Study all these

Motion planning algorithm

1. Sensing: The robot gathers information about its environment using sensors, such as LiDAR, cameras, or proximity sensors.
2. Mapping: Based on the sensor data, the robot creates a map of the environment, identifying obstacles, boundaries, and other relevant features.
3. Path generation: The algorithm generates a path from the current position of the robot to the desired goal location. This may involve searching for a feasible path through the map while avoiding obstacles and considering constraints such as robot dynamics and workspace limitations.
4. Trajectory planning: Once a path is determined, the trajectory planning phase refines the path by considering factors such as robot dynamics, velocity profiles, and collision avoidance. This ensures smooth and efficient movement along the planned path.
5. Execution: The robot executes the planned trajectory, continuously monitoring its surroundings and making adjustments as necessary to adapt to changing conditions

Reactive sensing

Various sensory systems can be utilised. The simplest is a whisker sensor that triggers a switch upon encountering an obstacle. For avoiding any contact, a proximity sensor like reflected IR could be used. Range sensors such as sonar or laser sensors enable detection of obstacles over longer distances by sending out a signal and measuring the time for the reflection to return. However, they may encounter errors if the reflection is not clear.

Pre-planned routes (SLAM)

Robot to have a clearer understanding of its surroundings and current position. This typically involves creating a map of open space and finding a connected path through it to reach the goal. One example is using simultaneous localisation and mapping (SLAM).

Proportional-integral-derivative (PID) control

PID control is a classic feedback control technique that adjusts the output of a system based on the proportional, integral, and derivative terms of the error signal. It is effective for linear systems and offers simplicity and ease of implementation but may struggle with nonlinear or complex systems. PID control is more suitable for well-defined systems with known dynamics.

Fuzzy logic control

Fuzzy logic control uses linguistic variables and fuzzy rules to capture human-like decision-making processes. It can handle nonlinearities and uncertainties more effectively but may require more computational resources and tuning effort. Fuzzy logic control is more adaptable to uncertain and complex environments.

Inertial navigation systems (INS)

INS relies on accelerometers and gyroscopes to track position, velocity, and orientation without requiring external signals. This makes it highly effective in environments where GPS signals are weak or unavailable, such as underground, underwater, or indoors. However, INS suffers from drift over time, requiring periodic correction. INS is preferable in environments where continuous tracking without external signals is needed.

GPS-based navigation

GPS-based navigation provides accurate global positioning by receiving satellite signals, making it ideal for outdoor applications like autonomous vehicles and drones. GPS is limited by signal loss in tunnels, dense urban areas, or heavily forested regions. GPS is more suitable for large-scale outdoor navigation.

Robot Operating System

The *Robot Operating System* (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. ROS provides a broad collection of integrations open-source libraries that implement useful robot functionality, with a focus on mobility, manipulation, and perception. ROS also provides an extensive set of tools for configuring, starting, debugging, simulating, testing, and stopping robot systems.

The **ROS Master** provides name registration and lookup to the rest of the nodes. Nodes would not be able to find each other, exchange messages, or invoke services Without the ROS Master. Nodes are processes that perform computation. A **ROS node** is written with the use of a ROS client library, such as roscpp or rospy.

Publish-subscribe model of ROS

ROS topic operates on a data push mechanism. A node that produces data (a publisher) publishes messages to a specific topic, and any interested nodes (subscribers) receive those messages.

- The publisher pushes data to the topic, and subscribers pull the data when they are ready to process it.
- This decoupled communication paradigm enhances modularity and flexibility in robotic systems, allowing nodes to communicate asynchronously without requiring direct connections or knowledge of each other's existence.
- The data push mechanism enables real-time and efficient information exchange in ROS, supporting the development of modular and distributed robotic applications.

Modular design of Robot Operating System (ROS)

- The Robot Operating System (ROS) follows a distributed architecture based on nodes, topics, services, and actions, enabling modular and scalable robot development.
- Nodes are individual processes that perform specific tasks, such as sensor data processing or motor control.
- Topics facilitate communication between nodes using a publish-subscribe messaging system, where nodes can publish messages to a topic while others subscribe to receive them.
- Services enable synchronous communication between nodes through a request-response mechanism, allowing direct interactions when needed.
- Parameter server stores configuration parameters accessible to all nodes, enhancing reusability and system tuning.
- The ROS Master plays a crucial role in managing node registration and communication, ensuring connectivity across the system.
- By following this modular design, ROS promotes reusability, flexibility, and interoperability, making it a standard framework for modern robotics research and development.

Gazebo in ROS

- Gazebo provides realistic physics-based simulation, incorporating factors such as gravity, friction, and collisions to ensure accurate robot behavior.
- Gazebo supports sensor emulation, enabling the simulation of cameras, LiDAR, IMUs, and other sensors essential for perception algorithm development.
- Gazebo allows users to customise simulation environments to match real-world applications, making it a valuable tool for prototyping. By facilitating rapid prototyping, it helps reduce costs and risks by allowing developers to test algorithms without the need for physical robots.
- Gazebo supports multi-robot testing, making it ideal for evaluating coordination strategies and swarm robotics applications. This allows researchers and developers to simulate complex interactions between multiple robots before deployment.

- By using Gazebo, developers can iteratively refine robotic systems, optimise parameters, and troubleshoot issues in a safe and controlled environment before real-world deployment.

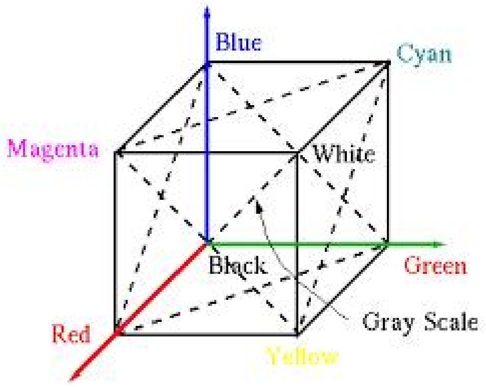
ROS package

- A ROS package is a collection of ROS nodes, configuration files, and other resources that together provide a specific functionality.
- A ROS package is structured as a directory that contains a manifest file, which describes the package dependencies and other metadata, as well as directories for nodes, configuration files, and other resources.

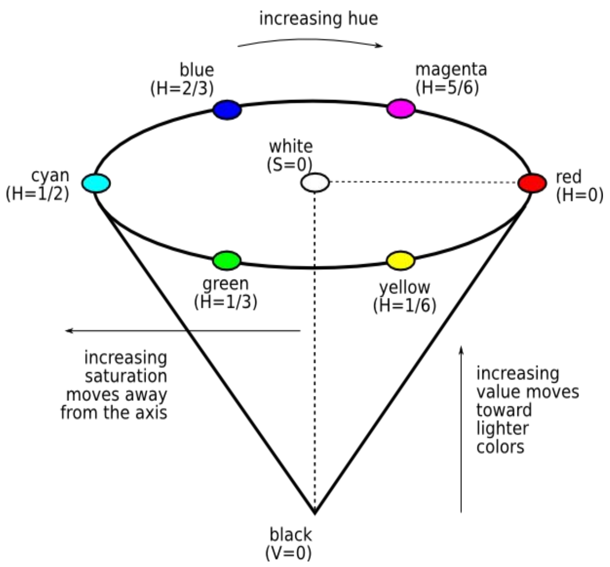
Application of Computer Vision in Robotics

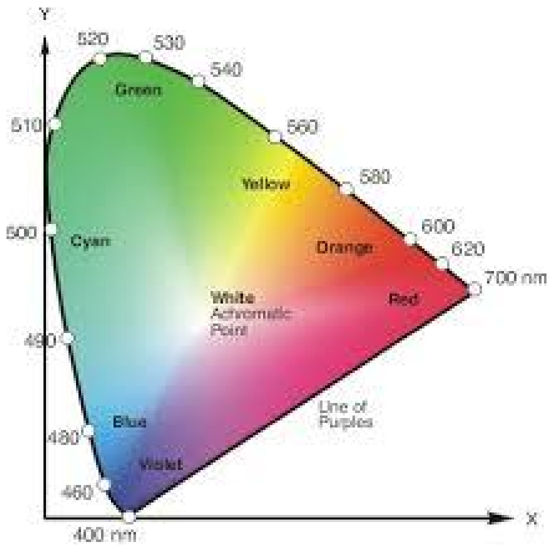
In a robot soccer competition, two cameras are mounted over the field. Each camera captures half of the field so that the whole field can be captured.

The first process to extract the information from the camera is data pre-processing. To pre-process the data, the continuous video is first sampled into images. The RGB image is then transformed to HSV or YUV which is a more perceptually uniform colour space. Color normalization is used to handle non-uniform lighting across the field. Adaptive colour histogram is used to remove the background, in this case the green field colour.



RGB colour model





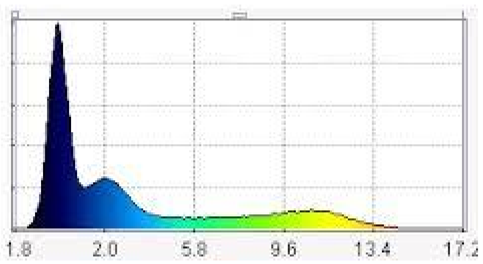
HSV colour model

YUV colour model

Histogram
- ● Number of bars 128
- ○ Bar resolution 1.0

Ignore Background ☐

Background level 0.0



Background removal

After the data is pre-processed, image segmentation is perform to extract useful information. To segment the image, each image pixel is classified into different classes of object (robots and ball) using thresholding method. The threshold image is dilated to fill holes which might appear inside a region. The relevant information such as the colour, area, and bounding box are computed from the segmented region. This information is stored in a data structure for recognition to be done.

After image segmentation, the objects in the image must be classified. In the classification process, the set of attributes assigned in the rules to the objects of interest is recognized. For example, the ball is an orange spherical object and smaller in size, thus it can be detected as a small orange circular region on the image plane. Robots are detected as red or blue regions (the team colors). The goals are detected as large yellow objects on the image. Thus, the objects can be classified.

One of the possible challenges of implementing computer vision is the narrow field of view achieved if a single camera is used. Although special lenses can be mounted to the camera to broaden the field of view, they cause image deformation. Using two cameras can improve the field of view. However, this method causes higher image processing complexity as well as doubling required hardware. Another challenge is real-time reaction has to be achieved. Higher image resolutions result in better precision to image segmentation but also increase the time to process each image since the bigger the image the longer it takes to be analysed.