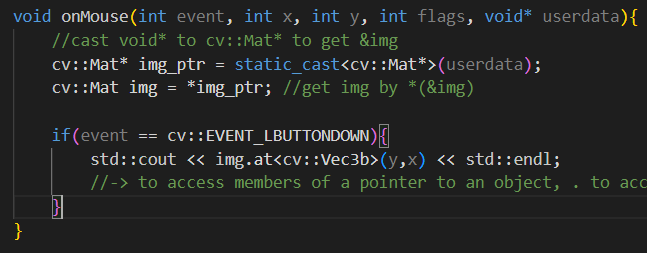
Lab 3 – Mouse callback and color segmentation Morselli Alberto (2102413)

***-short report-***

**≥ ./task1** – NSTR (nothing-significant-to-report).

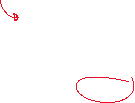
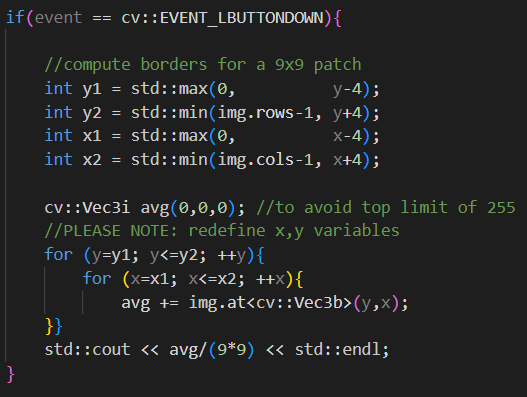
**≥ ./task2**

Except for the somewhat tricky (but brilliant) way of passing the arguments there’s nothing else to report, the code works well and does his job of printing the BGR-triplet on the cmd-line. The caller first casts the address of img &img to a void pointer void\* before passing it. Then inside the callback function onMouse it get cast-ed back to cv::Mat\* and finally accessed de-referencing the pointer.



**≥ ./task3**

Here i tackled the problem of the saturation on cv::Vec3b values. Indeed, summing up significant values for 81 (9x9) times almost always get clamped on the top 255 <uchar> saturation value. With the white color or similar bright ones we hit the saturation treshold in just 2 steps. I solved it switching to a cv::Vec3i to sum up values on the 9x9 patch and to finally divide it elementwise by the amount of pixels of the patch (81), thanks to the overloading of the division operator (/) for the cv::Vec3i typedef.



**≥ ./task4**

For this task i wanted to pass T by cmd-line to the callback function too, to avoid re-compiling for testing different tresholds so i had to first pack T and img inside an std::pair<..> and after unpack them inside the function. I then accessed img and T directly via -> operator to access members of a ‘pointer to an object’, avoiding to dereference it. I then organized the code inside separate small functions: one for computing the average color returned as cv::Vec3b, one for computing if the elementwise distance of two cv::Vec3b is (elementwise) smaller than the treshold T and one for creating the mask image. On the next page are shown some tests selecting the white body and the yellow t-shirt with two different tresholds of T=30 and T=60.

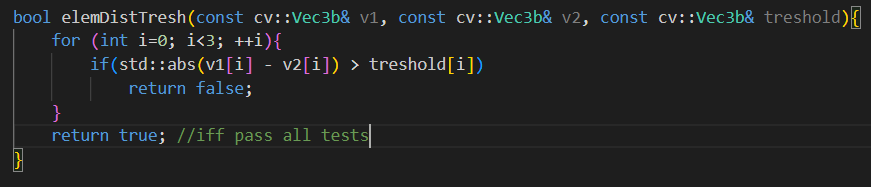
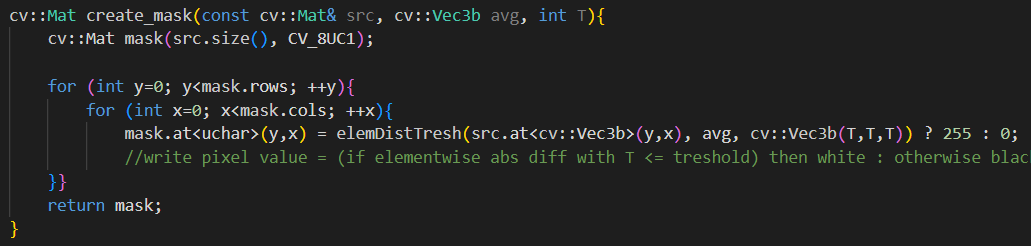
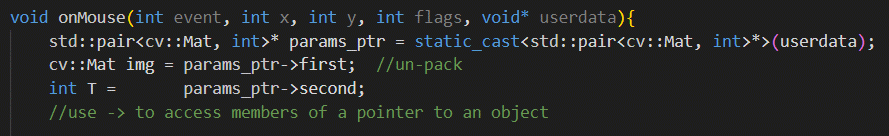


Immagine che contiene nero, arte, bianco e nero, monocromatico

Descrizione generata automaticamente Immagine che contiene bianco, cartone animato, nero, stile

Descrizione generata automaticamente

selecting white body, T=30 T=60

Immagine che contiene nero, oscurità, bianco e nero, schermata

Descrizione generata automaticamente Immagine che contiene nero, schermata, bianco e nero, Elementi grafici

Descrizione generata automaticamente

selecting yellow t-shirt, T=30 T=60

The BGR color space seems not to be the best suitable to perform color distance in this way, also because the treshold T is the same for all the three channels. As can be seen from the t-shirt tests, 60 is enough to well segment the t-shirt but also to include the (unwanted) orange ball on the floor.

**≥ ./task5**

Immagine che contiene testo, Carattere, schermata, Elementi grafici

Descrizione generata automaticamenteAs stated by documentation there are two ways to convert to HSV color space, the first one where the Hue value is converted to a 0-179 (degrees) range and the \_FULL one that stretch the value to fit the range in the entire <uchar> 0-255 range. I decided to use the second one for consistency with the BGR color space scenario to keep the same numerical treshold and compare the results.

Immagine che contiene nero, oscurità, bianco e nero, schermata

Descrizione generata automaticamente Immagine che contiene nero, bianco e nero, schermata, Elementi grafici

Descrizione generata automaticamente

T=30 in the HSV color space T=60

I didn’t noticed significant differences, then i tried to use a different treshold for every channel, being able to obtain the same results with a Hue (the richest in information that selects the chroma) treshold of ~10/20 rather than 60 as before, while keeping Saturation and Value tresholds very high (~200) to cover the entire shades of the colors. Probably the key is to use three different tresholds in the correct way on the HSV channel, or in some color space where the brightness difference of two pixel of the same color is not that influencing.

**≥ ./task6**

Getting back to BGR color space seems not to be the smartest idea, because trying to segment the ball, which has very different pixel values moving from the shaded to the illuminated side, implies having to select very high tresholds that include also unwanted elements, as can be seen by the next images.

Immagine che contiene palla, rosso, arancione

Descrizione generata automaticamente Immagine che contiene cartone animato, Elementi grafici, arte

Descrizione generata automaticamente Immagine che contiene schermata, nero, oscurità, bianco e nero

Descrizione generata automaticamente



Immagine che contiene schermata, schermo, testo, software

Descrizione generata automaticamente ball and zoom (below) of pixel values selecting the shaded side selecting the lighted side

Notice in particular the blue channel that spans between ≈50 to ≈160, with a treshold that should be >= 110 to cover entirely the ball. Below it’s shown the application of the mask on the ball with with a treshold value of T=60, while at the end of the page the application on the t-shirt with two treshold values of T=60 and T=90.

Immagine che contiene automa meccanico, elmetto, robot, giocattolo

Descrizione generata automaticamente zoom of the ball to appreciate the difference between pixel values selecting the ball, T=60

**Immagine che contiene automa meccanico, robot, giocattolo

Descrizione generata automaticamenteImmagine che contiene automa meccanico, robot, giocattolo, persona

Descrizione generata automaticamente**

selecting the t-shirt, T=60 T=90