Computer Vision Final Project
Auto Zooming Cameraman - Final Report

Group 30 - Akshat, Vedant, Umang

Fall 2024 - 12 December 2024

1 Introduction/Summary

Our group has developed an AI-powered system that acts as an autonomous cameraman. Using computer vision algorithms and models, the system automatically adjusts camera zoom and focus based on the positions and movements of players and referees or officials. This enables efficient and high-quality game recordings without manual camera control with the aim of making it accessible for different sports at various levels of play. Moreover, we have added post-game commentary as well as Post game statistics to summarize the match.

2 Problem Statement and Target Audience

2.1 Problem Statement

In traditional sports recording, capturing high-quality footage often relies on skilled camera operators who manually adjust zoom, focus, and framing to track fast-moving players and dynamic in-game events. This manual approach is labor-intensive, costly, and prone to inconsistencies, particularly in lower-budget scenarios or for amateur sports. For sports at various levels of play, from grassroots leagues to professional matches, there is a growing demand for affordable, efficient, and reliable systems to produce professional-grade video content. Additionally, summarizing the match through insightful statistics and post-game commentary is often limited to high-profile games, leaving amateur and community-level sports with minimal or no coverage.

2.2 Target Audience

The system targets a broad spectrum of users, including:

1. Amateur and Semi-Professional Sports Teams: Enabling these groups to affordably document matches for training, promotion, and review.
2. Sports Enthusiasts and Content Creators: Offering tools for fans and creators to produce engaging and high-quality sports content.
3. Local Sports Organizations and Schools: Providing a cost-effective solution for capturing and analyzing games for educational and strategic purposes.
4. Professional Sports Teams with Budget Constraints: Assisting lower-tier leagues or development teams in achieving professional-grade recordings without a large crew.
5. Broadcasters and Event Organizers: Enhancing live streaming or post-event content production with minimal resource investment.

By addressing these challenges, the autonomous cameraman system bridges the gap between high-cost professional solutions and low-budget manual alternatives, democratizing access to quality sports recording and analysis.

3 Solution

The proposed system automates camera operations for sports games using a multi-stage pipeline designed to deliver professional-quality video output. The solution begins with robust object detection and segmentation using the YOLOv11 model, followed by heatmap generation to identify regions of interest. Temporal smoothing algorithms ensure stable tracking and zoom transitions, while additional features like automated commentary generation, visual enhancements, and post-game analysis provide an enriched viewing experience. This comprehensive approach not only ensures smooth, high-quality footage but also offers actionable insights into game dynamics.

4 Pipeline and baseline results

The pipeline consists of five main stages that transform raw video input into smoothly tracked and zoomed output footage. Each stage is carefully designed to ensure stable and professional-looking results.

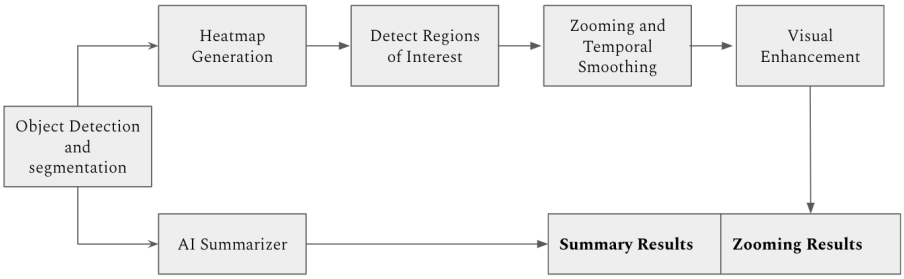


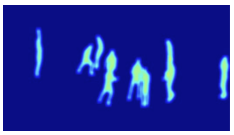
Figure 1: Pipeline Flowchart

1. **Object Detection and Segmentation** The first stage employs the You Only Look Once v11 (YOLOv11) model to generate bounding boxes for each tracked object per frame. This provides,

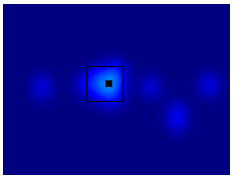
$$M_{i,t} = \text{YOLOv11}(F_t, O_i)$$

where $M_{i,t}$ is bounding box for object i at frame t, F_t is the frame at time t, and O_i represents object i.

(a) Object Detection and Segmentation



(b) Heatmap generation



(c) Region of Interest

Figure 2

2. **Heatmap Generation** Binary masks are merged and processed using Gaussian blur operations:

$$H_t = \mathcal{G}(\sum_i M_{i,t}, \sigma, k)$$

where H_t is the heatmap at time t, $\mathcal{G}$ is the Gaussian blur operator with standard deviation σ and kernel size k. Currently, we are blurring five times in our pipeline.

3. **Region of Interest Detection** For each frame's heatmap, the system:
   (a) Identifies maximum intensity point $(x_{\max}, y_{\max})$
   (b) Extracts surrounding contour C_t
   (c) Computes centroid coordinates for camera targeting

   Two example videos of this step can be seen here [2] and [3]

4. **Temporal Smoothing** A 30-frame rolling buffer implements weighted averaging,

$$W_t = \frac{\sum_{i=t-30}^{t} \alpha_i w_i}{\sum_{i=t-30}^{t} \alpha_i}$$

   where W_t is the smoothed window size at time t, w_i are window dimensions, and α_i are temporal weights. Two example videos of this step can be seen here [9] and [10]

5. **Commentary Generation** We utilized the OpenAI GPT-Vision model API for commentary generation. The processed video was sub-sampled to include every 11th frame (i.e., the 1st frame is shown, followed by skipping the next 10 frames, and so on). At a time, 5 frames were passed to the model, and important events or highlights were appended to a list.

   To make the process stateful, the context from the previous 5 frames was appended along with the next set of frames and prompt. This iterative approach ensured continuity in generating commentary.

   Once the entire list was processed, it was summarized to highlight only the key events and gameplay strategies. Finally, using an OpenAI text-to-speech model, the generated commentary was converted to speech. This audio commentary was then added to the "Commentary" tab after post-processing the video.

6. **Visual Enhancement** Optional visual filters, such as the Kuwahara filter, are applied to enhance the final output's aesthetic quality based on user preferences..

7. **Post Game analysis** To analyze the game after its completion, we initially attempted to cluster players by their jersey colors to determine team affiliations. However, this approach required frequent adjustments to the color thresholds whenever jersey colors varied, resulting in less accurate outcomes. To address this challenge, we introduced a manual color selection feature in the dashboard (Figure 3), simplifying the process for this iteration. In future work, we plan to explore automated methods to enhance player clustering and team identification.

   Once the players are accurately divided into two teams, our interactive tab provides detailed game insights[Figure 4], including:

   (a) **Team Identification and Clustering:** Players are grouped according to their jersey colors, enabling effective team tracking.
   (b) **Team Movement Visualization:** Heatmaps, generated through Gaussian blurring, visualize movement patterns over time.
   (c) **Ball Possession Analysis:** We determine which team is closest to the ball throughout the game and present the result as a percentage, offering insights into ball control dynamics.
   (d) **Frame-Wise Movement Analysis:** A scatter plot of the last 30 frames provides a granular view of in-game actions and strategies.

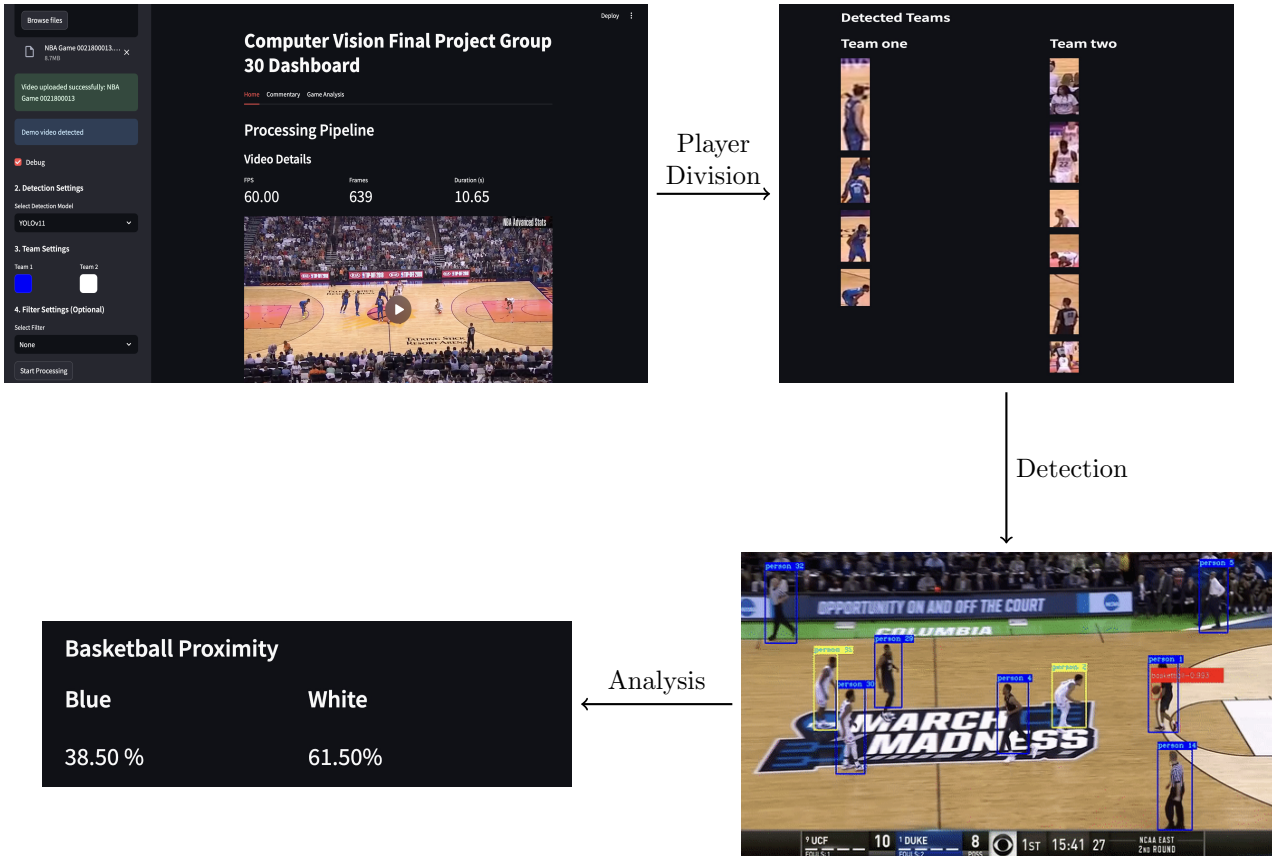
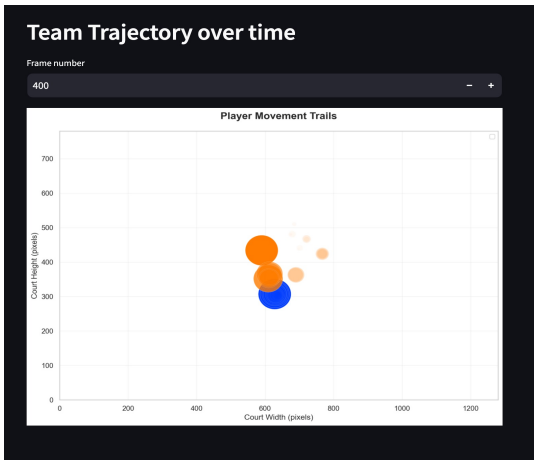


Figure 3: Process flow from manual color selection in the dashboard to detection and analysis results.

Final Post-Analysis Results:

(a) Average Team Movement Over Time



(b) Team Trajectory Over Time

Figure 4: Comprehensive visualization of player movement and gameplay strategies.

5 Iterative Experiments

5.1 Approach 1

Approach 1 represented our pipeline till midterm, where SAM was used to segment objects and create masks for identifying areas of interest.

(a) **Iteration 1:**

- **Approach:** Identify the area of interest, then crop directly to that region and resize the cropped frames to match the video dimensions.
- **Shortcomings:** Results in a jittery video, as even minor changes in the area of interest trigger new cropping. Rapid shifts in the region lead to sudden changes in the camera position, lacking a sense of smooth movement.

(b) **Iteration 2:**

- **Approach:** Build on Iteration 1 by adding a buffer to track past frame sizes and camera positions. Smooth out zoom transitions to reduce jittery effects.
- **Shortcomings:** Improves stability but can result in excessive zoom if the area of interest isn't accurately defined. Over-zooming may crop out people or important details.

(c) **Iteration 3:**

- **Approach:** Extend Iteration 1 and 2 by adding padding to prevent over-zooming. Darken frame borders to guide viewers' attention to the area of interest.
- **Shortcomings:** Border darkening remains experimental and currently appears as a rigid frame around the image. Requires more natural transitions to avoid a distracting "framed" look.

Two comparison videos comparing the original videos with the above iterations can be seen here. [4] and [5] The pipeline was upgraded to utilize the YOLOv11 model, replacing SAM2, which had performance limitations and required significant processing time.

5.2 Approach 2

In Approach 2, we assign weights to different actions and objects according to their priority and then create a weighted heatmap. This heatmap is processed in the same way as in Approach 1; however, in this case, we can always identify the area of interest, regardless of whether multiple people are present in the area of interest.

(a) **Iteration 1:**
Approach In this iteration, we used YOLO11 pose estimation model, and extract the key points from the model to capture player movements accurately. Further an annotated datasets from the Spacejam GitHub repository was used, featuring cropped player clips with joint coordinates marked for the specific action each player was performing. Using these annotated clips, we trained an XGBoost model to classify actions based on the labeled joint coordinates. Pose estimation sample results can be seen in figure 6, and a detailed overview of the model pipeline can be seen in Figure 5.

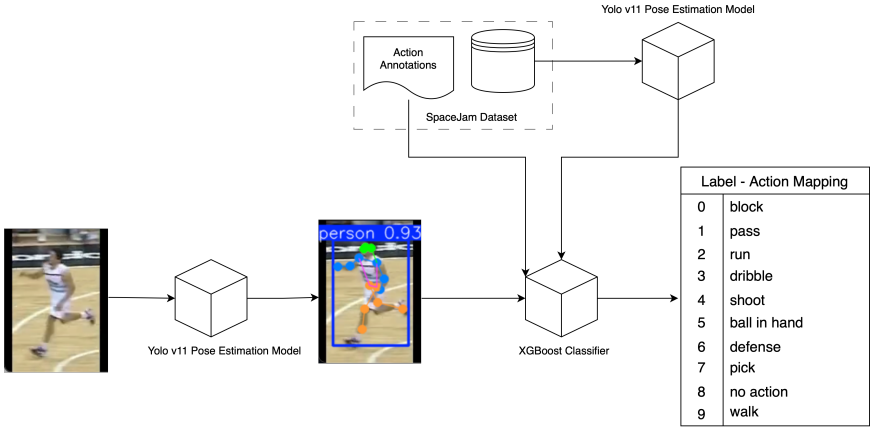


Figure 5: Model Pipeline

Shortcomings

A few challenges emerged with this approach. Since the XGBoost model was trained on individual frames, therefore it lacked temporal understanding of the joints, causing labels to switch abruptly between frames. This approach was unsuitable because we plan to give weights to each action, so that the area of interest can be determined. However, abrupt switching of labels across frames would result in unstable weight updates, as the model struggles to recognize patterns that unfold over time. Without a sequence-aware mechanism, the model's action predictions often lack consistency, impacting overall accuracy. Priority Table for weighing the actions can be found below in Table 1:

Table 1: Priority of Basketball Actions

Priority	Basketball Action
1	Shoot
2	Dribble
3	Block/Pick/Pass
4	Defense
5	Ball in Hand
6	Run
7	Walk
8	No Action

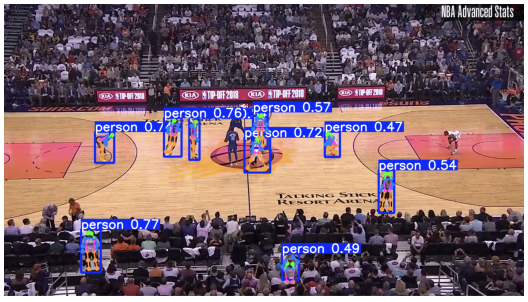


Figure 6: Multi-Person Pose Estimation

(b) **Iteration 2:**

Approach Seeing the need for a model that takes into account temporal movement, we planned to utilize a 3D Convolution Neural Network model for action detection task. The model we used was pretrained on the Sports1m dataset, a large-scale dataset specifically geared toward sports activities. We attempted to fine-tune this model on the SpaceJAM dataset to better cater to our needs. The details of the fine-tuning process and the model architecture used are outlined below:

- Categorical Cross Entropy as the loss function
- Trained for 20 epochs
- Adam optimizer with a learning rate of 10^{-4}
- Batch size: 8

We divided the dataset to prevent class imbalance, ensuring that each class contained 200 videos, resulting in a total of 2000 videos used for fine-tuning.

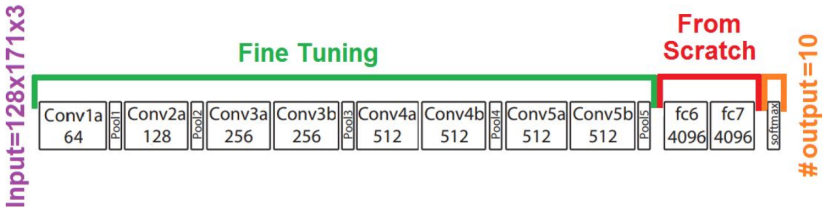


Figure 7: Fine Tuning 3D CNN

Shortcomings

This approach presented several challenges. First, due to limited native support for the specific pre-trained model we used, we had to implement significant portions of the fine-tuning and data handling code from scratch, which proved time-intensive. While many pre-trained models and datasets exist for action recognition, we chose SPORTS1M for its sports-specific focus, which seemed to best align with our goals. However, this choice increased development time. Due to bugs and issues in the pipeline that we are currently debugging, the model was not fine-tuned correctly, ultimately performing at a random sampling level with only 10% accuracy.

Having identified the difficulties in fine-tuning a model without native support, we are now considering alternatives that might streamline the process, such as exploring other 3D CNN models with easier fine-tuning and integration options. In particular, MMAction2 from the OpenMMLab ecosystem appears promising. It provides flexible 3D CNN architectures with extensive support for action recognition tasks, and we plan to explore this in our upcoming iteration.

(c) **Iteration 3:**

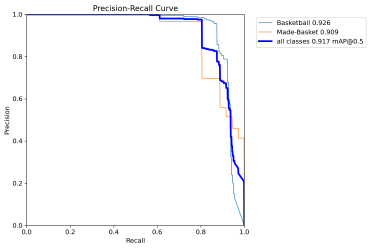
Approach

We simplified the approach by focusing on ball and player detection using a fine-tuned YOLOv11 model for basketball detection(the fine-tuning details will be discussed in the upcoming section) and a regular YOLOv11 model for player detection. In this setup, the ball is assigned a higher weight while players are given a relatively lower weight. This allowed us to create a pipeline to generate a weighted binary mask, which, when merged across frames, enables us to identify the point of maximum intensity within the frame.

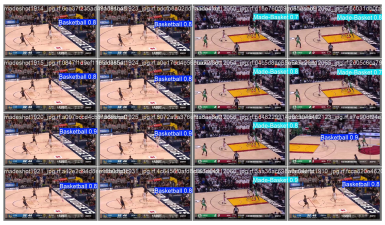
Along with the fine-tuned model for basketball detection, we incorporated a CSRT tracking algorithm. Once the basketball is initially detected, the tracker keeps following it for up to 30 frames or until the next detection, whichever occurs first. This approach effectively tracks the basketball whenever it is in the frame. A video representation of basketball tracking and player detection can be found here [7] and [8]. Sample images showing the results are displayed below:

Fine-Tuning Details: We initiated our fine-tuning process using the YOLOv11 medium model (yolov11.m) and employed a labeled basketball dataset from Roboflow, which includes two classes: 'basketball' and 'made basket'. The fine-tuning was conducted on Kaggle using two T4 GPUs with the following parameters:

- **Epochs:** 100
- **Image Size:** 640
- **Batch Size:** 16



(a) Precision Recall Curve

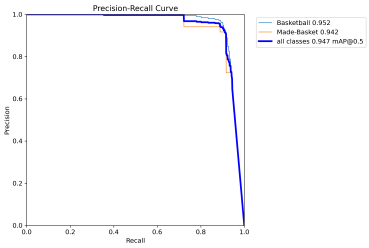


(b) Iteration 1 YOLOv11 Fine-Tuning Results

Figure 8: Results from the First Iteration

However, we observed that due to the small size of the basketball, the model frequently missed detecting the object. To address this issue, we conducted a second iteration of fine-tuning with an increased image size:

- **Epochs:** 100
- **Image Size:** 1280
- **Batch Size:** 4



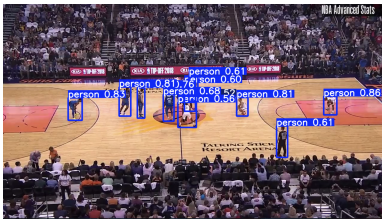
(a) Precision Recall Curve

(b) Iteration 2 YOLOv11 Fine-Tuning Results

Figure 9: Results from the Second Iteration

Based on the improved performance observed with the higher image size, we decided to proceed with this configuration for subsequent applications.

(a) Basketball detection and tracking using fine-tuned YOLOv11



(b) Player detection using YOLOv11

Figure 10: Comparison of Basketball and Player Detection

Future Plans

We plan to use a classification model to identify the court (area of interest) with a binary mask, so that all object detection and masking focus on the relevant area, ensuring that the audience or objects outside the court do not interfere with the model's performance.

In future iterations, selecting the top-k regions could enhance precision and allow for a split-screen feature. Finally, the area of interest would be cropped, resized to match the video frame dimensions, and converted back into a video format.

6 Timeline, Milestones and Duties

Following our experiments, this was the timeline and milestones we followed and achieved, as snapshots from our GitHub project [6].

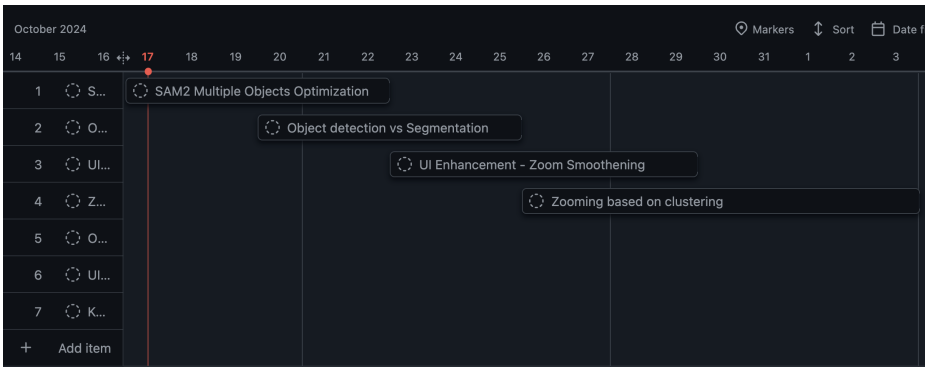


Figure 11: October Timeline and Milestones

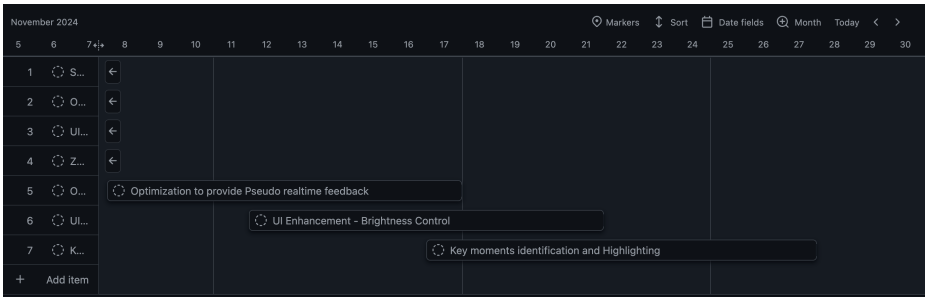


Figure 12: November Timeline and Milestones

6.1 Milestones Achieved

- Integration of SAM2 and YOLO Frameworks
- Implementation of Basic Zooming Algorithm (Approach 1)
- Parallel Development of Approach 2
- Potential Ensemble of Approaches

6.2 Future Milestones

- Weighted Object Detection and Refinement
- User Interface Development
- 3D Convolutional Network Integration
- Iterative Testing and Debugging

6.3 Team Responsibilities

Outlined below are the specific duties assigned to each team member:

(a) **Akshat Kaushal** – Responsible for documentation and experimentation, with a focus on refining the Zooming Algorithm and approach 2.

(b) **Vedant Zope** – Handles documentation and experimentation, with a focus on formulating approaches for Object Detection, tracking and model training/fine-tuning in Approach 2.

(c) **Umang Sharma** – In charge of documentation, integration, and experimentation, focusing on SAM2, as well as UI enhancements related to brightness adjustments and clustering.

7 Future Improvements

The planned improvements are categorized into four main areas to enhance processing efficiency, visual quality, detection accuracy, and user interface features:

(a) **Player Action Detection**
Approach 2 [Iteration 1/2] appears promising for fast and accurate AOI (Area of Interest) detection. However, due to various reasons, it has not been functioning correctly. If given more time, we would focus on refining this approach to gain a clearer understanding of the players' actions. This improvement would likely enhance AI-generated summaries/commentaries and enable more precise post-game analysis and AOI detection. Additionally, we plan to integrate social engineering to incorporate the vision direction and further refine our AOI detection based on observed interactions.

(b) **Optimizations**
The current SAM2 processing pipeline operates sequentially for each object, which increases processing time due to memory constraints. To address this, we are investigating alternative models, such as Approach 2 Iteration 3, which may allow for parallel processing, thereby reducing latency and improving overall efficiency.

(c) **Visual Enhancements**
Enhancing the visual clarity and quality within our pipeline is essential to improve the viewer's experience. The purpose of zooming is to direct the viewer's focus, especially in low-resolution videos, where additional cues such as lighting, highlighting, and smooth zoom transitions can greatly improve visibility. Currently, these enhancements are not automated; however, we aim to dynamically apply them when basic zooming alone may detract from the viewing experience.

(d) **Enhanced Area of Interest Detection**
To improve focus on key areas within the frame, we are exploring weighted object detection using Approach 2 Iterations 1 and 2. By assigning weights to detected objects, we can dynamically identify and emphasize regions of highest relevance, ensuring viewers can easily follow key elements in the scene.

(e) **User Interface Enhancements**
At present, the system focuses on a single area of interest. However, events may feature multiple simultaneous areas of high importance. To address this, we are exploring options for split-screen viewing, which would allow multiple areas of interest to be displayed concurrently, offering viewers a comprehensive view of the action.

8 Reflections

This project was a real, hands-on journey that taught us the value of trying out different ideas and knowing when to move on from those that aren't working. We explored a range of methods—such as SAM2, 3D CNNs, and pose estimation-based action recognition—and found that breaking a complex, real-world challenge into smaller, manageable tasks made it far more approachable.

In working through these approaches, we learned to anticipate potential issues and to keep refining our strategies until we found something robust and reliable. Our research was instrumental in guiding us toward valuable insights and techniques that we could adapt to our needs.

Most importantly, this project highlighted the benefits of teamwork. Collaborating with others not only made the process more efficient but also more enjoyable. The blend of experimentation, careful planning, and close cooperation is what ultimately made this experience both challenging and deeply rewarding.

9 Appendix

The complete project code is available in the following GitHub repository, which also includes a descriptive README with instructions for running the code (recommended if you plan to test/run the setup):

[6] https://github.com/VedantZope/CIS-5810---Auto-Zooming-Cameraman

A zipped version of the code is also available for download at the following Google Drive link:

https://drive.google.com/file/d/1A_AtahfyKrXuM-lMPZxYY45xNQMfTpv8/view?usp=sharing

A demo of the project can be viewed here:

[1] https://drive.google.com/file/d/1QUrT-Iza9JNBWBlaq41TLUGBiWdjsHOc/view?usp=sharing

References

[1] Team 30. *Project Demo*. Google Drive link. URL: https://drive.google.com/file/d/1QUrT-Iza9JNBWBlaq41TLUGBiWdjsHOc/view?usp=sharing.

[2] AreaOfInterestVideo1. *Area of Interest Video1*. Google Drive link. URL: https://drive.google.com/file/d/1o3Clo_wcoCjvdFQaQvScnYLF0FsI30xt/view?usp=sharing.

[3] AreaOfInterestVideo2. *Area of Interest Video2*. Google Drive link. URL: https://drive.google.com/file/d/10OGdAXAhM_tpy3r7db2-qIEpq6pksfp3/view?usp=drive_link.

[4] ComparisonVideo1. *Comparison Video1*. Google Drive link. URL: https://drive.google.com/file/d/1D7v75oraaq5Rp3ysPlV4G3kYXgJKJ845/view?usp=drive_link.

[5] ComparisonVideo2. *Comparison Video2*. Google Drive link. URL: https://drive.google.com/file/d/13c2XHaoGRMH8AvBjoJ4wv_AovPU96d2T/view?usp=drive_link.

[6] ComputerVisionGithubProject. *Github Project - Group 30*. GitHub link. URL: https://github.com/VedantZope/CIS-5810---Auto-Zooming-Cameraman.

[7] BasketBall Detection and Tracking. *BasketBall Detection and Tracking*. Google Drive link. URL: https://drive.google.com/file/d/11Fi8KarXkNJmJK58ZupKyTi2Jy3RduwK/view?usp=share_link.

[8] Player Detection. *BasketBall Detection*. Google Drive link. URL: https://drive.google.com/file/d/1ROm2V5TbkBtLN-NGBNmvN97aZGLiNCr4/view?usp=share_link.

[9] SmoothenedVideo1. *Smoothened Video1*. Google Drive link. URL: https://drive.google.com/file/d/1o31ZIjnyD3km1rPv8qwn5mUhglkYi5dH/view?usp=drive_link.

[10] SmoothenedVideo2. *Smoothened Video2*. Google Drive link. URL: https://drive.google.com/file/d/15HfMCwAqoAiUrP_3eQl8EPMjEk8TTbAc/view?usp=drive_link.