self.transform is None:
        # If no transform has been set yet, initialize it
        # with the new transform
        self.transform = new_transform
        else:
        # Adjust alpha based on the new data point
        self.adjust_alpha(new_transform)
        # Apply the adjusted alpha to compute
        # the new smoothed transform
        self.transform = self.alpha * new_transform +
        (1 - self.alpha) * self.transform
        return self.transform
```

(c) **Model Rendering and Overlay**

It is mainly implemented by *overlay_window.py*, in which the *VTKOverlayWindow* class combines the functionality of VTK, Qt and OpenCV for real-time video streaming and model overlay.

- *Multi-layer rendering settings for video and models:*
  When initialising the *VTKOverlayWindow* class, the VTK library's *vtkRenderWindow* and *vtkRenderer* are used to build a multi-layered rendering environment[18]. This multi-layer architecture allows independent control of content rendering on each layer, where the bottom layer is typically used to display live or pre-recorded video, while the top layer is used to add 3D models and other GUI elements[19], [20]. Such a layer setup ensures proper visual overlay between the rendered elements. The *SetNumberOfLayers* method is used to define the number of layers in the rendering window, and the *SetLayer* method is used to assign each renderer to a specific layer. This layering system ensures that the video stream and 3D models are correctly displayed and overlaid in the final output[18].

  Code for the multi-layer rendering settings:

```
# Create a rendering window and set multiple layers
renderWindow = vtk.vtkRenderWindow()
renderWindow.SetNumberOfLayers(3)

# Create renderers for each layer
videoRenderer = vtk.vtkRenderer()
videoRenderer.SetLayer(0)  # Bottom layer for video
modelRenderer = vtk.vtkRenderer()
modelRenderer.SetLayer(1)  # Middle layer for 3D models
uiRenderer = vtk.vtkRenderer()
uiRenderer.SetLayer(2)  # Top layer for UI components

# Add renderers to the rendering window
renderWindow.AddRenderer(videoRenderer)
renderWindow.AddRenderer(modelRenderer)
renderWindow.AddRenderer(uiRenderer)
```

- *Import and format conversion of video streams:*
  In order to implement VTK overlay, the video frames captured by OpenCV need to be converted into a format that VTK can handle. Here the video frames are converted by using VTK's *vtkImageImport* method. The conversion process requires setting the correct data types and sizes and updating the memory pointers to the video frames to ensure that the video stream data captured from OpenCV can be integrated into the VTK rendering process. This

ensures synchronisation and integrity of the video stream before and after conversion without causing video lag in VTK.[21], [22]

Code for the video stream conversion:

```
# Initialize video capture
cap = cv2.VideoCapture(0)

# Read a frame from OpenCV and import it into VTK
ret, frame = cap.read()
vtkImage = vtk.vtkImageData()
vtkImage.SetDimensions(frame.shape[1], frame.shape[0], 1)
vtkImage.SetScalarTypeToUnsignedChar()

# Create a VTK image importer and connect it to the image data
imageImporter = vtk.vtkImageImport()
imageImporter.SetInputData(vtkImage)
imageImporter.SetImportVoidPointer(frame.data, frame.nbytes)
imageImporter.Update()
```

- *Dynamic rendering and adjustment of 3D models:*
  Rendering of 3D models is achieved by using components such as VTK's *vtkActor* and *vtkPolyDataMapper*. These components support loading models from external files (e.g. STL format) and allow transformations such as translation, scaling, rotation, etc. to be applied to the model. These transformations of the model can be adjusted in real time to support dynamic user interaction and visual modifications[23]. The model can be rendered in the VTK overlay by using the *vtkPolyDataMapper* map the model data to geometry and set the actor using the *vtkActor* and *AddActor* method to add the model to the corresponding rendering layer. Then can use the *SetPosition* and *SetOrientation* method to adjust the model position and orientation to match the video[24].The function of modifying the position of the model will play an important role in section (f) Marker Detection and Tracking.

Code for the dynamic rendering and adjustment of 3D models:

```
# Load a 3D model from a file
modelReader = vtk.vtkSTLReader()
modelReader.SetFileName("model.stl")

# Map the model data to geometry
mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(modelReader.GetOutputPort())
```

```
8
9  # Create an actor for the model and add it to the renderer
10 actor = vtk.vtkActor()
11 actor.SetMapper(mapper)
12 modelRenderer.AddActor(actor)
13
14 # Adjust model position and orientation to match the video
15 # The position data can be varied depending on
16 # tracking data in (f) Marker Detection and Tracking
17 actor.SetPosition(10, 0, 0)
18 actor.SetOrientation(0, 90, 45)
```

(d) *Optimization of Visual Effects and Depth Processing:*
To enhance the visual quality of rendering, VTK's Depth Stripping technology can be enabled. Depth peeling is suitable for rendering transparent or semi-transparent objects and effectively solves visual artefacts and incorrect transparency problems during rendering. Through layer-by-layer rendering and compositing, Depth Stripping ensures visual correctness and is suitable for use in complex overlay or multi-object scenes.[24], [25] The *UseDepthPeelingOn* method is used to enable Depth Stripping, and the *SetMaximumNumberOfPeels* and *SetOcclusionRatio* methods are used to set the maximum number of peels and the occlusion ratio, respectively, to control the rendering quality and performance.

Code for Optimization of Visual Effects and Depth Processing:

```
1 # Enable depth peeling for better transparency handling
2 modelRenderer.UseDepthPeelingOn()
3 modelRenderer.SetMaximumNumberOfPeels(100)
4 modelRenderer.SetOcclusionRatio(0.1)
```

(e) *Precise Control of Camera and Viewing:*
During the implementation of model overlay using VTK, it is important to maintain an accurate overlay between the 3D model and the background video stream. When using different cameras, deviations in the model overlay may occur due to different camera parameters such as focal length, projection, and position. In order to make the model appear accurately at the right position in the background video, the parameters of the camera should be modified. By getting the camera's focal length, position and projection parameters to modify the variables in video processing, the viewpoint between the 3D model view and the video stream background can be consistently matched when using a camera with different characteristics. It might be necessary to addi-

tionally adjust the cropping range and field of view of the camera to ensure the correct range of visual output. The *vtkCamera* class is built to control the camera settings. Method like *SetPosition*, *SetFocalPoint*, and *SetViewUp* are used to adjust the camera's position, focus point, and view direction, respectively.[26], [27]

Code for Precise Control of Camera and Viewing:

```
# Configure the camera to match the video view
camera = modelRenderer.GetActiveCamera()
camera.SetPosition(0, 0, 100)
camera.SetFocalPoint(0, 0, 0)
camera.SetViewUp(0, 1, 0)
```

(f) **Marker Detection and Tracking**

- *Loading Calibration Data:*

Pseudocode for the Loading Calibration Data:

```
Function _load_calibration(textfile):
    Read 'projection_matrix' from
    '(e) Precise Control of Camera and Viewing'
    Read 'distortion' from
    '(e) Precise Control of Camera and Viewing'
    Return projection_matrix, distortion
```

- *Initializing Video Source and Tracker Configuration:*

Pseudocode for the Initializing Video Source and Tracker Configuration:

```
Method initialize_video_capture(video_source, configuration):
    If video_source is not 'none':
        self._capture = cv2.VideoCapture(video_source)
        If "capture properties" in configuration:
            For each property in configuration["capture properties"]:
                self._capture.set(cv2 property, value)
```

- *Load ArUco dictionary*

Pseudocode for the Load ArUco dictionary:

```
Function get_aruco_dictionary(configuration):
    aruco_dict =
    cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_ARUCO_ORIGINAL)
    Return aruco_dict
```

- *Real-Time Marker Detection using OpenCV:*

Pseudocode for the Real-Time Marker Detection and Data Capture:

```
Method detect_markers(frame):
    ar_dict = get_aruco_dictionary(configuration)
      # Default parameters
    parameters = cv2.aruco.DetectorParameters_create()
    marker_corners, marker_ids, _ =
    cv2.aruco.detectMarkers(frame, ar_dict, parameters=parameters)
    If marker_corners are not empty:
      # Optional: for visualization
        cv2.aruco.drawDetectedMarkers(frame, marker_corners)
    Return marker_corners, marker_ids
```

- *Estimate the pose of detected markers:*

Pseudocode for the Estimate the pose of detected markers:

```
Method estimate_pose(marker_corners, marker_ids):
    If marker_corners are not empty:
        For each marker in marker_corners:
            rvec, tvec, _ = cv2.aruco.estimatePoseSingleMarkers(marker_corners, m
            cv2.aruco.drawAxis(frame, self._camera_projection_matrix, self._camer
        Return rvec, tvec
```

- *Start Tracking and Outputting Marker Pose Data:*

Pseudocode for the Start Tracking and Outputting Marker Pose Data:

```
MMethod start_tracking():
    If self._state == "ready":
        self._state = "tracking"
        While self._state == "tracking":
            ret, frame = self._capture.read()
            If not ret:
                Break  # End tracking if video ends or error occurs
            marker_corners, marker_ids = detect_markers(frame)
            rvec, tvec = estimate_pose(marker_corners, marker_ids)
            Process and store marker pose data
```

- *Tracking State Management:*

Pseudocode for the Tracking State Management:

```
Method stop_tracking():
    If self._state == "tracking":
        self._state = "ready"
        self._capture.release()
```

- *Update the position of the model:*

Pseudocode for the Update the position of the model:

```
Method update_model_position:
    Read a frame from video source into 'image'
    If frame read is successful:
        Detect and follow ArUco markers in 'image'
            If markers detected:
                  Update model position based on marker pose
        Render the overlay window with updated model position
```

(g) **Model Color change**

(h) **ArUco Marker Settings**

(i) **ArUco Generator**

2. **Frontend Implementation**

2.2 Part2: Endovascular Intervention Simulation

2.2.1 Design

2.2.2 Implementation

3 Testing & Evaluation

4 Project Ethics

I have read and abide by the University's ethical guidelines[28]. The project did not involve direct interaction with human participants during the design, implementation or evaluation phases. An extensive review of the project scope and methodology confirmed that no personal data was collected, analyzed or used. In addition, all activities were within the scope of activities permitted by our ethical guidelines. It was verified with the project supervisor that no customized activities required separate ethical approval. Therefore, there are no other ethical issues involved in this project.

5 Conclusion & Future Work

5.1 Conclusion

5.2 Future Work

6 BCS Criteria & Self-Reflection

This section will be used to state that my project met the six outcomes expected by the Chartered Institute of Information Technology[29]. I will focus on illustrating an ability to self-manage a significant piece of work and the critical self-evaluation component.

6.1 An Ability to Apply Practical and Analytical Skills during the degree programme.

The project has demonstrated the practical and analytical skills I have learnt during my time at university. Throughout the degree I have gained a deeper understanding of programming languages such as Python, C# , Java and C and have gradually begun to experiment with them. The theoretical and practical foundations of these languages have been key in enabling me to achieve the complex functionality required for development and realisation projects. For example, in the Part1: Real-world Model Interaction and Tracking section of my project, which was written entirely in Python, there was a high level of theoretical and practical demand for the Python language. In my Part2: Endovascular Intervention Simulation, I needed to acquire and apply knowledge such as the application of Unity and the development and application of the C# language that I had learnt in my degree programme. These technical skills were acquired and refined through a careful learning process and were directly applied to the project, which dealt with the development of a real model interaction and tracking system and the development of a Unity-based Endovascular Intervention Simulation.

In developing Part1: Real-world Model Interaction and Tracking and Part2: Endovascular Intervention Simulation, I have also made extensive use of the Artificial Intelligence, Game Development and Computer Vision knowledge that I have learnt on my degree course.

For Part1: Real-world Model Interaction and Tracking, I utilised the techniques learnt in the Computer Vision course to process the images and used the OpenCV package usage learnt in the course to implement the tracking of the ArUco markers. By utilising the image processing and tracking capabilities of OpenCV, accurate

model interaction in complex environments is carried out in practice.

In Part2: Endovascular Intervention Simulation, I used my knowledge of game development to develop a Unity project, using Blender to modify and optimise the model, and applying Unity techniques to ensure that Rope interacts with the blood vessels.

This project, dedicated to the development of a realistic simulation used for endovascular interventions in a virtual reality environment, emphasised my ability to integrate practical skills and theoretical insights, demonstrating a deep understanding of the technical and theoretical aspects I have learnt during the course.

Overall, this project clearly demonstrated my ability to apply the analytical and practical skills acquired during my degree programme. It also demonstrated my understanding and use of complex programming techniques and frameworks, as well as my ability to use multidisciplinary knowledge to cross-cut problem solving. Through this project, I have accomplished the ability to translate my learning into practical applications in the real world.

6.2 Innovation and/or Creativity

There are some innovations in the field of medical simulation technology in my project, especially Part2: Endovascular Intervention Simulation, The aim of Part2 is to create one of the few open source endovascular intervention simulation projects in the field to make training tools more accessible to the medical community. The project combines traditional surgical simulation with augmented reality/virtual reality technology to make surgical simulation procedures visual and easy to practice. Compared to traditional simulations that are limited to 2D screen displays, this approach uses virtual reality to present surgical simulations in 3D space, which not only enhances the realism of the simulation and interactions, but also allows users to experience and understand the steps involved in the surgery more clearly by allowing them to interact with the simulated environment in a more intuitive and natural way.

The project uses an open source framework (Microsoft Mixed Reality Toolkit-Unity (MRTK)) and the integration of Augmented Reality/Virtual Reality (AR/VR) technology to provide a practical and innovative application for educational tools in the medical field.

6.3 Synthesis of Information, Ideas, and Practices

This project integrates development tools and theoretical principles from different fields to design the open source Endovascular Intervention Simulation to provide a convenient tool for medical surgery simulation or surgical training.

In the first part of the project, Real-world Model Interaction and Tracking, open source development tools such as OpenCV, VTK and SciKit-Surgery were used. The use of these powerful tools allows me to design user-friendly graphical interfaces or to enhance the model rendering capabilities and real-world model tracking capabilities of my application. OpenCV provides a rich set of image processing tools to help me perform complex image processing and tracking, while VTK and SciKit-Surgery provide powerful tools for medical impacts, such as multilayer image rendering and overlays. This can help me combine tracking capabilities with augmented reality to create a more interactive virtual reality system for users. This section combines knowledge, tools and ideas from various fields to create a high quality solution.

Part 2: Endovascular Intervention Simulation Developing an application on Microsoft Hololens using MRTK translates the theoretical knowledge I have learnt in my school course such as C# and Unity development into a practical solution. The development utilised model modification and knowledge related to Unity development, C# development, etc. to transform Endovascular Intervention Simulation, which is traditionally limited to a flat display, into a virtual reality simulation with immersive, interactive features. This part of the development demonstrates how AR application development techniques and Unity development can be combined to provide a virtual reality surgical simulation with multiple functionalities.

Both parts of the project exemplify how technical and theoretical knowledge from the fields of computer vision, artificial intelligence and software development can be applied to create effective and innovative medical training tools.

6.4 Meeting a Real Need in a Wider Context

Both parts of the project, Part1: Real-world Model Interaction and Tracking and Part2: Endovascular Intervention Simulation, address some of the broader needs in the medical field.

For Part1: Real-world Model Interaction and Trac, current market systems usually lack user-friendly graphical interfaces, and features such as model colour, selection of different ArUco markers, and resizing are lacking or incomplete, which can cause

some degree of difficulty for users. For Part1 the project adds a graphical interface and provides a variety of modifiable parameters to optimise these shortcomings, making the software less difficult to use and better adapted to the needs of a wide range of scenarios.

Part2 considers the lack of open source endovascular intervention simulation in the market and the fact that most existing simulation tools are limited to 2D planar presentation and cannot meet the complex 3D visual and operational needs. The aim is to develop an open source platform that supports immersive 3D simulation, AR/VR and other functions. The system can support 3D simulation in AR/VR (deployed in Microsoft Microsoft Hololens), and by lowering the barrier to use through more intuitive and simple controls, it can be used in the future to allow healthcare professionals or non-professionals alike to experience or learn surgical skills. This simulation tool can not only be used for professional training, but also meets the need for telemedicine services that can provide remote diagnosis and treatment in the future.

Overall, it is planned that these two components will be combined in future work, which can meet the needs for simulation of surgical training simulation for simplicity, remote operation, and 3D highly experiential simulation. In the future it may be possible to expand into more areas to meet a wider range of needs, such as providing an immersive experience of Endovascular Intervention Simulation for lay people.

6.5 An Ability to Self-Manage a Significant Piece of Work

In my project I demonstrated the ability to self-manage a significant piece of work, but it was partially flawed. The project consisted of two widely differing parts, which added to the difficulty of managing the project as a whole, and in order to keep both of the major parts of the project accurately planned and executed, I used a variety of tools and methods to ensure that the project ran smoothly.

Firstly, I used a number of time management tools to map out the timeline of the project, including key milestones, time required, and deadlines for each progress module. I created some Gantt charts and schedules to help me monitor the progress of the project and try my best to make sure that the tasks in each phase are completed on time. In this way I could get a clear picture of the overall progress of the project and adjust the plan as much as possible in time to cope with possible delays. However, as this was my second time working on a larger volume project (the last time was COMP208 Group Project), I was not able to be very perfect in creating the schedule and Gantt chart, and some mistakes were made.

Secondly, in terms of project management, I used GitHub to maintain version control of the project and writing between team members. I used GitHub to effectively track code updates and backups, as well as to enable team members to view the latest progress of the project in real time and provide feedback. GitHub's version control and backup features have many times saved errors caused by mistakes, effectively avoiding many accidents. Using this open source platform has helped me to manage a major task and increase the efficiency of multi-person collaboration.

In addition, in order to control the development progress and quality, I also hold weekly progress meetings with my team members and supervisor to report the progress of this week's work and discuss and plan the next work. In these meetings, I can get sufficient feedback to help me modify and optimise my previous work, and make reasonable planning and arrangement for the next work. This regular reporting and discussion has ensured that the project has developed according to the set objectives.

I also focus on stage-by-stage problem analysis and risk management during project implementation. Whenever the project progresses to a certain stage, I will review the previous work, check and improve any possible problems in the completed work, and make sure that the previous work is accurate before proceeding to the next stage1. This is a good way to ensure that I make fewer mistakes when managing a large volume of work.

Whilst I have adopted a variety of methods during the project management process to ensure that the project is executed efficiently and to a high quality as planned, I still have some shortcomings in my ability to self-manage a significant piece of work. For example, although I produced a Gantt chart and schedule to monitor the progress of the project, at the beginning of the project I did not properly consider the time required for some parts and the difficulties I may have encountered, for example, I encountered great difficulties in carrying out the initial design and import of the Rope in Part2: Endovascular Intervention Simulation, etc., and the rate of progress was not as fast as expected. This resulted in the project progressing at a much slower pace than expected and led to the project being put on hold for some time. In addition, although we had weekly meetings, some of them were of minimal effect. These meetings usually took place when the project was experiencing some major difficulties, and in these meetings the solutions to the problems and the planning for the next phase of the project progress were not discussed very effectively.

In addition, I had problems with teamwork when using GitHub for project management. Poor documentation, different operating systems used by each member, conflicting versions of various software packages, and GitHub's file size limitations for uploading files caused many difficulties for team members when sharing through GitHub.

Overall, this project demonstrated my ability to self-manage a significant piece of work, but it also demonstrated my shortcomings in some of these areas. This project gave me a great opportunity to optimise my ability to manage projects, such as time planning skills and communication with team members, as well as making me realise what I need to learn and improve in project management.

6.6 Critical Self-Evaluation of the Process

In my projects, Part1: Real-world Model Interaction and Tracking and Part2: Endovascular Intervention Simulation, although both have been accomplished, there have been many challenges and difficulties in the development process, and so far there have been some shortcomings. Through in-depth critical self-evaluation, I was able to comprehensively analyse the successes and shortcomings of the project, as well as gain experience and lessons learned.

First of all, the functionality of my project Part1 is relatively complete, which can effectively implement real-time model overlay and ArUco marker tracking, and complete the basic function of my plan. At the same time, I also added extra features such as ArUco icon parameter tuning and ArUco icon generator. However, there are some technical limitations in this part, mainly in platform compatibility. Currently, the system only runs on the Linux platform and has not yet implemented support for other operating systems such as Windows or macOS, nor has it been able to complete deployment on VR/AR devices such as HoloLens. This limitation may have impacted the project's widespread adoption. In this regard, I believe that my Part1 met the requirements of my plan, but still needs to be improved and extended in terms of compatibility.

For Part2, the development process encountered significant technical challenges, especially during the stages of designing the Rope and developing the method of interaction between the Rope and the vessel wall. These technical issues led to delays in the development progress and the final product implemented only the most basic functionality and did not achieve the level of Rope-vessel wall interaction required at the beginning of the design. This difficulty stemmed from my lack of skill in using development tools such as Unity and Blender, and my un-

derestimation of the complexity of the interaction logic of physical simulation and model interaction. Nonetheless, the development process has greatly strengthened my technical skills in 3D modelling, Unity development and physics simulation simulation.

By critically reflecting on and analysing these issues, I realised that I should plan better in the upfront technical assessment and time management phases when undertaking future project management. This includes analysing in detail the technical difficulty and time required for each development phase during the project planning stage, as well as being prepared in advance to deal with unforeseen circumstances as they occur. In addition, it is also important to communicate more with team members and supervisors at the technical level during the project to accelerate the speed of breaking through the development challenges.

Overall, the development process of this project was full of difficulties and challenges, but it also strengthened my technical level, project management skills and the ability to solve unknown problems. Through this project, I was able to improve my professional skills in a variety of different technical areas, as well as develop my ability to effectively self-manage and work in a team on projects with complex environments. Through this critical self-assessment and reflection, I was able to better identify and improve on my shortcomings in my work and prepare myself for the greater challenges I may face in the future.

References

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] S. Thompson, T. Dowrick, M. Ahmad, *et al.*, "SciKit-Surgery: Compact Libraries for Surgical Navigation," *International journal of computer assisted radiology and surgery*, vol. 15, no. 7, pp. 1075–1084, 2020. DOI: 10.1007/s11548-020-02180-5.

[3] The Qt Company, *Qt - cross-platform software development for embedded and desktop*, [Online; accessed 29-April-2024], 2024. [Online]. Available: https://www.qt.io.

[4] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit (4th ed.)* Kitware, 2006, ISBN: 978-1-930934-19-1.

[5] M. Sun and S. Wu, "A software development of dicom image processing based on qt, vtk and itk," in *2013 IEEE International Conference on Medical Imaging Physics and Engineering*, 2013, pp. 231–235. DOI: 10.1109/ICMIPE.2013.6864541.

[6] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, 590–596 vol. 2. DOI: 10.1109/CVPR.2005.74.

[7] Wikipedia contributors, *Alpha compositing — Wikipedia, the free encyclopedia*, [Online; accessed 1-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Alpha_compositing&oldid=1220212606.

[8] C. Liu, Y. Liang, and W. Wen, "Fire image augmentation based on diverse alpha compositing for fire detection," in *2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2022, pp. 1–6. DOI: 10.1109/CISP-BMEI56279.2022.9979846.

[9] A. Settimi, J. Gamerro, and Y. Weinand, "Augmented-reality-assisted timber drilling with smart retrofitted tools," *Automation in Construction*, vol. 139, p. 104 272, 2022, ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2022.104272. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580522001455.

[10] S. Wang and J. Zhang, "Research and implementation of real-time render optimization algorithm based on gpu," *Journal of Physics: Conference Series*, vol. 2136, no. 1, p. 012 059, Dec. 2021. DOI: 10.1088/1742-6596/2136/1/012059. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2136/1/012059.

[11] K. Dang, J. Yang, and J. Yuan, "Adaptive exponential smoothing for online filtering of pixel prediction maps," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3209–3217. DOI: 10.1109/ICCV.2015.367.

[12] S. Ekern, "Adaptive exponential smoothing revisited," *The Journal of the Operational Research Society*, vol. 32, no. 9, pp. 775–782, 1981, ISSN: 01605682, 14769360. [Online]. Available: http://www.jstor.org/stable/2581393 (visited on 05/01/2024).

[13] J. W. Taylor, "Smooth transition exponential smoothing," *Journal of Forecasting*, vol. 23, no. 6, pp. 385–404, 2004. DOI: https://doi.org/10.1002/for.918. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/

for.918. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.918.

[14] Investopedia, *Exponential moving average (ema) - definition*, https://www.investopedia.com/terms/e/ema.asp, Accessed: 2024-05-01, 2024.

[15] P.-L. Hsieh, C. Ma, J. Yu, and H. Li, "Unconstrained realtime facial performance capture," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1675–1683. DOI: 10.1109/CVPR.2015.7298776.

[16] Vinay, *Complex exponential smoothing (ces)*, https://medium.com/@vinay1996/complex-exponential-smoothing-ces-63d1c6ddfc78, Accessed: 2024-05-01, 2018.

[17] I. Svetunkov, N. Kourentzes, and J. K. Ord, "Complex exponential smoothing," *Naval Research Logistics (NRL)*, vol. 69, no. 8, pp. 1108–1123, 2022. DOI: https://doi.org/10.1002/nav.22074. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.22074. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.22074.

[18] J. Tan, J. Chen, Y. Wang, L. Li, and Y. Bao, "Design of 3d visualization system based on vtk utilizing marching cubes and ray casting algorithm," in *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 02, 2016, pp. 192–197. DOI: 10.1109/IHMSC.2016.153.

[19] S. Sun, J. He, and L. Ma, "3d visualization system for medical image based on vtk and ok series image board," in *2011 International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 951–954. DOI: 10.1109/CSSS.2011.5974642.

[20] Y. Wang, W. Wan, X. Zhou, S. Mao, and HuiLi, "Local transparency technique for heart model from vtk," in *IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2011)*, 2011, pp. 139–142. DOI: 10.1049/cp.2011.0863.

[21] vtk2jbean, *Vtkimageimport documentation*, https://jbeanvtk.sourceforge.net/apidocs/vtk/vtkImageImport.html, Accessed: 2024-05-05.

[22] *Vtkimageimport class reference*, https://vtk.org/doc/nightly/html/classvtkImageImport.html, Accessed: 2024-05-05.

[23] L. Xiaoqi, J. Dongzheng, Z. Baohua, and R. Xiaoying, "Study and implementation for interactive cutting of 3d medical image based on vtk," in *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, vol. 2, 2011, pp. 1351–1354. DOI: 10.1109/CSQRWC.2011.6037214.

[24] W. Yang, Z. Zeng, and Y. Bai, "3d reconstruction system on coarse aggregate microfabric based on vtk," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 4, 2009, pp. 85–88. DOI: 10.1109/ICICISYS.2009.5357714.

[25] A. Vacanti and D. DeMarle, *VTK Technical Highlight: Dual Depth Peeling*, https://www.kitware.com/vtk-technical-highlight-dual-depth-peeling/, Accessed: 2024-05-05, Oct. 2016.

[26] *Vtk examples for camera in c++*, https://examples.vtk.org/site/Cxx/Visualization/Camera/, Accessed: 2024-05-05.

[27] *Documentation for vtkcamera*, https://www.geologie.uni-freiburg.de/root/manuals/vtkman/vtkCamera.html, Accessed: 2024-05-05.

[28] University of Liverpool, *Comp390 honours year project 2023-24 ethical guidance*, Accessed: 2023-11-03, 2023. [Online]. Available: https://student.csc.liv.ac.uk/internal/modules/comp390/2023-24/ethics.php.

[29] BCS, *Guidelines on course accreditation: Information for universities and colleges*, Accessed: 2024-05-01, Jan. 2020. [Online]. Available: https://www.bcs.org/media/11ofljxo/course-accreditation-guidelines.pdf.

[30] "Aruco markers." (2021), [Online]. Available: https://fab.cba.mit.edu/classes/865.21/people/zach/arucomarkers.html (visited on 04/29/2024).

[31] Blender Foundation, *Blender*, https://www.blender.org, Accessed on: 2024-05-01, 2024. [Online]. Available: https://www.blender.org.

[32] Wikipedia contributors, *Blender (software) — Wikipedia, the free encyclopedia*, [Online; accessed 1-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Blender_(software)&oldid=1220122561.

[33] Wikipedia contributors, *Github — Wikipedia, the free encyclopedia*, [Online; accessed 3-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=GitHub&oldid=1221852262.

[34] Wikipedia contributors, *Microsoft hololens — Wikipedia, the free encyclopedia*, [Online; accessed 3-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Microsoft_HoloLens&oldid=1221749039.

[35] Microsoft, *Microsoft hololens*, https://www.microsoft.com/opencv_libraryen-us/hololens, Accessed: 2024-05-01, 2024.

[36] Wikipedia contributors, *Qt (software) — Wikipedia, the free encyclopedia*, [Online; accessed 29-April-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Qt_(software)&oldid=1219937369.

[37] Wikipedia contributors, *Opencv — Wikipedia, the free encyclopedia*, [Online; accessed 29-April-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=OpenCV&oldid=1208982530.

Glossary

Adaptive Exponential Smoothing Adaptive exponential smoothing models are designed to improve performance by letting the smoothing parameter vary according to the most recent forecasting accuracy. [12], [13] 17

AES Adaptive Exponential Smoothing 17

Alpha Blending In computer graphics, alpha compositing or alpha blending is the process of combining one image with a background to create the appearance of partial or full transparency. It is often useful to render picture elements (pixels) in separate passes or layers and then combine the resulting 2D images into a single, final image called the composite. Compositing is used extensively in film when combining computer-rendered image elements with live footage. Alpha blending is also used in 2D computer graphics to put rasterized foreground elements over a background.[7] 17

Aruco Marker ArUco markers are 2D binary-encoded fiducial patterns designed to be quickly located by computer vision systems. ArUco marker patterns are defined by a binary dictionary in OpenCV, and the various library functions return pattern IDs and pose information from scanned images.[30] 3

Blender Blender is a free and open-source 3D computer graphics software tool set used for creating animated films, visual effects, art, 3D-printed models, motion graphics, interactive 3D applications, virtual reality, and, formerly, video games. Blender's features include 3D modelling, UV mapping, texturing, digital drawing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animation, match moving, rendering, motion graphics, video editing, and compositing[31], [32]. 27

CES Complex Exponential Smoothing 18

Complex Exponential Smoothing Complex exponential smoothing is a time series forecasting method that combines exponential smoothing with trend and seasonality. It is a variant of the standard exponential smoothing method, which is a simple technique for smoothing out data by using a weighted average of past observations.[16] 18

EMA Exponential Moving Average 17

Exponential Moving Average An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average simple moving average (SMA), which applies an equal weight to all observations in the period.[14] 17

GitHub GitHub is a developer platform that allows developers to create, store, manage and share their code. It uses Git software, providing the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. Headquartered in California, it has been a subsidiary of Microsoft since 2018. It is commonly used to host open source software development projects. As of January 2023, GitHub reported having over 100 million developers and more than 420 million repositories, including at least 28 million public repositories. It is the world's largest source code host as of June 2023.[33] 30

Microsoft Hololens Microsoft HoloLens is an augmented reality (AR)/mixed reality (MR) headset developed and manufactured by Microsoft. HoloLens runs the Windows Mixed Reality platform under the Windows 10 operating system. Some of the positional tracking technology used in HoloLens can trace its lineage to the Microsoft Kinect, an accessory for Microsoft's Xbox 360 and Xbox One game consoles that was introduced in 2010[34], [35] 28, 29

MRTK Microsoft Mixed Reality Toolkit-Unity 27, 28

OpenCV OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and licensed as free and open-source software under Apache License 2. Starting in 2011, OpenCV features GPU acceleration for real-time operations.[36] 3

Qt Qt (pronounced ”cute” or as an initialism) is cross-platform application development framework for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms such as Linux, Windows, macOS, Android or embedded systems with little or no change in the underlying codebase while still being a native application with native capabilities and speed.[37] 3

VTK The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and 2D plotting. It supports a wide variety of visualization algorithms and advanced modeling techniques, and it takes advantage of both threaded and distributed memory parallel processing for speed and scalability, respectively.[4] 3