Analyzing top-view sports replays: From video preparation to player tracking

Hadrien Crassous and Etienne Bonnand

Date: May 22, 2025

University of Trento
Department of Information Engineering
and Computer Science
Via Sommarive 9, 38123
Povo (TN), Italy

1 Stitching

1.1 With the stich function from cv2

To get a superior view of the volleyball court, we used the original video from the top ("out10.mp4"). By splitting it in half, we obtained two videos, possibly from cameras at opposite ends of the court.

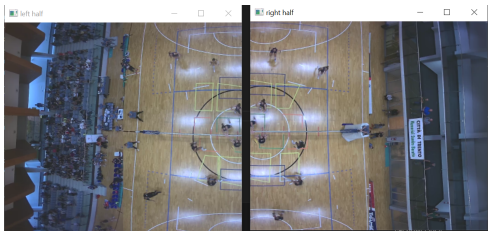


Figure 1: **Top view of the volleyball court in half**

We also decided to split these videos in half to exclude the stands, which are not of interest to us, in order to decrease the image size and to decrease the processing time required for our subsequent algorithms.

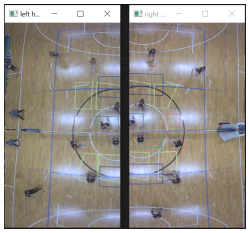


Figure 2: **Top view of the volleyball court in half**

We used OpenCV2's stitching function to merge the images but failed due to minimal overlap. To improve, we created masks to highlight regions of interest, increasing stitching chances.

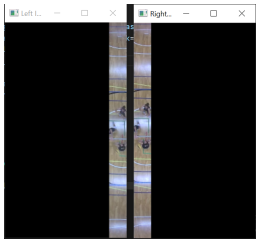
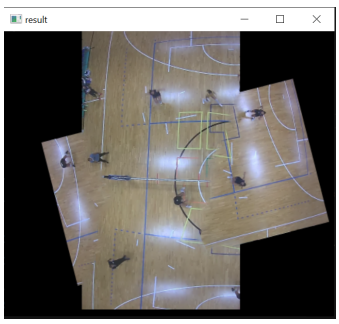
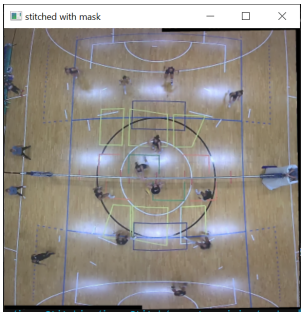


Figure 3: **Top view of the volleyball court in half**

This method was unsuccessful, so we switched to using homography as described in part 2. We later found that using the thresholding function allowed forced stitching, but results were still inconsistent.



(a) **stitching with a low threshold level without mask**



(b) **stitching with a low threshold level with mask**

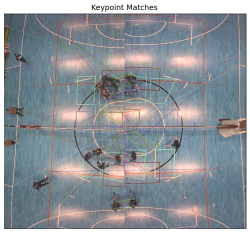
Figure 4: Comparison of stitching results with and without mask

Using masks and enabling "SCANS" improved the results, but video reconstruction was challenging due to long processing times and slight jumps.

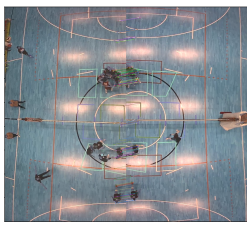
1.2 With Homography

We analyzed each half of the split image to extract keypoints using the SIFT function based on Wei Lyu paper [3], and then aligned the images based on these keypoints. FLANN library was used for efficient matching of keypoints. We applied Lowe's Ratio Test to filter unreliable matches. Defining a Region of Interest (ROI) improved match accuracy, leading to better alignment and stitching.

Figure 5: **Keypoints on the two images**



(a) Matches without ROI



(b) Matches with consideration of the ROI

Figure 6: Comparison of matches with and without ROI

With the matches, we computed a homography to align the photos. Since our camera is stationary, we calculated the homography matrix once and applied it to all frames, ensuring consistent perspective. Using cv2.RANSAC method for homography estimation yielded improved results.

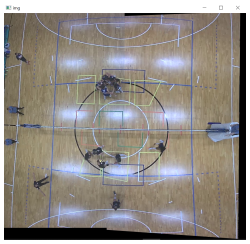


Figure 7: **Stitched image through homography**

1.3 Final treatment of pictures to make a video

Post-processing involved adding black borders to images and applying a binary threshold to detect and remove unwanted areas. This transformation was saved and applied to all video frames, which were then compiled into a video.

2 Detecting the players

Human-annotations of the players positions on a large set of videos are expensive, so we propose a few automatic methods. The detection method should be: **Highly Reliable:** Consistently distinguish players from other objects, despite varying camera settings and player positions. **Fast:**

Capable of processing frames in real-time or near real-time to handle large replay databases efficiently. The tracking method have similar requirements.

2.1 Yolo inference

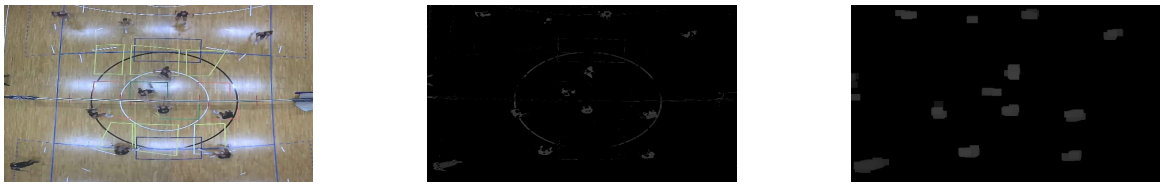
The state-of-the-art solution for multiple object detection is the deep-learning vision model YOLO [5]. It can predict quite accurately where the players are in a single image. It was the very first tool we experimented with Still, YOLO has some limitations.

1. It does not include any tracking solution (usually combined with ByteTrack).
2. It has a high computational cost: up to 5 seconds on a CPU.
3. It performed poorly on top-view videos.

2.2 Region-based detection

Instead, we went for a different approach, that is not deep-learning based and should be more robust to the variety of camera angles that can be used.

It consists of finding the darkest part of the image, by applying a mask, and apply erosion and dilation to retrieve only some "blobs" that represent the players. The erosion is applied to erase the thin dark lines in the background that are left in the mask image, and the dilation is represent each player by a continuous block.



(a) Original image (b) Masked image (c) With erosion and dilation

Figure 8: Steps of the region-based detection

Depending on the field of view, size of the players, the camera-angle, the detection requires different dilation and erosion parameters. So we integrated a sweep for different values of erosion and dilation: different settings are tested and the algorithm stops when it has found the right number of players.

3 Player tracking methods

3.1 Naive method

The naive player tracking method involves detecting players in each video frame and using proximity to the previous position for tracking continuity. However, it suffers from:

Slow Computation: YOLO inference takes approximately 5 seconds on CPU per frame, slowing down real-time processing. **Detection Errors:** Initial frame suitability for detection may not apply uniformly across all frames. **Handling Sudden Movements:** Proximity-based tracking is unreliable with rapid displacements, common at low frame rates. Given these limitations, we focused on more advanced methods capable of addressing these challenges effectively.

3.2 Mean-Shift Tracking

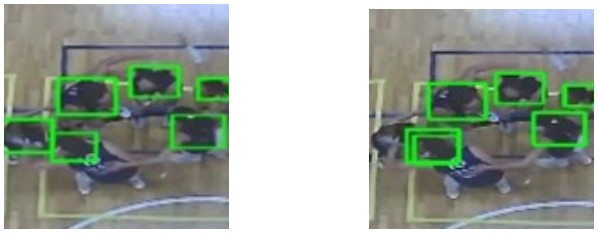
We got inspired by the Computer Vision laboratories for designing our mean-shift tracker. We used multiple instances of mean-shift trackers, one for each player detected.

3.2.1 What worked

This technique was time-efficient: it can process $\approx$ 10 frames per second. One can use this technique for live games, by using a better computer, or skipping a fraction of frames. This technique was quite robust to most tracking difficulties (other than player overlapping), like big accelerations, jumps, players mixing with the backgrounds. The tracker also offer reliable speed estimation, since the center of the box always follows the same part of the player.

3.2.2 Some problems encountered and how we tackled them

Firstly, we struggled to find the correct mask to use for the pre-computation of the histogram. We struggled to pick "good masks", ones that could make the tool stable and generalizable to multiple videos. Secondly, the overlapping players lead to serious mistakes made by trackers, as illustrated in figure 9.



(a) First frame (b) Next frame

Figure 9: Problems with overlapping players

To prevent these mistakes, we tried to disallow the updates that leave 2 boxes overlapped, and instead do no updates on these 2 boxes.

3.3 Evaluation

For evaluation, we used the annotated dataset TeamTrack [6] that features top-view basketball videos that looked similar to the top-view volleyball videos. The most important differences are the wider field of view, smaller players, and different background colors and artifacts in the videos. This dataset has controlled the generalization ability of our tool.

To compute an error of detection, we counted how many times the center of the predicted box would leave the "actual" box (the "actual" box is given by the manual annotation). With this simple metrics, the tracker achieved a perfect score of 0 errors over 3 different videos

4 Additional Features That Could Be Added

Here are proposed additions and their implementation approaches:

4.1 Ball Tracking

Adding ball tracking would allow for the analysis of metrics such as spike strengths and optimal ball trajectories. To achieve this, two primary approaches can be considered: Deep Learning: training a vision model with annotations on where the ball is on given pictures [2, 4] Traditional Methods: Such as color-based tracking for faster implementation in simpler scenarios.

4.2 Action Analysis

While the current tool tracks player positions, understanding specific actions (like spikes or blocks). Existing tools, such as the one described in [1], can recognize volleyball actions using

side-view videos. Challenges for top-view videos include data availability and model adaptation.

4.3 Field Bounds and Team Identification

We have tried some solutions to automatically detect the field bounds.

Hough Line Transform: We experimented with this method that can detect lines in an image but struggles with multiple overlapping lines common in the videos and multiple good candidates.

Deep Learning Models: A neural network trained to recognize field boundaries could be more robust. Although no specific dataset exists, we tried to synthetized one, but without success.

Team Identification: Simple methods can be used to distinguish team players by their jersey colors. This can be done by isolating specific color ranges in the video. Overall, these enhancements could make the tool more versatile and reduce the need for manual input.

References

[1] Mostafa S. Ibrahim et al. "A Hierarchical Deep Temporal Model for Group Activity Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[2] P.R. Kamble, A.G. Keskar, and K.M. Bhurchandi. "A deep learning ball tracking system in soccer videos". In: *Opto-Electronics Review* vol. 27.No 1 (2019), pp. 58–69. URL: http://czasopisma.pan.pl/Content/115237/PDF/opelre_2019_27_1_58_69.pdf.

[3] Wei Lyu et al. "A survey on image and video stitching". In: *Virtual Reality Intelligent Hardware* vol. 1.Issue 1 (2019), pp. 55–83. URL: https://www.sciencedirect.com/science/article/pii/S2096579619300063.

[4] Yoshinori Ohno, Jun Miura, and Yoshiaki Shirai. "Tracking players and a ball in soccer games". In: Feb. 1999, pp. 147–152. ISBN: 0-7803-5801-5. DOI: 10.1109/MFI.1999.815980.

[5] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[6] Atom Scott et al. "TeamTrack: An Algorithm and Benchmark Dataset for Multi-Sport Multi-Object Tracking in Full-pitch Videos". In: *arXiv preprint arXiv:submit/5550700* (2023).