UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Development of a strategy to automatically detect highlights in martial arts tricking videos

Marcos De Rodrigo Talavera

2020

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Título: Development of a strategy to automatically detect highlights in martial arts tricking videos

Autor: D. Marcos De Rodrigo Talavera

Tutor: D. Carlos Cuevas Rodríguez

Cotutor: D. Daniel Berjón Díez

Ponente: D. ……

Departamento: Grupo de Tratamiento de Imágenes (GTI)

MIEMBROS DEL TRIBUNAL

Presidente: D. ……

Vocal: D. ……..

Secretario: D. ……..

Suplente: D. ……..

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: ……

Madrid, a de de 2020

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Development of a strategy to automatically detect highlights in martial arts tricking videos

Marcos De Rodrigo Talavera

2020

RESUMEN

Los avances tecnológicos de los últimos años han permitido aumentar la capacidad de almacenamiento en teléfonos móviles y cámaras de vídeo sustancialmente, dando lugar a que los vídeos que grabamos cada vez sean más y más largos. Este fenómeno ha motivado la aparición de distintos métodos de detección automática de eventos destacados a partir de información de las imágenes del vídeo, con el objetivo de extraer solamente los eventos más importantes de una secuencia de vídeo larga de manera rápida y eficiente. Estos métodos suponen una mejora cualitativa para el mundo relacionado con la creación de contenidos audiovisuales, automatizando el proceso de selección de eventos destacados previo a cualquier producción audiovisual.

Este trabajo propone una estrategia para la detección automática de eventos destacados en vídeos de Tricking basada en la extracción de puntos significativos y en la estimación de sus desplazamientos entre imágenes.

Los puntos significativos identificados pasan por una serie de filtros diseñados para descartar aquellos que carezcan de movimiento (y por tanto de interés) o que no pertenezcan a una zona de interés en la imagen (como el plano del fondo). Estos puntos son posteriormente agrupados en regiones, permitiendo así identificar y caracterizar objetos de interés en la imagen.

Una posterior etapa evalúa el comportamiento en el tiempo de las regiones detectadas con el fin de identificar posibles eventos de interés, que serán finalmente clasificados como destacados o no y filtrados en base a unas variables de usuario que permiten ajustar los resultados obtenidos a las preferencias del usuario.

El resultado final de la estrategia propuesta para un vídeo dado sería una serie de secuencias de vídeo donde se han detectado eventos destacados. Este método permitiría extraer eventos que resaltan por su cantidad de movimiento de largas secuencias de vídeo. Para evaluar su funcionamiento, la estrategia ha sido aplicada a una serie de secuencias de vídeo que muestran distintos patrones de movimiento y los resultados obtenidos han demostrado ser prometedores, mostrando los eventos detectados grandes similitudes con los verdaderos eventos presentes.

PALABRAS CLAVE

Procesado de imagen, detección de eventos, resumen automático de vídeo, deportes.

SUMMARY

Technological advances in recent years have allowed the increase in storage capacity of mobile phones and video cameras substantially, propitiating the increase in number and length of the videos we record. This phenomenon has led to the emergence of different methods to automatically detect highlight events based on information from video images, with the aim of quickly and efficiently extracting only the most important events from long video sequences. These methods represent a qualitative improvement for the world related to the creation of audiovisual content, automatizing the process of selecting highlight events prior to any audiovisual production.

This work proposes a strategy for the automatic detection of highlight events in Tricking videos based on the extraction of key points and the estimation of their displacements between images.

The identified key points go through a series of filters designed to discard those that lack movement (and therefore interest) or that do not belong to an area of interest in the image (such as the background plane). These points are later grouped into regions, thus allowing objects of interest in the image to be identified and characterized.

A later stage evaluates the behavior over time of the detected regions in order to identify possible events of interest, which will finally be classified as highlights or not and filtered based on user variables that allow adjusting the results obtained to the preferences of the user.

The end result of the proposed strategy for a given video would be a series of video sequences where highlight events have been detected. This method would allow extracting events that stand out for their amount of movement from long video sequences. To evaluate its performance, the strategy has been applied to a series of video sequences that show different movement patterns and the results obtained have shown promising, showing detected events closely similar to the true events present.

KEYWORDS

Image processing, event detection, automatic video summarization, sports.

ÍNDICE DEL CONTENIDO

1. INTRODUCTION AND OBJECTIVES 1
   1.1. INTRODUCTION 1
   1.1. PROBLEM DESCRIPTION 1
2. RELATED WORK 2
3. SYSTEM OVERVIEW 3
4. PROPOSAL 5
   4.1. FEATURE EXTRACTION AND TRACKING 5
      4.1.1. FEATURE EXTRACTION 5
      4.1.2. TRACKING 9
   4.2. FOREGROUND FEATURE SEGMENTATION 17
      4.2.1. BACKGROUND SUBTRACTION 17
   4.3. PRELIMINARY EVENT DETECTION 19
   4.4. REGION-BASED ANALYSIS 22
      4.4.1. REGION DETECTION 22
      4.4.2. REGION FILTERING 24
   4.5. TEMPORAL COHERENCE ANALYSIS 28
   4.6. EVENT DETECTION 32
5. RESULTS 35
   5.1. DATABASE 35
   5.2. RESULTS FOR VIDEO 1 38
   5.3. RESULTS FOR VIDEO 2 42
   5.4. RESULTS FOR VIDEO 3 46
6. CONCLUSIONS AND FEATURE LINES 50
   6.1. CONCLUSIONS 50
   6.2. FUTURE LINES 50
7. REFERENCES 51

APPENDIX A: ETHICAL, ECONOMIC, SOCIAL AND ENVIRONMENTAL ASPECTS 53
   A.1 INTRODUCTION 53
   A.2 DESCRIPTION OF RELEVANT IMPACTS RELATED TO THE PROJECT 53
   A.3 CONCLUSIONS 53

APPENDIX B: ECONOMIC BUDGET 55

1. INTRODUCTION AND OBJECTIVES

1.1. INTRODUCTION

Video summarization is a very time-consuming process usually done manually that requires of someone with previous knowledge on the subject. The past few years have experienced an increased interest in developing algorithms that can automatically detect highlights, especially for sport field applications.

In this work we introduce a new dataset, which contains annotated frames with activities from martial arts tricking, and we propose a strategy to automatically detect highlight events in videos of the sport, although it could potentially be viable for a number of different sports. This strategy is based entirely on feature motion estimation. By extracting and tracking frame key points, we are able to estimate the motion present in a video frame, which is then processed to identify regions of interest. These regions of interest are refined to reveal candidate highlight events that are finally evaluated to produce the final output, which is the set of video clips identified as highlight events.

In section 1 we introduce this work and set the problem context. Section 2 presents works that approach related problems. Section 3 introduces the proposed strategy and briefly describes the processes that take place. In section 4, we explain in full detail all processes of the strategy. Section 5 is dedicated to an analysis of the obtained results. Section 6 encompasses a set of conclusions withdrawn after the finalization of this work and an approach that could potentially enhance the results obtained with this method.

1.1. PROBLEM DESCRIPTION

This work has its focus on a particular sport, that is, martial arts tricking. This is a relatively new sport that emerged around 2000 from martial arts exhibitions (or katas), when athletes started incorporating different flips in their routines. Martial arts tricking is mostly known as just tricking, and it incorporates moves from a mix of several different areas, including taekwondo, gymnastics and breakdance among others. Athletes usually perform passes, also referred to as combinations, in which they combine kicks, transitions, flips and twists in quick succession. In the last few years tricking athletes have broken several different world records, mainly in the gymnastics area.

Tricking was born in social media platforms and most of the community uses these to share videos. In the community samplers are known as highlight reels in which a person summarizes his progression over an interval of time. Currently there are no federations or legislations of the sport and competitions are not what one could expect. Actually, competitions in this sport are better known as gatherings, which better describes these, as people usually gather to train together, and the competition just takes a small place in the event.

Given the nature of the sport, it is fairly common for people that practice it to record their training sessions and during gatherings, for future samplers or for learning purposes. This usually entails very long sequences of video in which passes only suppose a small fraction. Here is where automatic highlight detection algorithms come in handy.

The specific scenario that has been contemplated for this work is that of a static recording camera, which is the most common recording setup. For a scenario such as this, the camera is usually placed at a certain distance aiming at the center of the training environment and athletes take turns to perform their passes while the rest of people present at the session wait their turn in the background while moving or stretching.

So, a general video of this sport would show a training environment with athletes performing passes in the center of the frame while others move in the background. What differentiates best a pass from other events is movement, and that is why the focus of this work is in feature motion estimation. With this information the proposed strategy is able to automatically extract video highlights from a given video.

2. RELATED WORK

The research in video summarization has gained a lot of momentum in recent years. In this section we introduce some existing methods.

Video summarization in broadcasted sports takes advantage of common editing practices such as transitions or replays to extract high-level semantics. For instance, [1] summarized broadcasted soccer games by detecting elements in the field (i.e. goal posts), taking advantage of predefined camera angles in the edited videos. [2] determined key events in a game by leveraging slow-motion replays and [3] identified scenes in which players scored in soccer and basketball games making use of predefined camera motion patterns. There are also a number of different methods that do not rely on heuristics. These methods exploit variations between scenes found in broadcasted videos, like for instance the change of shot from a wide one to a close-up in a soccer goal celebration. [4] summarized soccer, basketball, and tennis videos by segmenting relevant events based on intensity variations detected in the color frames. [5] summarized broadcasted rugby videos by detecting the extrema in the optical flow so as to extract the frames with the highest action content.

In contrast, user-generated video summarization in general does not have a universal criterion to identify interesting events. Hence, many video summarization methods try to reduce the redundancy of lengthy videos rather than determining interesting events. Traditional methods uniformly sample frames or cluster them based on low-level features, such as color [6], to extract a brief synopsis of a lengthy video. Other types of summarization criteria are important objects [7], attention [8] and user preferences [9].

Recent methods use deep neural networks to automatically learn a criterion to model highlights. [10] extracted features from ground-truth video summaries to train a model for highlight detection. [11] used a set of both original videos and their textual summaries (generated via majority voting by multiple annotators) to train a model to find video highlights. Video titles [12], descriptions [13], and other side information [14] can also be used to learn a criterion to generate summaries. The aforementioned methods employed networks with CNNs and LSTMs, and these require a large amount of data for training. The generation of these types of large summarization datasets for training their network is non-viable for most researchers, and thus their models are built on pre-trained networks such as VGG [15] and GoogLeNet [16].

3. SYSTEM OVERVIEW

The proposed strategy comprises several processing blocks, as shown in Figure 1. The basic outline of this strategy is that, given an input video, corners or key points present in each frame are extracted and their motion is estimated with respect to the previous frame. These key points are filtered down through a series of techniques so as to keep only those that are more likely to belong to the foreground, which represents our region of interest. Resulting key points are grouped up by vicinity in order to reveal groups or objects in the foreground of each frame. An analysis of the evolution of these objects across frames is performed so as to detect candidate highlight events. Finally, these candidate events will be refined in order to output the final highlight events detected in the input video.

In the feature extraction and tracking block (section 4.1), the most prominent corners in the current frame are determined and their motion is estimated with respect to the previous frame. The output of this block is a set of key points along with a motion vector that contains information regarding the displacement of each key point.

The foreground feature segmentation block (section 4.2) identifies foreground key points in a probabilistic way, dependent on each key point matching history. If a key point is consistently identified across video frames, it is more likely to belong to the background.

In the region-based analysis block (section 4.4), foreground key points are grouped up in regions and each region is described by the foreground key points present in it. All regions present within a frame are stored for following blocks.

The temporal coherence analysis block (section 4.5) will apply once all frames of the input video have gone through the previous blocks. The information contained in every stored region will be used to link regions across frames. This way we can track how regions evolve across frames: how they appear, disappear, move, split or converge. The output of this block is a set of regions that are candidates to be a highlight event, which are stored for the following and last block.

In the event detection block (section 4.6), the information contained in each candidate event is analyzed in order to classify it as a highlight or not. Additionally, detected highlights are further processed, grouping up those that are close in time and thus, are likely be part of the same event. The resulting events are characterized by their probability of being a highlight and two user variables are used to filter down these events by duration and by probability of being a highlight. The output of this block are the highlight clips detected in the input video.

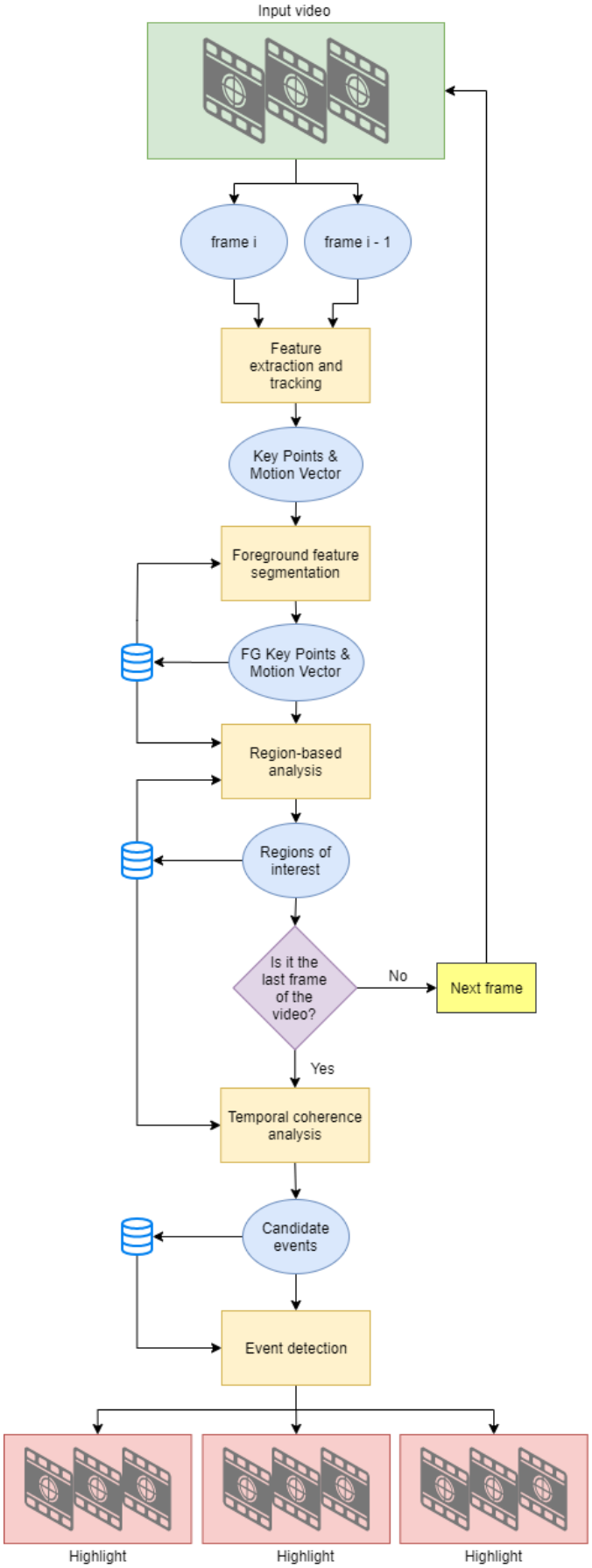

Figure 1. Block diagram of the proposed strategy. Round-edge blocks denote data, rectangular blocks denote processes and the diamond block indicates decisions.

4. PROPOSAL

4.1. FEATURE EXTRACTION AND TRACKING

This processing block allows to extract features from images and track them along a sequence, giving us an estimation of the motion present in each frame based in some interesting key points or features. For this purpose, RGB images are not actually needed as we will see in this section. Thus, we will work with grayscale representations of these.

4.1.1. FEATURE EXTRACTION

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are important features in the image, and they are generally termed as interest points or key points. Although corners are only a small percentage of the image, they contain the most important features in restoring image information, and they can be used to minimize the amount of processed data for computer vision applications.

We determine the most prominent corners in the grayscale frames as described by Shi-Tomasi [17], a corner detector based entirely on the Harris corner detector [18], with a slight variation in the corner selection criteria. The aim of Harris corner detector is to find little image patches or windows that generate very large variations in intensity when moved around. For each window found, an R-score is computed, and a threshold is applied to select only those that have a score above it.

For instance, let a window be located at a position (x,y) and the intensity of the pixel at this location be $I(x,y)$. If this window slightly shifts to a new location with displacement (u,v), the intensity of the pixel at this location will be $I(x+u,y+v)$. Hence, $I(x+u,y+v)$-$I(x,y)$ will be the difference in intensities of the window shift. For a corner, this difference will be very large. Thus, we maximize this term by differentiating it with respect to the X and Y axes. Let $w(x,y)$ be the pixels weights of a given window (uniform or Gaussian). Then, $E(u,v)$ is defined as the weighted sum of squared intensity differences for all pixels in a window:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \quad (1)$$

Now, computing $E(u,v)$ by the above formula would be really slow, but by using the Taylor series expansion of first order we can rewrite $I(x+u,y+v)$ as:

$$I(x+u,y+v) \approx I(x,y) + \frac{\partial I(x,y)}{\partial x}u + \frac{\partial I(x,y)}{\partial y}v = I(x,y) + I_x u + I_y v \quad (2)$$

Then,

$$E(u,v) = \sum_{x,y} w(x,y)[I(x,y) + I_x u + I_y v - I(x,y)]^2 = \sum_{x,y} w(x,y)[I_x u + I_y v]^2 \quad (3)$$

Matrix notation can be obtained by expanding $E(u,v)$ and rewriting the equation:

$$E(u,v) = \sum_{x,y} w(x,y)[{I_x}^2u^2 + {I_y}^2v^2 + 2I_xI_yuv]$$
$$E(u,v) \approx (u,v)M\begin{pmatrix}x\\y\end{pmatrix} \quad (4)$$

Where *M* is the structure tensor or second-moment matrix.

$$M = w(x,y)\begin{pmatrix}\sum_{x,y} I_x^2 & \sum_{x,y} I_xI_y\\ \sum_{x,y} I_xI_y & \sum_{x,y} I_y^2\end{pmatrix} \quad (5)$$

This matrix is derived from the gradient of a function and summarizes the predominant directions of the gradient in a specified neighborhood of a point, and the degree to which those directions are coherent. In other words, it captures the intensity structure of the local neighborhood.

The suitability of a window is determined by an R-score, which in the case of the Harris corner detector is calculated as follows:

$$R = \det(M) - k(trace(M))^2$$
$$\det(M) = \lambda_1\lambda_2 \quad (6)$$
$$trace(M) = \lambda_1 + \lambda_2$$

Where λ_1 and λ_2 are eigenvalues of *M*, and *k* is a tunable sensitivity parameter which value has to be determined empirically, although literature values between 0.04-0.06 have been reported as feasible.

Since the exact computation of the eigenvalues is computationally expensive, as it requires the computation of a square root, Harris and Stephens suggested the above-mentioned R-score. This algorithm does not actually compute the eigenvalue decomposition of the matrix *M* and instead it is sufficient to evaluate the determinant and trace of *M* to find corners. A large R-score indicates a corner, a negative value indicates an edge and values close to zero indicate a flat region.

Commonly, non-maximum suppression is performed at last in order to get the optimal R-score values to indicate corners, by finding the local maxima within a local neighborhood.

The Shi-Tomasi corner detector is almost identical to the Harris corner detector except for the corner selection criteria. Whereas Harris and Stephens suggested the above explained R-score, Shi and Tomasi suggested only the eigenvalues should be used to evaluate corners, because under certain assumptions, these are more stable for tracking. The R-score of Shi-Tomasi corner detector is determined by:

$$R = min\,(\lambda_1, \lambda_2) \quad (7)$$

In their paper, Shi and Tomasi demonstrated experimentally that this score criteria was much better. The proposed score identifies corners where a window R-score is greater than a certain predefined value and thus, the effect region for a point to be considered as a corner would be as shown in Figure 2.

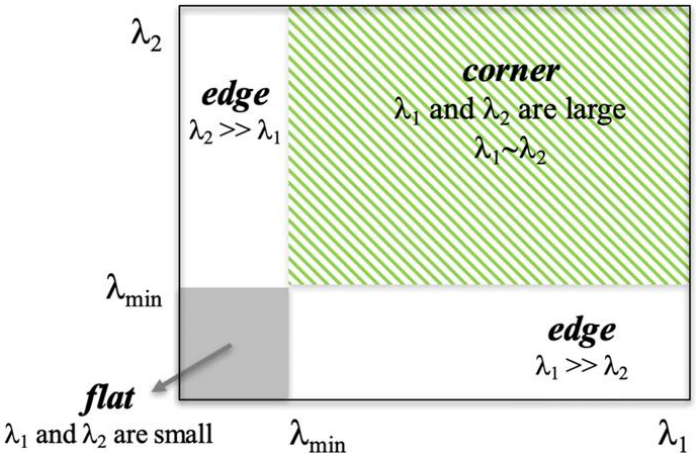

Figure 2. Shi-Tomasi region criteria for detecting corners.

Figure 3 shows the difference in results between both Harris and Shi-Tomasi corner detection algorithms. Detected corners are virtually identical, but the difference in the score criteria achieve better results with Shi-Tomasi algorithm.

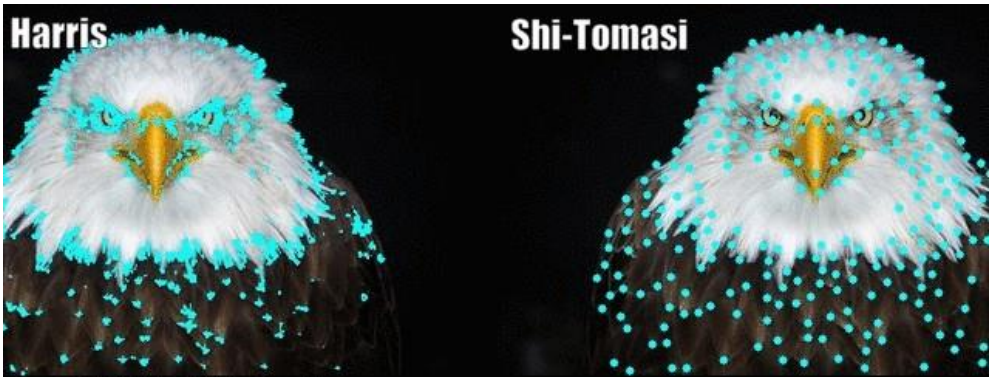

Figure 3. Example of the difference in corners obtained when using Harris and Shi-Tomasi corner detection algorithms.

In Figure 4 we see the resulting key points obtained after employing the Shi-Tomasi corner detection algorithm. These key points present a series of characteristics that made them perfect for the purpose of this work. The most important of these is that detected key points are well distributed all across the frame. People in the frame present enough key points to estimate their motions later in this block and the background also presents a significant number of key points, which will allow to differentiate foreground objects in a later block.

Several other feature descriptors were tested in order to compare their results, among others: SIFT, AKAZE, Harris and Shi-Tomasi. In Figure 5 we show a comparison between some of these algorithms. We finally opted for Shi-Tomasi corner detector as it was able to obtain a reasonable number of key points distributed all over the frame. Harris detection criteria makes it so that many more key points are detected, resulting in large redundant information that would slow down following processes. SIFT identified good set of key points but these were not well distributed across the frame and so were discarded.

Figure 4. Example of the key points detected with Shi-Tomasi algorithm for two different frames.

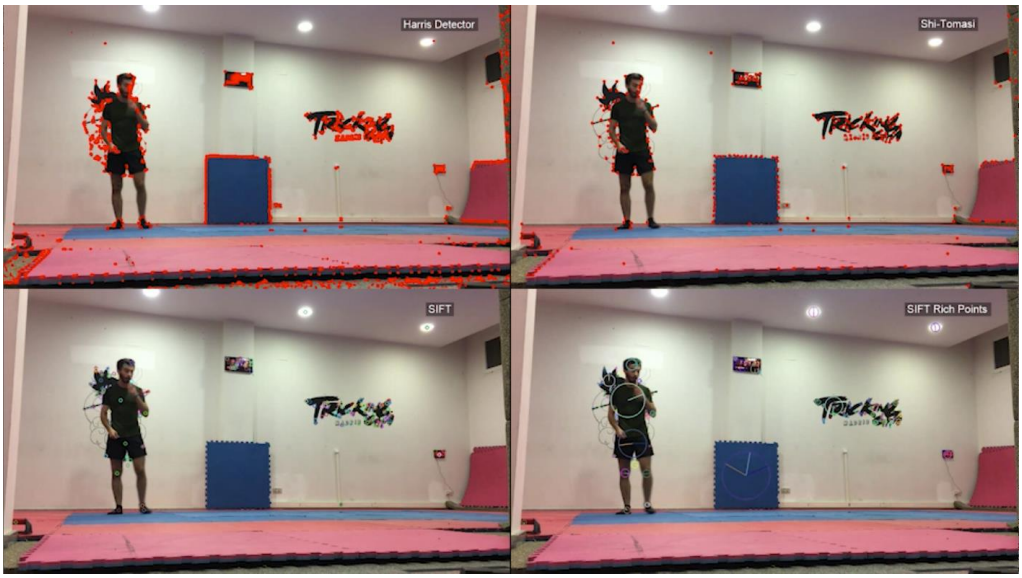

Figure 5. Comparison between key points obtained with different feature descriptors. On the top image corner key points detected with Harris corner detector. On the top right key points obtained with Shi-Tomasi corner detector. Both bottom images represent SIFT key points, on the left only the center of the key point are represented whereas on the right circles and lines add information about size and orientation detected with this algorithm.

4.1.2. TRACKING

Optical flow can be described as the pattern or apparent motion of image objects between two consecutive frames in a visual scene caused by the relative motion between an observer and the scene. We estimate the optical flow of the previously detected Shi-Tomasi corners making use of the iterative Lucas-Kanade method with pyramids for feature tracking purposes.

The Lucas-Kanade optical flow algorithm is a simple technique which can provide an estimate of the movement of interesting features in successive frames of a scene. We would like to associate a movement vector (u,v) to every such interesting pixel in the scene, obtained by comparing two consecutive frames. This algorithm makes some implicit assumptions:

- Two frames are separated by a small-time increment Δt, in such a way that objects have not displaced significantly (that is, the algorithm works best with slow moving objects).
- The images depict a natural scene containing textured objects exhibiting shades of gray (different intensity or brightness levels) which change smoothly.

The algorithm does not use color information in an explicit way, and it does not scan the second image looking for a match for a given pixel. It works by trying to guess in which direction an object has moved so that local changes in intensity can be explained.

Let I be a grayscale image and $I(x,y)$ be the intensity level at a pixel (x,y). The increase in brightness in the x and y directions is given by $I_x(x,y)$ and $I_y(x,y)$, which are the partial derivatives of the image I in both directions. The total increase in brightness after a movement by u pixels in the x direction and v pixels in the y direction can be represented as:

$$I_x(x,y)u + I_y(x,y)v \tag{8}$$

This matches the local difference in brightness, denoted by $I_t(x,y)$, so that:

$$I_x(x,y)u + I_y(x,y)v = -I_t(x,y) \tag{9}$$

Of course, a simple pixel does not usually contain enough structure useful for matching with another pixel. So, it is better to use a neighborhood of pixels, for instance a 3×3 neighborhood around the pixel (x,y). In that case we would have 9 linear equations for pixels in the neighborhood of $\Delta x=\{-1,0,1\}$ and $\Delta y=\{-1,0,1\}$:

$$I_x(x+\Delta x, y+\Delta y)u + I_y(+\Delta x, y+\Delta y)v = -I_t(+\Delta x, y+\Delta y) \tag{10}$$

These equations can be summarized in matrix notation as:

$$A\begin{pmatrix} u \\ v \end{pmatrix} = \boldsymbol{b} \tag{11}$$

Where A is a 9×2 matrix containing the rows $I_x(x+\Delta x,y+\Delta y)$ and $I_y(x+\Delta x,y+\Delta y)$, and $\boldsymbol{b}$ is the column vector containing the nine terms in $-I_t(x+\Delta x,y+\Delta y)$. This equation cannot be resolved exactly in the general case as the system has more equations than unknowns and thus, it is usually over-determined. The Lucas-Kanade method obtains a compromise solution by the least squares criteria. Namely, it solves the 2×2 system:

$$(A^T A)\begin{pmatrix} u \\ v \end{pmatrix} = A^T b$$

$$\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{x,t} I_x I_t \\ \sum_{y,t} I_y I_t \end{bmatrix} \tag{12}$$

And inverting $(A^T A)$ we get:

$$\begin{pmatrix} u \\ v \end{pmatrix} = (A^T A)^{-1} A^T b$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{x,t} I_x I_t \\ -\sum_{y,t} I_y I_t \end{bmatrix} \tag{13}$$

An optimal optical flow (u,v) satisfies Lucas-Kanade equations but its solution depends on how $(A^T A)$ is. As we can see from equation (5), $(A^T A)$ is the structure tensor or second-moment matrix. The solution given above is best when this matrix is invertible, which might not always be the case. For instance, if a pixel (x,y) is located in a region with no structure (for example, if I_x, I_y and I_t are all zero for all pixels in the neighborhood, as in a flat surface). Even if the matrix is invertible it can be ill conditioned, if its elements are very small and close to zero.

Since $(A^T A)$ is a symmetrical matrix, it can be diagonalized and rewritten in the form:

$$A^T A = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^T \tag{14}$$

Where λ_1 and λ_2 are the eigenvalues of the matrix and U is a unitary 2×2 matrix. If $(A^T A)$ is not invertible, either λ_1 or λ_2 or both, are zero. If one of them, or both, are very small, then the inverse matrix is ill-conditioned. This should sound familiar to the Harris corner detector criteria explained in section 4.1.1. Lucas-Kanade algorithm eliminates regions without structure by looking at the invertibility of the matrix $(A^T A)$ in an indirect way, that is, through the eigenvalues of this matrix.

To sum up, Lucas-Kanade original algorithm makes a best guess of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel we have two unknowns (u,v) and one equation (that is, the system is under-determined). We need a neighborhood in order to get more equations. Doing so makes the system over-determined and so we have to find a least squares solution. The least squares solution averages the optical flow guesses over a neighborhood. The result of the algorithm is a set of optical flow vectors distributed over the image which give an estimation idea of the movement of objects in the scene.

The main advantage of the algorithm is that for a neighborhood of fixed size the number of operations needed to compute $(A^T A)^{-1} A^T \boldsymbol{b}$ are constant. Therefore, the complexity of the algorithm is linear with the number of pixels examined in the image. Alternative algorithms that match similar regions using a neighborhood, and scanning the second image, have quadratic complexity.

However, as it has been explained in this chapter the Lucas-Kanade original algorithm assumes that all intensity changes can be explained by intensity gradients and so it fails when gradients are random or negligible (no structure, as in flat surfaces). It also fails when motions are not small or when points do not move like their neighbors (is the window size too large? what is the ideal window size?).

To deal with these problems an iterative application of the Lucas-Kanade algorithm with pyramids is needed [19]. The pyramidal implementation of this algorithm computes the optical flow in a coarse-to-fine iterative manner. Iterative coarse-to-fine optical flow is simply put, building image pyramids of different sizes for each image, and by computing the optical flow on each layer of the pyramid, get rid of the small motion constraint of the original algorithm. First, we start from the lowest resolution level. We use one iteration of Lucas-Kanade to estimate potential motion velocity at this level and then expand it to a higher resolution level as initial velocity for this level to apply Lucas-Kanade iteratively, as depicted in Figure 6.

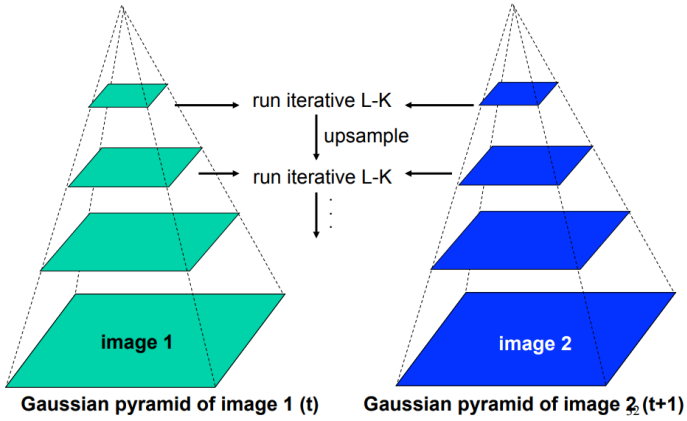

Figure 6. Overview of a generic pyramidal implementation of Lucas-Kanade algorithm.

Let I and J be two 2D grayscale images. The two quantities $I(x)=I(x,y)$ and $J(x)=J(x,y)$ are then the intensities of the two images at the location $\boldsymbol{x}=[x\ y]^T$, where x and y are the two pixel coordinates of a generic image point. For practical issues, the images I and J are discrete function (or arrays), and the upper left corner pixel coordinate vector is $[0\ 0]^T$. Let n_x and n_y be the width and height of the two images. Then the lower right pixel coordinate vector is $[n_x-1\ n_y-1]^T$.

Consider an image point $\boldsymbol{u}=[u_x\ u_y]^T$ on the first image I. The goal of feature tracking is to find the location $\boldsymbol{v}=\boldsymbol{u}+\boldsymbol{d}=[u_x+d_x\ \ u_y+d_y]^T$ on the second image J such as $I(\boldsymbol{u})$ and $J(\boldsymbol{v})$ are similar. The vector $\boldsymbol{d}=[d_x\ d_y]^T$ is the image velocity at the point under analysis, also known as its optical flow. Because of the aperture problem, it is essential to define the notion of similarity in a 2D neighborhood sense. Let ω_x and ω_y be two integers. We define the image velocity $\boldsymbol{d}$ as being the vector that minimizes the residual function ϵ defined as follows:

$$\epsilon(d)=\epsilon(d_x,d_y)=\sum_{x=u_x-w_x}^{u_x+w_x}\sum_{y=u_y-w_y}^{u_y+w_y}[I(x,y)-J(x+d_x,y+d_y)]^2 \qquad (15)$$

Following that definition, the similarity function is measured on an image neighborhood of size $(2\omega_x+1)\times(2\omega_y+1)$. This neighborhood will be also called integration window.

The two key components to any feature tracker are accuracy and robustness. The accuracy component relates to the local sub-pixel accuracy attached to tracking. Intuitively, a small integration window would be preferable in order not to smooth out the details contained in the images (i.e. small values of ω_x and ω_y). The robustness component relates to sensitivity of tracking with respect to changes of

lighting, size of image motion, etc. In particular, in order to handle large motions, it is intuitively preferable to pick a large integration window. There is therefore a natural tradeoff between local accuracy and robustness when choosing the integration window size. A pyramidal implementation of the Lucas-Kanade algorithm provides sufficient local tracking accuracy.

Let us define the pyramid representation of a generic image I of size $n_x \times n_y$. Let $I^0=I$ be the "zeroth" level image. This image is essentially the highest resolution image or raw image. The image width and height at that level are defined as $n_x^0=n_x$ and $n_y^0=n_y$. The pyramid representation is then built in a recursive fashion: compute I^1 from I^0, then compute I^2 from I^1, and so on. Let $L=1, 2, \ldots$ be a generic pyramidal level, and let I^{L-1} be the image at level $L-1$. Denote n_x^{L-1} and n_y^{L-1} the width and height of I^{L-1}. The image I^{L-1} is then defined as follows:

$$\begin{aligned} I^L(x,y) = {} & \frac{1}{4} I^{L-1}(2x,2y) \\ & + \frac{1}{8}[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)] \\ & + \frac{1}{16}[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) \\ & + I^{L-1}(2x+1,2y+1)] \end{aligned} \tag{16}$$

For simplicity in the notation, let us define dummy image values one pixel around the image I^{L-1} (for $0 \le x \le n_x^{L-1}-1$ and $0 \le y \le n_y^{L-1}-1$). Then, equation (16) is only defined for values of x and y such that $0 \le 2x \le n_x^{L-1}-1$ and $0 \le 2y \le n_y^{L-1}-1$. Therefore, the width n_x^L and height n_y^L of I^L are the largest integers that satisfy the two conditions:

$$n_x^L \le \frac{n_x^{L-1}+1}{2} \tag{17}$$

$$n_y^L \le \frac{n_y^{L-1}+1}{2} \tag{18}$$

Equations (16), (17) and (18) are used to construct recursively the pyramidal representations of the two images I and J: $\{I^L\}_{L=0,\ldots,Lm}$ *and* $\{J^L\}_{L=0,\ldots,Lm}$. The value Lm is the height of the pyramid. Practical values of Lm are 2,3,4; for typical image sizes, it makes no sense to go above a level 4. For example, for an image I of size 640×480, the images I^1, I^2, I^3 and I^4 are of respective sizes 320×240, 160×120, 80×60 and 40×30. The central motivation behind pyramidal representation is to be able to handle large pixel motions (larger than the integration window sizes ω_x and ω_y). Therefore, the pyramid height (Lm) should also be picked appropriately according to the maximum expected optical flow in the image

Recalling the goal of feature tracking, for a given point $\boldsymbol{u}$ in image I, we want to find its corresponding location $\boldsymbol{v}=\boldsymbol{u}+\boldsymbol{d}$ in image J, or alternatively find its pixel displacement vector $\boldsymbol{d}$ (see equation 15). For $L=0, \ldots, Lm$, define $\boldsymbol{u}^L=[u_x^L\ u_y^L]$, the corresponding coordinates of the point $\boldsymbol{u}$ on the pyramidal images I^L. Following the definition of the pyramid representation equations (16), (17) and (18), the vectors $\boldsymbol{u}^L$ are computed as follows:

$$\boldsymbol{u}^L = \frac{\boldsymbol{u}}{2^L} \tag{19}$$

The division operation in equation (19) is applied to both coordinates independently. Observe that in particular, $\boldsymbol{u}^0=\boldsymbol{u}$.

The overall pyramidal tracking algorithm proceeds as follows: first, the optical flow is computed at the deepest pyramid level *Lm.* Then, the result of the that computation is propagated to the upper level *Lm-1* in a form of an initial guess for the pixel displacement (at level *Lm-1*). Given that initial guess, the refined optical flow is computed at level *Lm-1*, and the result is propagated to level *Lm-2* and so on up to the level 0 (the original image).

Let us now describe the recursive operation between two generics levels $L+1$ and L in more mathematical details. Assume that an initial guess for optical flow at level L, $\boldsymbol{g}^L=[g_x^L\ g_y^L]^T$, is available from the computations done from level L to level $L+1$. Then, in order to compute the optical flow at level L, it is necessary to find the residual pixel displacement vector $\boldsymbol{d}^L=[d_x^L\ d_y^L]^T$ that minimizes the new image matching error function ϵ^L:

$$\epsilon^L(d^L)=\epsilon^L\left(d_x^L,d_y^L\right)=\sum_{x=u_x^L-w_x}^{u_x^L+w_x}\sum_{y=u_y^L-w_y}^{u_y^L+w_y}[I^L(x,y)-J^L\left(x+g_x^L+d_x^L,y+g_y^L+d_y^L\right)]^2 \qquad (20)$$

Observe that the window of integration is of constant size $(2\omega_x+1)\times(2\omega_y+1)$ for all values of L. Notice that the initial guess flow vector $\boldsymbol{g}^L$ is used to pre-translate the image patch in the second image J. That way, the residual flow vector $\boldsymbol{d}^L=[d_x^L\ d_y^L]^T$ is small and therefore easy to compute through a standard Lucas-Kanade step.

Then, the result of this computation is propagated to the next level *L-1* by passing the new initial guess $\boldsymbol{g}^{L-1}$ of expression:

$$\boldsymbol{g}^{L-1}=2(\boldsymbol{g}^L+\boldsymbol{d}^L) \qquad (21)$$

The next level optical flow residual vector $\boldsymbol{d}^{L-1}$ is then computed through the same procedure. This vector, computed by optical flow computation, minimizes the functional $\epsilon^L(\boldsymbol{d}^{L-1})$ (equation 20). This procedure goes on until the finest image resolution is reached ($L=0$). The algorithm is initialized by setting the initial guess for level *Lm* to zero (no initial guess is available at the deepest level of the pyramid):

$$\boldsymbol{g}^{Lm}=[0\ 0]^T \qquad (22)$$

The final optical flow solution $\boldsymbol{d}$ (refer to equation 15) is then available after the finest optical flow computation:

$$\boldsymbol{d}=\sum_{L=0}^{Lm}2^L\boldsymbol{d}^L \qquad (23)$$

The clear advantage of a pyramidal implementation is that each residual optical flow vector $\boldsymbol{d}^L$ can be kept very small while computing a large overall pixel displacement vector $\boldsymbol{d}$. Assuming that each elementary optical flow computation step can handle pixel motions up to d_{max}, then the overall pixel motion that the pyramidal implementation can handle becomes $d_{max\ final}=(2^{Lm+1}-1)d_{max}$. For example, for a pyramid depth of $Lm=3$, this means a maximum pixel displacement gain of 15. This enables large pixel motions, while keeping the size of the integration window relatively small.

The implementation steps of this algorithm are summed up in Figure 7 and in the pseudo-code of Figure 8 and are as follow:

– Top Level.
  - Apply Lucas-Kanade to get a flow field representing the flow from one frame to another.
  - Apply this flow field to warp the first frame toward the second one.
  - Rerun Lucas-Kanade on the new warped image to get a flow field from it to the second frame.
  - Repeat until convergence.

– Next Level.
  - Upsample the flow field to the next level as the first guess on the flow at that level.
  - Apply this flow field to warp the first frame toward the second one.
  - Rerun Lucas-Kanade and warping until convergence.

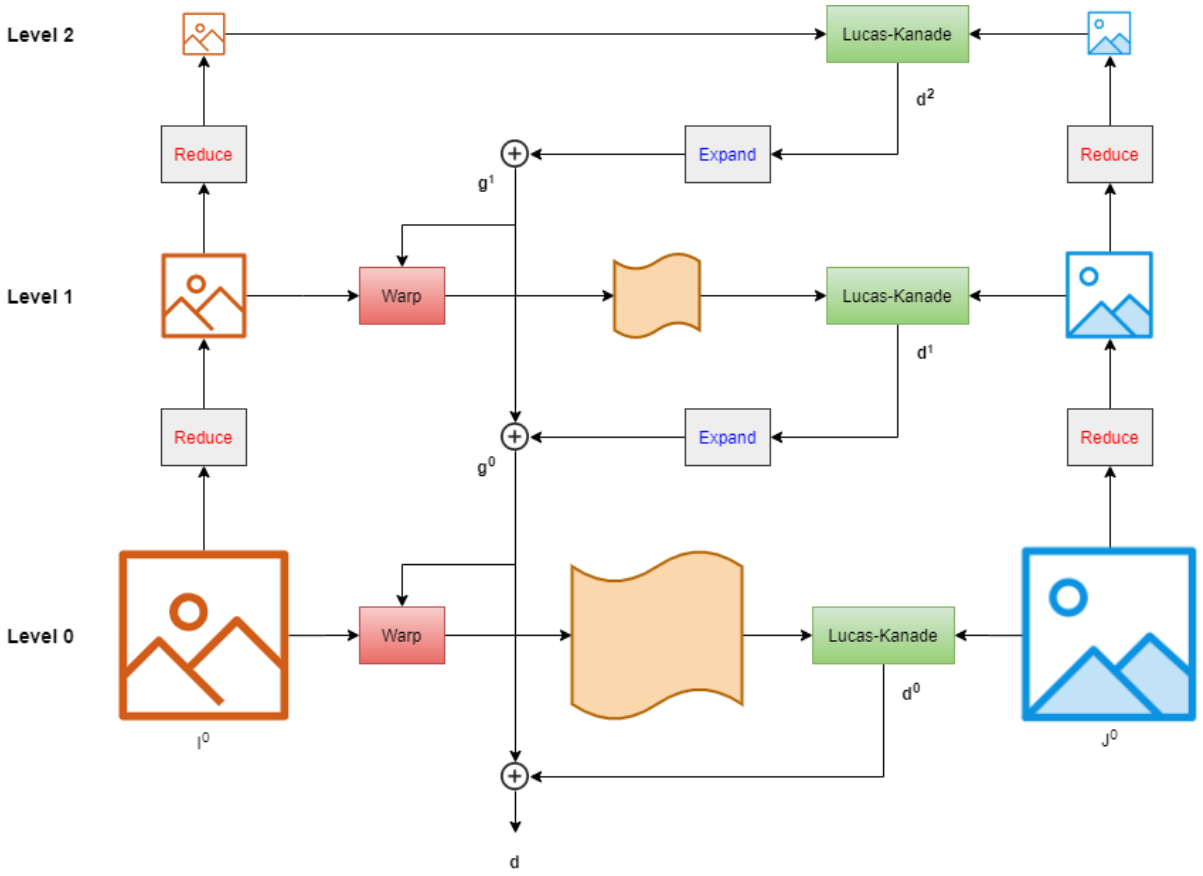

Figure 7. Steps in a generic Lucas-Kanade pyramidal implementation.

Goal: Let $\mathbf{u}$ be a point on image I. Find its corresponding location $\mathbf{v}$ on image J

Build pyramid representations of I *and* J: $\{I^L\}_{L=0,\dots,L_m}$ and $\{J^L\}_{L=0,\dots,L_m}$

Initialization of pyramidal guess: $\mathbf{g}^{L_m} = [g_x^{L_m} \;\; g_x^{L_m}]^T = [0 \;\; 0]^T$

for $L = L_m$ **down to** 0 **with step of** -1

Location of point $\mathbf{u}$ *on image* I^L: $\mathbf{u}^L = [p_x \;\; p_y]^T = \mathbf{u}/2^L$

Derivative of I^L *with respect to* x: $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$

Derivative of I^L *with respect to* y: $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$

Spatial gradient matrix: $G = \displaystyle\sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)\,I_y(x,y) \\ I_x(x,y)\,I_y(x,y) & I_y^2(x,y) \end{bmatrix}$

Initialization of iterative L-K: $\overline{\nu}^0 = [0 \;\; 0]^T$

for $k = 1$ **to** K **with step of** 1 (or until $\|\overline{\eta}^k\| <$ accuracy threshold)

Image difference: $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$

Image mismatch vector: $\overline{b}_k = \displaystyle\sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\,I_x(x,y) \\ \delta I_k(x,y)\,I_y(x,y) \end{bmatrix}$

Optical flow (Lucas-Kanade): $\overline{\eta}^k = G^{-1}\,\overline{b}_k$

Guess for next iteration: $\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k$

end of for-loop on k

Final optical flow at level L: $\mathbf{d}^L = \overline{\nu}^K$

Guess for next level $L-1$: $\mathbf{g}^{L-1} = [g_x^{L-1} \;\; g_y^{L-1}]^T = 2\,(\mathbf{g}^L + \mathbf{d}^L)$

end of for-loop on L

Final optical flow vector: $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$

Location of point on J: $\mathbf{v} = \mathbf{u} + \mathbf{d}$

Solution: The corresponding point is at location $\mathbf{v}$ on image J

Figure 8. Pseudo-code of the pyramidal implementation of Lucas-Kanade.

Figure 9 displays the key points previously detected in this block along with their motion estimations. It can be appreciated that lines now appear where we only had dots before. This figure allows to visually understand the motions detected when using a pyramidal implementation of Lucas-Kanade algorithm. For each key point we have a motion vector that describes its motion and it can be noted in the figure that people performing passes indeed present larger motion vectors.

Figure 9. Example of the motions estimated using a pyramidal implementation of Lucas-Kanade algorithm for two different frames.

4.2. FOREGROUND FEATURE SEGMENTATION

The goal for this step is to obtain a set of foreground key points which most likely belong to the foreground object of interest, which is achieved by identifying key points across video frames in a probabilistic way, as we discuss below.

The final aim of this project is to detect highlights in a scenario with a stationary camera where highlights consist on athletes performing passes, which are characterized by a lot more movement than other irrelevant events. Given the previous statement it is intuitively sensible to segment foreground features, as it reduces the amount of data to process and let us focus on how features present in the region of interest evolve.

4.2.1. BACKGROUND SUBTRACTION

Similar to [20], we provide a probabilistic way to update the probability of assigning an extracted key point as foreground, depending on its key point matching history. As opposed to [20], this work contemplates a stationary camera, so, if a key point is consistently identified across video frames, it is more likely to belong to the background object.

As the initialization stage of this step, all pixel probabilities $f_{point_{BG}}(x, y, t)$ within a frame of belonging to background are set to a threshold that will separate foreground from background, since we have no prior knowledge of whether these points belong to one or the other. Background point probability function is defined as:

$$f_{point\_BG}(x, y, t) = \begin{cases} f_{point\_BG}(x, y, t-1) \cdot \lambda + (1 - \lambda), & p_{x,y}^{t} \exists \\ f_{point\_BG}(x, y, t-1) \cdot \lambda, & p_{x,y}^{t} \nexists \end{cases} \quad (23)$$

Where λ is learning factor and is set to 0.95 as suggested in [21] and $p_{x,y}^{t}$ represents a detected key point in coordinates (*x*,*y*) and frame *t*. The above equation allows to assign new detected key points to either background (if the accumulated probability function is high) or foreground (if it is low), based on that point history.

As for the scenario of this project (stationary camera), it is common to detect more corners in the background as objects move slower or nothing at all and thus, appear sharper than moving objects in the foreground. By keeping track of where key points extracted in the previous processing block appear, we can determine if they belong to the foreground or the background. For instance, in a scenario with only one athlete performing and a background rich in texture, key points in the background will be consistently detected in the same locations across frames, whereas key points in the athlete will be detected in different locations across frames.

Based on the optical flow estimation commented in previous section 4.1.2, key points assigned to foreground are further filtered by removing those which displacements are small, as most cameras introduce some noise that can cause apparent movement in a stationary background point.

In Figure 10 we present the resulting foreground key points, along with their motion estimations, obtained through the proposed method. Results are quite good and do an overall good job at identifying foreground key points, which are represented by the green lines. What appears to be red and blue dots are in fact lines as well, but their motion vectors are so short they might be interpreted as just dots. Of these, red lines represent key points identified in the background and blue lines represent key points that although are identified in the foreground, present too short motions and thus, are not kept as they can be interpreted as a consequence of image noise. For instance, in this figure we see that the

blackboard hanging on the right wall presents a lot of texture that, although static to the human eye, can present some apparent motion when image noise is present and thus, blue lines are frequently detected.

Figure 10. Example of the foreground key points identified for two different frames using the proposed method.

4.3. PRELIMINARY EVENT DETECTION

In the previous section, we obtained a set of foreground key points along with their motion vectors for each video frame. However, the goal of this project is not to determine highlight points within a frame but rather highlight events within a video. The difference between highlight point and event in this context is simple, a point is a spatial location within a frame, and it is described by its motion. Events on the other hand, are temporal locations within a video, and are described by the frames that form them.

So, in order to detect highlight events, we must first analyze the information contained in the foreground key points previously stored. For didactic purposes, Figure 11 shows a variation of the block diagram of the proposed strategy of Figure 1 where event detection is performed directly on these key points, without the previous identification of regions of interest and candidate events that appear in the original diagram (see section 3). The proposed strategy do not perform event detection directly on these key points, but the results obtained with them motivate the existence of the following blocks (see section 4.4 and section 4.5) at the same time that illustrate how motion information obtained so far in the chain can prove to be useful for detecting highlight events.

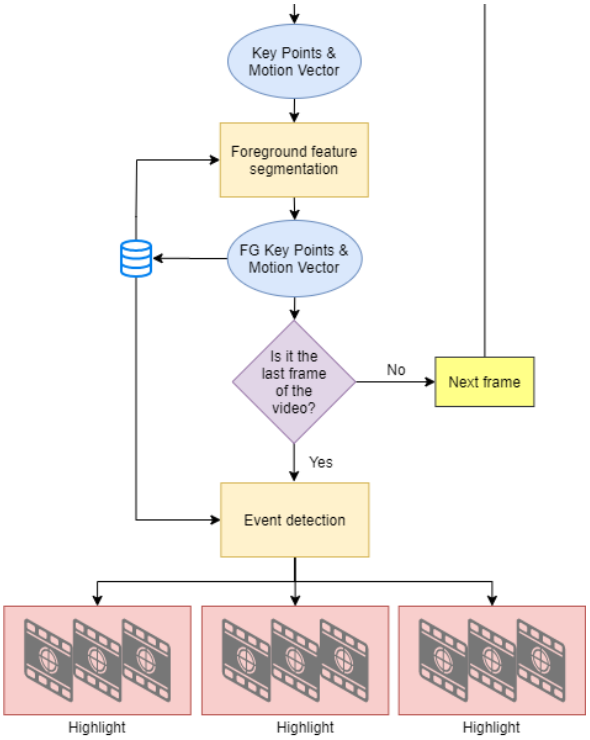

Figure 11. Block diagram variation of the proposed strategy. Round-edge blocks denote data, rectangular blocks denote processes and the diamond block indicates decisions

Key points are described, as we have seen in the previous sections, by their motion estimation. In this section we assess how this information can be used to classify a frame as part of a highlight. In order to illustrate the information that is being managed, let us set an example input video, for which frames have already passed through the previous processing blocks, resulting in a set of foreground key points for each frame, along with their motion vectors.

Figure 12 depicts how the number of key points evolve over frames, taking into account all foreground key points identified in the previous processing block for the example set. The blue line represents the total number of key points present in each frame. The red and green lines represent the mean of the last 1 and 10 seconds respectively, which serve to smooth the curve and compare the difference between a short and a medium-term temporal segment. In this figure we can appreciate that the number of key

points vary between 0 and 90 for each frame, which is a significant smaller value than that we would have gotten if taking all key points detected in the first processing block. The variation in number of key points over frames is fast and not much information can be interpolated from them alone.

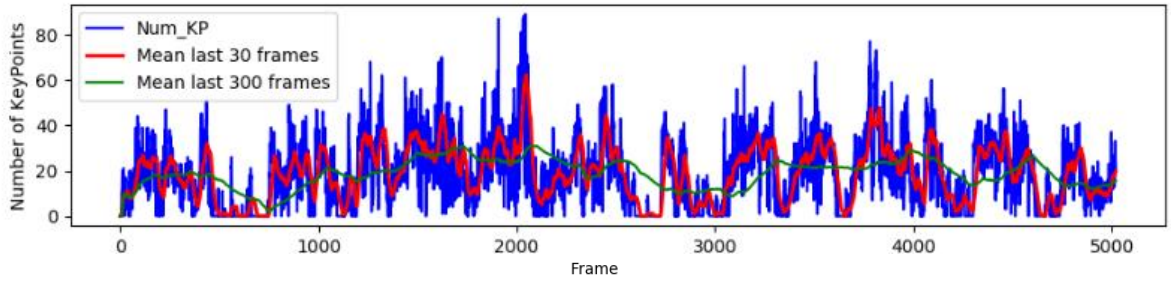

Figure 12. Graphic showing the number of key points identified in each frame, taking into account all candidate regions.

Now, Figure 13 shows the total motion present in each frame as the sum of all motion vectors magnitudes for all foreground key points in it, and the same color legend of the previous figure is followed. This figure provides information that can be better understood than that of the previous one. We can appreciate frames that appear to have a lot more motion than others, and by comparing the red and green lines we can detect events that present more motion than previous ones.

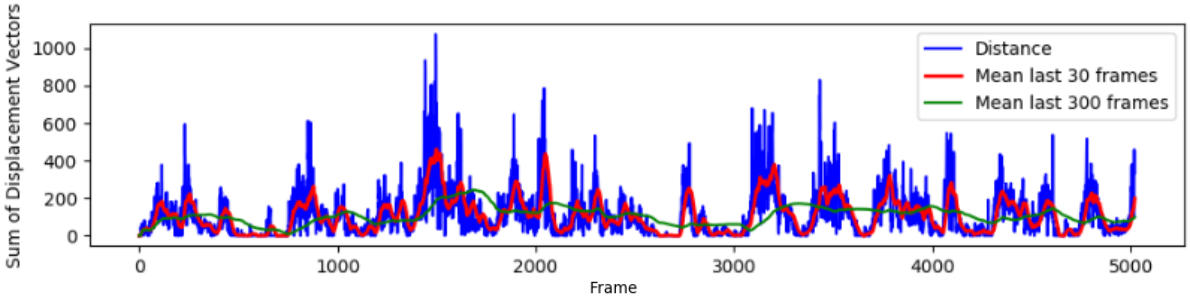

Figure 13. Graphic showing the sum of magnitude motions estimated in each frame, taking into account all candidate regions.

However, we can compute a normalized motion that relates both previous figures and better summarizes the motion information of the foreground key points, as it can be seen in Figure 14. Information provided in this figure takes into account the number of key points and their motion estimations and combines them. It can be immediately appreciated that this figure shows a more coherent and stable graphic than the previous two, and by comparing the means of the last 1 and 10 seconds we can better tell when an event is more relevant in motion than the previous ones.

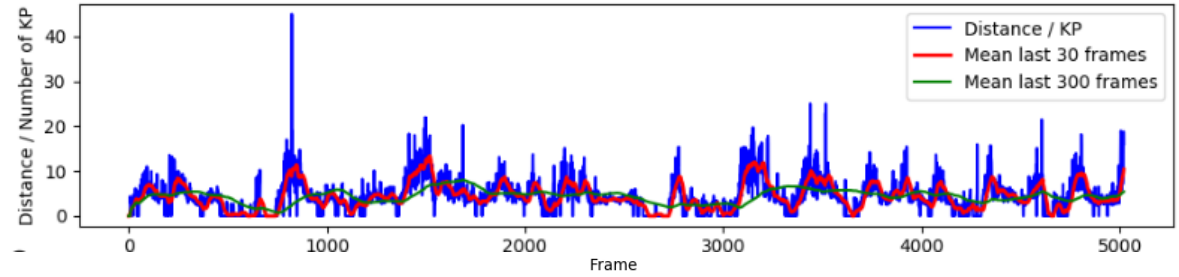

Figure 14. Graphic showing the normalized motion estimated in each frame, taking into account all candidate regions.

There are many potential methods that could be used to detect highlight events from this information. For instance, we could determine that every frame where the red line of the previous figure (representing the mean normalized motion of the last 1 second) is above the green line (representing the mean normalized motion of the last 10 seconds) is a highlight. We could also set a certain threshold to determine how much different a highlight event should be with respect to previous events. In Figure 15 we can see the results that would be obtained by determining highlights as those events where the last second event shows larger normalized motions than the normalized motion of the previous 10 seconds. This figure presents 2 horizontal bars, where the top one corresponds to the ground truth of the events in the input video and the lower bar corresponds to the identified events. Events in this figure are represented as vertical color bars, red indicates no highlight, green indicates highlight and blue indicates uncertainty. At first glance, we can see the detected events are quite similar to those of the ground truth for this very simple method.

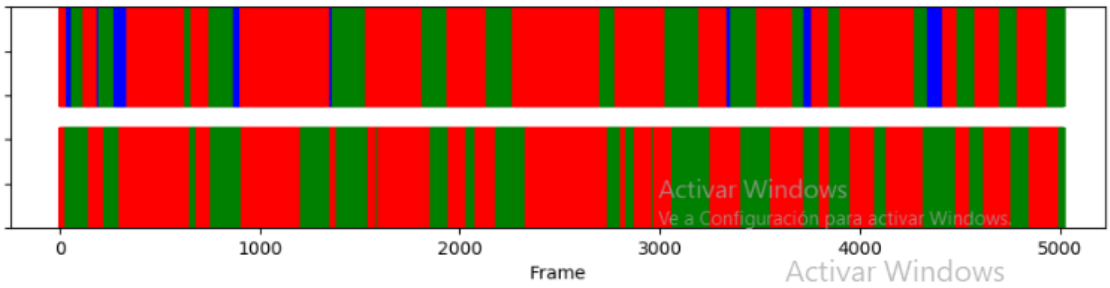

Figure 15. Ground truth and detected events. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

The methods and results explained so far could be used for real time applications because events are detected only based on previous frames, and the results could simply be tuned in a later processing stage (as will be explained in section 4.6), but still working in real time. The strategy we propose however, is aimed at off-line applications as it incorporates a temporal coherence analysis block (see section 4.5) that studies the evolution of previously stored regions of interest (see section 4.4) and links them across frames, allowing to base our results in the information contained in the regions identified in all frames and enabling us to get rid of some of the false positive detections that can be seen in Figure 15.

4.4. REGION-BASED ANALYSIS

This processing block is based on the underlying idea that motion of foreground objects present within a frame can provide more robust and reliable information than that provided by the sparse key points that form each object. In the previous processing block we obtained a set of foreground key points along with their motion estimations for a frame. In this section, we will discuss how regions of interest are identified within the frame. These regions of interest will be stored for a later post-processing stage that will analyze their evolution across frames (see section 4.5).

4.4.1. REGION DETECTION

At this point in the processing block chain, we have a set of key points scattered all across the foreground of the frame. The motion analysis of all individual key points would give global information about the foreground, that is, how much motion exists in the foreground. However, this information is too general for classifying highlights. For instance, a frame with 10 slow-moving objects could present the same motion as a frame with 1 fast-moving object. That is why this block is in charge of grouping up in regions key points present within the frame, each region being described by the sum of all individual key points that it contains. Following the previous example, this would enable us to identify 10 regions in the first frame and 1 in the other, as well as how much motion is present in each of those regions.

For this purpose, the frame is divided in a grid of cells, where these are large enough to be able to represent a region but small enough so that they provide local information. Each cell is described by the number of key points present in it, and by the sum of their motions. In other words, foreground key points are mapped into these cells, enabling us to summarize all information in a more compact and efficient way. This would be equivalent to creating a grid matrix $G(i,j)$ of dimensions $m \times n$, where each element (i,j) represents a cell, which contains the number of key points and motion present in it. Figure 16 shows how foreground key points are mapped to their corresponding cells in a 15×10 grid matrix.

Figure 16. Example of how key points are mapped into their corresponding cells.

Let us consider an active cell in the grid that one that contains at least one foreground key point (along with its motion) and an inactive one that does not contain any key point. This approach is equivalent to a binary version of the grid matrix $G_B(i,j)$ as shown in Figure 17, where 1's (or whites) represent active cells and 0's (or blacks) inactive ones. Active cells in the grid are grouped up by vicinity in an iterative manner. For an active cell of the grid $G_B(i,j)$, neighbors are those active cells adjacent either horizontally, vertically or diagonally. We iterate through each cell in $G_B(i,j)$, starting from $(0,0)$ up to (m,n). For each cell (i,j), we check if it is active and if it has been visited before. If it is active and it has not been visited before, we mark that cell (i,j) as visited and proceed with a depth first search (DFS) at that cell.

DFS is a recursive function that iterates through each cell in the set of neighbors of cell (*i*,*j*). If a cell of the set of neighbors, for instance (*i+1*,*j*), is active and has not been visited, it is marked as visited and a recursive call to DFS is done for that cell (*i+1*,*j*). This process runs recursively until it gets to a cell for which its set of neighbors are either inactive or have already been visited. Then, it returns to the previous cell under analysis, for which it continues to run on the remaining set of neighbors. This process allows to group up active cells that are adjacent in any direction with one another. In the example of Figure 16, this method would find 2 regions in the first frame and only 1 in the second frame, as shown in Figure 17.

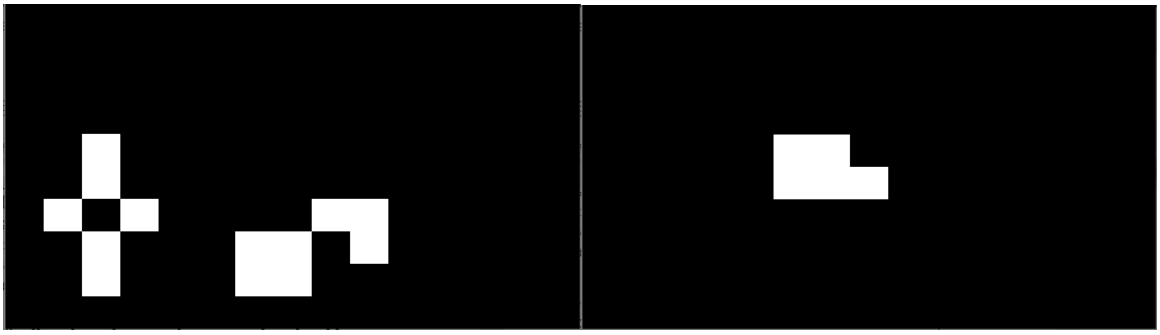

Figure 17. Active cells grid of the example set.

This process identifies multiple foreground active regions in the frame. Each region being described by the cells (*i*,*j*) that compose it, by the number of foreground key points that fall within them and by the total amount of motion that they present. When taking motion into account we do not consider its direction but only its magnitude, as we are only interested in how much motion exists in that region. This information is simple yet effective for representing the motion of a region as it will be discussed next.

Intuitively, a region containing many key points but with a small overall motion might correspond to a slow-moving object. Whereas a region with few key points but overall large motion might correspond to a fast-moving object. This is because fast-moving objects are commonly blurrier than objects moving slower, and thus, less key points will be extracted from them in the first processing block and the remaining key points are likely to present larger motions. So, in order to be able to compare motion amongst different regions we must first normalize it. The normalized motion of a region is the average displacement of the motion vectors contained within it. This variable takes high values for regions with large motions and few key points, and low values for regions with small motions and many key points. In summary, it allows for characterizing each region's motion. An example is shown in Figure 18.

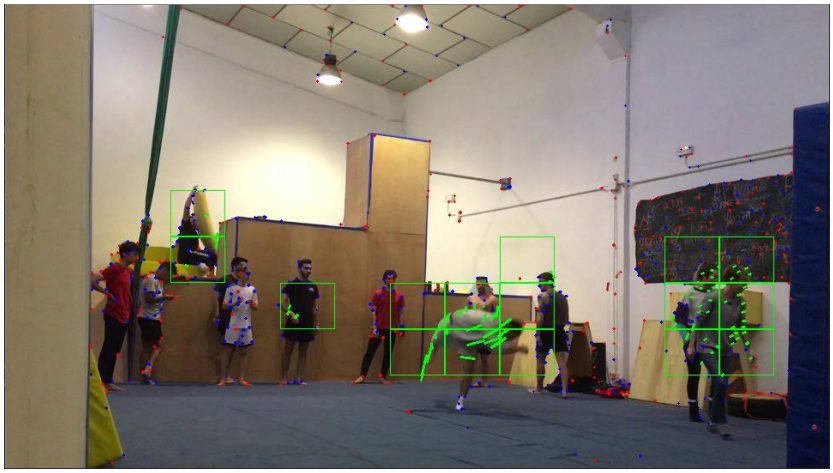

Figure 18. Example of identified regions using the proposed method.

In the previous figure we can see 4 regions, which information is described in Table 1. Regions from left to right correspond to a person moving at the back of the scene, the waiving of a hand of another person, a person performing a pass, and two people walking together.

Table 1. Regions description.

Region	Cells	Nº of key points	Motion	Normalized motion
1	(3,4)/(3,5)	4	22.55	5.64
2	(5,6)	3	11.70	3.90
3	(7,6)/(7,7)/(8,6)/(8,7)/(9,5)/(9,6)/(9,7)	26	273.11	10.50
4	(12,5)/(12,6)/(12,7)/(13,5)/(13,6)/(13,7)	46	130.40	2.83

It can be appreciated that regions 3 and 4 seem much more relevant than the other two regarding their number of key points and motion, which coincides with what we can visually tell from Figure 18. Although, just by looking at the number of key points and motion columns it would be difficult to compare these regions. However, the normalized motion column does a great job at summarizing the information needed for this purpose. Region 3 having half as many points as region 4, but an overall motion twice as large, presents a normalized motion almost four times greater than region 4.

To sum up, this processing block identifies regions of interest in each frame and characterizes them based on their motion features, information that will be stored for a later post-processing stage (see section 4.5) that will analyze their behavior over time.

4.4.2. REGION FILTERING

Regions of interest can be further filtered at this processing block and prior to any post-processing stage in order to save some computational cost at later processing blocks, storing only regions of interest that are good candidates to be a part of a highlight. This is done under two simple assumptions: good candidate regions are at least of a certain size and are likely to be located around the same area than a good candidate region of a previous frame. Previously detected regions of interest that do not fulfill these assumptions are discarded, and only those that do fulfill them are stored.

The first assumption is easy to interpret. A highlight region must at least be of a certain size, or in other words, its region must contain at least certain cells. Cells are dimensioned large enough so that they can represent a region but not large enough so that they can represent a region of interest by themselves. Thus, it makes sense to remove those detected regions too small to feature a person (for instance, 2) as they surely do not represent a good candidate region to be a part of a highlight, which usually entails long motion vectors that attach several different cells. An example of filtering of small regions is shown in Figure 19, where we can see 4 different regions. One of them contains several cells whereas the others only amount for one or two cells. After filtering, only one region would be retained.

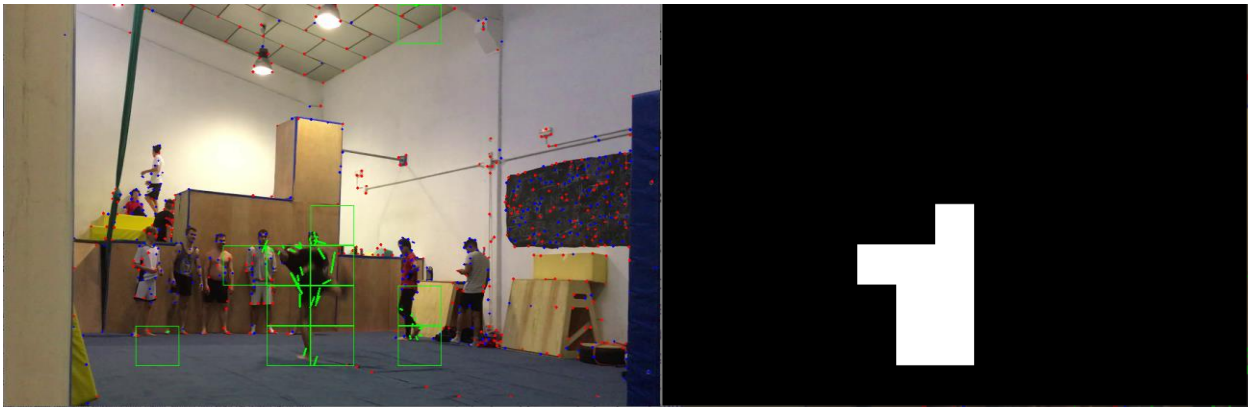

Figure 19. Example of region filtering by dimensions.

The second assumption allows further filtering these, removing sporadic regions that have no cells neighboring with good candidate regions of the previous frame. That is, regions of interest that appear at a location in a frame for which there are not any good candidate regions in the previous frame are discarded. The underlying idea is that highlight regions will smoothly evolve along frames, so it is highly unlikely for a highlight region to appear at a location in a frame and in the next frame be at a completely different location. This filter allows removing regions that appear sporadically over frames by seeking neighbors with the good candidate regions identified and stored for the previous frame. It is important to remark that when processing a current frame, this filter applies to the set of regions identified in it after filtering them by size. So, when comparing these candidate regions to those of the previous frame, we are comparing the regions of the current frame that have passed the size filter with those regions of the previous frame that have passed both filters and thus, represent good candidate regions.

Let us imagine two consecutive frames, where the previous frame is shown in Figure 20 and the current frame is shown in Figure 21. Figure 20 shows the frame with its identified regions on the left, and on the right, we have the result after applying both filters, that is, the good candidate regions identified for the previous frame.

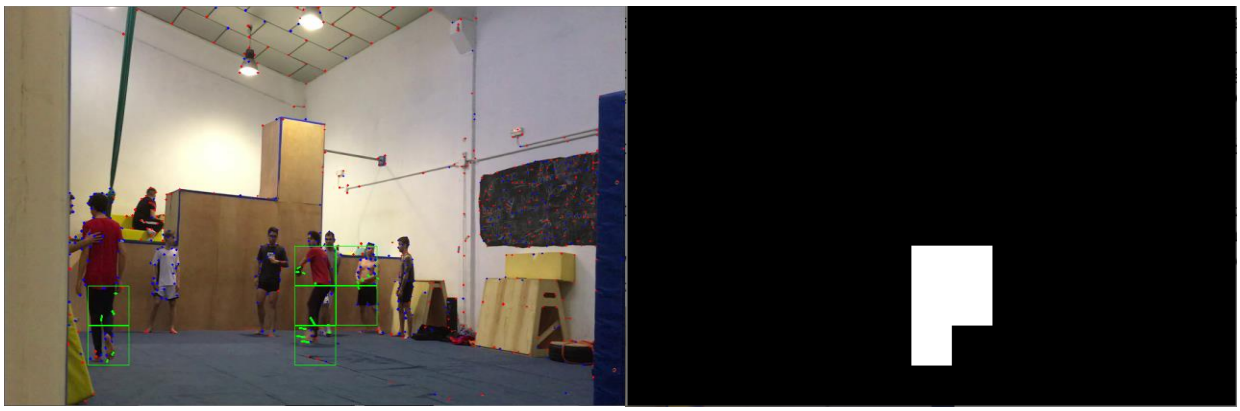

Figure 20. Example of a frame that presents a set of active cells on the left, and on the right, the good candidate regions identified.

Figure 21, however, shows the current frame with its identified regions on the left, but on the right, we have the result after applying only the size filter.

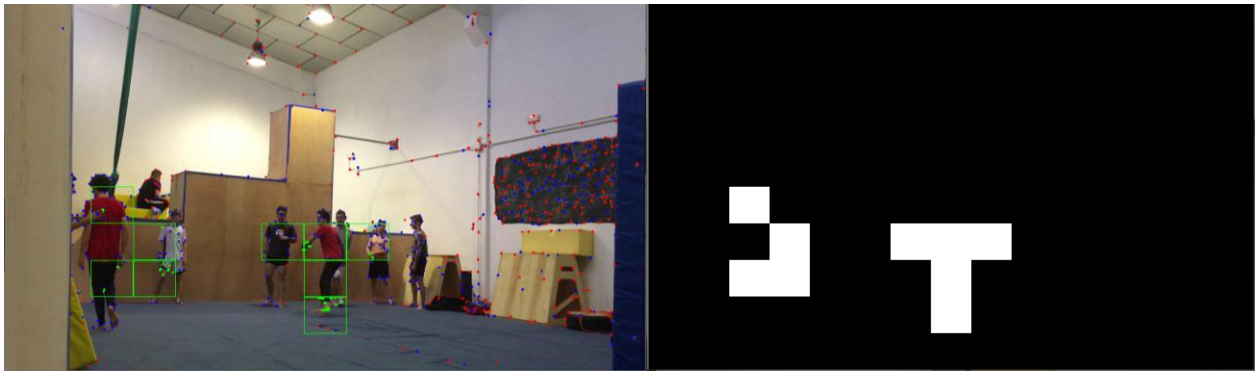

Figure 21. Example of a frame that presents a set of active cells on the left, and on the right, the candidate regions that have passed the size filter.

Now, when applying the second filter to the current frame, we compare it to the good candidate regions of the previous frame. So, if a region of the current frame has passed the filter size and also has neighboring cells with a good candidate region of the previous frame, it is considered to be a good candidate region and it is stored for next iterations of this block as well as for the following block.

In Figure 22 we can see that the region of the current frame that has passed the first filter, has neighboring cells with a good candidate region of the previous frame, and thus, supposes a good candidate region for the current frame. Regions that have passed the size filter but do not have neighboring cells with previous good candidate regions will be discarded, preventing sporadic regions from being stored. This also implies that a highlight region in a frame will not be identified as a good candidate the first time it appears, but it will for all future frames and hence it is not a drawback as highlights usually encompass multiple frames.

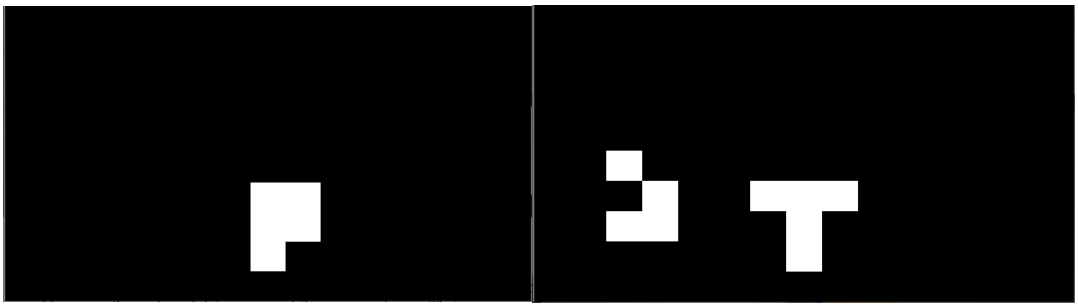

Figure 22. Comparison side by side of the good candidate regions of the previous frame on the left, and on the right, the regions of the current frame that have passed the size filter.

The good candidate region identified in the current frame after both filters is shown in Figure 23 and it will be stored for next iterations.

Figure 23. Good candidate regions identified for the current frame.

The output of this processing block is a set of regions of interest for each frame of the input video that are good candidate regions. This method allows summarizing an entire video sequence in just a list of regions that store all relevant motion information. Figure 24 shows a general example of the output of this processing block.

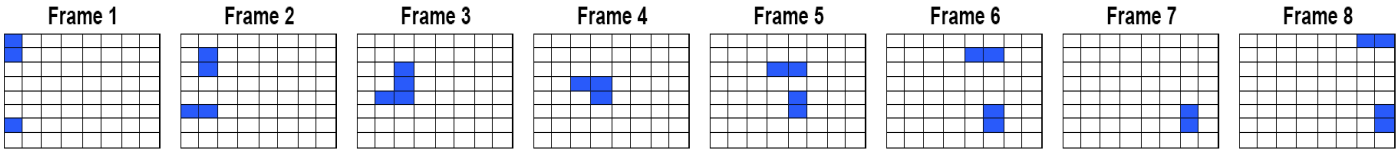

Figure 24. Output example of this processing block, formed by a list of frames, each containing a variable number of good candidate regions.

4.5. TEMPORAL COHERENCE ANALYSIS

This processing block performs as a post-processing stage before the final event detection of the proposed strategy. This block starts once an entire video clip has gone through all previous blocks and it takes as input the list of regions of interest of each frame. The information contained in each stored region is used to link regions of different frames following a similar procedure to that described in section 4.4.2 for the second filter. Doing so enables us to track the evolution of regions across frames. That is, how regions appear, disappear, move, split or join along frames. With this information we can better decide which region of the interest set supposes the best candidate region for each frame as it will be further explained along this section.

Let us set an example in order to better understand the workflow of this block. Figure 25 represents the candidate regions identified for each frame in the previous processing block. A human being can tell that 2 different regions can be appreciated along these 8 frames. One region appears in the top-left corner of the first frame and makes its way to the bottom-right corner of the last frame. And another region appears in the bottom-left corner of the first frame and analogously, makes its way to the top-right corner of the last frame. However, there is a difference between these 2 regions, and it is that the second region disappears in the 7th frame. This could potentially happen as consequence of different factors, i.e. foreground key points along with their motions have been identified in the frame at that location and grouped up forming a region of interest, but the size of this region has not passed the size filter that determines good candidate regions.

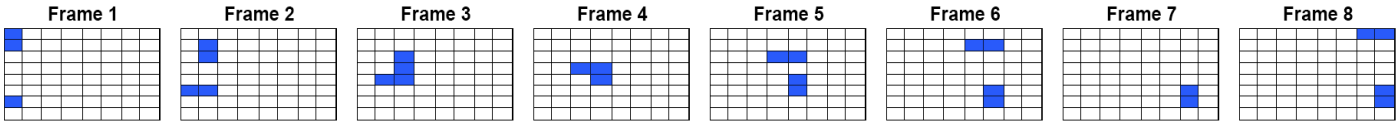

Figure 25. Example of a set of candidate regions identified in the previous processing block. Regions are represented in blue.

The purpose of this block is linking regions across frames. Now, let us dive into how this process is done. It can be appreciated in Figure 25 that different frames have different number of regions and at different locations. Over time (frames) regions may appear, disappear, move, split or converge, and it is important that we can track this evolution as we will see later in this section. For this purpose and following a similar approach to that described in section 4.4.2, we seek neighboring regions between frames. Figure 26 shows the neighborhood of each region in purple.

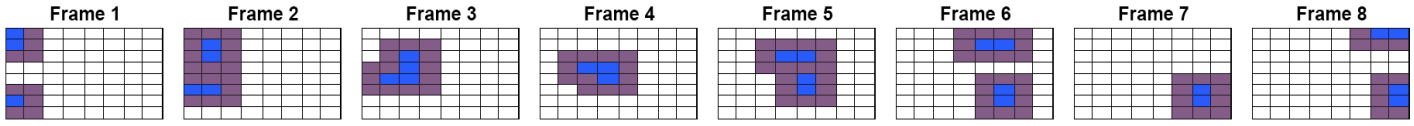

Figure 26. Neighborhood of the set of regions of interest. Regions are represented in blue and in purple we have its neighborhood.

So, if a good candidate region of the previous frame is located in the neighborhood of another region of the current frame, these are linked. Figure 27 shows in green the candidate regions of the previous frames in the current ones.

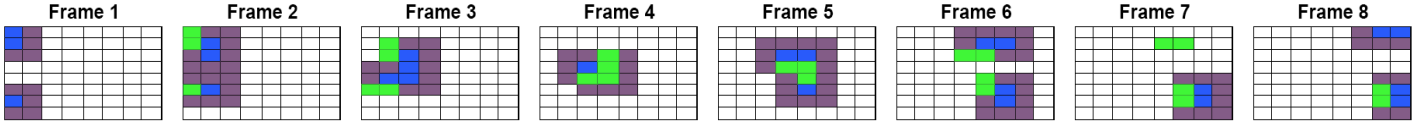

Figure 27. Overlay of the regions of interest of the previous frames. Regions are represented in blue, in purple we have its neighborhood and green represent the regions of the previous frame.

This simple method allows linking all regions of interest across frames, which let us describe regions not only by their motion, but also by their behavior over time. In Figure 28 we can see how the regions of the example would be linked. Regions are identified by a number that indicates which one is found first in a frame from the top-left corner to the bottom-right corner. For instance, in frame 6 region 1 would correspond to the upper region and region 2 would correspond to the lower one.

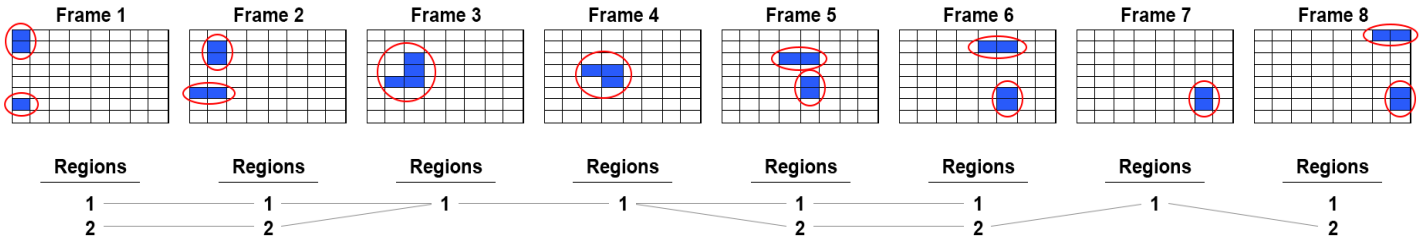

Figure 28. Linking of regions of interest.

This simple diagram enables to track regions over frames in a very simple way. Just by looking at Figure 28 we can see that regions of frame 1 move separately into frame 2, as both regions of the first frame have one-on-one correspondences with both regions of the second frame. We can also see that the only region that appears in frame 3 is a result of the merging of the previous frame regions. In frame 5 we have the opposite case, where the only region of frame 4 splits into two regions. This diagram also allows identifying starting and finishing points of linked regions. For instance, both regions of frame 1 and region 1 of frame 8, do not have any previous links, meaning they are the starting points. Region 1 of frame 6 however, would be an ending point, as it does not have following links.

Taking into account all possible starting and ending points we can define series of events as described in Figure 29. Each of these events represents how a starting region might have evolved through frames.

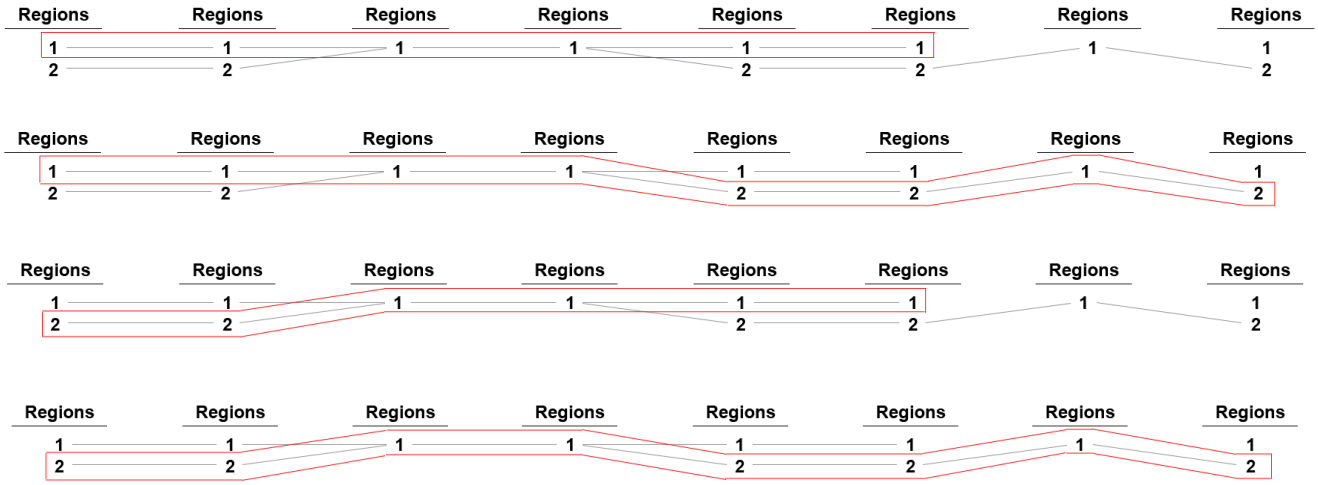

Figure 29. Possible events identified through the linked regions.

Let us remember at this point that the aim of this processing block is to detect candidate highlight events, that is, a group of frames, from the good candidate regions identified in the previous block (see section 4.4). Thus, for a frame to be considered a part of a highlight, it is sufficient condition that the best candidate region of that frame is considered a highlight region.

In section 4.4 we talked about how regions of interest are described by their normalized motion, which are the average displacement of the motion vectors they contain. Under previous assumptions, the best candidate region of a frame to be a highlight would be that with a larger normalized motion. But temporal information of these regions plays a key role when assessing their motion and has to be accounted for too. This way, for each frame we can analyze the best candidate to be a highlight not only based in the normalized motions of the regions present in that frame, but also taking into account those linked to them in time.

Figure 30 shows the regions involved in the four different events identified in previous Figure 25. To illustrate this process, imagine each blue cell in Figure 30 amounts for a unit motion. Then, the first event has a total motion of 15 along 6 frames. The second event has a total motion of 19 along 8 frames. Third event amounts for a total motion of 14 along 6 frames and finally, the last event amounts for a total motion of 18 along 8 frames. So, the respective normalized motions of each event would be: 2.5 for the first event, 2.375 for the second one, 2.333 for the third and 2.25 for the last one.

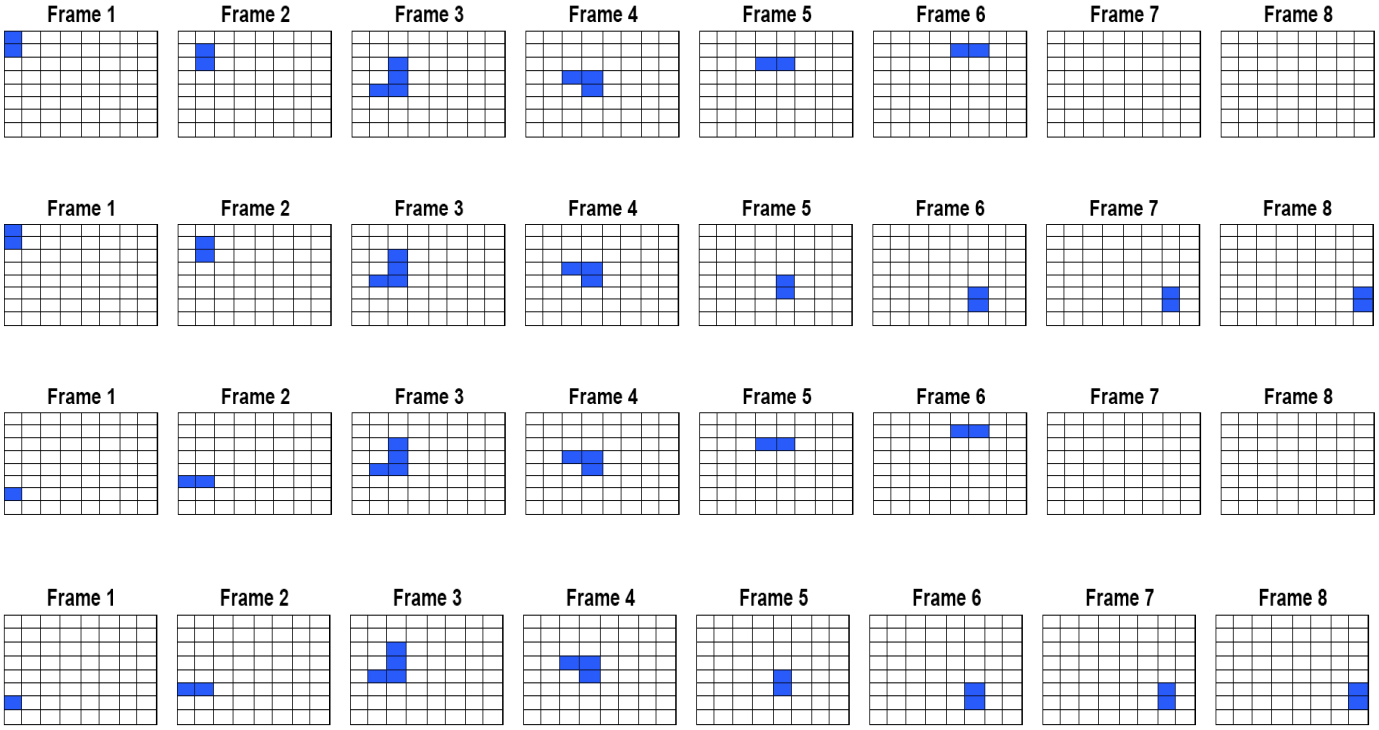

Figure 30. Regions involved in each of the identified events.

Now, when choosing the best candidate region for the frame 1, where four events concur, we select the region that appears in the event with a larger normalized motion, which for this case would be the first event. This process iterates through all frames to produce a final output consisting on one region (or none if there are no candidate events) per frame that is considered to be the best candidate to be a highlight. For the example set along this section, this output would correspond to:

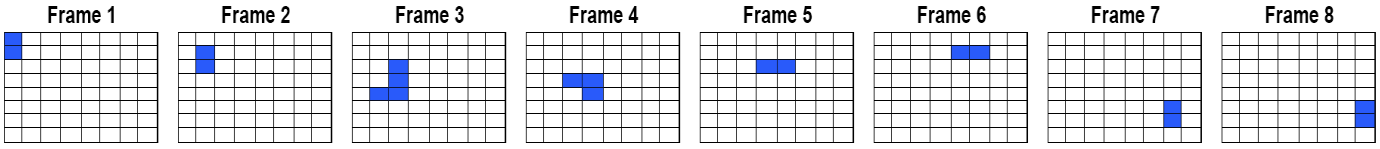

Figure 31. Example of the output of this block, which is a set of regions that are part of good candidate events.

Comparing the previous Figure 31, which corresponds to the output of this block, with Figure 25, which corresponds to the input, we can see that the output can be interpreted as an attention map that indicates for each frame where is the region more likely of being part of a highlight. Additionally, events can be filtered analogously to how we filtered regions in section 4.4, allowing to get rid of small duration events and storing only those that suppose good candidates.

In summary, this processing block allows identifying and filtering events that although relevant in motion are not so much in time. For instance, events of less than a certain duration (i.e. 1 second) could be filtered as they are not likely to represent a highlight. The resulting events are compared when concurring in a frame in order to select the best candidate regions, which are stored for the final event detection block.

4.6. EVENT DETECTION

All previous processing blocks serve to output the best candidate events found in the input video and in this section, we analyze the information contained in the regions that compose these events. The procedure follows the same idea than that presented in section 4.3, although this block does not consider all foreground key points but rather those regions of interest that appear in the best candidate events identified, that is, only one region of interest per frame. This means that instead of contemplating the information of all foreground key points for each frame, this processing block just takes into account the region of interest belonging to the best candidate event for each frame.

Figure 32 shows the results obtained for the same example video of section 4.3 but following our proposed method and so, taking into account all processing blocks. The main difference between these results and those obtained in the previous section is that now, we are only considering the best candidate region of each frame after a long refinement process. It is easy to appreciate that Figure 32 presents many frames where nothing is detected (no key points nor motions). This is because of the temporal filtering applied at the end of section 4.5, which helps discard many events that due to their duration do not represent good candidate events. All processing blocks are aimed towards generating the graphics of this figure, which describe how motion evolves along frames for the most relevant event identified in each frame, allowing to detect highlight events in a much more rigorous manner than before.

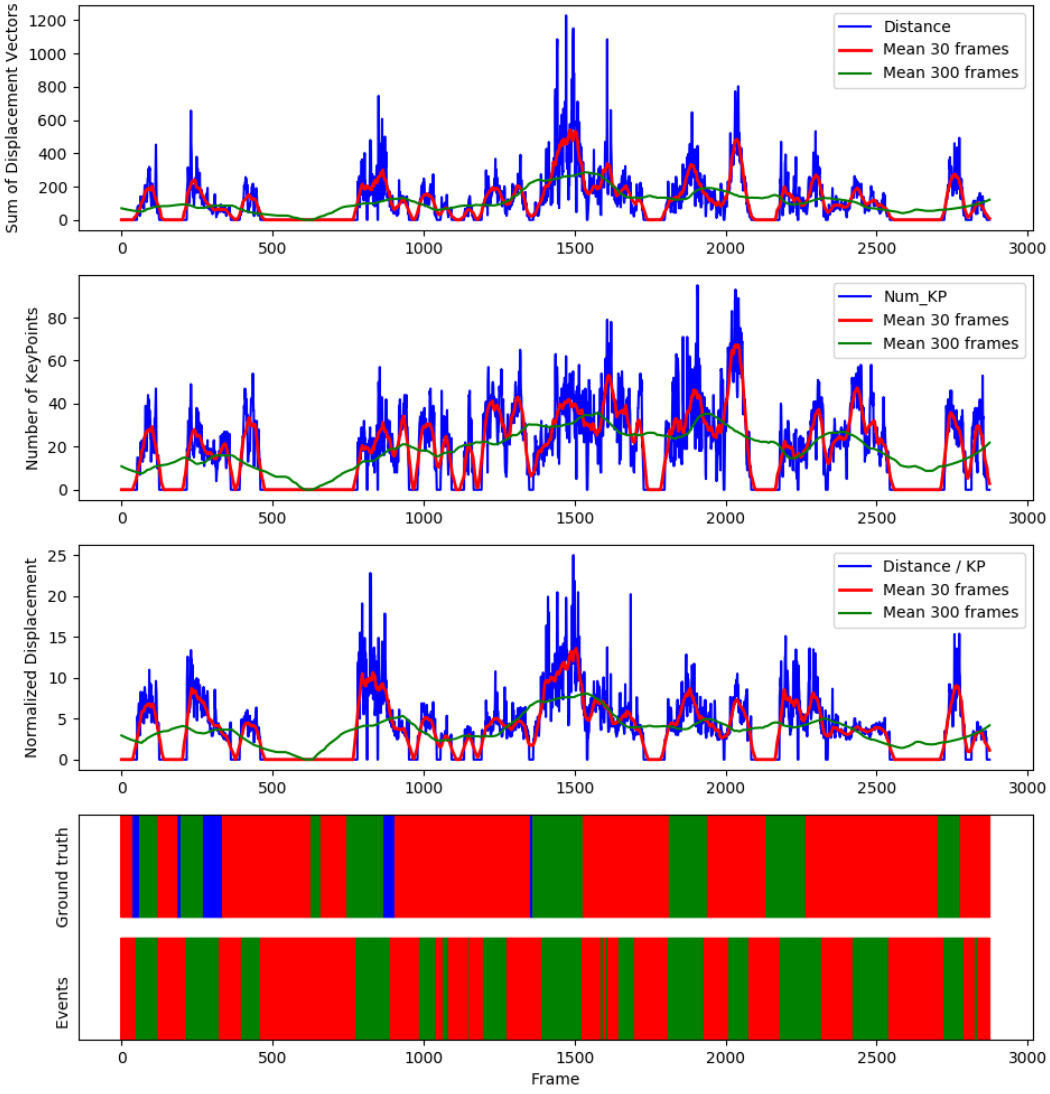

Figure 32. Summary of the results obtained for an example video following the proposed strategy.

These results describe the motion of the most relevant region in each frame, whereas the results obtained previously represented the motion of all foreground key points present in a frame. What this means is that now, the detected events that appear in the lower color bar of Figure 32 are based on the comparison between the most relevant events over time instead of for a given instant. It can also be noted that the mean values of the first three graphics (red and green lines) have been computed with a time window centered at each frame value (blue lines), instead of using a time window containing only previous frames as in the example of section 4.3.

Detected events presented in the last graphic of the previous figure can be further processed as it was briefly mentioned in section 4.3. By performing a dilation and a later erosion of the results, we can group up events that are close in time and thus, likely belong to the same event, as shown in Figure 33. Dilation is performed over one dimension on the detected events of the upper color bar, stretching them. This process allows joining near events, that are later eroded so as to restore their initial width.

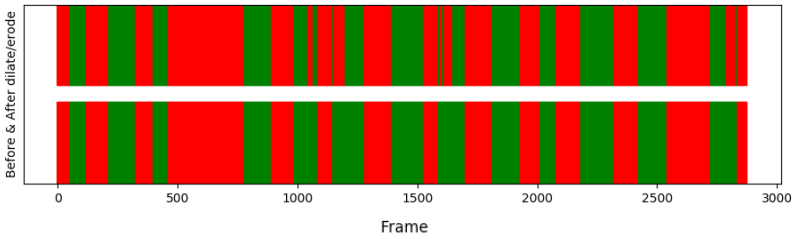

Figure 33. Dilation and erosion of the detected highlights in order to group up events that are likely part of the same one.

If we take a look at the time means of the normalized motion graphic of Figure 34 for the detected events after grouping them up, we get the following figure.

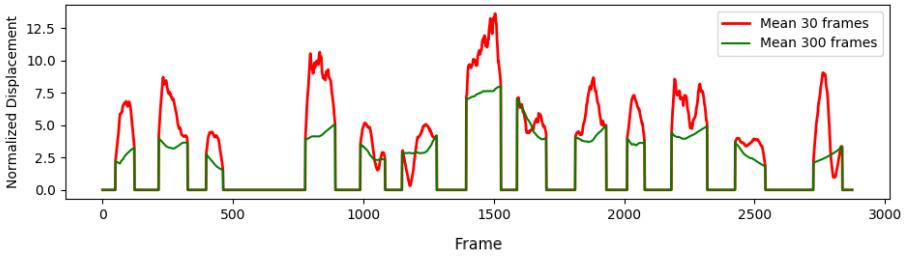

Figure 34. Mean normalized motions for two time windows of different size for the regions were events have been detected.

So, if we were to characterize each of these events, a viable option would be by taking the enclosed area between means. As explained in previous section 4.3, these means serve to represent how different the motion of a 1 second event is with respect to that motion of the 10 seconds event that surround it. The areas enclosed are represented in Figure 35 and no negative values have been taken into consideration, as they could affect negatively some events.

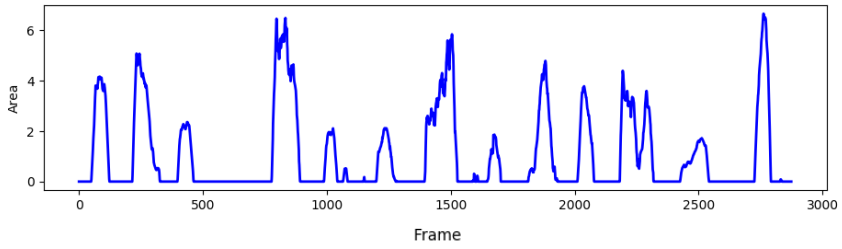

Figure 35. Area enclosed under the mean normalized motions for two time windows of different size.

We make use of this enclosed areas after a normalization process to model the detected events of Figure 33 as described in the following Figure 36. This modelling serves to further describe detected events in a probabilistic way, as the larger the normalized enclosed area the more probabilities an event has of being a highlight. This is due to the nature of what the red and green lines of Figure 34 represent, where large enclosed areas suppose events which motions are much greater than those events that surround it.

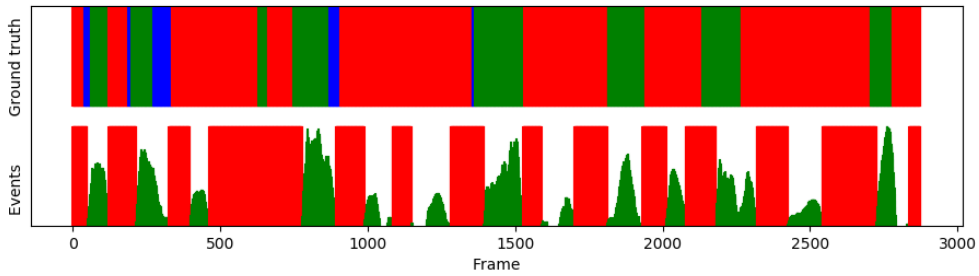

Figure 36. Result of modelling the previously detected events with the area enclosed by two mean normalized motions. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Finally, a decision has to be made in order to classify events as highlights or not and thus, we have to take a decision on whether we want to reliably detect highlight events even if that implies having some false positives, or on the contrary, we want to strictly detect highlights with as few false positives as possible. For this purpose, we make available 2 user variables that can stablish the minimum highlight event duration and the minimum probability of it being a highlight event (normalized enclosed area). For instance, by setting a minimum highlight event duration of 2 seconds (60 frames) and a minimum probability, we can filter the results of the previous figure to output the final result of this strategy, which is shown in Figure 37.

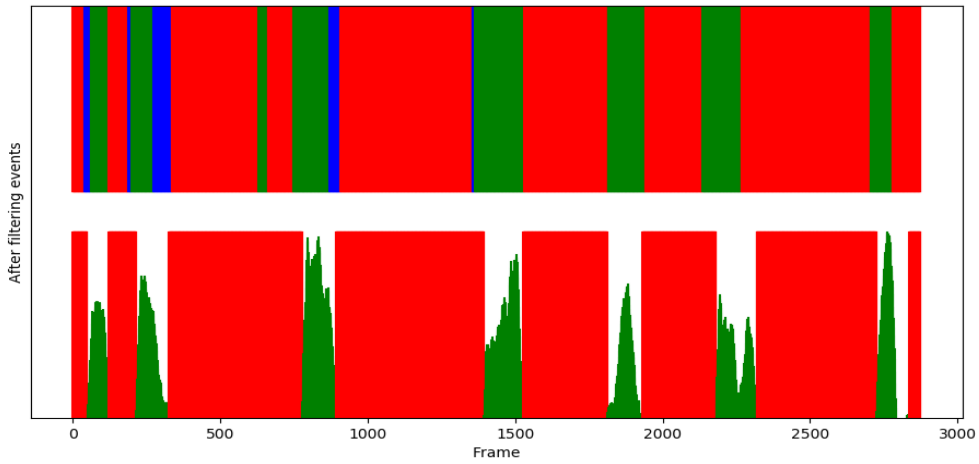

Figure 37. Final result of the proposed strategy, after filtering events based on user variables. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

These processed results are quite good, and it can be noted how close they are to the ground truth events of this particular video example. In the next section we will discuss in detail the different results obtained along this work. The output of this block are the video clips formed by the frames identified as highlights.

5. RESULTS

5.1. DATABASE

Along this work a database has been created from scratch. Several hours of video were recorded and events were manually annotated with a video editing software that allowed exporting these markers as a CSV file. These annotations can include 3 different categories: highlight, no highlight, or uncertainty. The existence of uncertainty as a ground truth event is due to the nature of highlights in this sport, which make difficult even for a human being to tell where a highlight begins or ends.

The results described in this section correspond to the results obtained for 3 different videos following the proposed strategy for detecting highlights. These videos were all recorded using an iPhone 6S and they share the following properties:

- File format: MPEG (.mpeg).
- Spatial resolution: 1920×1080.
- Frame rate: 29.97 fps.
- Codec: MPEG-1/2 video (mpgv).

The CSV files with event annotations are used as ground truth and are described as shown in Figure 38. Marker name column corresponds to the 3 categories mentioned above, and the in and out columns represent the starting and ending frames of each event.

	Marker Name	In	Out
0	No Truco	0	40
1	Incertidumbre	41	62
2	Truco	63	124
3	No Truco	125	193
4	Incertidumbre	194	204
..	...	...	...
98	No Truco	17409	17473
99	Truco	17474	17542
100	No Truco	17543	17654
101	Truco	17655	17871
102	No Truco	17872	18250

Figure 38. Example of the information stored in a CSV file containing the ground truth events for a given video.

The first video has a duration of 21 minutes. The maximum number of people that appear in the frame at any given time is of 7, although only 6 of them are performing passes. Background is quite static with people moving in the background. Highlights are of short durations and pace is not fast.

Figure 39. Example scenes of video 1.

The second video has a duration of 22 minutes. Similar to the first video, the maximum number of people that appear in a frame is not high, counting only 6, although only 4 of them are performing passes. Background is mostly static throughout the entire video and highlights are mostly of short durations too.

Figure 40. Example scenes of video 2.

The third video has a duration of 10 minutes and there is a significant increase of people, with a maximum of 12 people in frame, 8 of which are performing passes. Background is highly dynamic with people moving all around. Highlights are of longer durations and with much more motion than the previous two. This is the video that has been used in the examples of previous sections.

Figure 41. Example scenes of video 3.

Table 2 summarizes the main events found in these three videos. As we can see from this table, highlight events suppose only a small fraction of a video. For the first two videos, which show less affluence of people, highlights correspond to around 10% of the total video (20% if uncertainty is assumed to be part of the highlight). This evidence manifests the necessity for algorithms that can automatically detect highlights. Highlights on these two videos were described as short before and indeed we can see that the average highlight event duration in these two videos is around 70 frames (a little over two seconds). The third video presents a significant increase of affluence and highlights are longer, as it is reflected in the highlight values of this video.

Table 2. Video events description.

Video	Event	Avg. event duration	Total event durations	%
Video 1	Highlight	67	4.414	11,8
	No highlight	447	29.484	78,6
	Uncertainty	32	3.605	9,6
Video 2	Highlight	73	3.797	9,6
	No highlight	649	33.759	85,7
	Uncertainty	30	1.842	4,7
Video 3	Highlight	106	4.979	27,4
	No highlight	269	12.897	71,1
	Uncertainty	34	272	1,5

5.2. RESULTS FOR VIDEO 1

The results obtained for this video after passing through all processing blocks of the proposed strategy are represented in Figure 42. The upper color bar represents the ground truth event of each frame, where red indicates no highlight, green indicates highlight and blue indicates uncertainty. The lower color bar represents the automatically detected highlights obtained.

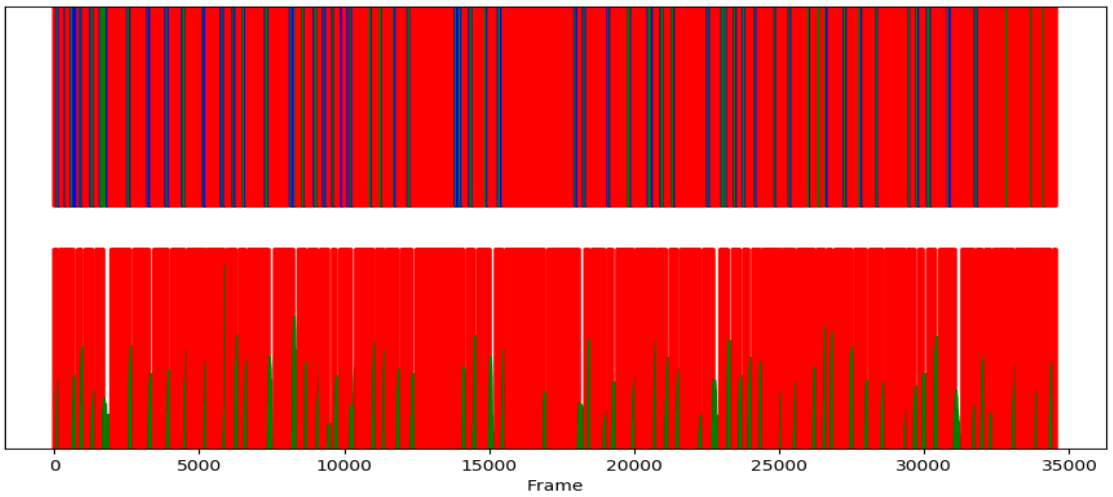

Figure 42. Results obtained for the whole video sequence following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

It might result visually difficult discerning how good these results are when taking a look at the whole sequence. For that purpose, Figure 43 shows the results of just a portion of the video. In this figure we can better appreciate how the detected events relate to the ground truth. Out of the 10 highlights present in the ground truth of this figure, the proposed strategy is able to correctly identify 9 of them with really good accuracy, only missing one very short highlight that was filtered out by its duration. And not only is able to detect these, but it also characterizes them by their motion.

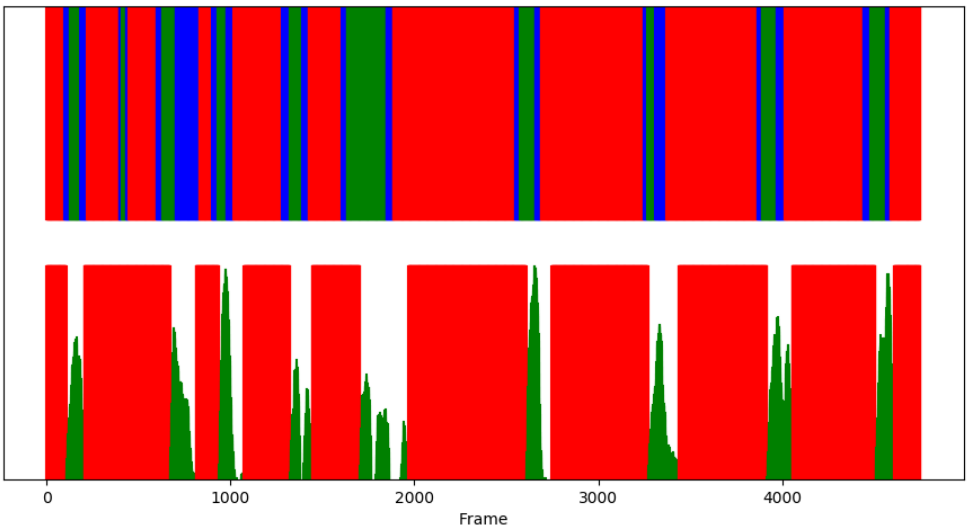

Figure 43. Fragment of the obtained results following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Figure 44 shows the obtained results for the whole video sequence divided in smaller time segments. This figure let us visually evaluate the overall performance of the proposed strategy. Detected events show close relation with those of the ground truth, although it can be noted that they seem to be a little displaced to the right.

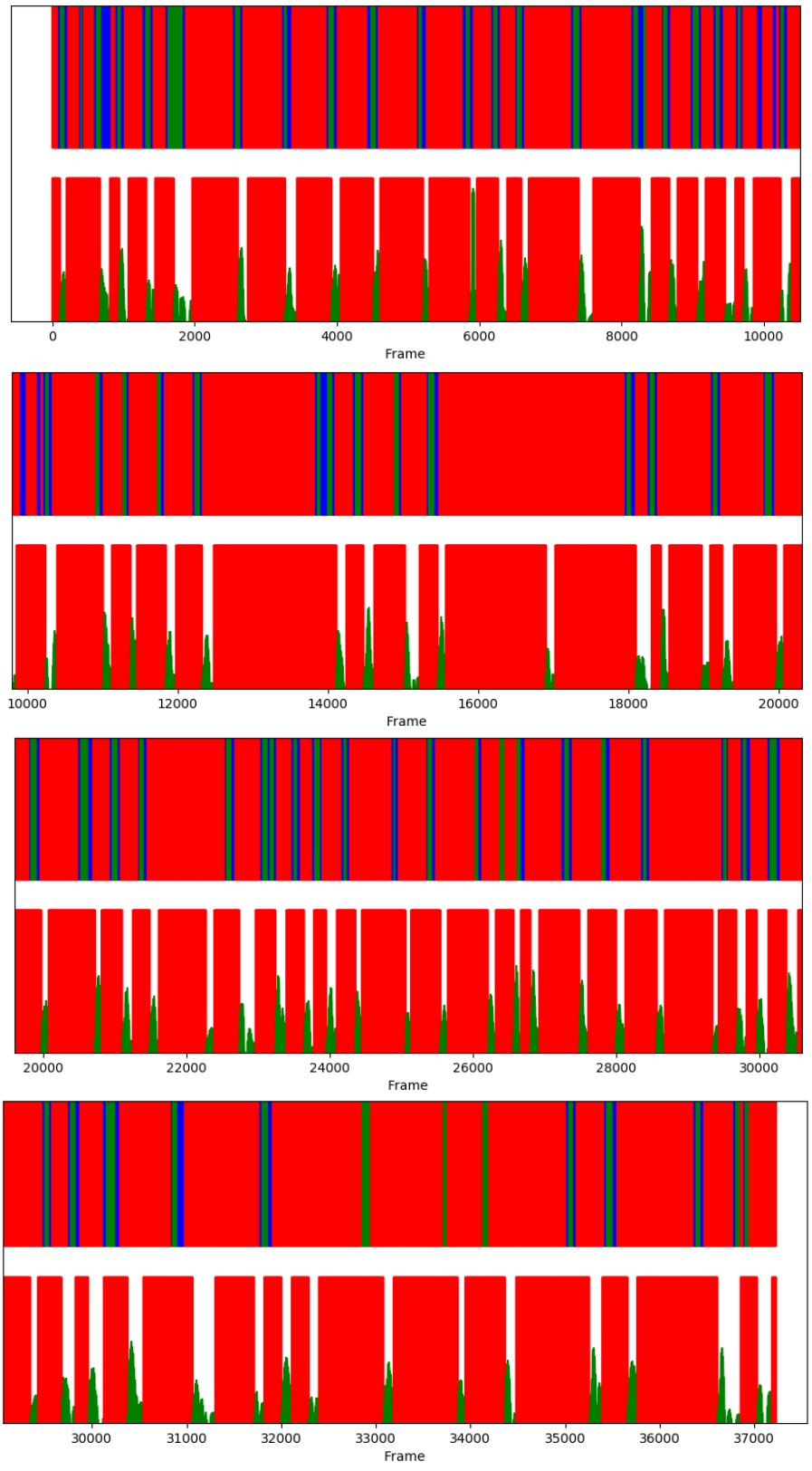

Figure 44. Decomposition of the obtained results following the proposed strategy for a whole video sequence. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Table 3 depicts the confusion matrix of the events for this video. It can be seen that the number of true negatives is the highest one. Out of the 37.503 frames that compose this video, 22.243 have been correctly identified as not a highlight, which accounts for approximately 12 and a half minutes of the original 21-minute video. This value alone manifests the good job the proposed strategy does when discarding non relevant events. Regarding the ground truth highlight events, it appears detection could still be tuned to achieve better results. As commented along this project we define some variables that allow adjusting the results based on user preferences.

Table 3. Confusion matrix.

		Ground truth events	
		Highlight	Not a Highlight
Detected events	Highlight	1.984	5.041
	Not a Highlight	2.430	24.443

The recall and precision scores achieved for video 1 are of 45% and 28.2% respectively, which in turn lead to an F-score of 17.3%. Although parameters could be tuned to achieve a more desirable output, we must have in mind that the results obtained through the proposed strategy, that is, the highlight events detected, are not thought to perfectly fit the ground truth events.

Due to the nature of the sport is difficult to accurately annotate ground truth events on videos in the first place. So even for a human being it is a challenging task to decide where highlights begin or end, which is reflected in the false positive and negative values depicted in Table 3. Following the previous idea, it is expected that the results obtained following the proposed strategy do not perfectly fit the ground truth events but rather aim to give the best estimations as to where these are. For what we can in Figure 44, detected events are really close to the ground truth events even when numerical results do not support it. Additionally, an output buffer could be implemented to select the surroundings of each event, to ensure that the output clips contain the whole event and not only its peak of motion.

In Figure 45, some examples of highlight events correctly identified are displayed. In contrast, Figure 46 shows some false positive examples events detected as highlights when they were not. We noted most false positives found actually correspond to highlights that were just not considered in the annotation phase because of different reasons.

Figure 45. Examples of highlight events correctly identified.

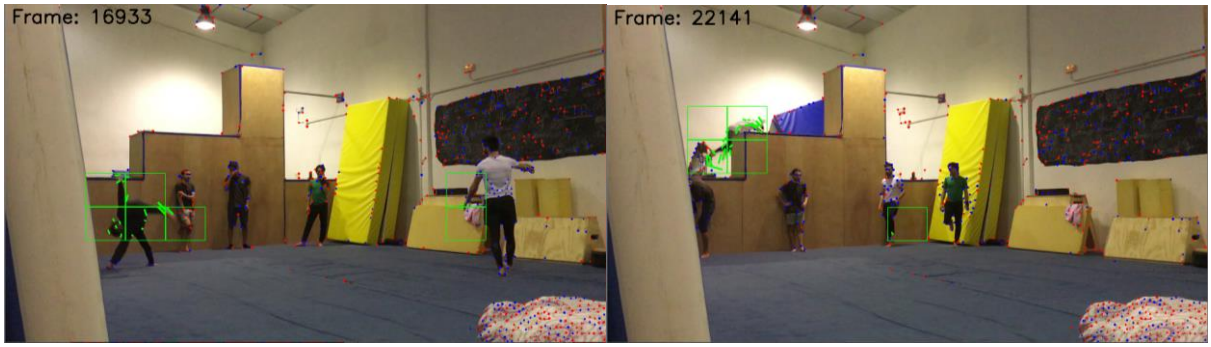

Figure 46. Examples of events incorrectly identified as highlights.

5.3. RESULTS FOR VIDEO 2

Similar results to those of the previous video are described in this section, where Figure 47 represents the ground truth and the detected events after applying the proposed strategy. The upper color bar represents the ground truth event of each frame, where red indicates no highlight, green indicates highlight and blue indicates uncertainty. The lower color bar represents the automatically detected highlights obtained for this video.

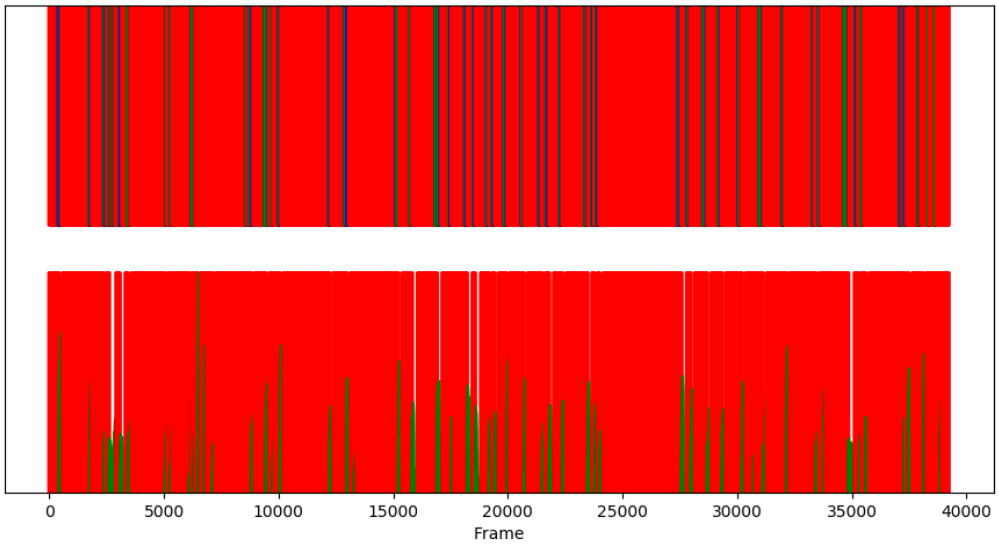

Figure 47. Results obtained for the whole video sequence following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Same as before, Figure 47 spans many frames and so, it is difficult to tell how well it is performing. In Figure 48 we show a portion of these results, which again show really promising. Out of the 10 highlight events present in this portion of the video, the proposed strategy is able to correctly identify all of them.

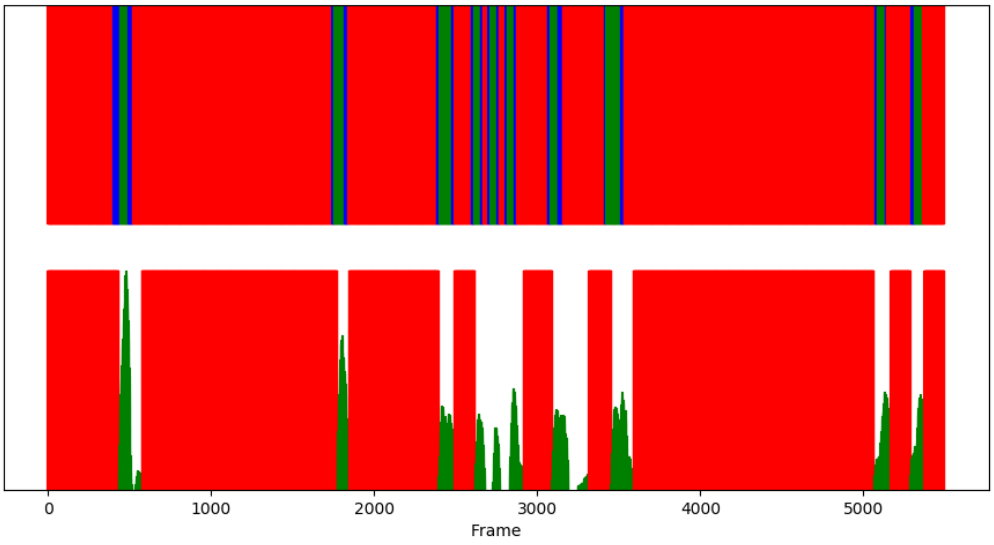

Figure 48. Fragment of the obtained results following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Figure 49 shows the obtained results for the whole video sequence divided in smaller time segments. This figure let us visually evaluate the overall performance of the proposed strategy. Detected events show close relation with those of the ground truth.

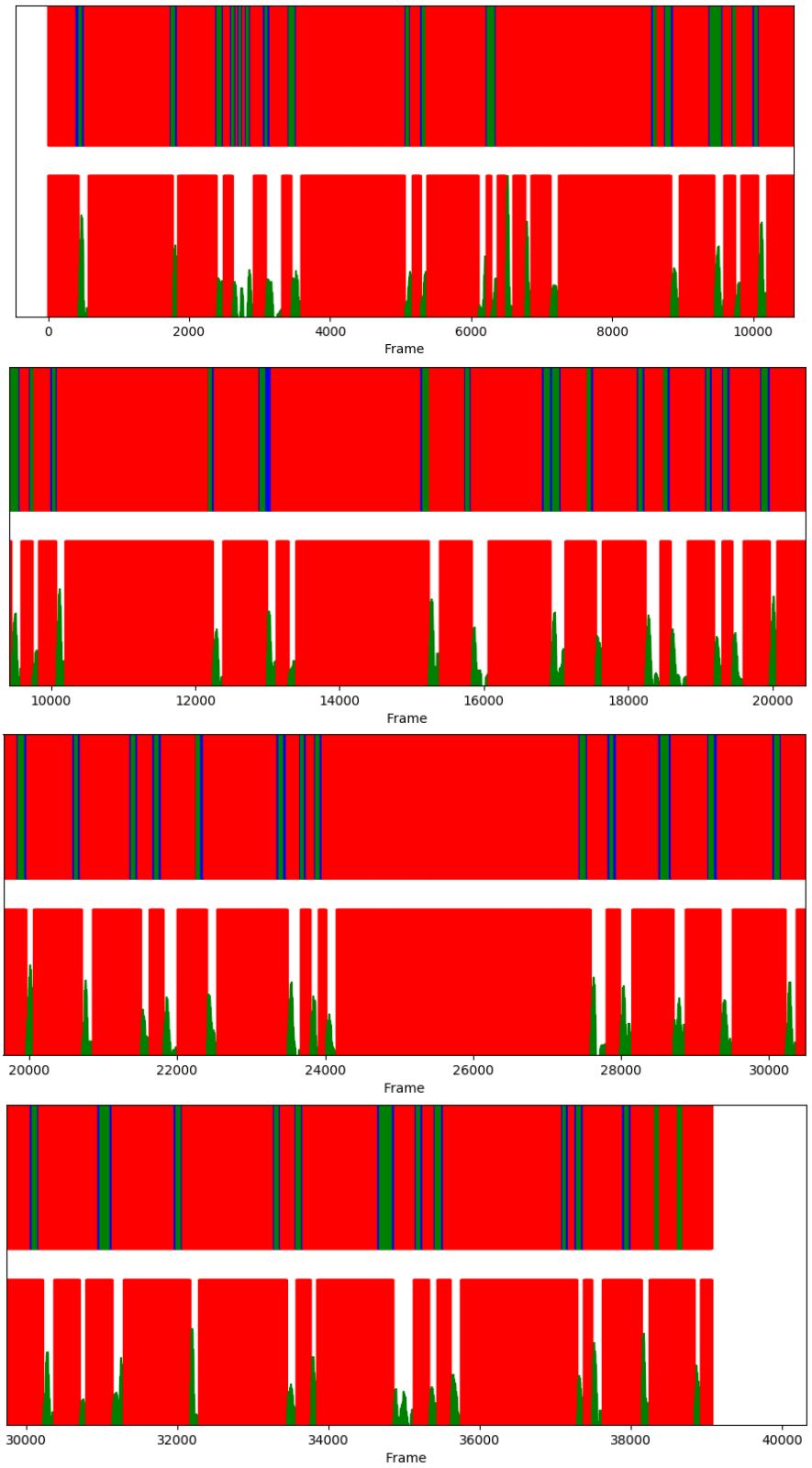

Figure 49. Decomposition of the obtained results following the proposed strategy for a whole video sequence. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Table 4 depicts the confusion matrix of the events for this video. It can be seen that the number of true negatives is the highest one. Out of the 39.398 frames that compose this video, 27.856 have been correctly identified as not a highlight, which accounts for approximately 15 and a half minutes of the original 22-minute video. This value alone manifests the good job the proposed strategy does when discarding non relevant events. Regarding the ground truth highlight events, it appears detection could still be tuned to achieve better results. As commented along this project we define some variables that allow adjusting the results based on user preferences.

Table 4. Confusion matrix.

		Ground truth events	
		Highlight	Not a Highlight
Detected events	Highlight	1.974	5.903
	Not a Highlight	1.823	27.856

The recall and precision scores achieved for video 2 are of 52% and 25.1% respectively, which in turn lead to an F-score of 17%. Although parameters could be tuned to achieve a more desirable output, we must have in mind that the results obtained through the proposed strategy, that is, the highlight events detected, are not thought to perfectly fit the ground truth events.

Due to the nature of the sport is difficult to accurately annotate ground truth events on videos in the first place. So even for a human being it is a challenging task to decide where highlights begin or end, which is reflected in the false positive and negative values depicted in Table 4. Following the previous idea, it is expected that the results obtained following the proposed strategy do not perfectly fit the ground truth events but rather aim to give the best estimations as to where these are. For what we can in Figure 49, detected events are really close to the ground truth events even when numerical results do not support it. Additionally, an output buffer could be implemented to select the surroundings of each event.

In Figure 50, some examples of highlight events correctly identified are displayed. In contrast, Figure 51 shows some false positive examples events detected as highlights when they were not. We noted most false positives found actually correspond to highlights that were just not considered in the annotation phase or due to people crossing the camera plane.

Figure 50. Examples of highlight events correctly identified.

Figure 51. Examples of events incorrectly identified as highlights.

5.4. RESULTS FOR VIDEO 3

Results obtained for this last video are no different from the previous ones. Once again, Figure 52 describes the ground truth and the detected highlight events result of applying the proposed strategy and the same color labels are followed.

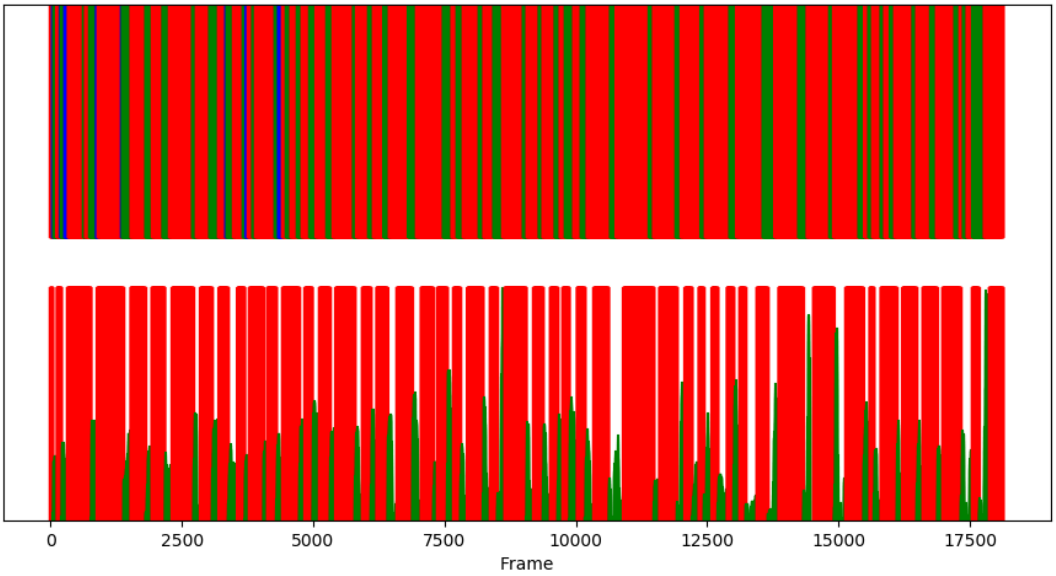

Figure 52. Results obtained for the whole video sequence following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Figure 53 shows just a portion of the video once more, so that detected events can be better evaluated. Out of the 10 highlights present in this portion, 9 are correctly detected, manifesting again the good results the proposed strategy provide.

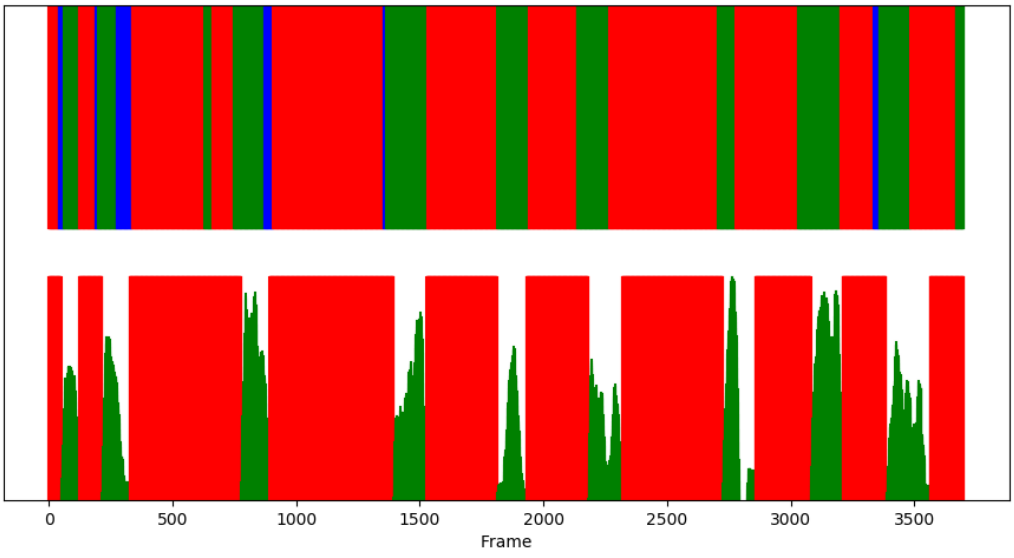

Figure 53. Fragment of the obtained results following the proposed strategy. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Figure 54 shows the obtained results for the whole video sequence divided in smaller time segments. This figure let us visually evaluate the overall performance of the proposed strategy. Detected events show close relation with those of the ground truth.

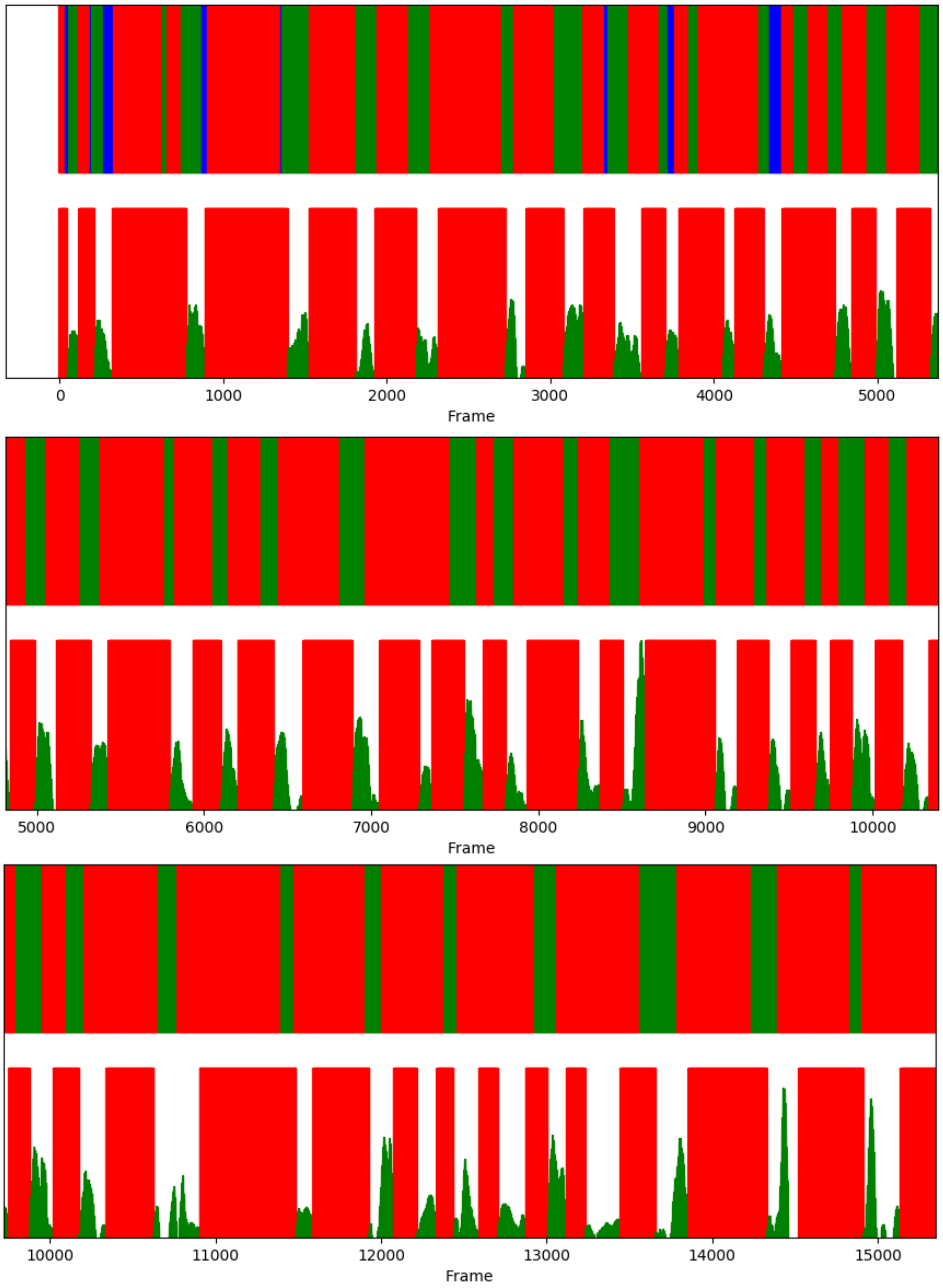

Figure 54. Decomposition of the obtained results following the proposed strategy for a whole video sequence. The top color bar represents the ground truth events, red indicates no highlight, green indicates highlight and blue indicates uncertainty. The bottom color bar represents the detected events.

Table 5 depicts the confusion matrix of the events for this video. It can be seen that the number of true negatives is the highest one. Out of the 18.148 frames that compose this video, 8.876 have been correctly identified as not a highlight, which accounts for approximately 5 minutes of the original 10-minute video. This value alone manifests the good job the proposed strategy does when discarding non relevant events. Regarding the ground truth highlight events, it appears detection could still be tuned to achieve better results. As commented along this project we define some variables that allow adjusting the results based on user preferences.

Table 5. Confusion matrix.

		Ground truth events	
		Highlight	Not a Highlight
Detected events	Highlight	2.288	4.021
	Not a Highlight	2.691	8.876

The recall and precision scores achieved for video 3 are of 50% and 36.3% respectively, which in turn lead to an F-score of 21%. Although parameters could be tuned to achieve a more desirable output, we must have in mind that the results obtained through the proposed strategy, that is, the highlight events detected, are not thought to perfectly fit the ground truth events.

Due to the nature of the sport is difficult to accurately annotate ground truth events on videos in the first place. So even for a human being it is a challenging task to decide where highlights begin or end, which is reflected in the false positive and negative values depicted in Table 5. Following the previous idea, it is expected that the results obtained following the proposed strategy do not perfectly fit the ground truth events but rather aim to give the best estimations as to where these are. For what we can in Figure 54, detected events are really close to the ground truth events even when numerical results do not support it. Additionally, an output buffer could be implemented to select the surroundings of each event.

In Figure 55, some examples of highlight events correctly identified are displayed. In contrast, Figure 56 shows some false positive examples events detected as highlights when they were not. We noted most false positives found actually correspond to highlights that were just not considered in the annotation phase or to events with lot of motion right after a highlight event.

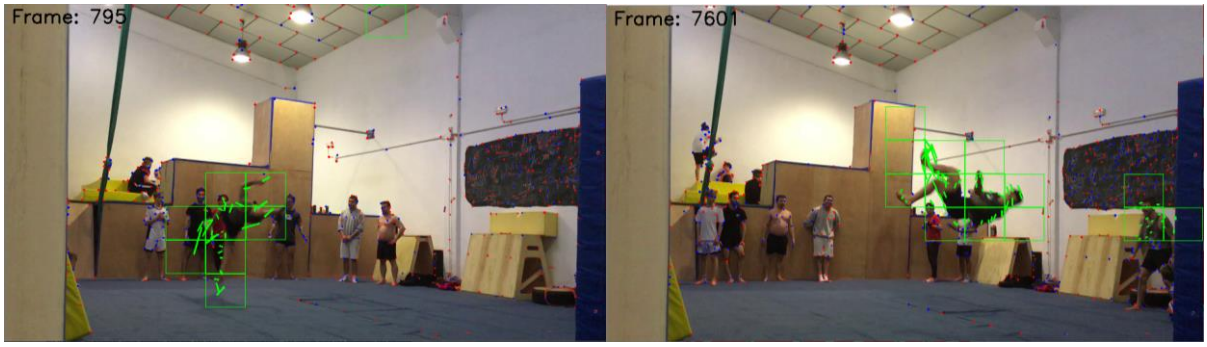

Figure 55. Examples of highlight events correctly identified.

Figure 56. Examples of events incorrectly identified as highlights.

6. CONCLUSIONS AND FEATURE LINES

This section contains a series of conclusions obtained from the work carried out, as well as an approach of possible future lines of work with the aim of improving the proposed strategy.

6.1. CONCLUSIONS

This work has allowed us to develop a strategy to automatically detect video highlights for a specific sport from input video sequences obtained with a mobile phone. Additionally, this strategy could easily be adapted to other sports or fields of applications to detect highlight events based on motion features.

The proposed strategy is built around motion features. In the first processing blocks we extracted a set of key points, for which different methods were analyzed and Shi-Tomasi algorithm was finally selected. Then, we estimated the motion of each of these key points making use of a pyramidal implementation of the Lucas-Kanade optical flow, that allowed to track large displacements in an efficient way. Both these processes give rise to motion features.

In a later processing stage, we grouped up this motion features forming regions, adding sustain to the previous identified key points, as now we could identify objects in the frame and estimate their motions. These regions were filtered through some common assumptions to output a set of regions that pose good candidate highlight regions. At this point we presented 2 alternative applications, one aimed at real time applications and another one at off-line applications, which is the one we opted for and is described in much more detail.

The temporal coherence analysis block provided us with valuable insight to identify all possible events present in a video. These events were also refined in order to output a set of good candidate events.

Finally, in the last processing block we observed the evolution of the motion information stored in the previous events over frames, which helped in visually understanding the effect of the different blocks in the motion features initially extracted.

Results have shown to be reasonably good when detecting highlight events for different videos with different environments, although they could be enhanced as we will discuss next. Additionally, we provide two user variables that allow filtering detected events to provide a better output based on user preferences.

6.2. FUTURE LINES

It has been seen that the strategy implemented in this work has provided good results on numerous occasions. However, it is still possible to improve its performance. Throughout its development, we have come up with an idea that could serve as future work and is described below.

Deep neural networks have shown incredible results in numerous types of applications, being image classification a recurrent one. Along this project we performed a preliminary study with a deep neural network based on VGG-16 which showed promising validation accuracies with just the image frames of the videos. However, a neural network trained not only with the image frames but also with the motion features obtained through this strategy could potentially find better patterns to represent highlights and thus provide better results.

7. REFERENCES

[1] A. Ekin, A. M. Tekalp, and R. Mehrotra, “Automatic soccer video analysis and summarization,” IEEE Trans. Image Process., vol. 12, no. 7, pp. 796-807, Jul. 2003.

[2] H. Pan, P. Van-Beek, and M. I. Sezan, “Detection of slow-motion replay segments in sports video for highlights generation,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2001, vol. 3, pp. 1649-1652.

[3] G. Xu, Y. F. Ma, H. J. Zhang, and S. Q. Yang, “An HMM-based framework for video semantic analysis,” IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 11, pp. 1422-1433, Nov. 2005.

[4] C. Y. Chen, J. C. Wang, J. F. Wang, and Y. H. Hu, “Motion entropy feature and its applications to event-based segmentation of sports video,” EURASIP J. Adv. Signal Process., vol. 2008, pp. 1-8, 2008.

[5] E. Mendi, H. B. Clemente, and C. Bayrak, “Sports video summarization based on motion analysis,” Comput. Electric. Eng., vol. 39, no. 3, pp. 790-796, 2013.

[6] R. W. Lienhart, “Dynamic video summarization of home video,” Proc. SPIE Electron. Imag., vol. 3972, pp. 378-389, 1999.

[7] J. Meng, H. Wang, J. Yuan, and Y. P. Tan, “From keyframes to key objects: Video summarization by representative object proposal selection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1039-1048.

[8] G. Evangelopoulos et al., “Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention,” IEEE Trans. Multimedia, vol. 15, no. 7, pp. 1553-1568, Nov. 2013.

[9] A. Garcia-del-Molino, X. Boix, J. H. Lim, and A. H. Tan, “Active video summarization: Customized summaries via on-line interaction with the user,” in Proc. AAAI Conf. Artif. Intell., 2017, pp. 4046-4052.

[10] H. Yang et al., “Unsupervised extraction of video highlights via robust recurrent auto-encoders,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 4633-4641.

[11] M. Otani, Y. Nakashima, T. Sato, and N. Yokoya, “Video summarization using textual descriptions for authoring video blogs,” Multimedia Tools Appl., vol. 76, no. 9, pp. 12 097-12 115, 2017.

[12] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, “TVSum: Summarizing web videos using titles,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 5179-5187.

[13] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkila, and N. Yokoya, “Video ¨ summarization using deep semantic features,” in Proc. Asian Conf. Comput. Vis., 2016, pp. 361-377.

[14] Y. Yuan, T. Mei, P. Cui, and W. Zhu, “Video summarization by learning deep side semantic embedding,” IEEE Trans. Circuits Syst. Video Technol., 2017.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learning Representations, 2015, pp. 1-10.

[16] C. Szegedy et al., “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1-9.

[17] J. Shi and C. Tomasi. “Good Features to Track”. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.593-600, June 1994.

[18] C. Harris and M. Stephens (1998). “A combined corner edge detector”. Proceedings of the 4th Alvey Vision Conference. pp. 147-151.

[19] Bouguet, J.-Y. “Pyramidal implementation of the Lucas-Kanade feature tracker.” (1999).

[20] Shih-Wei Sun, Yu-Chiang Frank Wang, Fay Huang, Hong-Yuan Mark Liao. “Moving foreground object detection via robust SIFT trajectories”. Journal of Visual Communication and Image Representation, Volume 24, Issue 3, pp.232-243, 2013.

[21] A. Senior. “Tracking people with probabilistic appearance models”, Proc. IEEE Perform. Eval. Track. Surveill. (2002) pp. 48-55.

[22] C. Cuevas, D. Quilón, and N. García, "Techniques and applications for soccer video analysis: A survey", Multimedia Tools and Applications, [under review].

APPENDIX A: ETHICAL, ECONOMIC, SOCIAL AND ENVIRONMENTAL ASPECTS

In its technical sense, the term impact is the change produced by some human activity. In this section, we consider all possible changes in the environment and at social and economic levels. Aspect refers to any circumstance related to the project that may be considered controversial for posing a conflict of interest or values (security vs. privacy, contamination vs. cost reduction, etc.).

A.1 INTRODUCTION

The work carried out is framed within the audiovisual sector and proposes the development of an alternative strategy to automatically detect highlight events in long-duration videos. The life cycle of this project can be summarized in obtaining an initial video, processing it through the proposed strategy and in the final obtention of results.

The cultural context in which the activity is carried out implies a wide use of social networks to share videos and the social group of greatest interest for the project would be that formed by young and adult people who practice or enjoy this activity and who have access to a recording device and a computer.

This work aims to cover the growing need to detect highlight events in long-duration videos, as a result of the increased storage capacity of recording devices. This would mean an improvement in quality of life for those people who dedicate themselves to creating audiovisual content, allowing them to spend less time selecting interesting content and thus significantly reducing exposure time to a screen. Consequently, by reducing the time spent on editing, which can be hours depending on the project, energy consumption is also achieved.

A.2 DESCRIPTION OF RELEVANT IMPACTS RELATED TO THE PROJECT

A significant segment of potential users currently use video editing tools for the simple purpose of extracting highlight events. The proposed strategy extracts these highlight events quickly, automatically and free of charge. This saves users the time it usually takes to select interesting events prior to any editing, which in turn leads to an improvement in quality of life due to less screen exposure time. Regarding the economic aspect, this work could suppose money savings on video editing software licenses for those people who only use them with the intention of extracting highlight events, as well as saving energy consumption.

More and more people are dedicated to the world of content creation on platforms like YouTube. This social segment usually bases its income exclusively on the benefit that their videos bring. The strategy proposed in this work would allow these to increase the speed at which they are able to create content and would therefore mean a greater economic benefit, since they can spend more time obtaining audiovisual material and editing.

This work proposes a strategy that reduces the environmental impact by allowing a user to spend less time on the event selection process, which depending on the project can take between a few and several hours, in which energy resources are consumed.

A.3 CONCLUSIONS

The proposed strategy incorporates positive impacts such as the reduction of time in front of a screen for the health of a user, the improvement of productivity for those people dedicated to the creation of content or the improvement of income that this increase in productivity entails.

Additionally, this work proposes a strategy that would allow a reduction in energy consumption, which depending on the project could be less or more, by dedicating less time and resources to the selection of events, an essential stage in any audiovisual production.

APPENDIX B: ECONOMIC BUDGET

MANPOWER COST (direct cost)

Hours	Cost/hour	Total
300	15 €	**4.500 €**

MATERIAL RESOURCES EXPENSES (direct cost)	Buying price	Use in months	Amortization (in years)	Total
Personal computer (Software included)	800,00 €	6	5	80,00 €
Recording device	450,00 €	6	5	45,00 €

MATERIAL RESOURCES TOTAL EXPENSES	**125,00 €**

GENERAL EXPENSES (indirect cost)	15%	over DC	**693,75 €**
INDUSTRIAL BENEFIT	6%	over DC+IC	**319,13 €**

SUBTOTAL BUDGET		**5.637,88 €**
APPLICABLE TAXES	21%	**1.183,95 €**
TOTAL BUDGET		**6.821,83 €**