Interim Report

Project objective
This project aims to automate the process of data collection, capturing, and displaying through computer vision technology, specifically Optical Character Recognition (OCR). The collected data is then used to generate graphics in real-time for live streaming. Reducing the amount of manpower needed to run a livestream. And use computer vision technologies to assist the livestream team to reduce human errors, while adding even more graphics and information onto the livestream video. Allowing the viewers to enjoy an athletics livestream with more information.

Project background
When an athletics event is streamed, one of the challenges is to generate and display the information of the race (participating athletes, time, record, athlete performance) in the form of graphic in real time.

There is currently no integration between the judges who record the results, the judges who run the race, and the live-streaming team who broadcast the event live. The live-streaming team rely on traditional methods including looking at the manual results board that is operated by the judges manually, manually looking at the “remaining laps indicator” and manually inputting the data into the graphics system if they would like to show any graphics in the livestream.

Project methodology
Data will be captured via computer vision technology including OCR, and/or deep learning, combined with video cameras connected to the computer via USB or network video stream.

The cameras may be fixed, or not fixed, and are mostly placed at the spectator’s stand of a sports ground, aimed at objects of interest (scoreboard, results board, remaining laps display)

The captured data is then sent to a server for integration with other components, including graphics generator (graphics playout, takes in data and generates the graphics and outputs it via HDMI in real time for live streaming), or exposed via APIs.

There are multiple components in this system.

1. **Central data server** containing pre-imported information in the system. The information may include: list of athletes in each event, athletes' personal information, record of the event. The data is imported via a CSV file in advance.
2. **Management web interface** providing multiple features:
   a. Web portal for managing the pre-imported information. Including viewing, updating, and modifying.
   b. Web portal for monitoring the status of the system
3. **Remaining laps indicator** camera – the camera recognizes the number on the remaining laps indicator at the finishing line of the sports ground. And outputs it as a text to be used in the graphics generator component.

This only applies to long runs, where the race is longer than 400 meters (more than one lap). As the remaining laps indicator will only operate when athletes run more than 400 meters.
4. **Current event camera**: The camera will aim at the LED TV wall at the sports ground, which displays information of the current race. The information of the current race is then extracted and attempts to match with the pre-imported data set to use as display for the graphics playout component. The sports ground’s LED TV wall is considered as the only source of truth for race information. Currently, the “current event” is manually selected, by observing the scoreboard / LED TV wall at the venue. Therefore, it is possible that the “current event” displayed via overlays in the live stream is inconsistent with the actual current event at the venue due to human error. Through using OCR to recognize the text on the LED TV wall, the “current event” can be recognized. A comparison of the graphics operator’s chosen event and the actual event displayed on LED TV wall can be compared and prompt a warning message to the graphics operator when there are any inconsistencies.

5. **Results board camera**: The camera will aim at the results board that is manually operated by the judges. OCR will be carried out on the board, and post-filtering will be applied to achieve the most accurate results possible. For example, if the digit shows the number of attempt of long jump, since long jump should only have 3 attempts, if number “6” is recognized at that location, it should be disregarded. And an alert will be sent to the user of the system.

6. **Graphics playout**: This component takes all the information captured by the cameras above, and generates graphics in real time via web technologies, effectively a webpage. The webpage can then be routed into video mixers via HDMI as an overlay to cleanly and easily overlaid onto a camera feed.

7. **Graphics playout controller**: This component will provide controls for the graphics playout component above. It may be an Arduino hardware-based device, or a webpage that can be opened on any computer/mobile device / tablet in the same LAN. For non-touch screen
operations, keyboard shortcuts can be provided to make it easier to control the graphics. Examples of elements that can be controlled by the controller would be:
- Show/hide “remaining laps indicator”
- Show/hide “athlete’s names screen”
- Toggle next/previous event/race.

8. **Commentator Information System**: A webpage that can be opened on laptops/tablets. Which shows the current event, current athletes’ names, or commentator added notes for a particular athlete. This component is especially designed for commentators to provide more information about the background of the athletes.

9. **Integration with video mixer via IP technologies** (specifically BlackmagicDesign Atem Mini-series). The graphics playout system may be connected to Blackmagic video mixers via IP to control/receive information. The BlackmagicDesign Atem Mini-series was chosen due to its affordability and ease of integration. SDKs are provided by the manufacturer for integration, and it is one of the most popular choices of video mixers for low to medium budget video production companies. The same SDK can also be used to integrate all video mixers manufactured by Blackmagic Design with minimal changes.

10. **APIs for other integrations** on information systems. To extract information for a certain athlete, certain race/event, current race, via APIs.

**Detail implementation:**
The above components are explained with more detail below.

**Current event camera for LED TV Wall**

The LED TV wall is a dot matrix display.

Affine transformation is first carried out to transform the image to a “flat” image.

OCR is then carried out to extract the information on the LED TV wall, and automatically find the appropriate competition record in a pre-entered database to display as graphics and making it accessible via API for further integration.
Remaining laps indicator

Above: the original image from the camera. Remaining laps indicator (top center)

User specifies the 4 corners of the display. The camera, and the remaining laps indicator almost never moves after the camera and the remaining laps indicator is set up. Therefore, the user only needs to indicate the 4 corners once each time.

Through affine transformation, the indicator is “stretched” to a proper orientation and rotation. Simulations carried out in Adobe Photoshop.

After then, image cleanup and processing are applied to optimize the image for recognition. (Contrast, brightness, and black and white filter applied here)
(Taking advantage of the display being yellow, a yellow filter can be applied, boosting the brightness of yellow/orange)

**Text recognition part:**
Identify the 7 segments through simple algorithm: Since the 7 segments of the 7-segment display will always be at the same position after applying affine transformation, the “on/off” status of each of the 7 segment displays can be easily identified. Through combination of the “on/off” status of each of the segments, a number can be identified.

Left: Example of simulation. Green lines indicate the 7 segments of the 7-segment display. If 80% of the pixels in the segment of that area is “on”, then the segment is considered as “on”.

**Challenge:** athletes running in front of the display may cause problems, depending on the angle of the camera. Lighting conditions may also cause problems.

After identifying the remaining laps, a post-identification processing filter will be applied to eliminate any identification results that does not make sense: The filter may be a simple algorithm that only allows decrease of the number (since the remaining laps will only decrease by 1 each time). Combined with a time filter that flags the recognition result when it is decreased too quickly, since it
takes a certain length of time for the athlete to run 400 meters on the track. Therefore, decrement detected shorter than that will be abnormal.

What has been accomplished

After doing some research and analyzing some video recordings collected from a local athletics event in October 2022, I noticed that the initially planned bib number recognition may be too challenging, and not reliable enough to be used in a live-streaming situation. The description are as follows:

**Posture of athletes:** Due to the nature of the sport, athletes swing their arms, or have their arms swinging in front of the bib number. This causes the bib to be partially or fully obstructed from the camera’s view. Making it hard, or nearly impossible to reliably recognize the number on the bib reliably with a frame extracted from a video camera.

**Power supply of the camera in practical use:** In practical use, there is usually no power outlet on the field of the sports ground. Therefore, placing a camera around the sand pit will mean that it will have to be powered by a battery. Which will make it difficult for this solution to be deployed to be in operation for long periods of time in real life.

**Determining the current athlete reliably:** In certain sports such as long jump, the non-racing athletes will often gather around the starting point of the track. The current racing athlete can only be determined by knowing which athlete is precisely located at the center of the track and is away from the crowd (see Figure 1). This requires the camera to be able to find the athlete who is nearer to the camera to find out the current athlete. And therefore, it is not reliable enough to be used. For example, when the racing athlete (in green) walks to the side of the track, the camera may confuse the non-racing athlete that is behind the racing athlete (light blue) to be the current racing athlete.

Based on the problems above, we decided to go for a more reliable, and easier way to determine the current racing athlete – by using the “results board” (Figure 2). The results board is manually operated by the judges of the race. It shows the bib number of the current racing athlete clearly, and the result of the attempt. It is also easier and more feasible to reliably recognize the numbers on the results board. Therefore, this project will be proceeded by carrying out OCR on the results board.
Feasibility test of OCR of the manual results board (Figure 2).

For the Optical Character Recognition (OCR) parts of this project, I chose the Tesseract OCR engine for carrying out OCR of text in images. Mainly due to the complete Python, and C++ libraries that makes it possible to easily use and integrate the Tesseract engine into custom written program.

Tesseract 4.x expects a dark colored text, on a white background. Since the original image (Figure 2) is colored and lacks sufficient contrast for tesseract to recognize the text, further processing is required.

I first converted the image to greyscale, cropped, and increased contrast and sharpness. I passed it to Tesseract for OCR, and the results were not good:
Inputting the image above (Figure 3) resulted in the following OCR result:

The recognized text is

```
31814)
(1K
18k
```

Clearly, further work is needed. Further analyzing the output of tesseract with boxes, it is apparent that the black parts along the edge, and the grid that separates the numbers are interfering with the accuracy of the output. Therefore, further processing is carried out: black borders and the non-character parts of the image are removed to achieve Figure 4 below. This resulted in a much more usable result.

*Figure 4 Non-character areas removed*
The result of Tesseract is much more usable.

The conclusion from this research is that Tesseract expects a clean input image. Therefore, I must process the input video in a way that resembles Figure 4.

To speed up the development of the “results board OCR” component, the borders of the digits will currently be specified by the user manually.

An example is illustrated in Figure 5 below. The green rectangles denote the user input – the borders of each of the numbers.

Figure 5 Results board with user specified borders

System architecture design:

The system architecture design is completed as follows. The system is divided into 5 major components in a microservice-like design. This is to minimize the dependencies between the different components.
The 5 main components are as follows:

1. **Central Data Server** – This is the core component of our architectural design. It stores all the necessary data including the name of the athletes, events, races and their results in the database. This server binds the other 4 components together via socket or HTTP calls. We plan to use PHP and MariaDB to complete this component.

2. **OCR Components** – The OCR component will take video signal of video cameras either via USB (via USB Video device class webcams), or RTSP (Real Time Streaming Protocol). It powers the “remaining laps indicator camera”, “current event camera”, and “results board camera”. We make use of Tesseract to carry out OCR to recognize the information of the current race from the board. This information is then sent to the Central Data Server to be used in other components. Or for “current event camera”, it will also match the OCR result with events and participant list stored in the database to ensure correctness. The Central Data Server records and stores this accurate information. We plan to use OpenCV to capture and cleanup the input video, Tesseract to carry out OCR.

3. **Graphics playout** – This acts as a Display Component. It takes the data from the central data server to be displayed in the form of a webpage with custom graphics design. We make use of a web technologies (HTML, CSS, JS) to complete this component.

4. **Graphics playout controller** – This component aids the user to display their desired information on “ 3. **Graphics playout**” using prompts and a control interface via a controlling webpage (web control). The live-stream operator can easily show or hide the participants information graphics and details of the race for composing better viewing experience when used as an overlay to the camera video feed.
5. **Commentator Information System** – This is a view-only component. This component receives events and takes race data from the central data server. We plan to use WebSocket to open a two-way interactive communication session between the commentator information system server and the webpage. This way we can receive event-driven responses without prompting any manually and have a regular test of our sequential events.

**Event broadcasting:**

When an event is received or triggered by any of the components of the system. E.g. “new race identified” event fired by the “Current event camera” when a race is about to begin, the information will be sent to the central data server. The central data server will then broadcast the message “new race identified” to all of the other components. The components will then carry out the relevant activity, or no activity at all, depending on its own function and role.

This design minimizes the dependencies between the components and avoids a lot of cross function calls between each component of the system. This helps in speeding up the development and avoiding changes in one component breaking another component.

**What will be done**

Since the detailed plan and system design is now completed. We will complete the central data server, live-stream graphics output OCR component according to the plan, code the control component, commentator information system. Finally, we will also complete the Blackmagic Atem series video switcher integration if time allows.

Further potential improvements to the manual results board OCR includes training a deep learning model to recognize the orientation, and find the manual results board in a video stream, without user to specify the location of the board in the picture.