Computer Vision Project

Written by David Petrovic, Federico Luisetto, Umberto Salviati

Food recognition and leftover estimation

Project Description

The objective of this project is to create a computer vision system with the capability to scan a canteen consumer's food tray after a meal and estimate the amount of leftovers for each type of food. The system works by analyzing pairs of images: one taken before the meal and one after the meal. In the initial image, the system identifies the different types of food ordered and records their initial quantities. At the end of the meal, the system analyzes a new image of the tray, recognizing the remaining types of food and their respective quantities.
More precisely, the project is divided into 3 main part:

1. food recognition and localization in the tray images based on the given dataset
2. food segmentation for computing the quantity of each aliment
3. comparison between the two given images (“before meal” and “after meal”) in order to find which food was eaten and which was not.

Project Implementation

In the images there is a lot of “noise” (something we are not interested in) so in order to achieve the goals we have decided to work separately in different portions of the images. First of all, since the food is placed inside the plates our system extracts the plates and segments the food inside each plate one by one.
We have tried a lot of techniques for segmenting the food but we did not find a method that works well with all food items. Thus, we have chosen to divide the food by looking at the color. However, this is not a “direct” way since the original colors do not allow to segment the aliments because often they have very similar paint. Therefore, we have emphasized the differences between the food items by increasing the saturation of images. Then, for each aliment present in the dataset, we have selected the range of colors of each channel and used a filter for applying segmentation.
At this point, we need to know which food items are on the plate in which we want to apply segmentation. We have tried different approaches but no one worked well in our case.
After some searches, we decided to use a neural network model. To be more precise, we have used CLIP (Contrastive Language-Image Pre-Training) which is a neural network introduced by OpenAI that receives as input an image and a list of labels and for each label returns the probability that the item characterized by that label is present in the image. We have applied a “training phase” in natural language for choosing the labels to pass to the model because this type of neural network could be instructed by doing this, without directly optimizing for the specific task.

Considering the fact that bread in the images is not in a plate and could be all over the tray, we segment it in a different way with respect to the other food items.
We have decided to look at the texture of the images in order to locate the bread in the tray but in the texture of the original images there are some things that produce noise, for example the plates and the glass.
In order to fix this problem, we have decided to cover with a black circle all the elements that produce noise and then look at the texture for segmenting the bread (if it is present in the image).
For what concerns the salad, it is simpler to locate it because the radius of its plate is lower than the one of the other plates.

For doing this, our project is composed by some macro tasks:

Circles extraction

This task is used for finding all the plates in the input images in order to:

- create the images with black background and one plate with its content;
- cover the “noise” for the segmentation of the bread.

For this task, the Hough transform with different parameters as input was used to detect the circles in the tray.

Bread extraction

To extract the bread from the dataset images, we employed a texture-based approach. First, we applied a gamma transform and histogram equalization to preprocess each image. This step enhanced the overall contrast and homogenized our input for the algorithm, ensuring better results when we employed the subsequent techniques.

Next, we experimented with different methods for bread detection. Initially, we explored using entropy, but after several hours of testing, we found it to be less effective than Niblack thresholding. Niblack offered more user-friendly parameters and delivered a more promising identification of the bread.

While Niblack thresholding provided good results overall, there were some areas where the precision was lacking. To tackle this issue, we incorporated the GrabCut algorithm, utilizing the previously obtained mask. By refining and cleaning the mask, we used it as the input mask for the GrabCut algorithm's INIT_WITH_MASK parameter. This further segmented the bread from the background, enhancing the accuracy of our extraction process.

However, we observed that certain pain trays in the canteen were more sensitive to the viewpoint of the Niblack method, leading to false positives. To mitigate this, we

selectively removed some masks when the bounding box of the GrabCut mask appeared excessively filled. This helped eliminate the false positives originating from the tray's texture.

In summary, our approach involved preprocessing each image with a gamma transform and histogram equalization to create a more uniform input for the algorithm. Then, we combined the saturation mask from the HSV color space with the Niblack thresholding mask to identify the bread. The GrabCut algorithm was then employed to refine the segmentation. Despite encountering some challenges, such as false positives from sensitive tray textures, our method achieved a relatively accurate detection of all bread items in the images taken at our university canteen.

As we can observe from the metrics, the Mean Intersection over Union (MIoU) of the bread is among the best results. In fact, we can confidently state that it is the best-performing class, even compared to the salad, which also exhibits a good performance value. The bread class demonstrates consistently correct classification even when it is present in multiple trays. This indicates that our bread extraction approach has been successful in accurately identifying and segmenting the bread items in the images from

Salad Extraction

To extract the salad, we employed the simplest approach that we have discovered so far.
Since the plate of the salad is different from the other plates, we have looked at the radius of all of them and we have found that the radius of the salad’s plate is smaller. So we are able to detect the salad’s plate just by looking at the radius of the circles returning by the Hough transform.
We began by creating a mask, ensuring that we performed some preprocessing steps to achieve a homogenous input. Using the HSV color space, we applied a straightforward saturation mask, followed by some morphological operations to refine and enhance the precision of the output. This technique proved to be fairly consistent and yielded satisfactory results in our salad extraction process.

As we review the results, we notice that the salad's mIoU has the one of highest value among all the foods present in the trays. This indicates that the segmentation performance for the salad is the best among them all.

CLIP

CLIP is a neural network that efficiently learns visual concepts from natural language supervision.
In github there is a repository in which there are the basic functions needed for using the model. Since the model uses the provided labels for making predictions, it was necessary a “training phase” in order to choose the best ones.
One of the problems was to find the right name since the labels must be in English and the food items are italian.
Another problem was to manage the communication between python and c++ (and vice versa), since the python script is launched from the c++ code. This problem was fixed by using txt files in which both write and read.
Also, we have to manage the constraints, for example in the image taken before the meal there must be at most four food items: one of the first plates, one for the second plate and at most two of the third plates. It was quite difficult to write all the constraints useful for making the right predictions, for example for making a prediction in an image taken after a meal, we wanted the model to make it consistent with respect to the prediction taken before meal. Also, since the images of the plates with black background are processed one at a time, we wanted the predictions to be consistent with respect to the whole tray image.

Segmentation of the plate food

Segmenting the food was the most challenging aspect of the whole project, requiring hundreds of hours of trial and error. Initially, we attempted various machine learning algorithms, such as YOLO, but found that a hybrid approach involving both ML and color segmentation yielded better results. Our process involved using CLIP to predict the food in a plate and then segmenting the food based on the colors present in each plate. Although this approach showed promising results, it did not achieve 100 percent accuracy, demonstrating the difficulty of segmenting different types of food on the same plate.

Our approach focused on taking single plate images and segmenting them one at a time using color-based segmentation. We utilized the "inRange" function, allowing us to extract specific colors from the input BGR values within the given minimum and maximum thresholds. You can find all the specific values used in the "Segmentedfood.cpp" file. Despite the challenges, our work provides valuable insights into the complexities of food segmentation and distinguishes the need for further improvement in this area.

Bounding Box

In our food tray image segmentation project, we generate the bounding boxes around the segmented food regions by thresholding the combined mask for each image. This approach is utilized to evaluate and refine the segmentation results.
As previously mentioned, the bounding boxes are not only used during the segmentation process but also in the final evaluation of the cleaning action on the image. After generating the bounding boxes, the coordinates and dimensions of these boxes are recorded and saved in a text file for further analysis and documentation.

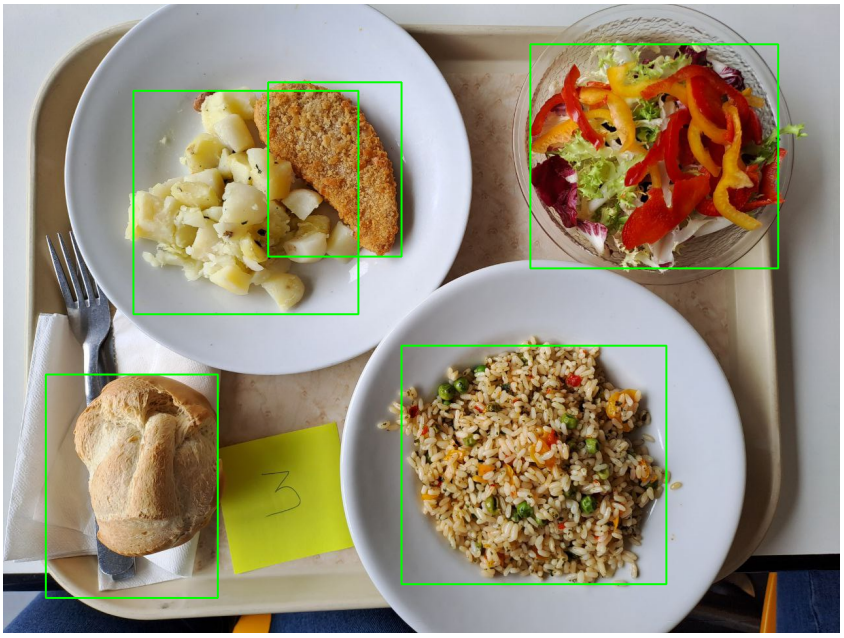
In the .txt files, we follow a standardized format to output the bounding boxes, which mimics the assignment dataset format. This format typically consists of lines for each bounding box, where each line contains the relevant information for a single box.

The standard format for each line in the .txt file is as follows:

<class_label> <x_center> <y_center> <width> <height>

Where:

<class_label> represents the label or class of the segmented food item.
<x_center> and <y_center> indicate the normalized coordinates of the box's center point with respect to the image dimensions.
<width> and <height> represent the normalized dimensions of the bounding box, again relative to the image size.
By adhering to this standard format, we ensure consistency and compatibility with the assignment dataset and facilitate easier evaluation and analysis of the bounding box outputs. This organized structure enables further processing and comparison with ground-truth data, which is essential for measuring the performance of our segmentation model.



Performance measurement

mAP

In the mean Average Precision performance measurement we divided the problem in two subproblems:

- detecting true positive/false positive for predicted bounding boxes.
- calculating average precision for each class.

In the first subproblem we have used a map to keep track of the list of true positives/false positives for each class. A true positive bounding box is the one with an Intersection over Union (with ground truth bounding box) over the threshold of 0,5. Then for each class it calculates the average precision, returning the latter divided by the size of the map to calculate the mean.
Result of mAP: 0.566239

mIoU

In the mean Intersection over Union performance measurement we use two arrays of size N_CLASSES (N_CLASSES = number of classes segmented) to keep track of the pixels of a given class (class number = array index). For every mask image, we compare the ground truth mask and the predicted mask pixel by pixel, incrementing the intersection array if the two pixels have the same value (class), or implementing the not intersection array of both pixels if they have different values (class). Then we use the two arrays to compute the mean intersection over union of each class.
Results:
mIoU for class 0: 0.937827
mIoU for class 1: 0.583389
mIoU for class 2: 0.355873
mIoU for class 3: 0.86535
mIoU for class 4: 0.736324
mIoU for class 5: 0.854773
mIoU for class 6: 0.322954
mIoU for class 7: 0.661467
mIoU for class 8: 0.35157
mIoU for class 9: 0.398029
mIoU for class 10: 0.507972
mIoU for class 11: 0.668223
mIoU for class 12: 0.854927
mIoU for class 13: 0.756953

Food leftover estimation

In the food leftover estimation performance measurement we use two arrays of size N_CLASSES (N_CLASSES = number of classes segmented) to keep track of the pixels of a given class (class number = array index).
We compute the food leftover estimation over this three masks couples:

- food_image_mask and leftover1.
- food_image_mask and leftover2.
- food_image_mask and leftover3.

Each performance of the couples is computed over every tray by counting the number of pixels of a given class, saving it in bfarr[] if belonging to before image, afarr[] otherwise. Then we compute the ratio of these two arrays over every class detected during the segmentation process.
Result of food leftover estimation is saved in data/result/results.txt

About the project

	Working hours*	Ideas / Implementation**
David Petrovic	130	circles extraction and covering python script interaction between python and c++ softwares other tests not implemented in the final solution
Federico Luisetto	~50	plate extraction through HoughCircles performance measurements other tests not implemented in the final solution
Umberto Salviati	180	machine learning tests bread segmentation salad segmentation food color segmentation other tests not implemented in the final solution

*The hours include the time dedicated to search and try possible ways to reach the goals, not only the final solution.
**Most of the time, the ideas was provided during a brainstorming session, in which all the group members participate by discussing possible solutions.

HOW TO RUN OUR CODE

In order to run our code, it is necessary to run the following commands:

cd out/build/x64-release

cmake ../../..

make

./CV-proj

The latter starts the execution of the project which will process all the images in the dataset.

The **final results** will be written in the folder “data/result”, “data” folder is at the same level as “out”.

Conclusion

In conclusion, our hybrid approach for food tray image segmentation has proven to be effective, but there is room for further improvement. By selecting more precise and accurate labels to feed into CLIP or using a machine learning algorithm specifically trained to our problem, we can refine our project even more. The validity of our idea is evident, and with additional data and time, we could achieve even better results.

As seen in the CLIP labels, we chose not to refine the labels extensively to avoid overfitting our trays. Overfitting could lead to a model that performs well on the training data but poorly on unseen data, which is not desirable in our context.

In the future, conducting more experiments with a diverse and larger dataset could help enhance the accuracy and generalizability of our segmentation model. Additionally, exploring more advanced techniques and algorithms can contribute to further refining our food tray image segmentation project, making it more robust and efficient.