Technische
Hochschule
Rosenheim

RoboSoccer – Design and development of an image processing system

Source: ssl.robocup.org

Project Supervisor : Professor Dietrich

Master project of the student : Siamak Mirifar

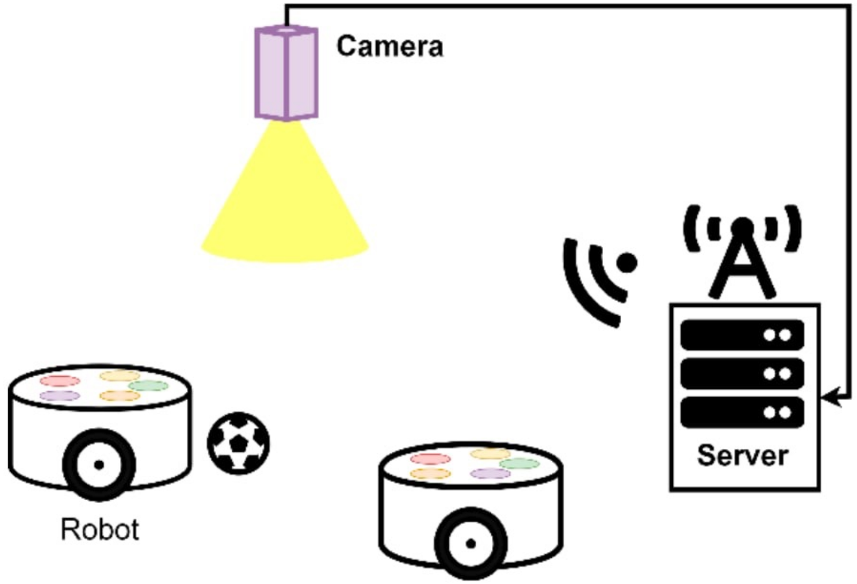
Master Ingenieurwissenschaften, Master Angewandte Forschung und Entwicklung

Sommer Semester 2022

Brief Description

1: Goal My Project

2: The Construction



Challenges of the project

- Video recording
- Image processing
- Robot pattern recognition
- Real-time data processing and application performance
- Server connection
- Graphical user interface (GUI)

Structure of the soccer field

Structure :

- Camera: Logitech Brio 4k
- Field size : 277*188 cm
- Background color : Dark green
- Material: Carpet
- Robot radius size : 85 cm

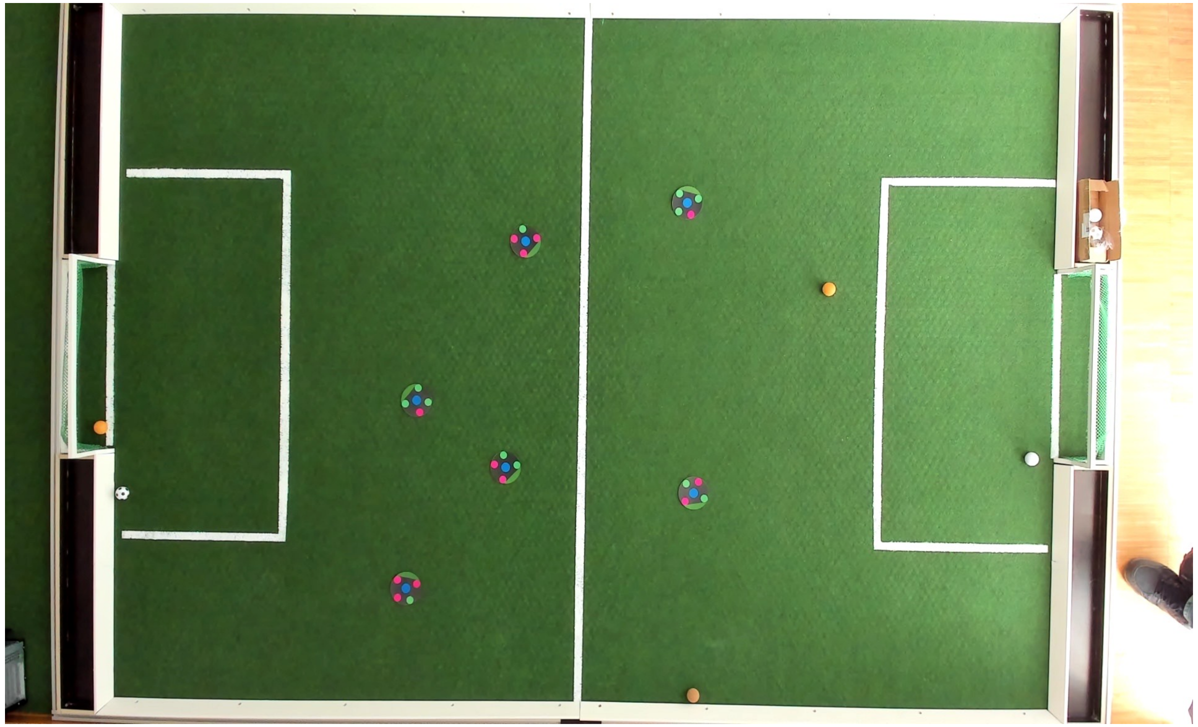


Image Processing

Possibilities of image processing :

- OpenCV
- Loading image
- Video recording

http://opencv.willowgarage.com/wiki/OpenCVLogo

Image Processing

Steps through the process :

- Reduce the image size
- **Blue color detection**
- Is the area of the blue circle is accepted ?
- Saving the pixel position

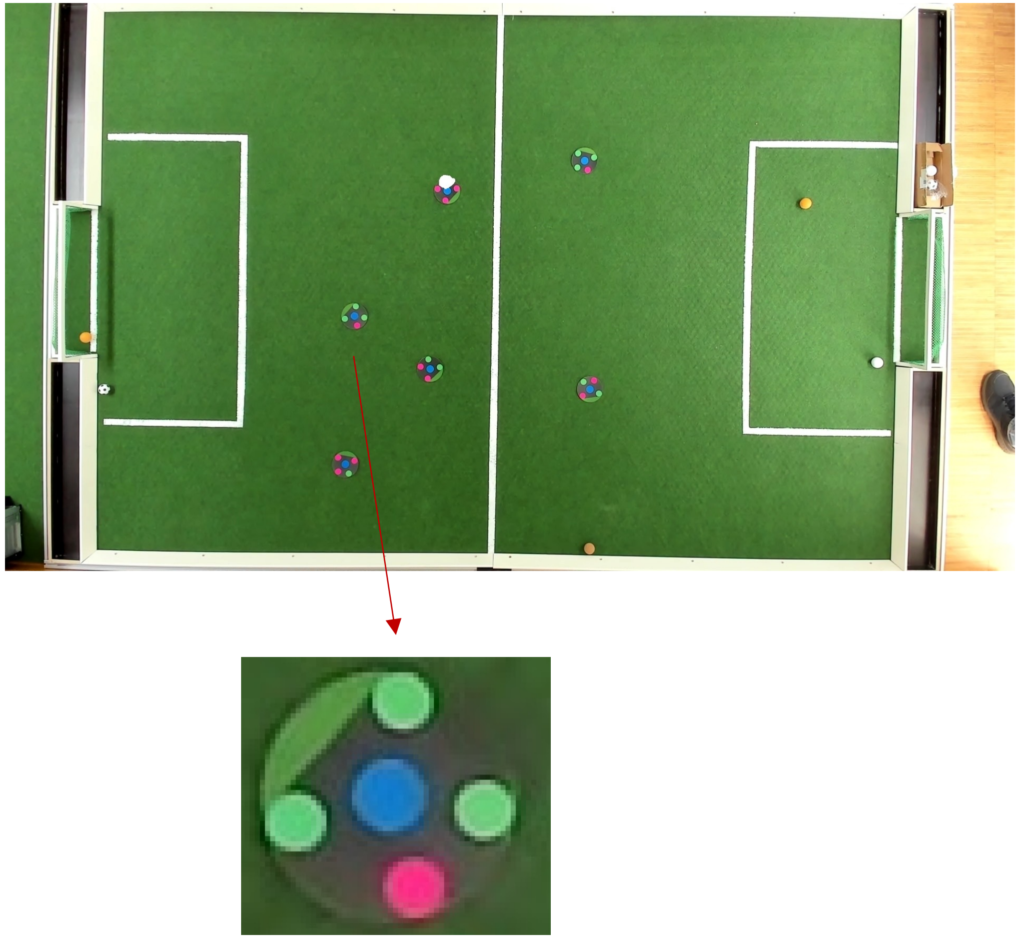
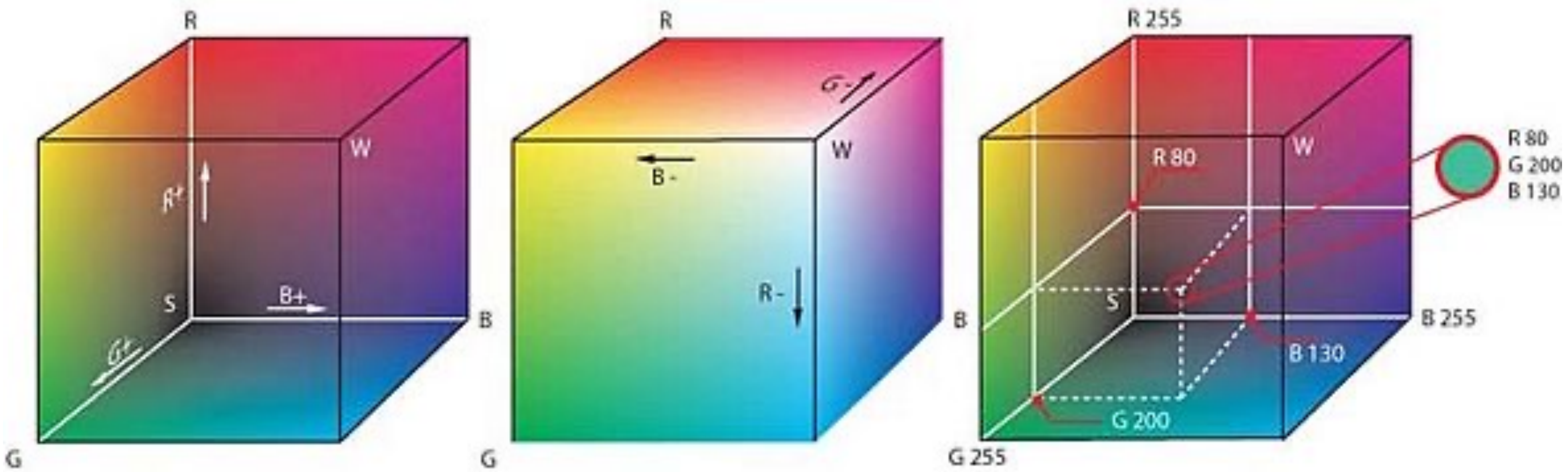


Image Processing

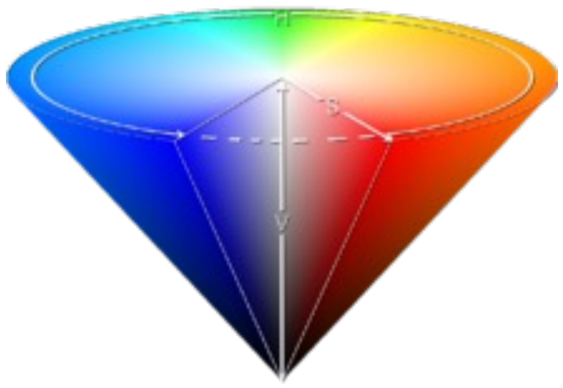
Color space:

The captured images are in RGB color space.

To detect the blue color, we need to use the HSV color space.



https://de.wikipedia.org/wiki/RGB-Farbraum

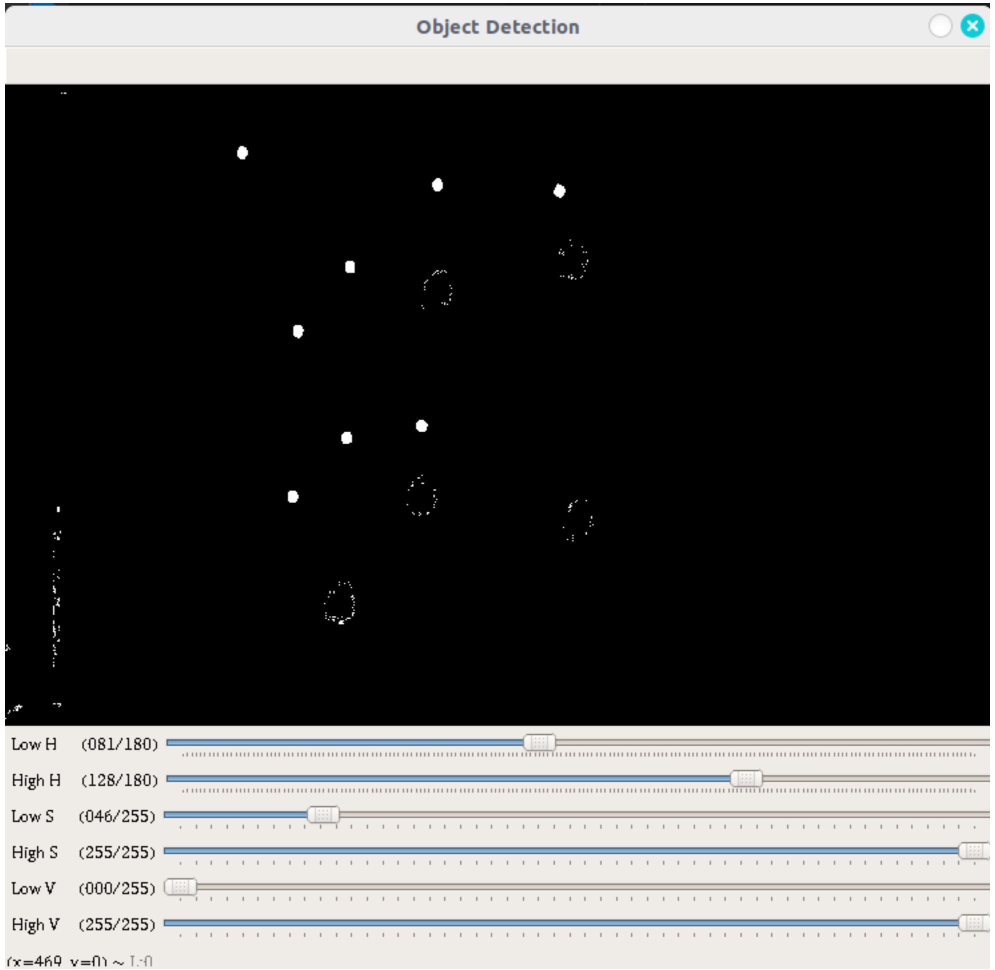


https://en.wikipedia.org/wiki/HSL_and_HSV

Image Processing

Mask :

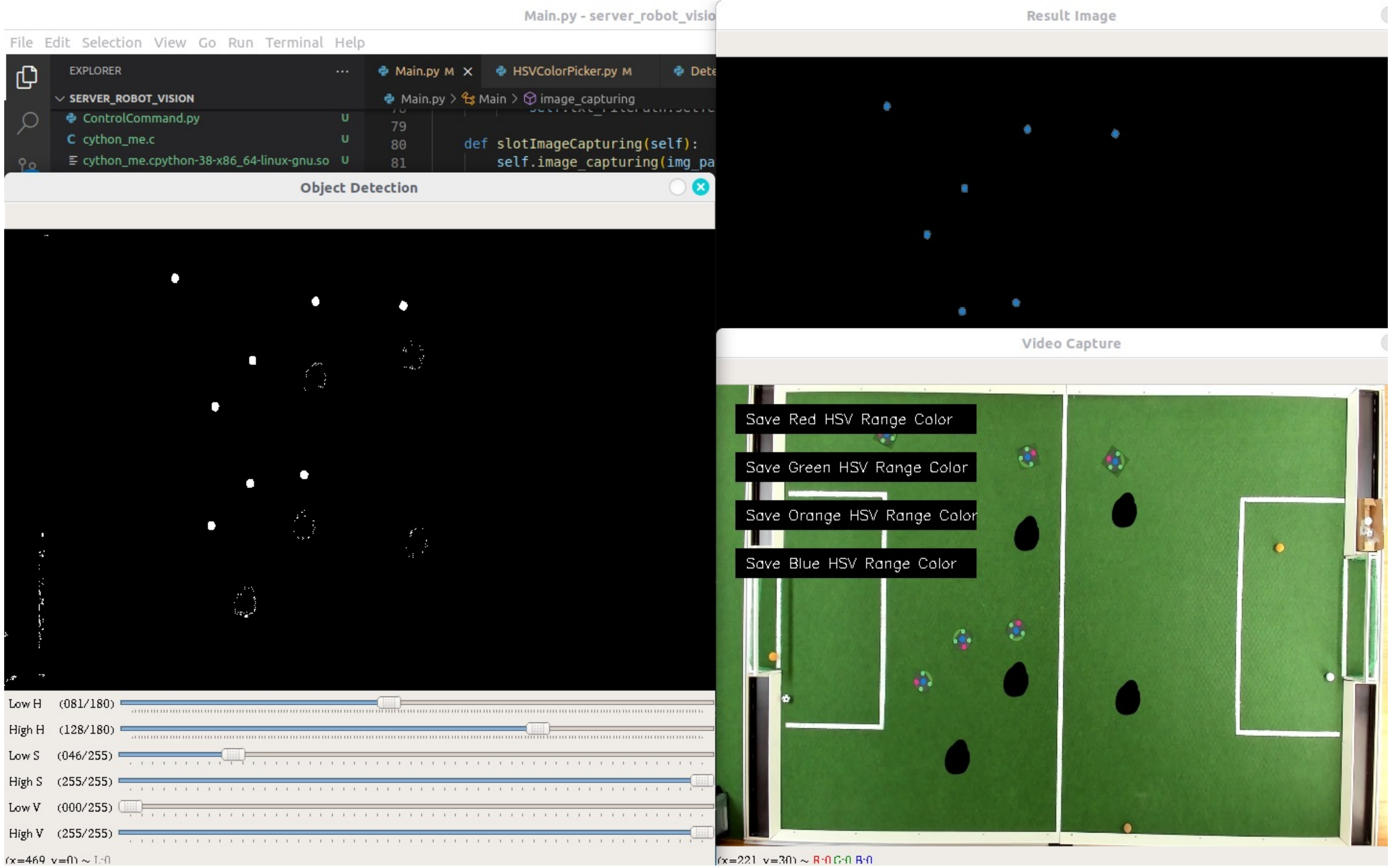
In OpenCV, by specifying the lower limits of the color and the upper limits of the color (blue in our case), we can disable all other color spaces.



```
Low H   (081/180)
High H  (128/180)
Low S   (046/255)
High S  (255/255)
Low V   (000/255)
High V  (255/255)
(x=469, y=0) ~ L:0
```

Image Processing

Mask :



Main.py - server_robot_visio
File Edit Selection View Go Run Terminal Help
EXPLORER
SERVER_ROBOT_VISION
ControlCommand.py U
cython_me.c U
cython_me.cpython-38-x86_64-linux-gnu.so U
Main.py M
HSVColorPicker.py M
Dete
Main.py > Main > image_capturing
Object Detection
Result Image
Video Capture
Save Red HSV Range Color
Save Green HSV Range Color
Save Orange HSV Range Color
Save Blue HSV Range Color

```
    def slotImageCapturing(self):
        self.image_capturing(img_pa
```

Low H (081/180)
High H (128/180)
Low S (046/255)
High S (255/255)
Low V (000/255)
High V (255/255)

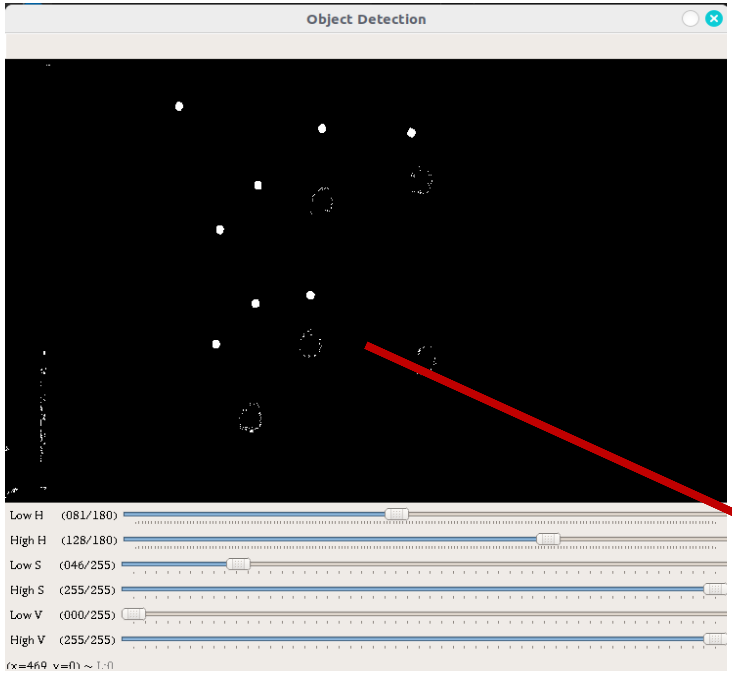
(x=469, y=0) ~ L:0

(x=221, y=30) ~ R:0 G:0 B:0

Image Processing

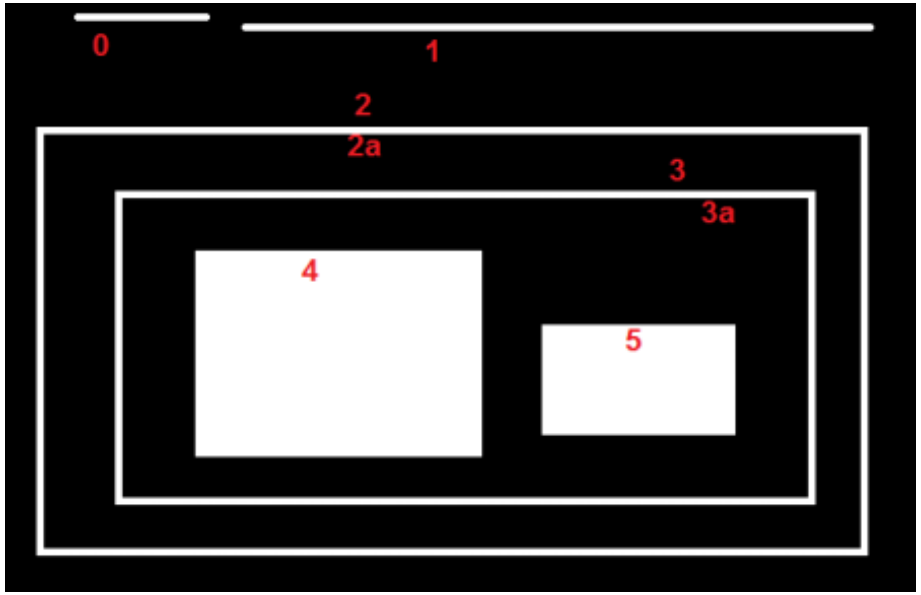
Contours :

Contours can be simply explained as a curve that connects all points constantly (in sequence) that have the same color or intensity (in our case, the color mask is blue).



Object Detection

```
Low H   (081/180)
High H  (128/180)
Low S   (046/255)
High S  (255/255)
Low V   (000/255)
High V  (255/255)
(x=469, y=0) ~ L:0
```



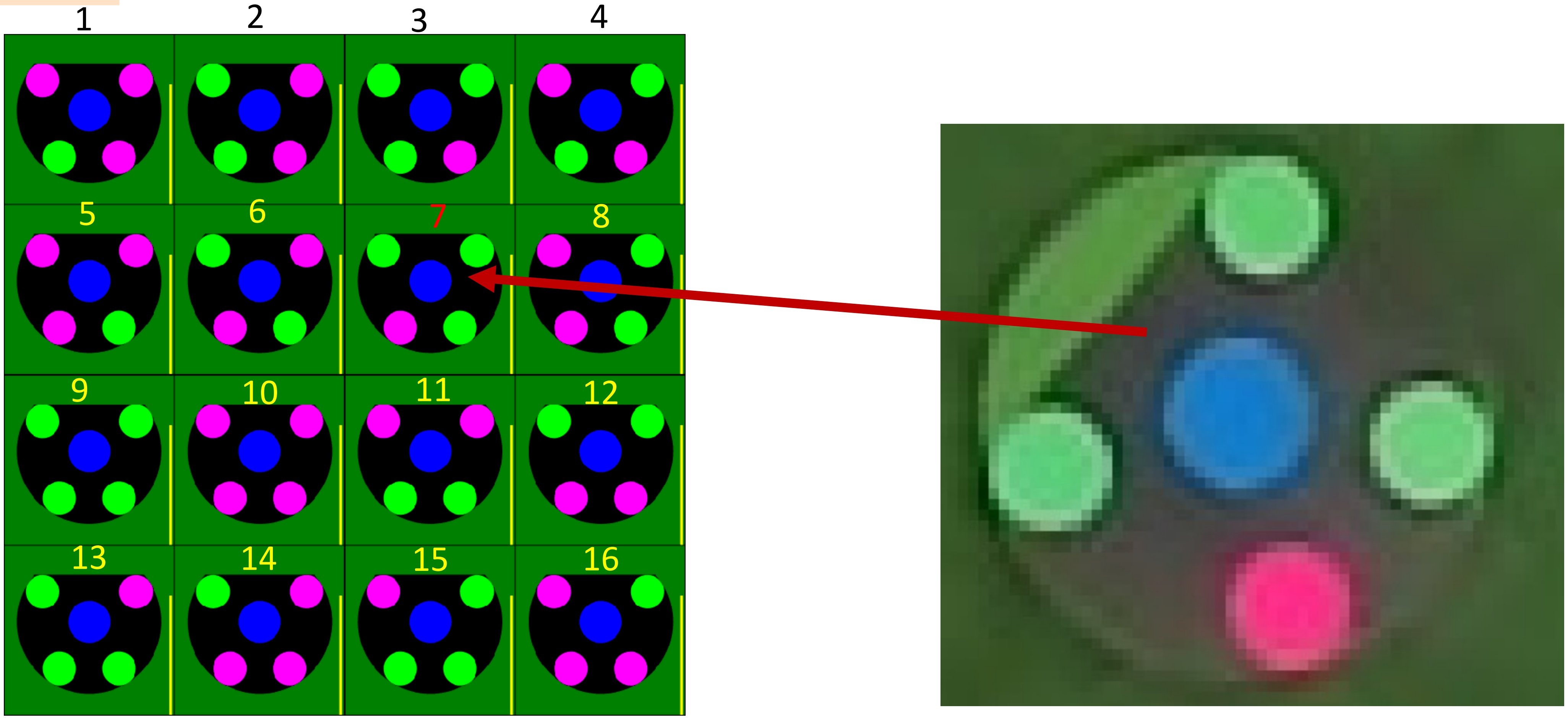
Areas that are not included

Image Processing

Crop image :

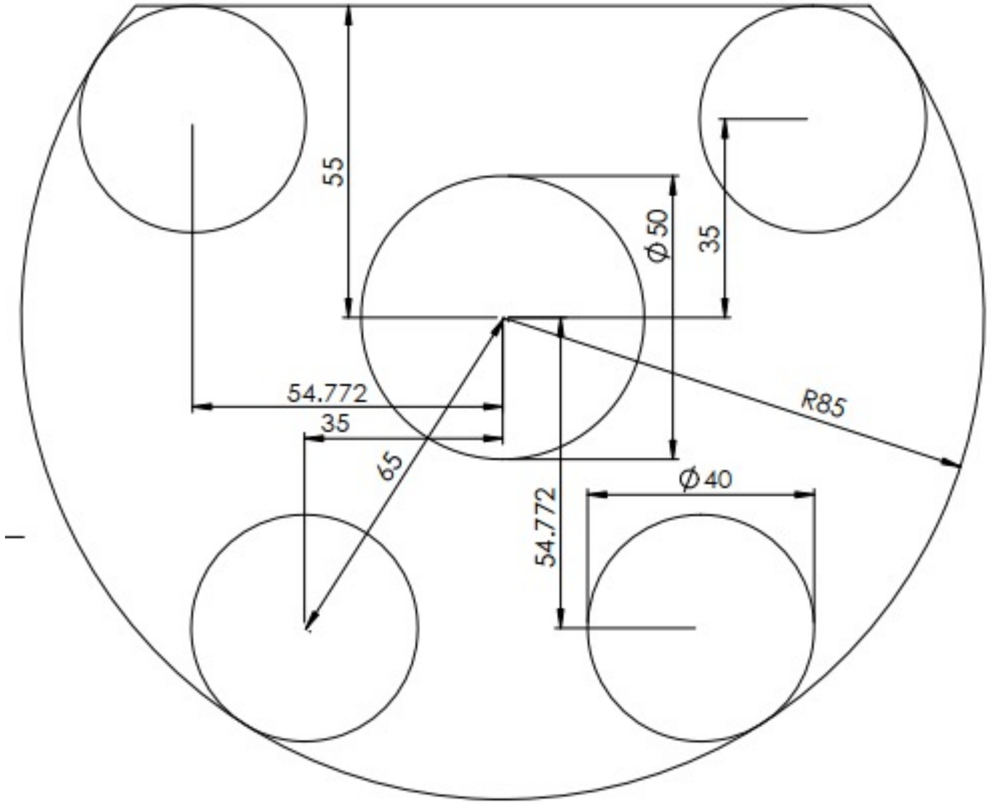
Recognition of robot patterns

Pattern :

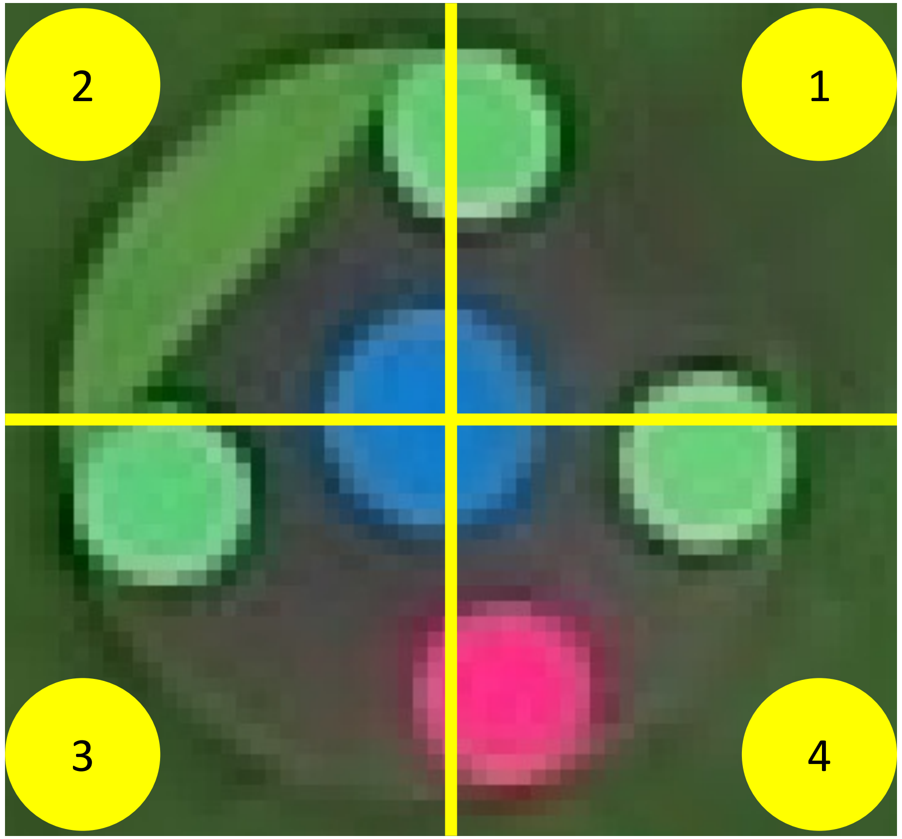


Recognition of robot patterns

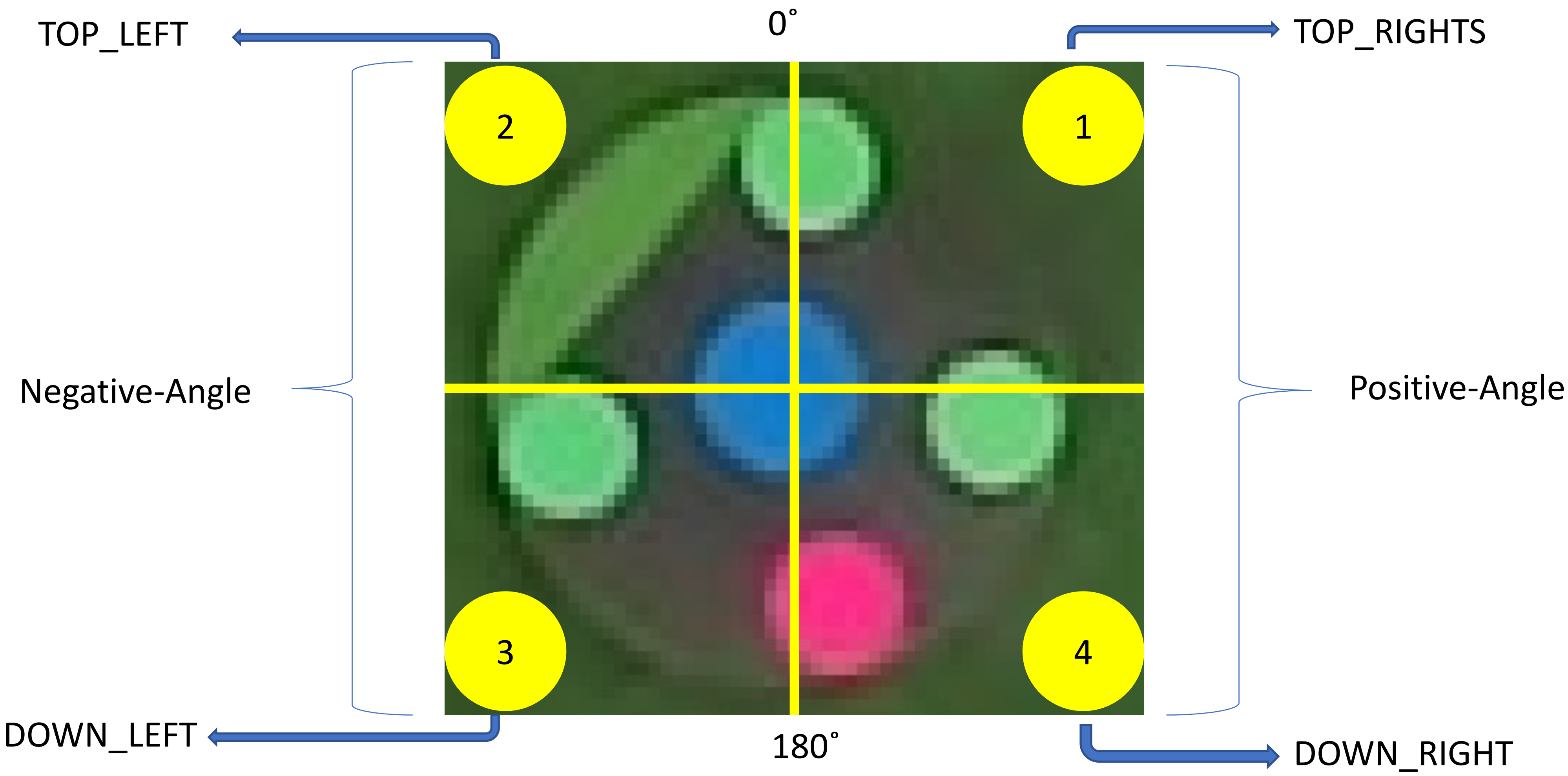
Pattern recognition :



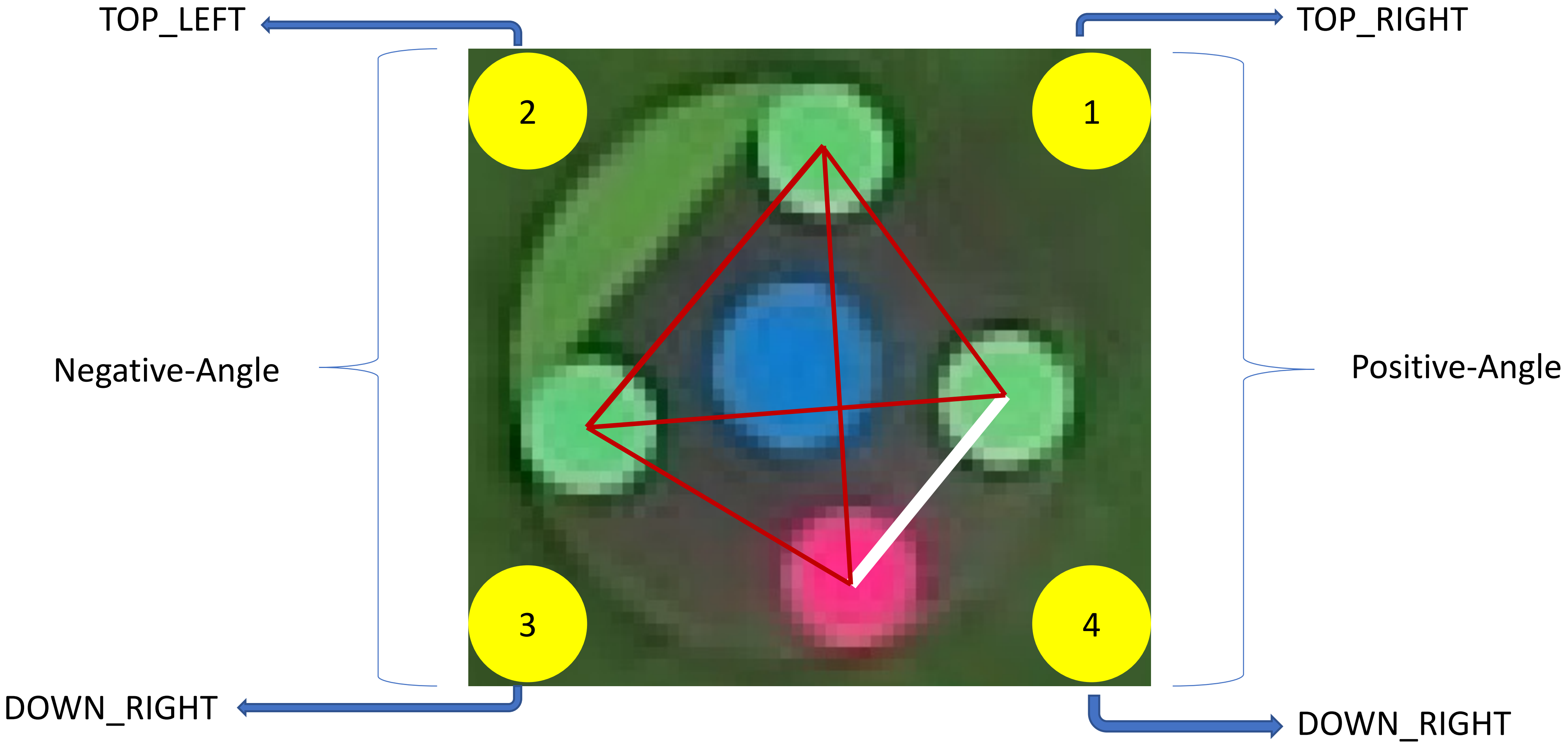
https://github.com/RoboCup-SSL/ssl-vision/blob/master/patterns/teams/standard/standard2010_template_all_dimensions.pdf



Recognition of robot patterns



Recognition of robot patterns



TOP_LEFT

TOP_RIGHT

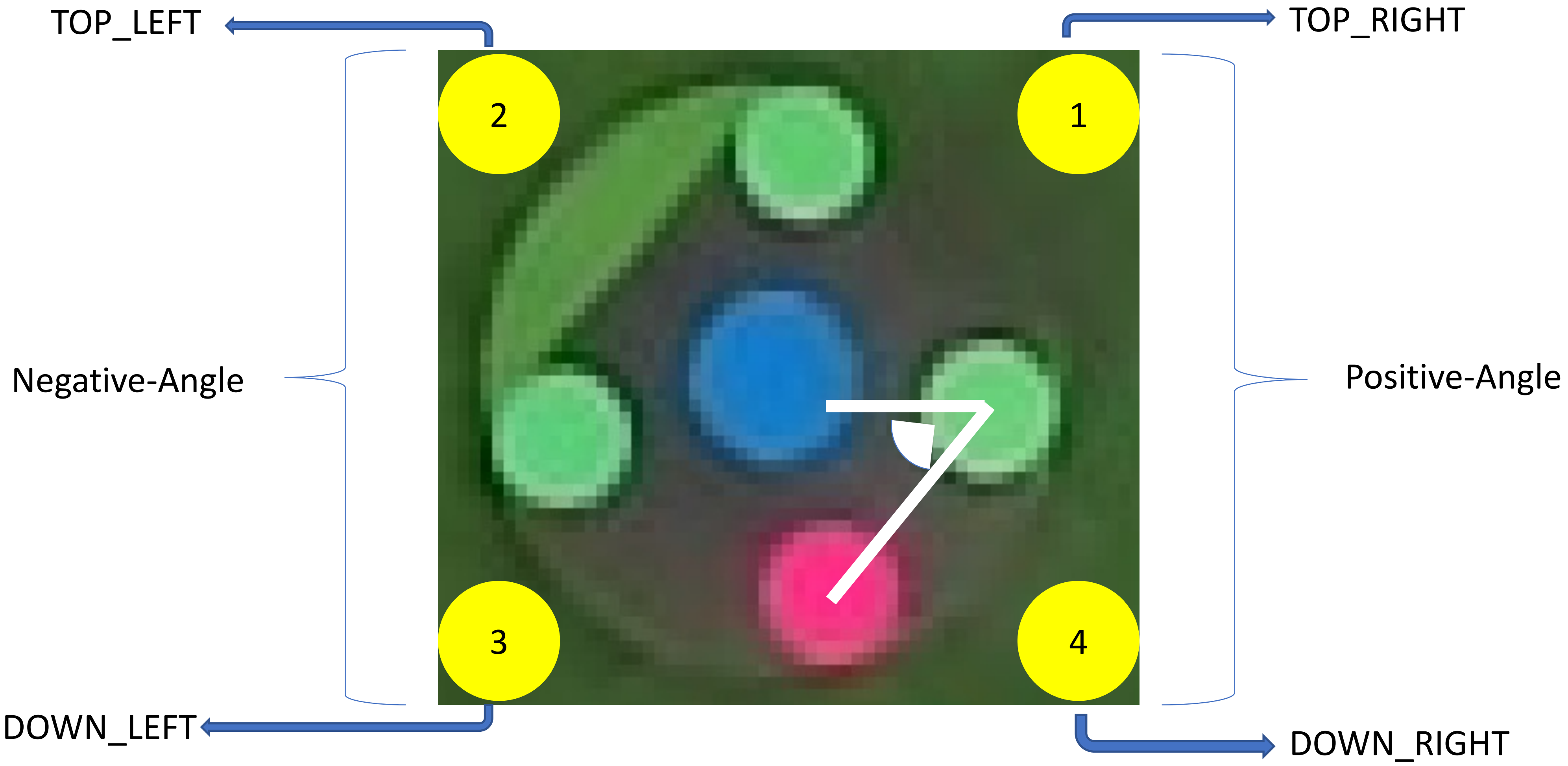
Negative-Angle

Positive-Angle

DOWN_RIGHT

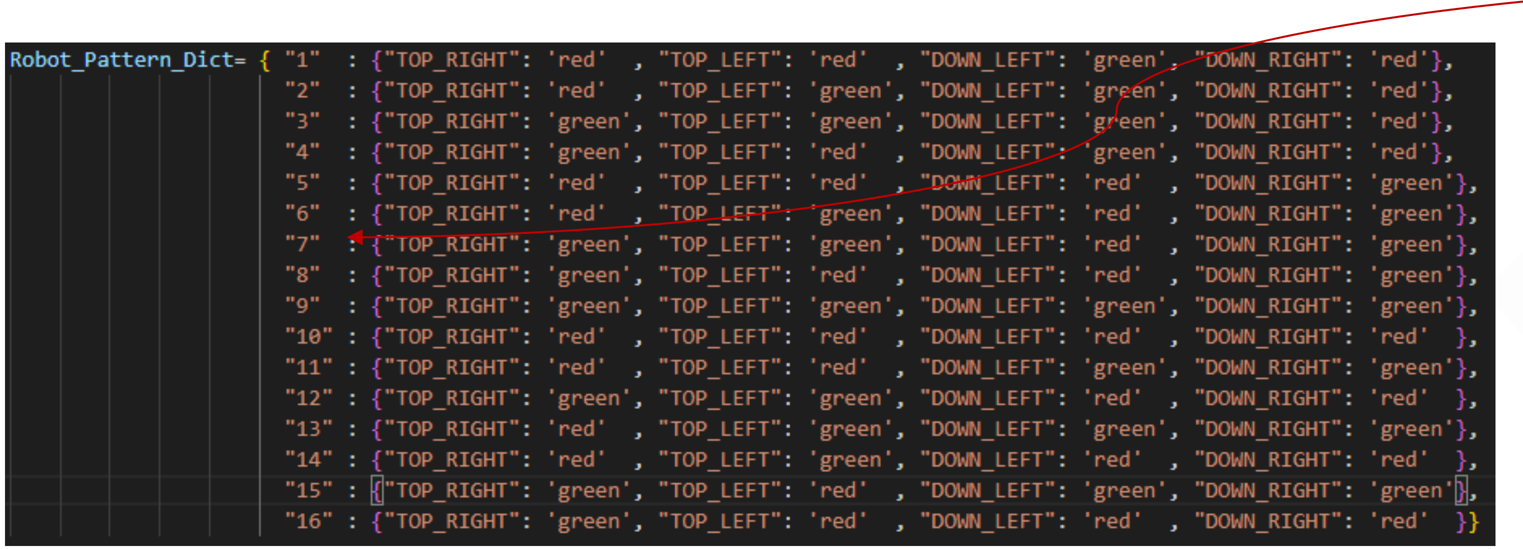
DOWN_RIGHT

Recognition of robot patterns

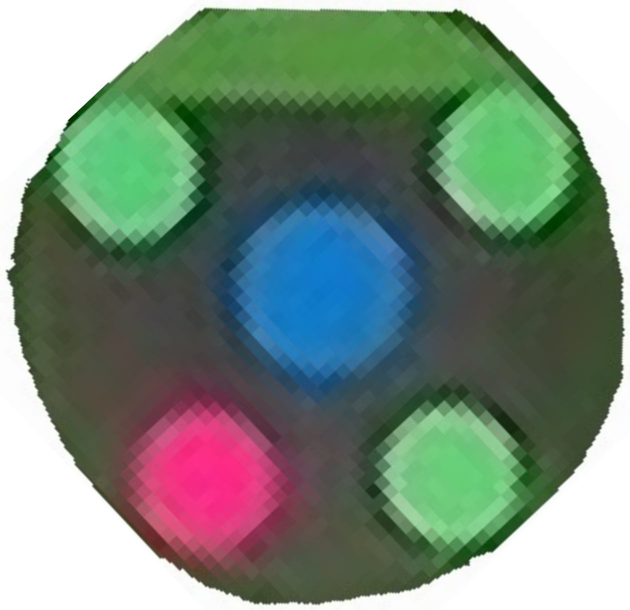


Recognition of robot patterns

finding robot:

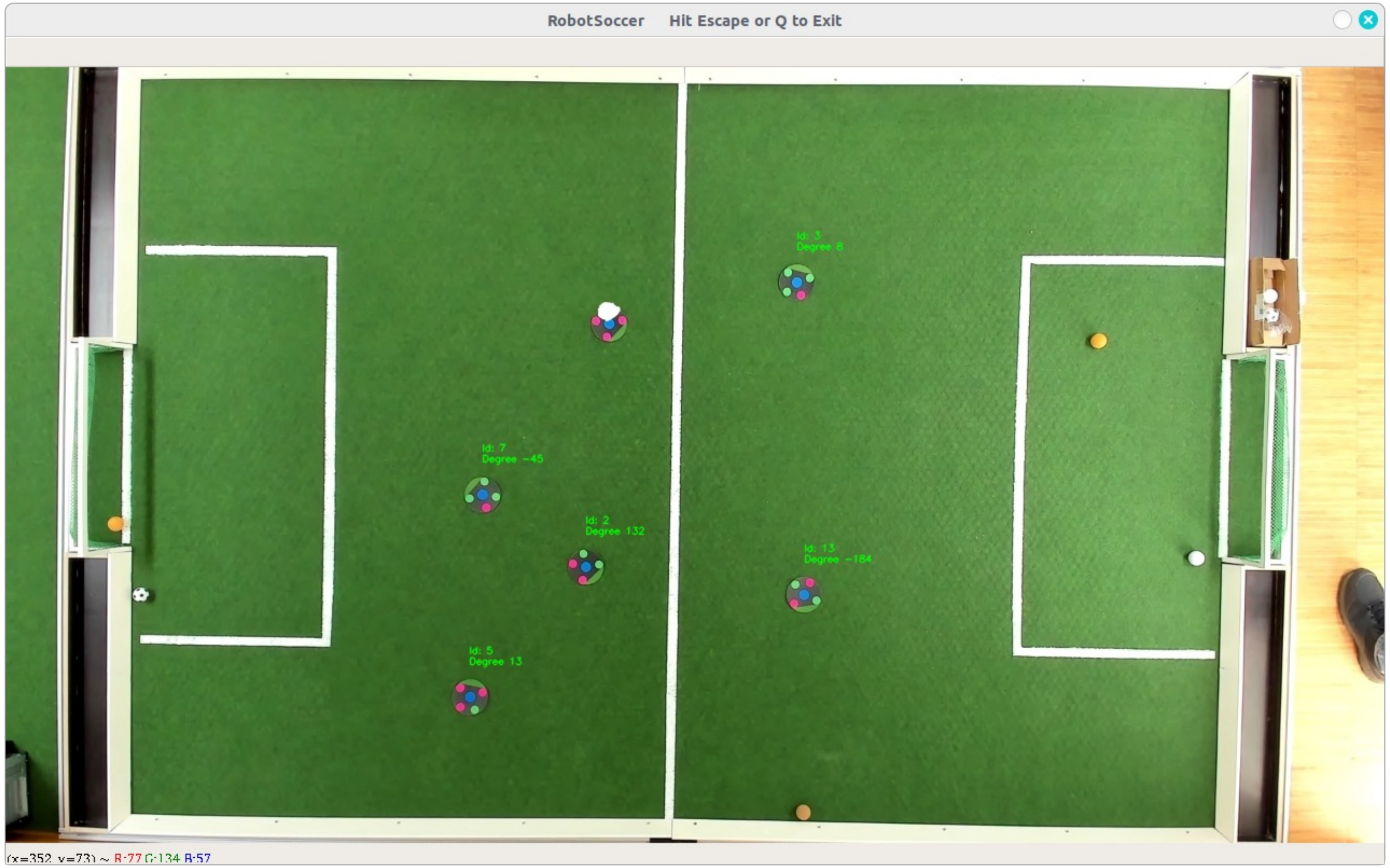


```
Robot_Pattern_Dict= { "1"  : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'red'  , "DOWN_LEFT": 'green', "DOWN_RIGHT": 'red'},
                      "2"  : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'green', "DOWN_LEFT": 'green', "DOWN_RIGHT": 'red'},
                      "3"  : {"TOP_RIGHT": 'green', "TOP_LEFT": 'green', "DOWN_LEFT": 'green', "DOWN_RIGHT": 'red'},
                      "4"  : {"TOP_RIGHT": 'green', "TOP_LEFT": 'red'  , "DOWN_LEFT": 'green', "DOWN_RIGHT": 'red'},
                      "5"  : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'red'  , "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'green'},
                      "6"  : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'green', "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'green'},
                      "7"  : {"TOP_RIGHT": 'green', "TOP_LEFT": 'green', "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'green'},
                      "8"  : {"TOP_RIGHT": 'green', "TOP_LEFT": 'red'  , "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'green'},
                      "9"  : {"TOP_RIGHT": 'green', "TOP_LEFT": 'green', "DOWN_LEFT": 'green', "DOWN_RIGHT": 'green'},
                      "10" : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'red'  , "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'red'  },
                      "11" : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'red'  , "DOWN_LEFT": 'green', "DOWN_RIGHT": 'green'},
                      "12" : {"TOP_RIGHT": 'green', "TOP_LEFT": 'green', "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'red'  },
                      "13" : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'green', "DOWN_LEFT": 'green', "DOWN_RIGHT": 'green'},
                      "14" : {"TOP_RIGHT": 'red'  , "TOP_LEFT": 'green', "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'red'  },
                      "15" : {"TOP_RIGHT": 'green', "TOP_LEFT": 'red'  , "DOWN_LEFT": 'green', "DOWN_RIGHT": 'green'},
                      "16" : {"TOP_RIGHT": 'green', "TOP_LEFT": 'red'  , "DOWN_LEFT": 'red'  , "DOWN_RIGHT": 'red'  }}
```



Recognition of robot patterns

The final results for all robots with the same pattern recognition:



(x=352, y=73) ~ R:77 G:134 B:57

Real-time data processing and application performance

Multithreading:

Multiprocessing

Advantage

- Faster for light process
- Shared memory
- One thread shares the data, the code
- Threads take less time to terminate
- Communication between threads takes less time than between processes

Disadvantage

(This is how the thread works in Python)

GIL
(Global Interpreter Lock), Globale Interpreter-Sperre

Real-time data processing and application performance

Multithreading

Multiprocessing:

Advantage

- There is no GIL (Global Interpreter Lock).
- When a process is blocked, the remaining processes can continue execution.
- A process can run completely independently like a separate program

Disadvantage

- Processes are completely independent and do not share memory
- Processes require more time for creation
- Processes require more time for termination
- Communication between processes requires more time (QUEUE)

Real-time data processing and application performance

Python :

```
def for_loop(x):

    for i in range(0,x):

        x += 2

   return x
```

☹

Cython :

```
Cdef def for_loop(int x):

    cdef int i = 0

    for i in range(0,x):

        x += 2

   return x
```

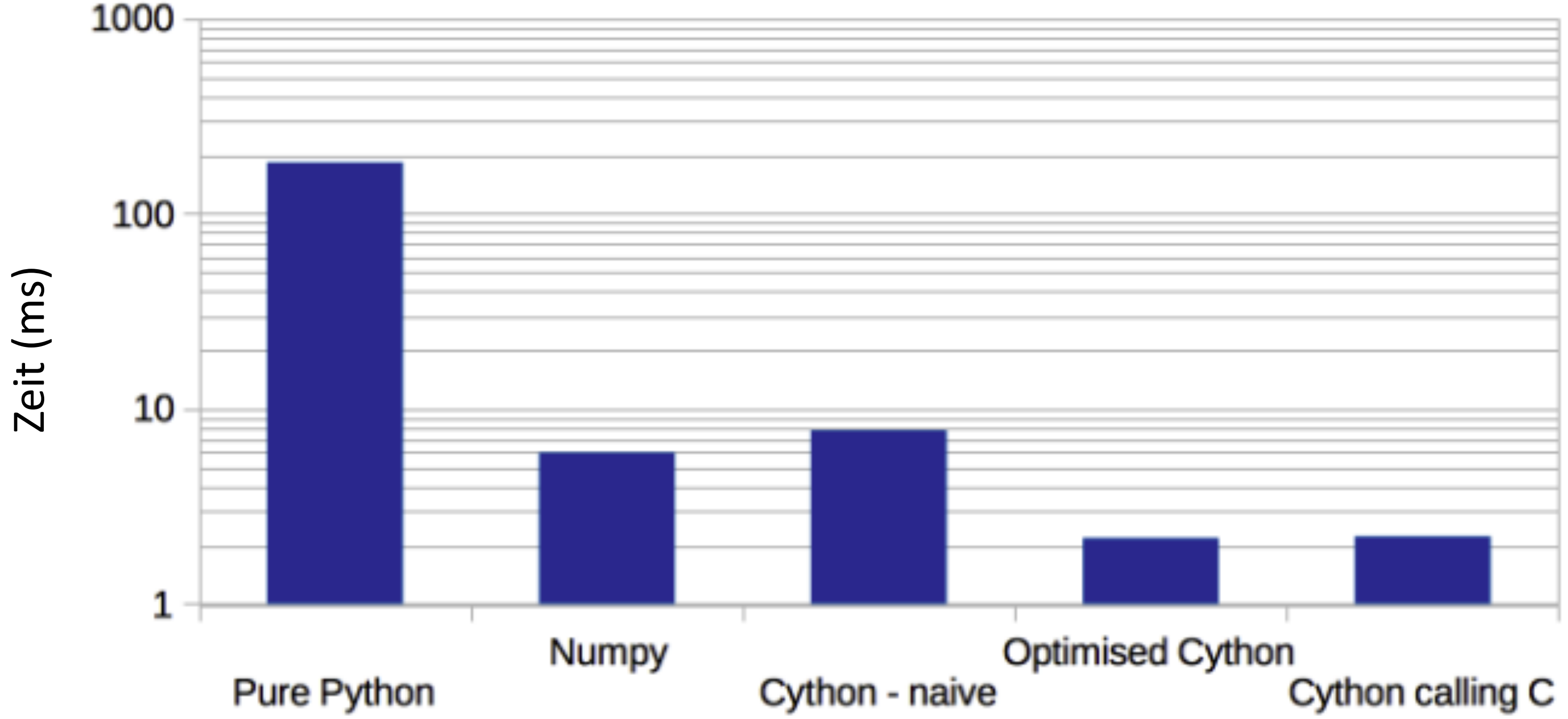
☺

https://en.wikipedia.org/wiki/Cython

Real-time data processing and application performance

Cython performance:

Standard deviation of 1e6 elements of the list



https://notes-on-cython.readthedocs.io/en/latest/std_dev.html

https://en.wikipedia.org/wiki/Cython

Server connection

Google Protocol Buffer :

The Google protocol makes it possible to develop universal code for several programming languages from one and the same file.

```
syntax = "proto3";
package protoblog;

message SSL_DetectionRobot {
        uint32 robot_id    =  2;
        float  x           =  3;
        float  y           =  4;
        float  orientation =  5;
        float  pixel_x     =  6;
        float  pixel_y     =  7;
        float  height      =  8;
        }
```

Server connection

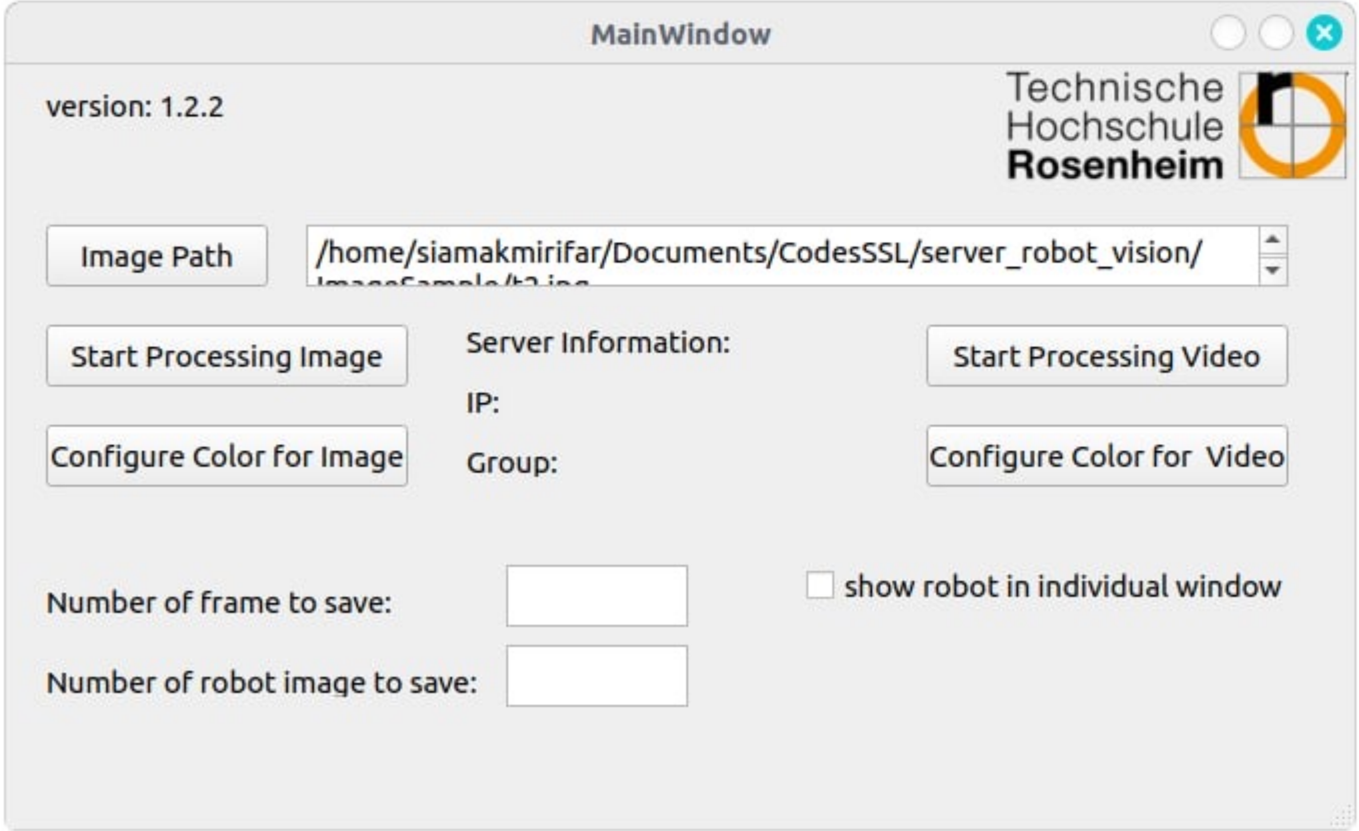
Server:

The User Datagram Protocol (UDP) is a minimal, connection network protocol that belongs to the transport layer of the Internet protocol family.

UDP enables applications to send datagrams in IP-based computer networks.

Graphical user interface

User interface :



version: 1.2.2

/home/siamakmirifar/Documents/CodesSSL/server_robot_vision/

END

Thank you for your attention