Authors:
Carlotta Schiavo, ID: 2076743
Qiu yi jian, ID: 2085730

“Food recognition and leftover estimation” REPORT

I. Our approach

We decided to split the work into two-part, food detection and food segmentation and assign each part to a group element. Below we are going to explain out approach to solve each issue.

Both for segmentation and food Detection we agree not to use deep leaning but instead a more manual approach exploiting and elaborating the techniques offered by the OpenCV libraries.

▪ Food Detection:

We decide not to use deep leaning and hence to use a more manual approach.
After trying several approaches, we finally adopted a method that work with the color of the images.
We assume that the food detection it always does on the “before” tray, because since the canteen goal is to track the food waste, we must have to take the initial food quantities.
In this way, we can use the results obtain from this one to find perform the localization of the plates in the leftover trays, without actually do again a detection as the one done for the “before” tray but an easier detection where the possible food classes that we can find in the tray, are already filtered.

Food detection on the “Before” trays:

We have taken for each class of food a set of color ranges working with the HSV format of each the image in the sample’s dataset.
The tray where the detection must be done, is taken as input and before starting the food detection, the image is elaborated using a function that, exploiting the Hough transform, is able to extract from the tray image a vector containing an image for each plate that is able to found, where only the food is highlighted (hence the plate was partially erase).
Having this kind of vector allow us to work better on each plate, erasing a big part of noisy element such as glasses, mobile phone and so on.
Having this vector of plate images, we can start the food detection.
First of all, we take the i^{th} plate saturation value.
Then, for each plate on the vector and for each possible set of ranges that we have for each food sample, we are going to count the number of pixels that we found in our plate image, having the colors in the set of ranges of the i^{th} sample plate.
After that the approach is to make a first filter action, ordering in descending order all the values of number of pixels found in each range.
From this list we’re going to select the 6 highest values founded. These set of samples are the most similar, in term of color, to the i^{th} plate we are classifying.
For each one of these 6 nearest samples, we are going to take the relative saturation.

At the end we will take as classification the sample that has the closest saturation to the i^{th} plate.

Food detection on the "After" trays:
For the detection in the after trays, we are going to take as input the list classifications done on the "before" tray.
The idea is to repeat a similar detection as the one done on the "before" tray but this time without perform the color filtering since we already have the list of the possible food we can find into the tray. What we do is just analyzing the saturations value of each leftover plate with the saturation values of the samples of the foods list obtained by the "before" tray detection

At the end of each detection, the image with the bounding box and the labels of the food found, will be shown.

▪ Food Segmentation:

First of all, the image is preprocessed by the function Hough Circle that extract the plates that contain foods from the image, including the salad that is on a small transparent plate. Obviously, the preprocessing is not perfect, this is why after the Hough Circle we analyse the circles checking if there are circles with just glass or circles that don't contain any relevant information. After the preprocessing we start the main segmentation part, this task is assigned to the function grabCut that, from an input image, it extracts the foreground image which is the food on each plate. Before the grabCut function, we did other operations on the image allowing the function to perform better. For each image that contain only one plate, we decided to color all the other pixels and also the pixels of the plate with a gray color. In particular the pixels that will be coloured are all the one whose threshold of the color in the channels are all less than 1.3. Moreover, grabCut function needs a box in which it will highlight the foreground, so we roughly created the bounding box using squares with the sides equal to two times the radius. So only after this we applied the function that extract the food. Now each food on a single plate will be coloured with a predefined color and put together in a single image and given in output.
We used a similar approach to compute the masks of the foods, in particular we extracted the plates with the HoughCircles function and then after grabCut function we create a vector of images and centres (the centres are computed by HoughCircles), the centres will be used to get the id of the food from the food detection, and it will be used to color the food mask with that color of gray. After getting all the images of the food, we used the function entireMask that color each food with the associated id got from the foodSegmentation function and will merge all the single masks into one that will be given in output. Furthermore, for colouring the single masks, we got from input also a vector in which there are the labels of the food with the respective center, radius and id, with these data we check the correctness of the color that we will use to colour the food mask.
Lastly the bounding box task has been achieved by using two functions, "computeBoundingBox" and "drawBoundingBox". The first one is used to compute the bounding boxes, in particular, this function takes in input the tray image and extract a

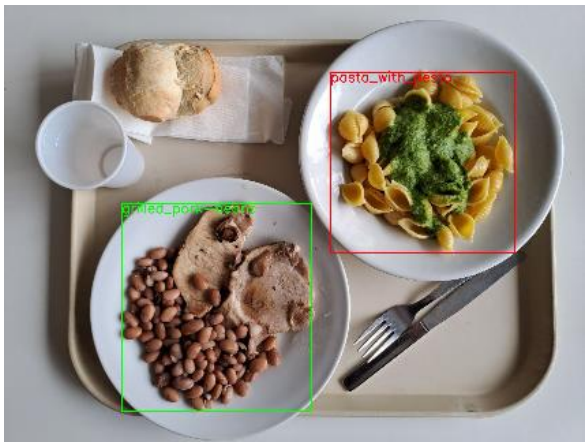
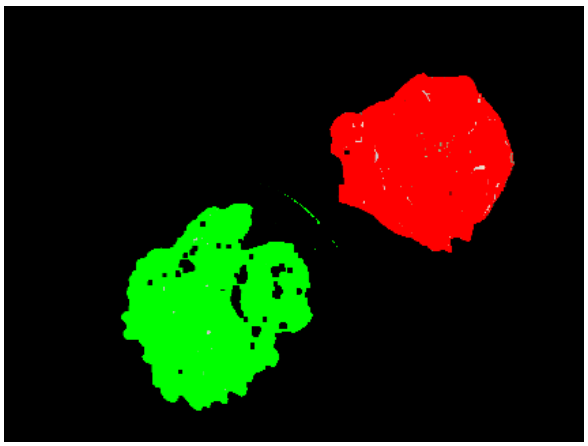
vector of image in which there are the images with single plates, after this, we applied the function applyPolyDP that calculates the contours of the food and allow us to compute the bounding box. The second function, “drawBoundingBox” draw the bounding box on the input image using the rectangle function.

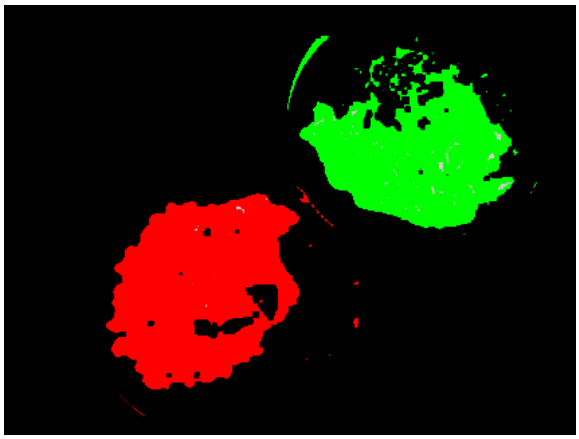
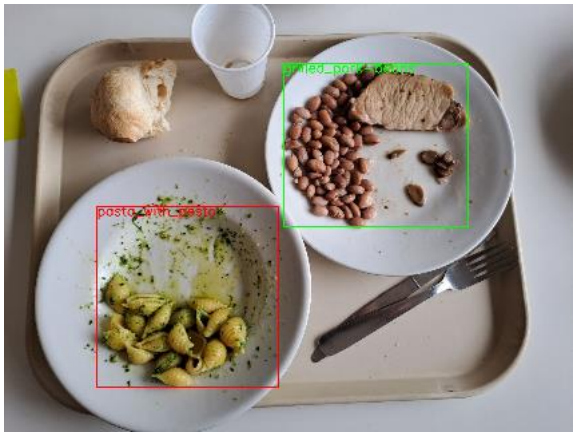
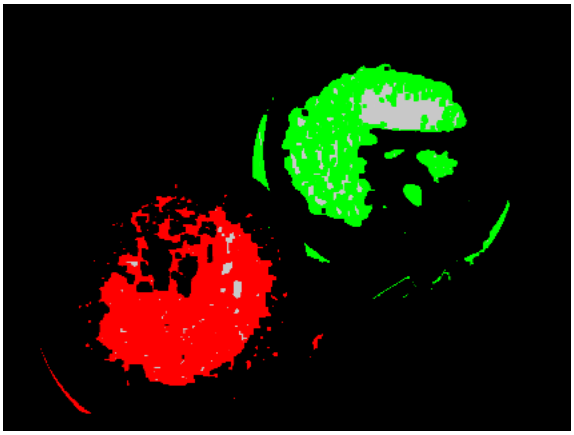
II. Results:

Images of the results. For each tray are reported the results for all the three versions (before, leftover1 and leftover2), both of food detection and segmentation.

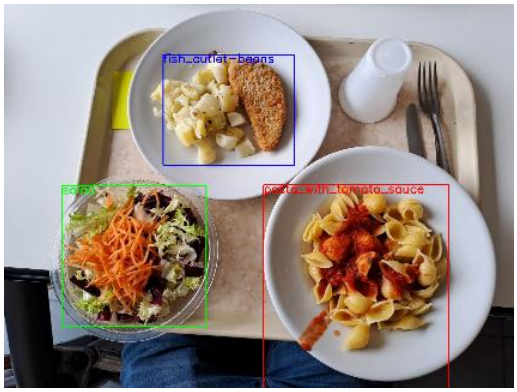
(**NB**: here the segmentations images are not the mask using for the measurements. In this set of images, the segmented foods are randomly colored because we aim to show only the segmentation (splitted by the food detection). In order to see the masks containing the segmentation combined with the food classification (where each pixel is classifies) look at the folder "detected_mask" into the project).

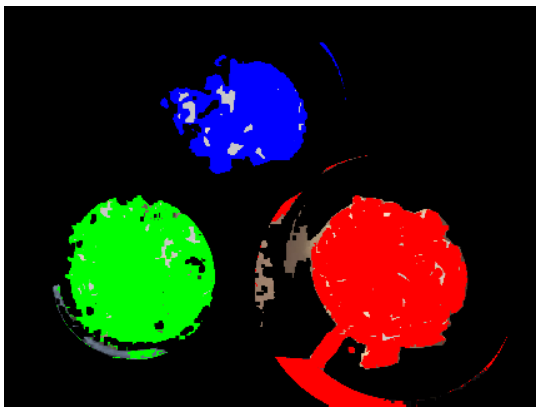
Tray 1:

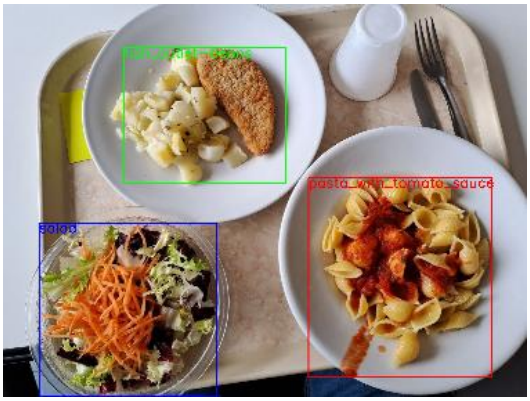



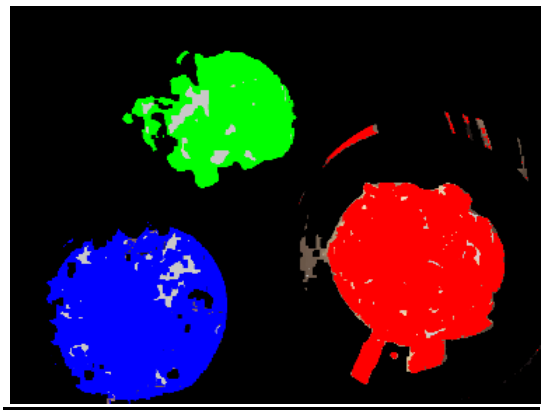




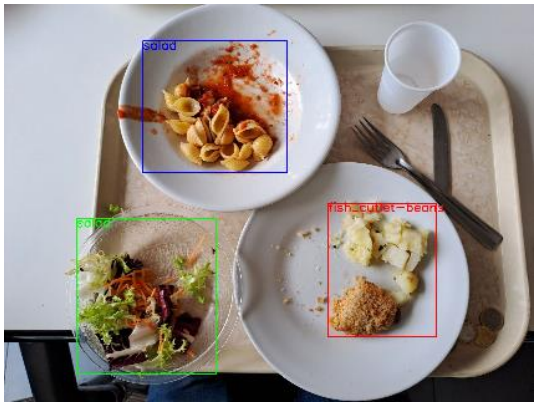
Tray2:

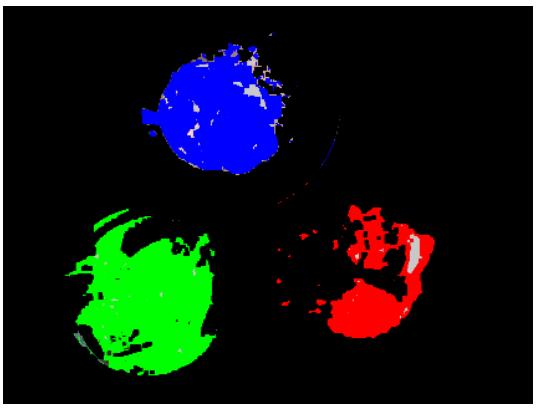

fish_cutlet_beans
pasta_with_tomato_sauce



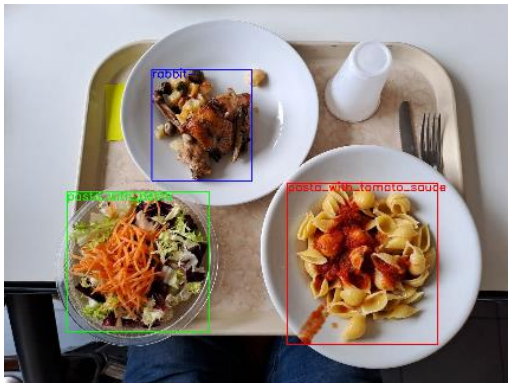

pasta_with_tomato_sauce
salad

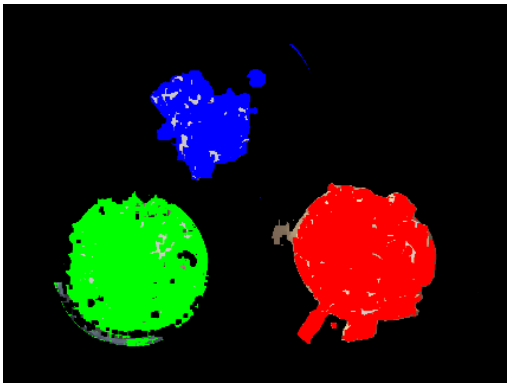


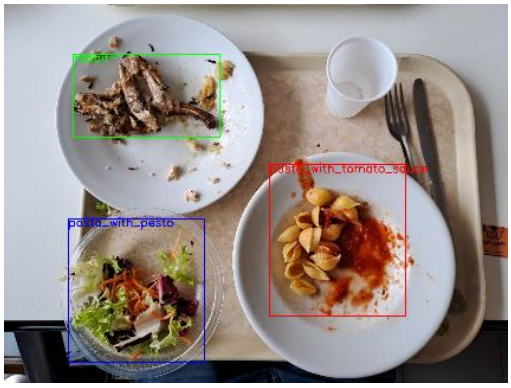

salad
fish_cutlet_beans

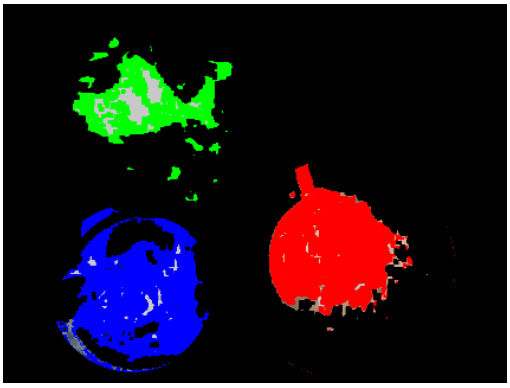


Tray 3:

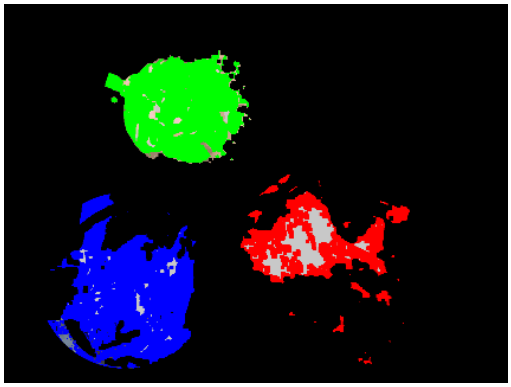

rabbit
pasta_with_pesto
pasta_with_tomato_sauce



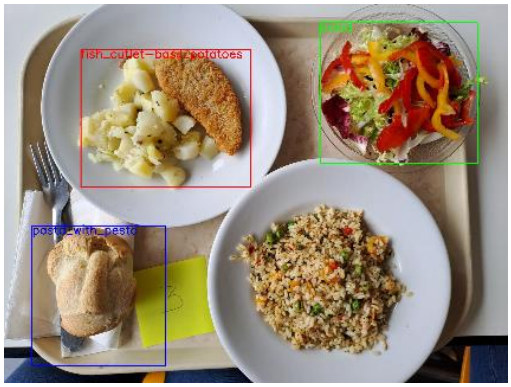

pasta_with_pesto

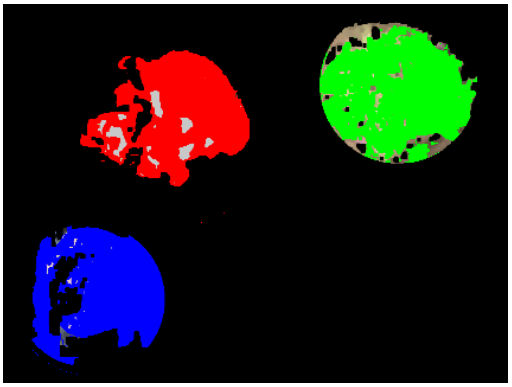



pasta_with_tomato_sauce

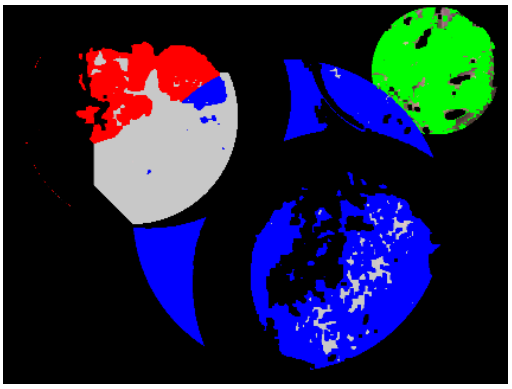


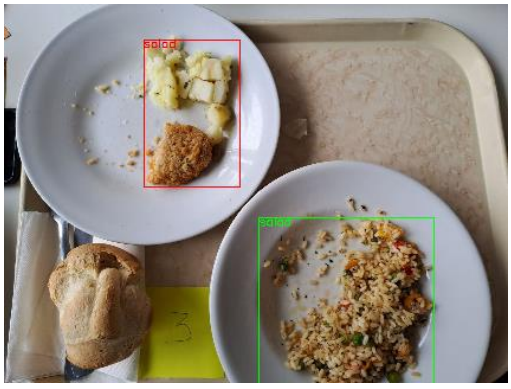
Tray 4:

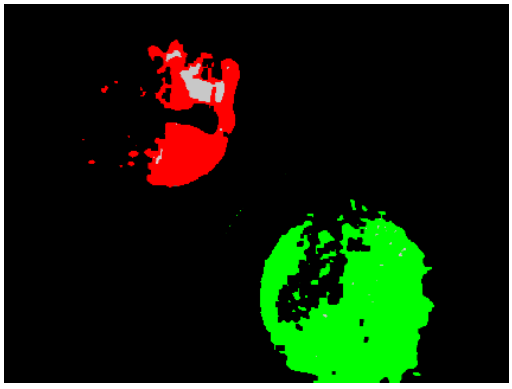

fish_cutlet-basil_potatoes
salad
pasta_with_pesto



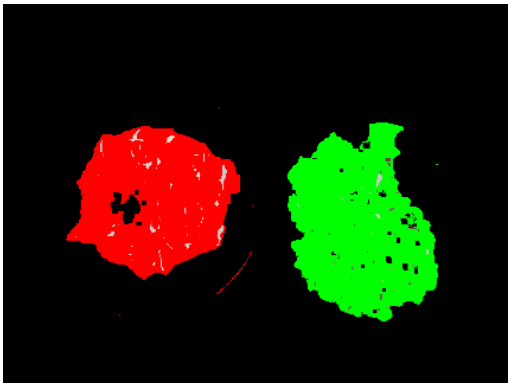

fish_cutlet-basil_potatoes
salad
fish_cutlet-basil_potatoes
fish_cutlet-basil_potatoes

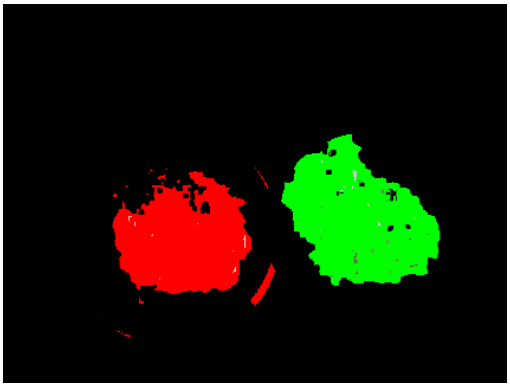



salad
salad

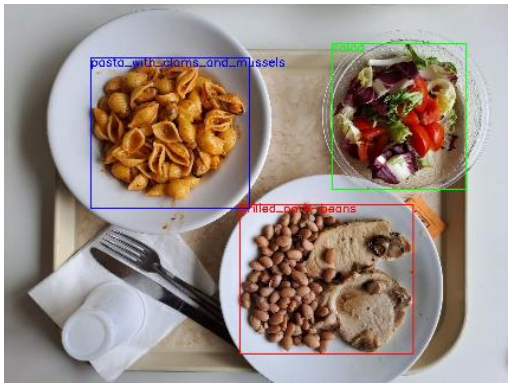


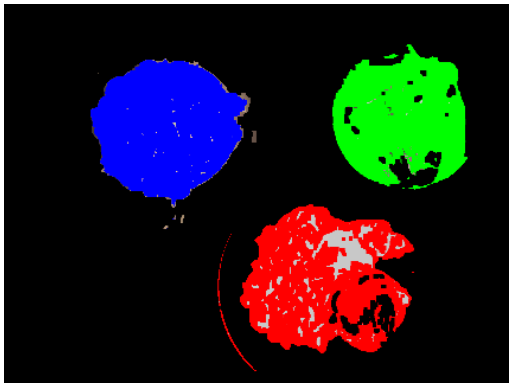
Tray 5:

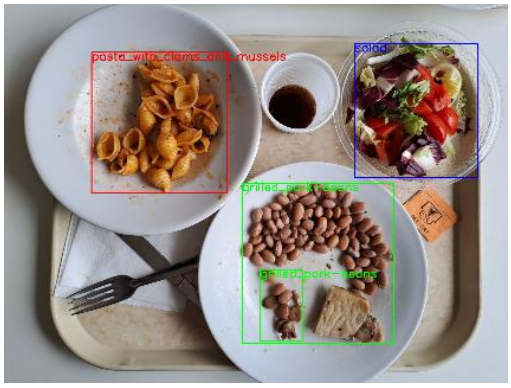


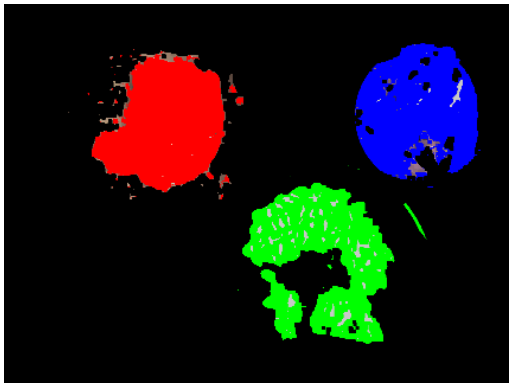


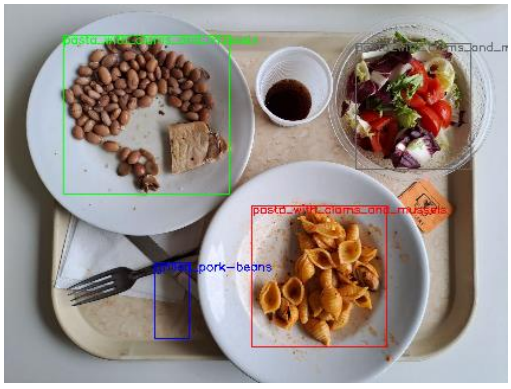
Tray 6:

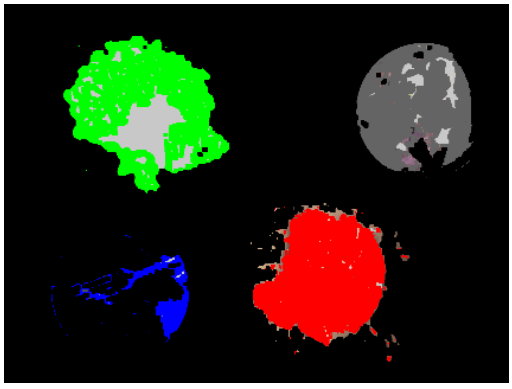

pasta_with_clams_and_mussels
salad
grilled_pork_beans



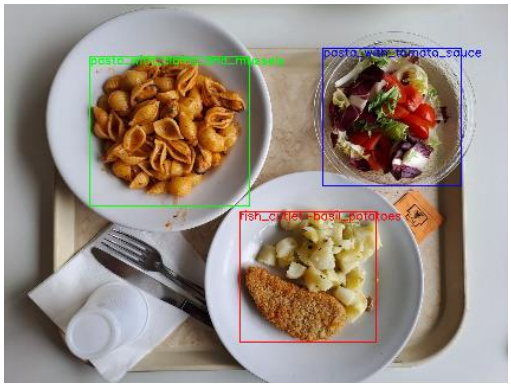

pasta_with_clams_and_mussels
salad
grilled_pork_beans

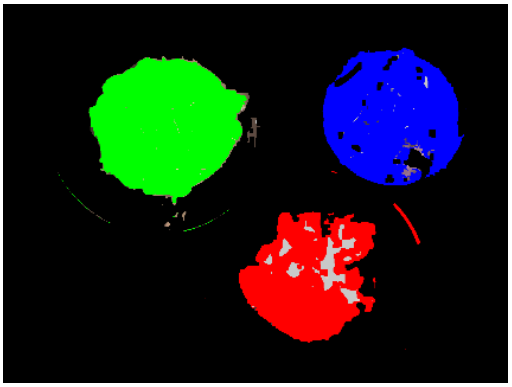


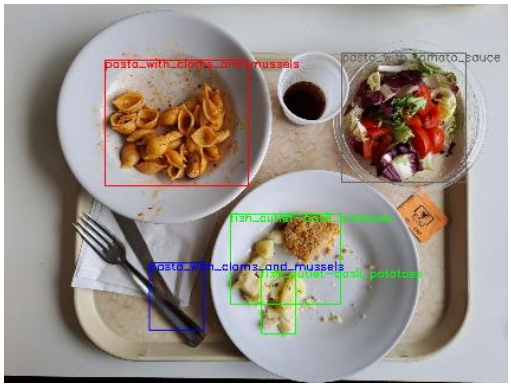

pasta_with_clams_and_mussels
grilled_pork_beans

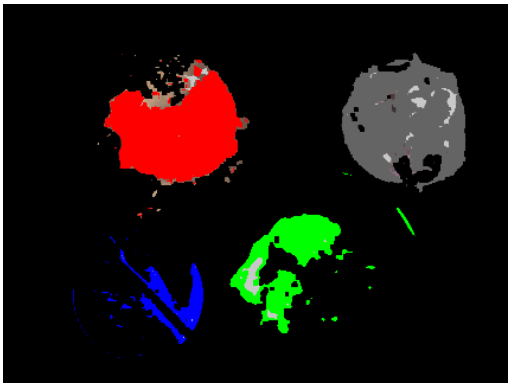


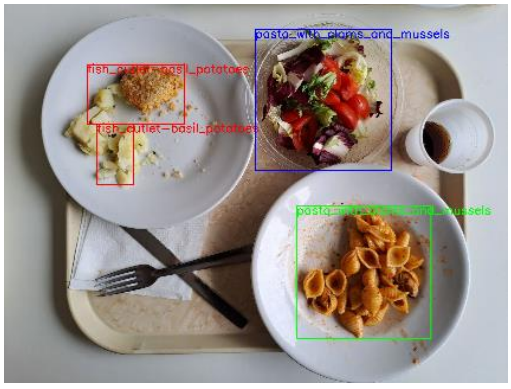
Tray 7:

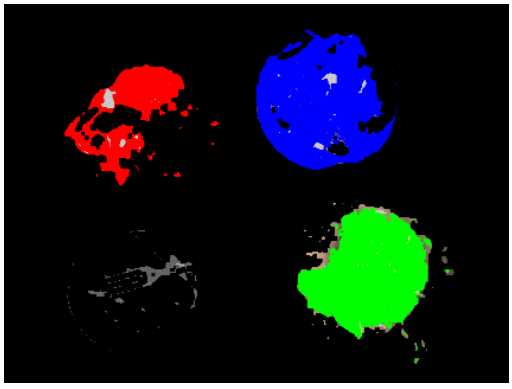

pasta_with_tomato_sauce
fish_cutlet_basil_potatoes



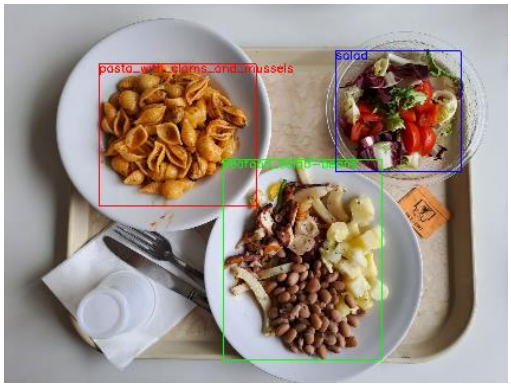

pasta_with_clams_and_mussels
pasta_with_tomato_sauce
pasta_with_clams_and_mussels

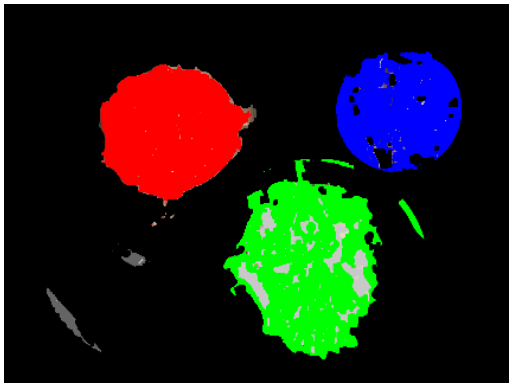


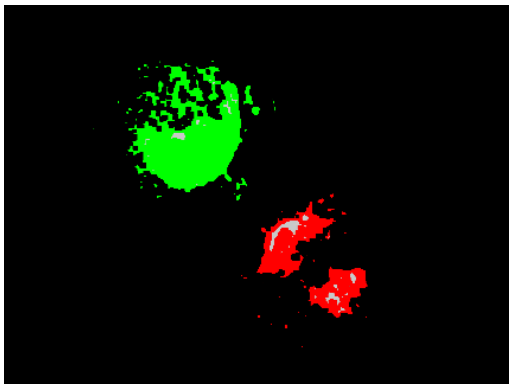

pasta_with_clams_and_mussels
fish_cutlet_basil_potatoes
fish_cutlet_basil_potatoes

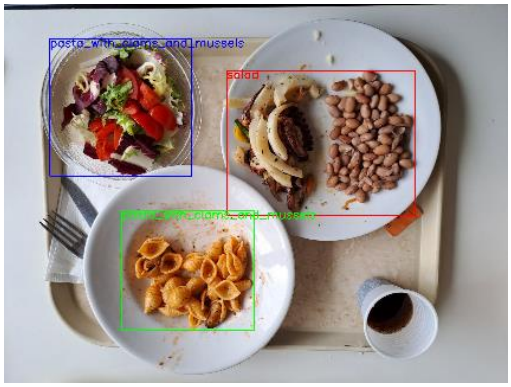


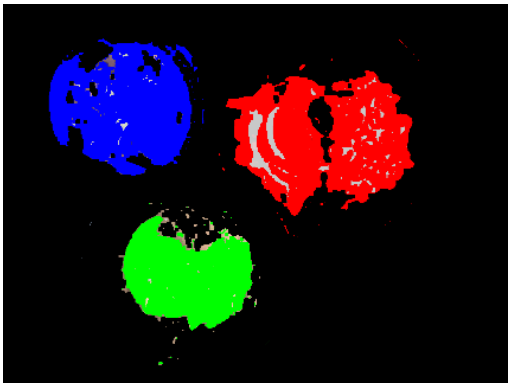
Tray 8:











III. Metrics results:

- Metric for “Food leftover estimation”:

	leftover1	leftover2
R1	0.8980	0.3525
R2	0.8567	0.4460
R3	0.7106	0.521
R4	1.327	0.547
R5	0.639	0
R6	0.7229	0.7750
R7	0.580	0.5157
R8	0.1972	0.7725

- Metric “mIoU” for segmentation:

	mIoU before	mIoU -leftover1	mIoU -leftover2
Tray 1	0.332593	0.332593	0.665186
Tray 2	0.468447	0.936894	1.40534
Tray 3	0.573891	1.14778	1.72167
Tray 4	0.248302	0.496603	0.744905
Tray 5	0.135472	0.270944	0.406415
Tray 6	0.514566	1.02913	1.5437
Tray 7	0.312635	0.625269	0.937904
Tray 8	0.389598	0.779195	1.16879

- Metric “mAP” for food Detection:
  For this metrics we’re not able to obtain a significative result since we have some problems with the implementation of the function to compute it. Anywhere we decide to leave our implementation on the library file “metricsLibs.cpp”. The function is “double mAP()”. The idea, since we do not have any image with multiclass in it, was to compute such metrics above all the 24 trays inside the GroundTruth folder.

IV. Hours of work:

Starting date: 27th of June 2023

First week: 5/6 hours per day for each member.

Second and Third week: 7/8 hours per day for each member.

NB: we start as a group of three people but the third member was mostly absent and decided to leave the group.