**Camera Sensor simulation codes**

<sensor name="camera" type="camera">

*//This defines a sensor with the name "camera" and specifies that it is of type "camera."*

<camera name="camera">

*//Within the camera sensor, this specifies the name "camera" for the camera component.*

<horizontal\_fov>1.3962634</horizontal\_fov>

*//This sets the horizontal field of view (FOV) of the camera sensor. The FOV determines the width of the camera's view in radians.*

<image>

*//This section defines properties related to the camera image.*

<width>640</width>

*//This sets the width of the camera image in pixels. In this case, it's 640 pixels.*

<height>480</height>

*//This sets the height of the camera image in pixels. In this case, it's 480 pixels.*

</image>

</camera>

<always\_on>true</always\_on>

*//This specifies that the sensor is always enabled and active.*

<update\_rate>30</update\_rate>

*//This sets the update rate of the camera sensor to 30 Hz. It means that the camera captures and publishes images at a rate of 30 frames per second.*

<visualize>true</visualize>

*//This specifies that the sensor should be visualized in the Gazebo GUI when the simulation is running.*

</sensor>

**Matlab script for blue object detection and pose estimation**

% Connect to ROS

rosinit('http://localhost:11311'); % Assuming ROS master is running locally

% Subscribe to the camera image topic

imageSub = rossubscriber('/camera/image', 'sensor\_msgs/Image');

while true

imgMsg = receive(imageSub); % Receive the image message

image = readImage(imgMsg); % Convert ROS image to MATLAB image

% Convert the image to HSV color space

hsvImage = rgb2hsv(image);

% Define the blue color range in HSV

lowerBlue = [0.55, 0.5, 0.3];

upperBlue = [0.75, 1.0, 1.0];

% Create a binary mask for the blue region

mask = (hsvImage(:,:,1) >= lowerBlue(1) & hsvImage(:,:,1) <= upperBlue(1)) & ...

(hsvImage(:,:,2) >= lowerBlue(2) & hsvImage(:,:,2) <= upperBlue(2)) & ...

(hsvImage(:,:,3) >= lowerBlue(3) & hsvImage(:,:,3) <= upperBlue(3));

% Perform object detection and pose estimation based on the mask

% You can use computer vision techniques like blob analysis or contour detection

% to find the position and size of the blue object.

% Implement your object detection and pose estimation algorithm here

% Publish the estimated pose or take other actions as needed

end

% Shutdown ROS when done

rosshutdown;