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

UNIVERSITY OF
LIVERPOOL

COMP390 2023/24

Computer Vision and AR for Endovascular Intervention

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Contents

1 Introduction & Background **3**

2 Design & Implementation **3**

2.1 Part1: Real-world Model Interaction and Tracking 3

2.1.1 Design 3

2.1.2 Implementation 8

2.2 Part2: Endovascular Intervention Simulation 8

2.2.1 Design 8

2.2.2 Implementation 8

3 Testing & Evaluation **8**

4 Project Ethics **8**

5 Conclusion & Future Work **9**

5.1 Conclusion 9

5.2 Future Work 9

6 BCS Criteria & Self-Reflection **9**

6.1 An Ability to Apply Practical and Analytical Skills 9

6.2 Innovation and/or Creativity 9

6.3 Synthesis of Information, Ideas, and Practices 9

6.4 Meeting a Real Need in a Wider Context 9

6.5 An Ability to Self-Manage a Significant Piece of Work 9

6.6 Critical Self-Evaluation of the Process 9

Glossary **11**

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 also VTK[4]. OpenCV and SciKit-Surgery libraries help me process images and track Aruco Marker[5] 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. **Data Processing and Tracking Algorithms**

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

(b) **Feature Data Structures**

(c) **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[6] 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[7]. Ensuring overlay accuracy and visual fidelity is crucial for applications such as augmented reality[8].
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[9].
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[10]. 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 [10].
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[11]–[13]. 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 [14].
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 straight-

forward and effective as AES. Because Complex Exponential Smoothing (CES) is designed to provide a more comprehensive understanding of the multiple influences on the data [15], [16], 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:

```
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
```

   (d) **Marker Detection and Tracking**

   (e) **Model Positioning and Rendering**

   (f) **Optimization Techniques**

   (g) **Model Color change**

   (h) **ArUco Generator**

3. **User Interface Design**

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

   (b) **ArUco Generator**

   (c) **Screen Mockups, Sketches, and Screenshots**

4. **Design Notation and Diagrams**

   (a) **Use Case Diagrams**

   (b) **Interaction Diagrams**

   (c) **Data Flow Diagrams**

2.1.2 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[17]. 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

6.1 An Ability to Apply Practical and Analytical Skills

6.2 Innovation and/or Creativity

6.3 Synthesis of Information, Ideas, and Practices

6.4 Meeting a Real Need in a Wider Context

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

6.6 Critical Self-Evaluation of the Process

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. 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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).

[12] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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. [11], [12] 5

AES Adaptive Exponential Smoothing 5

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.[6] 5

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.[18] 3

CES Complex Exponential Smoothing 6

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.[15] 5

EMA Exponential Moving Average 5

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.[13] 5

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.[19] 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.[20] 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