University of Southern Denmark

Project in Real-Time Systems
Camera Vision

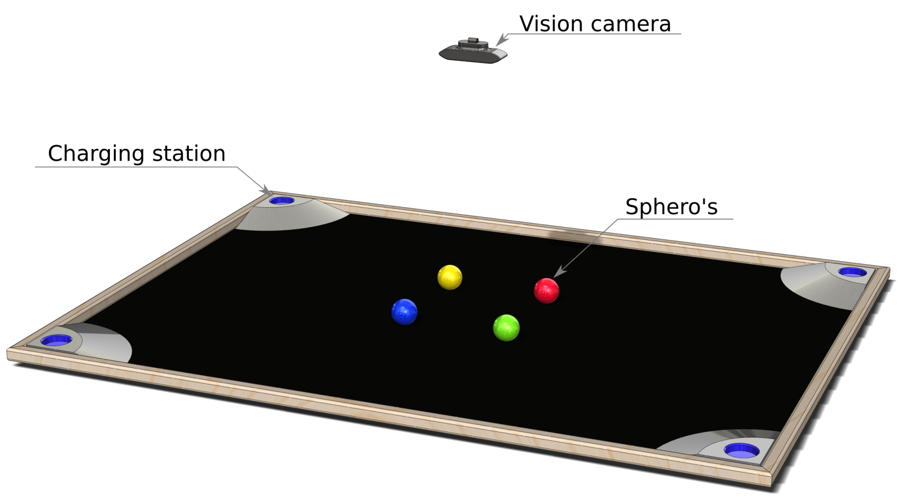
Report Author:
Kun Qian
Martin Hviid Kristensen

Course:
Project in Real-Time Systems
2017

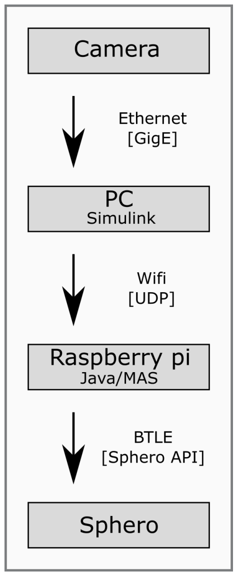
Contents

1 Project structure

The project in real-time systems was divided into different parts. The goal is to use a camera vision system for 'global' positioning of the Sphero robots. This data should then be send to a Raspberry pi running a Multi-Agent-System (MAS) which controls the Spheros via a Sphero c++ API.
This report will cover the camera vision aspect of the project. On Figure 1a the camera vision setup and the operating area of the Spheros is shown. A simplified diagram of the project structure and communication can be seen on Figure 1b.



(a) The operating area of the Spheros



(b) The project structure

Figure 1: The project setup

2 Camera vision system

The camera vision system will use a overhanging vision camera looking down on a black felt background. The Spheros will be tracked by the colour emitted from the internal LEDs. A figure of the positioning as seen from the camera is shown on 2. Here a custom Region-Of-Interest (ROI) of 800x1400[px] is used to achieve the wanted field of view.

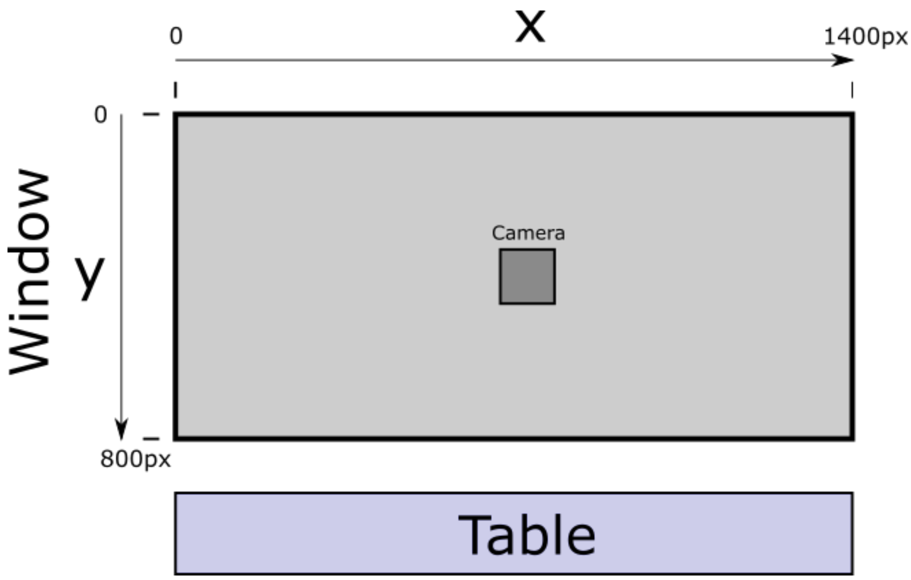


Figure 2: Positioning as seen from the camera

The camera used is a 'Allied Vision Mako G-192' sensor[3] with a 'Kowa LM5JCM' lens[4]. This device uses the Ethernet GigE vision interface standard[2] and it also implements Power-Over-Ethernet (POE).
Allied Vision supplies drives for Matlab for all their GigE and USB3.0 cameras. Matlab Simulink will be used to acquire the image from the camera, process the image to locate the Spheros and then broadcast the position data via the local wifi using UDP.

3 Vision camera and Matlab

3.1 Camera setup

To install all the needed software for the camera download and install the Allied Vision Vimba SDK with all extensions.[5] The Ethernet network adapter needs to be configures to achieve the best performance and to reduce the likelihood of dropped frames when using a GigE device. This is described in the Matlab Mathworks GigE setup guide.[1]
Once the above mention setup is complete the camera should be visible as a device in the Vimba Viewer program which can be used to preview the live image.

3.2 Matlab setup

To acquire the camera input in Matlab and Simulink the Matlab Add-on 'Image Acquisition Toolbox' needs to be installed.

3.3 Camera configuration

The configuration of the camera can now be done from Matlab using the 'Image Acquisition Toolbox'. Only a few camera parameters needs to be changed for tracking the Spheros and the rest are kept at default. These parameters are:

- **ROI and ROI offset** is set to 800x1400[px] and the offset is set to 100[px] and 110[px] for the x and y direction respectively.
- **FPS** is normally set to 60
- **Exposure** is set to 10000
- **Brightness Gain** is set to 0

The parameters can be set both from Matlab and from Simulink. The focus is set manually on the camera lens.

4 Simulink

As shown in figure 3, there is RGB image coming from Video Input. Usually, as recommended, we would better convert RGB images to HSV images because the R, G, and B components of an object's color in a digital image are all correlated with the amount of light hitting the object, and therefore with each other, image descriptions in terms of those components make object discrimination difficult. Descriptions in terms of hue/saturation/value are often more relevant. Then with the thresholds applied, we can get black/white images, which can be analysed in Blob Analysis module to get the centroid of white part. Then we pack the centroids in a UDP package and broadcast it.

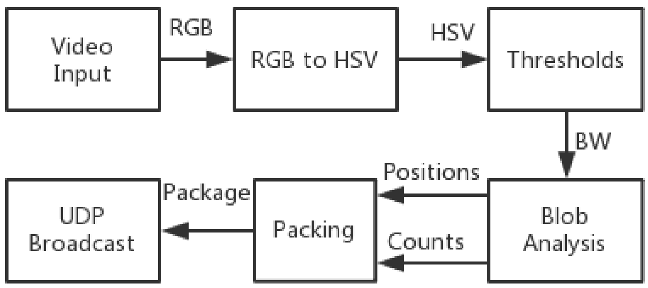


Figure 3: Structure diagram

4.1 Find the thresholds

Since the color of light is not ideally the same with what we expect, and the final image also changes using different camera setting, we made a MATLAB program for easily finding the threshold of a certain color of the light. To set the thresholds values, we can just move slide bars as shown in figure 4.

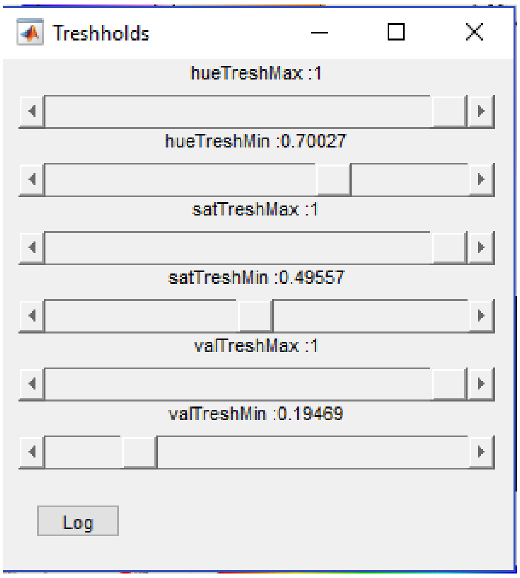


Figure 4: Set the thresholds

With the video views, we can see the result of choosing different thresholds as shown in figure 5 and 6.

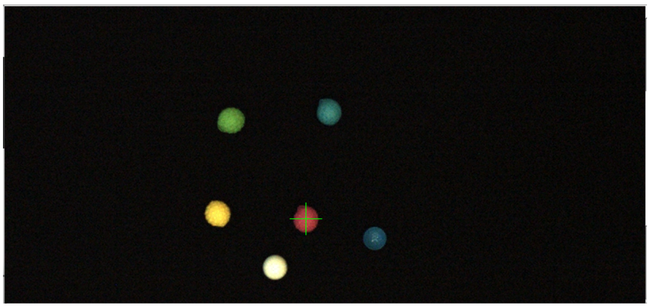


Figure 5: Original picture

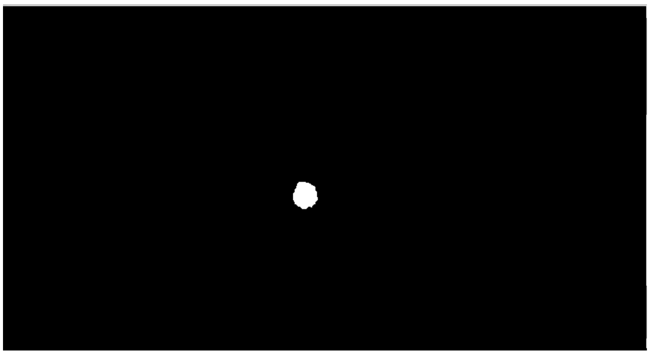


Figure 6: Thresholds applied

4.2 Final implementation

The setup of the camera is already described before. After applying the thresholds found before, we can get the black/white image. With Blob Analysis, the recognised white part in the image can be marked using the centroid position information. If there is no recognised white area, the Blob Analysis will set the counts as zero and also the positions to zeros. For sending out the position information, we need first pack them. The Packing module is actually a function of converting all the position information into a string. Then the UDP Broadcast module will send out the string over the local network. If the Blob Analysis count is zero, meaning it is not tracking a Sphero, the Packing module will send an error message insted of a position. The final implementation in Simulink is as figure 7.

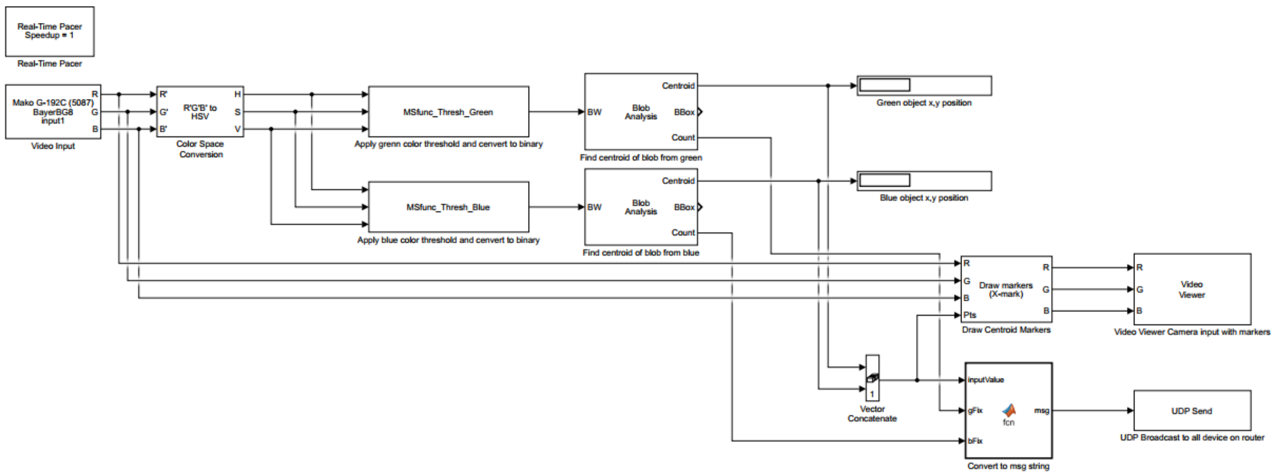


Figure 7: Final implementation

References

[1] *Gige vision quick start configuration guide*, https://www.mathworks.com/matlabcentral/answers/uploaded_files/41167/GigEVisionQuickStart.pdf, (Accessed on 12/30/2017),

[2] *Gige vision - wikipedia*, https://en.wikipedia.org/wiki/GigE_Vision, (Accessed on 12/29/2017),

[3] *Mako g-192, gige camera, teledyne e2v cmos sensor, power over ethernet (poe), 60 fps - allied vision*, https://www.alliedvision.com/en/products/cameras/detail/Mako%20G/G-192.html, (Accessed on 12/29/2017),

[4] *Me10*, https://www.alliedvision.com/fileadmin/content/documents/products/accessories/lenses/Kowa/Data_sheet/LM5JCM.pdf, (Accessed on 12/29/2017),

[5] *Vimba: The sdk for allied vision cameras - allied vision*, https://www.alliedvision.com/en/products/software.html, (Accessed on 12/30/2017),