École Polytechnique Fédérale de Lausanne

Automatic camera control for sports videos

by Zad ABI FADEL

Bachelor Thesis

Dr. Adam J. Scholefield
Thesis Supervisor

EPFL IC LCAV
BC 332 (Bâtiment BC)
Station 14
CH-1015 Lausanne
SWITZERLAND

January 15, 2021

Acknowledgments

First and foremost, I would like to thank my thesis supervisor Adam J. Scholefield for offering me the opportunity to carry out my Bachelor project within the AudioVisual Communications Laboratory at EPFL and for his guidance and advice throughout the semester.

Additionally, I am grateful to "The AI Guy" YouTube channel who thought me the basics of machine learning and object detectors as well as contributing a google colab notebook that I used as a stepping stone.

One big thanks goes out to the online LaTeX community and the online tech community in general who helped me along each step of the way.

Last but not least, I would like to thank Mathias Payer for providing me with this template.

Lausanne, January 15, 2021 Zad ABI FADEL

Abstract

This project enables the integration of multiple input videos (taken from different angles) in order to output a single consistent video feed with automatic zooming on the region of interest.

It is aimed at amateur/intermediate level sports events. The goal is to be able to easily set up a filming environment where you can film the playing field and quickly obtain a reliable recording focusing on the immediate playing area. Source code is at `https://colab.research.google.com/drive/15OU8wIqu1kC6YxkmIzAgZNcl8Bc42bcO?usp=sharing`

Contents

List of Figures

Chapter 1

Introduction

Across every continent, in every country, in every city, there are sports enthusiasts (from amateurs to professionals) playing their favorite sport in indoors gyms, parks, public spaces, etc... These enthusiasts would like to have a recording of their games for various reasons ranging from collecting memorable moments to analyzing and working on their game.

Most of the time, it is burdensome to find a way to film the game. This usually requires resorting to an outsider to film the event, which is time-consuming and unrealistic when it's not a professional game. Other issues arise, such as lack of space to be able to have a wide global view (frequently happens in indoor gyms), no access to professional filming equipment, etc...

Available resources to address this problem require dedicated hardware. Some solutions are based on fixed cameras that are interconnected in order to synchronize the various feeds. Others use professional grade equipment such as high-quality cameras and tripods that have to be set up in a specific way. Bottom line is these solutions do not target a wide audience and certainly not amateur players and people who enjoy doing sports for fun.

This is where the approach this project relies on comes in handy. The only requirement is access to two smartphones. One to film each side of the court. They would preferably be mounted on a tripod facing slightly different directions in order the capture all the area to be filmed. They can also be fixed on a chair/table (anything stable) with a little manoeuvring and creativity.

After being fed the videos as input, the program does not need specific hardware to run. Run-time is reasonable with regards to video length and quality. All the code is integrated in a single google notebook. After a few manual adjustments, code can be run and will output a single video which covers the whole area of playing and automatic zooming on the region where the play is happening. The algorithms used in the code intend to mimic the job of a sports broadcaster by zooming accordingly to the region of interest.

Chapter 2

Background

In order to understand the code, basic understanding of python is required, as well as fundamentals of computer vision (such as understanding of SIFT, homography, transformations, etc...) and machine learning (object detectors).

Chapter 3

Design

First decision to take was choosing between running the object detection first then proceeding with stitching the two-videos together or vice-versa. The choice was relatively straightforward and easy to make. Stitching the videos together before running the object detection gets rid of the problem of detecting the same object twice in the different video feeds and having to track an object from one feed to another. The other way around doesn't provide us with clear advantages.

Subsequently, when treating the issue of image stitching, multiple obvious candidates arose such as the Scale-Invariant Feature Transform (SIFT) detector and descriptor [1], Speeded-Up Robust Features (SURF) [2] and Seamless Stitching using Multi-Perspective Plane Sweep (MPPS) [3]. MPPS was ruled out due to the fact that it relies on dedicated hardware to run. SIFT was selected due to it's efficiency, robustness, easiness to implement and adapt, and most of all the fact that we can run it once on the first frame of each video then apply the same transformation to the rest of the outstanding frames thus saving a lot of computation and time.

Figure 3.1: Example of two input images to be stitched

Figure 3.2: Example of corresponding output (SIFT)

Moving on to the second main pillar of the project which is the object detection, we rely on AlexeyAB's famous repository (darknet) which is based on YOLOv4 [4].

Finally, we solve the automatic zooming problem by relying on a simple algorithm which aims at expanding and contracting a rectangle (with a fixed ratio (e.g:16/9) in order to fit within its borders the persons that have been detected and optionally some prefixed coordinates. For example, if we're filming a basketball game that is played on a half-court we would like to always have the basket in the rectangle.

Chapter 4

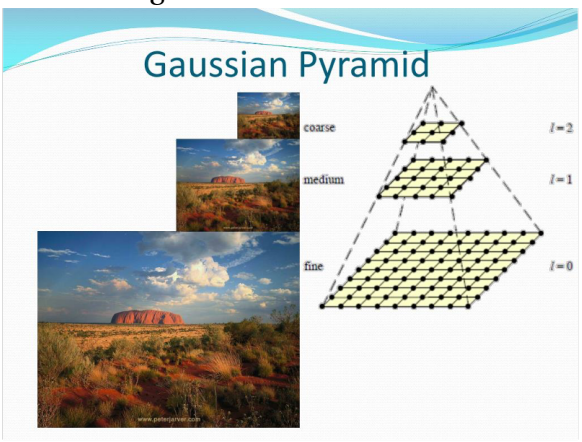
Implementation

In this section, we dive in the details of the implementation of the SIFT detector and descriptor.

SIFT is an algorithm used to detect and describe local features in an image. These features will then be used to stitch the two images together. The algorithm consists of four main steps. The four steps are, in order: scale-space extrema detection, keypoint localization, orientation assignment, keypoint description.

The goal of the scale-space extrema detection is to identify locations and scales that can be repeatably assigned under different views of the same object by searching for stable features across multiple scales using a continuous function of scale. The scale space of an image is a function $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ that is produced from the convolution of a Gaussian kernel ($G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$) at different scales with the input image ($I(x, y)$).

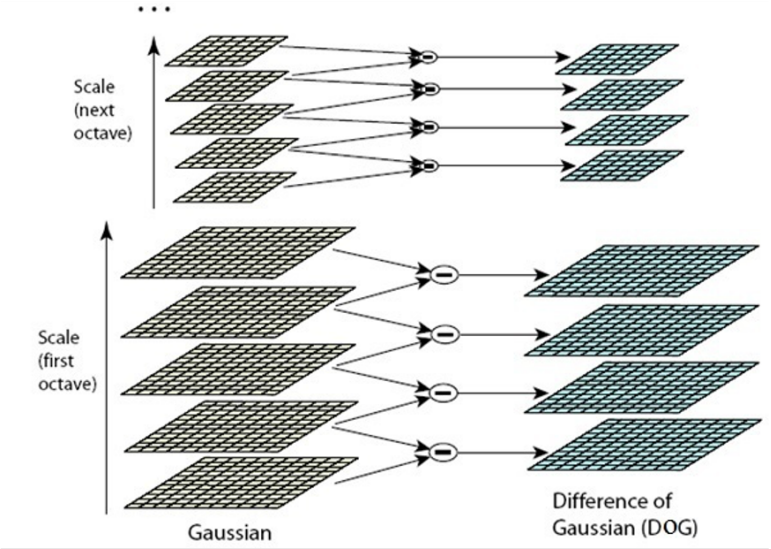
Figure 4.1: Gaussian filter



https://www.slideserve.com/ziv/image-pyramids

Scale-space is separated into octaves $(\sigma, 2\sigma, ...)$. For each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images. Then we compute the difference of adjacent images to obtain the Difference of Gaussians (DoG). This is called Lowe's Pyramid Scheme.

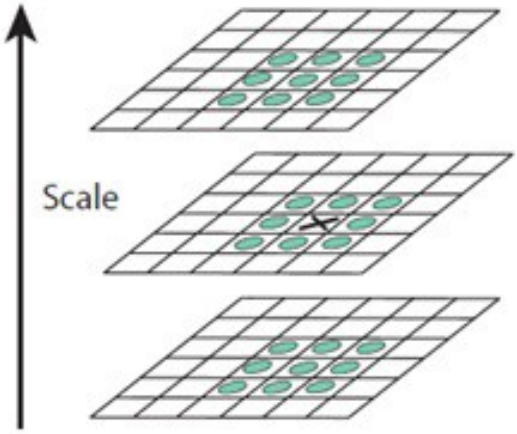
Figure 4.2: Lowe's Pyramid Scheme



https://www.researchgate.net/figure/The-process-for-constructing-the-DoG-pyramid-For-each-octave-of-scale-space-the_fig9_308747165

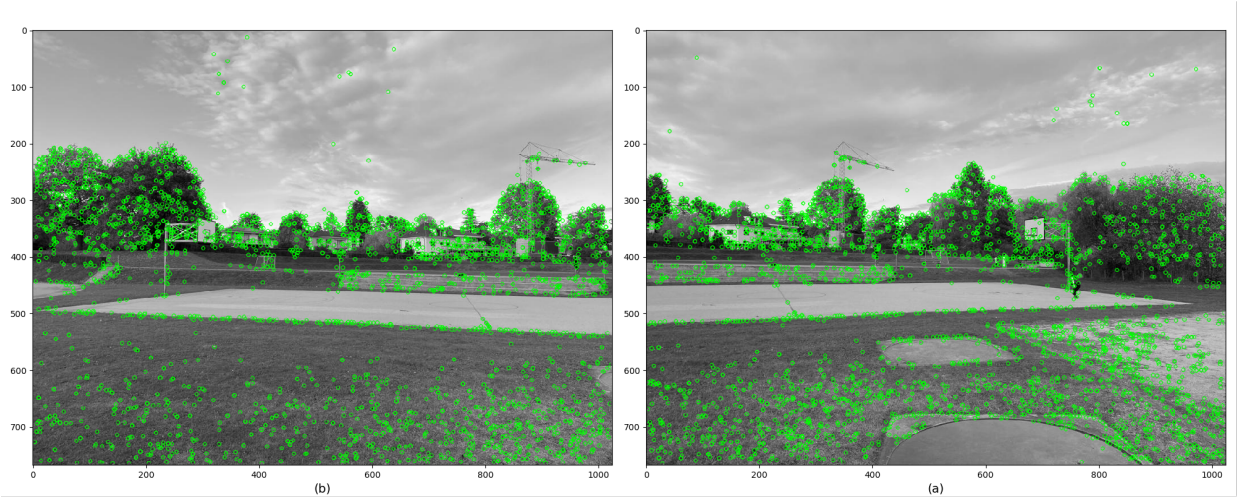
We proceed onto step two: keypoint localization. We want to detect maxima and minima of DoG in scale space by comparing each pixel with it's 8 neighbours in the current scale and 9 corresponding neighbours in previous and next scales. We filter all these potential keypoints, by using threshold values to remove low contrast and computing the trace of the hessian matrix to get rid of edge keypoints .

Figure 4.3: Keypoint localization



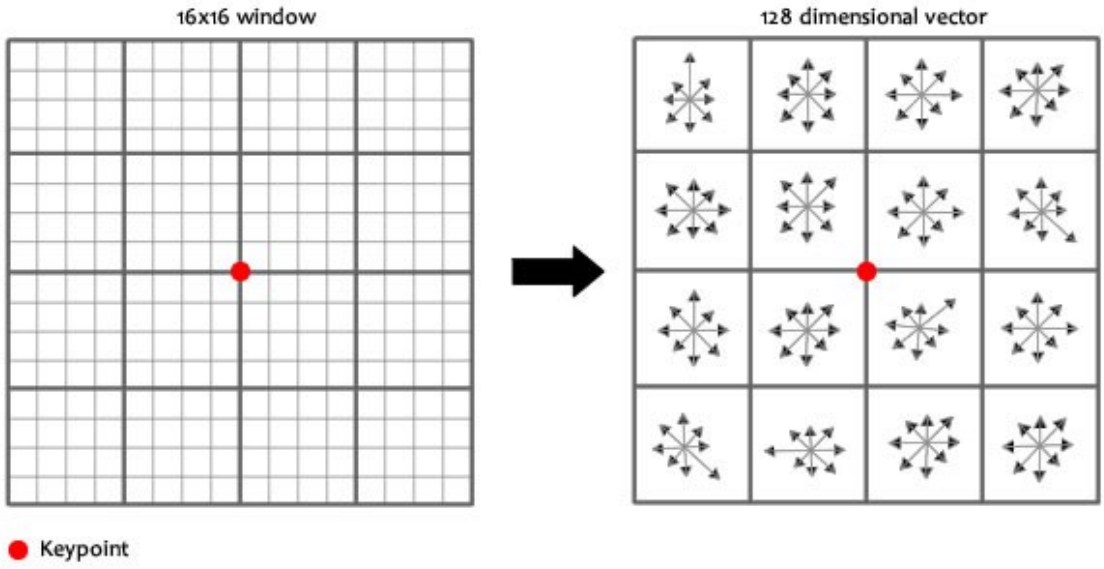
https://miro.medium.com/max/466/0*nbK933cOIyNrmhWi.jpg

Figure 4.4: Result at the end of step two



Next step is to give a thorough description to each of these keypoint to be able to match them. We assign an orientation to each keypoint by creating a histogram of local gradient directions at the scale that has been used previously in the neighbourhood of the keypoint. We choose the most frequent direction. At this point, the keypoints have a location, scale and orientation. We proceed by taking a 16x16 neighbourhood around the keypoint which will be divided into 16 sub-blocks of 4x4 pixels. For each sub-block, we create an 8 bin orientation histogram and we represent it as a vector to form our keypoint descriptor.

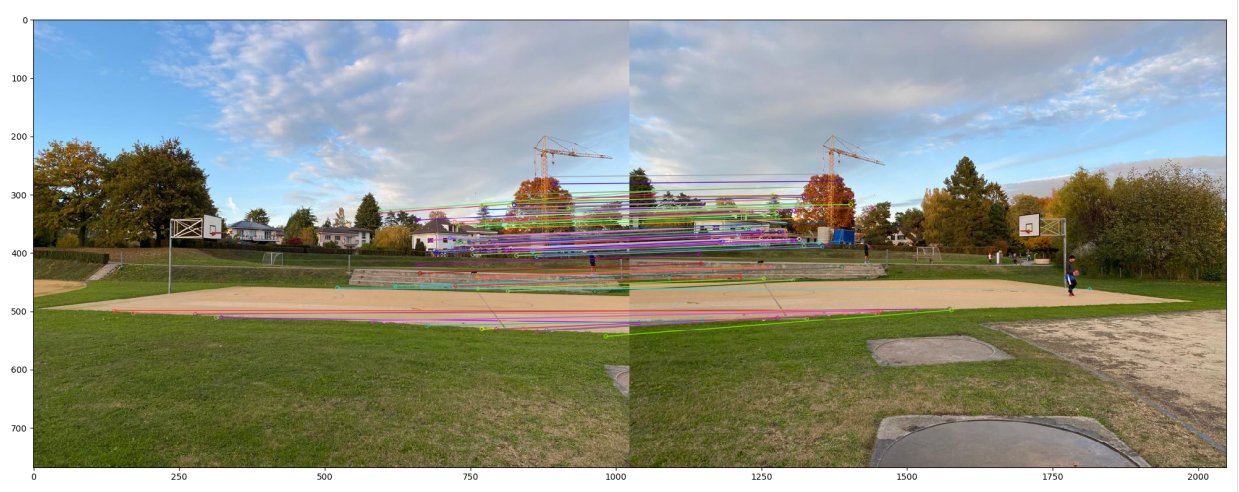
Figure 4.5: Lowe's Keypoint Descriptor



https://miro.medium.com/max/1280/0*r_IdkzMnVs0FfbdI.jpg

Finally , in order to match the descriptors from both pictures, we rely on the K-Nearest-Neighbours algorithm which is easy to implement and resource and computation-friendly.

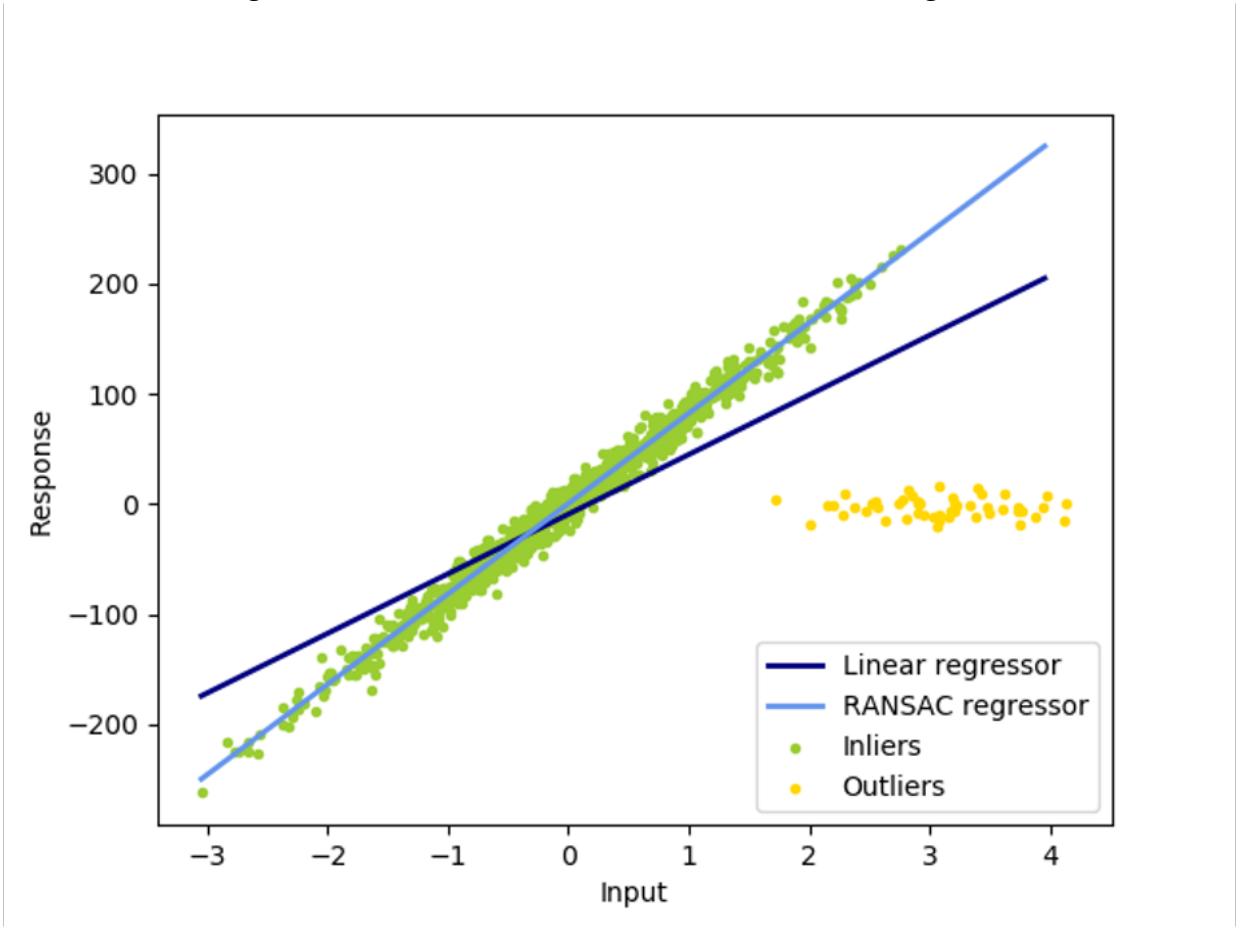
Figure 4.6: Example of matching the descriptors



After pairing the potential keypoints from both pictures together, the homography matrix has to be computed. Briefly, the homography matrix is a planar homography that relates the transformation between two planes. We use this matrix to warp one image onto the other. Since we are matching a lot of points together, we'll have different candidates for the homography

matrix. And we're looking to find the best estimate. To do that, we use a random sample consensus (RanSaC) regressor. This provides a clear advantage over a linear regressor because it is not as susceptible to outliers.

Figure 4.7: Difference between a linear and RanSaC regressor



Once the image stitching is done for the first frame, we reuse the same homography matrix to stitch the remaining frames of the videos.

The resulting output at this stage is a stable video which is the result of merging the two recordings taken from different angles.

The object detector system chosen is YOLOv4 (you only look once version 4). It is an object detection system in real-time that recognizes various objects in a single enclosure. Moreover, it identifies objects more rapidly and more precisely than other recognition systems. It is completely based on Convolutional Neural Network (CNN), it isolates a particular image into regions and envisioned the confined-edge box and probabilities of every region. Concurrently, it also anticipates various confined-edge boxes and probabilities of these classes.

YOLO observes the intact image through training and test time in order to encode circumstantial information classes and their features quietly.

Finally, we take the output of the YOLOv4 detection and we filter it by retaining only the person class that has been detected. We then run a very simple algorithm which consists of drawing a rectangular box which encompasses within it's edges all the players as well as some optional predefined coordinates. We preset the ratio of the box's width to height to 16:9 in order to get a result consistent with what you would expect to watch in a professional sports game. We also add the restriction that absolute difference of the coordinates of the corners of the box between two consecutive frames cannot exceed a certain preset threshold in order to have a smooth deformation of the box throughout the output video.

Chapter 5

Conclusion & Discussion

In conclusion, this projects offers a simple way for sports enthusiasts to obtain reliable footage of their games. The simple set-up as well the fact that it doesn't rely on professional equipment makes it appealing due to its vast outreach. In addition to that, it relies on well-know technologies and the source code is open-source.

The project could be improved by automatically synchronizing both input videos. One way to do it would be to use distinctive events in the audios (e.g: a whistle would show a clear spike in the audio chart) and overlap them.

Image stitching could be improved by computing the homography matrix for multiple frames instead of one unique frame, then use a regressor to find the best estimate (a linear regressor would do just fine since the cameras are stable therefore the homography matrices won't differ by much and there shouldn't be many outliers).

After the object detection, filter the persons detected by removing the boxes whose coordinates are not on the playing field i.e. persons that are not of interest to us (spectators, bystanders, etc...). This could be done by manually defining border coordinates for the playing field or by running a filter (e.g: HSB thresholding) to automatically get the coordinates of the field. The latter can only work when the area aroud the playing field has a different color.

An automatic event detection could be added in order to shorten the footage and get a highlight reel.

Finally, we could use a tracker like the Deep Simple Online and Realtime Tracker (DeepSORT) [5] in order to track the ball. The coordinates of the location of the ball would be used to improve the automatic zooming.

Bibliography

[1] G Lowe. “SIFT-the scale invariant feature transform”. In: *Int. J* 2 (2004), pp. 91–110.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[3] Sing Bing Kang, Richard Szeliski, and Matthew Uyttendaele. *Seamless Stitching using Multi-Perspective Plane Sweep*. Tech. rep. MSR-TR-2004-48. One Microsoft Way, Redmond, WA 98052: Microsoft Research, 2004.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[5] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.