Zero Waste: a Computer Vision approach for food recognition and leftover estimation

Alberto Battiston Lorenzo Spagnolo Francesco Visentin

July 20, 2023

1 Introduction

Zero Waste is a computer vision system capable of scanning a canteen consumer's food tray at the end of the meal to estimate the amount of leftovers for each type of food. Zero Waste analyzes two main images: one before the meal and one at the end of the meal. From the first image the system is able to recognize the various types of food, keeping track of the initial quantity of each food. From the last image of the tray (the one taken at the end) the system recognize which types of food are still present and in which quantity, providing a leftover estimation. In particular, Zero Waste is able to recognize all food on the tray and to segment each food, associating it with a category.
To assess the system's performance, Zero Waste implements the mean Average Precision (mAP) and the mean Intersection over Union (mIoU) metrics, which provide a performance evaluation regarding the food recognition.
Zero Waste is developed in C++ and uses the OpenCV library [1].
In the following sections, a more detailed description of the structure of Zero Waste is given.

2 General Approach

As described before, some of the main goals of Zero Waste is food identification and classification. The system firstly detect regions containing food and then segment the foregorund, returning the portion of the image containing only the food. This region is then passed to the classification function, which assign a label to each food. Zero Waste approach is based on several operations performed from the first image of the tray.
All the images of each tray are processed sequentially. After a light noise removal procedure, the process of food recognition starts. Assuming that all the food is on the tray, Zero Waste consider three macro-categories that are

- Food-on-plates

[1] https://docs.opencv.org/

- Salad
- Food-out-of-plates

In this way, different features of these three categories can be exploited to facilitate the process of identifying and classifying food. For instance, by exploiting the shape and the size of the plates, Zero Waste is capable of distinguish plates that contains salad from plates that contains other types of food, helping also the classification procedure. For the food segmentation, Zero Waste exploit the saturation channel of the image, which contains important information.

2.1 Food-on-plates recognition pipeline

The overall processing pipeline for the *Food-on-plates* category is shown in Figure 1.

(a) Original tray (b) Detect plates (c) Plate cropped (d) Segmentation

Figure 1: processing pipeline for segmentation of *Food-on-plates*

The original tray image is firstly processed for detecting the circular shape of the plates, using the OpenCV *HoughCircles*. Then each plate (it is shown only an example) is cropped by means of a square region, built exploiting the detected circle shape. Finally the food is segmented using the *grabCut* function, that takes as input a binary mask for extracting the foreground (see below for more details).

2.2 Salad recognition (and classification) pipeline

The overall processing pipeline for the *Salad* category is shown in Figure 2.

(a) Original tray (b) Detect plates (c) Plate cropped (d) Segmentation

Figure 2: processing pipeline for segmentation of *Salad*

The pipeline follows the same flow as for the *Food-on-plates* pipeline, but some difference on the plates detection, n particular, due to the transparency of

the plate, it is more difficult to detect always the salad plates. For this reason, the solution to this problem is that of identify also false positives circles and, with a post-processing function, filter-out the circles based on the dimension (exploiting a feature of the salad plates).

2.3 Food-out-of-plates recognition (and classification) pipeline

This category is more complex to deal with since there are no objective information on which to base in order to identify the position of the foods. The overall processing pipeline for the *Food-out-of-plates* category is shown in Figure 3.

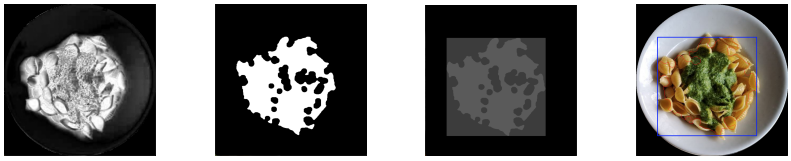
(a) Original tray (b) Regions (c) Colors (d) Segmentation

Figure 3: processing pipeline for segmentation of *Food-out-of-plates*

The *Food-out-of-plates* identification and segmentation procedure starts with removal of regions in which there are food belonging to the remaining macro-categories, by exploiting the detected plates computed in the previous steps, and a light noise reduction. Then the saturation channel of the resulting image is extracted and, using such channel, regions in which it is more likely the presence of food are computed, using a threshold on gray level range. Such regions are filtered based on some criteria, like dimension and color information, and, in the end, the segmentation is performed (see details below).

2.4 Food segmentation

Food segmentation is one of the main goals of Zero Waste. The following procedure is used for the segmentation of the two categories *Food-on-plates* and *Salad*. For the *Food-out-of-plates* category, another approach has been used. The segmentation pipeline is shown in the Figure 4



(a) Saturation (b) Mask (c) Grabcut (d) Segmentation

Figure 4: processing pipeline for food segmentation

As first step, the saturation channel is extracted from the Region Of Interest detected according to the previously described pipelines. Then it is created a binary mask, using a threshold on gray level range, which indicates which part of the plate more probably belongs to the food, useful for the segmenting function (more details below). Finally the segmentation mask is generated: such mask, applied to the original image, generates the final segmentation as shown in the last images of the previous pipelines.

2.5 Food classification

Another important goal of Zero Waste is that of assigning a class to each segmented food. Each food is assigned to one of the following classes:

- Background
- Pasta with pesto
- Pasta with tomato sauce
- Pasta with meat sauce
- Pasta with clams and mussels
- Pillow rice with peppers and peas
- Grilled pork cutlet
- Fish cutlet
- Rabbit
- Seafood salad
- Beans
- Basil potatoes
- Salad
- Bread

The previously defined macro-categories are used for the classification of Bread and Salad, by exploiting the information given by the ROIs extracted: in this way the identification and the classification are embedded into one single step. The approach adopted for this task, in the case of *Food-on-plates*, is described by the pipeline shown in Figure 5.

(a) Original tray (b) First course (c) Colors

Figure 5: processing pipeline for food classification

(a) Original tray (b) Second course (c) Colors

Figure 6: processing pipeline for food classification

The Zero Waste food classification strategy is based on color information of the plates. The key-point of the approach is that the food information is obtained from the initial tray image and then, this information, is stored and exploited in the subsequent images. From the initial image, each detected food is trivially classified into *first course* and *second course*, by a 4-patch subdivision of the plate region. The differences between the 4 patches is an index that helps to understand if the food is part of a first course or not. Intuitively, if the 4 patch of a plate are all very similar this means that the food is more probably a first course. On the other hand, if the patches have some marked difference, then with high probability the food belongs to a second course, since there are different foods on the same plate. Once understood this first classification, a label is assigned according to color information of the food. Given the relation between the label and the colors, it is easy to classify the food in the subsequent images.

3 Code Structure

Zero Waste is based on a main.cpp file supported by several OpenCV library headers and other project files, specifically created by us which are segmentation.hpp, classifier.hpp, metrics.hpp and common.hpp. These headers are needed to provide functionality for the food segmentation process and the analysis of the results.

3.1 getOutputImg function

This function is used to create an output image showing the results of processing the trays. It takes as input a series of images of the processed trays (trayOutputs), a series of masks indicating the detected foods (detectedFoodsMask) and a series of rectangles which delimit the detected foods for each tray (detectedItemsPerTray).The function also returns as output an image showing the trays with the foods detected, highlighting each one with rectangles.

3.2 processTray function

Only one tray is processed in this function. It takes as input the path to the directory containing the tray images (trayPath) and a reference of type out to store the output image. The function reads tray images from the specified directory, then performs a series of image preprocessing operations to obtain masks of the detected foods and rectangles of regions of interest (ROIs). The foods within the ROIs are then segmented and the results are stored in vectors. This function also calculates the required metrics:

- The mean Average Precision with the function averagePrecision
- The mean Intersection over Union with the function mIoU
- the leftover ratio R_i calling the function leftoverRatio

3.3 main function

This is the main function of the program. Checks if the path containing the trays on which to apply the program has been correctly supplied in input and, if not, displays an error message. If the path is valid, the program searches for directories starting with "tray" within the specified path. Each directory represents a tray. For each tray found, call the processTray function and display the resulting output image.

4 Libraries

This section presents the support files written for the main program and necessary for the correct execution of the program. The libraries used are the following:

- classifier.cpp
- metrics.cpp
- segmentation.cpp
- segmentation_utils.cpp

4.1 classifier.cpp

4.1.1 computeHist

This function calculates a color histogram for the input image src in BGR and HSV formats. Use a mask called mask to consider only the pixels of interest in the image. The BGR histogram is calculated separately for the Blue, Green and Red channels The HSV histogram is calculated separately for the Hue, Saturation and Value channels.

4.1.2 isFirstCourse

This function takes an image of a dish as input and determines whether it is a first or second course by proceeding as follows: calculates extract the saturation channel of the input image and theshold it to extract the main regions of interest, then splits the image into four quadrants and calculates an RGB and HSV histograms for each quadrant. Finally, using the Euclidean distance, computes the average distance between the histograms of the 4 quadrants to determine, based on a threshold, whether the image is a first plate or a second plate.

4.1.3 classifyPlate

This function classifies an input dish using its dominant color and k-means. More precisely it starts by extracting an saturation mask for the dish and refines it via morphological operations. Then considering only the pixel within the mask, calculates a RGB features vector for each pixel and applies k-means clustering to obtain the three dominant colors inside the image. Finally calculates the distance between the three dominant colors and the various reference dish colors (saved in dishRefColors). The closest triplet of reference colors is used determine the content of the plate and to assign an ID to the each dish.

4.1.4 classifyAndUpdate

This function classifies the dishes present in the considered image. In fact, the function receives as input an image, a list of ROIs (Region of Interest) representing the plates and the relative masks (contained respectively in platesROI and platesMask). The function calculates histograms and assigns initial labels to dishes or updates histograms and labels based on images of previous dishes in the tray. Use the computeHist and classifyPlate functions, explained above, to compute histograms and assign labels based on dominant colors.

4.2 metrics.cpp

4.2.1 mAP

This function calculates the " Mean Average Precision" (mAP) for each tray in the input data set. For each tray the function considers the first three images and computes the "Average precision" (AP) for each dish class detected in theese

three images. Average Precision measures how well the algorithm correctly ranks dishes based on their class. To calculate the AP, the "ground truths" provided and the predictions made by the algorithm are used, calculating the precision and recall metrics. Finally, these scores are averaged to obtain the overall mAP for each tray.

4.2.2 mIoU

This function calculates the "mean Intersection over Union" for each tray image. The mIoU metric measures how well the algorithm successfully segments the plates in the image. To calculate the mIoU, the algorithm's prediction masks and ground truth masks are used to compare the overlapping areas of the segmented plates. As for the mAP, the mIoU is also calculated by averaging all the classes of the individual IoUs obtained.

4.2.3 leftoverRatio

This function calculates the ratio between the areas of dishes detected in the original tray image and those detected in the leftover images. For each item detected in each one of the leftover images the pixel count ratio is computed always with respect to the original base image.

4.3 segmentation.cpp

4.3.1 detectMainComponents

Helper function used by the segmentation and detection function described below. Given an input color image and saturation range the function:

1. smooths the image and extract its saturation channel
2. create a binary mask extracting the values inside the input saturation range and refine the mask with some morphological operators
3. selects the main components inside the binary mask and use them to define a second mask (similar to the one defined in specifically initialized to be used as the input mask for the grubCut segmentation algorithm

4.3.2 grabCutSeg

Uses the mask produced by detectMainComponents to run the standard opencv grabCut implementation and processes the returned mask to label the area detected as foreground with the desired food ID passed as input

4.3.3 Segmentation functions

These functions take a grayscale image as input and return:

- getPlatesROI: the mask of the regions of interest of the plates, together with the corresponding rectangles.
- getSaladROI: the salad regions of interest mask, along with the corresponding rectangles.

4.3.4 Segmentation and detection functions

The first two functions take as input a color image together with an array of regions of interest used to mark the plates detected by the above functions while the third functions, together with the input color image, receives a mask used to filter out the plates and salad bowls:

- segmentAndDetectPlates: segment each item inside the plates using the food segmentation and classification pipeline described in section 2.4 and 2.5
- segmentSalad: segment the salad within the regions of interest of the tray using a similar saturation based approach.
- segmentAndDetectBread: detect the plates using the food-out-of-plate recognition and classification pipeline.

4.4 segmentation_utils.cpp

4.4.1 filterCircles

This function is used to filter the circles found by the OpenCV native HoughCircles algorithm. Takes a vector of circles as input and returns a filtered vector. Filtered circles must meet a certain condition on the radius size. In this case, circles with a radius between 186 and 240 are treated as valid circles and added to the filtered vector.

4.4.2 filterBox

This function is used to filter bounding boxes based on their area and width-to-height ratio. Takes a reference to a rectangle as input and returns true or false. Rectangles with an area greater than 30000 and with a width-to-height ratio less than 2.5 for horizontal rectangles or less than 1.8 for vertical rectangles are considered valid and are kept. Otherwise, they are discarded.

5 Results

In this section are reported the Zero Waste outputs for each tray in the provided test dataset.

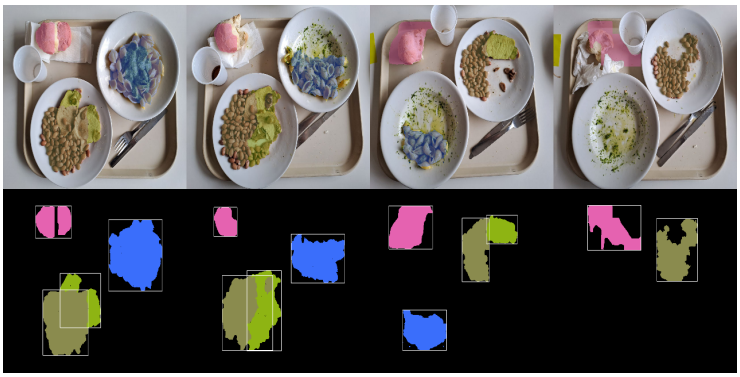


Figure 7: Tray 1

Figure 8: Tray 2

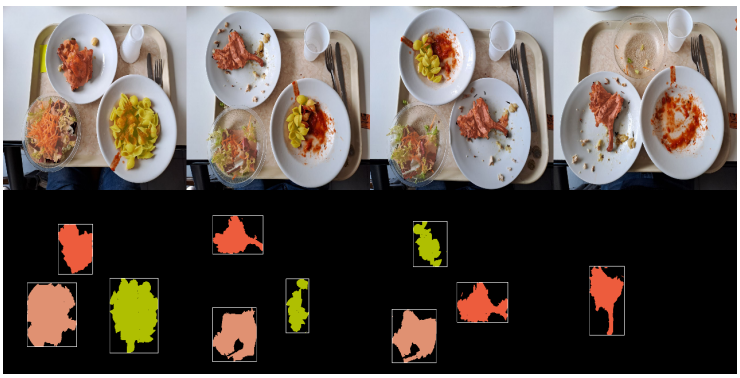


Figure 9: Tray 3

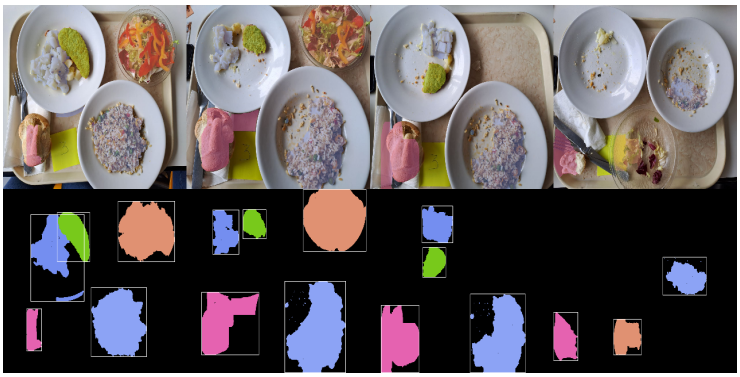


Figure 10: Tray 4

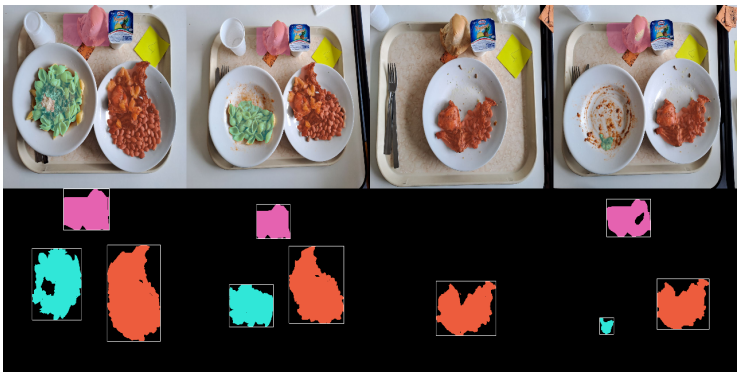


Figure 11: Tray 5

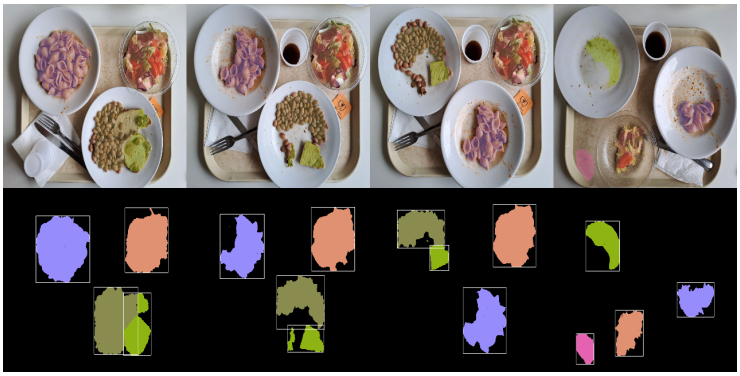


Figure 12: Tray 6

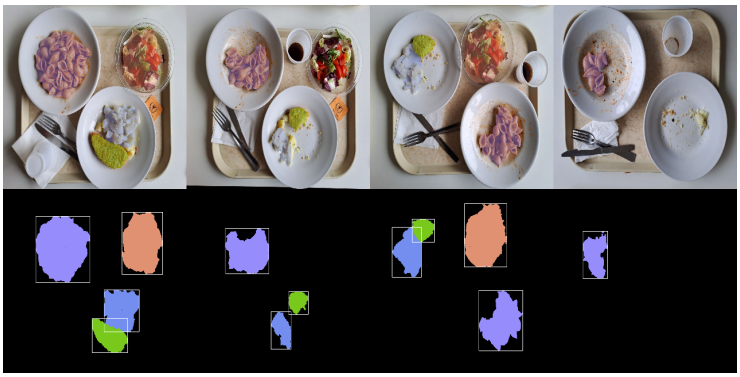


Figure 13: Tray 7

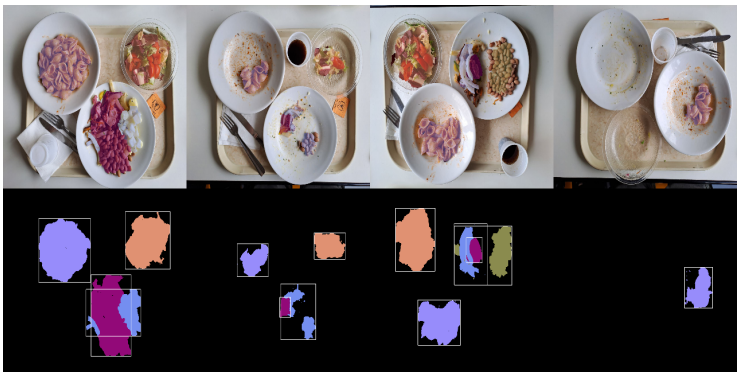


Figure 14: Tray 8

6 Metrics

In this section are reported the metrics computed for each input tray of the sample data set provided.
As described above, a single mAP score has been computed for each tray (considering the base "food_image.jpg", "leftover1.jpg" and "lefotver2.jpg") while mIoU and Ri have been compute for each pair of ground-truth/zeroWaste-output image (mIoU, similarly to mAP considers only the first two leftovers).
The results obtained for each tray are reported below and can also be found inside the "inputs" sub directory contained in the project structure.

6.1 Tray 1

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 4 detected items in the tray: mAP = 0.0909091
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.862037:
   - Img 2 mIoU = 0.685008:
   - Img 3 mIoU = 0.653488:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 1 Ri = 0.660958
   - ID: 6 Ri = 2.17043
   - ID: 10 Ri = 0.765323
   - ID: 13 Ri = 0.626049

   Leftover2:
   - ID: 1 Ri = 0.461156
   - ID: 6 Ri = 0.958899
   - ID: 10 Ri = 0.521623
   - ID: 13 Ri = 1.56553

   Leftover3:
   - ID: 10 Ri = 0.637127
   - ID: 13 Ri = 1.48366

6.2 Tray 2

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 4 detected items in the tray: mAP = 0.0681818
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.827828:
   - Img 2 mIoU = 0.779992:
   - Img 3 mIoU = 0.748762:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 2 Ri = 0.992612
   - ID: 7 Ri = 1.0547
   - ID: 11 Ri = 1.01459
   - ID: 12 Ri = 1.08898

   Leftover2:
   - ID: 2 Ri = 0.326874
   - ID: 7 Ri = 0.732046
   - ID: 11 Ri = 1.01327
   - ID: 12 Ri = 0.771589

   Leftover3:
   - ID: 2 Ri = 0.147034
   - ID: 7 Ri = 0.416762
   - ID: 11 Ri = 0.436597

6.3 Tray 3

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 3 detected items in the tray: mAP = 0.0909091
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.875761:
   - Img 2 mIoU = 0.841716:
   - Img 3 mIoU = 0.776015:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 2 Ri = 0.259705

- ID: 8 Ri = 0.800348
- ID: 12 Ri = 0.629695

Leftover2:

- ID: 2 Ri = 0.283421
- ID: 8 Ri = 0.938467
- ID: 12 Ri = 0.600767

Leftover3:

- ID: 8 Ri = 0.924945

6.4 Tray 4

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 5 detected items in the tray: mAP = 0.0727273
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.735657:
   - Img 2 mIoU = 0.788862:
   - Img 3 mIoU = 0.792735:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 5 Ri = 1.31292
   - ID: 7 Ri = 0.556622
   - ID: 11 Ri = 0.52218
   - ID: 12 Ri = 1.2472
   - ID: 13 Ri = 4.5357

   Leftover2:
   - ID: 5 Ri = 1.02974
   - ID: 7 Ri = 0.584956
   - ID: 11 Ri = 0.539127
   - ID: 13 Ri = 4.28319

   Leftover3:
   - ID: 5 Ri = 0.332986
   - ID: 12 Ri = 0.304156
   - ID: 13 Ri = 1.7757

6.5 Tray 5

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 3 detected items in the tray: mAP = 0.0909091
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.502152:
   - Img 2 mIoU = 0.51807:
   - Img 3 mIoU = 0.13452:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 3 Ri = 0.613707
   - ID: 10 Ri = 0.705997
   - ID: 13 Ri = 0.620503

   Leftover2:
   - ID: 10 Ri = 0.559825

   Leftover3:
   - ID: 3 Ri = 0.0588763
   - ID: 10 Ri = 0.491638
   - ID: 13 Ri = 0.774408

6.6 Tray 6

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 4 detected items in the tray: mAP = 0.0909091
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.799353:
   - Img 2 mIoU = 0.793381:
   - Img 3 mIoU = 0.793332:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 4 Ri = 0.603583
   - ID: 6 Ri = 0.568884
   - ID: 10 Ri = 0.68408
   - ID: 12 Ri = 0.978166

Leftover2:

- ID: 4 Ri = 0.619472
- ID: 6 Ri = 0.390221
- ID: 10 Ri = 0.60654
- ID: 12 Ri = 0.960498

Leftover3:

- ID: 4 Ri = 0.321279
- ID: 6 Ri = 0.947311
- ID: 13 Ri = 0.424432

6.7 Tray 7

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 4 detected items in the tray: mAP = 0.0909091
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.839546:
   - Img 2 mIoU = 0.59847:
   - Img 3 mIoU = 0.95073:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 4 Ri = 0.551097
   - ID: 7 Ri = 0.427401
   - ID: 11 Ri = 0.374774

   Leftover2:
   - ID: 4 Ri = 0.558169
   - ID: 7 Ri = 0.495108
   - ID: 11 Ri = 0.759198
   - ID: 12 Ri = 1.04819

   Leftover3:
   - ID: 4 Ri = 0.280453

6.8 Tray 8

1. FOOD LOCALIZATION (mAP):
   Mean average precision over 5 detected items in the tray: mAP = 0.0363636
2. FOOD SEGMENTATION (mIoU):
   - Img 1 mIoU = 0.558785:
   - Img 2 mIoU = 0.468061:
   - Img 3 mIoU = 0.587059:
3. FOOD LEFTOVER ESTIMATION:
   Leftover1:
   - ID: 4 Ri = 0.233769
   - ID: 9 Ri = 0.072546
   - ID: 11 Ri = 0.55979
   - ID: 12 Ri = 0.378125

   Leftover2:
   - ID: 4 Ri = 0.597806
   - ID: 9 Ri = 0.106171
   - ID: 11 Ri = 0.669512
   - ID: 12 Ri = 1.02916

   Leftover3:
   - ID: 4 Ri = 0.278743

7 Contribution of Each Member

In this section are reported the various contributions of each member of the group, with the related hours dedicated to the project. Many sections of the code were the result of contributions from the whole group, while others were written individually.

7.1 Alberto Battiston

- Food segmentation: plates recognition, Grab-cut segmentation (idea and test) with binary masks.
- Salad plates recognition and filtering.
- Sections "General approach" and "Results" of the report.
- Minor code fixing.

Time dedicated: 100 - 120 hours

7.2 Lorenzo Spagnolo

- study and part of the code for The mean Average Precision (mAP).
- study and part of the code for The mean Intersection over Union (mIoU).
- writing part of the report, particularly the sections Introduction, Code Structure and Libraries.
- research and test different combinations for the parameters used in the Hough transform, until finding the optimal ones used.

Time dedicated: 70 hours

7.3 Francesco Visentin

- Bulk of code implementation, whole main.cpp and general code organization/implementation for all the src and include files
- Main food segmentation implementation, food saturation ranges for all classes.
- Food classification: whole approach (whole classifier.cpp/hpp implementation) definition and implementation both for in-plate and for out-of-plate food item recognition
- Metric: only code implementation following the given models
- Lots of test classes for reaching the final submitted approaches (like really a lot)

Time dedicated: 180 hours