Usage of Video Analysis Algorithms for Goalball Scounting Purposes

Zygimantas Matusevičius[a], Julius Gudauskas[a]

[a]*Faculty of Informatics (Kaunas University of Technology), Kaunas, Lithuania*

Abstract

In the 21st century, IT innovations in sports have become very impactful. Highly innovative equipment allows athletes to stay efficient, reach their best results and even predict the opponent's strategies. Despite the growing interest in sport innovations, adaptive sports for disabled persons still feels the lack of modern solutions. We introduce the solution to analyze the video stream of goalball and using computing power to calculate the total performed throws and help to gather statistical data about the game

Keywords

Goalball, Video analysis, Transformation algorithms, Ball tracking

1. Introduction

In the 21^{st} century, IT innovations in sports have become very impactful. Highly innovative equipment allows athletes to stay efficient, reach their best results and even predict the opponent's strategies. Despite the growing interest in sport innovations, adaptive sports for disabled persons still feels the lack of modern solutions.

Contacting the Lithuanian goalball team revealed, that nowadays game innovations are a must to have an advantage against opponents, but software solutions are very expensive or not available. To maintain high results with the minimal cost of hardware, a system, which tracks the ball in real-time or from other video sources and gives the statistical information about ball positioning and movement is needed. To achieve this result, the most important task is to solve a video source distortion from the camera positioning problem.

2. Overview

Object recognition from video footage and images is widely used technology [1, 2]. Projects like football gate line and goals detection (such as "Once video analyzer" [3] that helps to explain a game concepts by creating animated and professionally designed graphics), basketball ball detection (such as "Hudl" [4] – sports analyzer for teams and athletes, "HomeCourt" app [5] for basketball players skills development), game strategy prediction and even face recognition. Most of the video tracking approaches are based on points of interest tracking [6], throughout video frames or color detection in three-dimensional color matrix. Video analysis is a complex process and the traditional analysis techniques usually requires a lot of calculation and becomes very costly. Kanade–Lucas–Tomasi (KLT) feature tracker [7] is one of the approaches to feature extraction that is proposed mainly for the purpose to deal with the video analysis special intensity and minimize the cost. The method is widely used in face recognition or motion tracking, where the different frames share a large number of tracking points. It was tested on eight broadcasts and achieved a 94% face track detection rate (the average processing speed was 3.8 ffs) [7]. Meanwhile the KLT algorithm based on points of interest, other popular tracking approach based on a color filtration. A color based video filtration uses the spectrum of three dimensional color matrix and the filtration executed by the desired color code. It requires less calculation and it's easier to implement, but in some cases it does not provide enough data to fully track or recognize an object. The other problem that the video analysis is facing is a particle detection and noise reduction. To retrieve statistical information from video sources requires the most accurate video analysis with minimal distortion level. Particle filtering and noise reduction use adaptive target model with prediction and correction calculations [8]. To achieve the most accurate results, the color distribution between different gradient positions in the image should be minimal and pixels nearest the corners should weight less. The effectiveness of the algorithm depends on the density of pixels in video source,

IVUS 2020: Information Society and University Studies, 23 April 2020, KTU Santaka Valley, Kaunas, Lithuania

zygimantas1997@gmail.com (Z. Matusevičius); gudauskas.julius@gmail.com (J. Gudauskas)

but the high level of intrinsic robustness ensures the proper performance and accurate filtration. After analyzing the possible image processing algorithms, tools, and technical requirements for preferred material, it was decided that the C# programming language and EmguCV [9] (a cross-platform .Net wrapper to the Open CV image processing library) contains a needed and adaptable functionality to the project. An image processing library can do plenty of things. EmguCV functionality provides a prepared video source filtration in different color formats and let us filter out objectionable objects. These are the main image-processing operations that provide the majority of needed information that could be used in further calculations and problem-solving. In this case, the main information source needed from the video stream is ball positioning coordinates in each frame.

3. Goalball Ball Movement Tracking Algorithm

In order to get a real playing field proportion coordinates of the ball positioning and movement in the playing field, it is necessary to use a transformation algorithm. Requirements for the method:

1. Sufficient vector recognition > 90%
2. Real-time performance

To ensure a correct method performance regardless of camera position, a playing zone calibration is needed. Selected corners of the area will later assist in calculating the position of the object and determine the suitability of the coordinates.

The movement of the object within the calibrated area is monitored by examining each frame of the video material and subsequently systematizing the data.

The method pseudocode:

```
Calibrate game zone.
Initialize game zone While frame isn‘t
null.
Read frame.
Get coordinates of object in video
frame.
Convert video coordinates to real game
zone coordinates.
Group coordinates into vectors.
```

3.1. Game Zone Preparation

Since the order of the marked playing zone corners points is unknown, we arrange them in their y coordinate and then in the x coordinate. Sorting of the points will guarantee the same order, regardless of the order, they were marked. Once we know the coordinates of each game area corner, the calculation can be done for the bottom edge midpoint of the game area and for the bottom edge length - these parameters will be necessary for further calculations. Now we can formulate equations describing the edges of an untwisted playing area. These equations will be used in filtering out points outside the playing area.

$$sina = \frac{Y_{BottomRightCorner} - Y_{BottomLeftCorner}}{Lengthof\,bottomside} \tag{1}$$

$$cosa = \frac{X_{BottomRightCorner} - X_{BottomLeftCorner}}{Lengthof\,bottomside} \tag{2}$$

Using the calculated parameters and the marked area corner coordinates, we can calculate the sin and cos values of the rotation angle for the playing area. Using the calculated trigonometric values, we apply the rotation transformation to the coordinates of the playing area corners.

$$x' = x\cos a + y\sin a \tag{3}$$

$$y' = -x\sin a + y\cos a \tag{4}$$

In order to improve the accuracy of the algorithm, we adjust the coordinates of the game zone upper corners: the ordinates of the game zone upper corners must coincide and the abscissa must be symmetrical to the vertical line passing through the middle point of the game zone lower edge. Once we have the exact coordinates of the corners, we can formulate equations that describe the rotated playing area edges. These boundary equations will later be used to perform coordinate transformations. Equations that used to calculate transformation coefficients are formed as well.

The method pseudocode:

```
Order corners by y
Then order corners by x
Find middle point of bottom side line
Calculate length of bottom side line
Calculate rotation sin and cos
Make equations for side lines in video
frame
Rotate corners
Fix top corners
Make equations for side lines in rotated
game zone
Make equations for calculating scale
coefficients
```

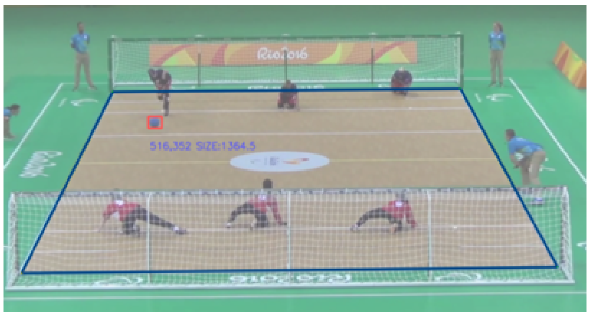

Figure 1: Game zone distortion from camera view.

3.2. Receiving of Object Coordinates

Once the game zone calibration is done, a ball tracking algorithm will begin to track the ball in the video stream. Object tracking is performed using every frame of the video stream and color filtration. In order to recognize objects by color, a frame has to be converted into HSV format. Then, applying the color filter, we can get a binary image where the color of interest is replaced by white and everything else is black. The binary image may have a lot of "noise" and unnecessary objects. To reduce it, we apply "noise" reduction operations [10, 11]. The frame processed in this way is already suitable for searching objects in it.

The method pseudocode:

```
While frame isn‘t null
    Read frame
    Convert frame from RGB to HSV
    Convert frame from HSV to binary by
filtering target color
    Reduce noise in binary video frame
    Get coordinates of object in binary
video frame
    If object size between MinSize and
MaxSize
       If object is in game zone
         Collect object‘s coordinates
```

3.3. Coordinates Transformation

As a result of video processing, we receive the coordinates of the ball in the distorted video frame (Fig. 1). To get coordinates in proportion to the real playing field, we have to perform a sequence of geometric transformations.

First of all, a rotation transformation is performed, which transfers the coordinates of a point from the designated coordinate system of the playing area to the rotated coordinate system that represents a real

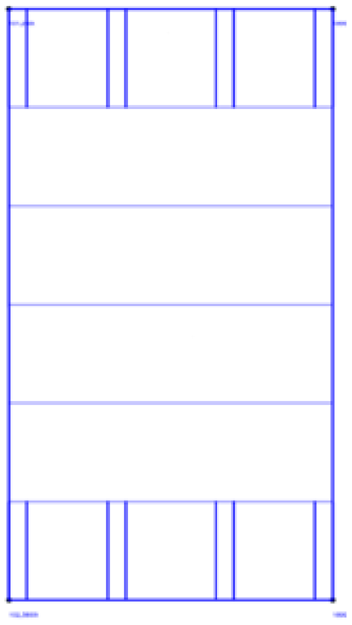

Figure 2: Game zone view after calibration.

playing field proportions. This transformation eliminates the asymmetry of the camera angle. Secondly, the coordinate reference point is moved to the midpoint of the playing field lower edge. Then a vertical and horizontal scaling factors are calculated and applied (in this case, coefficient equations are used). The scaling factors help to "stretch" the trapezoidal area of playing field from the footage into a rectangle form that is proportional to the real playing field (Fig. 2).

The method pseudocode:

```
Rotate point
Match up starting point with middle
point of bottom side line
Calculate vertical scale coefficient
Apply vertical scale coefficient
Calculate horizontal scale coefficient
Apply horizontal scale coefficient
```

3.4. Movement Vectors Analysis

Once the ball coordinates in the playing area are obtained, they need to be grouped into vectors. The main goal is to add received points into the vectors with the condition: point is added to a vector only when the distance between the point and the end of the vector is appropriate. If a new point is added to the vector, then the state of the vector is updated: the vector equation is corrected and an attempt to complete the vector is executed . Finally, all the completed and eligible vectors moved to the list of completed vectors. If the point cound not be added to the vector, a new vector is created, whose starting coordinates become the point that could not be added to a vector

The method pseudocode:

```
Foreach vector in remaining vectors
    Try add point into vector
      If distance between point and end
of vector is suitable
       Add point into vector
       Adjust equation of vector
       Try to complete vector
If point was not added into any vector
    If ordinate of point is between
MinVal and MaxVal
      Create new vector
      Add point to new vector
Else
    Filter completed vectors
    Filter suitable vectors from
completed vectors list
    Add suitable vectors into finished
vectors list
    Remove completed vectors from
remaining vectors list
```

4. Results and Discussion

For the research and testing of the software solution, two different video sources were selected, each was rendered into different frames per second ratio – 15 fps, 30 fps and 60 fps.

The main goal of the analysis is to investigate the accuracy of the algorithm, regarding the requirements mentioned in third section. First selected video for the research was male match between Algeria and Germany teams in Rio Olympics, 2016. To get the accuracy of the algorithm, we calculated the total throws performed by each team by hand and compared the results with the output after software processing. The comparison results displayed in chart (Fig. 3).

From the following data, we can calculate the accuracy of recognized vectors and get the average accuracy in this match. Results was calculated in different frame ratio and displayed in chart (Fig. 4). A second selected video for the research was female match between USA and Japanese teams in Rio Olympics, 2016. Using the same calculation principle, the number of throws was calculated by hand and compared with the result of the software analysis in different frame ratio.

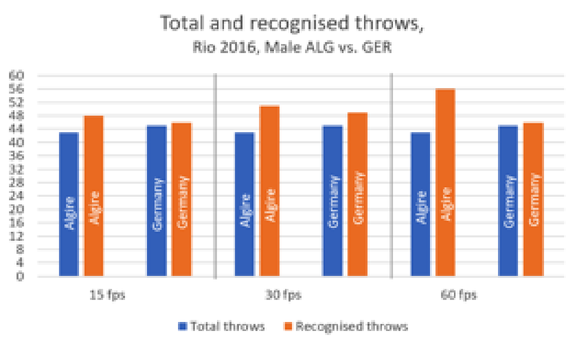

Figure 3: Results of total and recognized throws in video source (RIO 2016, ALG vs GER (Male)).

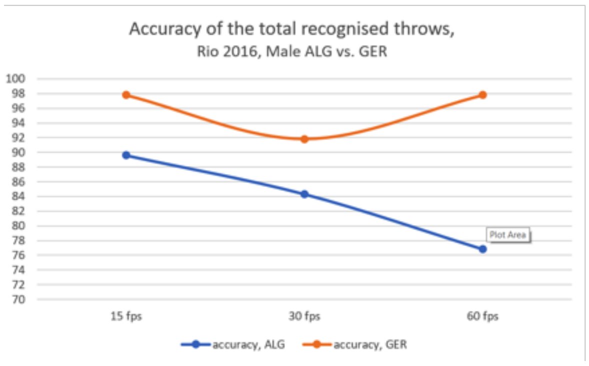

Figure 4: Accuracy of the recognized vectors in video stream (RIO 2016, ALG vs GER (Male)).

The comparison results and calculated accuracy was displayed in graphs (Fig. 5, Fig. 6, Fig. 7, Fig. 8).

Other important factor of the algorithm accuracy is the time duration that it takes to calculate throws from video stream. Analyzing the first video from Rio Olympics 2016, male match Algire vs. Germany, the time duration that it took to analyze the entire video in 15 fps (original video length 20:36), was 17:08 (video stream had a 120% of the original speed), meanwhile in the 30 fps video it took 26:40 (video stream had a 77% of the original speed). The analysis in 60 fps took 1:00:24 (video stream had a 49% of original video speed).

Analyzing the second video from Rio Olympics 2016, female match USA vs. Japan, the time duration that it took to analyze the entire video in 15 fps (original video length 46:18), was 31:55 (video stream had a 145% of the original speed), meanwhile in the 30 fps video it took 58:24 (video stream had a 79% of the original speed). The analysis in 60 fps took 1:56:39 (video stream had a 40% of original video speed).

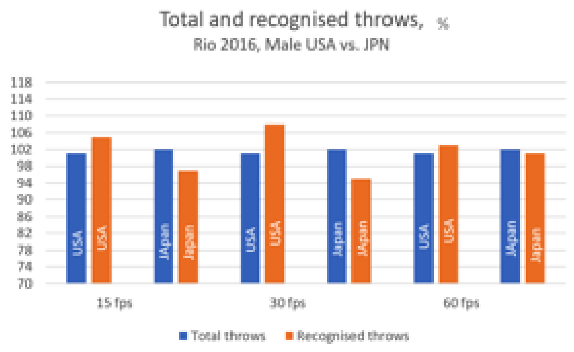

Figure 5: Results of total and recognized throws in video source (RIO 2016, USA vs JPN (Female)).

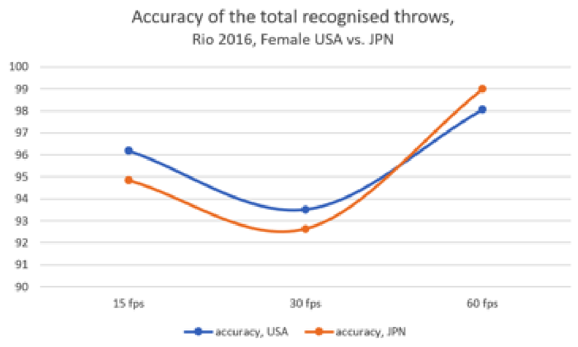

Figure 6: Accuracy of the recognized vectors in video stream (RIO 2016, USA vs JPN (Female)).

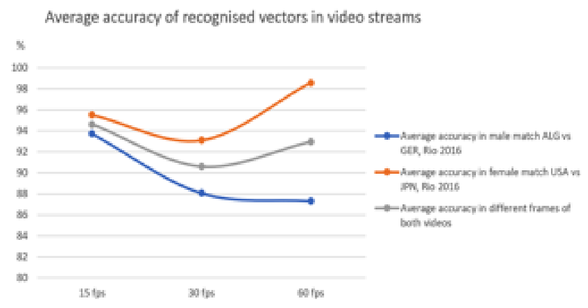

Figure 7: Average accuracy of recognized vectors in each video stream.

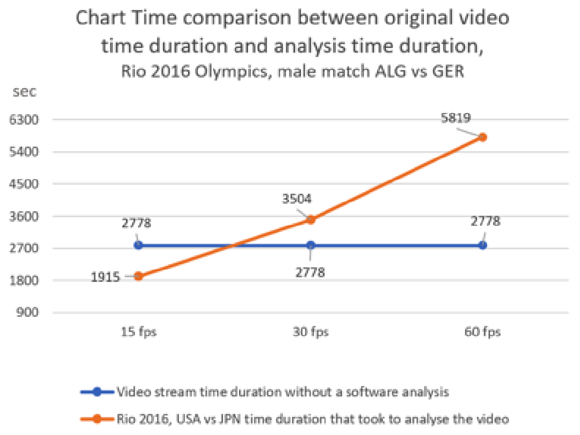

Figure 8: Time comparison (RIO 2016, ALG vs GER (Male)).

References

[1] G. Capizzi, G. Lo Sciuto, C. Napoli, D. Polap, M. Woźniak, Small lung nodules detection based on fuzzy-logic and probabilistic neural network with bio-inspired reinforcement learning, IEEE Transactions on Fuzzy Systems 6 (2020).

[2] S. Spanò, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, A. Nannarelli, M. Re, An efficient hardware implementation of reinforcement learning: The q-learning algorithm, Ieee Access 7 (2019) 186340–186351.

[3] Once football video analyser, 2020. URL: https://www.once.de.

[4] Hudl application, 2020. URL: www.hudl.com.

[5] Home court app, 2020. URL: https://www.homecourt.ai/.

[6] C. Tomasi, T. Kanade, Detection and tracking of point features, School of Computer Science, Carnegie Mellon Univ. Pittsburgh (1991).

[7] D. Chatterjee, S. Chandran, Comparative study of camshift and klt algorithms for real time face detection and tracking applications, in: 2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), IEEE, 2016, pp. 62–65.

[8] A. H. Sayed, P. M. Djurić, F. Hlawatsch, Distributed kalman and particle filtering, in: Cooperative and Graph Signal Processing, Elsevier, 2018, pp. 169–207.

[9] Emgu cv wiki page, 2020. URL: http://www.emgu.com/wiki/index.php/Main_Page.

[10] G. Capizzi, S. Coco, G. Lo Sciuto, C. Napoli, A new iterative fir filter design approach using a gaussian approximation, IEEE Signal Processing Letters 25 (2018) 1615–1619.

[11] R. Damaševičius, C. Napoli, T. Sidekerskienė, M. Woźniak, Imf remixing for mode demixing in emd and application for jitter analysis, in: 2016 IEEE Symposium on Computers and Communication (ISCC), IEEE, 2016, pp. 50–55.