Virtual X-Ray Vision to Work in Nuclear Containment (Hot Cell)

Presenter: Eric Guangjie Wei
Supervisor: Dr. Edward Obbard
Outline

- Virtual X-Ray Vision
- Current Observation Methods
- Project Goals
- Our Solution
- Implementation of Our Imaging System
- Results and Conclusion
- Future Plans
Virtual X-Ray Vision

- NOT really x-ray

Not the x-ray used for security check in an airport and body check in a hospital.

- An imaging system produces x-ray effect

An imaging system captures and processes images of interior of a space and displays it in digital devices. User will observe the inner environment of a space by viewing what has been captured by the system. The user sees what the cameras see.
## Current Observation Methods

<table>
<thead>
<tr>
<th>Windows</th>
<th>Cameras</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Checkmark] Provides intuitive experience within the field of view</td>
<td>![Checkmark] Low cost</td>
</tr>
<tr>
<td>![Checkmark] Reliable</td>
<td>![Checkmark] Extends the field of view provided by window</td>
</tr>
<tr>
<td>![X] Limited supply and long lead time</td>
<td>![Checkmark] Introduces redundancy</td>
</tr>
<tr>
<td>![X] Expensive</td>
<td>![X] Non intuitive compared to window</td>
</tr>
</tbody>
</table>
Project Goal

- A new imaging system that combines the advantages of both the windows and cameras for engineers to interact with hot cells, and runs in real-time.

Our Imaging System

- Provides intuitive experience within the field of view
- Extends the field of view with multiple view points
- Introduces redundancy
- Reliable
- Low cost
Our Solution

- **Hot Cell (target)**
- **Imaging System (depth cameras, VR gear set)**
- **Operator (human)**

Connections:
- Captured → Control
- Imaging System → Display
- Command → Imaging System
- Imaging System → Operator
Development Tools

- Depth Camera (Microsoft Kinect™)
- Inferred Sensor

The sensor generates depth values by calculating the reflection time of inferred between an object and the sensor.

For each pixel:

\[
(U, V) \quad D_x \quad V_x
\]

Depth image of our workspace

*Objects outside camera’s detecting range will also be black*
Development Tools

- **Gaming Engine (Unreal Engine™)**
  - The gaming engine is free with rich opensource codes in its community developed by game fans and developers.
  - Create multiple view points.
  - Provides footprints to track our project in the gaming industry.

Building a scene with VR in UE4

VR menu with motion detector in UE4
Pointcloud Construction

- Coordinate Mapping

\[
D_x \rightarrow (U_x, V_x) \rightarrow (Depth)_x \rightarrow WP = \psi((U, V, Depth)) \rightarrow (X_w, Y_w, Z_w)
\]

Depth Camera \hspace{2cm} Gaming Engine

Pointcloud: a massive number of points having specific relative locations in a 3D space and showing the outline of objects or scenes.

For all pixels

\[ (X_w, Y_w, Z_w) \]

e.g.

Pointcloud of a donut
Pointcloud Display

- **Outside game**
  
  User view and interact with a pointcloud through VR HMD*.

- **In game**
  
  User is represented by a character in game and can appear at any location with the pointcloud.

Creating multiple view points between user (character in game) and the pointcloud. This is extremely easy to perform in game by changing camera location or character location.

*HMD stands for Head Mount Device*
Implementation

- Working process of our imaging system with 2 channels.

Image capturing side
- Decoupling locations of user and captured scene (interior of a hot cell) by separating image capturing side and pointcloud rendering side.

Pointcloud rendering side
*HMD stands for Head Mount Device

[Diagram of the working process of the imaging system with 2 channels, showing the flow from Depth Cam1 and Depth Cam2 through pre-processing, depth data, LAN network, pointcloud rendering, and finally to VR HMD* or Monitor.]
Camera location

- Depth camera setup
Image Capturing

- Pre-processing
  - Noise reduction
    Reduce the amount of noise since noisy pixels will be randomly jumping particles in the reconstructed pointclouds.
  - Image encoding and packing
    Encoding depth values from pixel value to binary stream and construct a packet for each depth image before sending through LAN network.
LAN Network

- Python server (2 channels)

A server is setup with Python code, and the server is responsible for transmitting depth data from image capturing side to pointclouds rendering side.

- Improve stability

Significantly reduce packet drop during data transmission.

- Monitor data flow

Provides a platform to track data flow, network traffic condition for debugging and future implementation.
Pointcloud Rendering

- Gaming Engine

Receiving depth data streams from Python server in 2 channels, constructing 2 pointclouds, calibrating them to form a complete 3D scene, and rendering them in either in VR HMD or on traditional monitor.

*HMD stands for Head Mount Device*
Pointcloud Construction

Pointclouds constructed based on the depth information from corresponding depth images. Objects captured by the depth cameras can be recognized in the pointclouds.
Pointcloud Calibration

- In calibration, we fix one of the pointclouds and move another to find the best matching location. The calibration includes translation in x, y, z directions and rotation according to z direction.

- **Translation**
  - Pointcloud1 (fixed)
  - Pointcloud2 (moving)

- **Rotation**
  - Pointcloud1 (fixed)
  - Pointcloud2 (moving)
Calibration Outcome

- Reconstructed 3D scene by calibrating two pointclouds (with RGB depth color mapping)
Current Results

- We have successfully implemented an real-time imaging system that can capture depth images of our target scene with depth cameras and construct and render a complete 3D scene in a gaming engine.

  - Provides intuitive experience within the field of view
  - Extends the field of view with multiple view points
  - Introduces redundancy
  - Reliable
  - Low cost
Extra Values

- Besides original goals, extra values have been discovered for our system.
- Decouples location between user and target scenes (in our case, hot cells).
- Decouples view points of the user and locations of the cameras and windows.
  - Reconfigure the hot cell
  - Change to any view points
  - Navigate inside the scene
- Mixture of virtual reality and real world
  - Realistic training or demonstration
  - HMI information overlay
- Many more possibilities
Future Plans

- Increase performance using better hardware and data compression.
- Explore potential of this system and push the limit of the system in real applications.

Thank you!!!