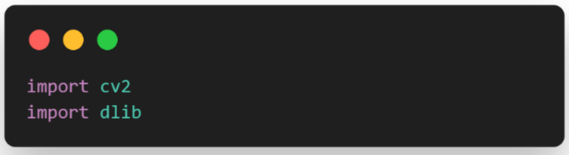
TRACKING MOVING OBJECTS WITH ARTIFICIAL VISION

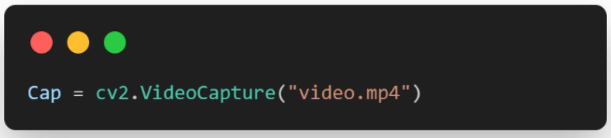
Maybe you have seen in some occasions videos which show a rectangle tracking a car, a person or anything in moving, and if you have asked how is it possible? Well, in this paper I will teach you a way of tracking moving objects with artificial vision. Once you understand how this works, you could track anything to be in motion of an easy way.

For this code we will use two libraries, which are Open CV and DLIB; we will import these libraries, and once done this we can start.



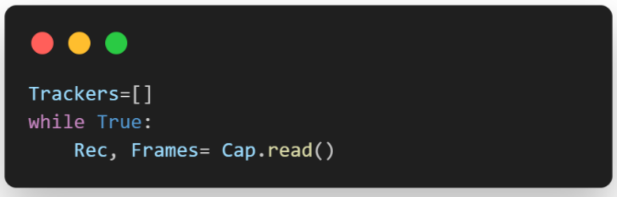
```
import cv2
import dlib
```

Now we will go to initialize our video capture with the function .videoCapture() and as parameter the name of the video we will use and we will have saved in a folder with the code, in case of it will do with a camera in real time we will use the variable of the camera.



```
Cap = cv2.VideoCapture("video.mp4")
```

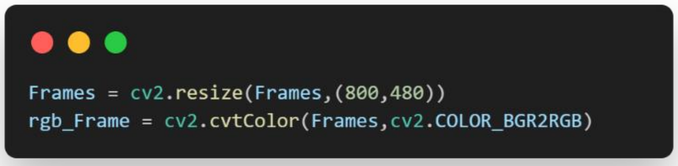
Before continuing, we will declare a variable, called in this case Trackers, which will be an empty list, that we will use later to obtain some important values. Once we have declared this variable, we will initialize the while-loop, and we will read our frames with the function .read().



```
Trackers=[]
while True:
    Rec, Frames= Cap.read()
```

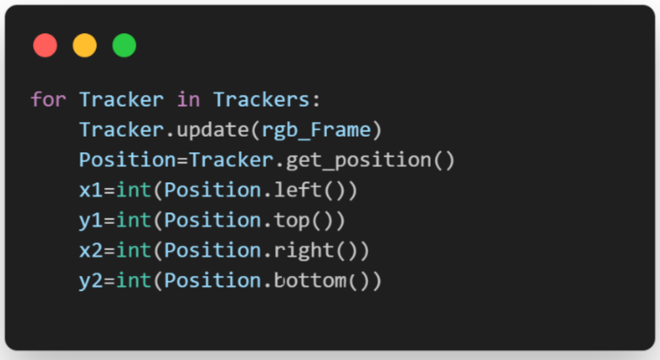
We are going to apply a resize to our frames, because the video is so big, whereby we will use the function .resize() and as parameters the variable that we want to resize and the values of width and height in pixels, which we want to do the resize.

Once we have this resize, we will apply a color scale change, because Open CV reads the frames as BGR by default, and we will need them in RGB for when we will want to obtain the position later, for this we will use the function .cvtColor() and for the parameters the variable to change and the change type.



```
Frames = cv2.resize(Frames,(800,480))
rgb_Frame = cv2.cvtColor(Frames,cv2.COLOR_BGR2RGB)
```

To use the empty list that we have declared earlier, we will create a for-loop to get the positions.



```
for Tracker in Trackers:
    Tracker.update(rgb_Frame)
    Position=Tracker.get_position()
    x1=int(Position.left())
    y1=int(Position.top())
    x2=int(Position.right())
    y2=int(Position.bottom())
```

What does this for-loop do? Well, first we have the variable Tracker that will iterate in Trackers, variable that we declared as empty list.

To this variable we will update through the function .update() and as parameter the variable with the change of scale from BGR to RGB, now, why did we do this update? Well this is because we have a video with movement and as the objects constantly change their position, we need to get the updated position of these changes, this for a better estimation of the position in real time.

Next, the function .get_position() will allow us to obtain the actual position that then it will be collocated in the coordinates with the variables: x1, y1, x2, and y2; these variables will give us the coordinates thanks to the functions: .left(), .top(), right() and .bottom() respectively.

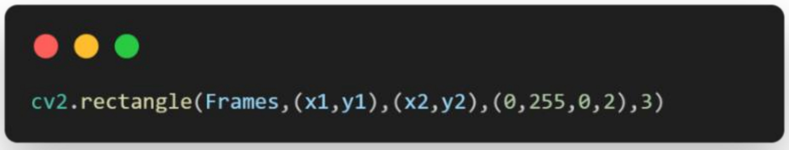
With these positions, we will create a rectangle that we will display when we follow the object in motion.

This will be created with the function .rectangle() and the next elements as parameters:

- The image in which we want to display the rectangle.
- The coordinates in *x* y *y* from the start.
- The coordinates in *x* y *y* from the end.

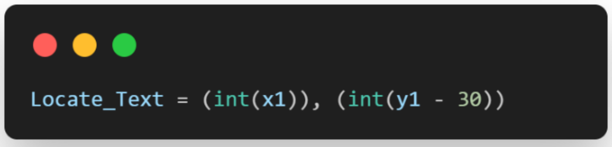
- The color of the rectangle.
- The thickness of the rectangle.

For the values of the coordinates, we will use the values obtained in the for-loop.



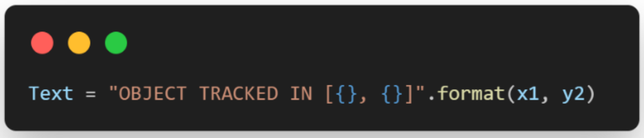
```
cv2.rectangle(Frames,(x1,y1),(x2,y2),(0,255,0,2),3)
```

Now, we will display a text with the coordinates where the tracked objects are located, to do this we will create a variable with the position where we want to display the text, in this case we have taken as reference the variables x1 and y1 subtracting a value of 30 from the latter.



```
Locate_Text = (int(x1)), (int(y1 - 30))
```

Having this, we will create another variable with the text that we want to display, for this we will use the method .format(), this method will help us to insert a variable as text thanks to the use of a list and curly brackets "{}", inside these curly brackets will be inserted the variables provided as parameters in the method.

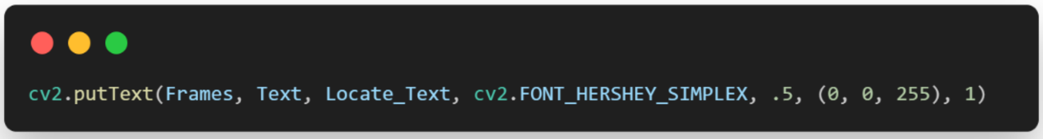


```
Text = "OBJECT TRACKED IN [{}, {}]".format(x1, y2)
```

Now we just need to display this text, and this will be done with the function .putText() and as parameters we will use the next elements:

- The location where the text will be displayed.
- The text to display.
- The position where the text will be displayed.
- The font.
- The font size.
- The color of the text.

- The thickness of the text.



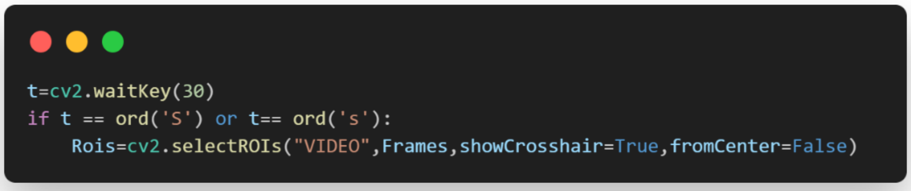
```
cv2.putText(Frames, Text, Locate_Text, cv2.FONT_HERSHEY_SIMPLEX, .5, (0, 0, 255), 1)
```

Now we will create some commands of indication through the keyboard, for this we will use the function .waitKey(), we will create a variable in which the reading time will be stored, once we have this variable, we will create a conditional if in which we will indicate that: if the "s" key, both lowercase and uppercase is pressed, the process of selection to the area of interest can start, this through the function .selectRIOs(), which has as parameters:

- The name of the window.
- The location where the selection will be displayed.
- The showCrosshair boolean parameter.
- The fromCenter boolean parameter.

If showCrosshair is True, the tracking rectangle is display with a kind of crosshair, if it is False only the tracking rectangle is displayed.

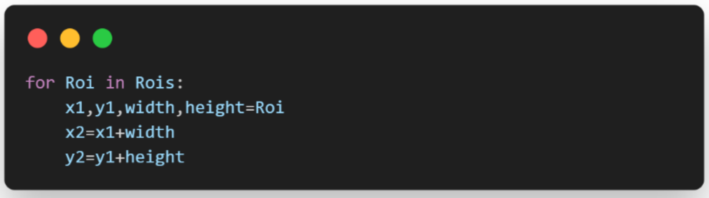
For the case of fromCenter, if it is True the selection will be done of central way extending for both sides, in case of it is False, it will be according to the selection goes extending.



```
t=cv2.waitKey(30)
if t == ord('S') or t== ord('s'):
    Rois=cv2.selectROIs("VIDEO",Frames,showCrosshair=True,fromCenter=False)
```

Now, we will have a for-loop where the variable Roi will be iterated in the variable Rois (variable to contain the selection), each iteration of Roi will obtain the coordinates of the selection where the variables x1, y1, width and height are equal to the variable Roi.

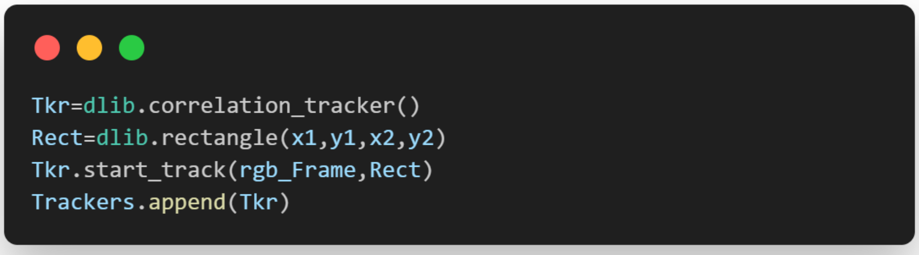
To obtain x2, that it is the second point of the rectangle we will use the coordinate of x1 plus width and for y2, the opposite, we will use y1 plus height.



```
for Roi in Rois:
    x1,y1,width,height=Roi
    x2=x1+width
    y2=y1+height
```

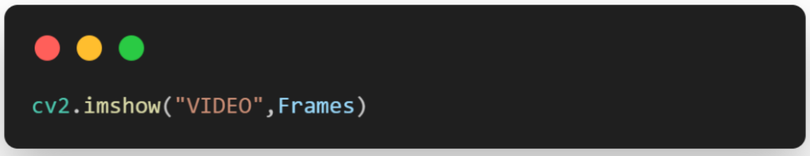
To finish the for-loop, we will use the library dlib, this library contains an algorithm of tracking objects that we will use, first we will initialize the function .correlation_tracker(), this function compares the similarities of the selection of the area of interest of the successive frames. Having done this, we will create a rectangle through the function .rectangle() and the variables x1, x2, y1 and y2, this rectangle will be the one in charge of compare the similarities.

Continuing with the loop, we will initialize the tracking with the function .start_track(), as parameters we will use the frames in RGB and the rectangle that we created with the function .rectangle(), and to finish we just have to add that we have done in the loop to our variables Trackers, this thanks to the function .append().



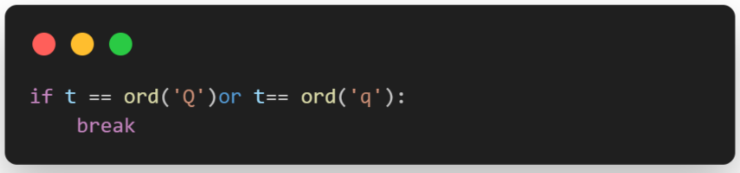
```
Tkr=dlib.correlation_tracker()
Rect=dlib.rectangle(x1,y1,x2,y2)
Tkr.start_track(rgb_Frame,Rect)
Trackers.append(Tkr)
```

Having this, we can display our video and to start the tracking we will use the function .imshow() with the name that the window will have and the variable that we will display as parameters.



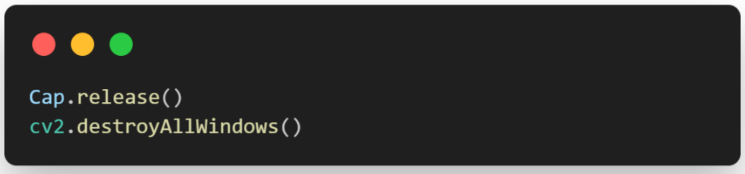
```
cv2.imshow("VIDEO",Frames)
```

We will create our exit command, which in this case will be to press the “q” key, either lowercase or uppercase.



```
if t == ord('Q')or t== ord('q'):
    break
```

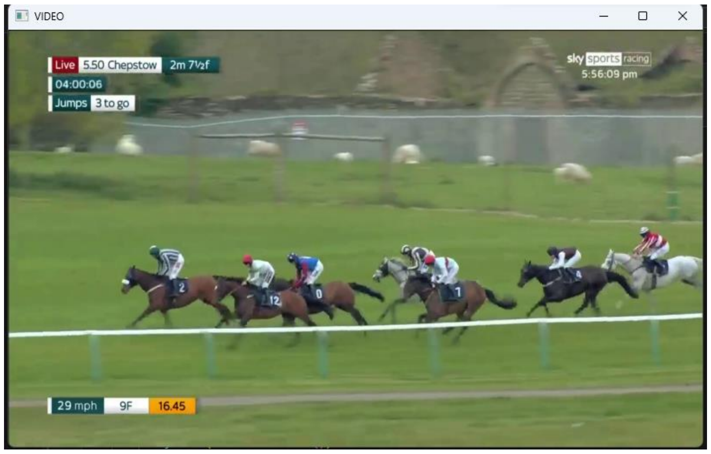
Finally, we will release the video capture with the function .release(), and we will destroy the windows with .destroyAllWindows().



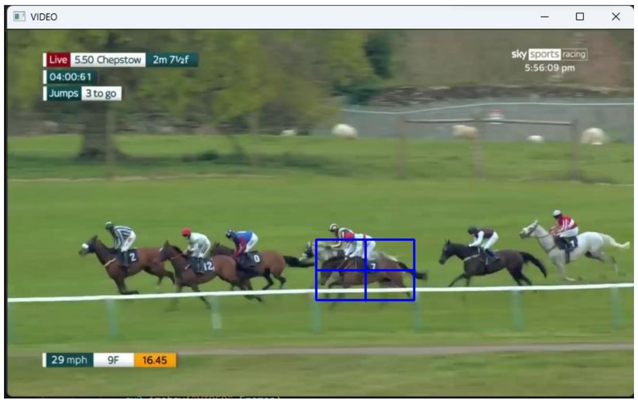
```
Cap.release()
cv2.destroyAllWindows()
```

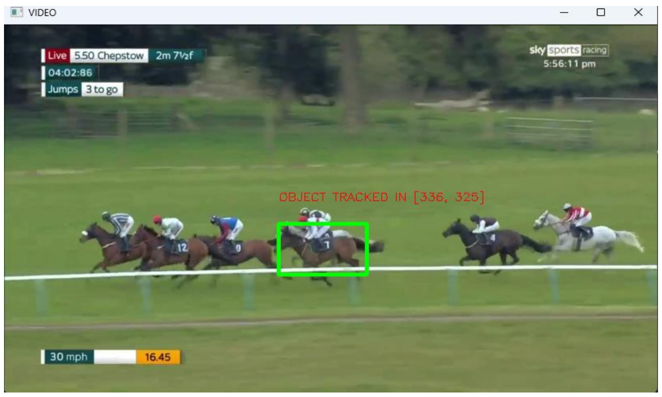
RESULTS OBTAINED

The first image is the video starting, in this video is where we will do the commands that we have done in the code.



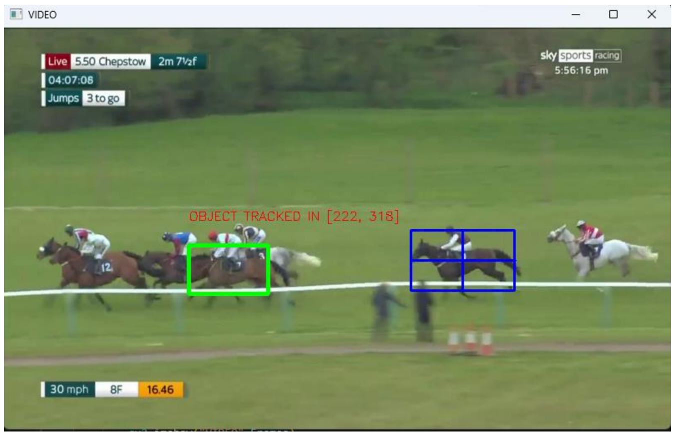
If we press the “s” key the video will be paused and we could select the area to tracking, next of select the area press the “Enter” key to run the video and see the area with the text that we have programed.

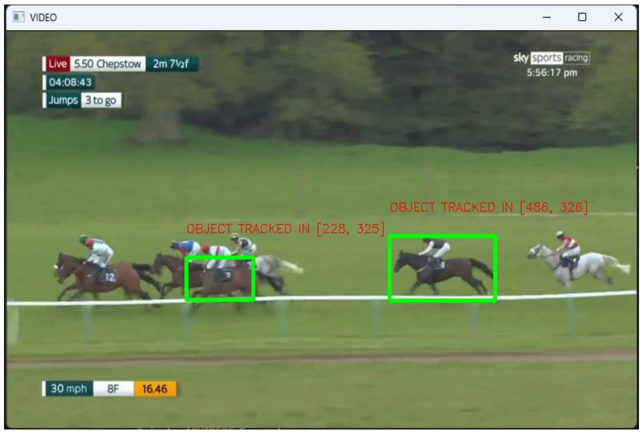




```
OBJECT TRACKED IN [336, 325]
```

This process could be repeated and we will have tracking two or more objects.




VIDEO
Live 5.50 Chepstow 2m 7½f
04:08:43
Jumps 3 to go
sky sports racing
5:56:17 pm
OBJECT TRACKED IN [486, 326]
OBJECT TRACKED IN [228, 325]
30 mph 8F 16.46

If you got to this part, I thank you for the time that you gave me and I hope that I have helped to you with you doubts and concerns of learning more, this is just a small fragment of explain if you want to continue learning and improve you can visit these pages that will help you in the topic of artificial vision and Python.

1.- https://omes-va.com/

2.- https://www.pythonpool.com/

3.- https://hetpro-store.com/TUTORIALES/

4.- https://learnopencv.com/

Wisdom is the principal thing; therefore get wisdom: yea, with all thou hast gotten get understanding. Proverbs 4:7