Tennis Ball Detection CDR

Group 2 - Detect Inc.

Joe Demaria
Andrew Sealing
Connor Ott
Isaac Southwell
Blizzard Finnegan
Wesley Madden

Other notes:

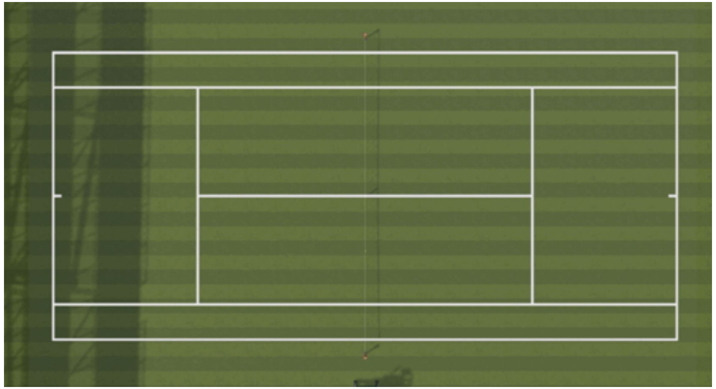
Make sure everyone is happy with talking about the tasks listed on the slides. confirm and edit if necessary.

The idea of each of the demonstration slides is to slowly walk through a demo, piece by piece and in order.

Purpose

During tennis matches, some serves or returns land near the boundary of the field-of-play. The high velocity of the tennis ball makes accurate human determinations of tennis ball position during bounce impossible.

This CDR will detail our design and implementation of a computer-vision object detection system to determine the in-bounds validity of a tennis ball bounce on a court.



Implementation

Consider this as a good place to have a diagram that I can talk about. Show the path between Unity -> SoC -> FPGA -> SoC -> Web Server and SoC -> Unity with delta(t)

If necessary, this is where we also mention any alternate paths we have to take (just briefly, will cover in risks)

Unity

- Capture still images of the tennis ball
- Produce motion of the tennis ball within frame from simulated or black-box function

FPGA

- Process images of the tennis ball using image subtraction

Algorithm (Tennis Ball Calculations)

- Using processed images, determine centroid of the tennis ball
- Determine physical location in 3D space that corresponds with pixel value

Communications Layer

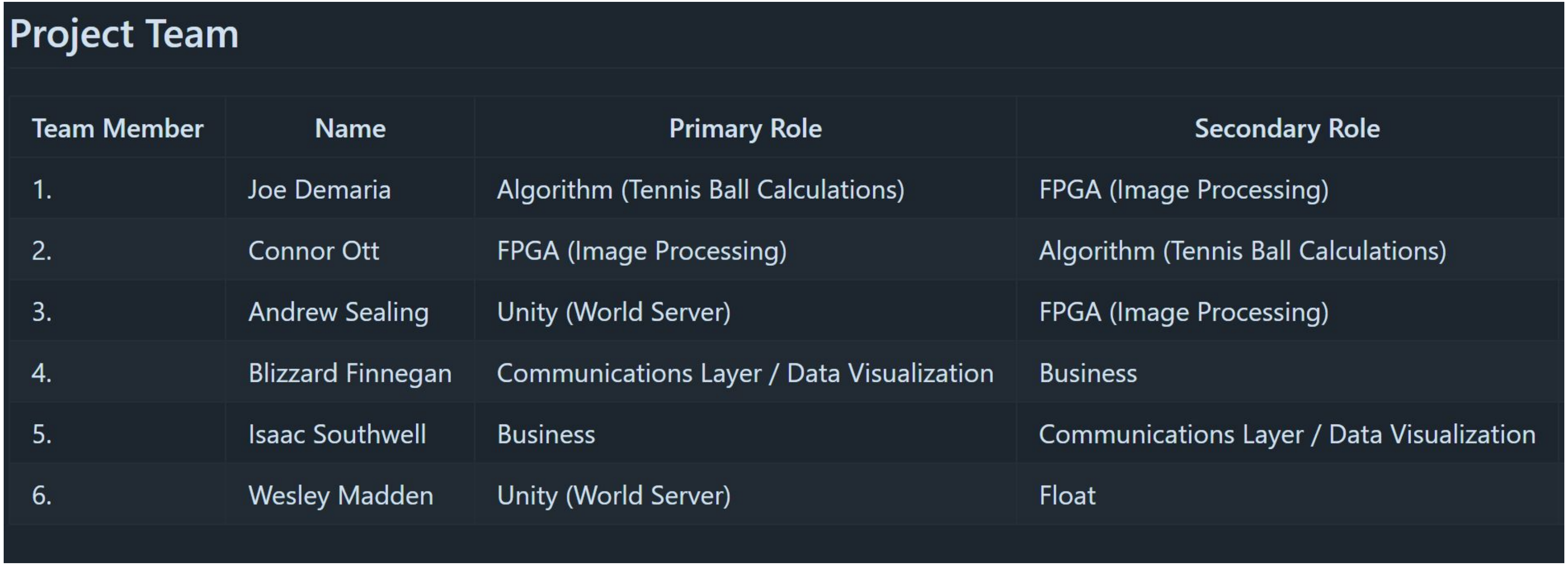
- Create safe data transfer layers for image transfer and data display

Data Display

- Model X, Y, and Z coordinates of each serve on a 3D graph, determine in bounds or out of bounds

Team Members and PDR Authority

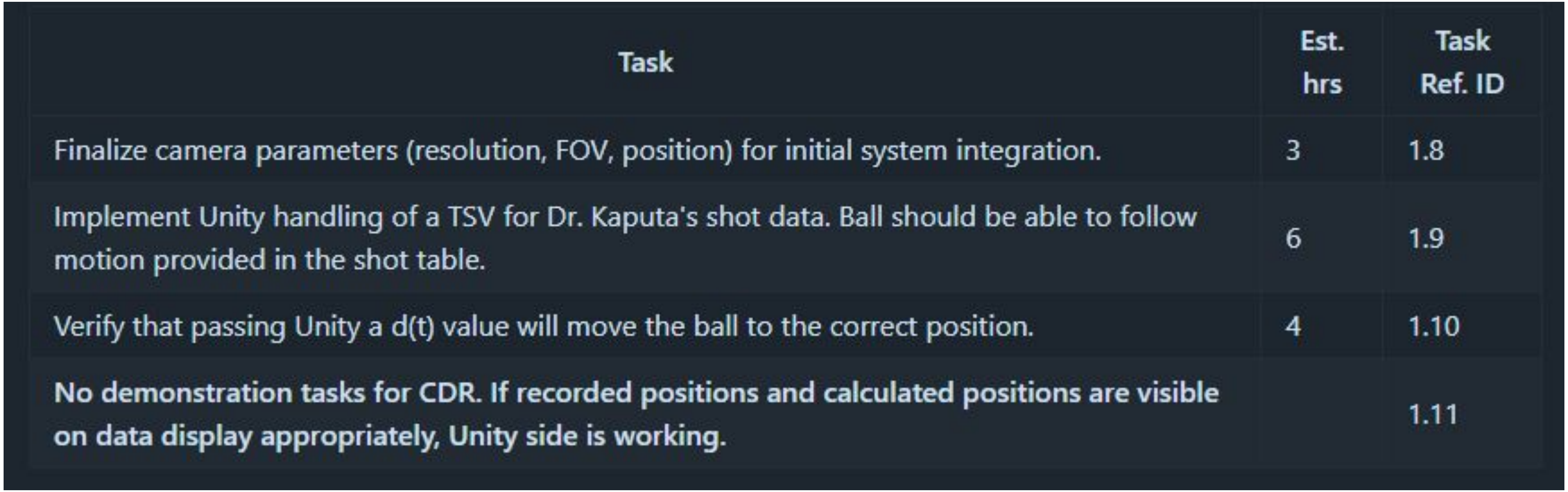
PDR Authority: Dr. Kaputa



Project Team

Team Member	Name	Primary Role	Secondary Role
1.	Joe Demaria	Algorithm (Tennis Ball Calculations)	FPGA (Image Processing)
2.	Connor Ott	FPGA (Image Processing)	Algorithm (Tennis Ball Calculations)
3.	Andrew Sealing	Unity (World Server)	FPGA (Image Processing)
4.	Blizzard Finnegan	Communications Layer / Data Visualization	Business
5.	Isaac Southwell	Business	Communications Layer / Data Visualization
6.	Wesley Madden	Unity (World Server)	Float

Unity Tasks for CDR



Task	Est. hrs	Task Ref. ID
Finalize camera parameters (resolution, FOV, position) for initial system integration.	3	1.8
Implement Unity handling of a TSV for Dr. Kaputa's shot data. Ball should be able to follow motion provided in the shot table.	6	1.9
Verify that passing Unity a d(t) value will move the ball to the correct position.	4	1.10
No demonstration tasks for CDR. If recorded positions and calculated positions are visible on data display appropriately, Unity side is working.		1.11

Unity Implementation in Demo

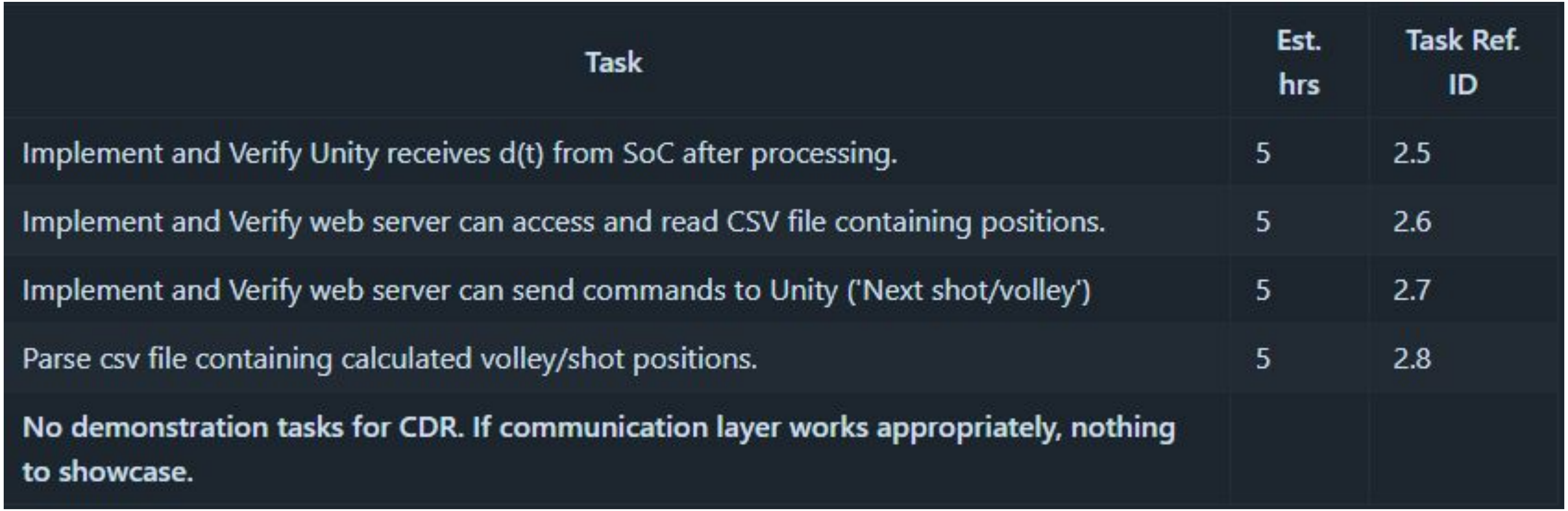
Make this slide a screenshot of our unity scene, including both cameras. Maybe add a gif of the ball moving in slow motion, zoomed in from the side camera.

Discuss the camera positioning and talk briefly about how camera FOV may slightly complicate things.

Unity Implementation in Demo (cont)

A super quick slide that just shows a screenshot of our final camera parameters. Can we get all of them in one readable screenshot?

Communications Layer Tasks for CDR

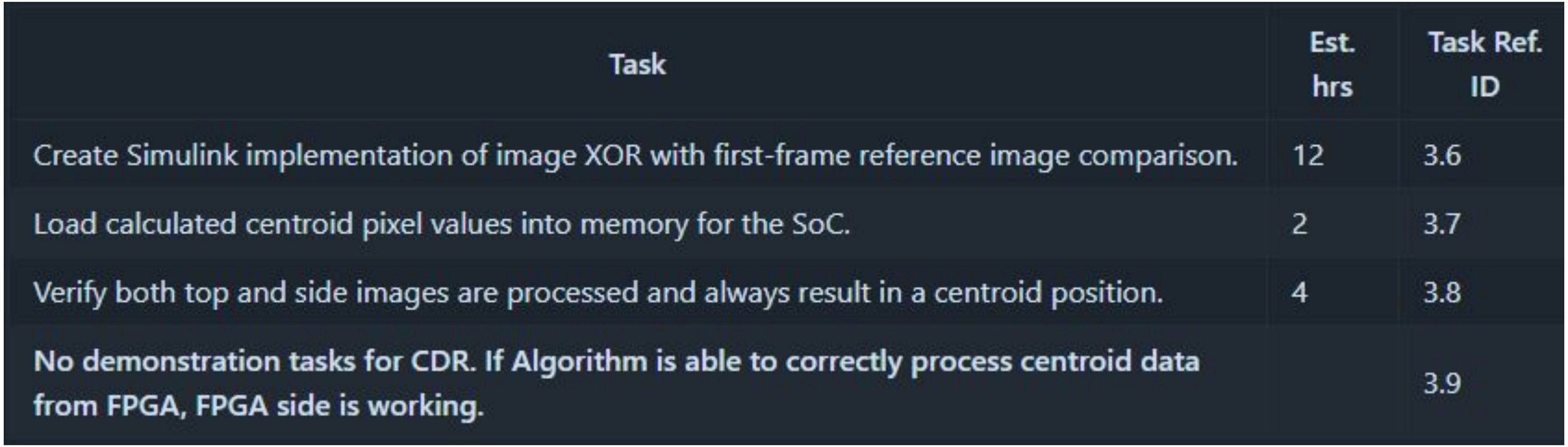


Task	Est. hrs	Task Ref. ID
Implement and Verify Unity receives d(t) from SoC after processing.	5	2.5
Implement and Verify web server can access and read CSV file containing positions.	5	2.6
Implement and Verify web server can send commands to Unity ('Next shot/volley')	5	2.7
Parse csv file containing calculated volley/shot positions.	5	2.8
No demonstration tasks for CDR. If communication layer works appropriately, nothing to showcase.		

Communications Layer Implementation in Demo (Unity -> SoC)

Would be great to have this one be a screenshot of the comms between SOC and Unity. This can literally be the same fucking slides from PDR lmao

FPGA Tasks for CDR



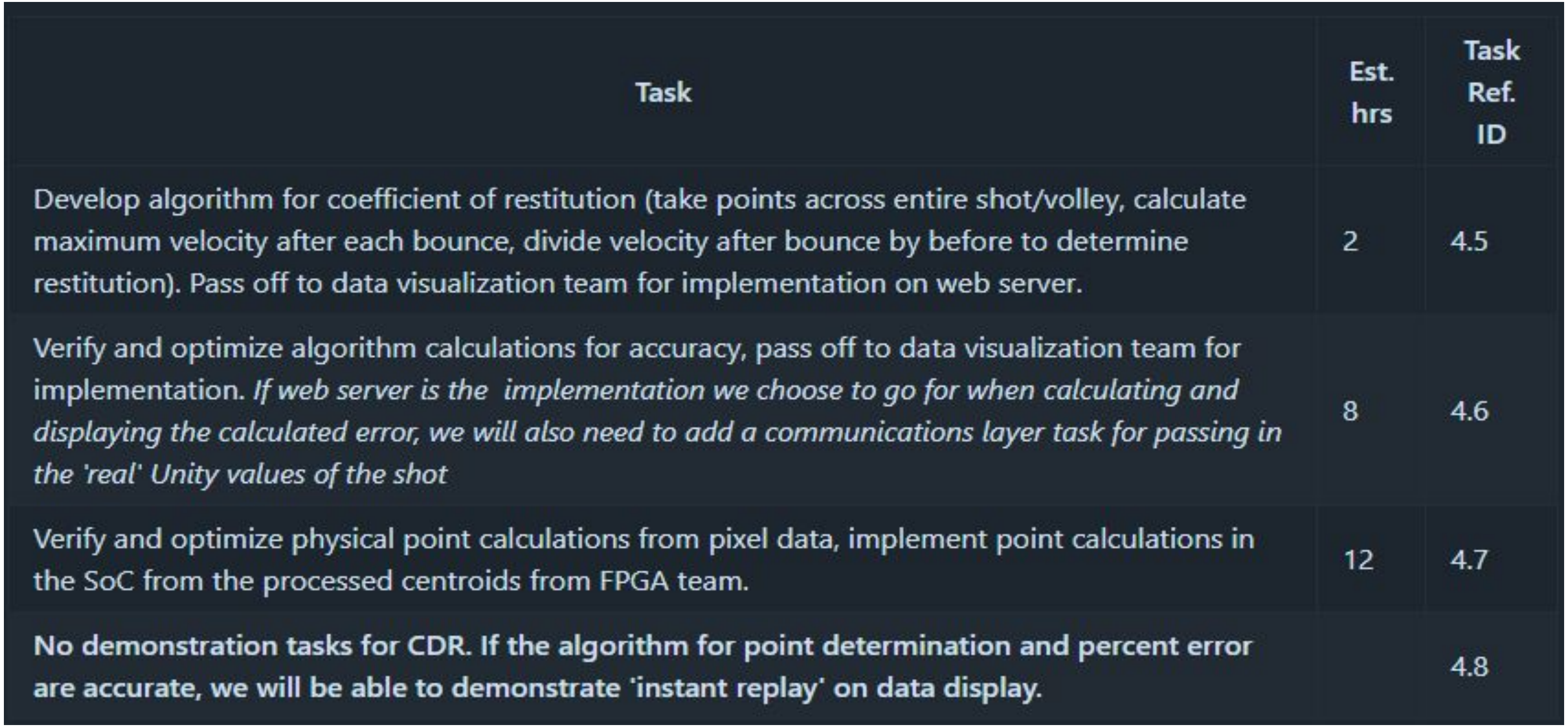
Task	Est. hrs	Task Ref. ID
Create Simulink implementation of image XOR with first-frame reference image comparison.	12	3.6
Load calculated centroid pixel values into memory for the SoC.	2	3.7
Verify both top and side images are processed and always result in a centroid position.	4	3.8
No demonstration tasks for CDR. If Algorithm is able to correctly process centroid data from FPGA, FPGA side is working.		3.9

FPGA Implementation in Demo

Include a screenshot of simulink design? VHDL code? Something to briefly show our implementation that includes the FPGA.

IF we end up not using the FPGA and instead handle the pixel by pixel processing through the SoC instead, maybe include FPGA design but briefly talk about what went wrong?

Algorithm Tasks for CDR



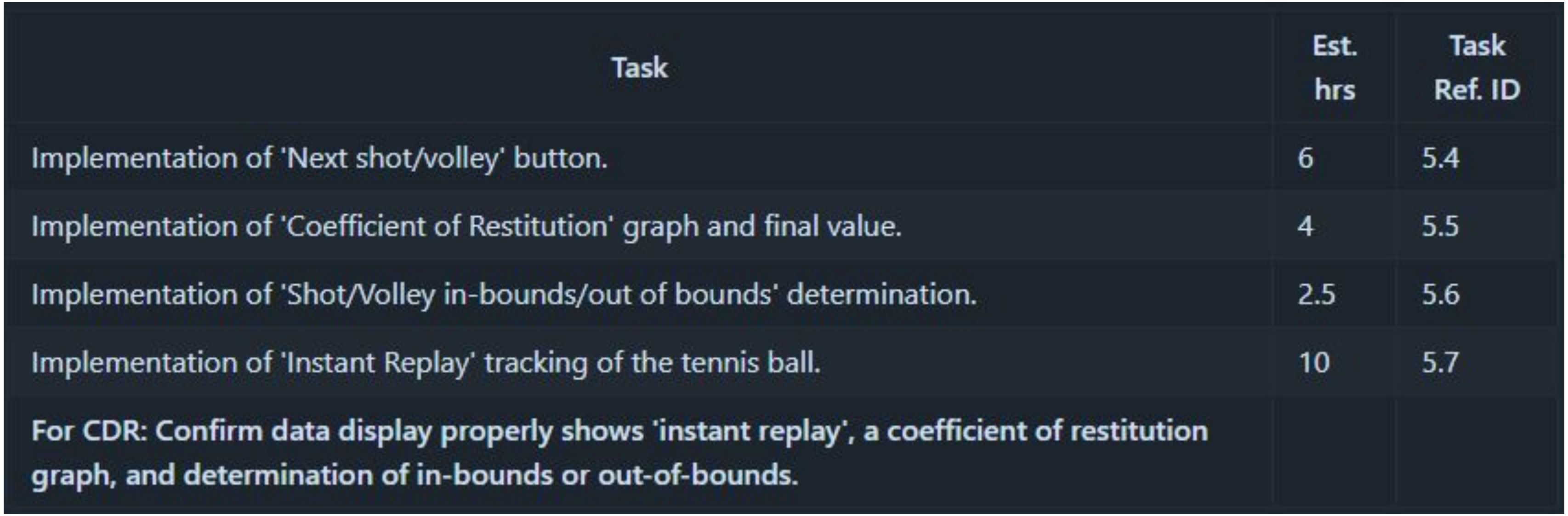
Task	Est. hrs	Task Ref. ID
Develop algorithm for coefficient of restitution (take points across entire shot/volley, calculate maximum velocity after each bounce, divide velocity after bounce by before to determine restitution). Pass off to data visualization team for implementation on web server.	2	4.5
Verify and optimize algorithm calculations for accuracy, pass off to data visualization team for implementation. *If web server is the implementation we choose to go for when calculating and displaying the calculated error, we will also need to add a communications layer task for passing in the 'real' Unity values of the shot*	8	4.6
Verify and optimize physical point calculations from pixel data, implement point calculations in the SoC from the processed centroids from FPGA team.	12	4.7
No demonstration tasks for CDR. If the algorithm for point determination and percent error are accurate, we will be able to demonstrate 'instant replay' on data display.		4.8

Algorithm Implementation in Demo

Screenshot of final code for taking centroid locations and turning them into real positions.

IF we remove the FPGA, include SoC’s rust implementation for calculating centroid positions.

Data Visualization Tasks for CDR



Task	Est. hrs	Task Ref. ID
Implementation of 'Next shot/volley' button.	6	5.4
Implementation of 'Coefficient of Restitution' graph and final value.	4	5.5
Implementation of 'Shot/Volley in-bounds/out of bounds' determination.	2.5	5.6
Implementation of 'Instant Replay' tracking of the tennis ball.	10	5.7
For CDR: Confirm data display properly shows 'instant replay', a coefficient of restitution graph, and determination of in-bounds or out-of-bounds.		

Data Visualization Implementation in Demo

Tennis ball tracking implementation! Show that we can model where the ball pos is in 3D space.

Showcase the “next shot / volley” button and the ability to import new data.

Determination in/out of bounds! Show that we can model whether the ball is in-bounds or out-of-bounds.

“LED”! Show that there is a part of the webpage that lights up according to the requirements in the Rubric

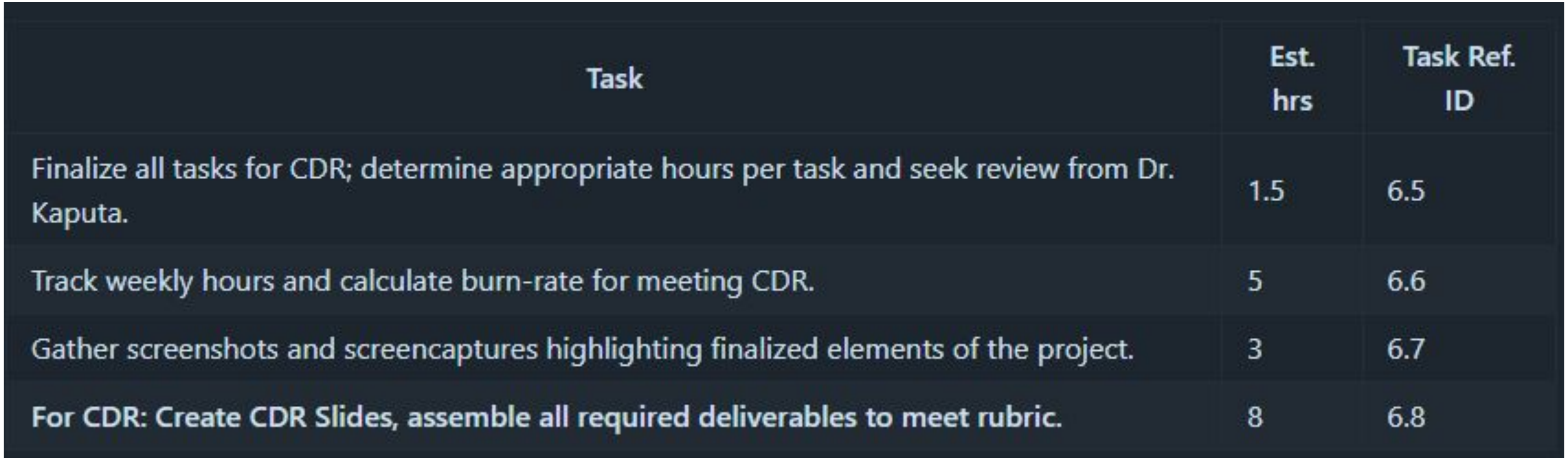
Data Visualization Implementation in Demo (cont)

Coefficient of Restitution! Show that we have the coefficient of restitution graph modeled and we calculate a real value.

Accuracy plot! Show that we have it modeled the real location vs processed location, come up with some accuracy number to brag about.

Accuracy over wind shift plot? Not sure whether we have even looked at this or not.

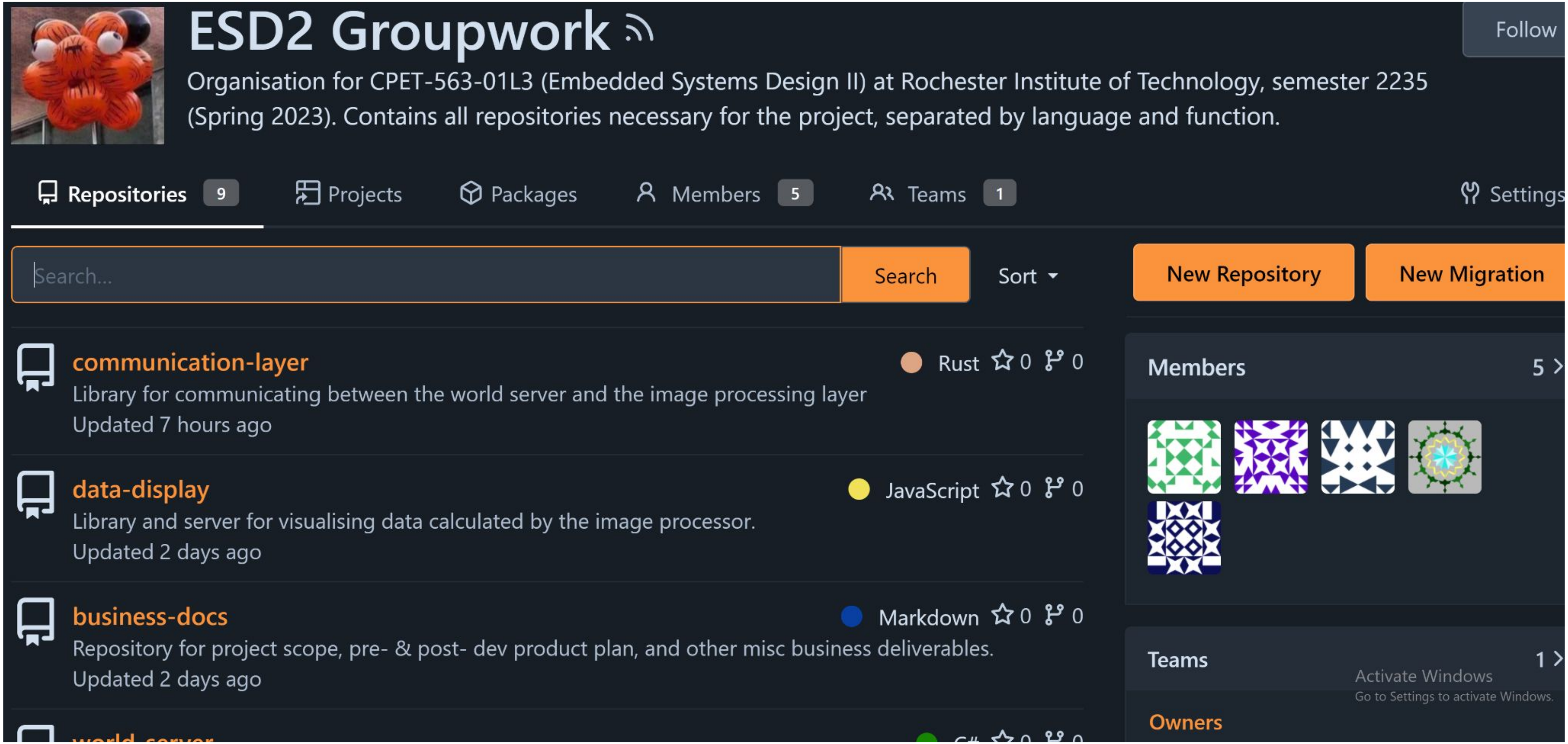
Business Tasks for PDR



Task	Est. hrs	Task Ref. ID
Finalize all tasks for CDR; determine appropriate hours per task and seek review from Dr. Kaputa.	1.5	6.5
Track weekly hours and calculate burn-rate for meeting CDR.	5	6.6
Gather screenshots and screencaptures highlighting finalized elements of the project.	3	6.7
For CDR: Create CDR Slides, assemble all required deliverables to meet rubric.	8	6.8

Graph of labor for CDR, including burndown rates

Labor Produced for CDR



ESD2 Groupwork

Organisation for CPET-563-01L3 (Embedded Systems Design II) at Rochester Institute of Technology, semester 2235 (Spring 2023). Contains all repositories necessary for the project, separated by language and function.

communication-layer ● Rust ☆ 0 ⑂ 0
Library for communicating between the world server and the image processing layer
Updated 7 hours ago

data-display ● JavaScript ☆ 0 ⑂ 0
Library and server for visualising data calculated by the image processor.
Updated 2 days ago

business-docs ● Markdown ☆ 0 ⑂ 0
Repository for project scope, pre- & post- dev product plan, and other misc business deliverables.
Updated 2 days ago

world-server ● C# ☆ 0 ⑂ 0

Organized Repositories

Risks

Camera specifications

- 3840 x 2160 resolution could create images too large to quickly process
- Distance from the court diminishes resolution of the tennis ball
- Camera angle creates distortion around the edges of the tennis court

Processing speed

- Large-format images will take longer to process
- Snickerdoodle may not be the best hardware

Major delays

- Issues with workflow or implementation
- Meeting due dates

Mention the things that really fucked us over during this project

Mention the problems that stopped us from getting any further

Future Work

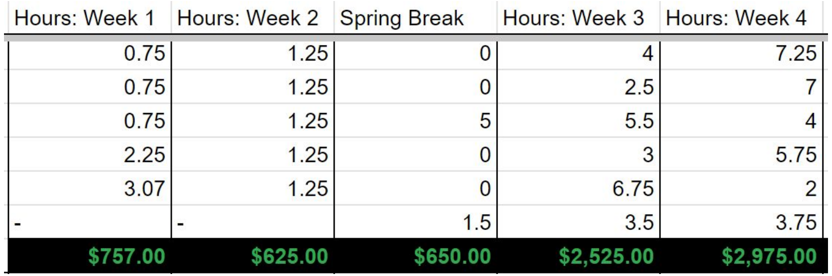
Detail exactly what else we would do if we had a larger budget.

Mention how we would connect the system to the physical cameras

Estimated final time and cost for full implementation?

Executive Summary

Final cost to meet PDR: $7,532   Estimated cost to meet PDR: $6,950



Hours: Week 1	Hours: Week 2	Spring Break	Hours: Week 3	Hours: Week 4
0.75	1.25	0	4	7.25
0.75	1.25	0	2.5	7
0.75	1.25	5	5.5	4
2.25	1.25	0	3	5.75
3.07	1.25	0	6.75	2
-	-	1.5	3.5	3.75
$757.00	$625.00	$650.00	$2,525.00	$2,975.00

From our own development, labor required from PDR to CDR is estimated at 113 hours.

Estimated cost: $11,300

Total for PDR and CDR = $18,532

Update with final CDR prices!

Hardware and Software decisions

- Use of the Snickerdoodle over Jetson TX1
- Discussions on FPGA implementation
- Google Sheets over Gantt chart
- Plotly for data visualization
- Camera positions

Hardware and Software Decisions may need to be a slide by themselves.

Immediate future for Group 2: Detect Inc.

questions about final implementation?