Report on Laboratory 3
Mouse callback and color segmentation

Computer Vision UniPD

Alessandro Viespoli 2120824

March 2024

Setup and instructions to run the code

OS: Linux (Pop!_OS)
Enviroment: CLion
CMake: 3.27.8
OpenCV: 4.5.4
Each task has its own `main.cpp` and `CMakeList.txt` file, which can be found in a dedicated directory for each task. The image to be load must be in the same folder as the `main.cpp`.
To run the code simply extract the content of the .zip file, enter task's directory and run in the terminal:

1. cmake .
2. cmake --build .

This will create an executable `task*`; in order to run it pass as `argv[1]` the image file.

Task 1

Simple use of the openCV libraries `imread` and `imshow`.

Task 2

In order to implement this task, I learned how to use the `cv::setMouseCallback`. In the program the specific line of code is:

```
cv::setMouseCallback(WINDOW_NAME,BGR_onMouse,&robocup);
```

`BGR_onMouse` is a function I wrote myself that if the left button of the mouse is pressed, it prints the BGR values of the pixel selected.

Task 3

Differently from task 2, the method name is different, `BGR_mean9x9_onMouse`. The behaviour is similar, however the computation of the mean around the 9x9 kernel is done inside this method (no extra functions called).

Task 4

In task 4, I expanded even further the method implemented in task3. First the name is changed, `Mask_onMouse`. Then, as task 3, after a left click the mean of the BGR components around the 9x9 kernel is computed and stored in variables. This values are then used for making the computations described in the laboratory assignment.

$$|img.at < Vec3b > (i,j)[z] - mean\_channel\_z| <= T$$

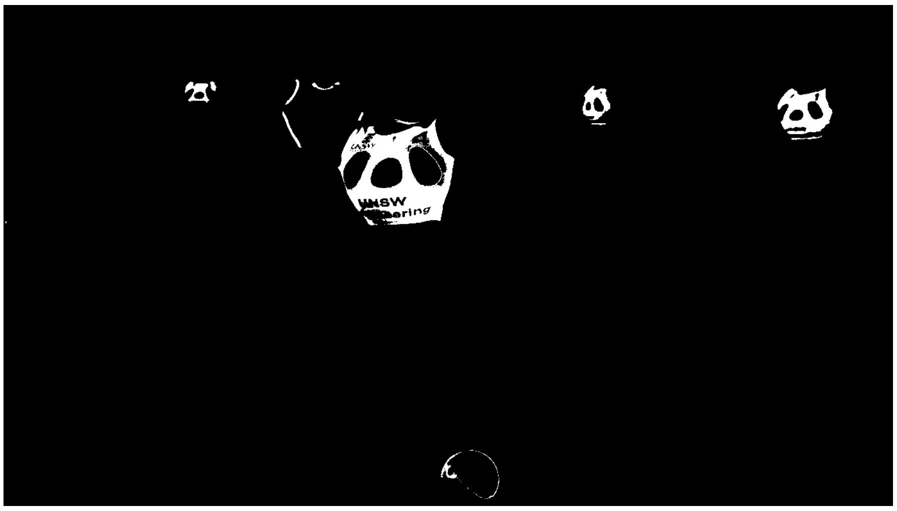

Figure 1: Mask of the robocup.jpg with threshold 60

Task 5

The overall calculations done here are the same as task 4, however the image is first converted in HSV encoding. Generally, the mask generated has little and insignificant variations with respect to the mask generated in task 4 for the same threshold T and point selected.

Task 6

The method presented in task 4 is expanded with new lines of codes that look at the value of the previously calculated mask, with the `cv::at` function, and in case the pixel is white the new color is applicated to the BGR robocup image. Then

the modified image is shown after every left click.
In general, $50 \leq T \leq 75$ are good values. A perfect selection is however impossible to achieve by simply applying the assignment; this is because the image has a lot of light points and shadows (e.g the ball is a perfect example of that). For me, the best threshold is $T = 55$ because it has less flaws (i.e other parts selected).

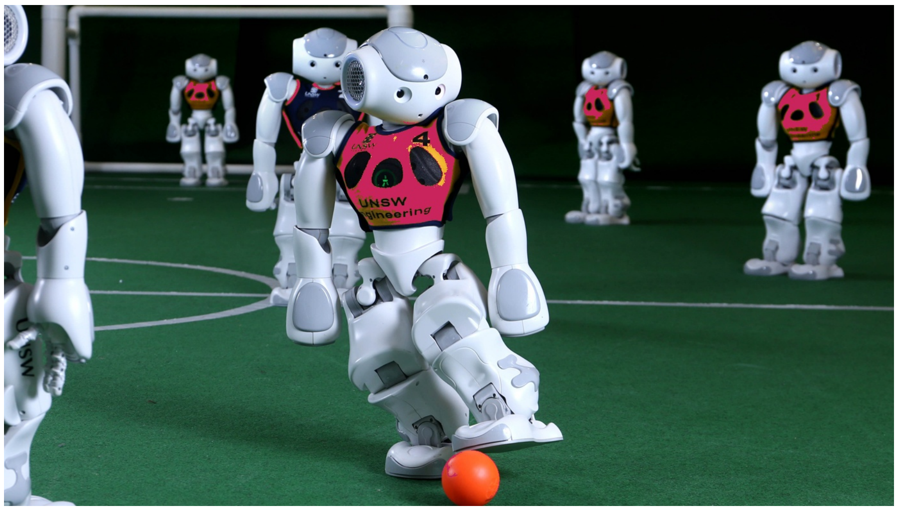

Figure 2: T-shirt color changed for robocup.jpg with threshold 55

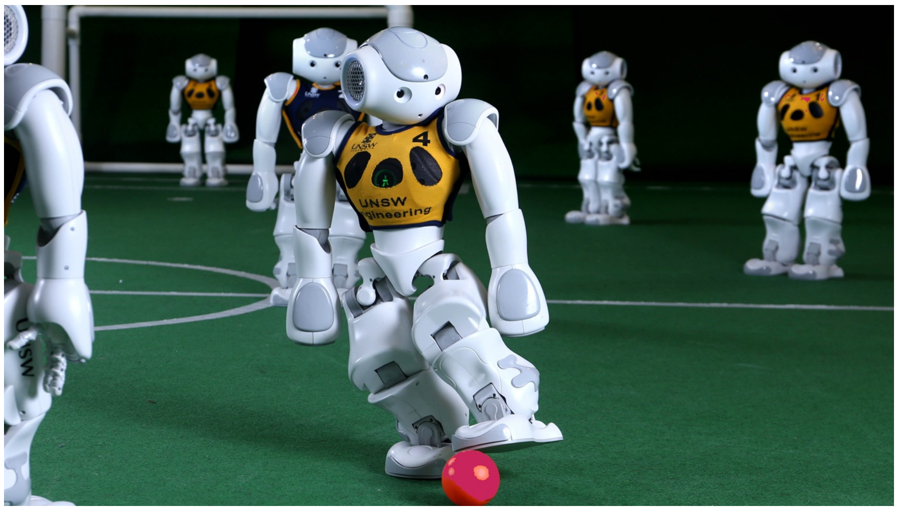

Figure 3: Ball color changed for robocup.jpg with threshold 55

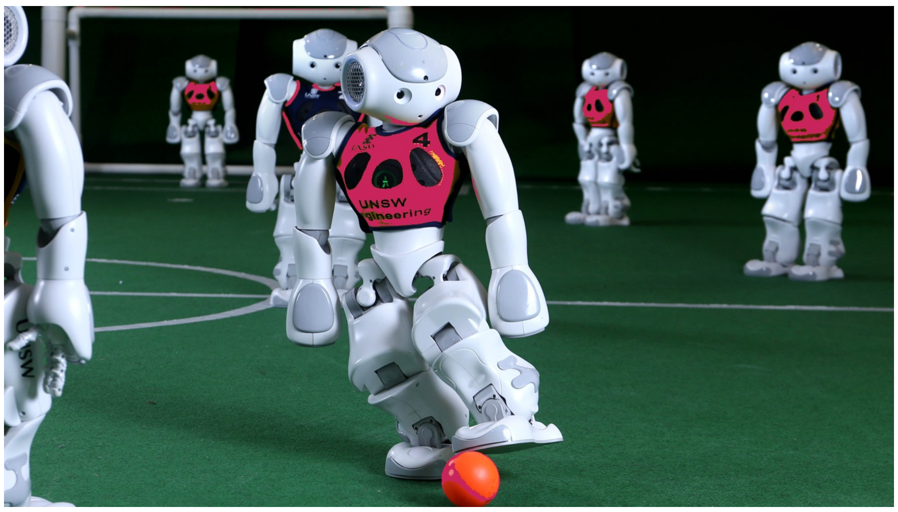

Figure 4: T-shirt color changed for robocup.jpg with threshold 72

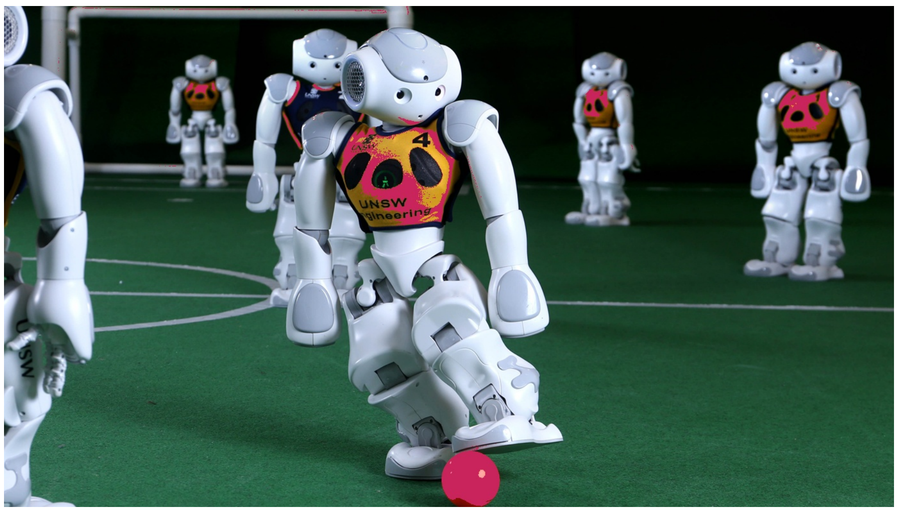

Figure 5: Ball color changed for robocup.jpg with threshold 72