Tutorial for the "Buoy Detection Using Simulink" Webinar

Table of Contents

1. Introduction

This document serves as a tutorial for users that have already seen the "Buoy Detection Using Simulink" webinar and are looking to use the code that was discussed in the webinar.

2. Download Folder

Download the folder "Buoy_Detection_Using_Simulink.zip" and extract

3. Raw and Buoy Detection Videos

All of the media can be found in the "Buoy_Detection_Using_Simulink\media" folder.

The raw video from a boat which navigates a buoy course is in this folder an. The resolution of this video is 640x480. To obtain it in other resolutions, the function in the media folder called "convertVideo" can be used. The syntax for this function is;

```
>>

convertVideo(resolution)
```

Where resolution is either the string '320x240' or '180x120'.

4. Intro to Buoy Detection Model

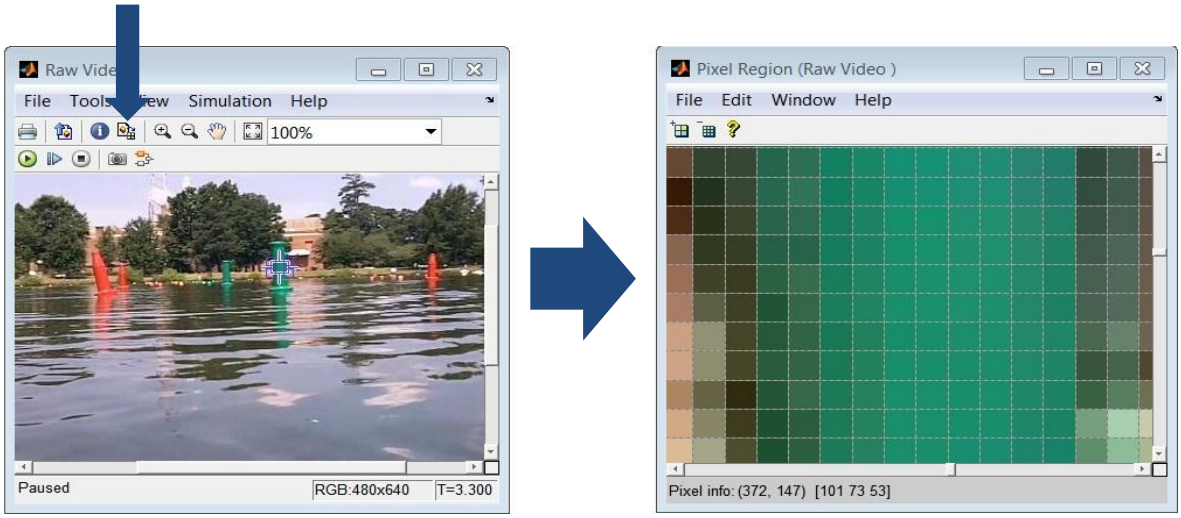
The introductory buoy detection model that was used in the "Implementing Buoy Detection" section was "target_transmit_singleThreshold.slx".

To run the model in simulation, execute the following steps;

1. Open the model "target_transmit_singleThreshold.slx"
2. Click the run button

5. Manual Tuning using Simulink UI - Pixel Viewer

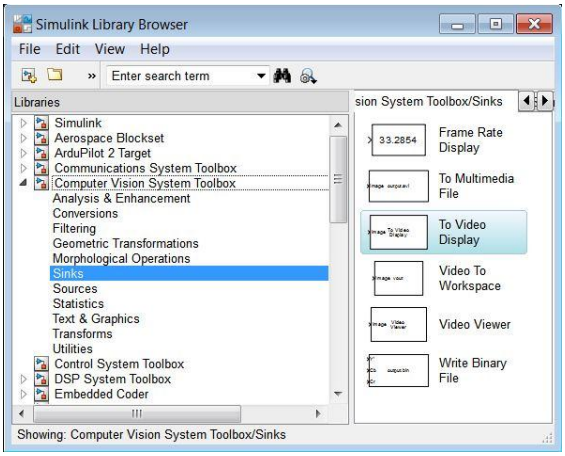
To manually tune thresholds using the Pixel Viewer, click the pixel viewer button shown here;



You can then manipulate the window that is overlaid onto the image by moving it or dragging the corners to enlarge it. After the window has been appropriately placed, dragging the cursor over each pixel in the viewer returns pixel info at the bottom left of the window in the format [x y] [R G B] if the video is in RGB format.

6. Manual Tuning using Custom App – ColorThreshold

To use the ColorThreshold App, you need to execute the function ColorThreshold(img) where img is a raw image that you would like to threshold. To obtain a raw image from a To Video Display block, run the model, pause it, click File->Export to Image Tool. When the Image Tool pops up, click File->Export to Workspace. Set the variable name for the screenshot image and click okay. Note that this workflow only exists for the “Video Viewer”. The “To Video Display” block provides similar functionality but is simpler.



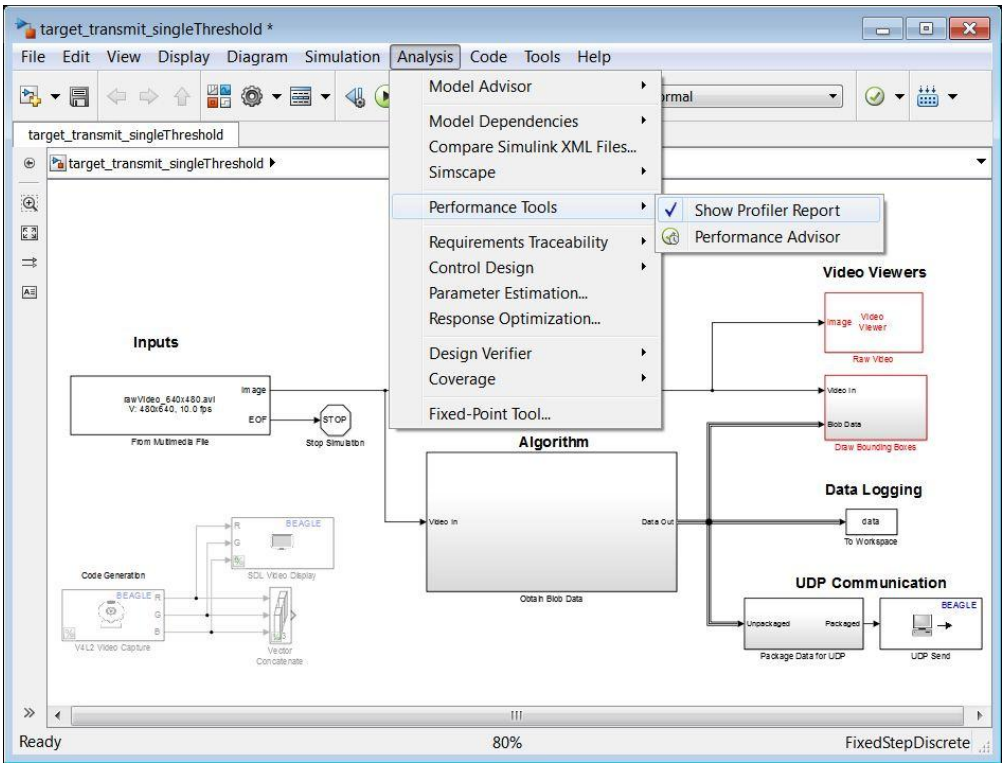
After obtaining the image, call the function ColorThreshold(img) where img is the variable that you saved.

The ColorThreshold app is not included in this submission. It can be found in the MATLAB File Exchange here;

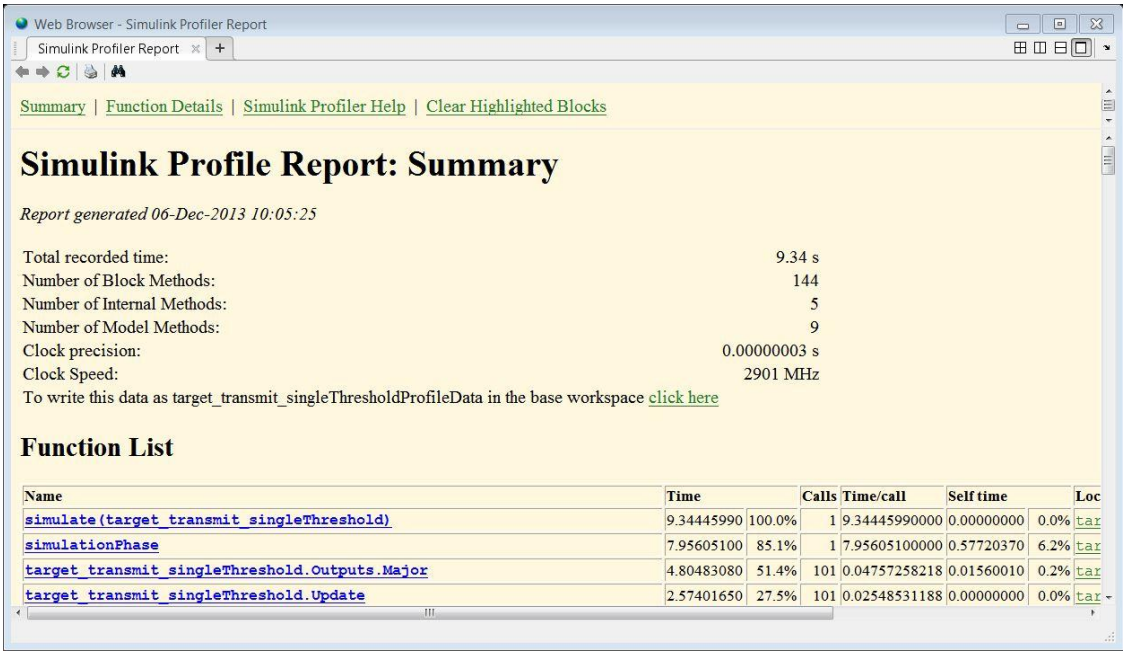
http://www.mathworks.com/matlabcentral/fileexchange/25682-color-threshold

7. Profiler

To enable the profiler, navigate to the following in the Simulink Explorer, Analysis->Performance Tools->Show Profiler Report. When a simulation stops, the resulting Profile Report is opened.



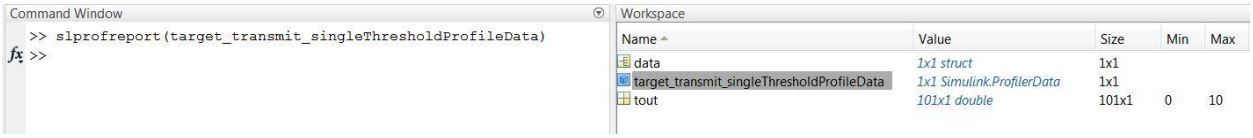
To save a Profiler Report to the base workspace, click on the “click here” button indicated in the following screenshot;



This report can then be extracted from the Simulink.ProfilerData object by using the SLPROFREPORT command as follows;

>> slprofreport(reportname)

The report name is usually of the format “modelnameReportData”.



8. Extracting Metrics – postSimulationAnalysis.m

To extract the metrics from an algorithm, we need the blob data from the best algorithm and the algorithm we would like to evaluate. The blob “data” is a structure which contains a time series for every recorded bob variable. The blob variables recorded are;

- green_area
- green_centroid
- green_bbox
- red_area
- red_centroid
- red_bbox

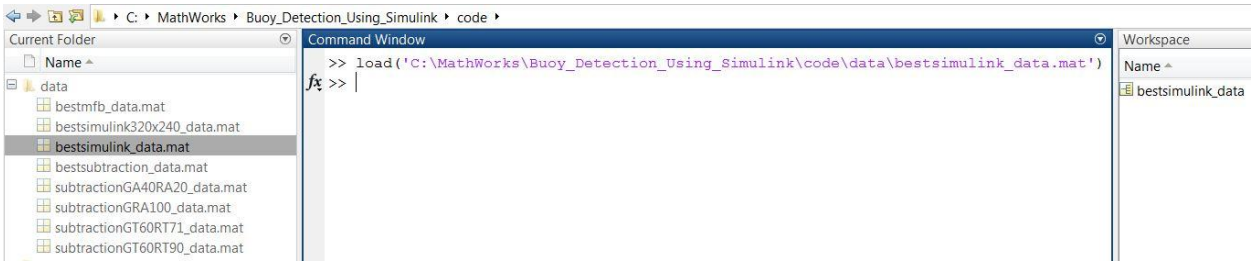
8.1. Obtain “best” and “algorithm” data

To use the “Best Simulink” algorithm “data” structure that has already been saved for the “rawVideo_640x480” file you can either;

1. Method 1: First make sure that your Current Working Directory is;
   \Buoy_Detection_Using_Simulink\code\.

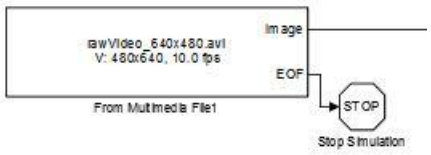
   Then execute the following command;
   >> load data/bestsimulink_data
2. Method 2: Drag the “bestsimulink_data.mat” file from the “Current Folder” window into the “Command Window”

To use the “Best Simulink” algorithm “data” structure that has been saved for the “rawVideo_320x240” file, just use the “data/bestsimulink320x240_data.mat” file instead.

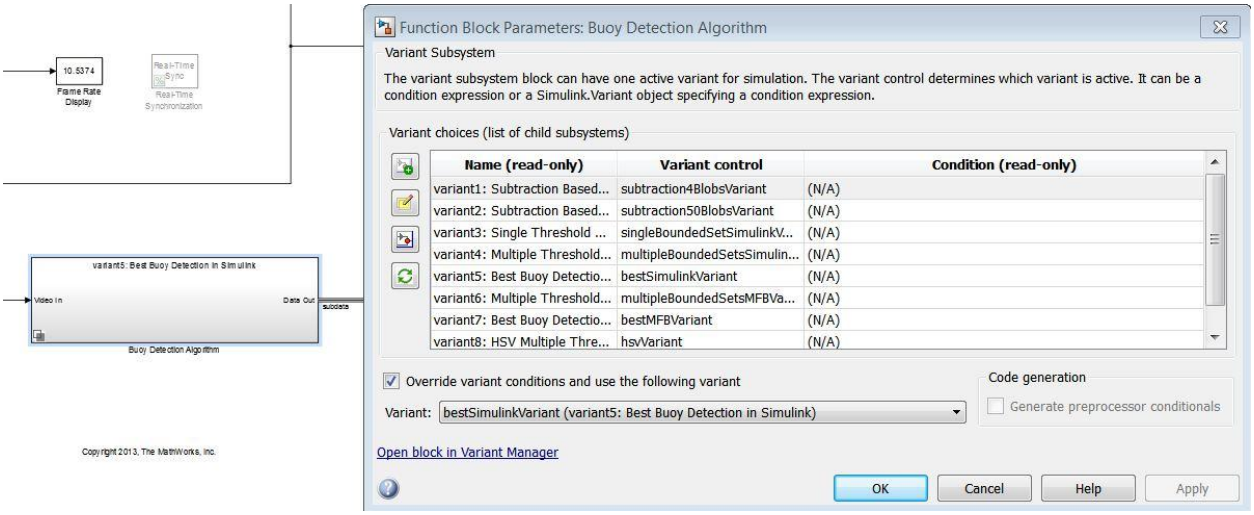


To create the data structure from the “Best” algorithm;

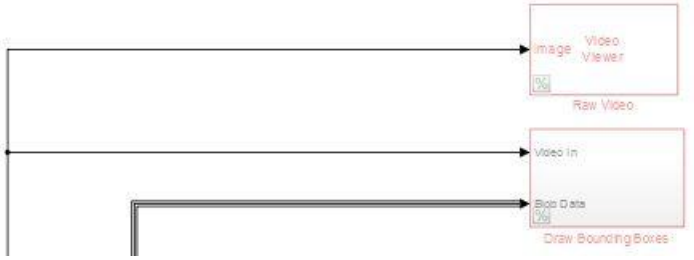
1. Open the target_transmit model
2. Make sure that the “From Multimedia File” block imports the “rawVideo_640x480.avi” file.



3. Set the algorithm variant to what you believe the “Best” algorithm should be using the “Buoy Detection Algorithm” Variant Subsystem’s dropdown. To access this dropdown, right-click on the “Buoy Detection Algorithm” subsystem, click “Block Parameters (Subsystem)” from the context menu, make sure that “Override variant conditions and use the following variant” is checked, then click on the variant dropdown and select the appropriate variant algorithm. I used “Best Simulink” which was variant5 and had the “bestSimulinkVariant” control variable.



4. Comment out the “Raw Video” Video Viewer and “Draw Bounding Boxes” subsystem to increase the speed of the simulation (if desired). To comment out a block, left-click on it and use the keyboard shortcut “Ctrl+shift+x” or right-click on it and select the “Comment” context menu item.



5. Make sure that the “To Workspace for blob data” block and its corresponding “Convert Bus to Double” subsystem are uncommented.
6. Run simulation for “inf”. Simulation will stop when video has stopped executing. When the simulation stops, the variable “data” from the “To Workspace for blob data” block will stored in the MATLAB Workspace. This “data” variable is the “best” data set.
7. You can assign the best data to a different name and save it to a .mat file by using the following MATLAB Commands;

   ```
   >> bestdata = data;
      save bestdata bestdata
   ```

To obtain the “algorithm data” either;

- Method 1: load a pre-saved .mat file i.e. “bestsubtraction_data.mat”
- Method 2: follow the same steps 1-7, which the exception that you should choose an algorithm other than “bestsimulink_data” if it was used as your “best” data.

8.2. Run postSimulationAnalysis

If you have the “best” and “algorithm” data sets in the MATLAB Workspace, you can compare the two by using the postSimulationAnalysis function. The function is called using the following syntax;

metrics = postSimulationAnalysis(bestdata,algdata)

metrics = postSimulationAnalysis(bestdata,algdata,flag1,flag2,…)

The flag can be one or a combination of the following strings 'plots','video','record'

i.e. metrics = postSimulationAnalysis(bestdata,data,'video') to show video

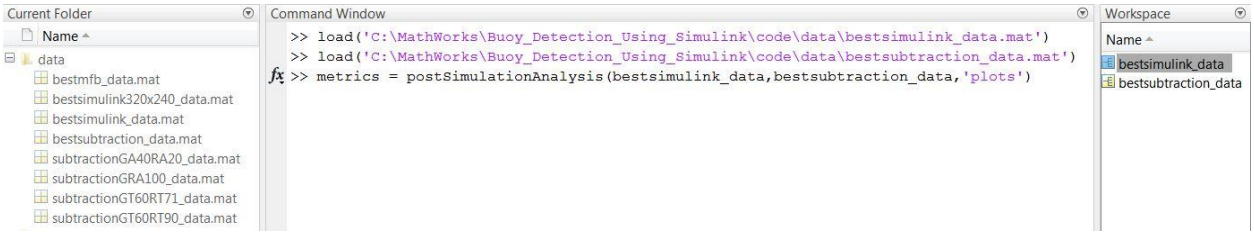
Flag explanation;

The flag ‘plots’ displays the “# Green/Red Algorithm Blobs per Closest Best Buoy” and whether the “Closest Best Buoy” has been missed by the Algorithm.

The flag ‘video’ displays the raw video with bounding boxes overlaid onto it.

The flag ‘record’ records the bounding boxes video to .avi format.

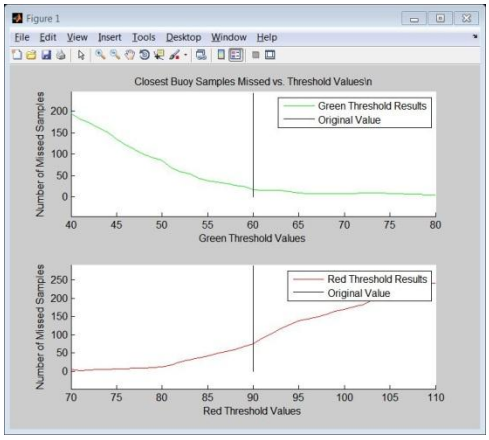
The full documentation for the function is located in the comments at the top of the file. Please see the documentation for more information.



The metrics output contains the following metrics;

- nClosestMissing – Number of closest best buoys that have been missed by the algorithm
- nAvgBlobsPerClosest – Average of the number of algorithm blobs that are used to identify a closest best buoy
- bestAvgAreaClosest – Average of the area of the closest best buoys
- algAvgAreaClosest - Average of the area of the algorithm blobs that are used to identify a closest best buoy

9. Sweeping Through Parameters – runParameterSweep



Running many simulations with varying parameters can serve many purposes. The two primary reasons to do so are;

1. To ensure that algorithms are robust against parameters differences
2. To determine the optimal algorithm parameters

Method 1 was used in the webinar when we applied gamma correction to brighten or darken an image to simulate sunlight. This allowed us to determine which algorithms would be more robust against lighting changes. Method 2 was used to determine the optimal gains for a subtraction algorithm.

The function “runParameterSweep” sweeps through a model (currently “target_transmit_subtraction”) with different threshold values To call the function, simply execute the following command at the MATLAB Command Window;

```
>>

output = runParameterSweep
```

The "output" is a structure containing the data and metrics at every time step. The elements output.data and output.metrics are nx1 vectors where n are the number of thresholds being swept through. The output also contains a table of data called "nclosestmissing_table" which can be used to plot the “Thresholds vs. Number of Missing Closest Best Buoys"

10. Creating an Algorithm Comparison Spreadsheet-runVariantSpreadsheet

	Video		Target Execution Time Metrics			Blob Analysis Metrics					
			PC	BeagleBoard							
				Standalone	External	# Closest Blobs Missed		Average Algorithm Blobs		Average Area Difference of Closest Blob	
Algorithms	Resolution	FPS	t (sec)	Lag (sec)	FPS	Green	Red	Green	Red	Green Difference	Red Difference
Subtraction 4 Blobs	640x480	10	29.7	0	6.76	3	2	1	1.25	-242	-93
Subtraction 50 Blobs	640x480	10	32.8	0.25	6.35	1	2	1.02	1.01	-302	-244
Single Bounded Set Simulink	640x480	10	49	1	3.3	25	88	1.32	1.21	177	745
Multiple Bounded Sets Simulink	640x480	10	73.2	1	4	25	7	1.32	0.9	177	2
Best Simulink	640x480	10	81.2	2	2.88	0	0	1	1	0	0
HSV Multiple Bounded Sets	640x480	10	97.3	4	1.16	1	2	1.01	1.03	-410	-183

Color Notes

	1*	2*	3*	4*	5*	6*	
1. T: Green if less than 128.5							
2. Lag: Green if <=0.25							
3. FPS: Green if >=7							
4. # Closest Missed: Green if <=10							
5. Average Algorthm Blobs: Green if >=0.9 and <=1.1							
6. Average Area Difference: Green if absolute magnitude <=500							

The function “runVariantSpreadsheet” executes the “target_transmit” model with the desired variant algorithms to output a spreadsheet which summarizes the metrics. It is configured upon downloading this, to compare the major 6 variant algorithms that are provided in the spreadsheet above.

This function calls a sub-function “runVariantSweep” which performs the sweep through the “target_transmit” model with the different variants to obtain a cell array of metrics. This cell array of metrics is used by the primary function “runVariantSpreadsheet” to build the spreadsheet.

The “runVariantSpreadsheet” function outputs a spreadsheet with the following metrics;

- Simulation time in Simulink on a PC
- Number of closest best buoys that have been missed by the algorithm
- Average of the number of algorithm blobs that are used to identify a closest best buoy
- Average of the area difference between the closest best buoys and the algorithm blobs that identify a closest best buoy

To call the function, simply execute the function’s name at the MATLAB Command Window;

```
>>

runVariantSpreadsheet
```

This uses the function “xlswrite” to build a spreadsheet. If you need better formatting, check out the tips section of the documentation for xlswrite. The “actxserver” function will need to be used instead.

11. Other Models

11.1. Target Transmit models

The primary model which I used to simulate, generate code, run parameter sweeps, run variant sweeps is the “target_transmit” model.

For demonstration purposes, the “target_transmit_singleThreshold” and “target_transmit_subtraction” models were used because they were simplified versions of the main model. These two models are essentially the “target_transmit” model with just one variant algorithm.

In the presentation, the following models were used;

1. Implementing Buoy Detection
    a. Intro
        i. target_transmit_singleThreshold
    b. Manually Adjusting Thresholds
        i. target_transmit_singleThreshold
2. Analyzing Algorithms
    a. Profiler
        i. target_transmit_singleThreshold
    b. Extracting Metrics – postSimulationAnalysis
        i. target_transmit_subtraction
    c. Automated Tuning
        i. target_transmit
    d. Algorithm Summary
        i. target_transmit
    e. Gamma Correction
        i. target_transmit
3. Run Algorithm on Hardware
    a. target_transmit

11.2. displayAlgorithm

If you already have the metrics dataset “data” and would like to visualize the data in simulation without re-running the algorithm, you can use the “displayAlgorithm” model. This model simply imports the metrics “data” and overlaps the bounding boxes onto the raw video.

11.3. host_receive

The “host_receive” model runs on the host computer and receives UDP data from the “target_transmit” model. This is useful if you want to visualize the data without external mode. To test out this model, set up the UDP send block in the “target_transmit” model and UDP Receive block in the “host_receive” model. For each of the targets used (BeagleBoard-xM, PandaBoard and Raspberry Pi), the tutorial called “Communicating with <target>” can help you set up the appropriate values for the UDP Send and Receive blocks.

To use this model;

Deploy the “target_transmit” model to the target.

Run the “host_receive” model in simulation.

11.4. target_transmit_singleThreshold_matlab/MFB

In the “matlab code” folder, there are two files. The function “target_transmit_singleThreshold” performs buoy detection using MATLAB code only. There is nothing which manages the rate at which this thresholding is performed. One idea is to use a timer object to manage the rate at which the algorithm is performed. See documentation for “timer object” for more information.

The other file is a model called “target_transmit_singleThreshold_MFB”. This model performs RGB Single Bounded Set thresholding using a MATLAB Function Block (MFB). A MFB is a Simulink block in which MATLAB Code can be written.