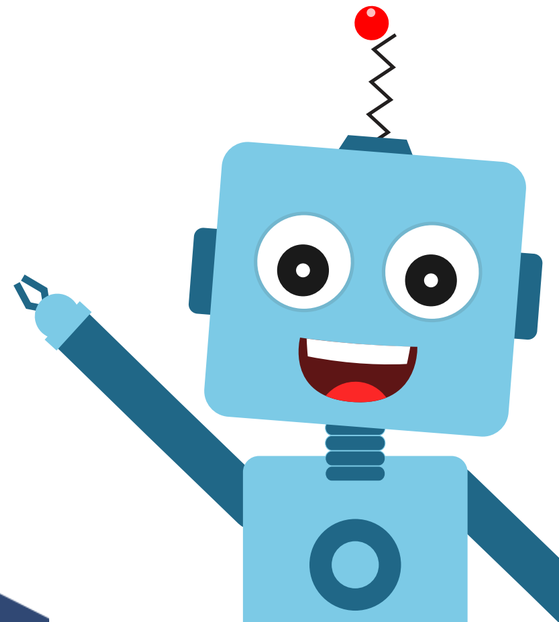
POWERED BY STEMpedia

Computer Vision Object Detection - Part 1

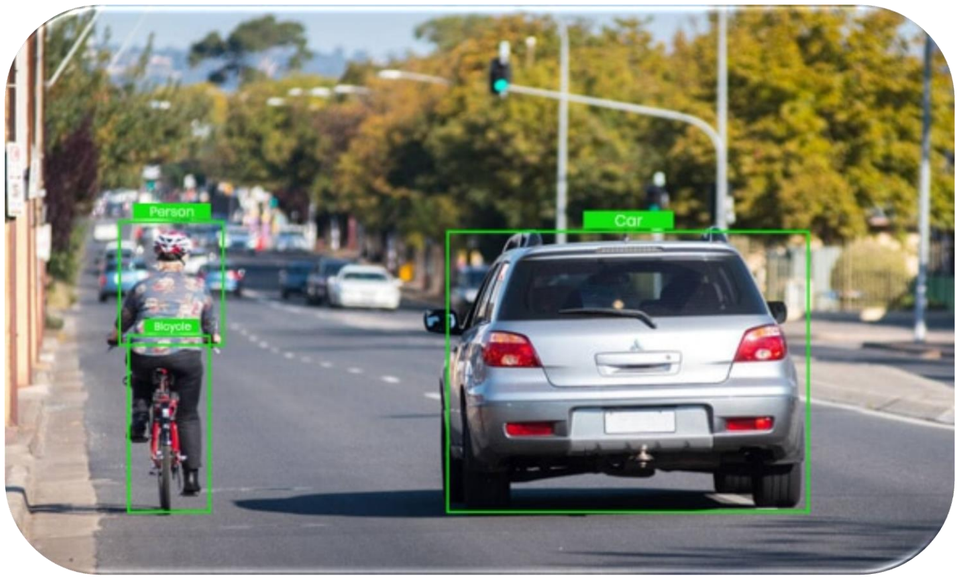
Session 17



Topics covered

POWERED BY STEMpedia

- OPENCV BASIC
- EXAMPLE: SELF-DRIVING CAR USING COMPUTER VISION
- ACTIVITY: OBJECT DETECTION

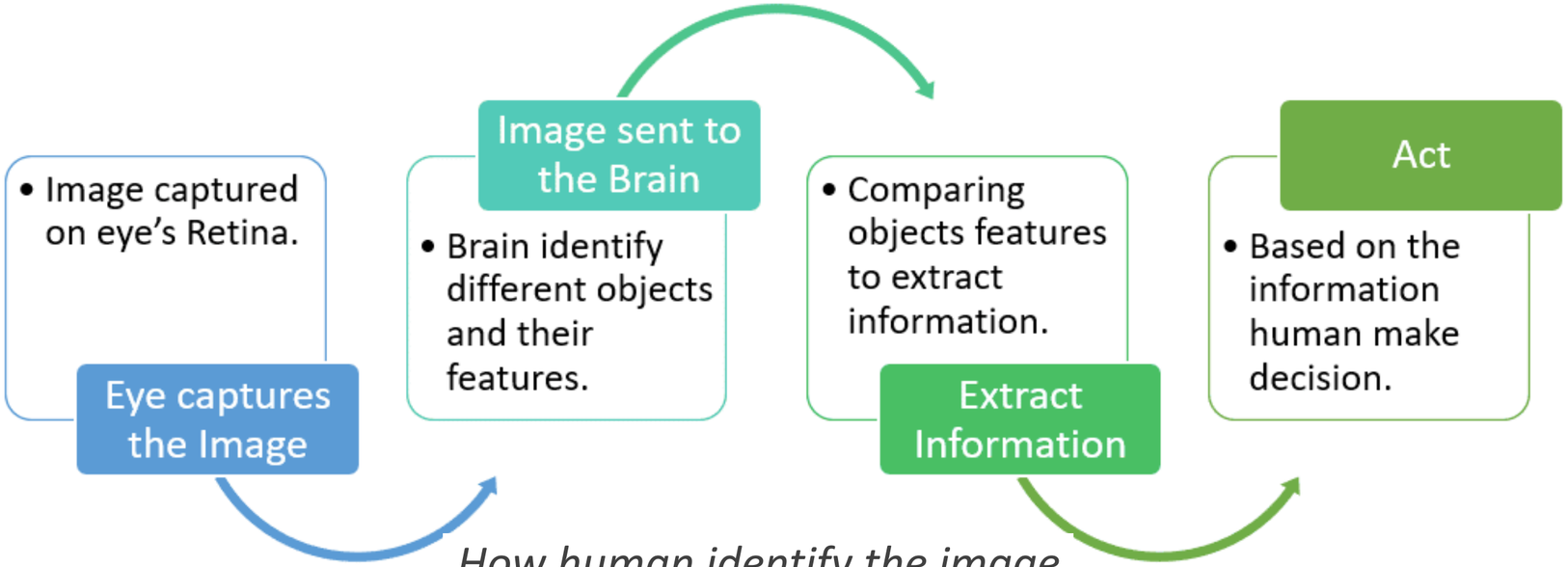


Open Cv

Computer vision

Computer vision deals with how computers can be made to gain a high-level understanding of digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human vision can do.

Computer vision tasks include methods for acquiring, processing, analyzing, and understanding digital images.



How human identify the image

Open Cv Basic

POWERED BY STEMpedia

OpenCV (Open-Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is open-source and free to use. It is written in C++, but there are also Python, Java, and MATLAB bindings.

Some of the basic functionality provided by OpenCV includes:

- Object Detection: OpenCV can be used for object detection in images and videos using methods such as Haar cascades, HOG (**Histogram of Oriented Gradients**), and deep learning.
- Image Processing: OpenCV provides a wide range of image processing functions, including **image filtering, thresholding, morphological operations,** and **edge detection**.
- Face and Body Analysis: OpenCV can be used for face detection, facial landmark detection, **facial expression recognition, body pose estimation,** and **object tracking**.
- Video Analysis: OpenCV provides tools for video analysis, including **video stabilization**, **motion analysis,** and **optical flow estimation**.
- Camera Calibration: OpenCV provides functions for **camera calibration**, which is the process of estimating a camera's intrinsic and extrinsic parameters.

Open Cv Basic

POWERED BY STEMpedia

- Image Segmentation: OpenCV provides functions for **image segmentation**, which is the process of dividing an image into **multiple regions** based on their properties.
- Augmented Reality: OpenCV can be used for augmented reality applications, such as **object detection**, **tracking**, and **overlay**.

OpenCV also provides a GUI (Graphical User Interface) module, which allows the user to display images and draw on them.

It is widely used in various fields, such as robotics, surveillance, medical imaging, and autonomous vehicles.

In simple terms, OpenCV is a library that helps developers create applications that can process and analyze visual data, such as images and videos. It provides a wide range of tools and functions that can be used to perform various computer vision tasks such as image processing, object detection, and video analysis.

EXAMPLE: SELF DRIVING CAR USING COMPUTER VISION

POWERED BY STEMpedia

A self-driving car is a vehicle capable of sensing its environment and moving safely without human input.

For this example, let us consider that a self-driving car can go forward, turn left, right, or stop. Next, let's see how the car would react if a pedestrian approached it.

- **Acquire**: Self-driving cars use cameras to acquire images. They receive and process images at a very high rate. For example, let's consider that our camera has received this image.

Input

Object Detection

- **Process**: The computer identifies all the objects in the image and lists them with their position. In this case, there is something on the road. However, the computer still has no information about what object it is.
- **Analyze**: The computer then classifies each object into different categories. In this case, it identifies the object as a girl. It also tags some information about the object as harmfulness, distance, and other parameters. These tags are the higher-level information used to make a decision.
- **Act**: Based on the higher-level information, the computer can act. In this case, the car will stop.

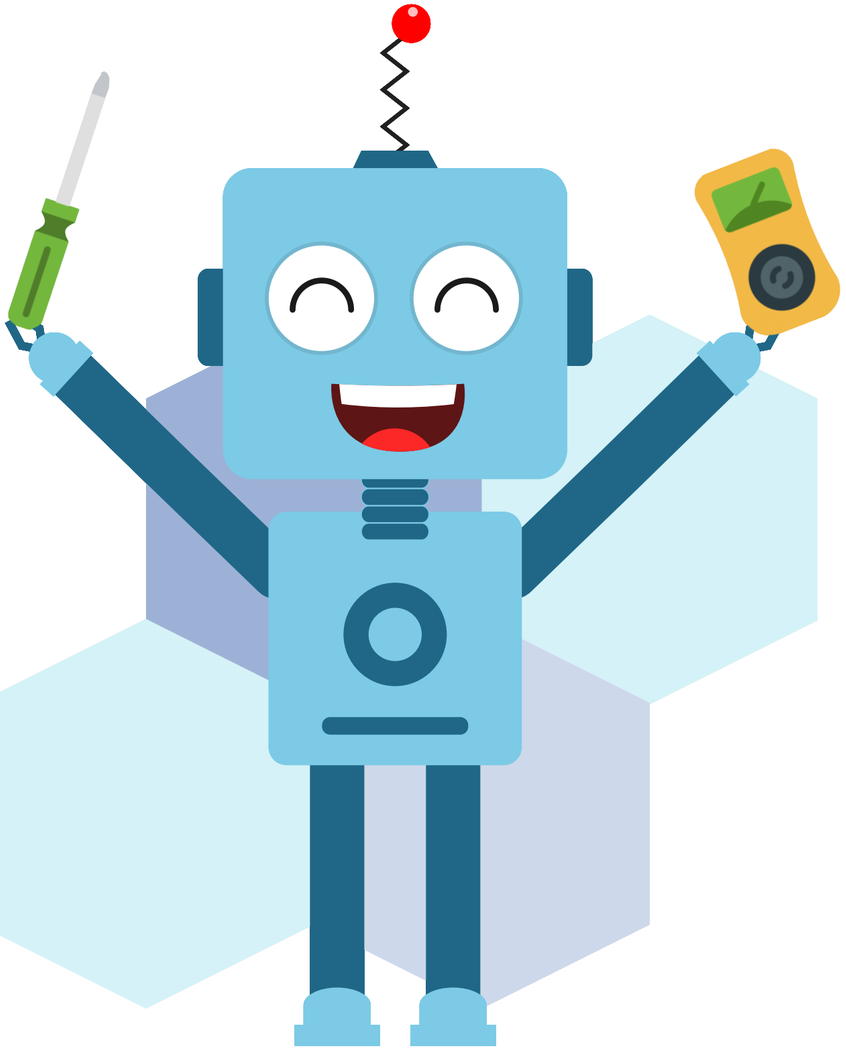
ACTIVITY OBJECT DETECTION

POWERED BY STEMpedia

Let’s detect object from image or videos using PictoBlox

Step – by – step:

- Open a **New file** in PictoBlox: **1.1.** Select the coding environment as **Python Coding**.
- **1.2.** Delete the sprite and add a sprite from the library. Click "**Square Box**" sprite.
- **1.3.** Add **Object Detection** and **Pen** extensions and initialize the objects.
- **2.** Initialize the Object Detection with following: **2.1. setthreshold(0.5):** Set the object detection threshold to with
- **50% confidence.**
- **2.2. analysestage():** for analysing the image on stage.
- **2.3. disablebox():** for disabling boxes that appear on the stage, around objects.



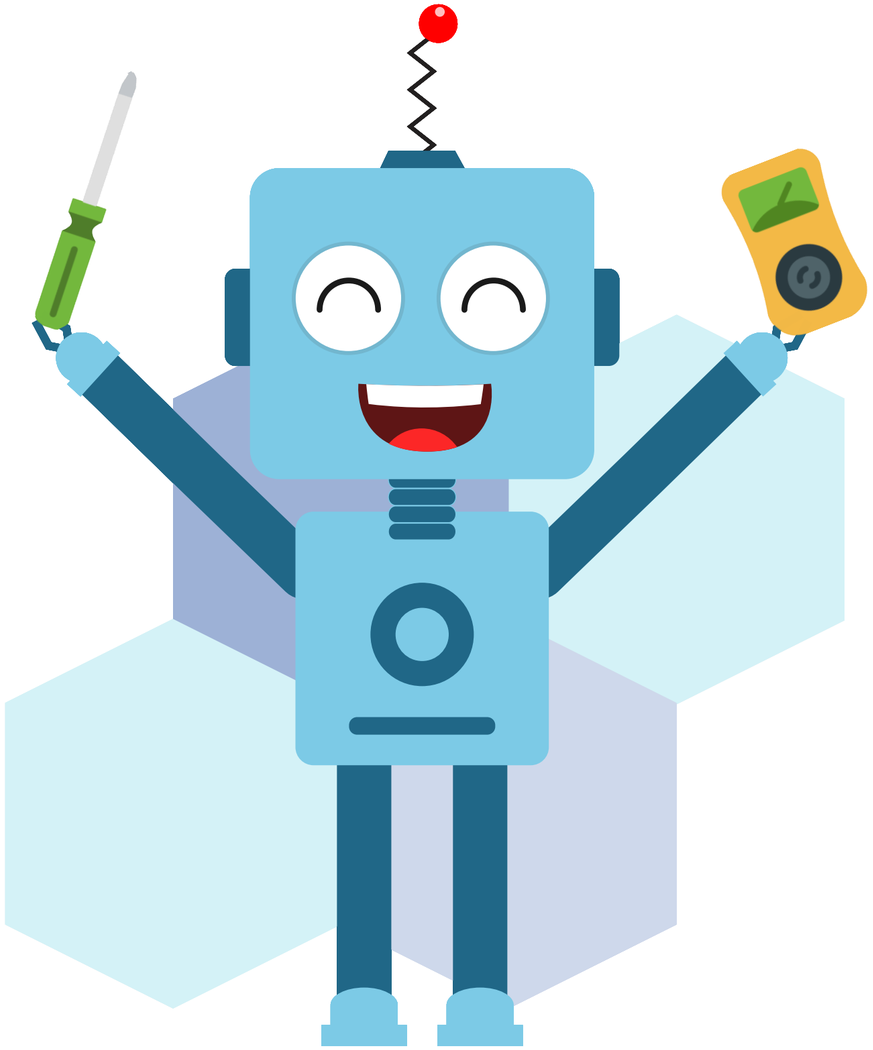
ACTIVITY OBJECT DETECTION

POWERED BY STEMpedia

- Now, we want to detect all the objects in the image, one by one and put the Square Box sprite on them. For this, we will iterate through all the objects using the **for loop.**
- We will iterate through all the objects in the image, one by one, using a for loop, and place the sprite on each of them. To do this, we will employ the following functions: Inbuilt function **range()** to loop through all the objects.
- **count()** function of **Object Detection** class gives us the maximum number of objects detected. We add '+1' since the end parameter of **range()** function is excluded.

```
for i in range(1,obj.count()+1):
```

- Let's set the x-position, y-position and width of the Square Box according to each object. We will use the following functions:
  - **sprite.setx()** – Set the x-position of the sprite (Square Box)
  - **sprite.sety()** – Set the y-position of the sprite (Square Box)
  - **obj.x(i)** – Get the x-position of the object '**i**' (1 to **obj.count**())

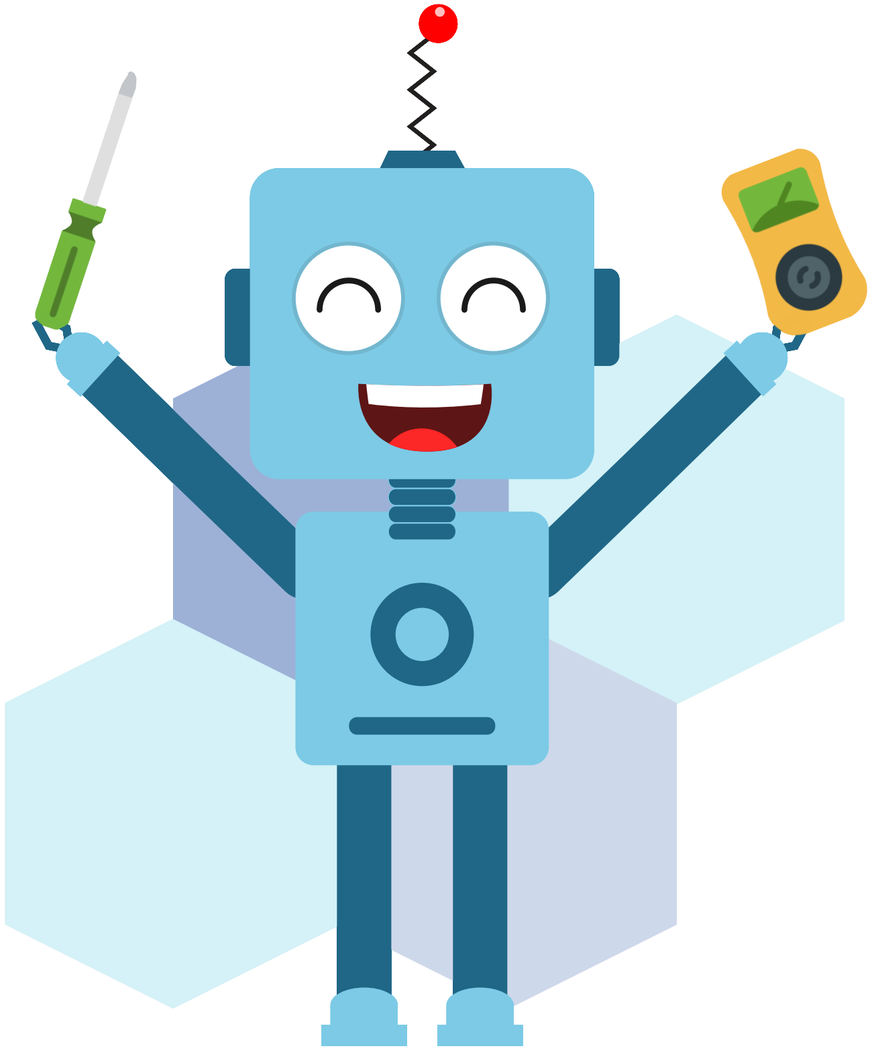


ACTIVITY OBJECT DETECTION

POWERED BY STEMpedia

- **obj.y(i)** – Get the y-position of the object '**i**'
- **sprite.setsize()** – Set the size of the sprite (**Square Box**)
- **obj.width(i)** – Get the width of the object '**i**'

```
sprite.setx(obj.x(i))
sprite.sety(obj.y(i))
sprite.setsize(obj.width(i))
```



- We will make the Square Box say the class of the respective object that it detects using the **sprite.say()** function.
- The Pen class's **pen.stamp()** function will be used to stamp a Square Box on each detected object.
- We will add a **time.sleep()** function to have time between detecting each object.
- Press "Run" button to test the code.

Lets Code

POWERED BY STEMpedia

```
sprite = Sprite('Square Box')
import time
obj = ObjectDetection()
pen = Pen()
pen.clear()
obj.setthreshold(0.5)
obj.analysestage()
obj.disablebox()

for i in range(1,obj.count()+1):
  sprite.setx(obj.x(i))
  sprite.sety(obj.y(i))
  sprite.setsize(obj.width(i))
  sprite.say(obj.classname(i))
  pen.stamp()
  time.sleep(2)
```

Lets Code

Input

Output

SAVING THE PROGRAM

Save the project file as **Object Detection**.

POWERED BY
STEMpedia

www.ai.thestempedia.com