Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Fall 2024

MosaicMovement
In/E Motion

Derrick Perry
Sophia Dao
John Calma
Asad Mirza

Revision History

Revision	Date	Author(s)	Description
0.1	9.20.2024	DP, JC, SD, AM	document creation
0.2	9.23.2024	DP, JC, SD, AM	complete draft

CONTENTS

List of Figures

List of Tables

1 Introduction

The In/E Motion project, developed by the MosaicMovement team, aims to integrate advanced motion tracking technology with real-time animation projection to enhance live performances. By capturing detailed movement data from performers and participants using high-resolution Intel RealSense cameras and motion tracking software, the system processes this data to generate dynamic animations projected onto the performance space. This creates an immersive and interactive experience, enhancing the visual appeal of performances in theater, dance, and live art installations.

2 System Overview

The In/E Motion system is in three distinct layers Input, Translation, and Output each responsible for specific functionalities essential to the overall performance of the system.

- Input Layer: This layer captures raw motion data using Intel RealSense D435 cameras. The data is then transmitted to the Translation Layer for processing.
- Translation Layer: Here, the raw data is processed by the motion tracking software, analyzed for patterns and interactions, and prepared for rendering. This layer ensures that the captured movements are translated into meaningful animations.
- Output Layer: The final processed animations are projected onto the performance space using high-resolution projectors. The Output Layer also facilitates interaction with participants, adjusting animations in real-time based on their movements.

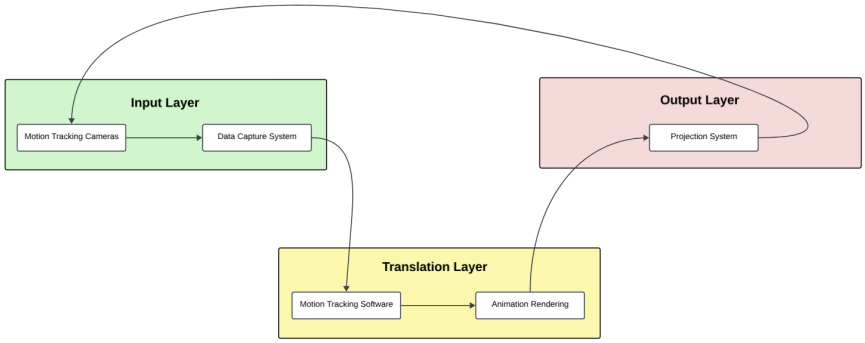

Figure 1: System architecture

3 Input Layer Subsystems

The Input Layer is the foundational component of the In/E Motion system, responsible for capturing raw motion data that will be processed and transformed into real-time animations. This layer includes high-resolution motion tracking cameras and sophisticated data capture software, both of which are crucial for ensuring accurate and synchronized data collection.

3.1 Layer Hardware

The Input Layer hardware consists primarily of four Intel RealSense D435 cameras, which are used for motion tracking. These cameras are strategically placed around the performance space to ensure complete coverage and capture of performer and participant movements. The hardware also includes calibration tools to ensure accurate alignment of the cameras.

3.2 Layer Operating System

The operating system that manages the Input Layer's hardware is a real-time operating system (RTOS) capable of handling multiple camera inputs with high reliability and minimal latency.

3.3 Layer Software Dependencies

- Nuitrack SDK [1]: Used for skeleton tracking and depth sensing, necessary for processing the raw data captured by the Intel RealSense cameras.
- Intel RealSense SDK: Provides the drivers and tools needed to interface with the cameras, including calibration and data acquisition functionalities.
- Camera Control Software: A custom application developed in C++ that manages camera operations such as start/stop recording, setting adjustments, and synchronization.

3.4 Motion Tracking Cameras

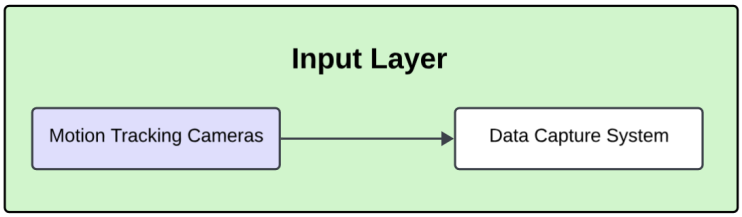

Figure 2: 4 Intel RealSense Cameras

3.4.1 Subsystem Hardware

The Motion Tracking Cameras subsystem consists of four Intel RealSense D435 cameras. These cameras capture high-resolution video at a required frame rate, ensuring detailed and accurate movement data. The subsystem also includes calibration tools for ensuring proper camera alignment.

3.4.2 Subsystem Operating System

The cameras and their control systems operate on a specialized RTOS that ensures the cameras are managed with high precision, including real-time synchronization.

3.4.3 Subsystem Software Dependencies

- Intel RealSense SDK: For camera interface and control.
- Nuitrack SDK [1]: For processing the raw data into actionable skeleton tracking and depth information.

3.4.4 Subsystem Programming Languages

The primary programming language used in this subsystem is C++, which is employed to develop the camera control software, ensuring efficient handling of camera operations and real-time data processing.

3.4.5 Subsystem Data Structures

- Frame Buffer: Stores raw video data temporarily before it is processed.
- Calibration Data Structures: Hold the calibration settings and alignment data for each camera.
- Synchronization Timestamps: Time-stamps for ensuring that data from different cameras are aligned correctly in time.

3.4.6 Subsystem Data Processing

- Calibration: Ensuring all cameras are aligned to capture synchronized and accurate data.
- Data Capture: High-resolution video data is captured and buffered.
- Synchronization: The captured data is time-stamped to ensure it can be synchronized with data from other cameras, enabling accurate multi-camera data processing.

3.5 Data Capture System

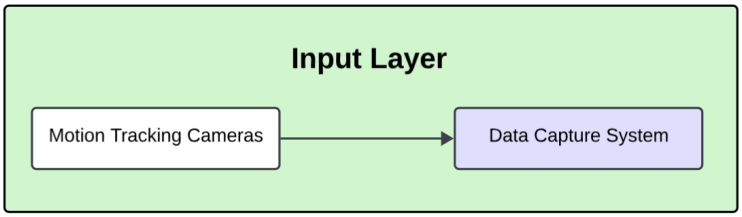

Figure 3: RealSenses' VPU and PC's GPU

3.5.1 Subsystem Hardware

The Data Capture System includes the computing hardware required to handle the input from multiple Intel RealSense cameras. The cameras are equipped with Movidius Myriad X, a vision processing unit. This includes high-performance processors and GPUs that can process high-resolution video data in real-time.

3.5.2 Subsystem Operating System

The Data Capture System operates on a high-performance Linux distribution optimized for real-time data processing, capable of handling the demands of simultaneous multi-camera input.

3.5.3 Subsystem Software Dependencies

- Nuitrack SDK [1]: To process the incoming data from the cameras for skeleton tracking.
- Intel RealSense SDK: To manage the camera interfaces and ensure accurate data capture.
- Custom Synchronization Module: Ensures that data from different cameras is aligned in time.

3.5.4 Subsystem Programming Languages

- C++: For developing the core data capture and synchronization functionalities.
- Python: For scripting and automation tasks related to data management and preprocessing.

3.5.5 Subsystem Data Structures

- Synchronized Video Stream: A data structure that holds video data from multiple cameras, aligned by timestamps.
- Metadata for Calibration: Stores the calibration settings and synchronization data for each video stream.

3.5.6 Subsystem Data Processing

- Synchronization: Aligning video data from multiple cameras using time-stamps to ensure the data can be processed as a single cohesive stream.
- Data Integration: Combining the synchronized data into a single stream that can be used for further analysis in the Translation Layer.

4 Translation Layer Subsystems

The Translation Layer processes the captured motion data, preparing it for animation rendering. This layer includes the Motion Tracking Software, Data Analysis, and Animation Rendering subsystems, each responsible for specific tasks in transforming raw data into interactive animations.

4.1 Layer Hardware

The Translation Layer primarily relies on the computing hardware that runs the software responsible for processing the captured motion data. This hardware includes high-performance CPUs and GPUs, which are necessary for real-time data processing, pattern recognition, and rendering tasks. The hardware must be capable of handling the intensive computational requirements of both the Nuitrack SDK and Unity engine.

4.2 Layer Operating System

The operating system managing the Translation Layer is a Windows environment. The OS will provide a stable platform for high-performance processing and real-time data analysis, ensuring low latency in data handling and graphics rendering.

4.3 Layer Software Dependencies

- Nuitrack SDK: This SDK is crucial for skeletal tracking and depth sensing, processing the raw data captured from the Intel RealSense cameras.
- Unity Engine [2]: Unity is used to create, render, and manage the animations and interactive environments based on the processed data. It integrates seamlessly with the Nuitrack SDK for real-time rendering and interaction.

4.4 Motion Tracking Software - Nuitrack

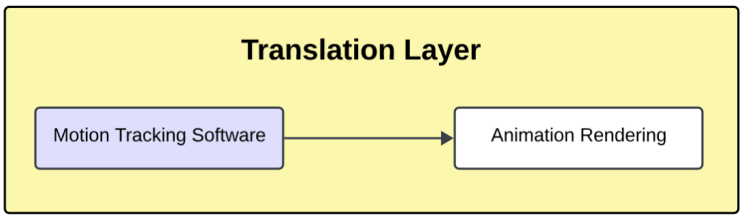

Figure 4: Nuitrack SDK

4.4.1 Subsystem Hardware

The Nuitrack SDK operates on the same high-performance computing hardware described in the Layer Hardware section. This includes powerful CPUs and GPUs required to process the depth data and track skeletal movements in real-time.

4.4.2 Subsystem Operating System

The Nuitrack software development kit will operate in a Windows environment.

4.4.3 Subsystem Software Dependencies

- Intel RealSense SDK: Necessary for interfacing with the cameras and retrieving the depth and skeletal data.

- OpenGL/DirectX: Depending on the platform, these graphics libraries are used for rendering the skeletal tracking data for verification and debugging purposes.
- Unity Integration Package [1]: A bridge that allows the Nuitrack SDK to communicate directly with Unity for real-time updates and interactions.

4.4.4 Subsystem Programming Languages

The Nuitrack SDK is primarily interfaced with using C++ for performance-critical components and C# when integrated with Unity, allowing for smooth communication between the tracking system and the rendering engine.

4.4.5 Subsystem Data Structures

- Skeleton Data Structure: Stores information about the tracked skeletons, including joint positions and orientations.
- Depth Map: A 2D array containing depth information captured by the cameras, which is processed to identify skeletal structures.
- User Data: Tracks individual users, differentiating between multiple participants in the tracking space.

4.4.6 Subsystem Data Processing

- Skeletal Tracking: Extracting skeletal positions and movements from the raw depth data.
- User Identification: Differentiating and tracking multiple users in the environment.
- Data Transmission: Sending processed skeletal data to Unity for real-time animation and interaction.

4.5 Data Analysis / Animation Rendering - Unity

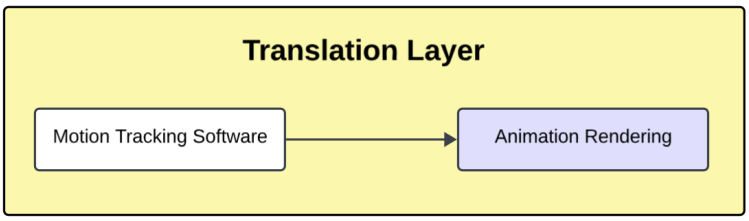

Figure 5: Unity

4.5.1 Subsystem Hardware

Unity operates on high-performance computing hardware equipped with powerful GPUs, necessary for rendering real-time animations based on the processed motion data. The hardware requirements for Unity include a high-end graphics card, sufficient RAM, and a multi-core CPU to handle the real-time demands of the project.

4.5.2 Subsystem Operating System

Unity will operate in a Windows environment.

4.5.3 Subsystem Software Dependencies

- Nuitrack SDK: Provides the skeletal data and depth information necessary for driving animations.
- Graphics APIs (OpenGL/DirectX): These APIs are essential for rendering the animations and visuals in Unity.
- Unity Asset Store Packages [2]: Various plugins and assets that aid in creating the virtual environments and interactions.

4.5.4 Subsystem Programming Languages

Unity primarily uses C# for scripting and development of interactive features, while C++ will be used for lower-level plugin development and performance optimization.

4.5.5 Subsystem Data Structures

- Game Objects: Represent the characters, environments, and interactive elements in the virtual space.
- Animation Controllers: Manage the states and transitions of animations based on input data from the Nuitrack SDK.
- Scene Graph: A hierarchical structure that organizes the objects and their transformations within the virtual space.

4.5.6 Subsystem Data Processing

- Animation Rendering: Real-time rendering of animations based on the skeletal data received from the Nuitrack SDK.
- Interaction Management: Handling interactions between virtual objects and the tracked users.
- Synchronization: Ensuring that the animations are synchronized with the live data to create a seamless experience.

5 Output Layer Subsystems

The Output Layer is the final component of the In/E Motion system, responsible for presenting the processed animations and facilitating interactions with the participants. This layer includes the Projection System and the Interaction System, ensuring that animations are projected accurately and that participant movements are effectively integrated into the visual experience.

5.1 Layer Hardware

The Output Layer hardware consists of four high-resolution projectors used to display animations onto the performance space. The projectors are strategically positioned to cover the entire area, ensuring that the visuals are clear and well-aligned with the physical environment. Additionally, sensors or tracking devices might be deployed to monitor audience reactions and movements.

5.2 Layer Operating System

A Windows environment will be used in this layer to be consistent with the other layers.

5.3 Layer Software Dependencies

- Unity Engine [2]: Used to render the final animations and control the projectors, ensuring that the visuals are synchronized with audience interactions.
- Projector Control Software: Manages the operation of the projectors, including adjustments, alignment, and synchronization with the content being displayed.
- Motion Tracking System: Intel Realsense Cameras used for motion capture to generate new animations.

5.4 Projection System - Projectors

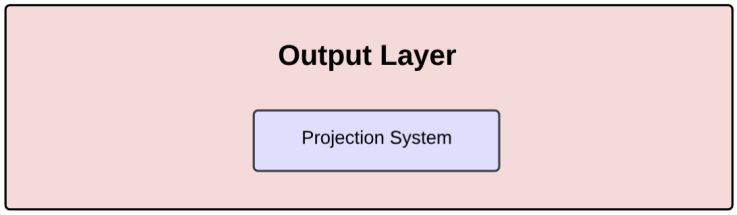

Figure 6: Projectors

5.4.1 Subsystem Hardware

The Projection System consists of four high-resolution projectors capable of displaying detailed animations in real-time. These projectors are calibrated and aligned to cover the entire performance area without overlaps or gaps, ensuring that the visual experience is immersive and seamless.

5.4.2 Subsystem Operating System

The projection control system will operate in a Windows environment.

5.4.3 Subsystem Software Dependencies

- Unity Engine [2]: For rendering and managing the content that is projected.
- Projector Control Software: Manages the calibration, alignment, and real-time operation of the projectors.

- Graphics APIs (OpenGL/DirectX): Essential for rendering the visual content that the projectors display.

5.4.4 Subsystem Programming Languages

- C#: Used within Unity to script and manage the projection of animations.
- C++: For lower-level control of the projectors and any necessary plugins.

5.4.5 Subsystem Data Structures

- Projection Maps: Data structures that define how the animations are mapped onto the physical space, ensuring that they align with the environment.
- Calibration Data: Stores settings related to the alignment and configuration of the projectors.
- Animation Sequences: Data structures that define the order and timing of animations based on audience interactions.

5.4.6 Subsystem Data Processing

- Calibration and Alignment: Ensuring that the projectors are correctly aligned to avoid overlaps or gaps in the projected images.
- Real-Time Rendering: Rendering animations in real-time, based on the data received from the Translation Layer.
- Synchronization: Synchronizing the projections with audience reactions and movements to create a responsive visual experience.

REFERENCES

[1] 3DiVi. nuitrack-sdk. https://github.com/3DiVi/nuitrack-sdk, 2024. [Accessed: Sept. 22, 2024].

[2] Unity Technologies. Tutorials. https://learn.unity.com/tutorials, 2024. [Accessed: Sept. 22, 2024].