Principles of Robot Autonomy I
Problem Set 3

Problem 1: Camera Calibration

Results of code inside Gradescope.

Problem 2: Line Extraction

MIN_SEG_LENGTH = 0.05 # minimum length of each line segment (m)

LINE_POINT_DIST_THRESHOLD = 0.02 # max distance of pt from line to split

MIN_POINTS_PER_SEGMENT = 2 # minimum number of points per line segment

MAX_P2P_DIST = 0.25 # max distance between two adjent pts within a segment

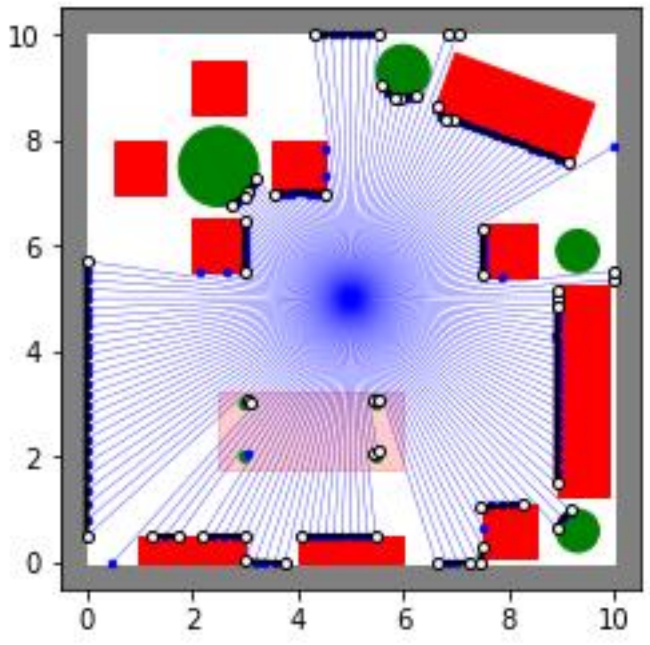


Figure 1 - rangeData_5_5_180.csv

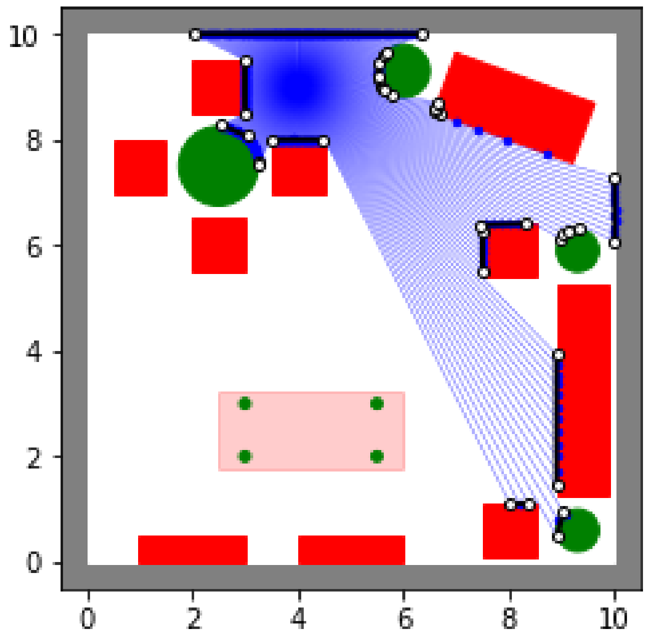


Figure 2 - rangeData_4_9_360.csv

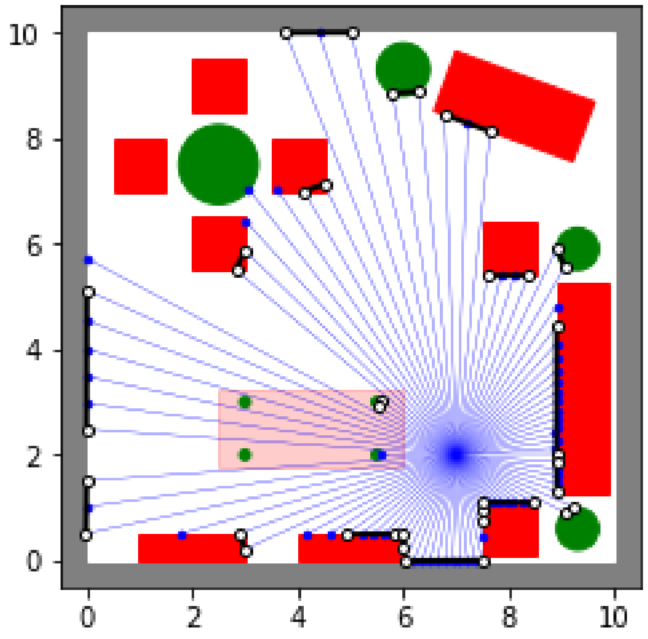


Figure 3 - rangeData_7_2_90.csv

Problem 3: Linear Filtering

(i) Given Image Matrix:

$$I = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

First needed to apply zero-pad:

$$I \in \mathbb{R}^{(m+2*(k//2))x(n+2*(l//2))}, \text{ for } F \in \mathbb{R}^{kxl}$$

$$I_{zero-pad} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Compute G as the result of $G = F \otimes I$:

(a) $F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, F \otimes I = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

(b) $F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, F \otimes I = \begin{bmatrix} 2 & 3 & 0 \\ 5 & 6 & 0 \\ 8 & 9 & 0 \end{bmatrix}$

(c) $F = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, F \otimes I = \begin{bmatrix} 7 & 4 & -7 \\ 15 & 6 & -15 \\ 13 & 4 & -13 \end{bmatrix}$

* This filter type is an edge detection feature. It can be used in Computer vision to proceed feature extraction.

(d) $F = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, F \otimes I = \frac{1}{16}\begin{bmatrix} 21 & 32 & 33 \\ 52 & 80 & 68 \\ 57 & 84 & 69 \end{bmatrix}$

* This type of filter is a Gaussian Filter that could be used to blur (smooth, average) the original image.

(ii) Prove that:

$$G(i,j) = \sum_{u=1}^{k}\sum_{v=1}^{l}\sum_{w=1}^{c} F(u,v,w).I(i+u-1, j+v-1, w)$$

can be written as,

$$G(i,j) = f^T t_{ij}$$

Change sum for channel w

$$G(i,j) = \sum_{w=1}^{c}\sum_{u=1}^{k}\sum_{v=1}^{j} F(u,v,w).I(i+u-1, j+v-1, w)$$

For the innermost sum respect v,

$$G(i,j) = \sum_{w=1}^{c}\sum_{u=1}^{k} F(u,1,w).I(i+u-1,j,w) + F(u,2,w).I(i+u-1,j+1,w) + \cdots + F(u,l,w).I(i+u-1,j+l-1,w)$$

For second sum respect to u,

$$\begin{aligned}G(i,j) = \sum_{w=1}^{c} & F(1,1,w).I(i,j,w) + F(1,2,w).I(i,j+1,w) + \cdots + F(1,l,w).I(i,j+l-1,w) \\ & + F(2,1,w).I(i+1,j,w) + F(2,2,w).I(i+1,j+1,w) + \cdots \\ & + F(2,l,w).I(i+1,j+l-1,w) + \cdots + F(k,1,w).I(i+k-1,j,w) \\ & + F(k,2,w).I(i+k-1,j+j,w) + \cdots + F(k,l,w).I(i+k-1,j+l-1,w)\end{aligned}$$

And finally, for the last sum respect w,

$$\begin{aligned}G(i,j) = & F(1,1,1).I(i,j,1) + \cdots + F(1,l,1).I(i,j+l-1,1) + F(2,1,1).I(i+1,j,1) + \cdots \\ & + F(2,l,1).I(i+1,j+l-1,1) + F(k,1,1).I(i+k-1,j,1) + \cdots \\ & + F(k,l,1).I(i+k-1,j+l-1,1) + \cdots + F(k,1,c).I(i+k-1,j,c) + \cdots \\ & + F(k,l,c).I(i+k-1,j+l-1,c)\end{aligned}$$

Rewritten in a correlation operation form,

$$\begin{bmatrix} F(1,1,1) & F(1,2,1) & \cdots & F(1,l,1) \\ F(2,1,1) & F(2,2,1) & \cdots & F(2,l,1) \\ \vdots & \vdots & \ddots & \vdots \\ F(k,1,1) & F(k,2,1) & \cdots & F(k,l,1) \\ \vdots & \vdots & \ddots & \vdots \\ F(1,1,c) & F(1,2,c) & \cdots & F(1,l,c) \\ F(2,1,c) & F(2,2,c) & \cdots & F(2,l,c) \\ \vdots & \vdots & \ddots & \vdots \\ F(k,1,c) & F(k,2,c) & \cdots & F(k,l,c) \end{bmatrix}^T$$

$$\otimes$$

$$\begin{bmatrix} I(i,j,1) & I(i,j+1,1) & \cdots & I(i,j+l-1,1) \\ I(i+1,j,1) & I(i+1,j+1,1) & \cdots & I(i+1,j+l-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ I(i+k-1,j,1) & I(i+k-1,j+1,1) & \cdots & I(i+k-1,j+l-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ I(i,j,c) & I(i,j+1,c) & \cdots & I(i,j+l-1,c) \\ I(i+1,j,c) & I(i+1,j+1,c) & \cdots & I(i+1,j+l-1,c) \\ \vdots & \vdots & \ddots & \vdots \\ I(i+k-1,j,c) & I(i+k-1,j+1,c) & \cdots & I(i+k-1,j+l-1,c) \end{bmatrix}$$

And,

$$f^T = [F(1,1,1) \quad F(1,2,1) \quad \cdots F(k,1,c) \quad F(k,2,c) \quad \cdots \quad F(k,l,c)]_{1x(kxlxc)}$$

$$t_{ij} = \begin{bmatrix} I(i,j,1) \\ I(i,j+1,1) \\ \vdots \\ I(i+k-1,j,c) \\ I(i+k-1,j+1,c) \\ \vdots \\ I(i+k-1,j+l-1,c) \end{bmatrix}_{(kxlxc)x1}$$

Where $G(i,j) = f^T t_{ij}$.

(iii) Results:

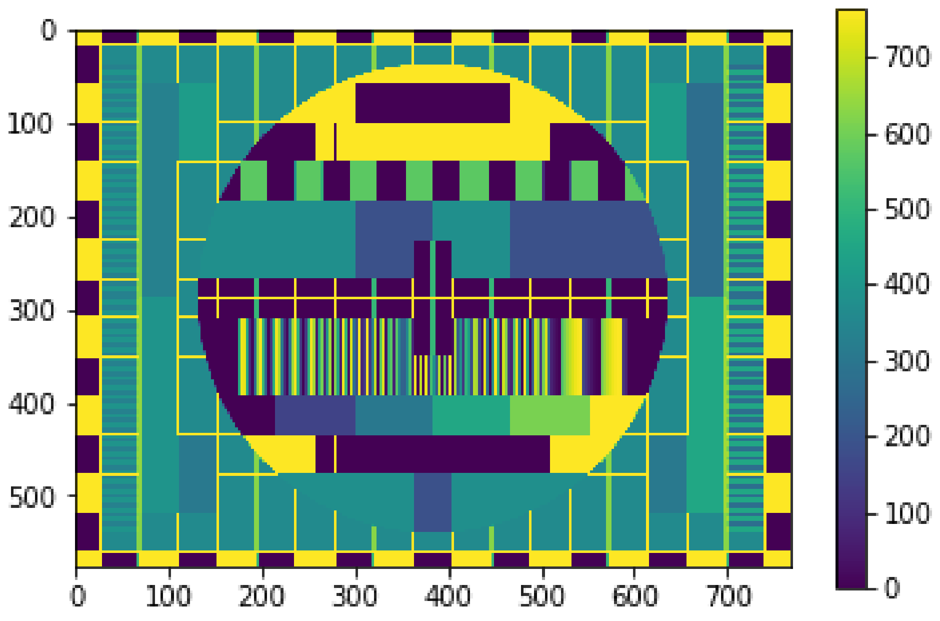


Figure 4 - Filter 01

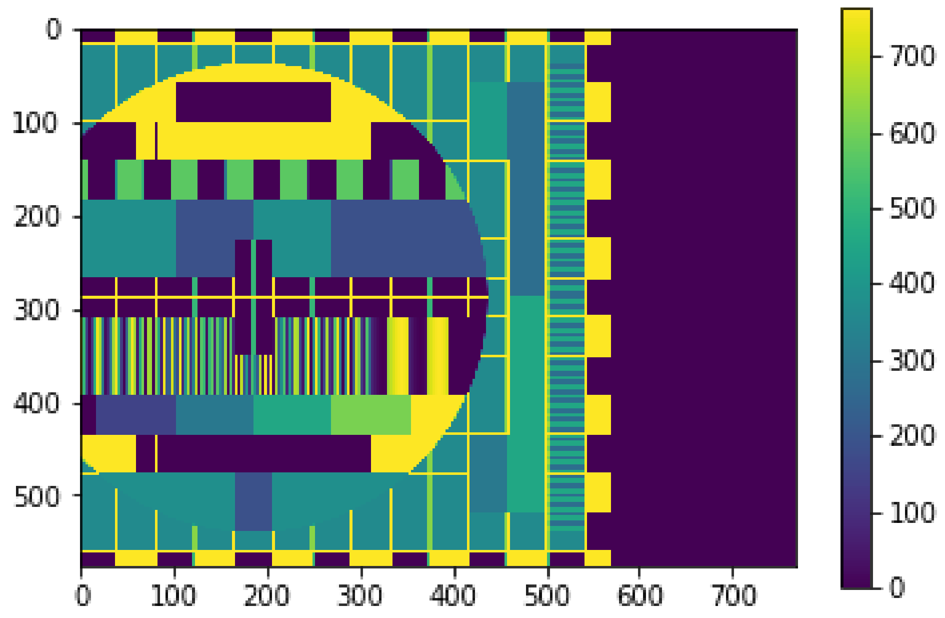


Figure 5 - Filter 02

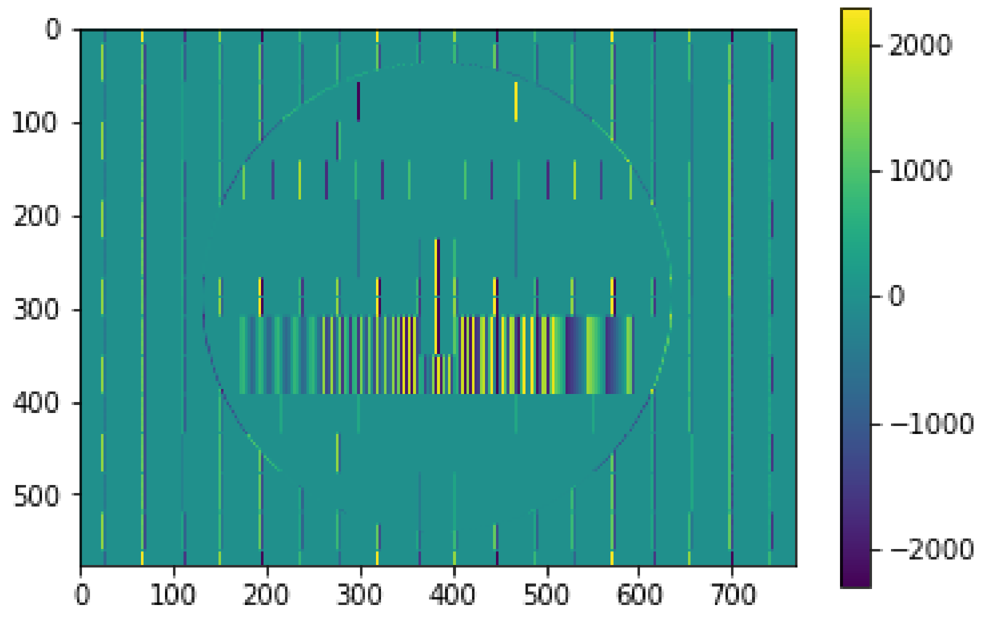


Figure 6 - Filter 03

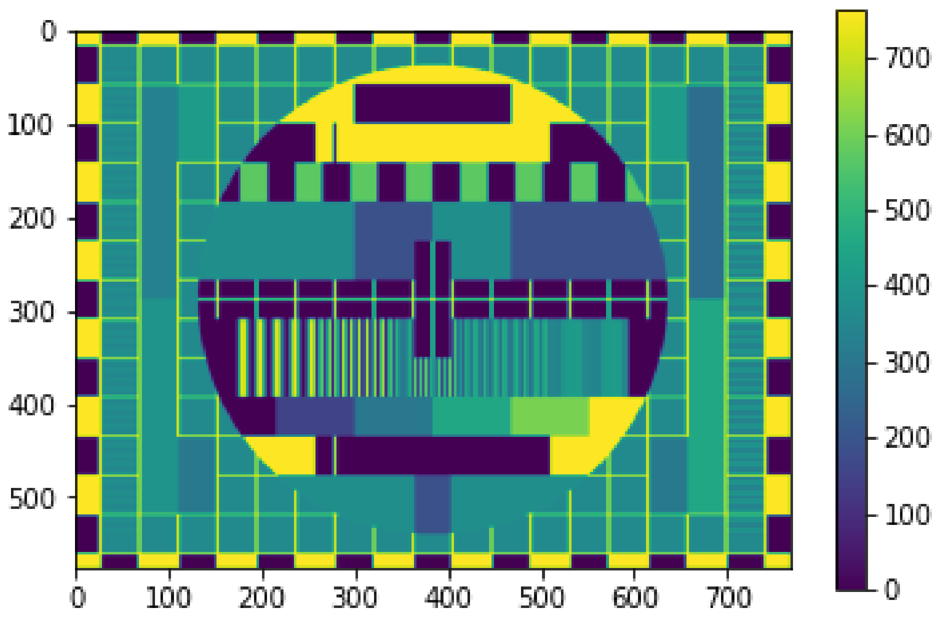


Figure 7 - Filter 04

(iv)

```
Correlation function runtime: 2.45300006866 s
Correlation function runtime: 5.51600003242 s
Correlation function runtime: 2.51799988747 s
Correlation function runtime: 2.73399996758 s
```

(v) Matrix written as an outer product, $F = ff^T$, where $f \in \mathbb{R}^{kx1}$ and $F \in \mathbb{R}^{kxk}$:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_k \end{bmatrix} [f_1 \quad f_2 \quad f_3 \quad \cdots \quad f_k] = \begin{bmatrix} f_1^2 & f_1f_2 & f_1f_3 & \cdots & f_1f_k \\ f_2f_1 & f_2^2 & f_2f_3 & \cdots & f_2f_k \\ f_3f_1 & f_3f_2 & f_3^2 & \cdots & f_3f_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_kf_1 & f_kf_2 & f_kf_3 & \cdots & f_k^2 \end{bmatrix}$$

Then, the final outer product is a F matrix which all columns are linearly dependent from the first one, this affect the Rank of matrix, Rank(F) = 1, so this the mathematical condition to be write as outer product. To find the f vector, take the first column of matrix F and divide by the first element f_{11}.

(vii) Convolution implemented with correlation:

$$G(i,j) = \sum_{u=1}^{k} \sum_{v=1}^{l} F(u,v).I(i-u+1, j-v+1), \text{ Convolution Formula}$$

$$G(i,j) = \sum_{u=1}^{k} \sum_{v=1}^{l} F(u,v).I(i+u-1, j+v-1), \text{ Correlation Formula}$$

In the formula of correlation, applying u =- u and v =- v,

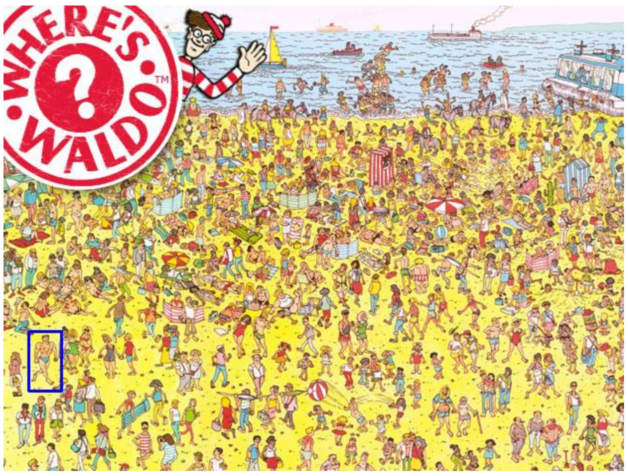
$$G(i,j) = \sum_{u=1}^{k} \sum_{v=1}^{l} F(-u,-v).I(i-u+1, j-v+1)$$

The parameter $F(-u, -v)$ is the correlation filter rotation by π, we get:

$$\begin{bmatrix} F(k,l) & \cdots & F(2,l) & F(1,l) \\ \vdots & \ddots & \vdots & \vdots \\ F(k,2) & \cdots & F(2,2) & F(1,2) \\ F(k,1) & \cdots & F(2,1) & F(1,1) \end{bmatrix} * I(i-u+1, j-v+1)$$

Problem 4: Template Matching

(i)

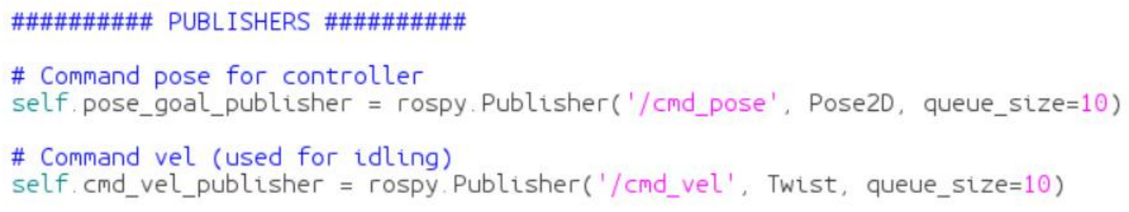


(iii) The template matching gives us poor results when running with the stop signal template, the template as smaller in relation some images and in another cases the stop signal has different perspectives and sizes. One method to improve the Template Matching can be made by using more than one template these other templates has different sizes and perspectives. The second method proposed is by hybridizing the feature-based and template-based approaches.

(source: https://en.wikipedia.org/wiki/Template_matching)

Problem 5: Stop Sign Detection and FSM in ROS

(i)



```
########## PUBLISHERS ##########

# Command pose for controller
self.pose_goal_publisher = rospy.Publisher('/cmd_pose', Pose2D, queue_size=10)

# Command vel (used for idling)
self.cmd_vel_publisher = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
```

subscriber.py publishes to the cmd_pose and cmd_vel topics. The message types are Pose2D and Twist respectively.

(iv)

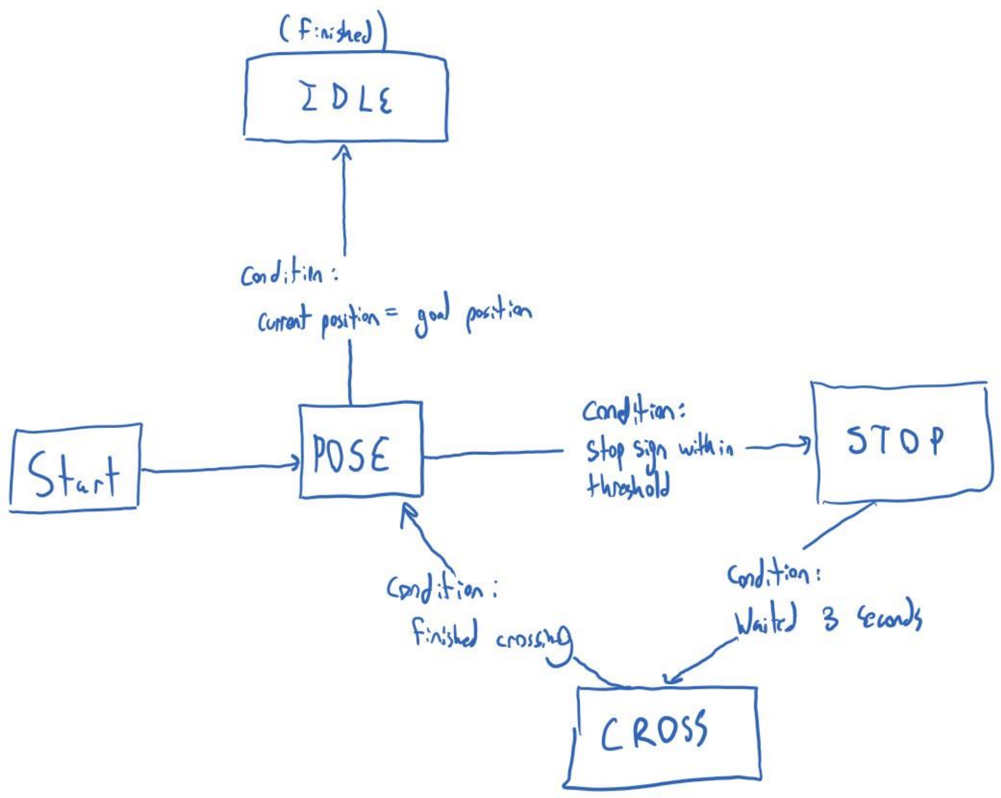


Figure 8 – FSM

(viii)

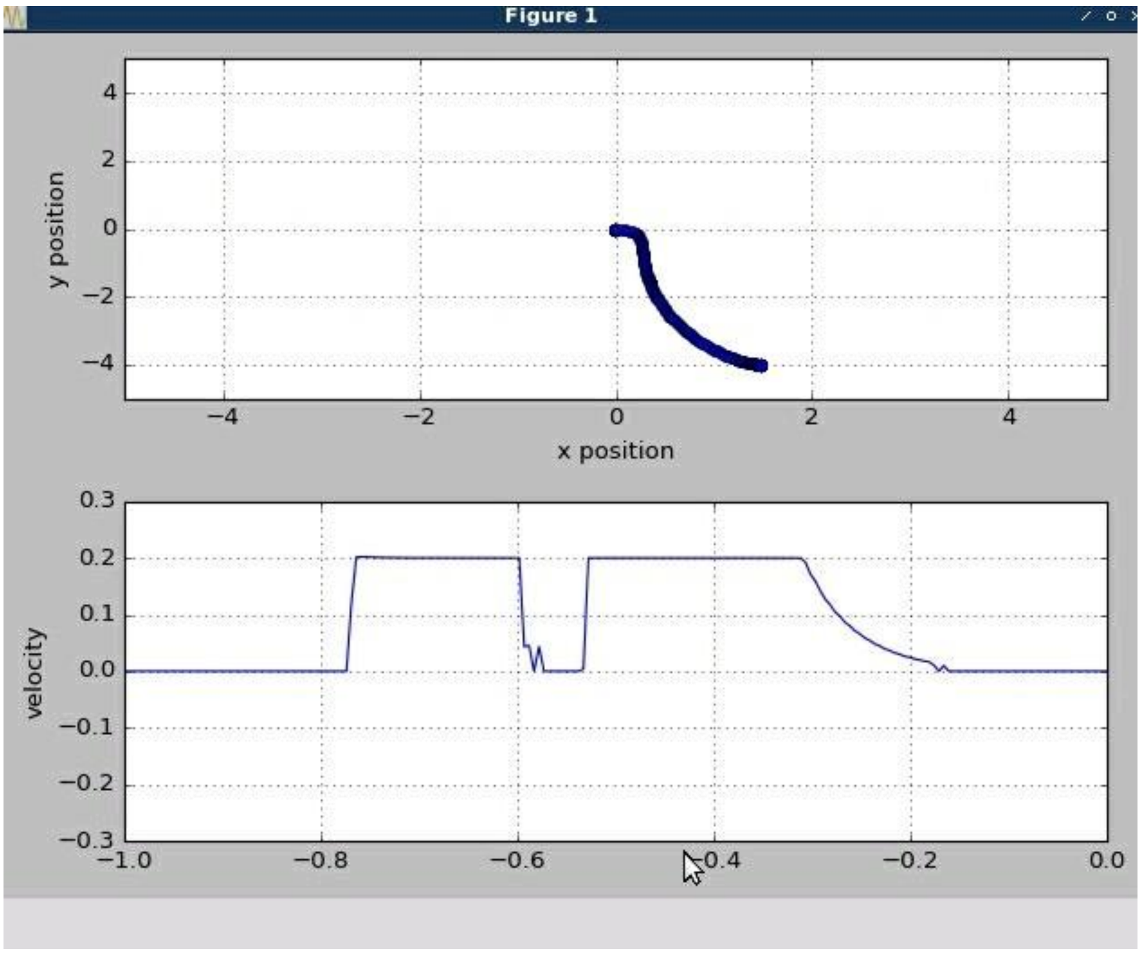


Figure 9 - Velocity and Position Profile

Extra Problem: Image Pyramids

(ii) Comparing the downscaled images to original we can see that the quality decreases, it’s happens by the fact that in the downscale process we remove pixels from the original image.

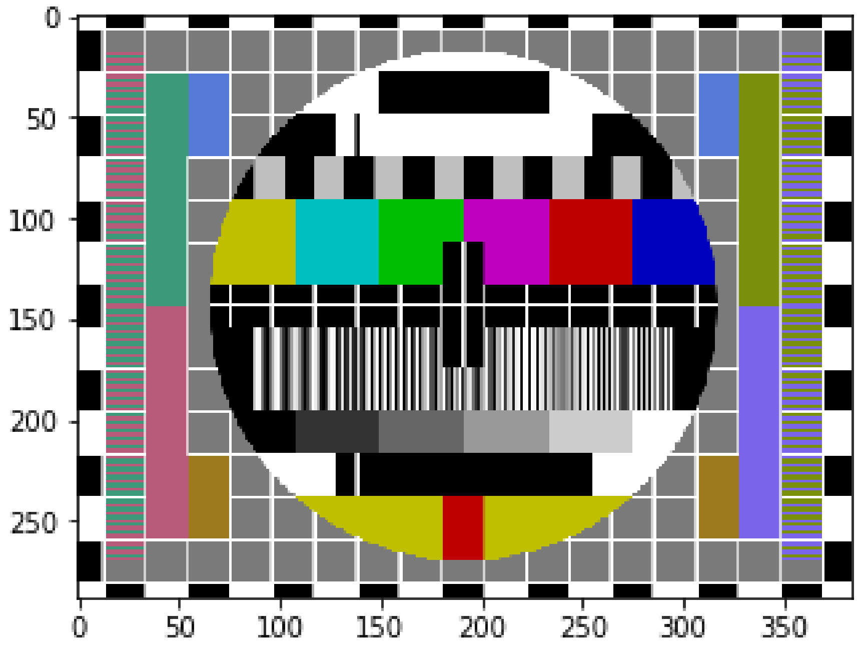


Figure 10 - Half downscale 01

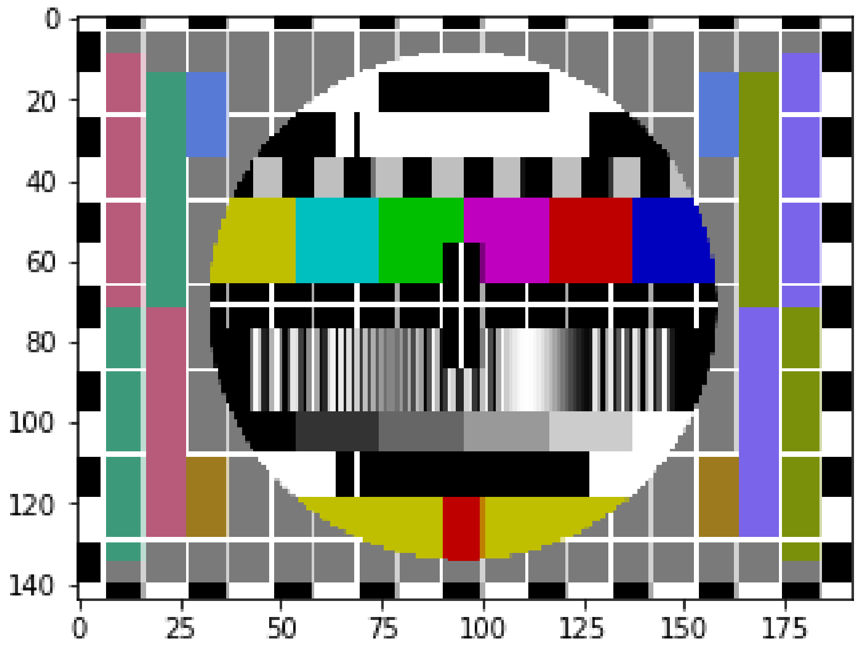


Figure 11 - Half downscale 02

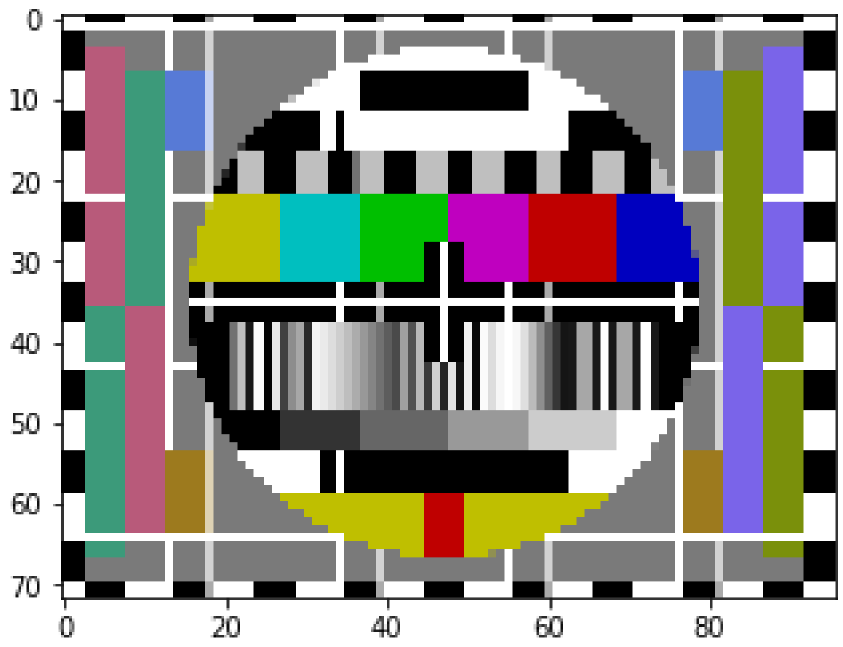


Figure 12 - Half downscale 03

(iii) Applying Blur to the image we are calculating an average of pixel intensity, we are smoothing our image and removes outlier's pixels that can be cause some noise in the image, then when we applying the downscaled the removed pixels don't affect the image details as before.

(iv) How to expected the results are better than *half_downscale*.

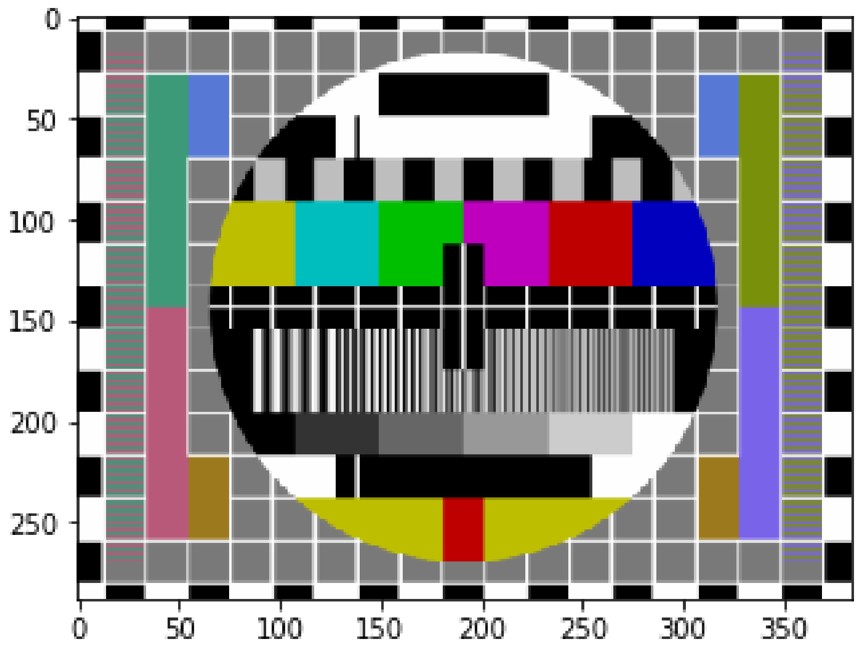


Figure 13 - Blur half downscale 01

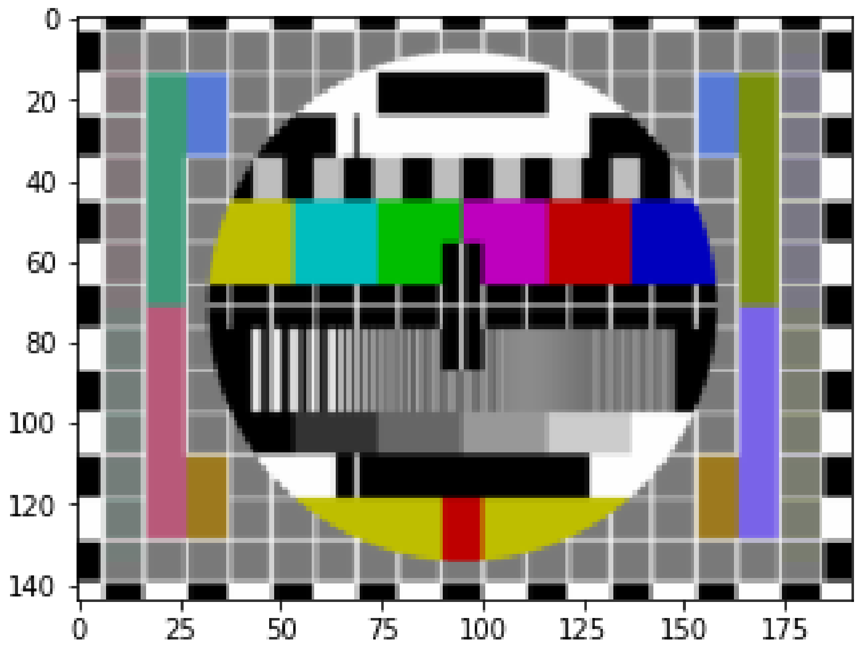


Figure 14- Blur Half downscale 02

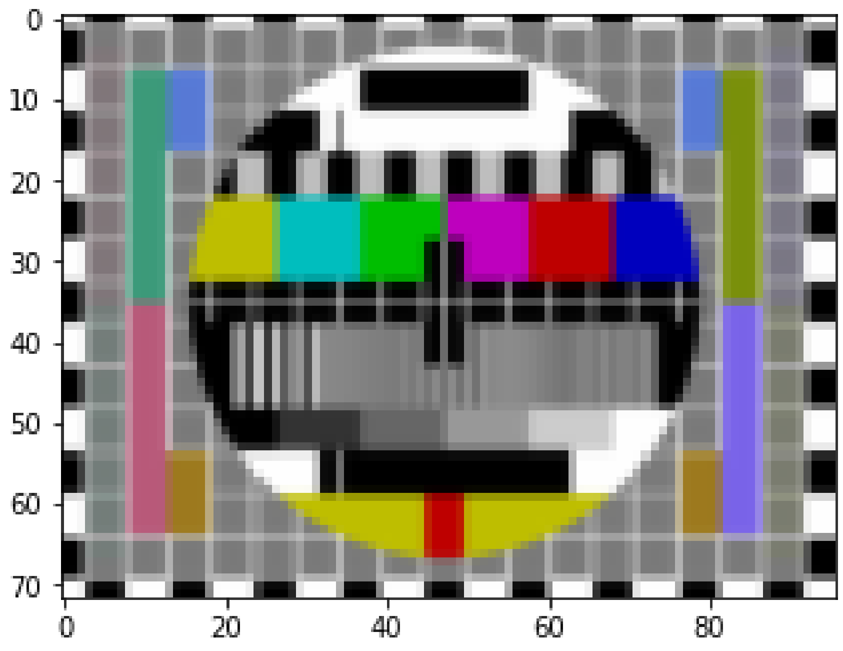


Figure 15 – Blur Half downscale 03

(vi)

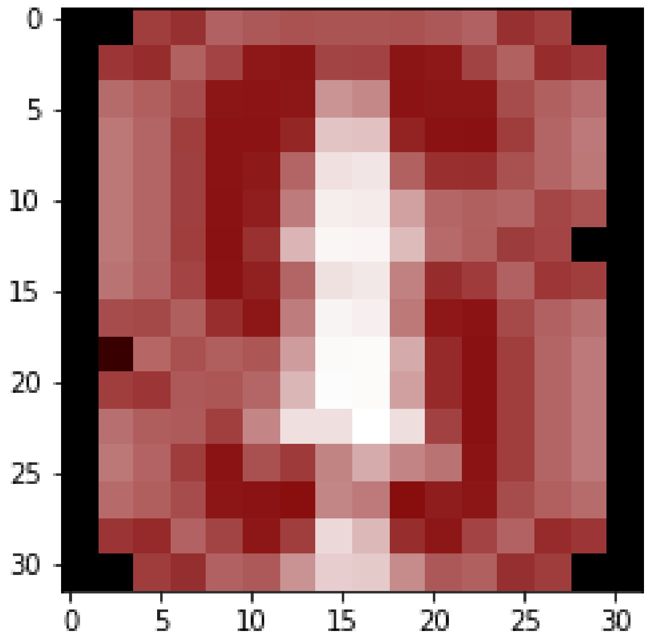


Figure 16 - Two upscale 01

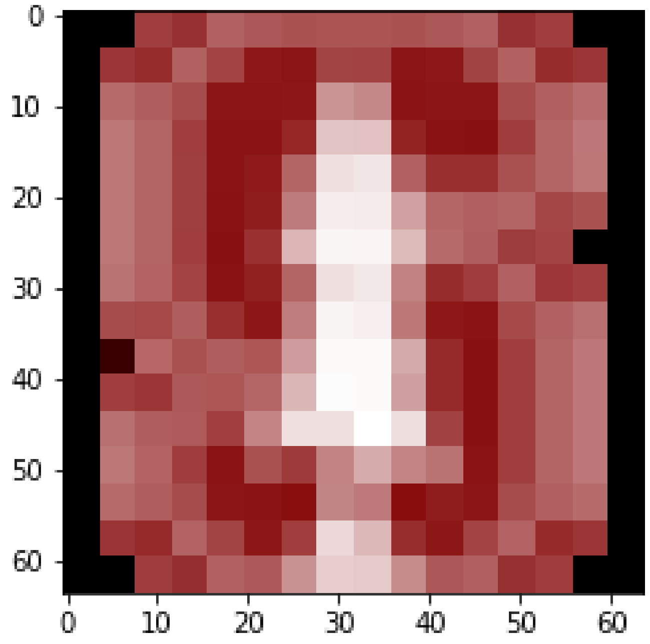


Figure 17 - Two upscale 02

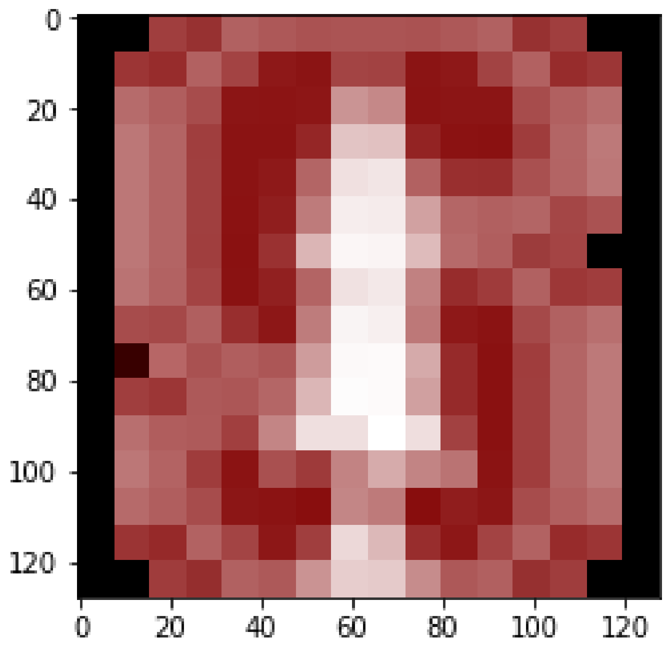


Figure 18 - Two upscale 03

(vii)

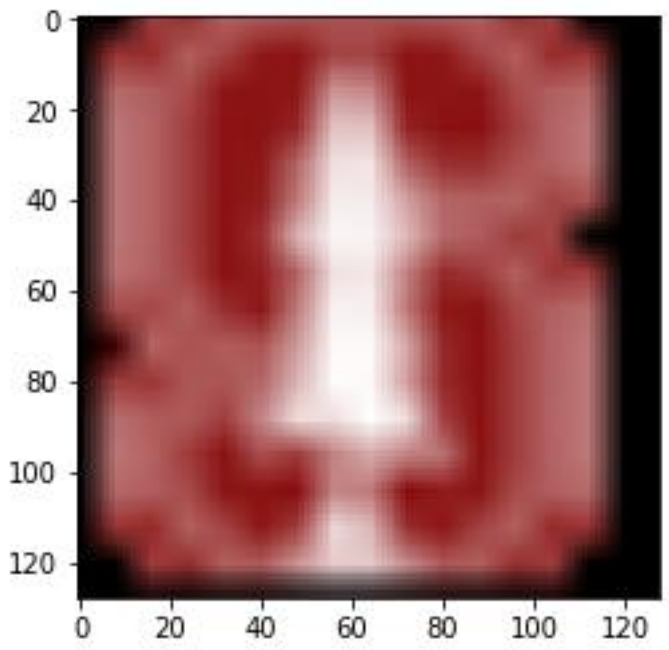


Figure 19 - bilinear interpolation, rescale = 8

* The bilinear transformation is used when an image need to be upscaled e.g., in our case each pixel is stretched to borders of the rescaled image by the scaler factor, the "holes" between this interpolation need to be calculated and filled with appropriate values. The bilinear interpolation uses the values of the nearest pixels located in diagonal directions and take the weighted average (accordingly with the filter used) of these values to fill the value of specific pixel.

(viii)

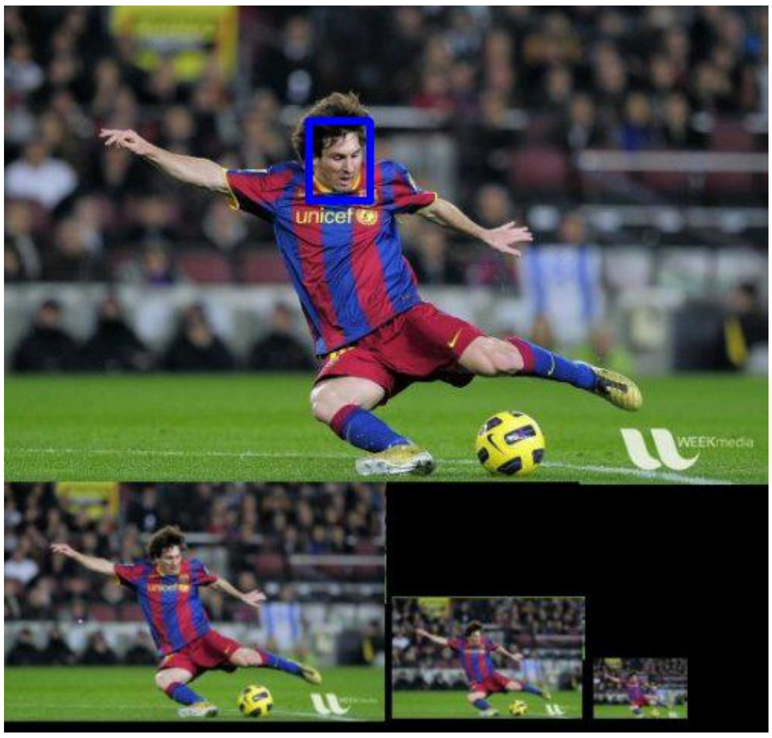


Figure 20 - Messi

(ix)

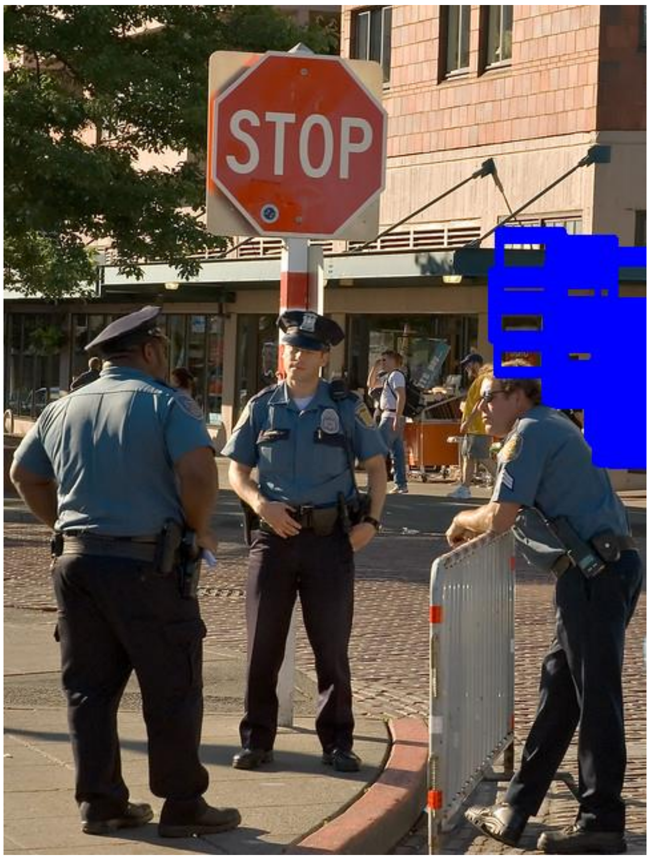


Figure 21 - Stop sign 01

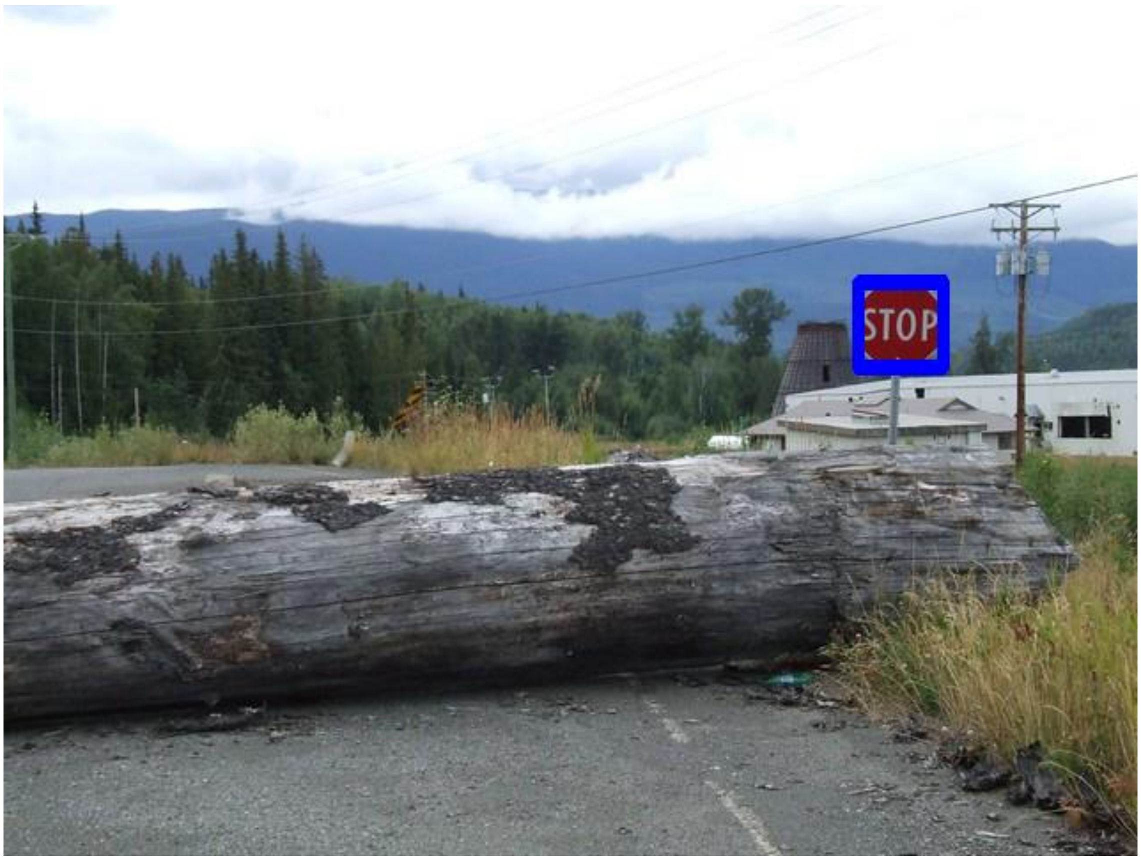


Figure 22 - Stop sign 02

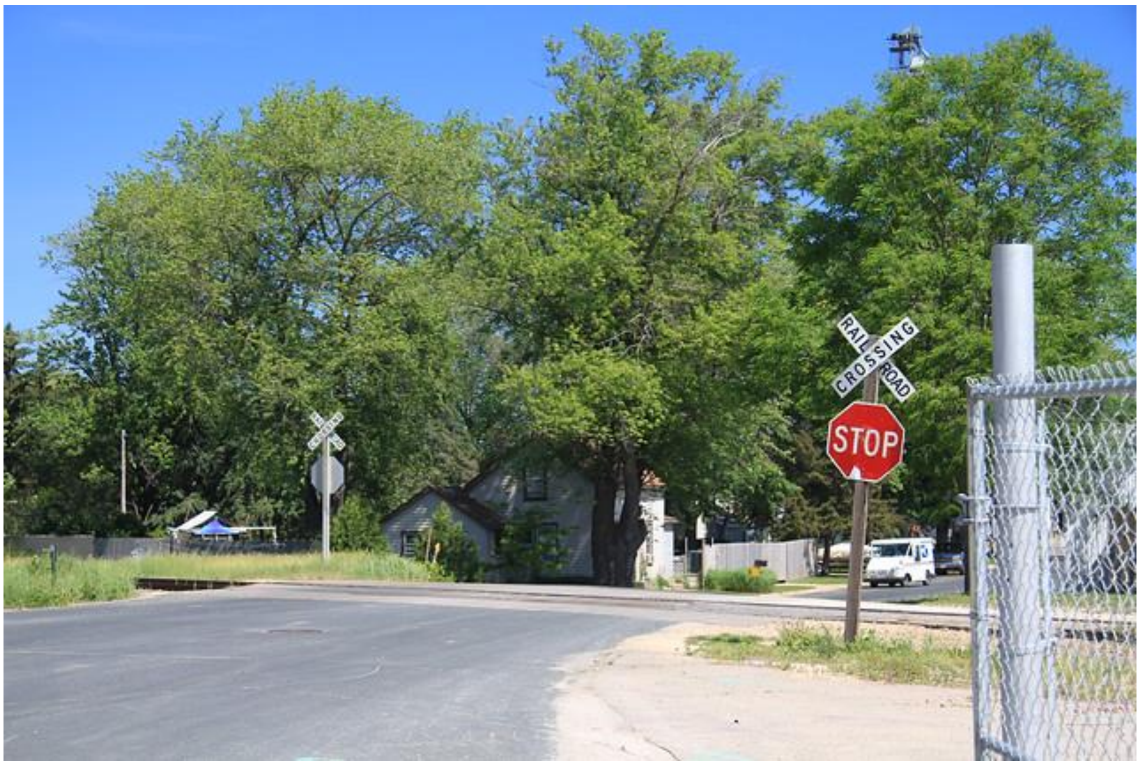


Figure 23 - Stop sign 03

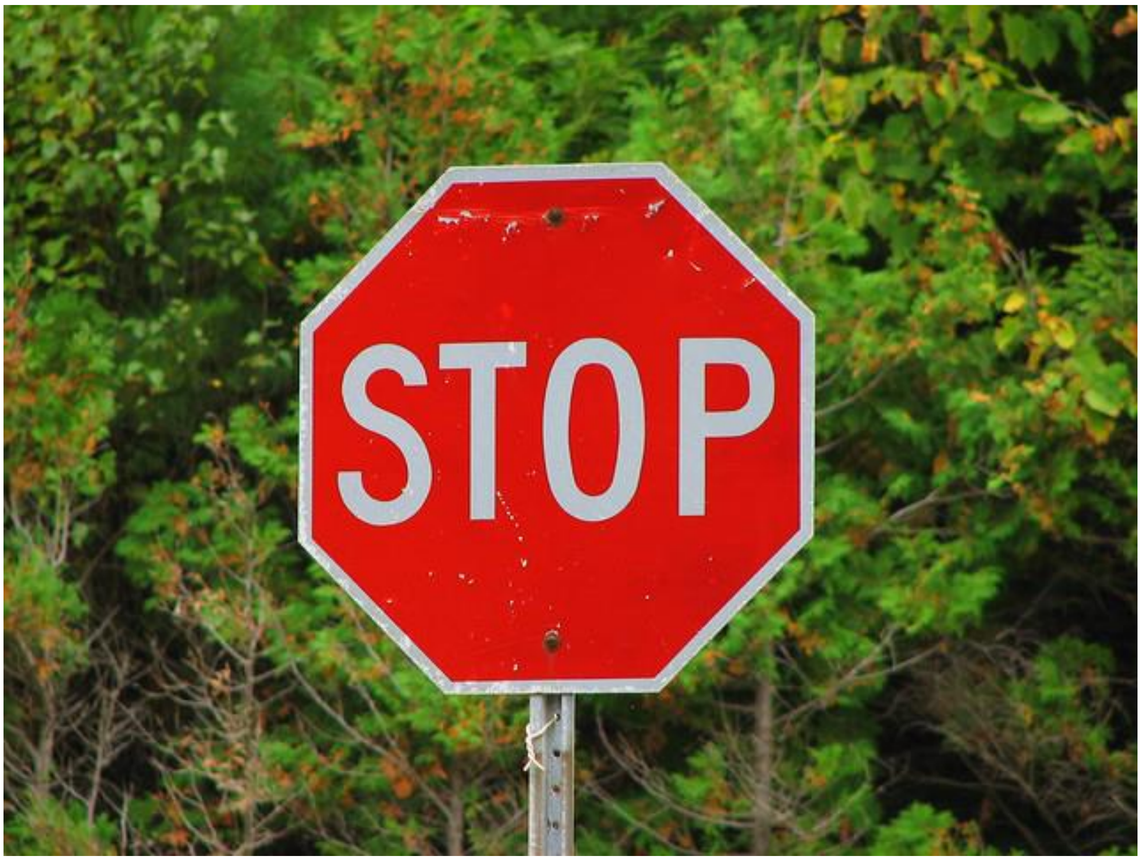


Figure 24 - Stop sign 04

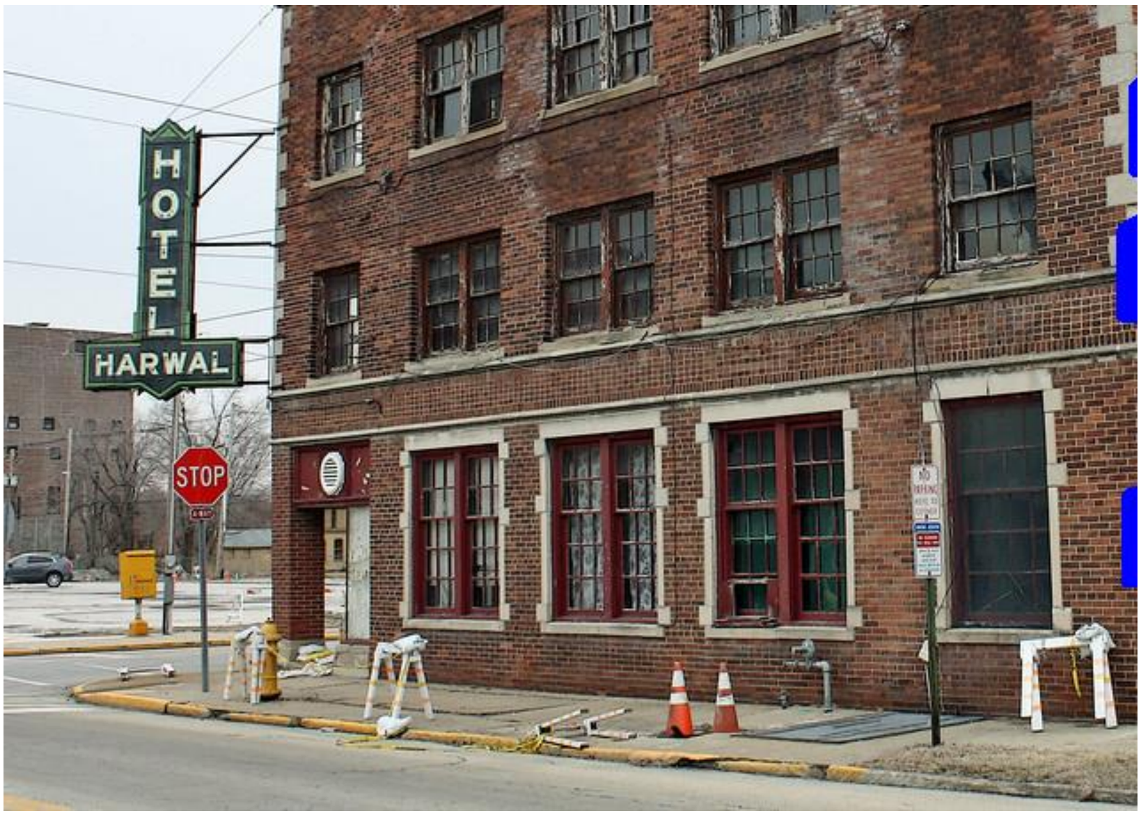


Figure 25 - Stop sign 05

* The results were bad in my case.