University of Trento

Project Report
for the Computer Vision course

Multi-Camera 3D Ball Tracking System

Lorenzo Orsingher
Alessia Pivotto

Supervised by
Nicolò Bisagno

June 28, 2024

Abstract

This report provides a comprehensive technical overview of the various tasks conducted for the project of the Computer Vision course.
Accurate tracking of objects in three-dimensional space is essential for sports analytics, where precise motion capture enhances performance analysis, training, and strategy development. This project uses the multi-camera system at Sambapolis facilities and aims to achieve high-precision 3D ball tracking.

As a starting point we had a previous project developed by other students that covered the part of calibration of the cameras. However, we opted to rebuild the calibration process from the ground up, addressing limitations to deliver superior performance and reliability. The new calibration methodology introduces refined camera alignment techniques and an improved selection process for calibration frames. These enhancements ensure more precise calibration, facilitating seamless integration and alignment of multiple cameras.

In addition to technical refinements, the system features an intuitive user interface that simplifies the configuration and monitoring of the tracking setup. This interface supports users in managing camera positions, calibrating settings, and visualizing the tracking process, making the system accessible even to non-technical users.

Our enhanced system effectively tracks a ball across multiple video feeds, reconstructing its 3D trajectory with high accuracy. The system consistently provides accurate tracking data and 3D reconstructions, this efficiency and precision translate into more reliable data for performance analysis and tactical evaluations.

These advancements offer a robust, user-friendly solution tailored to the specific needs of sports analytics, providing valuable insights into ball dynamics and player interactions. This system has the potential to help sports teams analyze game play and improve training methodologies by delivering detailed, real-time tracking information with minimal setup time.

So, in this report we elucidate the methodologies employed to develop the project, and finally, the concluding section provides a comprehensive summary of the results obtained throughout the project. Additionally, it outlines potential avenues for further applications and future developments.

Table of Contents

Chapter 1

TODO

Insert images and snippets for the calibration steps
Check everything
results and conclusion (with further improvements)

1.1 Command-Line Argument Parsing

non sarà un capitolo, mi serve solo per ricordarmi

Corner Detection: `get_args_corners`

The `get_args_corners` function is designed for configuring the parameters used in the corner detection phase of camera calibration.

- `-cs, --cameras`: Specifies the indexes of the cameras to be calibrated, separated by commas.
- `-dn, --detect-num`: Sets the minimum number of detections required to conclude the process.
- `-dt, --distance-threshold`: Defines the minimum distance threshold between detections to stop the process.

1.1.1 Multi-Camera Tracking Demo: `get_args_demo`

For demonstrating multi-camera tracking, the `get_args_demo` function defines several arguments:

- `-m, --mode`: Sets the performance mode, with higher values indicating more precise but slower processing.
- `-s, --start`: Defines the starting frame for the demo.

- `-e, --end`: Sets the end frame for the demo.
- `-F, --from_file`: If set, uses detections from a file instead of real-time input.

1.1.2 Pose Estimation: `get_args_pose`

The `get_args_pose` function manages the parameters for pose estimation. It supports the following arguments:

- `-c, --camera`: Specifies the index of the camera for which pose estimation is to be performed.
- `-R, --reuse`: If set, reuses old screen point data, optimizing the process by avoiding redundant calculations.

Chapter 2

Calibration

Our project at the Sambapolis sports facility in Trento involves a multi-camera system designed to capture and analyze ball trajectories. In sports analytics, precise camera calibration is essential for accurate tracking and 3D reconstruction. The primary steps in our calibration process include capturing multiple frames of a checkerboard, identifying reference points such as corners or line intersections, computing the intrinsic and extrinsic camera parameters, and then refining and validating the calibration results. A notable advancement in our process is the improved detection of the checkerboard pattern, which is crucial for achieving precise camera alignment. This chapter delves into our calibration methodology, with particular emphasis on the enhancements.

2.1 Camera Calibration Overview

Our camera calibration system is designed to streamline the calibration of multiple cameras through a user-friendly command-line interface. This tool allows users to select the cameras to be calibrated and define reference points efficiently, providing real-time visual feedback and 3D plotting. This interactive approach simplified the calibration process, reduced manual intervention, and enhanced the overall accuracy. The calibration process involves several key steps, users initiate the calibration process by selecting the cameras from a predefined list. This selection can be executed directly from the command line, allowing for a swift setup and enabling the calibration of specific cameras based on the project requirements. After camera selection, users are prompted to identify and mark reference points. These points serve as a basis for calibration and are crucial for ensuring the accuracy of the system. The tool provides options for manual input or the use of pre-existing calibration data, enhancing flexibility and reusability of calibration results. The core of the calibration process consists of the computation of intrinsic and extrinsic parameters of the cameras based on the reference points. Obtaining these parameters we can move on to validate the accuracy by comparing the computed camera

parameters against known geometric relationships or previously established calibration data.

2.2 Improvements

As in the basic project, we decided to use checkerboards in our calibration method. However, we insert some improvements such as a distance check and a system to fasten the pattern detection process.

2.2.1 Distance Validation

To ensure the quality of the detected corners, the check distance function evaluates the proximity of a given point to a list of midpoints. This is particularly useful for filtering out redundant or erroneous detections. The function computes the Euclidean distance between the given point and each midpoint in the list. Then, keep the point only if it is within a specified threshold distance from any midpoint.

2.2.2 Fast Pattern Detection

The fast detection function enhances the efficiency of detecting chessboard patterns in images for camera calibration. It begins by resizing the image to a lower resolution, which speeds up the detection process while maintaining the pattern's visibility. It then identifies the chessboard pattern in the resized image using the function get corners. Following this initial detection, the function performs a proximity check to ensure that the midpoint of the detected pattern is not too close to previously detected patterns. If the detected pattern is sufficiently distinct from prior detections, the function refines the detection by calling get corners again on the original, high-resolution image to obtain more precise corner coordinates. This method balances speed and accuracy, optimizing the calibration process.

2.3 Visualization and Interaction

As users interact with the system, during identification of the reference points, the visualizations update in real time. Indeed we provide a little and minimal representation of the field with the key points highlighted, the current point is identified with a color, and as you proceed with identification the points already saved are also changed in color. This dynamic feedback loop ensures that users can immediately see the impact of their actions, leading to more precise and accurate calibration results. Upon successful calibration, the system generates a comprehensive visualization of the field and the cameras' positions. The

system produces a 3D plot that represents the field and the calibrated positions of the cameras. This plot provides a spatial overview, showing how each camera is oriented and positioned relative to the field. Users can interactively explore the field through the synchronized views from all cameras. When a user selects a point in any camera view or on the 3D plot, the corresponding point is highlighted across all other camera views and on the 3D plot.

2.4 Final Results and Benefits

The final output of our calibration system includes:

- Accurate Camera Alignment: The calibrated parameters ensure that the cameras are precisely aligned with the field, allowing for accurate capture and interpretation of spatial data.
- Enhanced Field Understanding: The 3D plot and synchronized views provide a holistic understanding of the field from multiple perspectives. This visualization aids in identifying potential issues with camera placement or field coverage and allows for adjustments to be made before data collection or analysis begins.
- User Interaction and Flexibility: The command-line interface and interactive visualizations offer users significant control over the calibration process. The ability to use old calibrations or create new ones on-the-fly reduces the need for repetitive setups and enhances operational efficiency.

In conclusion, our camera calibration system offers a robust, interactive, and user-friendly approach to aligning multiple cameras with a field, ensuring precise data capture and facilitating comprehensive spatial analysis.

Chapter 3

Ball Tracking Overview

3.1 Object Detection and Tracking in Video Processing

In our Python-based project on object detection and tracking, we have implemented several scripts to facilitate efficient video processing and data augmentation tasks, not to mention the interface that allows user to switch cameras, skip frames, or control playback.
One of the core components of our project is the use of YOLO (You Only Look Once). This script starts by loading a pretrained YOLO model (`best.pt`) designed for robust object detection. To optimize performance with high-resolution videos, we employ a customized `SlicedYOLO` approach designed to handle high-resolution videos by dividing them into manageable segments for efficient processing. This variant processes video frames in slices, allowing for efficient computation by ensuring comprehensive coverage of the image for detection.
The script operates in a loop, sequentially processing frames captured by multiple cameras. Iteratively reads frames from each camera's feed, applies the SlicedYOLO model for object detection, and highlights detected objects within the frame. This functionality is crucial for sports analytics, where accurate object detection across various viewpoints is essential.
In addition to video processing, our project includes a Python script dedicated to data augmentation. This script utilizes the Albumentations library to apply transformations such as random cropping, horizontal flipping, and adjustments to brightness and contrast. These transformations are vital for enriching our dataset and improving the robustness of machine learning models trained on annotated data.
The script reads images and their corresponding bounding box labels in YOLO format from a dataset. It then applies specified transformations and saves the augmented images and labels to a target dataset. Visual validation is incorporated to ensure the quality of augmented data, where transformed images

with overlaid bounding boxes are displayed for inspection. This step allows us to verify the effectiveness of transformations and adjust parameters as needed, ensuring high-quality training data.

Another essential script in our project is `checklabels.py`, which serves as a tool for visualizing bounding box annotations on images within a dataset. This script provides a user-friendly interface for verifying the accuracy of annotations. It reads image files and their associated label files, extracts bounding box parameters, and visually renders them on the images using OpenCV. This interactive process allows users to navigate through annotated images, inspect bounding boxes, and confirm their correctness. The script enhances quality assurance in dataset annotation, ensuring reliable input data for subsequent machine learning tasks.

Chapter 4

Results and Conclusion

Do we have some numeric comparison between our method and the previous one? [1]

References

[1] Sagar S Gulawani, Jyeshtharaj B Joshi, Manish S Shah, Chaganti S RamaPrasad, and Daya S Shukla. Cfd analysis of flow pattern and heat transfer in direct contact steam condensation. *Chemical Engineering Science*, 61(16):5204–5220, 2006.