Goal

Perform real time tracking of an object by controlling a robot using android smartphone which has limited processing power and accessible camera.

Description

The object we chose is a red colored paper. The object is distinguished by its color.

The smartphone is physically placed on top of the robot.

The robot is controlled by the phone , which communicates with the IOIO board through bluetooth.

Task stages

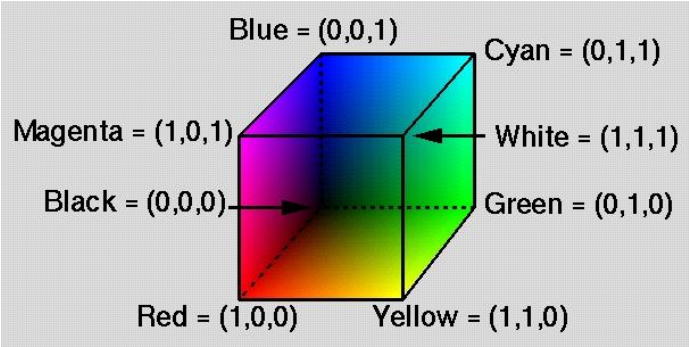
1. Setting up and getting to know android environment:
   Getting familiar with java programming language which is used in android development downloading and installing both the android SDK and eclipse.
2. Getting to know OpenCV library and integrating it with our working environment:
   OpenCV (**Open** Source **C**omputer **V**ision) is a library of programming functions for real time computer vision. (Reference: http://opencv.willowgarage.com/wiki/). Exploring this library and its various functions was essential for our task.
3. Development and implementation of a tracking algorithm:
   The detection of our target was done with color segmentation (red). We had to detect the object (color), determine its distance and angle.
4. Connecting the IOIO board with robot, and learning more about our robot:
   Physically connecting between the robot motors and the IOIO board pins. Understating how the robot is controlled.
5. Communicating with the board (from android) and controlling the robot:
   Communication between the IOIO board and the phone was done via Bluetooth. Setting up the IOIO libraries and integrating them to our project. Establish a connection and move the robot in all directions. Setting the robot's speed.
6. Integrating the tracking algorithm with the robot control and movement:
   moving the robot according to our algorithm and testing it.

Tools and resources:

- Android smartphone.
- IOIO board.
- Robot Rover5.

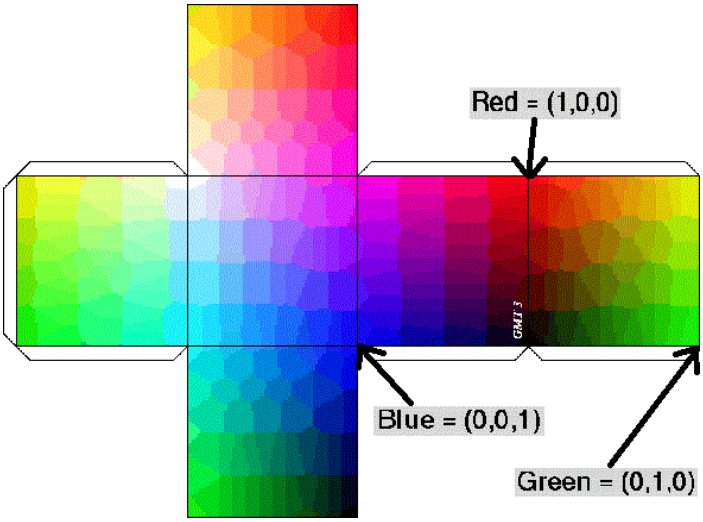
Color tracking algorithm

- Grab a frame from the Smartphone camera.
  The camera returns the frame in format of **RGB**.

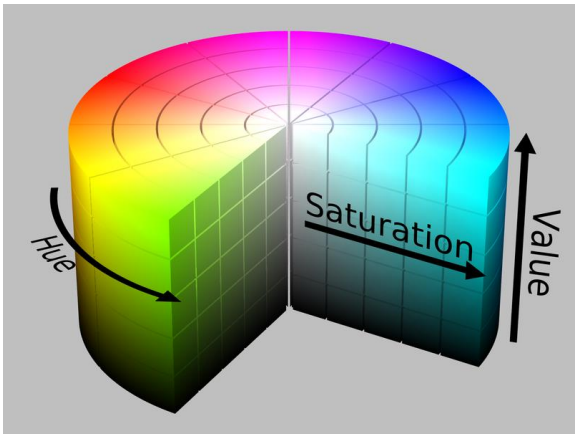


- Color detection

  Problem: defining a range for certain hue in RGB space is difficult. Since it can be represented in RGB space by various combinations.



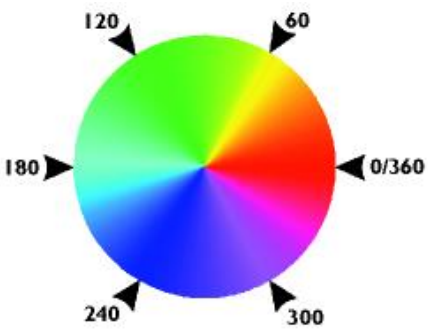
  Solution: Convert the frame from RGB space to HSV color space.



In this HSV (**H**ue, **S**aturation, **V**alue) color space the color's hue is represented by one single coordinate, making the identification of a certain hue easier and more precise than in RGB color space.

- Defining the desired color
  the desired color is red, so every pixel that is in the following range is considered as red:

$$Hue \in [-15°, +15°]$$

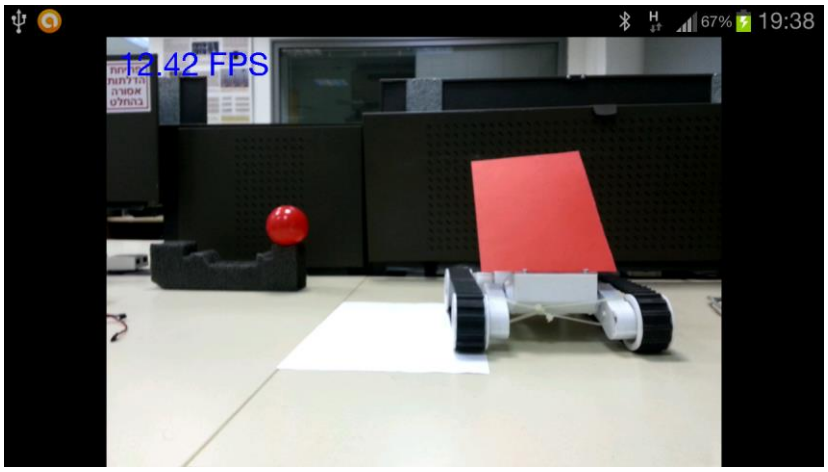


$$Saturation \in [50\%, 100\%]$$

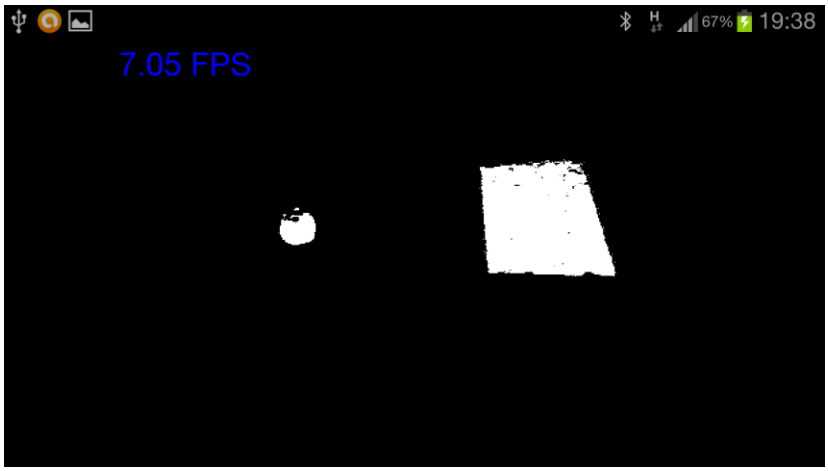
$$Value \in [50\%, 100\%]$$

The saturation and value coordinates set to 50% and higher (in order to minimize the light effect):

- <u>Color Segmentation</u>
  the pixels within the above range are set to white, while the others are blackened.
  an example:
  Original frame



Binary frame

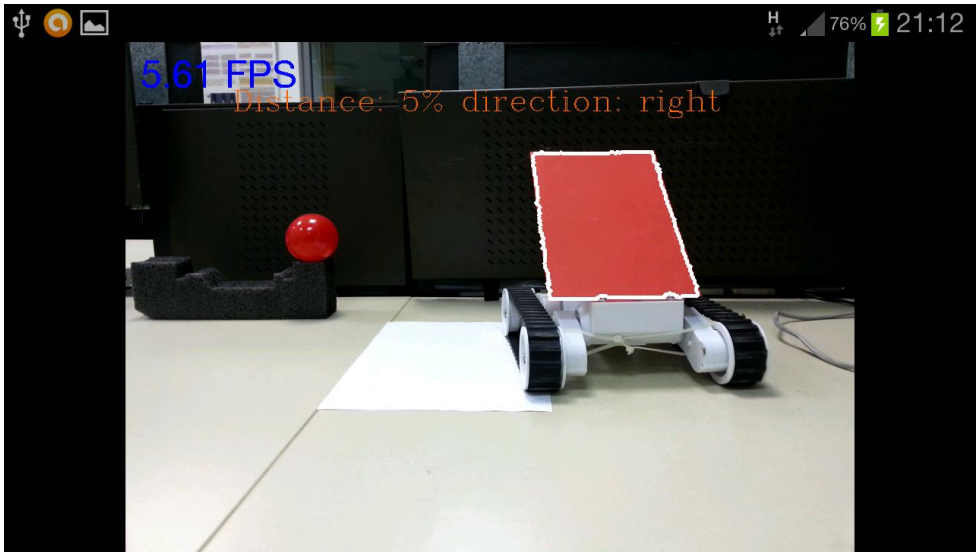


- Object detection

  in order to detect the target object we apply OpenCV's implementation of contour extraction algorithm based on (Suzuki 85) to the binary image. This generates a collection of external contours surrounding the white areas in the binary image. We assume that our target object is the biggest contour and all others are simply a noise (standard laboratory conditions – no big red objects are present besides the target).

  Reference: S.Suzuki, K.Abe. (1985) Topological structural analysis of digital binary image by border following.

Example of filtering other red objects:



only the bigger object is detected.

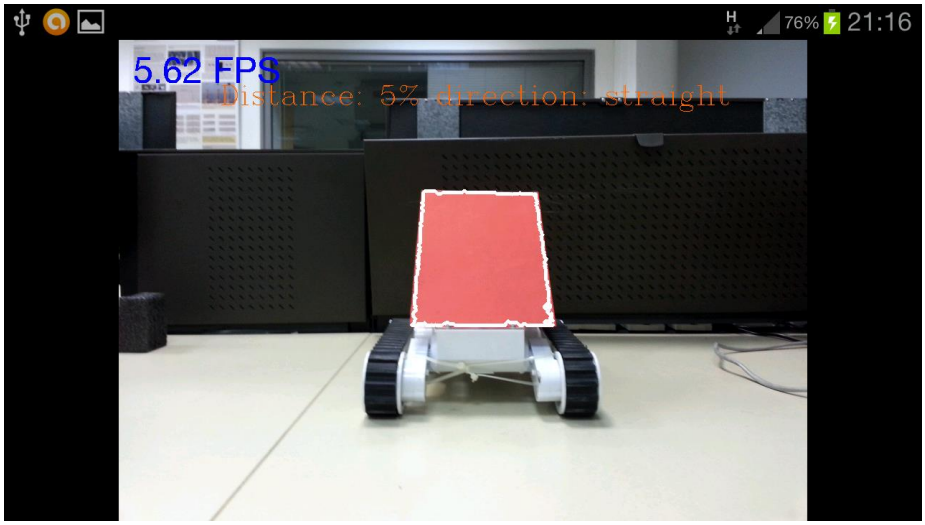
- Find the target distance

  the object's distance is determined based on the object's area (number of the object's pixels).

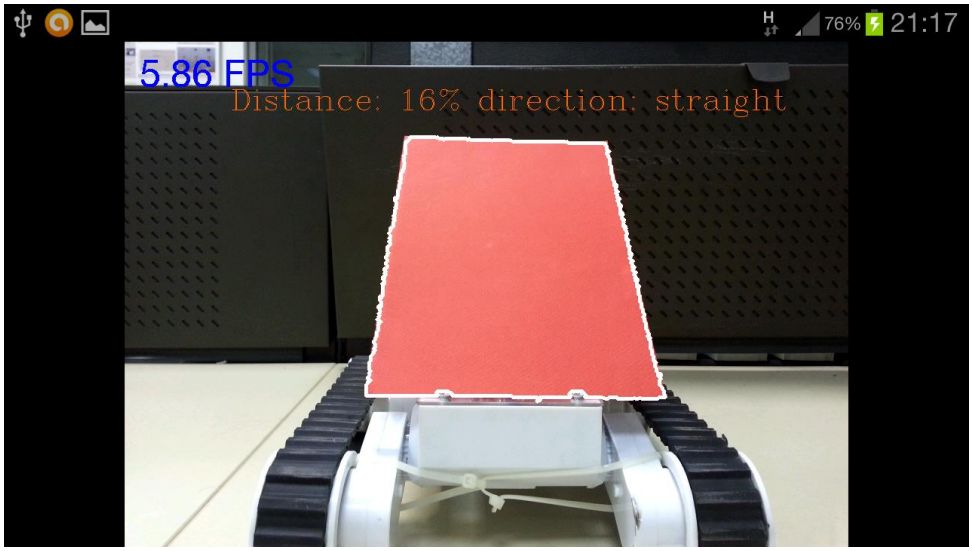
  In order to determine whether the object is far or close we inspect the percentage of the object's pixels from the total frame pixels.

When the distance is greater than 12% the object is considered far and the robot should move closer to the object (forward).
When the distance is less than 9% the object is considered close and the robot should move backward.
When the distance is between 9%-12% the robot is idle and doesn't move.
Example of distance detection:

Far object:



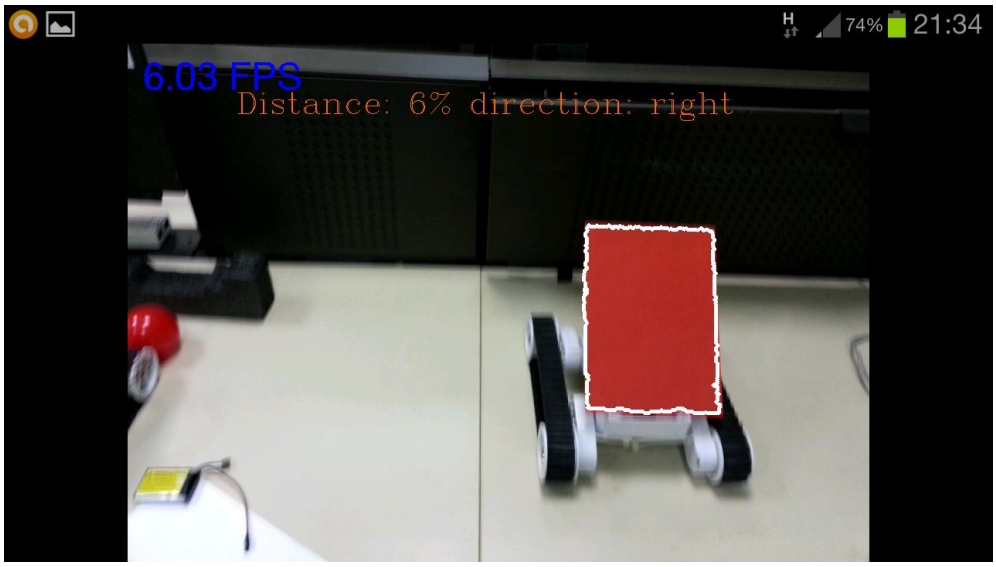
Close object:



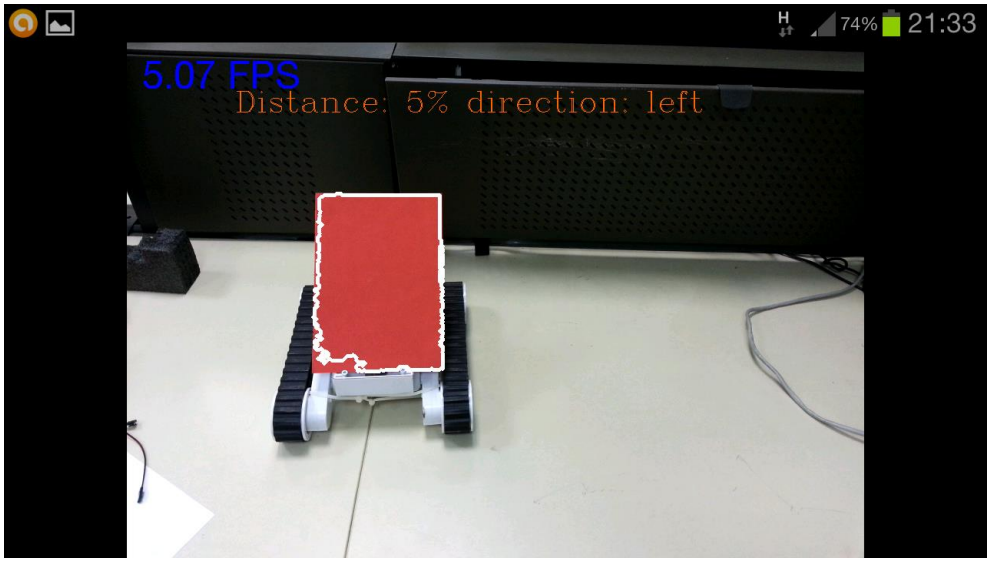
- <u>Find the target direction</u>
  the direction of the object is determined by calculating the contour pixel's average coordinate X relative to the whole frame width.
  If the average coordinate x is in the left third of the frame, the robot should move left.
  If the average coordinate x is in the right third of the frame width, the robot should move right.

If the average coordinate x is in the middle third of the frame, the robot should move straight.
Example of different directions:

Right direction



Left direction



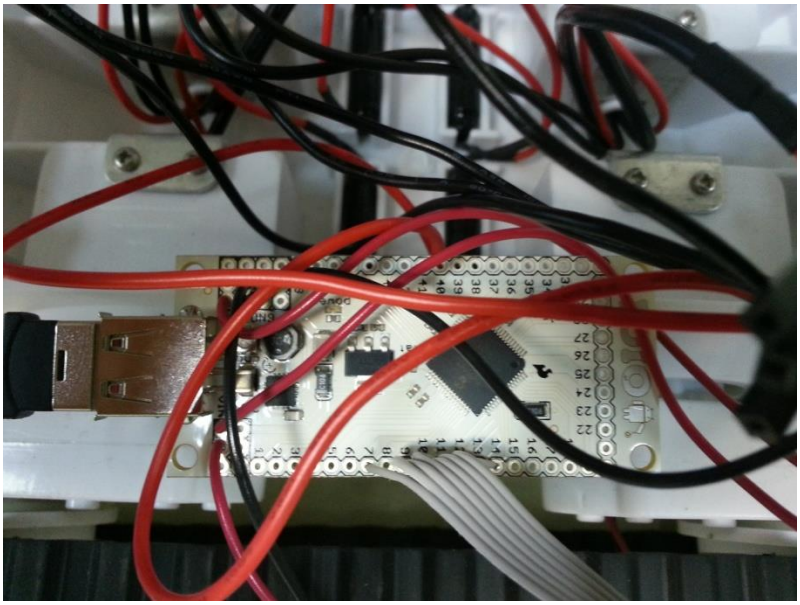
Controlling the robot

The robot have two motors* one to each side .moving the robot in certain direction is done by moving certain motor backward or forward accordingly .

The microcontroller which is the IOIO board is the one responsible to supply the voltage to move the motors and supply the logic to determine its direction, this is done using 6 pins:

- 2 pins are PWM (pulse width modulation) signals, each one is responsible for moving one side of the robots motors in certain speed determined by the pulse width.
- 4 pins are digital signals, which is responsible for the logic to determine the robot direction { forward , backward , Clockwise , Counter clockwise }

The microcontroller:

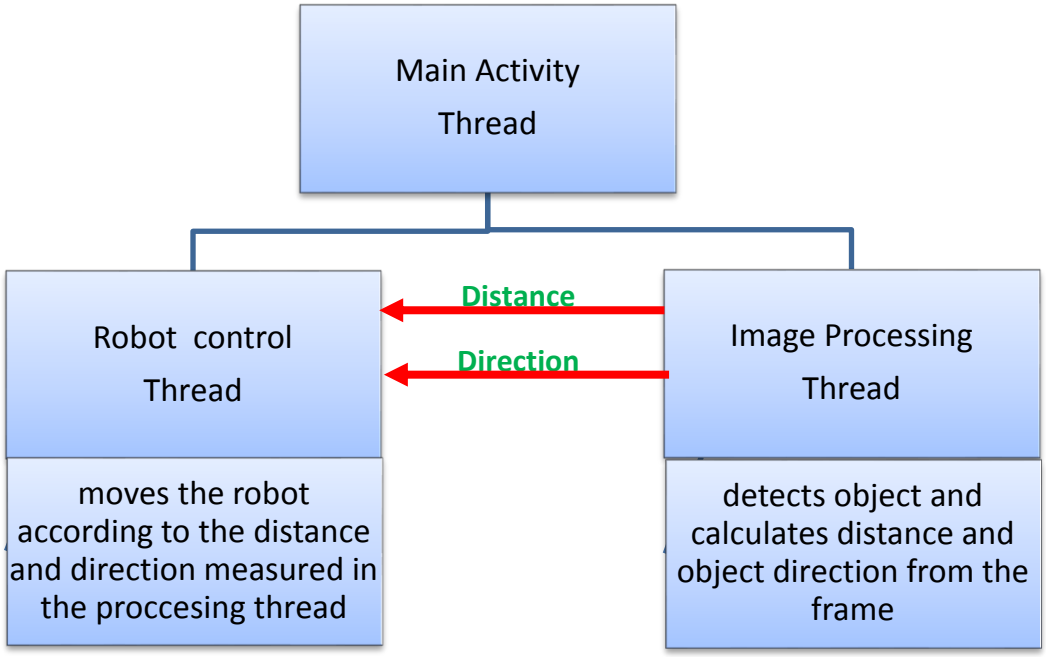


*The USB connected to the BT dongle, the white cables are the above 6 pins used.

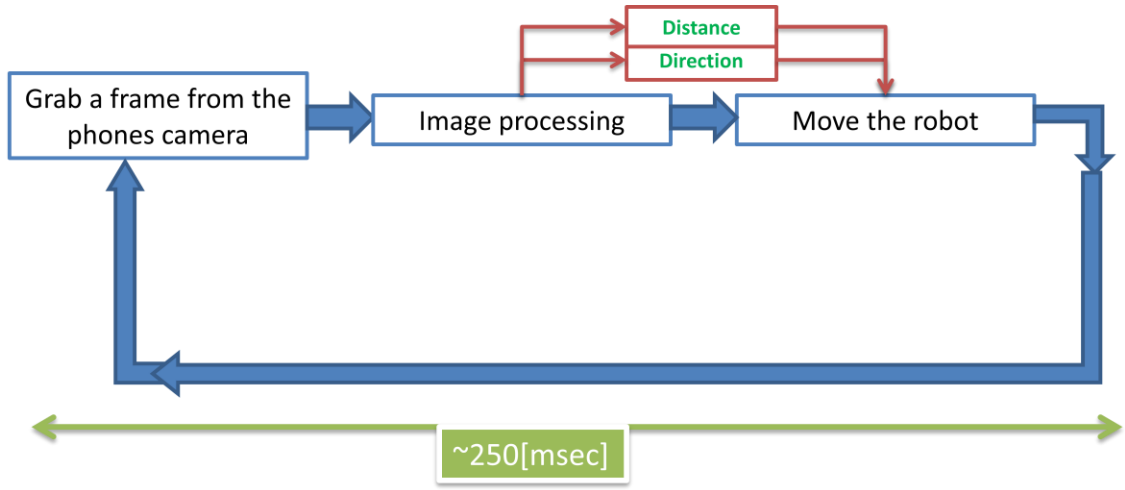
Running Threads:

The flow is comprised of three threads:

1. Image processing thread: this thread is responsible for the processing described in the previous sections. (calculates distance and direction form the object)
2. Robot controlling thread: this thread communicates with robot. This thread wakes every 250 msec, and moves the robot according to the distance and direction calculated in the latest run of the above thread.
3. Main activity thread: opens the smartphone's camera and wakes the above threads.



Control Loop:



Difficulties

- Identifying the object color without letting the surrounding environment to damage the object detection.
- Filtering out red obstacles and keeping out the performance (FPS) as high as possible.

Possible improvements

We can suggest some improvements for future use:

- Defining the object automatically
  defining the object from the first frame grabbed, and then setting the hue range that represents the object.
- When the object is missing search the room in order to identify it
  if the object is not found scan the room $360°$ degrees in order to locate the object.
- Dynamic speed robot
  when object is far the robot moves faster than when it is close. This can save battery when the object is close and might prevent contact loss when the object is fast or far.
- Broadcasting the robot's sight
  broadcasting the robot's sight to the internet in order to follow its movement remotely.