POLITECNICO DI MILANO

AUTONOMOUS VEHICLES

S. Arrigoni

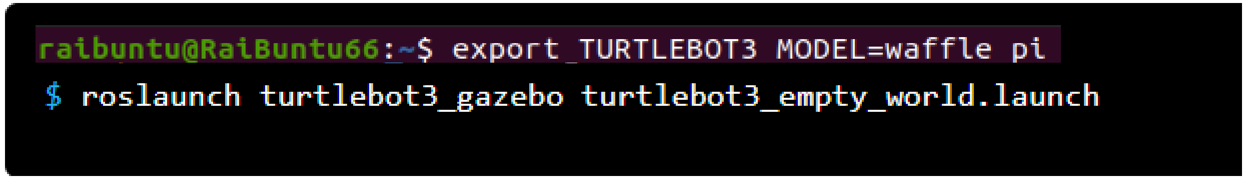
POLITECNICO
MILANO 1863

Software requirements

ROS + MATLAB®

Initialize the system

- Initialize gazebo with turtlebot3_empty_world.launch



```
raibuntu@RaiBuntu66:~$ export TURTLEBOT3 MODEL=waffle pi
 $ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

- Initialize MATLAB/Simulink and connect it to ROS.

rosinit

rosinit(‘ip_address of the master’)

(if using Virtual Machine)

ROS

Modify

GAZEBO

Modify the scenario in GAZEBO

```
% connect to GAZEBO


rosinit % IPadress if Vmachine

gazebo = ExampleHelperGazeboCommunicator; % create communication object
```

Modify the scenario in GAZEBO

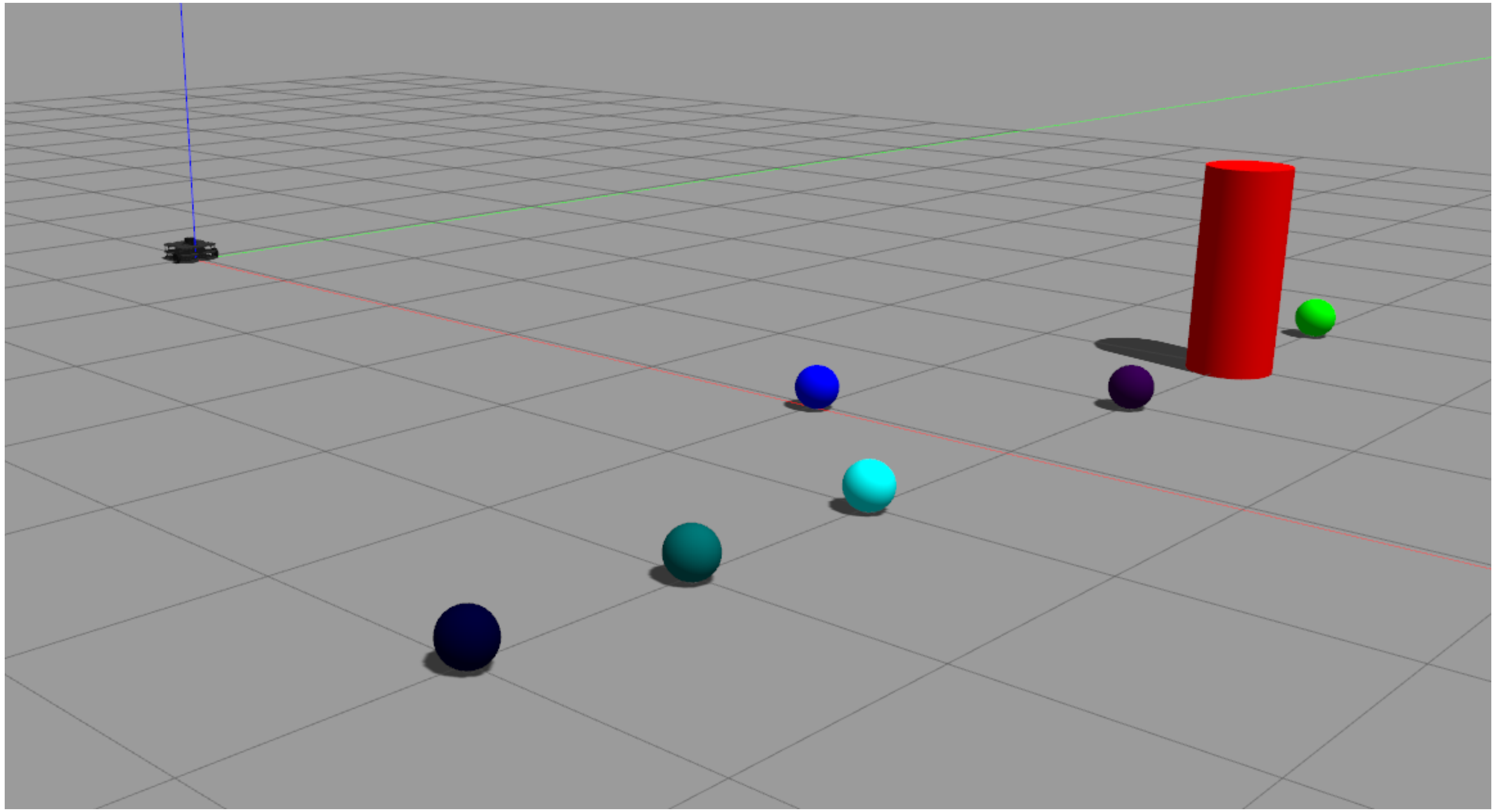
```
% add objects in GAZEBO


cil = ExampleHelperGazeboModel("Cil")
cilLink = addLink(cil,"cylinder",[1 0.2],"color",[1 0 0 1]) %type, [h rad], color
spawnModel(gazebo,cil,[6 2 1]) % position



ball1 = ExampleHelperGazeboModel("Ball")
sphereLink = addLink(ball1,"sphere",0.1,"color",[0 0 1 1]) %type, rad, color
spawnModel(gazebo,ball1,[5 0 0.1]) % position
```

Create MATLAB object

Define features

Define spawn position

Modify the scenario in GAZEBO



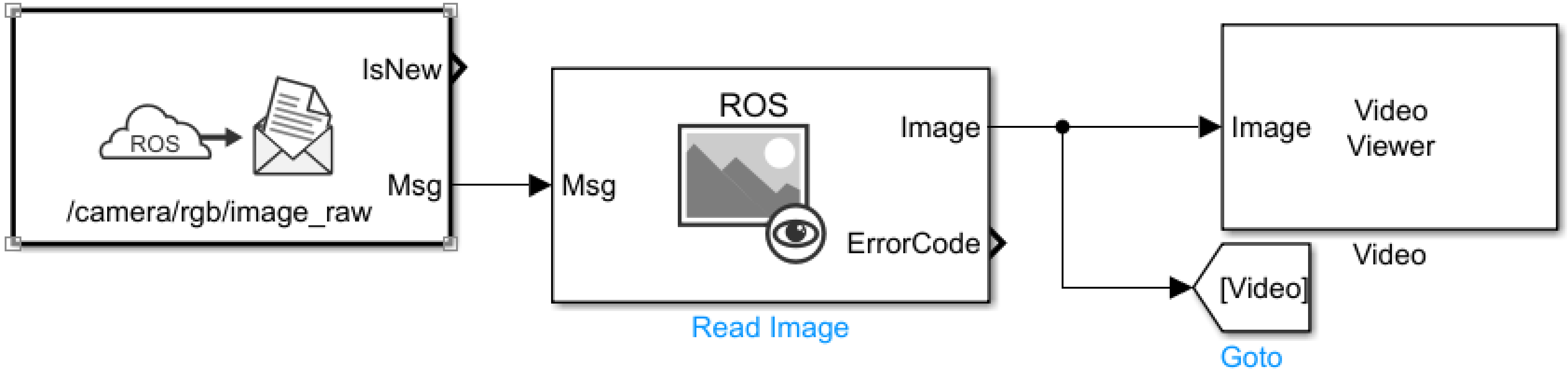
ROS

Image processing

SIMULINK®

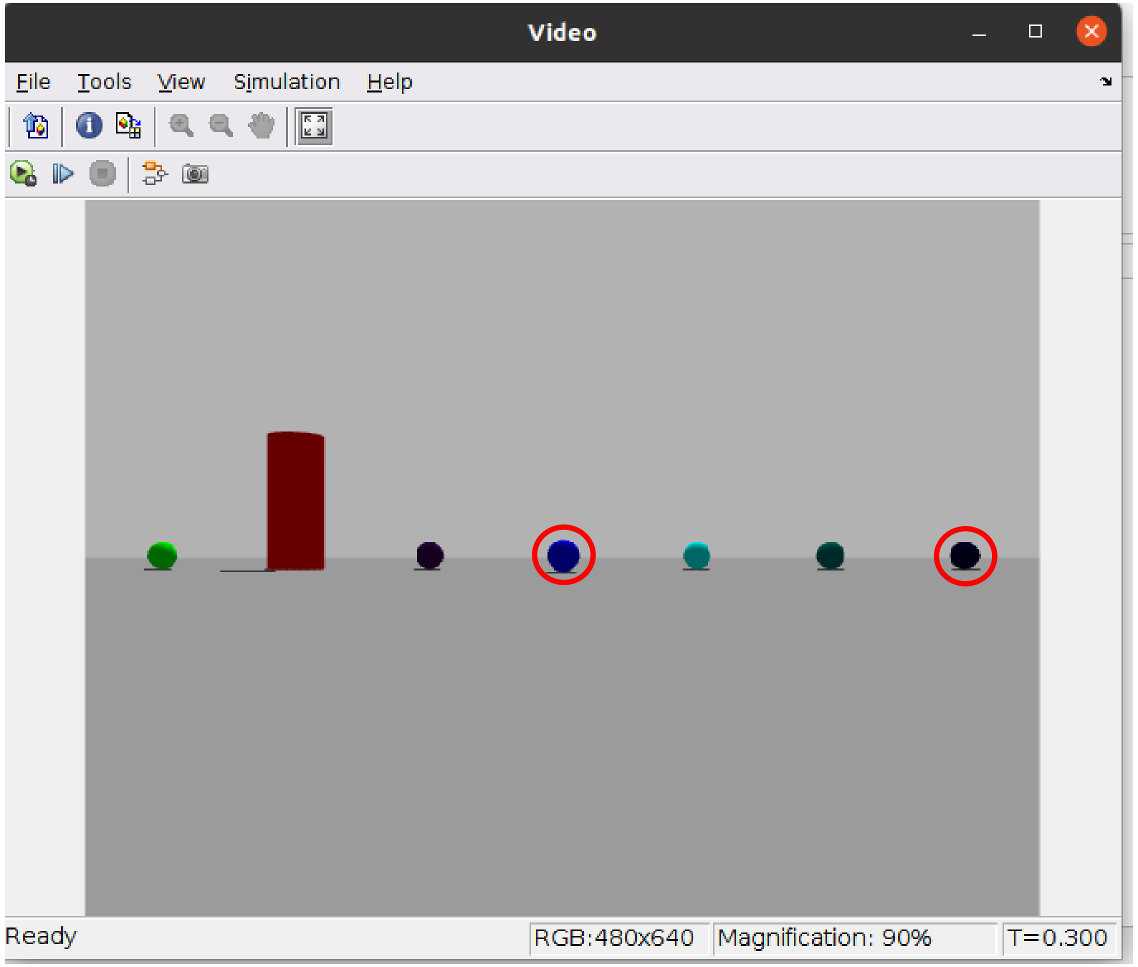
Simple image processing based on colors

#from lesson 4



Simple image processing based on colors

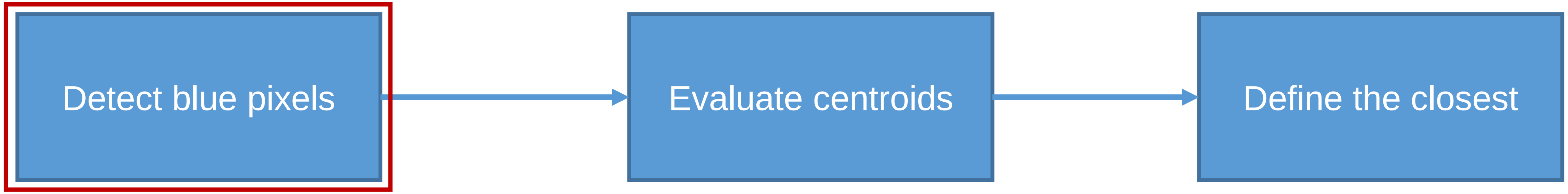
#expected result



1. *Let’s detect blue balls!*
2. *Define centroid pos of the closest*

Simple image processing based on colors

1. *Let’s detect blue balls!*
2. *Define centroid pos of the closest*



Detect blue pixels

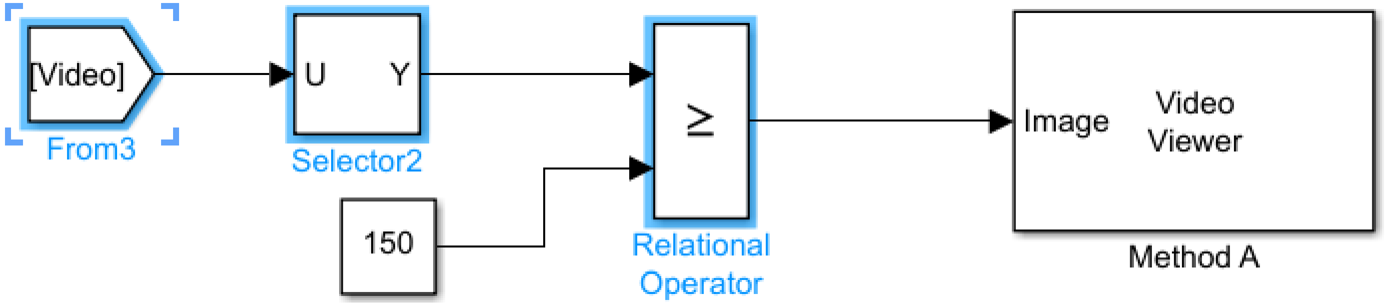
Evaluate centroids

Define the closest

Simple image processing based on colors

# Method A : RGB

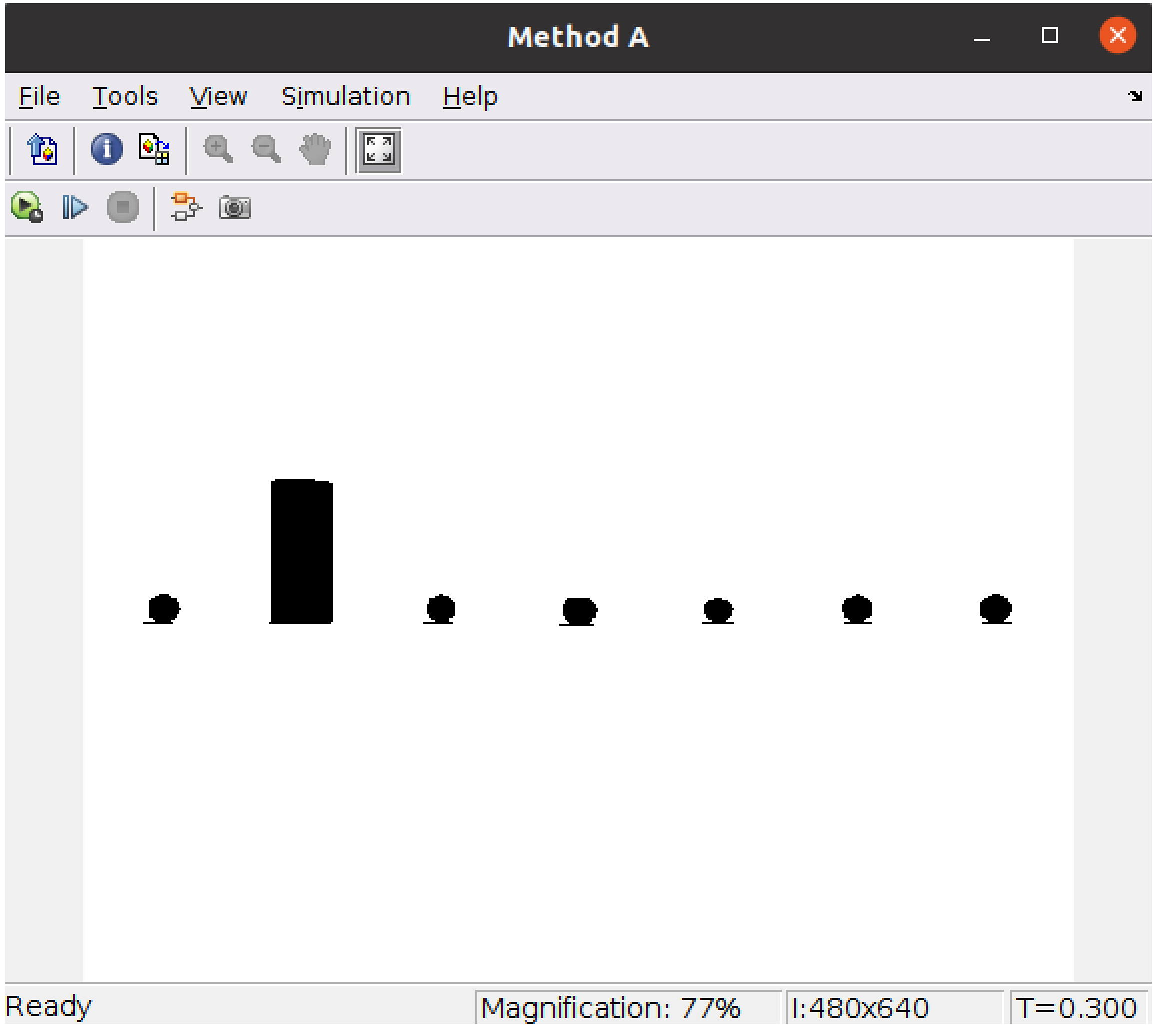
1. *Select Blue channel*
2. *Generate Bool Matrix*
3. *Evaluate threshold*



Simple image processing based on colors

# Method A : RGB

# result… ☹

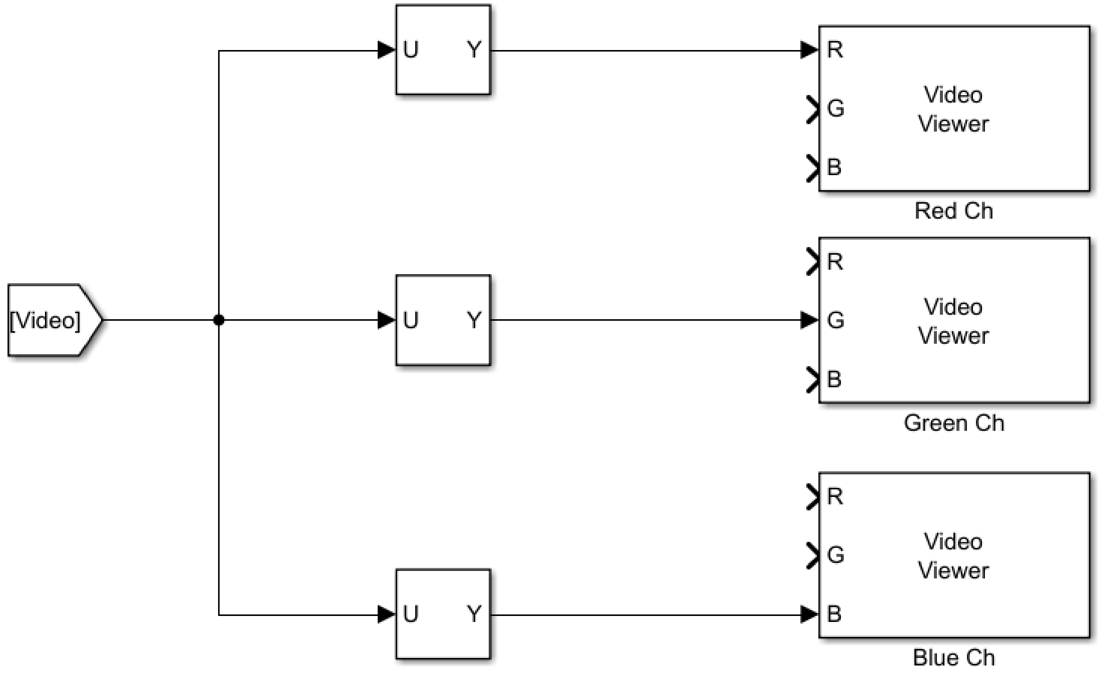


Simple image processing based on colors

Why is not working?

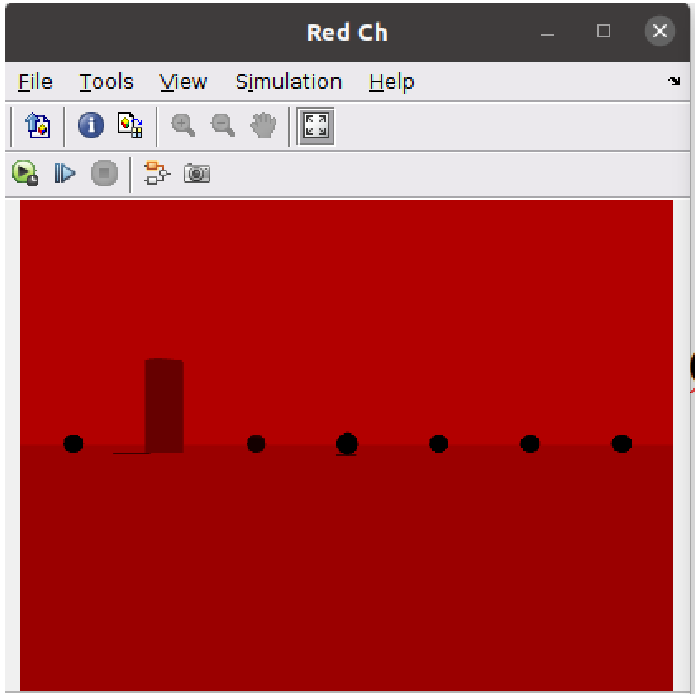
Simple image processing based on colors

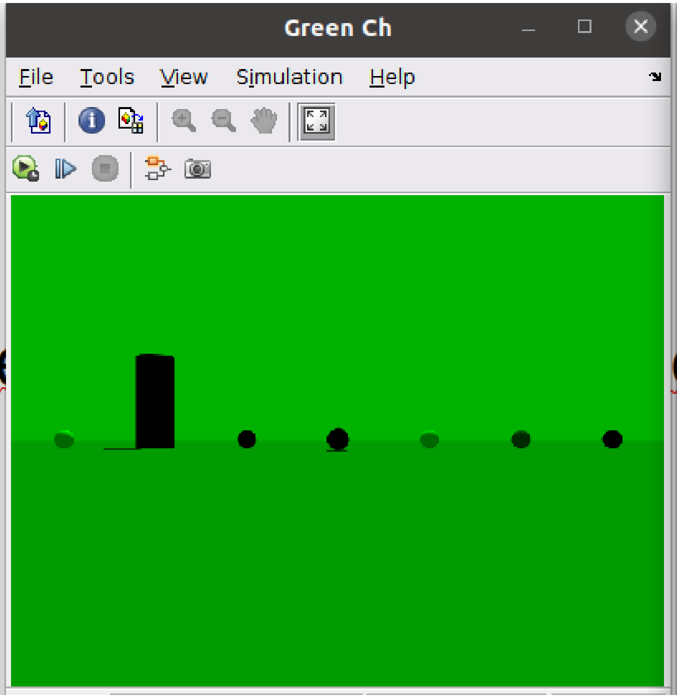
Let’s visualize RGBs channels

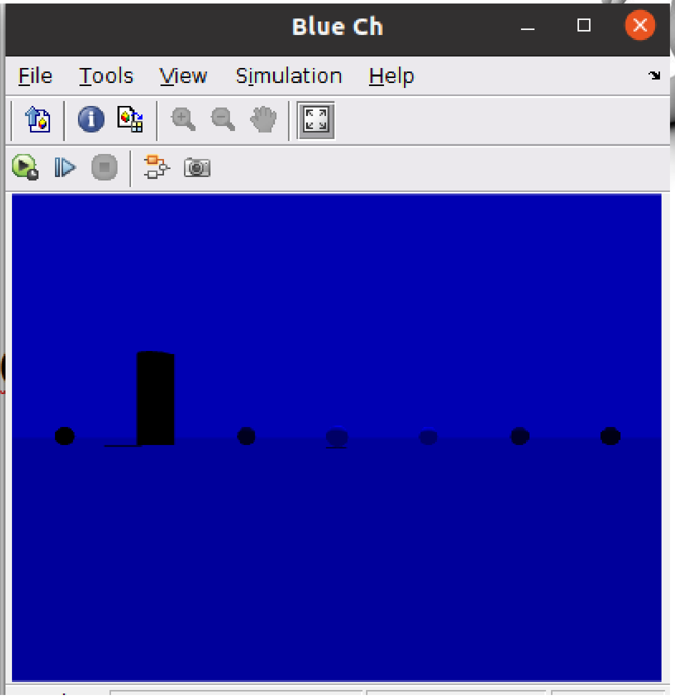


Simple image processing based on colors

…







Simple image processing based on colors

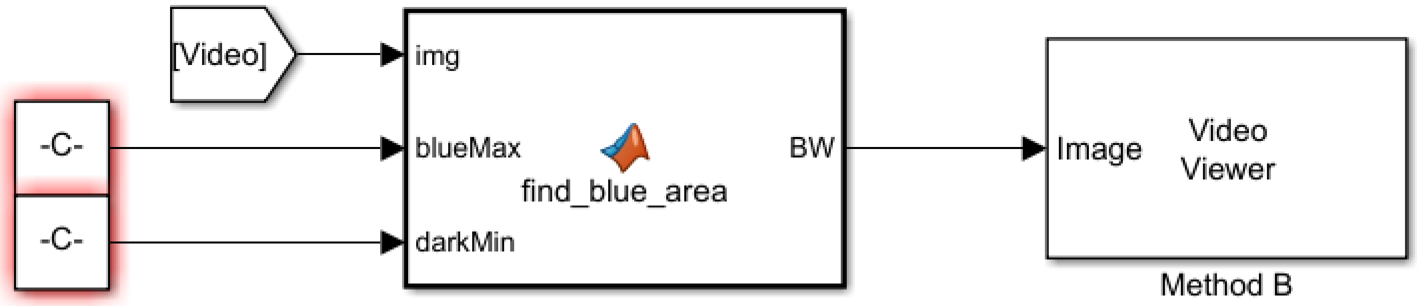
Let’s google it!

MATLAB® **TurtleBotObjectTrackingExample**

Ref: https://it.mathworks.com/help/supportpkg/turtlebotrobot/ug/track-and-follow-an-object-using-a-turtlebot.html

Simple image processing based on colors

# Method B : MATLABexample



Simple image processing based on colors

# Method B : MATLABexample

Let’s dig a little in the code:

```
% Isolate blue color by combining blue images and dark images together

blueImg = img(:,:,1)/2 + img(:,:,2)/2 - img(:,:,3)/2;
blueThresh = blueImg < params.blueMax;

darkIso = -img(:,:,1)/2 - img(:,:,2)/2 + 3*img(:,:,3) - 2*rgb2gray(img);
darkThresh = darkIso > params.darkMin;

ball1 = blueThresh & darkThresh;
```

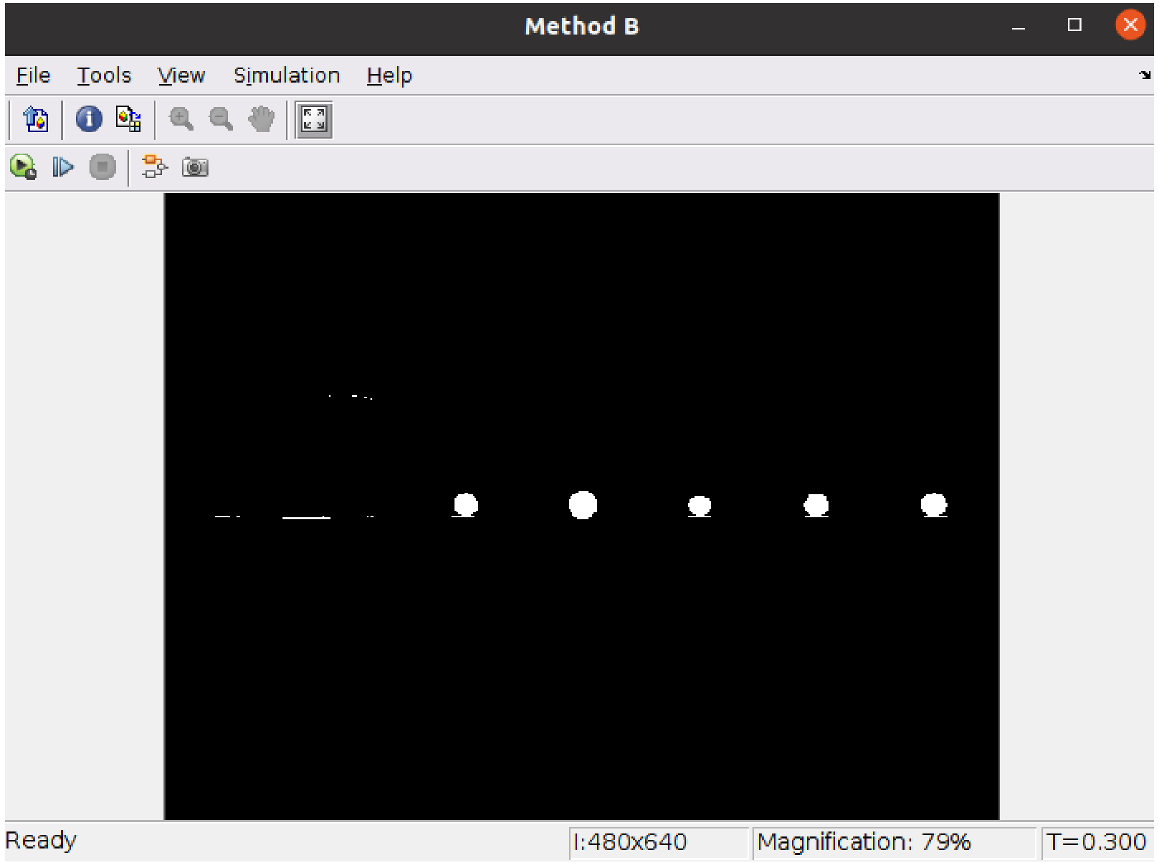
Isolate blue

Isolate dark

combine

Simple image processing based on colors

# Method B : MATLABexample



# result… ☹

1. *Slow*
2. *Not accurate*

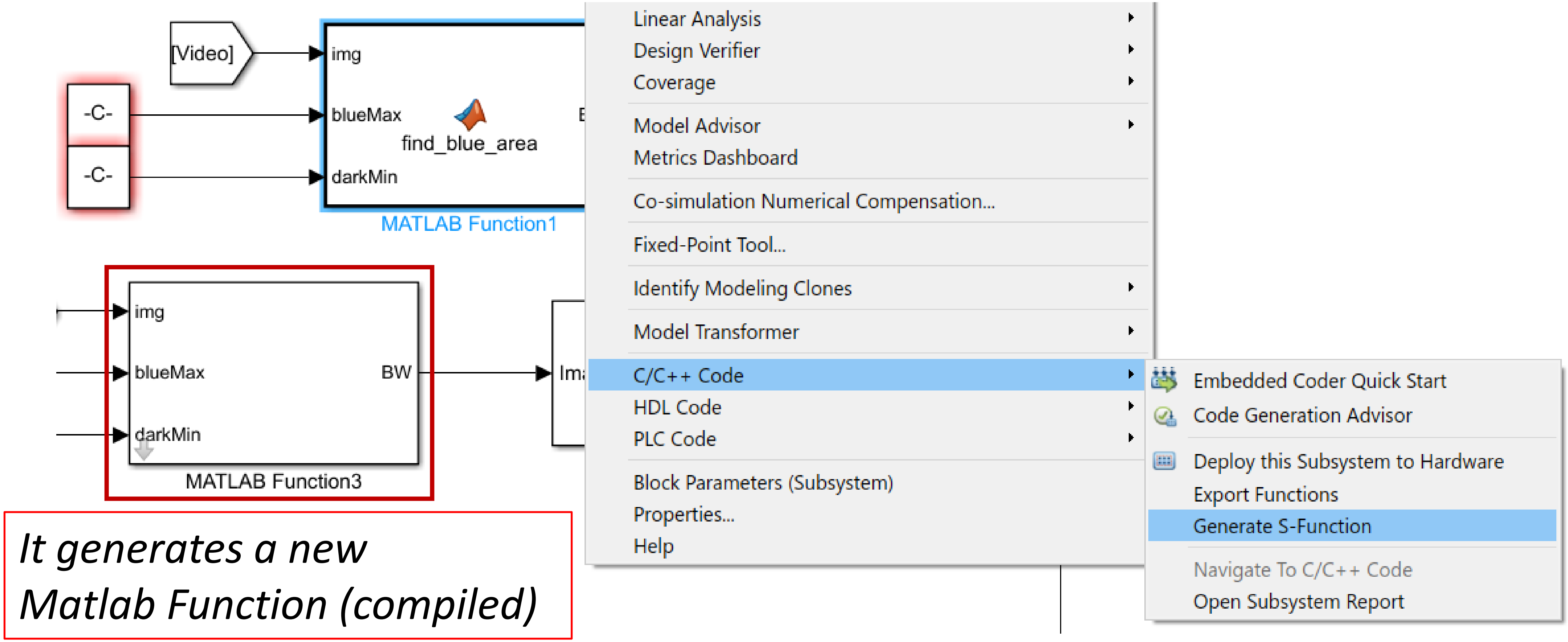
Simple image processing based on colors

# Method B_2 : MATLABexample_compiled

Let’s compile the code to speed up!
(using Simulink coder…)

Simple image processing based on colors

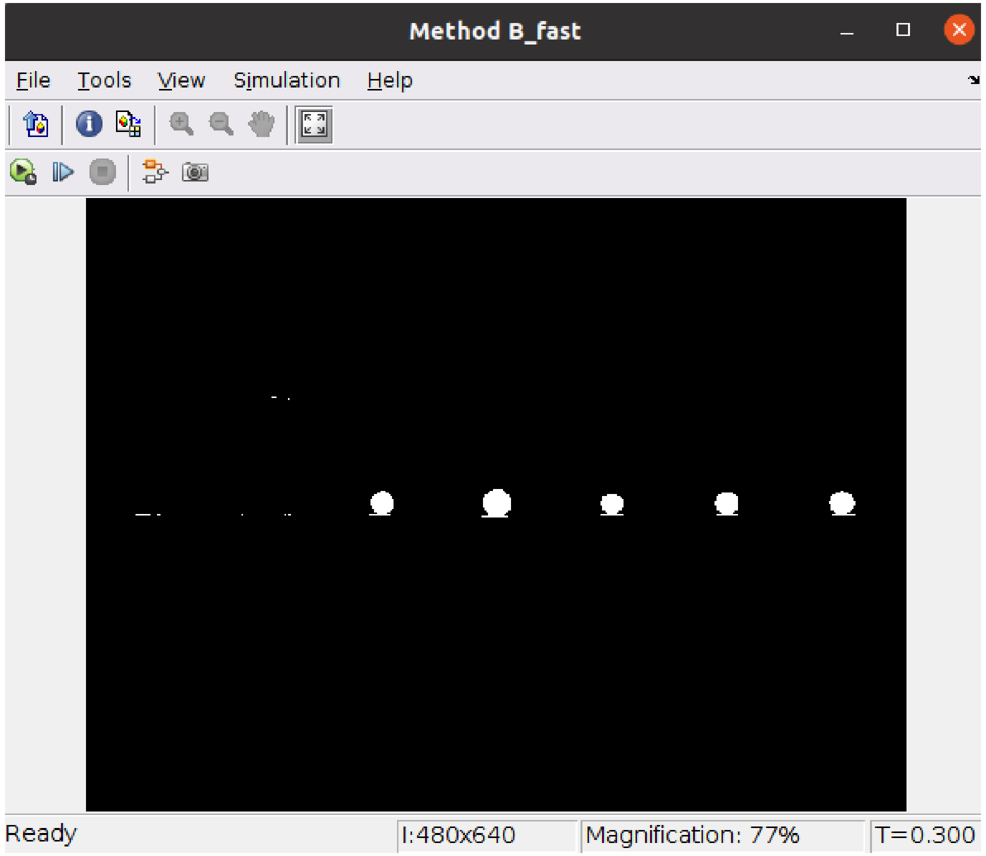
# Method B_2 : MATLABexample_compiled



It generates a new Matlab Function (compiled)

Simple image processing based on colors

# Method B_2 : MATLABexample_compiled



# result… ☹

1. ~~Slow~~
2. *Not accurate*

Simple image processing based on colors

# Method C : HSV approach

What is HSV?

Unlike RGB and CMYK, which use primary colors, HSV is closer to how humans perceive color.

It has three components: hue, saturation, and value.

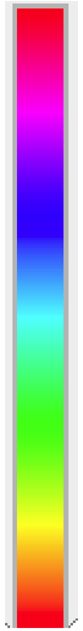
Simple image processing based on colors

# Method C : HSV approach

Hue

Hue is the color portion of the model, expressed as a number from 0 to 360 degrees:

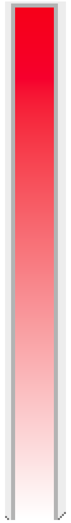
- **Red** falls between 0 and 60 degrees.
- **Yellow** falls between 61 and 120 degrees.
- **Green** falls between 121 and 180 degrees.
- **Cyan** falls between 181 and 240 degrees.
- **Blue** falls between 241 and 300 degrees.
- **Magenta** falls between 301 and 360 degrees.



Simple image processing based on colors

# Method C : HSV approach

Saturation



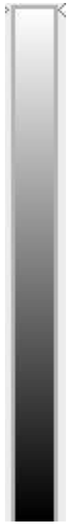
Saturation describes the amount of gray in a particular color, from 0 to 100 percent.

- **0** grey.
- **1** primary color.

Value

Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0 to 100 percent.

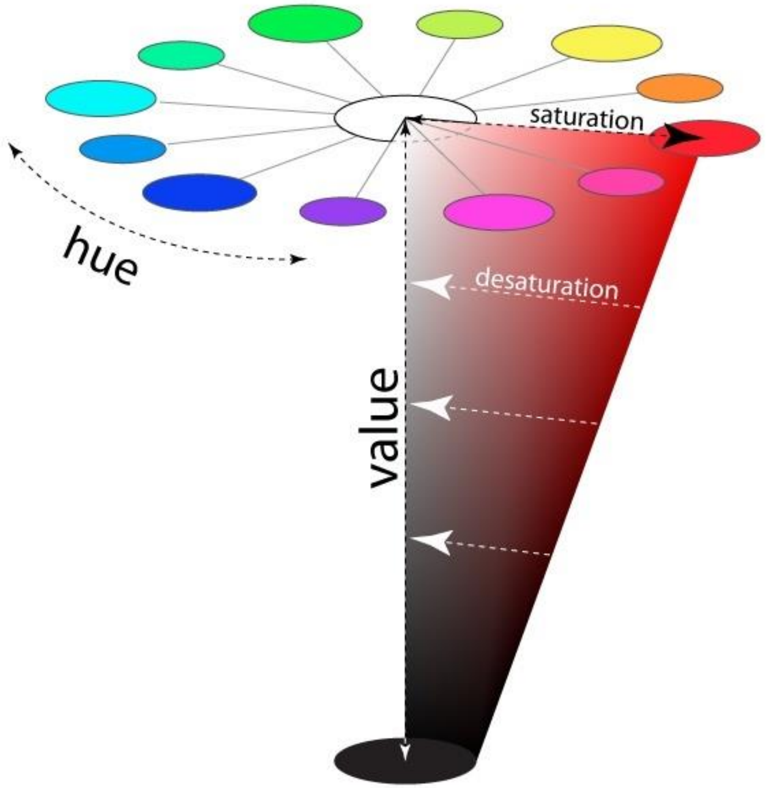
- **0** black.
- **1** primary color.



Simple image processing based on colors

# Method C : HSV approach

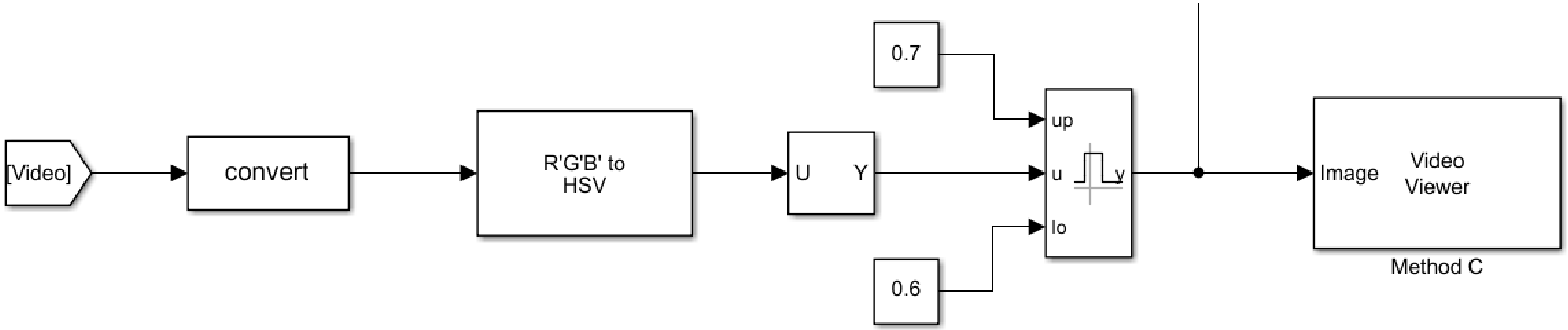
Sum Up:



Simple image processing based on colors

# Method C : HSV approach

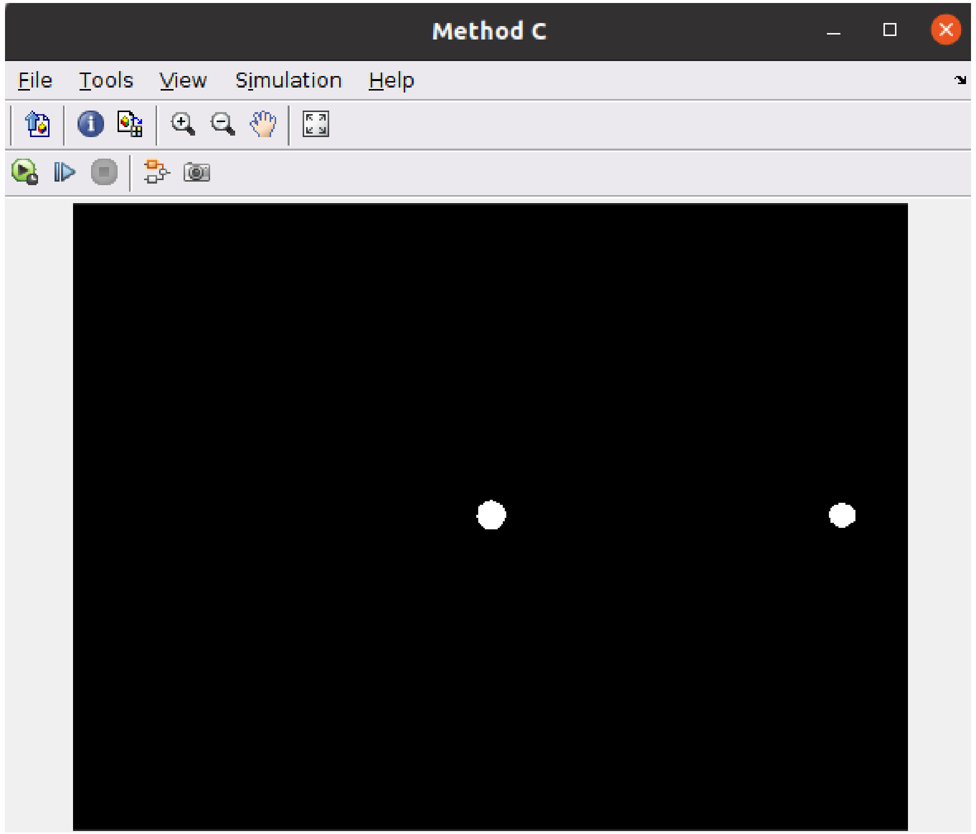
Implementation



Simple image processing based on colors

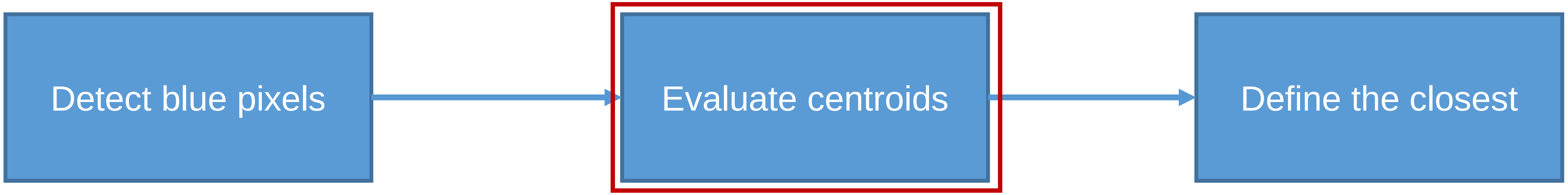
# Method C : HSV approach

# result… ☺

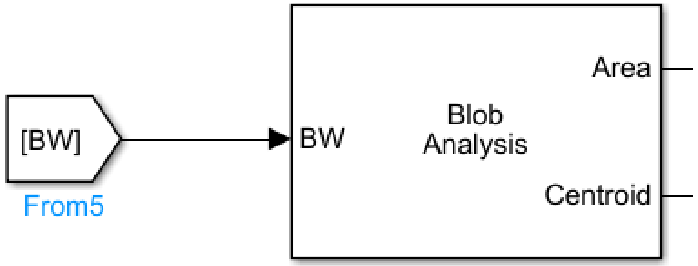


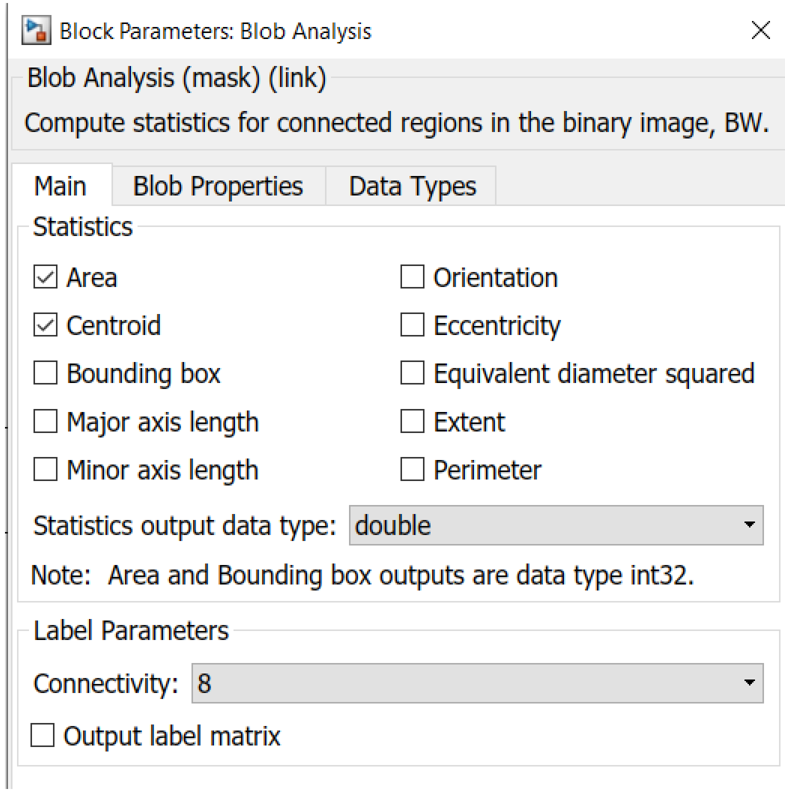
Simple image processing based on colors

#2: detect Objects in simulink



Simple image processing based on colors





Block Parameters: Blob Analysis

Blob Analysis (mask) (link)

Compute statistics for connected regions in the binary image, BW.

Main | Blob Properties | Data Types

Statistics

☑ Area

☑ Centroid

☐ Bounding box

☐ Major axis length

☐ Minor axis length

☐ Orientation

☐ Eccentricity

☐ Equivalent diameter squared

☐ Extent

☐ Perimeter

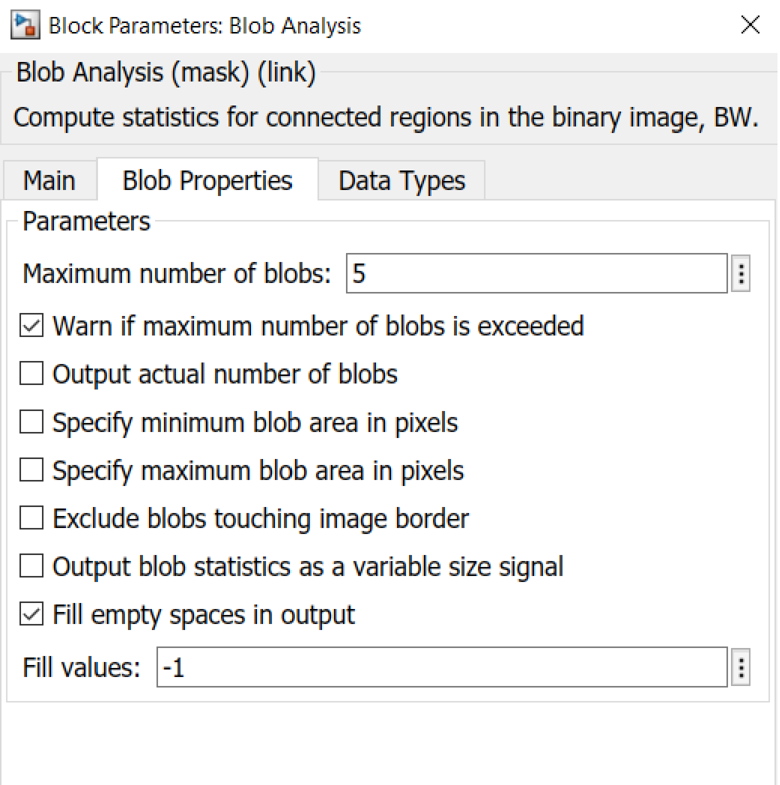
Statistics output data type: double

Note: Area and Bounding box outputs are data type int32.

Label Parameters

Connectivity: 8

☐ Output label matrix



Block Parameters: Blob Analysis

Blob Analysis (mask) (link)

Compute statistics for connected regions in the binary image, BW.

Main | Blob Properties | Data Types

Parameters

Maximum number of blobs: 5

☑ Warn if maximum number of blobs is exceeded

☐ Output actual number of blobs

☐ Specify minimum blob area in pixels

☐ Specify maximum blob area in pixels

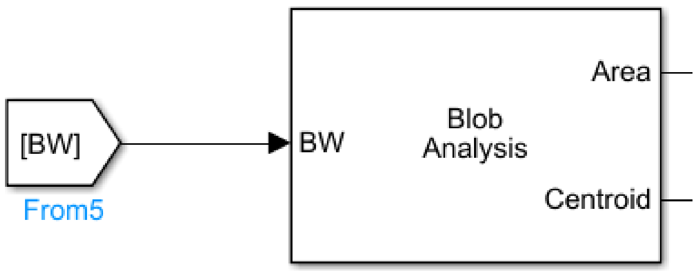
☐ Exclude blobs touching image border

☐ Output blob statistics as a variable size signal

☑ Fill empty spaces in output

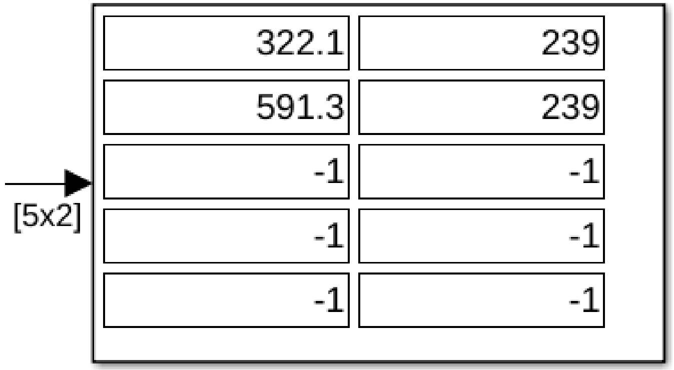
Fill values: -1

Simple image processing based on colors



- *I: BW matrix*
- *O: Area & Centroid coordinates*

# result… 😐

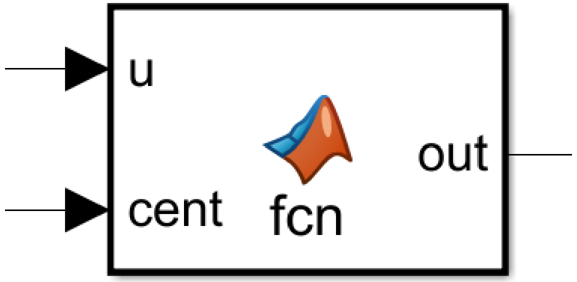


322.1	239
591.3	239
-1	-1
-1	-1
-1	-1

- *More than one!*

Simple image processing based on colors

#3: define the closest



```
function out = fcn(u,cent)

[~,i] = max(u);

if ~isempty(i)
        out=cent(i,:);
else
        out=[0,0];
end
```

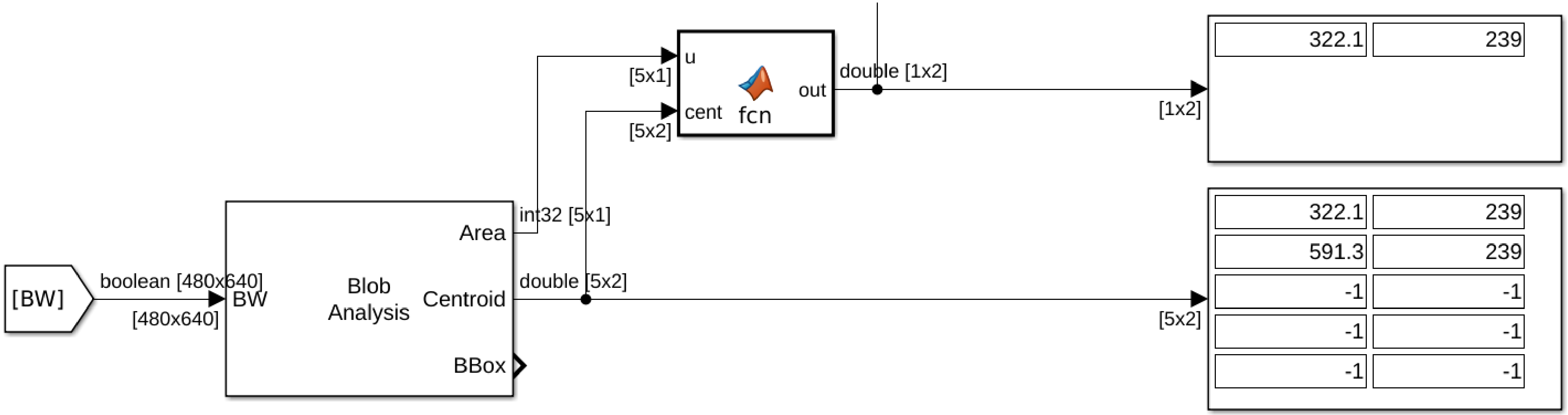
Find biggest (index)

Centroid coordinates, if any

Otherwise (0,0)

Simple image processing based on colors

#3: define the closest



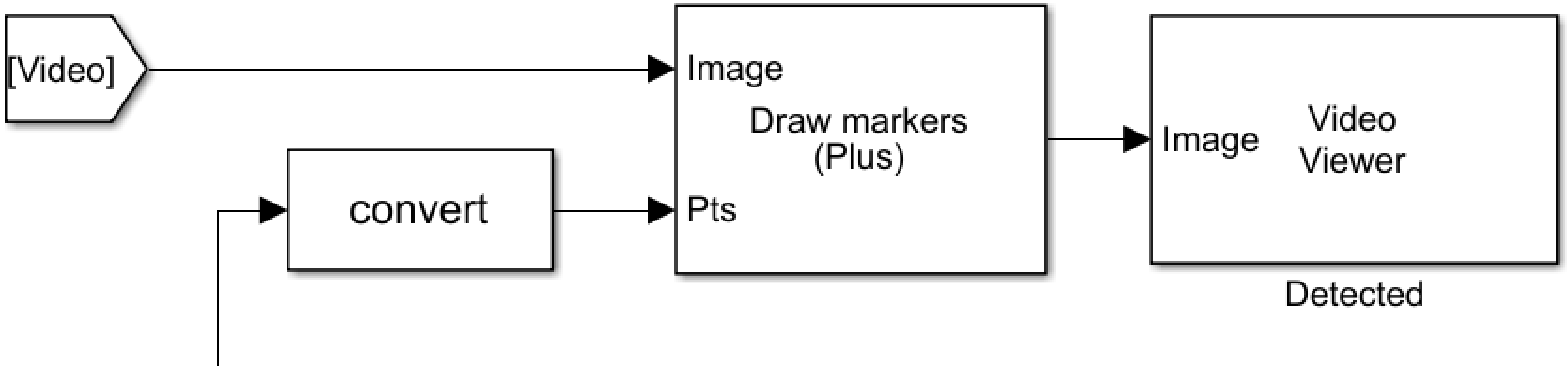
322.1	239

322.1	239
591.3	239
-1	-1
-1	-1
-1	-1

Simple image processing based on colors

#3: define the closest

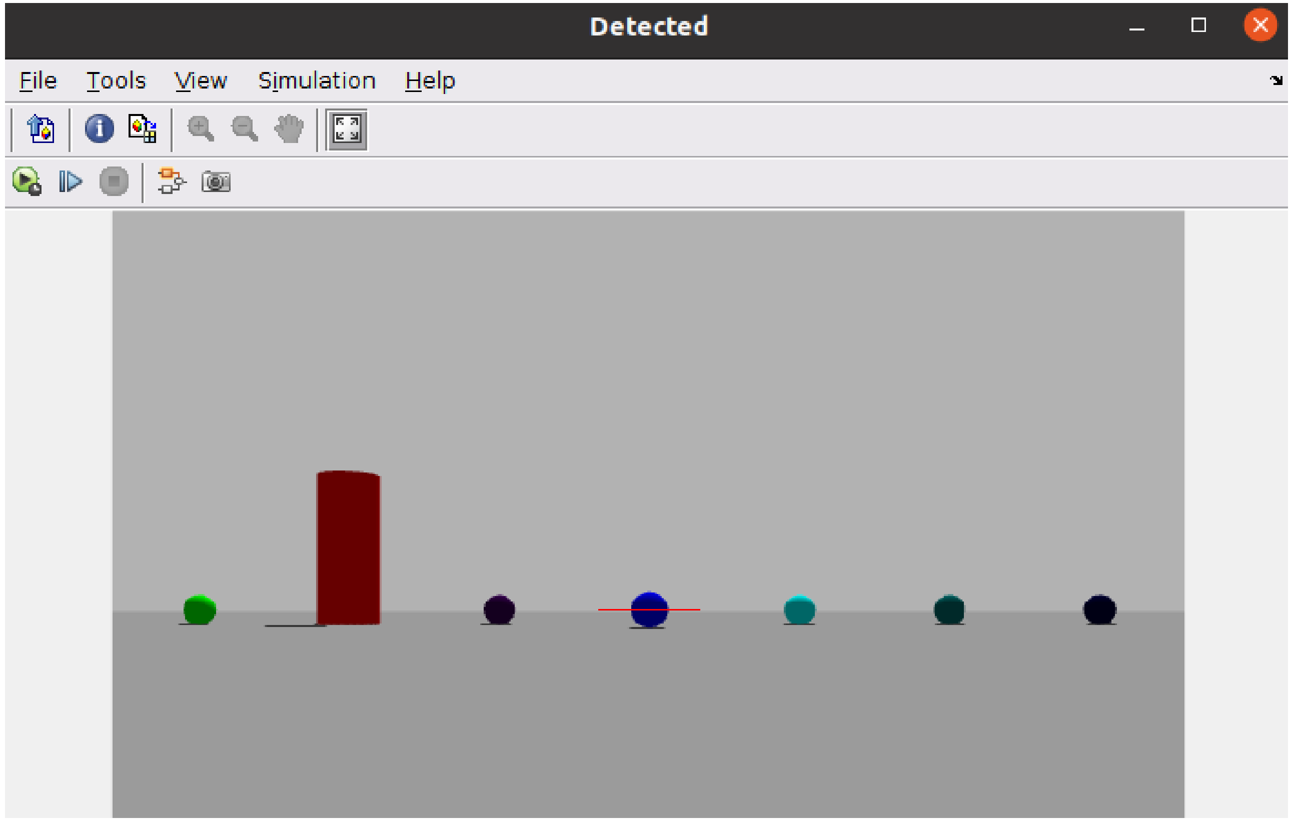
(add marks on video)



Simple image processing based on colors

#3: define the closest

# result… ☺



ROS

Assignment III

What we expect from you

Starting from initial position of the robot:

- Define a “red sphere” (#c83030) of 0.1m radius 6 m on the left
- Define a “purple sphere” (#c80067) of 0.2m radius 6 m in front
- Define a ***feedback control*** in order to move the #robot from initial position to collide with “red sphere”.

List of suggested steps

- Create a simulink block:
  - Subscribe to #robot position (/odom)
  - Subscribe to #Camera
  - Detect «red sphere»
  - Compute control command to go crash to the sphere

Results

- What is mandatory for the report?

  - Plot of #robot trajectory
  - Plot of centroid position

(save a .bag with #robot position, control command, centroid position)

That’s it for today…

See you next time!

S. Arrigoni