Computer Vision, OpenCV and YOLO

DASE7503 Robotic Systems Integration

Contents

- Introduction to Computer Vision and OpenCV
- OpenCV Real-time Applications in Computer Vision
- OpenCV Architecture
- Introduction to YOLO
- Main Components of the YOLO Algorithm
- ONNX and YOLO
- Integration of OpenCV and YOLO with ROS and Robotics

Introduction to Computer Vision

- **Computer Vision (CV)** is a field of computer science that uses Artificial Intelligence (AI) and Machine Learning (ML) algorithms to process digital images and videos.
- It is a subset of AI that focuses on giving computers the ability to understand and interpret the **visual world**.
- CV algorithms can be used to detect objects, identify faces, recognize patterns, classify images, segment images into meaningful parts, and more.
- CV has been widely used in many applications such as medical imaging, robotics, surveillance, facial recognition, and augmented reality.
- In robotics, CV algorithms are used to recognize objects and navigate around obstacles.

Introduction to OpenCV

- OpenCV (Open-Source Computer Vision Library) is an open-source computer vision library.
- It was originally developed by Intel Corporation in 1999 and later became a collaborative effort of several companies and universities.
- It is written in C++ and supports many programming languages.
- It is an open-source computer vision and machine learning software library, and its main goal is **real-time computer vision applications**.
- It was built to provide a good infrastructure for computer vision applications and to improve the use of machine vision in commercial products

Introduction to OpenCV

- OpenCV library has more than **2500 optimized algorithms** and has a **modular structure**, which means that the package includes several shared or static libraries, and developers can pick and choose which parts they wish to use.
- For example, OpenCV includes numerous features, such as **pattern recognition, motion detection, and object identification**.
- OpenCV can be used in a wide variety of applications and platforms, including **embedded systems, robotics, mobile phones, video surveillance, and computer vision**.
- Its code is **portable**, and it is suitable for real-time applications.
- OpenCV is also available as a library of optimized functions in various programming languages.

Introduction to OpenCV

- OpenCV has become the go-to library for **computer vision** and **machine learning**.
- It is being used in many industries, such as medical, automotive, retail, and robotics, to develop applications that can recognize objects, analyze images, and extract useful data from them.
- This can be used in **3D point clouds** from **stereo cameras**, **stitch images** together to produce a high-resolution image of an entire scene, find similar images from an image database, and much more.
- It is open-source and free to use, so developers can access and modify its source code according to their needs. The library is constantly updated with new algorithms and features, making it a powerful and constantly evolving library.

OpenCV Real-time Applications in Computer Vision

- **Image and video processing**: OpenCV provides a wide range of functions for image and video processing, such as image filtering, segmentation, and feature extraction.
- **Medical imaging**: OpenCV can be used to analyze medical images, such as X-rays and MRI scans, to assist with diagnosis and treatment.
- **Object detection and recognition**: OpenCV provides tools for detecting and recognizing objects in images and videos, including face detection, pedestrian detection, and object recognition.
- **Video surveillance**: OpenCV can be used to develop video surveillance systems, including motion detection and tracking.
- **Augmented reality**: OpenCV can be used to create augmented reality applications that overlay virtual objects onto the real world.
- **Robotics**: OpenCV can be used to develop algorithms for robots to navigate and interact with their environment, including obstacle detection and avoidance.

OpenCV Real-time Applications in Computer Vision

- **Autonomous vehicles**: OpenCV can be used to develop algorithms for autonomous vehicles, such as lane detection and object detection. It is also used to build similar products like Google street view and helps compete in challenges like the Defense Advanced Research Projects Agency (DARPA).
- **Gaming**: OpenCV can be used to create interactive games that use gestures and facial expressions as input.
- **Space**: OpenCV has been used for some of the most high-profile projects like space exploration. It was used in the Curiosity rover mission to Mars to help the rover navigate and avoid obstacles. The rover used a combination of stereo cameras and OpenCV algorithms to create 3D maps of the terrain and identify potential hazards.

OpenCV Architecture

- OpenCV architecture is divided into several components that work together to provide a **comprehensive framework** for image and video processing.
- The main components of the OpenCV architecture are as follows:
  - **Core Functionality:** This is the core of the OpenCV library and includes basic data structures, mathematical functions, and algorithms for image processing. It provides support for basic image operations such as image conversion, filtering, and morphological operations.
  - **High-Level GUI:** This component provides high-level graphical user interface functionality and includes functions for creating windows, displaying images, and handling mouse and keyboard events.

OpenCV Architecture

- **Video I/O:** This component provides support for reading and writing video files in various formats. It also provides functions for capturing video from cameras and other sources.
- **Image Processing:** This component includes a wide range of image processing algorithms as edge detection, image segmentation, object detection, and feature detection. It also provides support for image transformations, such as scaling, rotation, and affine transformations.
- **Feature Detection and Description:** This component provides functions for detecting and describing image features, such as corners, blobs, and edges. It also includes algorithms for matching features between images, which are useful for tasks such as object recognition and tracking.

OpenCV Architecture

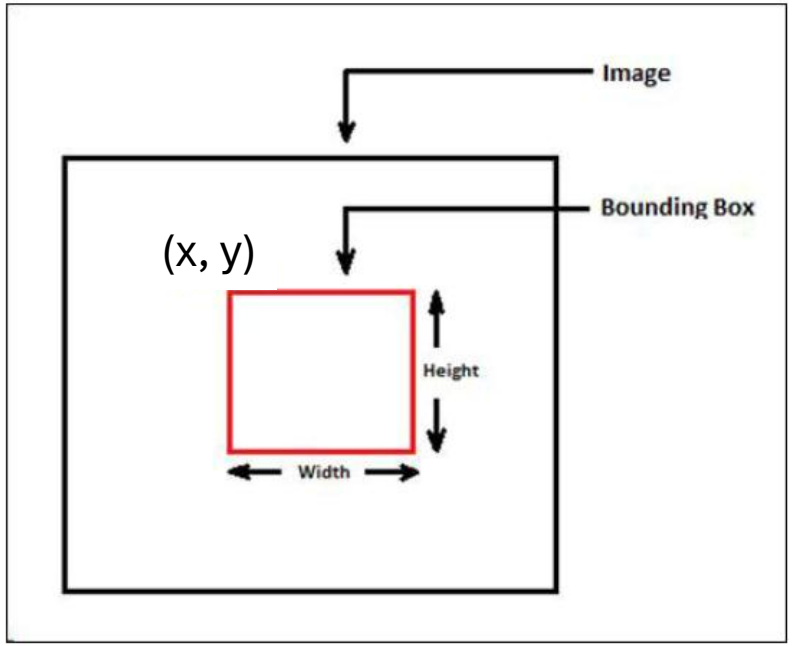
- **Object Detection and Recognition:** This component provides support for object detection and recognition, using techniques such as Haar cascades, HOG features, and deep learning-based methods.
- **Machine Learning:** OpenCV includes several machine learning algorithms, such as decision trees, support vector machines, and neural networks. These algorithms can be used for tasks such as classification, regression, and clustering.
- **Miscellaneous:** This component includes various other functions and algorithms, such as camera calibration, stereo vision, optical flow, and 3D reconstruction.

Introduction to YOLO

- You Only Look Once (YOLO) is a **real-time object detection algorithm** that uses deep learning to recognize objects in images and videos.
- Unlike other object detection algorithms that perform detection in multiple stages, YOLO performs detection in a **single stage**, making it much faster than other methods.
- The YOLO algorithm divides the input image into **a grid of cells** and predicts **the bounding box** and **class probabilities** for each cell. The bounding box coordinates are relative to the coordinates of the cell, which means that a single object can be detected in multiple cells.

Main Components of the YOLO Algorithm

- **Input**: YOLO takes an input image or video and resizes it to a fixed size.
- **Deep learning model:** YOLO uses a deep convolutional neural network (CNN) as its backbone architecture. The network is trained on a large dataset of labeled images to learn how to recognize objects.
- **Bounding box prediction:** The YOLO algorithm predicts the bounding boxes for each object in the input image. Each bounding box is represented by five values: **x, y, w, h,** and **confidence score**. The (x, y) coordinates represent the center of the box, while w and h represent the width and height of the box, respectively. The confidence score represents how confident the algorithm is that there is an object in the box.

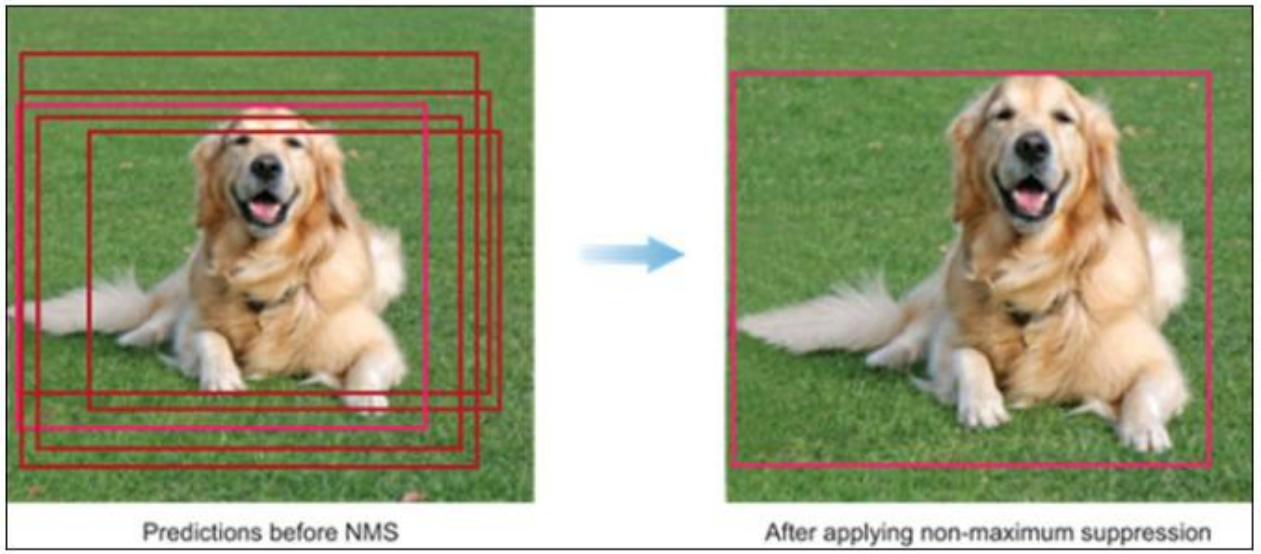


Main Components of the YOLO Algorithm

- **Non-maximum suppression:**
  - Non-maximum suppression (NMS) is a post-processing step used in object detection algorithms to eliminate duplicate detections of the same object.
  - It works by selecting the detection with the highest confidence score and suppressing all other overlapping detections that have a lower score.
  - This is done by defining a threshold for the **IoU** between the bounding boxes of the detections.
  - If the IoU between two bounding boxes is greater than the threshold, the detection with the lower score is suppressed. This process is repeated until all the remaining detections have an IoU below the threshold

Main Components of the YOLO Algorithm

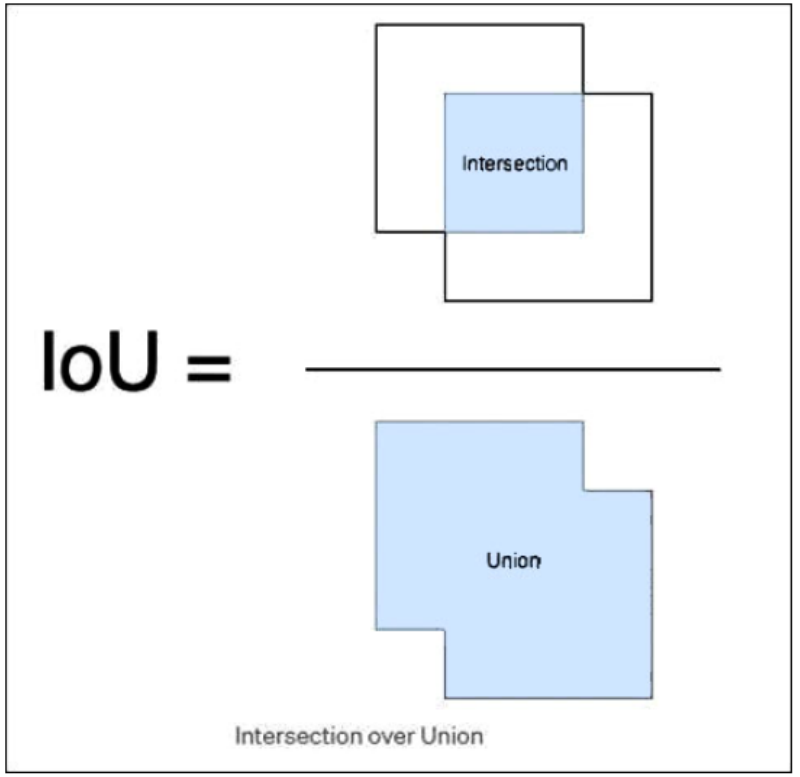
- **Non-maximum suppression:**
  - Since multiple bounding boxes may be predicted for a single object, non-maximum suppression is used to remove the redundant boxes.
  - Non-maximum suppression compares the confidence scores of overlapping boxes and keeps the box with the highest score while suppressing the others.



Non-Maximum Suppression in YOLO

Main Components of the YOLO Algorithm

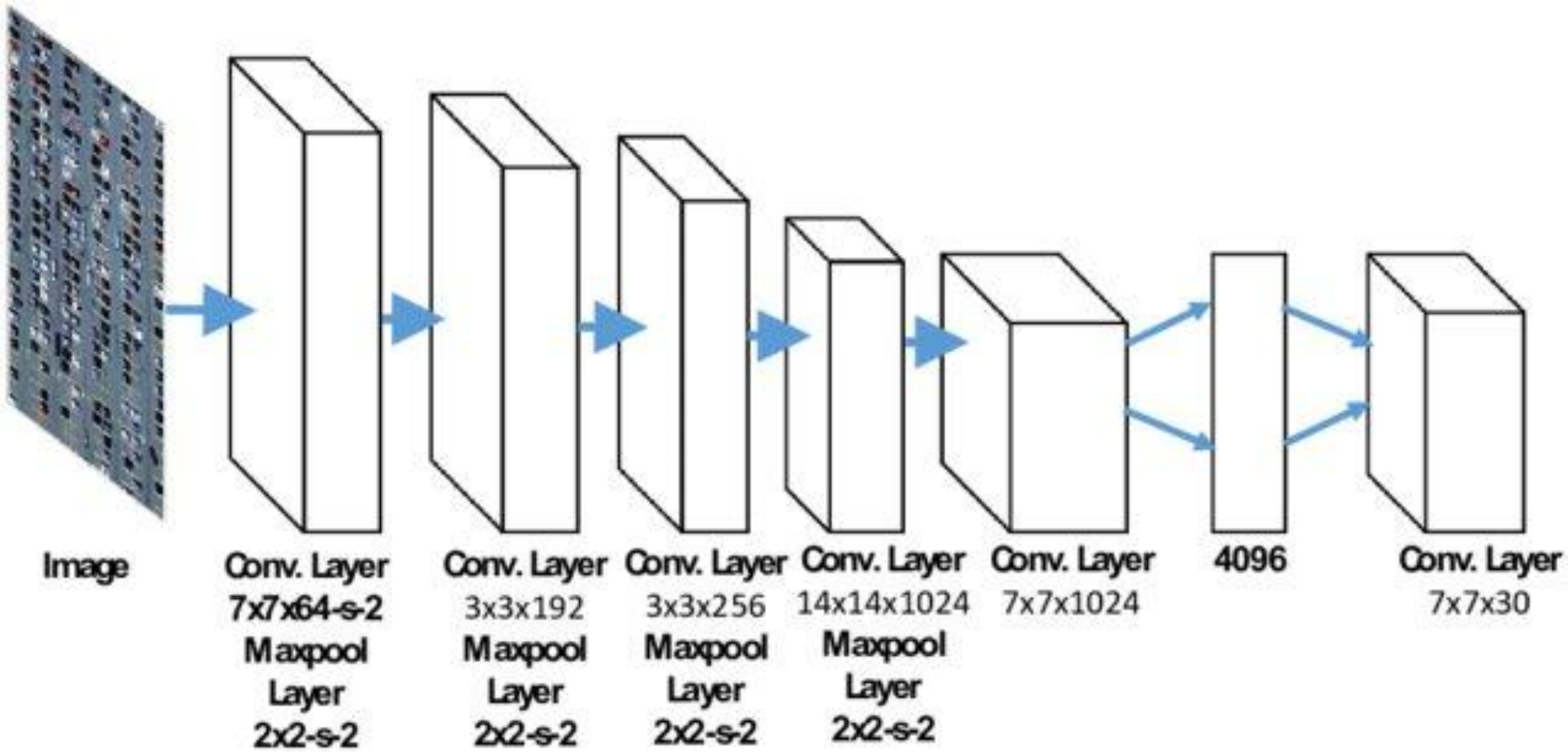
- **Intersection over Union (IoU):**
  - It is a measurement of the overlap between two bounding boxes.
  - It is calculated as the ratio of the intersection area between the two bounding boxes and the union area of the two bounding boxes.
  - In object detection tasks, IoU is used as a metric to evaluate the performance of the model by comparing the predicted bounding boxes with the ground truth bounding boxes.
  - A commonly used threshold for IoU is 0.5, which means that a predicted bounding box is considered a true positive if its IoU with the ground truth bounding box is greater than or equal to 0.5.
- **Output:** The final output of the YOLO algorithm is a list of bounding boxes and their corresponding class probabilities.



Advantages of YOLO over Other Object Detection Algorithms

- **Speed:** YOLO can perform object detection in real-time, making it suitable for applications where speed is important.
- **Accuracy:** YOLO has high accuracy in object detection, especially for large and small objects.
- **Flexibility:** YOLO can detect objects of different sizes, shapes, and orientations, making it suitable for a wide range of applications.
- **Simplicity:** YOLO performs object detection in a single stage, making it easier to implement and modify compared to other methods.

YOLO Architecture

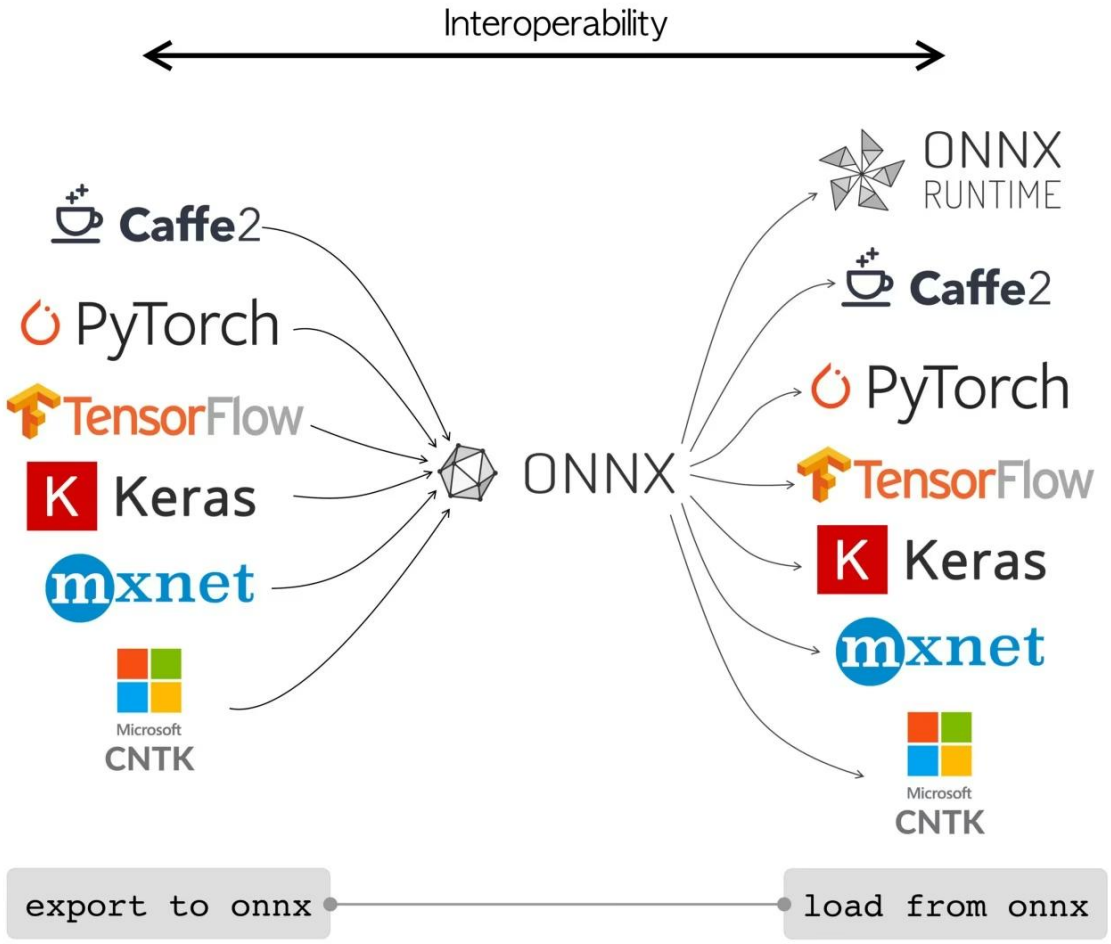


ONNX and YOLO

- **Open Neural Network Exchange (ONNX)** is a community project that Facebook and Microsoft initially developed.
- The ongoing development of ONNX is a collaborative effort supported by various organizations like IBM, Amazon (through AWS), and Google.
- The project aims to create an **open file format** designed to represent **machine learning models** in a way that allows them to be used across different AI frameworks and hardware.

ONNX and YOLO

- ONNX models can be used to transition between different frameworks seamlessly. For instance, a deep learning model trained in PyTorch can be exported to **ONNX format** and then easily imported into TensorFlow.



ONNX and YOLO

- ONNX models can be used with ONNX Runtime.
- **ONNX Runtime** is a versatile cross-platform accelerator for machine learning models that is compatible with frameworks like PyTorch, TensorFlow, TFLite, scikit-learn, etc.
- ONNX Runtime optimizes the execution of ONNX models by leveraging hardware-specific capabilities.
- This optimization allows the models to run efficiently and with high performance on various hardware platforms, including CPUs, GPUs, and specialized accelerators.

ONNX and YOLO

In the context of YOLO, ONNX is commonly used to:

- **Export YOLO Models:**
  Many YOLO implementations (such as YOLOv5, YOLOv7, YOLOv8, and others) allow exporting trained models to the ONNX format. This enables the model to be used across various platforms and tools that support ONNX.
- **Framework Interoperability:**
  By converting YOLO models to ONNX, users can deploy them in different environments—such as PyTorch, TensorFlow, OpenVINO, or other inference engines—without needing to retrain or modify the original model.
- **Optimization and Deployment:**
  ONNX models can be optimized using tools like ONNX Runtime or other acceleration libraries, resulting in faster inference times, reduced latency, and lower resource consumption, which is especially beneficial for deployment on edge devices or in production systems.
- **Conversion Workflow:**
  Typically, the workflow involves training the YOLO model in its original framework (e.g., PyTorch), exporting it to ONNX format, and then converting or deploying it with an inference engine that supports ONNX.

ONNX and YOLO

- Example Workflow
  - Train a YOLO model in PyTorch (e.g., YOLOv8).
  - Export the trained model to ONNX using provided export scripts (model.export() in YOLOv8).
  - Use ONNX Runtime or other compatible tools to run inference on new images or videos.
- ONNX serves as an intermediate, standardized format that simplifies deploying YOLO models across different platforms, enables performance optimizations, and promotes interoperability within the AI ecosystem.

Integration of OpenCV and YOLO with ROS and Robotics

- OpenCV, YOLO, and ROS can be used in conjunction with robotics to provide computer vision capabilities to robots. This integration can be used for a variety of tasks, such as **object recognition**, **navigation**, and **obstacle avoidance**.
- **Object recognition:**
  - One of the primary applications of computer vision in robotics is object recognition.
  - Robots need to be able to recognize objects in their environment so that they can interact with them appropriately.
  - OpenCV provides a number of algorithms for object recognition, including feature detection and matching, Haar cascades, and **deep learning-based approaches such as YOLO** and Single Shot MultiBox Detector (SSD).

Integration of OpenCV and YOLO with ROS and Robotics

- **Navigation:**
  - Navigation is another important application of computer vision in robotics.
  - Robots need to be able to navigate through their environment to reach their destination or complete a task.
  - OpenCV can be used to provide visual feedback to the robot, which can help it navigate through complex environments.
  - This can be achieved using techniques such as optical flow, stereo vision, and **SLAM** (**Simultaneous Localization and Mapping**).
- **Obstacle avoidance:**
  - Obstacle avoidance is another important application of computer vision in robotics.
  - Robots need to be able to detect and avoid obstacles in their environment to avoid collisions and navigate safely.
  - OpenCV can be used to provide obstacle detection and tracking capabilities to the robot.
  - This can be achieved using techniques such as **background subtraction**, **edge detection**, and **contour detection**.

Integration of OpenCV and YOLO with ROS and Robotics

- **Robot control:**
  - Once the robot has detected an object, navigated to it, and avoided any obstacles in its path, it needs to be able to interact with the object.
  - OpenCV can be used to provide visual feedback to the robot to help it manipulate objects.
  - This can be achieved using techniques such as **object tracking** and **motion detection**.
- **Integration with robot operating systems (ROS):**
  - Robot operating system (ROS) is an open-source framework for building robot applications.
  - OpenCV can be integrated with ROS to provide computer vision capabilities to robots.
  - This integration allows robots to receive visual feedback from their environment and make decisions based on that feedback.