INP9087774 - COMPUTER VISION

Lab 3
Mouse callback and color segmentation

Simen Nesland

03.27.2024

Task 1

Loading and displaying the image "robocup.jpg" was done easily using `cv::imread()`, `cv::namedWindow()` and `cv::imshow()`. Checks for length of argc and content of `cv::Mat::data` were also included to avoid errors. Included is the resulting window.

Figure 1: Window from opening image in Task 1.

Task 2

Task 2 was implemented using the `cv::setMouseCallback()` as explained in the task. A new function `printAvgPixelBGR()` was created and passed as the `onMouse` argument. The function simply checks if the event was a left button click. If this is the case the pixel values at this x and y coordinate is printed to the console. The image is accessed through the `userdata` parameter where the address of the image is passed.

Task 3

Task 3 was implemented very similarly to Task 2, however the `printAvgPixelBGR()` function is replaced by `printAvgPixelBGR()`. The only difference between these functions is that the second is implemented looping over the described 9x9 kernel. Three `std::vector<int>` objects are created storing the B, G and R values respectively. Finally the average of these are calculated and the result is printed to the console.

Task 4

In Task 4, the onMouse argument was changed to a new function called createMask(). This functions checks if the left button is clicked. If this is the case, the average BGR is found using a funcion `getAvgPixelBGR()` (which is the same as `printAvgPixelBGR()` except the result is returned in a `std::vector<int>` instead of being printed). An empty `cv::Mat` is created and looped over using a double for loop. The corresponding pixel in the original "robocup.jpg" image

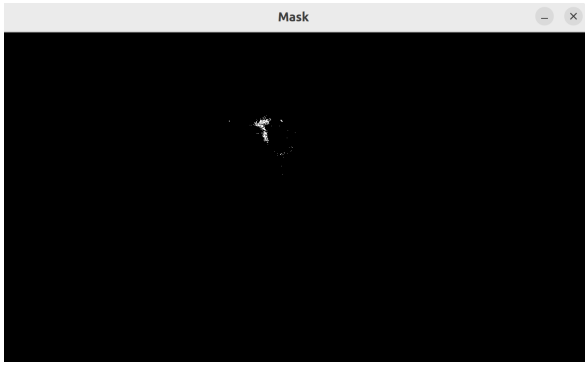

Figure 2: Mask for yellow shirt with threshold = 5.

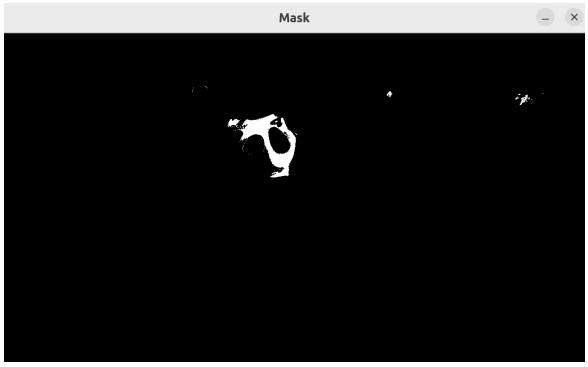

Figure 3: Mask for yellow shirt with threshold = 30.

is checked and colored to white if the B, G and R value is no more than the threshold away. This image is then showed using `cv::imshow()`.

Included are two of these masks where the same pixel was selected on one of the yellow "shirts", but with different thresholds.

Task 5

Task 5 was implemented very similarly to Task 4. A copy of the image was converted to HSV using `cv::cvtColor()`. The image shown uses the original image since `cv::imshow()` treats a `cv::vec3b` encoded image as BGR. However when the image is clicked the HSV-encoded image is passed to the `onMouse` function. This function is more or less identic to the function from Task 5 except it was found that only using the H (hue) value yielded better results. This is probably because the BGR endcoding is very dependent on f.ex. the brightness while using only the hue we are less dependent on where and what lighting different objects in the image have.

The image included uses the same pixel as from Task 4 but with only a "hue threshold" of 6.

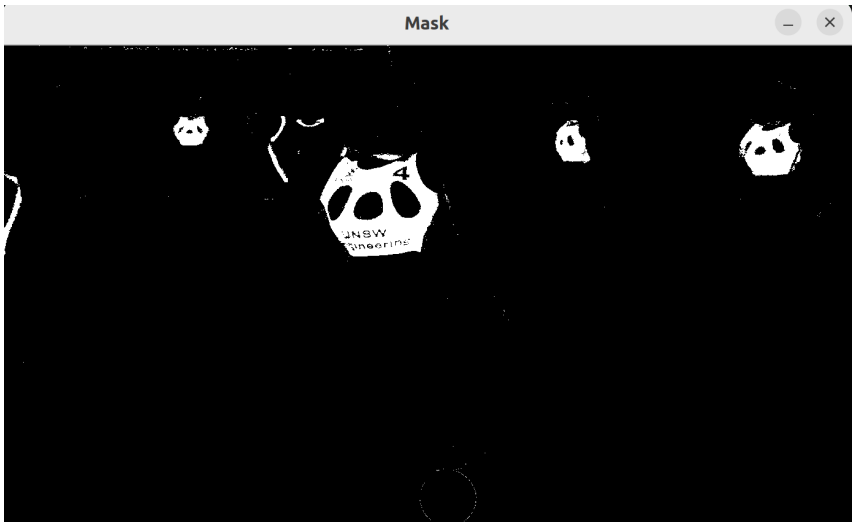

Figure 4: Mask for yellow shirt with threshold = 6 (only for hue value).

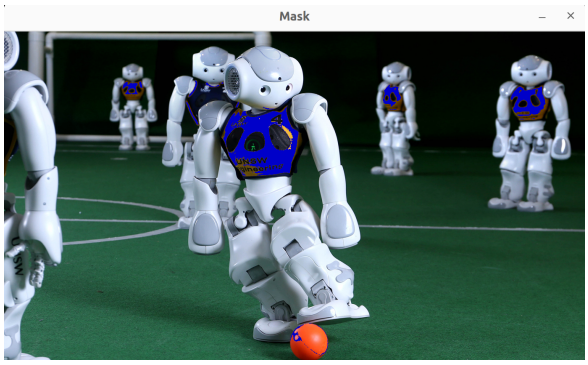

Figure 5: Mask for yellow shirt with threshold = 60. Similar pixels are colored to (92, 37, 201).

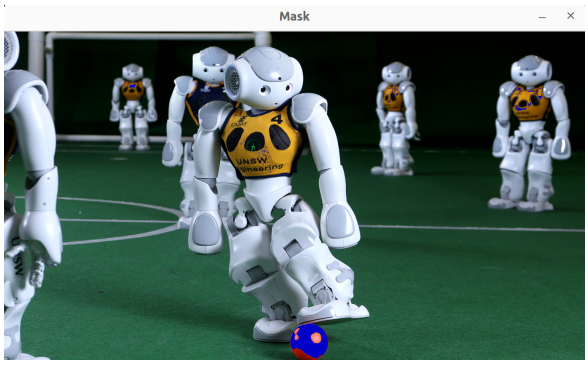

Figure 6: Mask for orange ball with threshold = 60. Smilar pixels are colored to (92, 37, 201).

Task 6

Task 6 was implemented almost identically to Task 4. The difference is only when filling the empty `cv::Mat mask` created. Instead of filling it with white, it is by standard put to the same pixel as the original image. If the BGR values are inside the threshold however, the value is set to the given (92, 37, 201) triplet.

The results were not very impressive. The two images included are the best results achieved using a threshold of 60. The results would probably be a lot better using the HSV encoding to create the mask.