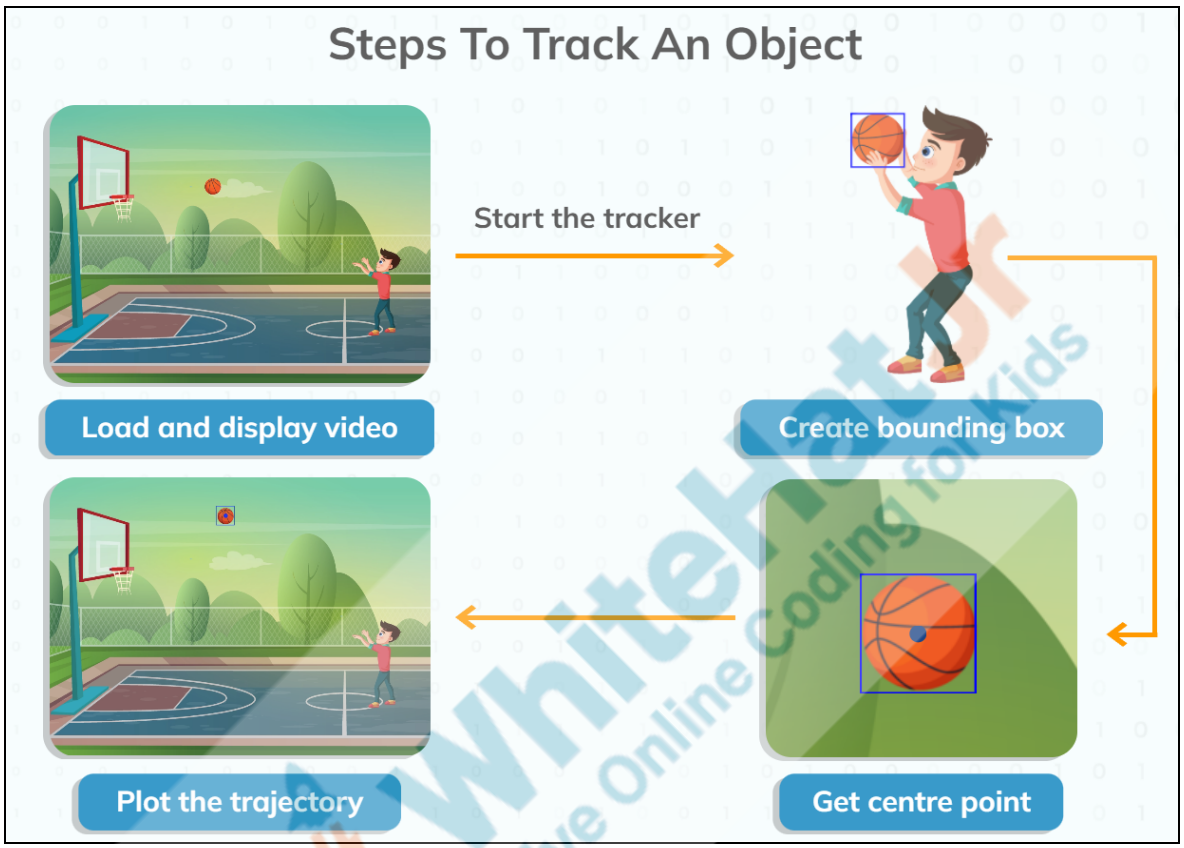
Topic	OBJECT DETECTION & TRACKING	
Class Description	**The student will learn to detect an object in a video, and they will track the object to plot its trajectory.**	
Class	**PRO C107**	
Class time	**45 mins**	
Goal	• Learn about object detection. • Plot the trajectory of an object.	
Resources Required	• Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
WARM-UP SESSION - 10 mins		
Teacher Starts Slideshow  **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.		
Teacher Action		**Student Action**

Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 4 to 16** 	
Following are the session deliverables: • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Ends Slideshow 	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
ACTIVITY • **Understanding Object Tracking and Object Detection** • **Use OpenCV CRST tracker to create a box on the tracked object.**	
Teacher Action	**Student Action**
In the previous class, we learned how to detect faces in the images and videos.	

But we were only detecting the faces. In this class, we are going to learn how to detect any object in a video and track the path of that object using OpenCV. Before we begin, let's understand object tracking. ***Note**: Open Visual Aids while explaining the concepts to the students.* **Object Tracking:** Object Tracking is a **process of finding and keeping a track of the position** of an object that is continuously moving in a video. To track an object in any video, the object has to be detected first and then we need to keep track of the position of the object. Can you tell me then how object tracking is different from object detection? **Object Detection vs Object Tracking:** The process of object detection involves only detecting an object in a still image(or one frame at a time in a video). Wherein the process object tracking involves keeping track of the object's position in a video. With this we can understand that, the computer won't be able to track the object if it is not detected! Let's understand the overall steps to detect and track an object in video using OpenCV. 1. Load the video. 2. Start the tracker(the inbuilt algorithm in OpenCV). 3. Object Selection by creating a bounding box around the object using the mouse.	**ESR:** Varied.

4. Get the center point of the bounding box.
5. Plot the trajectory using center points.



Teacher downloads the Teacher activity 1 boilerplate code and opens it in the VS Code Editor.

***Note**: The below is boilerplate code to read and display video using OpenCV that we have covered in previous classes.*

1. **Load the video**:

```
import cv2
import time
import math

video = cv2.VideoCapture("bb3.mp4")

while True:
    check,img = video.read()

    cv2.imshow("result",img)

    key = cv2.waitKey(25)

    if key == 32:
        print("Stopped!")
        break


video.release()
cv2.destroyALLwindows()
```

Before we start coding, we need to first install a library. This is an extension of OpenCV which helps us in object tracking.

Can you tell me how to install a library for python?

Great!

***Note**: If there is more than one Python version (Python 2 or Python 3) installed in your system, specify the pip or pip3 respectively.*

pip install opencv-contrib-python==3.4.13.47

The teacher runs this command in the command prompt or

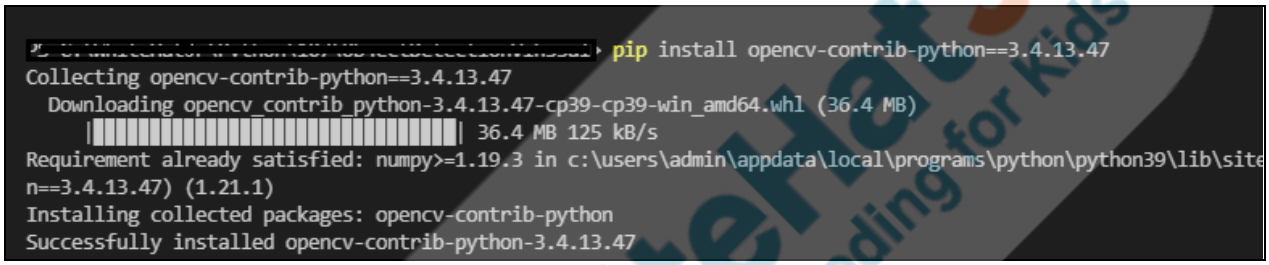
ESR:
Using pip command

terminal to install this library.

***IMP Note**: If the **installation fails**, try running the command **without specifying any version** as,*

pip install opencv-contrib-python

This could happen as the library version might NOT be compatible with the Python version installed in your system.



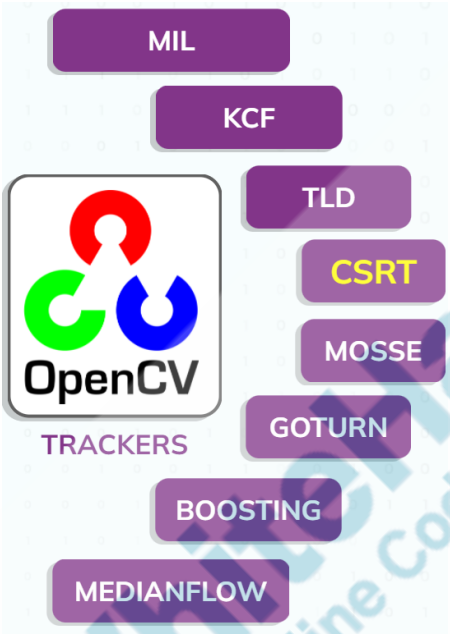
2. **Algorithm that will track the object:**

Next step is to load the tracker algorithm.

These are pre-built Machine Learning(ML) algorithms in the OpenCV library that can be used in our application. In upcoming classes we will build our own ML algorithms from scratch!

There are different trackers available in the OpenCV library such as:

- **BOOSTING**
- **MIL**
- **KCF**
- **TLD**
- **MEDIANFLOW**
- **GOTURN**
- **MOSSE**

Each tracker has some pros and cons such as MOSSE is very fast in tracking objects, but it lacks accuracy for our use, so we are going to use **CSRT**, because it is fast as well as accurate.	
	
We will use the **cv2.TrackerCSRT_create()** method to load the **CSRT** tracker. If you want to test some other tracker, then you only need to replace the **NAME** in the **cv2TrackerNAME_create()** method. Once the tracker is loaded, we need to load the first frame of our video. This is because when our code starts it should show an image from our video, then we can select any object by creating a bounding box on that image and our tracker will track the object present in the bounding box. We can use the **read()** method outside the **while** loop to read only the first frame of the video.	

returned, img = video.read().

This will load the first frame of our video and store that in the **img** variable.

```
video = cv2.VideoCapture("bb3.mp4")

# Load tracker
tracker = cv2.TrackerCSRT_create()

# Read the first frame of the video
returned, img = video.read()
```

3. **Object Selection creating a bounding box around the object using the mouse**:

The next step is to create a bounding box on this image for the object that we want to track.(Video Reference).

We will **track the basket ball** in the given video, hence the bounding box is drawn over the basket ball.

There is a very important concept to understand here that **to track the moving object in a video, we need to track**

it's position!

To track the position of a moving object, we will create a bounding box around the object, and use the bounding box to track the position of the object.

The bounding box determines the **Region Of Interest(ROI)** or the portion of the image that needs to be tracked.

To find out the ROI we have the OpenCV method **cv2.selectROI()** that helps us to select the region of the image during runtime(that means while the Python script is running).

This method takes three parameters:

- Name of this ROI(**R**egion **O**f **I**nterest),
- Image on which we create this ROI and I
- Boolean value(True/False) to set the origin of the bounding box, to rectangle to start from the center of the ROI selected then this will be **True** else it will be **False**.

The values returned by the **cv2.selectROI()** method is **bounding box data tuple (x, y, w, h)** where **x** and **y** are the coordinates of the starting point of the bounding box and **w** and **h** are width and height of the bounding box respectively.

We can store the result returned by the **cv2.selectROI()** method using a variable called **bbox**.

4. **Initialize the tracker(the inbuilt algorithm in OpenCV)**:

The last step of the process is to initialize the tracker. This is done using **tracker.init(img, bbox).** This function will initialize our tracker on the first frame of video using the bounding box created by us.

```
video = cv2.VideoCapture("bb3.mp4")

# Load tracker
tracker = cv2.TrackerCSRT_create()

# Read the first frame of the video
returned, img = video.read()

# Select the bounding box on the image
bbox = cv2.selectROI("Tracking", img, False)

# Initialise the tracker on the img and the bounding box
tracker.init(img, bbox)

print(bbox)
```

Run the code to see the output and then select the object to create the bounding box

Here we can see that the first frame of the video is loaded. Now select any object of this image using the mouse.(Video Reference).

To select the object:

1. Click and hold on the image where you want to start creating the bounding box.(We will track the basket ball in the given video, hence the bounding box is drawn over the basket ball).
2. Drag and drop to a point to complete the bounding box.

Once you are satisfied with your selection, you can press **Enter** to run the further program, which in our case only displays the video.

Video Reference

Output in Terminal: Bounding box (x, y, w, h) values

```
PRO C107>python object_tracking.py
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
(980, 400, 102, 60)
```

Now that we have the bounding box region ready, we need to keep the bounding box on every frame of the video.

To do that we need to continuously update the tracker in the **while** loop.

We will call the **tracker.update()** method on the target image which is **img** in our case.

The **tracker.update()** method updates the tracker on every frame of the video. It returns the **boolean value**(**True** or

False) and **updated bounding box data array [x, y, w, h]** where **x** and **y** are the coordinates of the starting point of the bounding box and **w** and **h** are width and height of the bounding box respectively. • Create a **success** variable to store the boolean value(True or False) and **bbox** variable to store bounding box data array returned by the **tracker.update()** method. If the detected object is in the frame, then it will be **True**, else it will be **False**. • If the **success** value is True, then we will write a function called **drawBox()** with two parameters, **img** and **bbox** array. • If the tracked object is not in the frame, then we will display the text on the video as "**Lost**"	

```
def drawBox(img,bbox):
    # Code to draw Rectangle and add Text
    

while True:

    check, img = video.read()

    # Update the tracker on the img and the bounding box
    success, bbox = tracker.update(img)

    if success:
        drawBox(img, bbox)
    else:
        cv2.putText(img,"Lost",(75,90),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

    cv2.imshow("result", img)

    key = cv2.waitKey(25)
    if key == 32:
        print("Stopped")
        break
```

Now let's define the **drawBox()** function:

Take four variables **x, y, w, h** to store the values of the **bbox** tuple after converting each value to **int**. This is important as the values returned will be of type float and to draw a rectangle we can only use integer values.

To convert each value to int, the values of bbox tuple can be accessed in a similar way as we access Python lists using index, for example the first value can be accessed using **bbox[0]** and converted to int as **int(bbox[0])**.

We will use the **cv2.rectangle()** method we used in the previous class, to draw the rectangle and **cv2.putText()** method to add text on the output window.

***Note**: The usage of **cv2.rectangle()** method and **cv2.putText()** method have been already covered in the previous class. Help the student to recollect how to use these methods.*

```
def drawBox(img,bbox):
    x, y, w, h = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])

    cv2.rectangle(img,(x,y),((x+w),(y+h)),(255,0,255),3,1)

    cv2.putText(img,"Tracking",(75,90),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)
```

Now let's run the code test the output
Object Tracking Output Reference

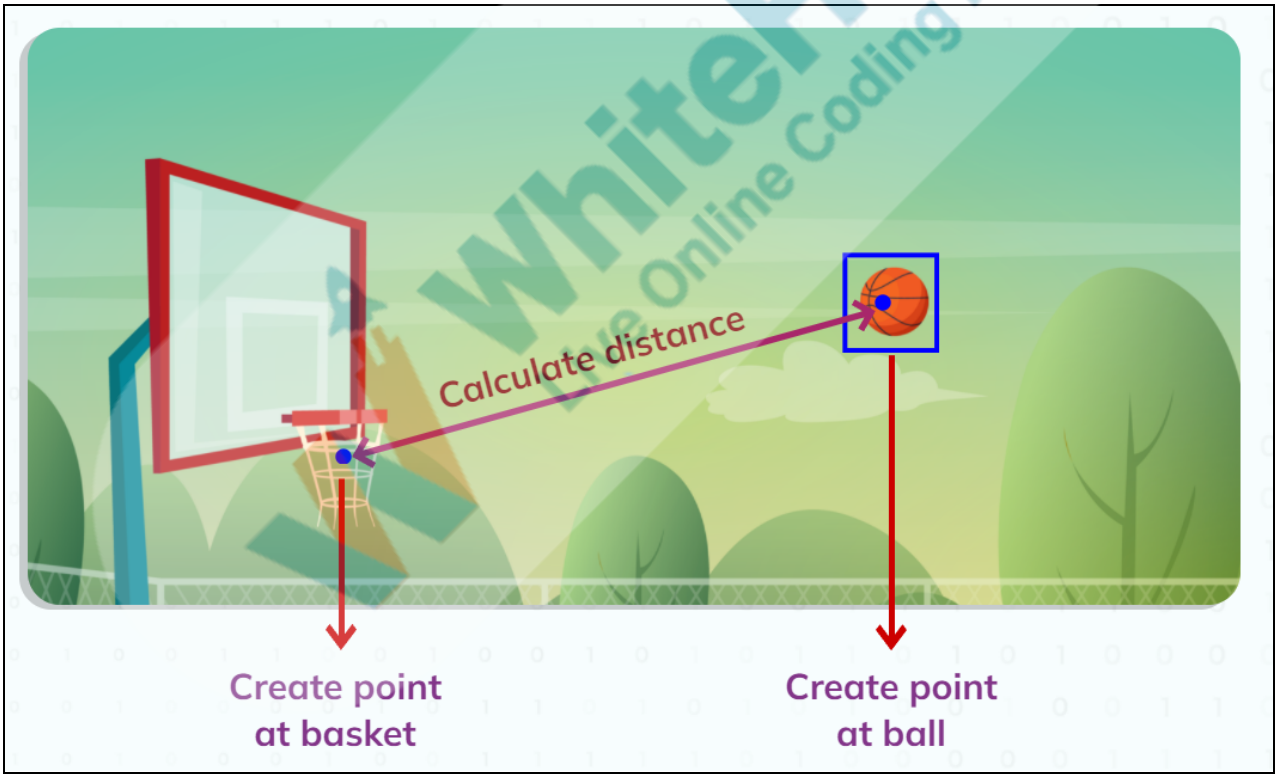
***Note**: If the bounding box drawn using is too small or too big, this might result in error. Re-run the script to create a new bounding box and check the output again.*

As you can see we are able to track the basket ball in the video using the OpenCV tracker. Now you can plot the trajectory(or path) followed by the basketball. Please share your screen with me.	
Teacher Stops Screen Share	
Teacher Starts Slideshow  **Slide 17 to 18** Refer to speaker notes and follow the instructions on each slide.	
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.	
Teacher Ends Slideshow 	
STUDENT-LED ACTIVITY - 20 mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Fullscreen.**	
ACTIVITY • **Code a condition to check if it is a goal or not** • **Plot the Trajectory of basketball.**	
In the previous activity, we created a tracker to track the basketball. Now we will plot the trajectory of the basketball in the video.	*The student downloads the Student Activity 1 code and opens it in the VS Code*

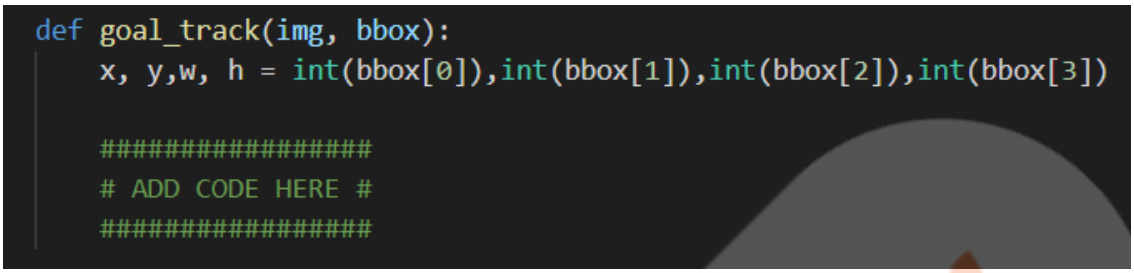
To do this we need to:

5. Get the center points:
    - Create a point at the center(approximately) of the **basketball hoop** and the **basketball bounding box**.
6. Plot the trajectory using center points
    - Calculate distance between the two points.
    - If distance is less than a value, then display the goal on the video.

editor.



Now we will write a function named **goal_track(),** plot the trajectory of the basketball. This function will take the **img**

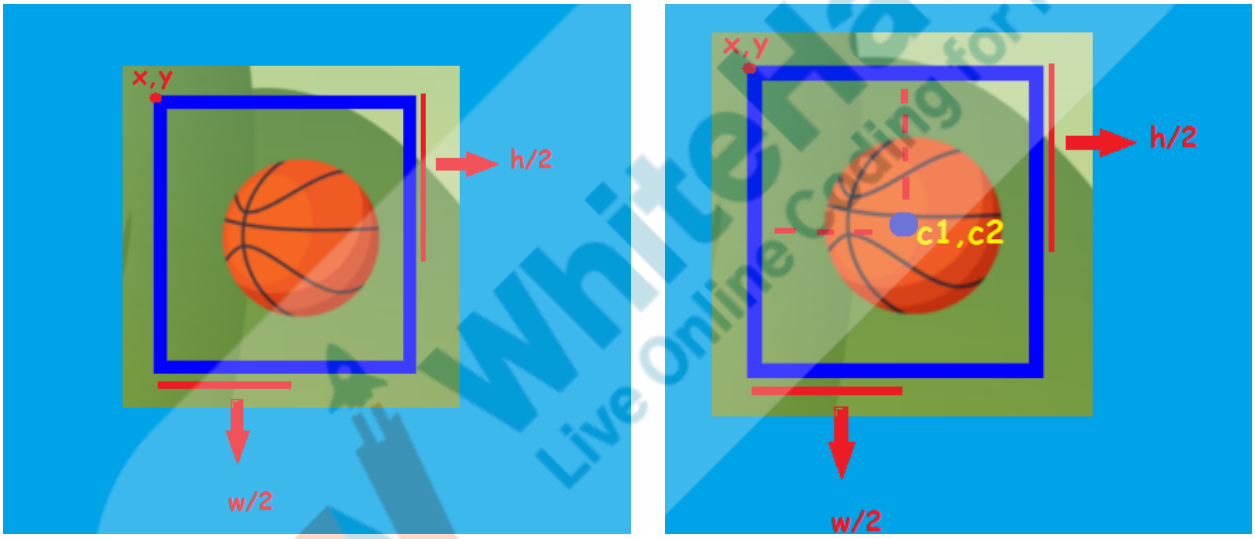
and bounding box**(bbox)** as input.	
	
To start with we need to locate the center of the basketball. Since the basketball is continuously moving, it's position is continuously changing. Can you think of how we can locate the center point of the continuously moving basketball? Amazing! The point will not exactly be a point, because OpenCV does not have a method to create a point. But we are going to draw a circle which will have a very small radius, and it will look like a point. To create a circle at the center of the bounding box which is tracking the basketball, we will get the **x, y, w, h** coordinates of the **bounding box** store in the variable **bbox**. The **center point of the basketball is located in the middle of the bounding box rectangle.** Can you tell me what can be done to get the middle point of any rectangle box?	**ESR:** We are tracking the position of the basketball using OpenCV tracker. We can use the values returned by the tracker to locate the center point of the basketball. **ESR:** Varied.

Middle point of any rectangle is always located at half of width and height from the starting point.

In the image below, **x** and **y** is the starting point located at top left on the bounding box. The middle point is represented using **c1** and **c2**.

***Note**: We need to keep these points as integers because OpenCV does not accept floating point(decimal) numbers.*

Then will draw a circle at the point **c1** and **c2**, using the **cv2.circle()** method.



Let's write the code to draw a circle at the center point of the basketball:

- Create a variable **c1** and **c2**

To get the center point of the bounding box, we will add half of width in x and half of height in y.

- Set the values of **c1** as x plus half the width of the bounding box.
- Set the values of **c2** as y plus half the height of the

bounding box

- Use **cv2.circle()** method to draw the circle.

  Parameters:
  - **image**: The image to draw the circle on.
  - **center points**: x and y coordinates of the center of the circle, which is represented by c1 and c2 in this case.
  - **radius**: Radius of the circle.
  - **color**: Color of the circle, red (0, 0, 255) in this case.
  - **thickness**: Thickness of the circle

```
def goal_track(img,bbox):
    x,y,w,h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    c1 = x + int(w/2)
    c2 = y + int(h/2)
    cv2.circle(img,(c1,c2),2,(0,0,255),5)
```

The next step is to create a circle at the basketball hoop. We have defined two points using the global variables, **p1** and **p2** such that **p1 = 530, p2 = 300.**

These points will be the center points of the circle on the basketball hoop.

***Note**: These points have been found out using hit and trial for the student.*

```
import cv2
import time
import math

p1 = 530
p2 = 300
```

Let's draw that circle again using the **cv2.circle()** method.

```
def goal_track(img,bbox):
    x,y,w,h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    c1 = x + int(w/2)
    c2 = y + int(h/2)
    cv2.circle(img,(c1,c2),2,(0,0,255),5)

    cv2.circle(img,(int(p1),int(p2)),2,(0,255,0),3)
```

Now let's call this **goal_track()** function in our **while** loop to see the output.

Here we can see points on the basketball and on the hoop.

```
while True:
    check,img = video.read()
    success,bbox = tracker.update(img)

    if success:
        drawBox(img,bbox)
    else:
        cv2.putText(img,"Lost",(75,90),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)


    goal_track(img,bbox)

    cv2.imshow("result",img)
```

Now let's find out the distance between these points to display whether it is a goal or not.

To calculate the distance we are going to use the distance formula. Mathematically, the formula is represented as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

X1, Y1 Represents the fist x and y coordinate.
X2, Y2 Represents the second x and y coordinate.

***Note**: We will deep dive into the mathematical calculation of the formula. Refer this for more information.*

To convert this into code, we will follow these steps:

- Subtract **(c1-p1)** and square the outcome **(c1-p1)**2**, here c1 is the center point of the bounding box and p1 is on the basketball hoop.

- Then subtract (**c2-p2**) and square the outcome **(c2-p2)**2**. C2 is the point on the bounding box and p2 is on the basketball hoop.

- At last, we will add the result and find the square root using the **math.sqrt()** function of the python math library.

After the distance calculation is done we will write a condition to check the distance value.

If the distance is less than **20** we will display "**Goal**"text on our video.

Now when you run the code you will be able to see that when basketball reaches the hoop it says goal.

```
def goal_track(img,bbox):
    x,y,w,h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    c1 = x + int(w/2)
    c2 = y + int(h/2)
    cv2.circle(img,(c1,c2),2,(0,0,255),5)

    cv2.circle(img,(int(p1),int(p2)),2,(0,255,0),3)
    dist = math.sqrt(((c1-p1)**2) + (c2-p2)**2)
    print(dist)

    if(dist<=20):
        cv2.putText(img,"Goal",(300,90),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)
```

Goal tracking is done, now let's plot the trajectory of the basketball.

Trajectory is the path taken by the moving object.

To plot trajectory we will begin with creating two arrays:

- Create two empty arrays at the beginning of the code , **xs =[ ]** and **ys= [ ].** The **xs** array will store x points and **ys** array will store y points.

`` import cv2 import time import math p1 = 530 p2 = 300 xs = [] ys = [] ```	
Now we will store the all x, y position of the center of the bounding box in **xs** and **ys** arrays. In the **goal_track()** function, add the **c1** to the **xs** array using **xs.append(c1)** method and **c2** to the **ys** array using **ys.append(c2)** method. Now we will use a **for** loop to traverse through **xs** array and draw a circle at each point. Now run the program to see the output.	

```
def goal_track(img,bbox):
    x,y,w,h = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    c1 = x + int(w/2)
    c2 = y + int(h/2)
    cv2.circle(img,(c1,c2),2,(0,0,255),5)

    cv2.circle(img,(int(p1),int(p2)),2,(0,255,0),3)
    dist = math.sqrt(((c1-p1)**2) + (c2-p2)**2)
    print(dist)

    if(dist<=20):
        cv2.putText(img,"Goal",(300,90),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)

    xs.append(c1)
    ys.append(c2)

    for i in range(len(xs)-1):
        cv2.circle(img,(xs[i],ys[i]),2,(0,0,255),5)
```

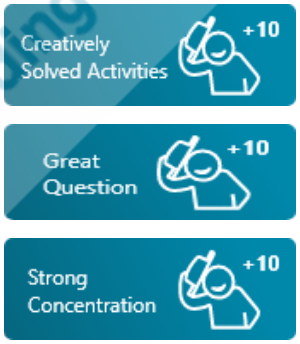
Output:
[Video Reference]

In this class we created a tracker and tracked the object in a video. We have learned to calculate the distance between two points. And at last we plotted the trajectory of the basketball. Isn't it amazing? Great! In the next class we are going to create a program to track hands and gestures.	**ESR:** Yes!
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 05 mins	
Teacher Starts Slideshow **Slide 20 to 24** 	
Activity Details: **Following are the WRAP-UP session deliverables:** • Appreciate the student. • Revise the current class activities. • Discuss the quizzes.	
WRAP-UP QUIZ Click on In-Class Quiz	
Continue WRAP-UP Session  **Slide 25 to 30**	
Activity Details: **Following are the session deliverables:**	

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate the student for his/her efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	Student Action
You get Hats off for your excellent work!	*Make sure you have given at least 2 Hats Off during the class for:* 

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C107-Teacher-Boilerplate
Teacher Activity 2	Reference Code	https://github.com/procodingclass/PRO-C107-Reference-Code
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C107-Student-Boilerplate
Teacher Reference 1	Object Selection	https://drive.google.com/file/d/17gi4r81984hGCQLGSRX967o9Jlz0f5Nv/view?usp=sharing
Teacher Reference 2	Output Ref 1: Track Basketball	https://drive.google.com/file/d/1UL7C0J46CnyvS4BB00Q4V4MNEYhJK-AH/view?usp=sharing
Teacher Reference 3	Distance Formula	https://byjus.com/jee/distance-formula/
Teacher Reference 4	Output Ref 2: Plot Trajectory	https://drive.google.com/file/d/1ZduRqFXyArr-1zaK9cc3hVwzBRlnwueO/view?usp=sharing
Teacher Reference 5	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/914766ab-f26e-4835-90c0-c3f6bd0426ad.pdf
Teacher Reference 6	Project Solution	https://github.com/pro-whitehatjr/proect-solution-C107
Teacher Reference 7	Visual Aid Link	https://s3-whjr-curriculum-uploads.whjr.online/9ac5949c-7796-44e8-8a99-d2e4064cff05.html
Teacher Reference 8	In Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/f3be1cb9-1839-43e5-8423-0416cfc51203.pdf