Segmentation and tracking in football video

Peter Zajac

Goals of this semester project was to achieve these goals:

- *Detect objects(players, referees) on football video.*
- *Track object's movement in video.*
- *Classify objects into groups(teams).*

Data
As my dataset i chose 2 minute broadcast of football match, consisting of 3 static cameras, recording whole pitch.

Process of detection of objects

Preprocessing of data
Load 3 videos and connect them, getting image of whole pitch as result, also applying **perspective transformation** on video to get better overview.

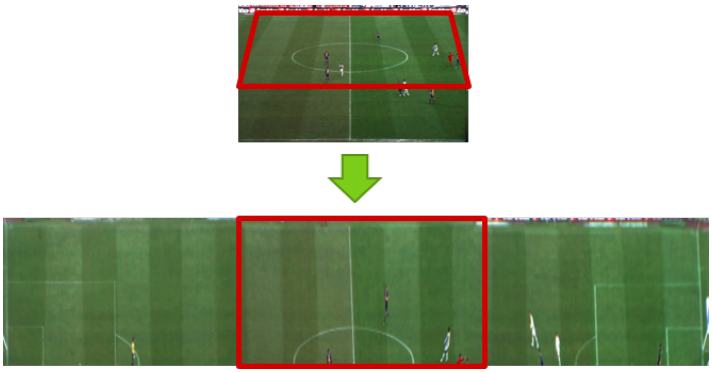

Figure 1 - Perspective transformation of middle part of pitch

Segmantation of objects
I used **background substraction** technique to get foreground mask of moving objects on pitch. I used MOG2 substractor. I also used logic AND of bites with mask of field, done on HSV extraction of 'green' color of pitch.

Figure 2 – Background segmentation and HSV extraction of green field

Smoothing and removing of noise
I used morphological operations - closing, opening to remove noise from mask.

Figure 3 - Mask after closing and opening.

Getting objects from foreground
I used detection of contours to get objects from foreground. I filtered contours based on regular size of objects (bounding box size of contour) i wanted to get (players, referees).

```
01. def filtercontours():
02.     playercontours = list()
03.     for c in contours:
04.         rect = cv2.boundingRect(c)
05.         # if contour is too big or too small its not player
06.         if (rect[2] < 7 or rect[3] < 20) or (rect[2] > 60 or rect[3] > 100): continue
07.         playercontours.append(c)
08.     return playercontours
```

Figure 4 - Code snippet of contour filtering

Classifying objects into groups
I labelled objects into 3 groups - team A, team B, referees.
I did it based on 2 methods:

- Mean color of contour
- Mask of label

Mean color of contour
I calculated mean color of contour and tried 2 ways to divide it into labels: **kmeans** and **threshold value comparison**.

Kmeans was more automated but had issues when some members of group disappeard from video so kmeans was working on either 1, 2 or 3 labels.

Simple threshold comparison of color label was decent enough to divide players based on their mean color. Problem of that method was that some connected objects invalid mean colors, if 2 players of opposing teams connected together in mask.

```
01. def classifycontours(contours):
02.     classifiedObjects = {}
03.     ateamplayers = list()
04.     bteamplayers = list()
05.
06.     for c in contours:
07.         rect = cv2.boundingRect(c)
08.         x, y, w, h = rect
09.         crop_img = bigFrame[y:y + h, x:x + w]
10.         # compute mean color with mask of background for better reults
11.         meanColor = cv2.mean(crop_img, mask =mask)
12.         # comparison of Green color range is threshold for labels
13.         if meanColor[1] > 100:
14.             ateamplayers.append(c)
15.         else:
16.             bteamplayers.append(c)
17.
18.     classifiedObjects['ateam'] = ateamplayers
19.     classifiedObjects['bteam'] = bteamplayers
20.     return classifiedObjects
```

Figure 5 - Code snippet of classifing contours based on mean color

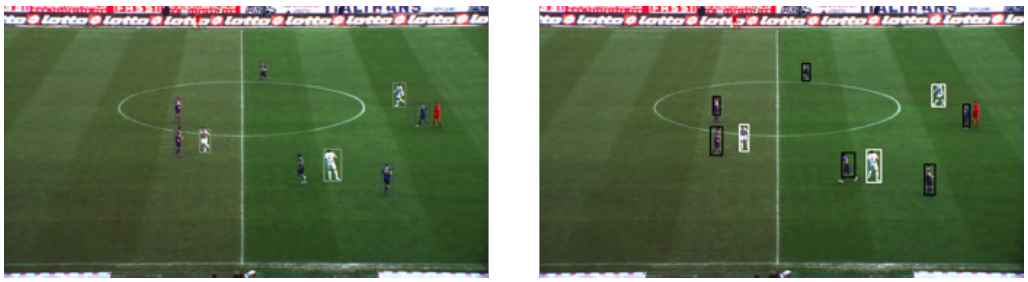

Figure 6 - Left - mean color bounding box, Right classified labels of players

Mask of label

I created this method for labelling referees, based on HSV range of their color(orange) and processing with **bitwise AND** with foreground mask. That way i got labelled referees. In comparison to mean color method, this was giving better results if referees were overlapping with another players.

Figure 7 - Mask of orange - referee label, extracted by HSV extraction

Tracking of objects

To track movement of objects i used **Mean shift algorithm**. First you choose windows (contour of player) and then its computed histogram of windows used for Mean shift algoritm. Histogram is calculated on mask of object label (white, blue) so even if player overlap with player from another team, tracking is working, it causes problems just when 2 players from same label overlap.

```
while(1):
    ret ,frame = cap.read()

    height, width, layers = frame.shape
    new_h = int(height / 2)
    new_w = int(width / 2)

    frame = cv2.resize(frame, (new_w, new_h))

    if ret == True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)

        # apply meanshift to get the new location
        ret, track_window = cv2.meanShift(dst, track_window, term_crit)

        # Draw it on image
        x,y,w,h = track_window
        #print bounding box of tracking object
        img2 = cv2.rectangle(frame, (x,y), (x+w,y+h), 255,2)
```

Figure 8 - Code snippet of Mean Shift tracking

Figure 9 - Left - Mask of label (white players), Right - tracked player with MeanShift

Summary

For segmentation I used following techniques:

- Background substraction
- HSV color extraction mask
- Contour detection and filtering
- Labelling objects
    - K-means, threshold
    - Mask of labelled objects
- Tracking objects
    - Mean-shift algoritm – on mask of labelled objects

Additions

Link to GitHub with source code of project: https://github.com/peterzajac39/football-processing-opencv/tree/master.