Intermediate Project Report

Chinello Alessandro, Piai Luca, Scantamburlo Mattia

Academic Year 2024-2025

Contents

1 Overview and project description

The members of this group ("Gli ultimi Sumiti") are the students Scantamburlo Mattia, Piai Luca, and Chinello Alessandro. In the following file, we have created a brief report on the project we carried out, explaining how we organized ourselves, the basic idea adopted to structure the code, the division of the work, the results obtained, and the main issues encountered. The project source code can be found at the following link.

1.1 How to compile and execute the code

To compile:

```
mkdir build && cd build
cmake .. && make
cd ..
```

To execute:

```
./obj_detector -p <path> -m <path> -s <path> -i <path> -l <path>
```

Where:
- -p is the power drill models dir path
- -m is the mustard bottle models dir path
- -s is the sugar box models dir path
- -i is the input scene image path
- -l is the label path associated with the scene

Example:

```
./obj_detector
-p ../data/035_power_drill/models
-m ../data/006_mustard_bottle/models
-s ../data/004_sugar_box/models
-i ../data/004_sugar_box/test_images/4_0001_000121-color.jpg
-l ../data/004_sugar_box/labels/4_0001_000121-box.txt
```

IMPORTANT NOTE: The models directories must not contain the masks but only the RGB views.

1.2 Project Organization

We organized our pipeline for the software intended to detect certain objects (such as a sugar box, mustard, and a drill) in images called *scenes*, following the logic described below:

1. Given a specific dataset for each individual object, containing images portraying different perspectives and angles, in the initial phase we focused on extracting features from each category of the dataset, storing them using appropriate practices. As the feature detection algorithm, we relied on SIFT, after experimenting with various alternatives such as segmentation combined with SURF, and even completely different approaches like template matching. It should be noted that for our work, the use of masks was not necessary.

2. Subsequently, we used this data to perform the matching with the scene provided as input by the user, following a sequential procedure: given a scene, the software first attempts to detect the sugar box, then the mustard, and finally the drill; in case of detection, a *bounding box* of the corresponding detection is displayed, and the related coordinates (top left and bottom right), according to the dataset labeling conventions, are saved in a dedicated file.

3. Finally, by processing the dataset labels together with the data returned by our software, we computed the evaluation metrics into a final file, referring to the analyzed scene. Testing the program across all scenes ultimately allowed us to obtain the overall mIoU for each object class.

Main structure of the algorithm For simplicity consider that we want to detect the presence of a generic object in a given scene. The steps performed are:

1. **Extract features and the descriptors** from the scene image and from each model view given in input (30 in our case). For this step we have used SIFT.

2. **Match the features** between each model view and the scene image and store the scene matched points. The matching strategy used is FLANN.

3. **First filter:** the matched points are filtered using the Lowe's ratio test with threshold value T_1.

4. **Second filter:** compute the center of mass (COM) of the matched points and neglect the ones that have a distance bigger than T_2 from the COM.

5. **Third filter:** now consider the points survived from the previous step. For each point we compute the number of points that have a distance from it that is below T_3. That is, we compute the number of neighbors of each point. If the number of neighbors is below T_4, then the point is neglected.

6. Compute the top left corner and the bottom right corner of the smallest box that contains all the survived points. Compute also the area of the box.

7. Define the *density* := number of survived points / area of the box. Iff density $\geq T_5$ then the box is drawn.

The steps are repeated sequentially for each object class (poewr drill, sugar box and mustard bottle). For each object class we have defined 5 parameters T_1, T_2, T_3, T_4, T_5, and each one of them was determined empirically.

1.3 How we organized the work

The work can be divided into several temporal macro-areas, which followed one another sequentially:

- An initial phase of discussion regarding the implementation organization;
- A phase of experimentation, during which we tested (in parallel) various strategies to identify the most effective one based on the available samples in the dataset;
- A phase of meetings and briefings where we moved on to the actual structuring of the code, dividing the development in terms of classes and specific implementations;
- A final phase of complete revision of the structure, with the introduction of improvements and modifications.

As for the time dedicated, we did not precisely track the hours each day, but we met for five days: on Friday and Saturday before Easter, working respectively one afternoon and the entire Saturday, and continuously from Tuesday afternoon every day for about 8 hours on average per day until Saturday, April 26th, in the afternoon. We always worked while staying synchronized using Discord as our meeting point.

Why all files have all members as authors Regarding the division of the files, it is difficult for us to indicate true authors since multiple people worked on the same file following a pseudo-agile approach. As mentioned earlier, we also point out that some parts of the code were deleted after noticing that alternative approaches performed poorly, even if we spend much time on it. Also for those reasons we have decided to put the name of each member in every file.

1.4 Issues Encountered and problems of the algorithm

The main difficulties arose during the second phase (the second point mentioned above), where we became "scattered" in trying to find the best approach. Whatever direction we took in terms of matching with the real scenes, we were never fully satisfied with the results. This was likely also due to the limited number of samples available for each object in the dataset.

The algorithm that we have developed has many problems. The biggest one is that we had to tune the values of many parameters and finding the optimal ones was a difficult task to do by hand. As a result the algorithm probably has bad performance with completely different scene. This problem can be seen also in the results section. In figure 16 the algorithm detect the presence of the mustard bottle but in figure 26 it doesn't, although the two scenes are almost identical.

2 Results and Metrics

Metrics:

- mIoU for sugar box = 0.5827, mIoU for power drill = 0.4022, mIoU for mustard bottle = 0.2455
- Accuracy = $\frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} = \frac{20+44}{20+48+33+4} \approx 0.63$

powerdrill() + mustard()
In the following, the image number is referred to the images reported below in the document.

Listing 1: Sotware log file for the metrics

```
Figure1:
        IoU of Sugar = 0.5286 is true positive
        IoU of Mustard = 0.0000 is NOT a true positive

Figure2:
        IoU of Sugar = 0.5097 is true positive
        IoU of Mustard = 0.0385 is NOT a true positive

Figure3:
        IoU of Sugar = 0.7051 is true positive

Figure4:
        IoU of Sugar = 0.6280 is true positive

Figure5:
        IoU of Sugar = 0.7014 is true positive

Figure6:
        IoU of Sugar = 0.6391 is true positive

Figure7:
        IoU of Sugar = 0.8218 is true positive
        IoU of Power Drill = 0.1576 is NOT a true positive

Figure8:
        IoU of Sugar = 0.4447 is NOT a true positive

Figure9:
        IoU of Sugar = 0.4414 is NOT a true positive

Figure10:
        IoU of Sugar = 0.6484 is true positive
        IoU of Power Drill = 0.0442 is NOT a true positive

Figure11:
        IoU of Power Drill = 0.5155 is true positive

Figure12:
        IoU of Power Drill = 0.5587 is true positive

Figure13:
        IoU of Power Drill = 0.2427 is NOT a true positive

Figure14:
        IoU of Power Drill = 0.7147 is true positive

Figure15:
        IoU of Power Drill = 0.3443 is NOT a true positive

Figure16:
```

```
        IoU of Mustard = 0.3708 is NOT a true positive
        IoU of Power Drill = 0.7169 is true positive

Figure17:
        IoU of Mustard = 0.1679 is NOT a true positive
        IoU of Power Drill = 0.0779 is NOT a true positive

Figure18:
        IoU of Power Drill = 0.6474 is true positive

Figure19:
        IoU of Sugar = 0.6271 is true positive
        IoU of Power Drill = 0.5977 is true positive

Figure20:
        IoU of Sugar = 0.4756 is NOT a true positive
        IoU of Power Drill = 0.3263 is NOT a true positive

Figure21:
        IoU of Sugar = 0.5286 is true positive
        IoU of Mustard = 0.0594 is NOT a true positive

Figure22:
        IoU of Sugar = 0.4581 is NOT a true positive
        IoU of Mustard = 0.1408 is NOT a true positive

Figure23:
        IoU of Mustard = 0.8242 is true positive

Figure24:
        IoU of Mustard = 0.0957 is NOT a true positive

Figure25:
        IoU of Mustard = 0.0000 is NOT a true positive

Figure26:
        IoU of Mustard = 0.2489 is NOT a true positive
        IoU of Power Drill = 0.6867 is true positive

Figure27:
        IoU of Mustard = 0.3031 is NOT a true positive
        IoU of Power Drill = 0.0000 is NOT a true positive

Figure28:
        IoU of Mustard = 0.5992 is true positive

Figure29:
        IoU of Mustard = 0.5884 is true positive

Figure30:
        IoU of Mustard = 0.0000 is NOT a true positive
```

3 Output Images with detections

Note that we use the Bounding box red to detect the power drill, the green one to detected the sugar and the blue one to detect the mustard. **Note** Images 6 of the mustard ...blablab depends on the parameters TOFIX

3.1 Images in 004_sugar_box/test_images/ directory

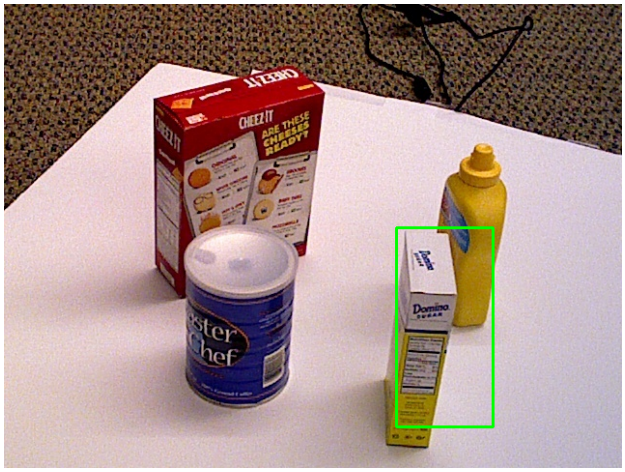

Figure 1:

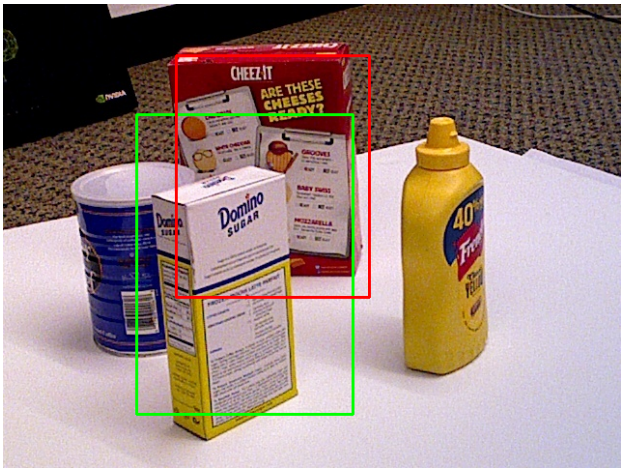

Figure 2:

Figure 3:

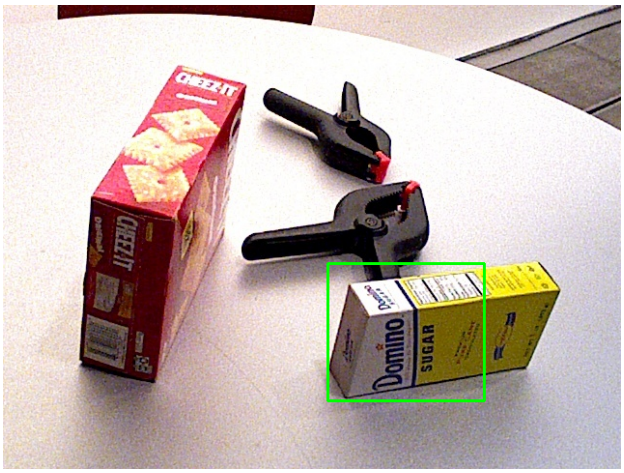

Figure 4:

Figure 5:

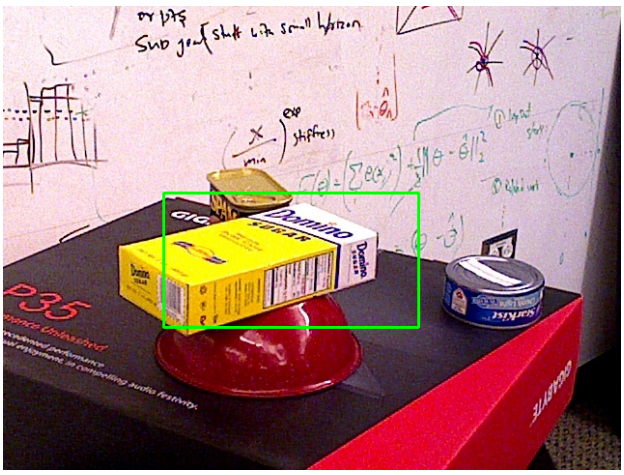

Figure 6:

Figure 7:

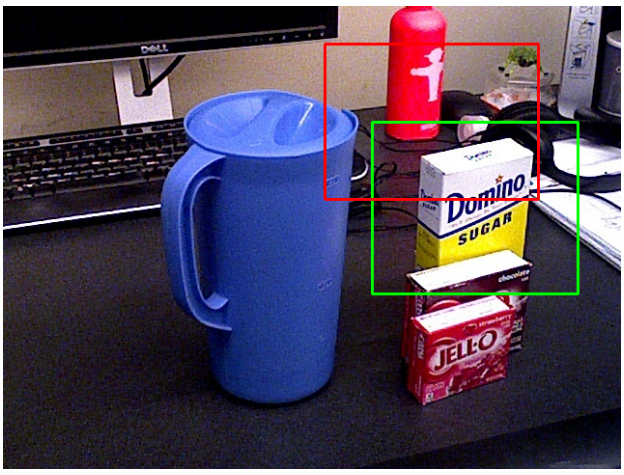

Figure 8:

Figure 9:

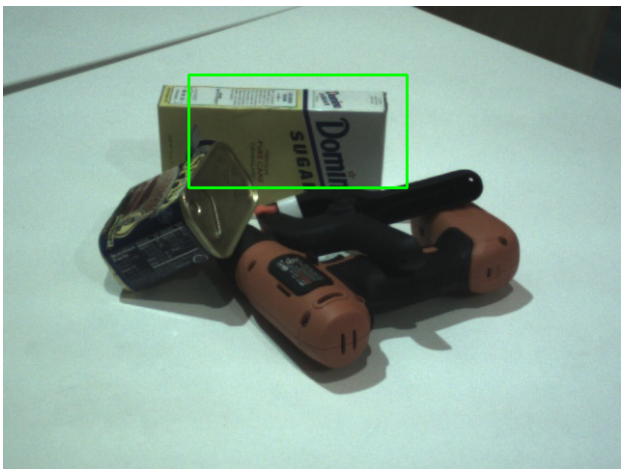

Figure 10:

3.2 Images in 035_power_drill/test_images/ directory

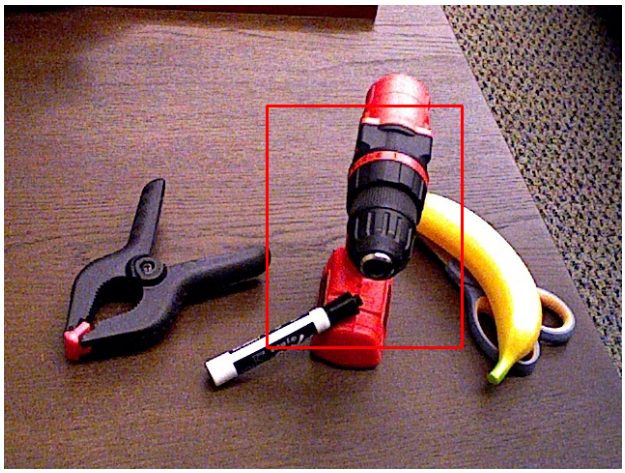

Figure 11:

Figure 12:

Figure 13:

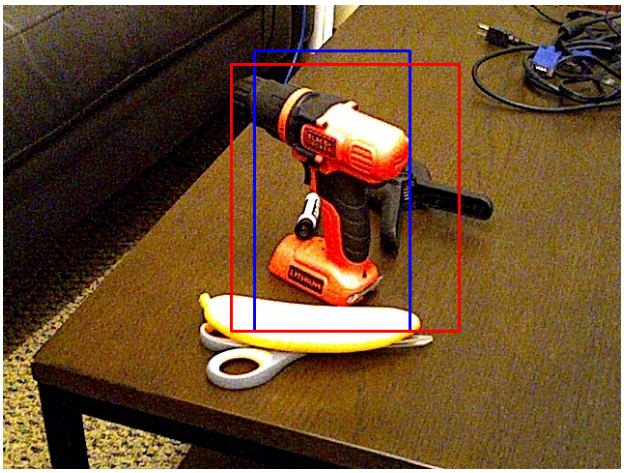

Figure 14:

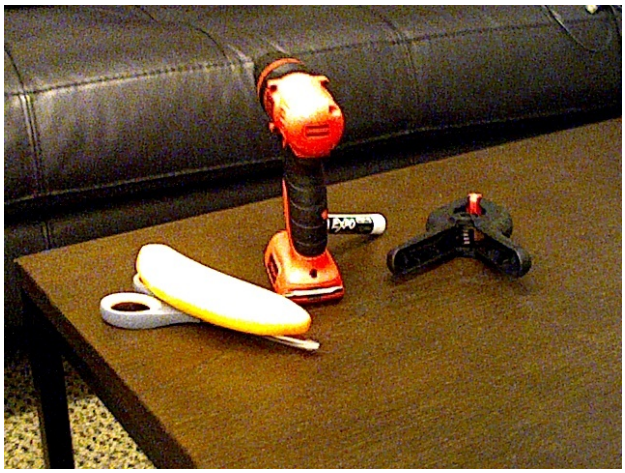

Figure 15:

Figure 16:

Figure 17:

Figure 18:

Figure 19:

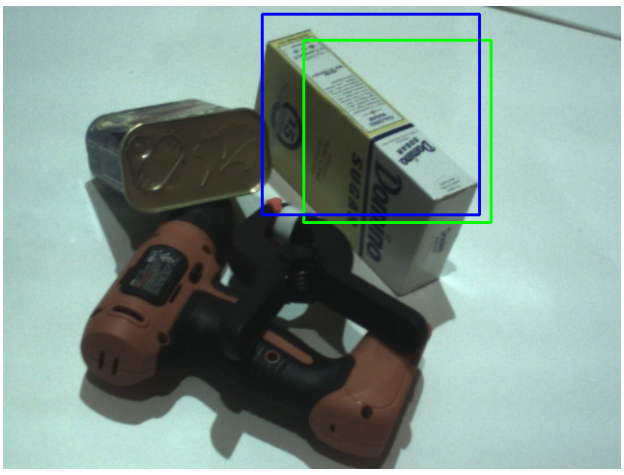

Figure 20:

3.3 Images in 006_mustard/test_images directory

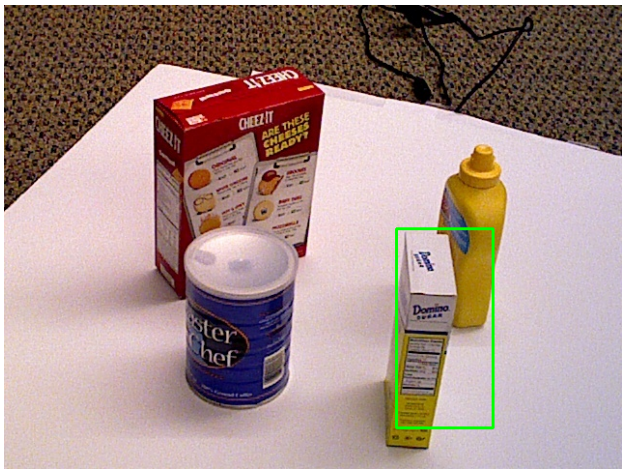

Figure 21:

Figure 22:

Figure 23:

Figure 24:

Figure 25:

Figure 26:

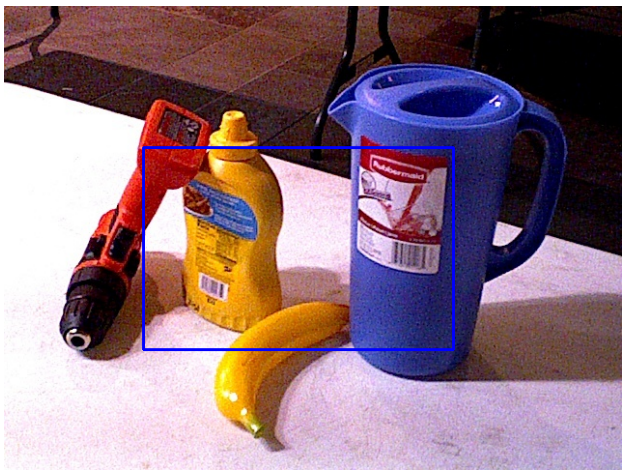

Figure 27:

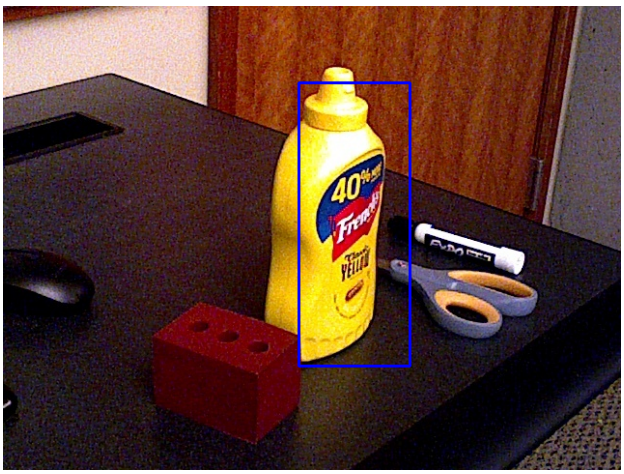

Figure 28:

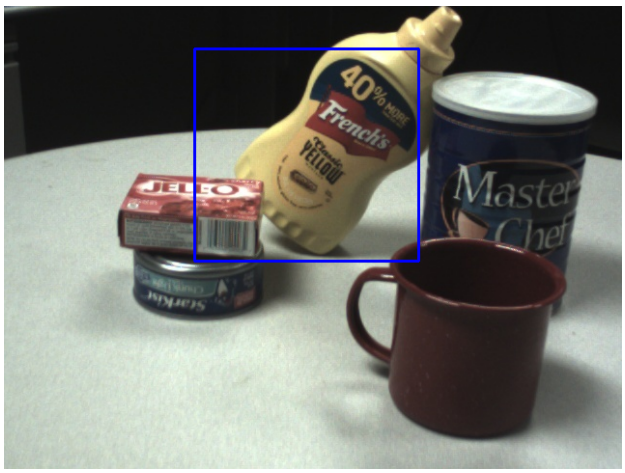

Figure 29:

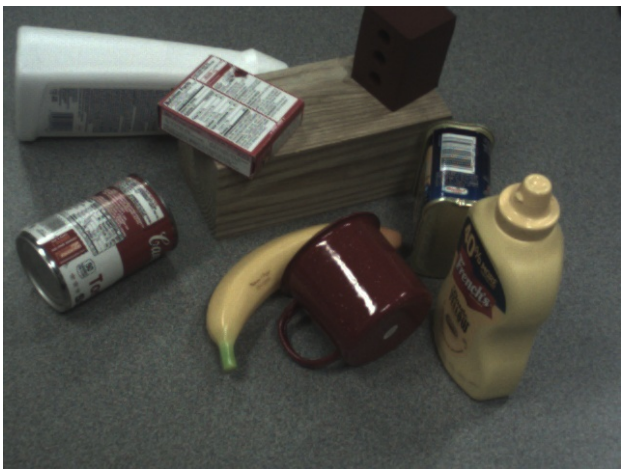

Figure 30: