Multi-sensor camera dewarping for basketball court view correction

Tommaso Canova
Master student in Artificial Intelligence Systems
University of Trento
tommaso.canova@studenti.@unitn.it

Mattia Franzin
Master student in Artificial Intelligence Systems
University of Trento
mattia.franzin@studenti.@unitn.it

Abstract

This project involves the development of a Python application using the OpenCV library to reduce artifacts resulting from the mixing of multiple video streams from three multi-sensor cameras in a video recording system within a sports center gymnasium. By defining points of interest, users can establish triangles that can be manipulated to correct distortion caused by the recording system. The mapping between original and new positions is achieved through a technique known as dewarping, which effectively repairs most artifacts caused by object duplication and non-converging lines. The project is open-source and available on GitHub [3].

1 Introduction

In the realm of sports technology, multi-sensor cameras stand out as a groundbreaking innovation in hardware design. Seamlessly incorporating multiple imaging sensors into a single device, these cameras are revolutionizing the recording of sporting events. They offer unparalleled benefits for comprehensive analysis and heightened viewer engagement. By capturing multiple perspectives **simultaneously**, they provide a rich and detailed portrayal of on-field actions, allowing coaches, analysts, and athletes to extract valuable insights into player movements, tactics, and game dynamics. The seamless integration of various angles facilitates precise performance evaluations, strategic planning, and post-match reviews. Furthermore, multi-sensor cameras enhance the spectator experience by delivering immersive and dynamic footage, fostering a deeper connection between audiences and sporting events. As an indispensable tool in the modern sports landscape, they not only redefine the way matches are recorded but also profoundly influence the understanding and appreciation of the intricacies inherent in athletic competition.

Most multi-sensor camera systems require a **calibration phase** to address artifacts that may arise when integrating different video sources. During this calibration process, adjustments are made to ensure seamless blending of the various perspectives, mitigating any discrepancies and ensuring optimal output quality. The calibration can occur either through software or hardware of the systems. Software calibration typically entails adjusting parameters like distortion correction, while hardware calibration might require physically aligning the cameras to reduce parallax errors. Given the setup of three fixed multi-sensor 180-degree cameras in the project system, we opted to address the distortion discussed in Section 2 through software calibration using a method known as *dewarping*.

In computer vision, warping refers to the **deformation** or **distortion** of an image caused by various factors such as lens characteristics, perspective, or geometric transformations. This distortion can

result in images that appear stretched, curved, or skewed, making them difficult to interpret accurately. Warping commonly occurs in images captured by wide-angle lenses or in panoramic views, where straight lines may appear curved towards the edges of the frame. To address this distortion and restore the image to its original perspective, dewarping techniques are employed. Dewarping involves applying mathematical algorithms to analyze and correct the distortion present in the image. By recalculating the pixel coordinates and re-mapping them according to the known properties of the camera or lens, dewarping transforms the distorted image into a rectilinear or more natural-looking perspective. This process enhances the visual accuracy of the image, making it easier to analyze and interpret for various computer vision applications such as object detection, recognition, and scene understanding.

In this project, we explain how a dewarping approach was implemented to correct artifacts resulting from multi-camera systems used to record matches of various sports, including volleyball and basketball, in a sports center.

2 Problem statement

For an innovative digital sports project in collaboration between *Sanbapolis* gym in Trento (Italy) and the University of Trento, three multi-sensor panoramic cameras have been installed in three well-defined positions, respectively two above the two basketball hoops and one at the center of the court to record sport matches and perform video analysis. The cameras in question are PNM-9022V and PNM-9020V models.

Since these cameras are equipped with multiple lenses, artifacts appear during video capture and mixing, as observed in Figure 1, where the hoop appears duplicated, and in Figure 2, where the lines of the court do not converge exactly at the same point. Furthermore, due to the panoramic function of the cameras, a certain curvature can be observed in the representation of the court (Figure 3), caused by a fish-eye effect.

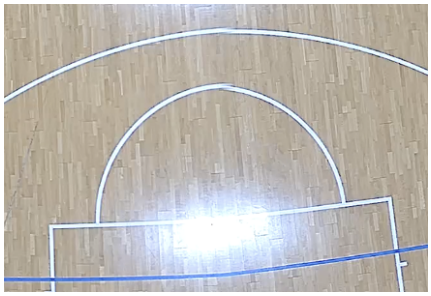
Because of the difficulty in maintaining and physically moving the cameras, it was therefore decided to address these issues at the software level using a technique called dewarping.

(a) Basket appears duplicated

(b) Player approaching the basket appears duplicated

Figure 1: Duplication artifacts

(a) Not converging lines in the middle of the pitch



(b) Not converging lines near the basket area

Figure 2: Line crossing artifacts

Figure 3: Fish-eye effect on the pitch aspect

3 Methods

The following section addresses the solution to the dewarping problem. We allowed the user to select *hotpoints* from which we define a set of triangles through a *Delaunay*[1] triangulation. Once these triangles are defined, users can freely manipulate them to manually correct the distortions described in Section 2. After determining the positions of the triangles, a transformation matrix is calculated to map the original position to the new one. Subsequently, this matrix is applied to the image using a custom blending technique, as defined in Subsection 3.1.

In order to obtain warped triangles, we propose a **three-phases** workflow: "Triangle Definition", "Warping - *Fix Anchors*" and "Warping - *Apply*".

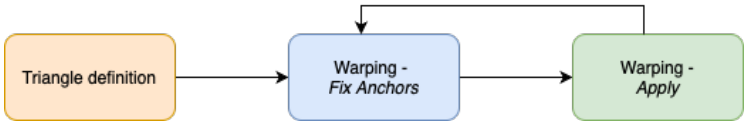


Figure 4: Block diagram of the workflow

3.0.1 Phase 1: Triangle Definition

The user's initial interaction with the application involves defining at least three hotpoints, from which one or more triangles are generated using Delaunay triangulation. This technique, developed in 1934, creates a mesh of triangles from a set of points without any points lying inside the circumcircle of a triangle. It maximizes the minimum angle of the triangles, avoiding skinny triangles and ensuring equiangular triangulation.

Moreover, it is possible to define multiple groups of triangles, thus enabling the isolation of specific areas of interest, without introducing unnecessary overhead in non-relevant parts.

3.0.2 Phase 2: Warping - Fix Anchors

After defining the triangles in Phase 1, it becomes possible to select one or more points from these triangles and then drag them to define new triangles in the desired position. In this phase, it will be possible to visualize the old triangle (depicted in blue in Figure 5c) and the new one (in gray in Figures 5d), without the transformation being executed.

3.0.3 Phase 3: Warping - Apply

Once the destination triangles are determined, a transformation matrix is computed for each pair of old and new triangles following the outlined procedure:

1. Two rectangles are defined, each circumscribing the old and new triangles, respectively.

2. The coordinates of both triangles are expressed relative to the coordinates of their respective rectangles.
3. The `getAffineTransform` function of *OpenCV* [2] is employed to obtain the transformation matrix between the first and second triangle.
4. The transformation is then applied to the old rectangle to obtain the second one using the `warpAffine` function.
5. A mask, sized according to the destination rectangle, is generated to retain only the pixels related to the final triangle, setting the rest to zero.
6. Subsequently, the new triangle is removed from the image and replaced by its warped version.

3.1 Blending triangles

In the case where one or more triangles overlap, merging the common part poses a challenge, as a simple layer sum would result in significant artifacts, including high exposure and loss of details. To address this, for each pair of triangles, the intersection area between them is calculated, which is temporarily subtracted from both triangles. Subsequently, the contribution of each triangle to the intersection is considered with a discount factor of 50%, so that the sum of the two retains the same intensity as the original image. This new intersection then replaces the previously removed area.

If there are intersections involving more than 2 triangles, the approach described above is not sufficient, as it would result in overexposure due to the fact that triangles are treated in pairs, without knowing if a certain portion of the image has already been altered.

To overcome this issue, if an overlapping area involving 3 triangles appears, the intersection, which consists of exactly 3 polygons, undergoes the same blending operation: the areas that do not contain the 3-way overlap are inserted as such, while for the remaining intersection, the smallest intersection among the 3 polygons is inserted, which turns out to be exactly the remaining hole.

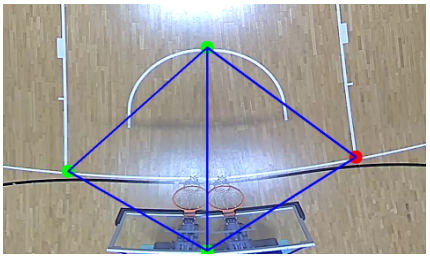
4 Results and conclusions

Overall, the developed application is capable of reducing most of the artifacts caused by multi-sensor cameras in the recording system. Additionally, it is possible to save a warping configuration and load it later, avoiding the need to calibrate the cameras each time.

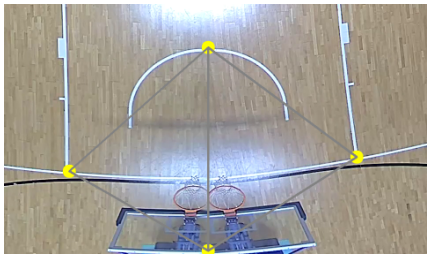
However, the rendering speed is **slowed down** due to the large number of calculations performed for each frame, all of which are handled by the CPU. To mitigate this issue, if an NVIDIA graphics card is available, it is possible to execute the OpenCV libraries with **CUDA** implementation, fully leveraging the parallelization provided by these GPUs.

Currently, the application is capable of correctly handling an overlapping area involving up to 3 triangles. Although this limitation may appear as a weakness of the application, conducting some tests has revealed that it is generally rare to overlap more than 3 triangles to perform a warping operation.

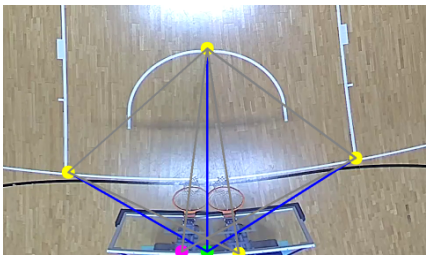
Another improvement could involve using a different triangulation method, as sometimes the configuration obtained after adding new points can drastically alter the arrangement of the triangles.



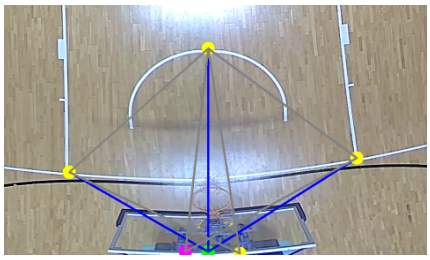
(a) Phase 1: Triangles definition



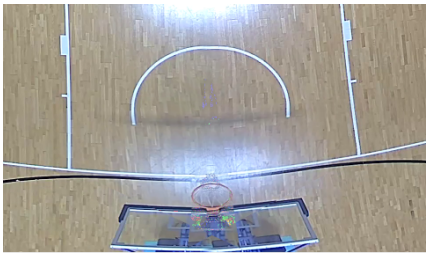
(b) Phase 2: Triangle ready to be warped (fixing anchors)



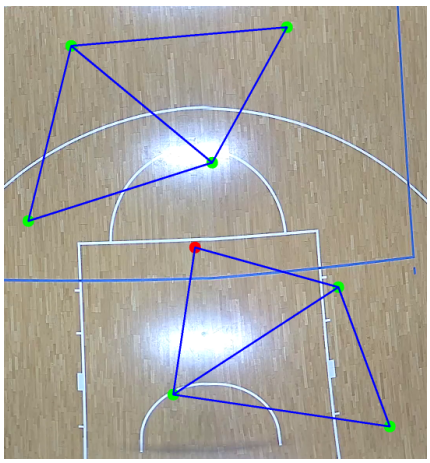
(c) Phase 2: Moving triangles



(d) Phase 3: Apply warping



(e) Final result (warped)

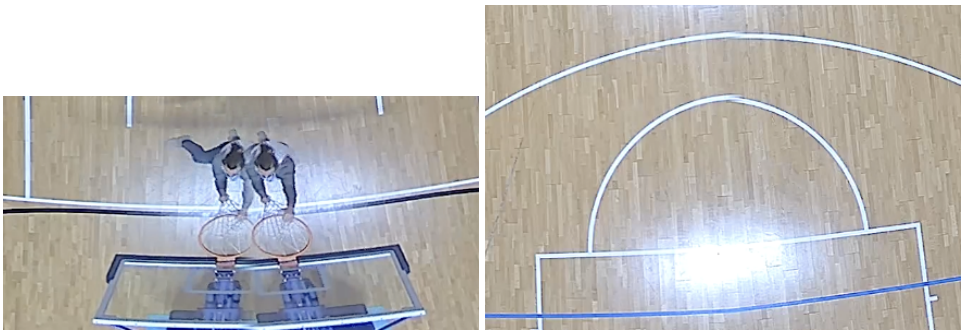


(f) Possibility to define different groups of triangles

Figure 5: Dewarping workflow based on user input

Appendix [3]

As described in the abstract, this project involves the development of a Python application to **reduce artifacts** resulting from the mixing of multiple video streams from three multi-sensor cameras in a video recording system.



Watch the video

The following is the documentation for the application, for ease of use and understanding, also listed on the GitHub repository

Install and Run

Requirements

- **Poetry** as package manager.
- **Python** $\geq$ 3.10
- **Git LFS** to store and download the video/assets used

Installation

1. Install manually the requirements
2. Clone the repo

   ```
   git clone https://github.com/cannox227/camera-dewarping
   ```

3. Access the folder

   ```
   cd camera-dewarping
   ```

4. Download assets *(optional)*

   ```
   git lfs pull
   ```

5. Install required packages using Poetry

   ```
   poetry install
   ```

6. Activate Poetry shell

   ```
   poetry shell
   ```

7. Run the script
   - **Option 1:** run the script for the first time on a specific video source (to define the transformation)

```
dewarp <filename>
```

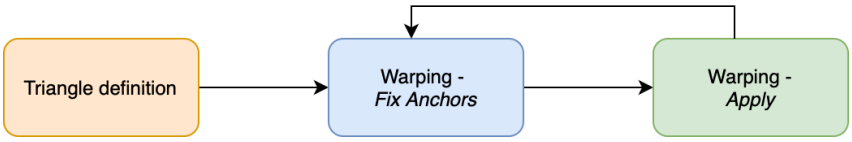
- **Option 2:** run the script using a previously stored transformation matrix (directly picked up from the output folder)

```
dewarp <filename> --load
```

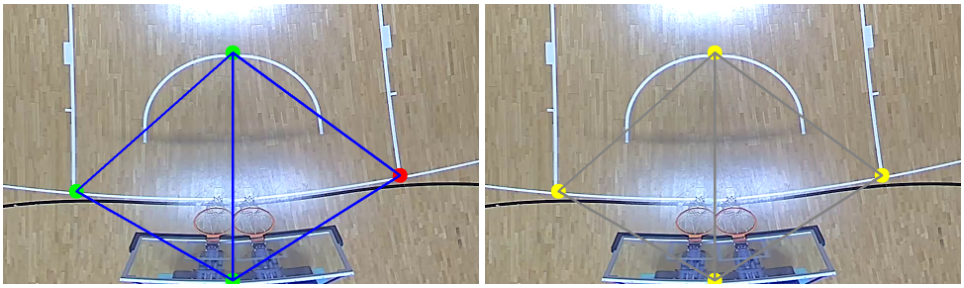
Run the tool

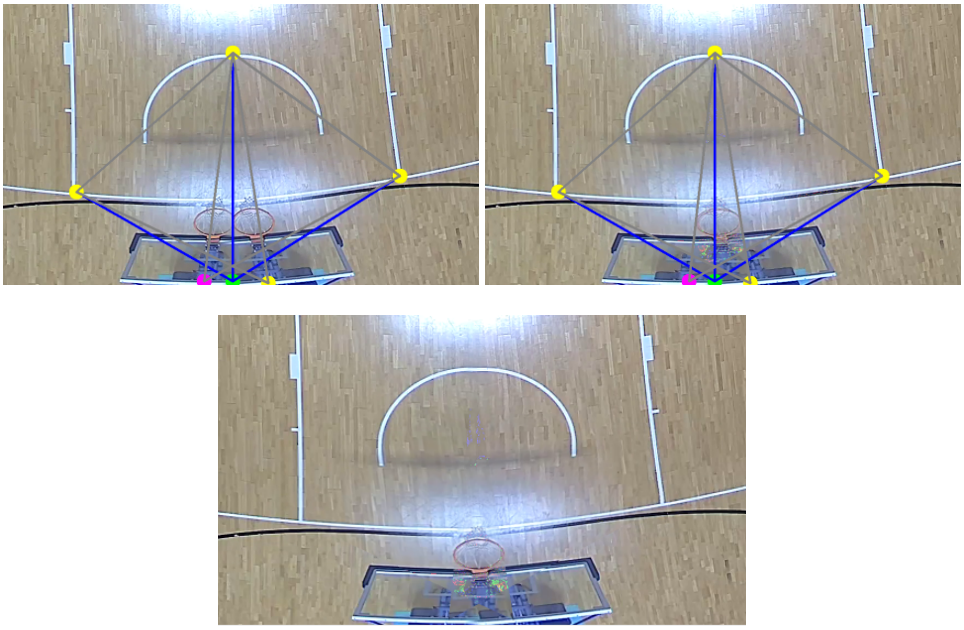
Here below are reported all the possible **flags** and **keybinding** that can be used in the tool

```
Usage: dewarp [OPTIONS] FILE

  Dewarping tool for cameras
  Keybindings:
    - q: exit                    - g: create a new group of points
    - c: cancel selected point   - r: release point selection (deselect)
    - p: pop the last point      - Enter: advance to the next state
    - s: save configuration      - v: toggle visibility of the warped area
    - l: load saved data         - d: toggle drawing of points and triangles
    - Arrows: move reference     - t: toggle reference visibility

Options:
  --load         Load presaved config (from output folder)
  --scale FLOAT  Window scale wrt. video size
  --help         Show this message and exit.
```



Triangle definition and warping When interacting with the video source it is possible to define three or more points from which one or more triangle will be defined using the Delaunay triangulation technique. After having defined the triangles (with blue lines), by pressing `Enter` key we can start moving the triangles to the new positions. In this phase we can move the triangleses (with gray lines), then when we are happy with the configuration we can press again `Enter` and effectively apply the warping.





Watch the video

Handling different groups of triangles Considering the fact that Delaunay triangulation returns a group of triangles that share the same points (some of which are indeed common vertices) and artifacts don't always occur in nearby areas of the court, it's necessary to ensure the user can define more than one group of triangles. By pressing the `g` key, it's possible to define additional groups of triangles, thus applying more than one Delaunay triangulation completely independently.

Watch the video

Erasing a specific point / Erasing the last added point Another proposed feature involves deleting a specific point by selecting it and pressing the `c` key (cancel), or deleting the last added point with the `p` key (pop).

Watch the video

Saving the transformation After having defined the new triangles and having applied the warping, it is possible to store the information regarding each group of old and new triangles and the corresponding transformation matrix. This can be achieved by pressing the key `s`. The files are stored as **pickle objects** in the `output` folder in `.pkl`

References

[1] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[2] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[3] Mattia Franzin Tommaso Canova. camera dewarping. `https://github.com/cannox227/camera-dewarping`, 2024.