Fabris Cristina ID number 1205722

# Report of Exam Project

## Goal of the project:

The aim of the project is to find the 10 templates with the best matching among 250 different positions of an object and 10 test images. The experiment is repeated using 3 different objects, each one associated with 10 different test images. To achieve the aim, 250 different views of each object are given (with their corresponding mask).

## My program:

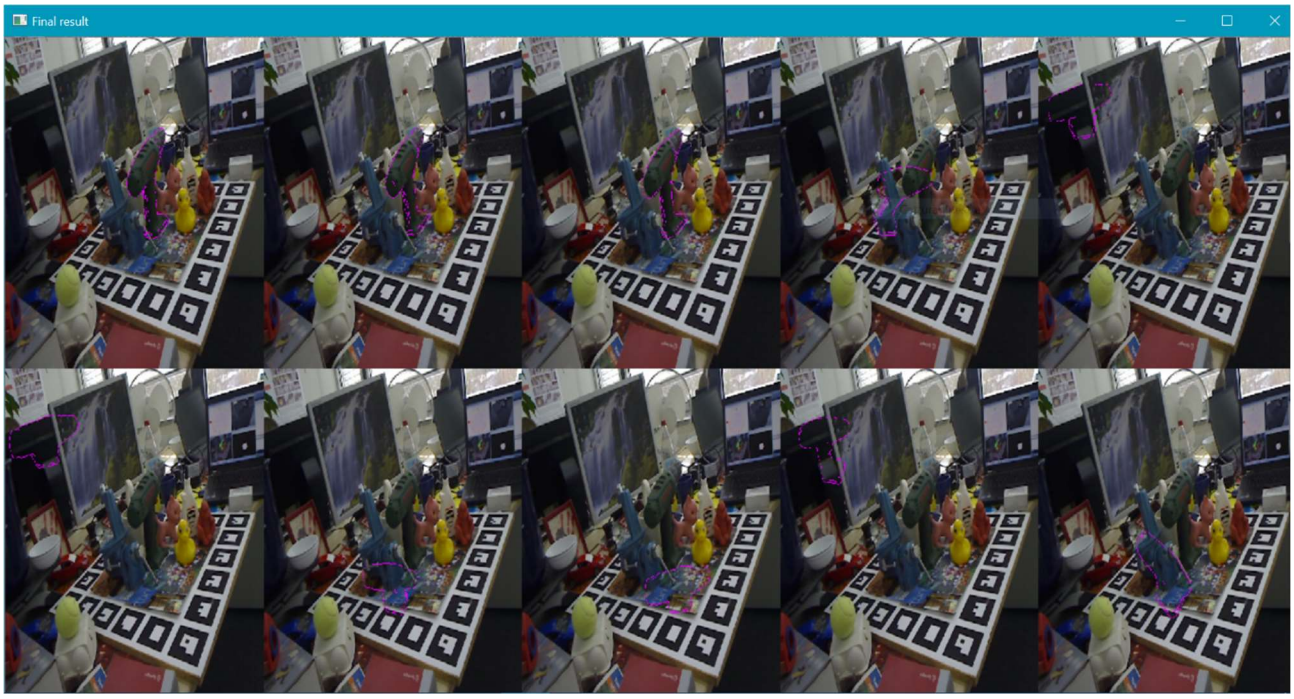
My program is composed by two source (main.cpp and utilities.cpp) and one header (utilities.h) files:

- main.cpp: it contains the main of the program. That is composed of a while cycle with a menu where to choose the object to be found in the images. The appearance of the menu is the following:

```
********************************************************************************
Choose the object and digit the corresponding number or 0 to exit:
1) can
2) driller
3) duck
********************************************************************************
```

As soon as the object is selected, the program:

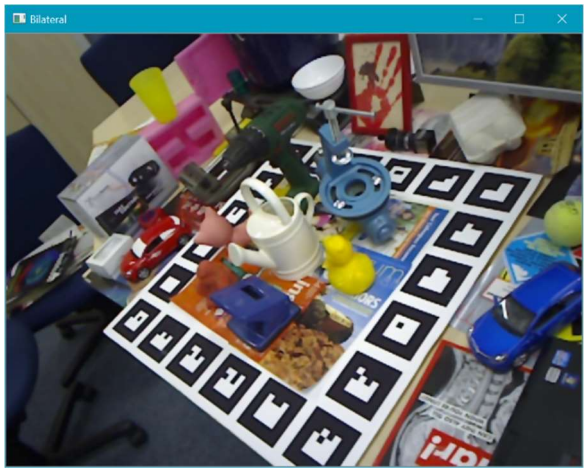
- Loads the images in which finding the match, the views of the object and their mask;
- Applies the Bilateral Filter to the test images, to simplify the successive detection of the edges;
- Modifies the value of the pixel colours, to reduce or add light to the image, to highlight the boundaries in the test and template images;
- Detects the edges in the template and test images, using the Canny algorithm;
- Computes the template matching, using the OpenCV function, and saves the 60 best matches;
- Selects the 10 best matches among the templates and each test image, using the more frequent colours in the selected template, and saves the result in a text file;
- Optionally, uncommenting the call to the dedicated function, shows the 10 best matches for each test image, drawing every template in a copy of the test image, and showing them in a unique window, like in the following picture.

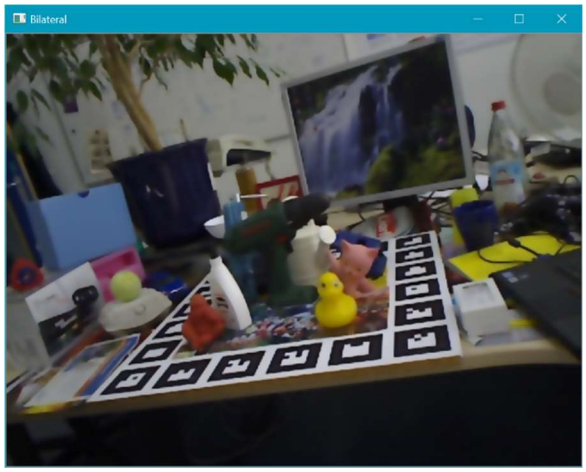



- utilities.cpp: it contains the implementation of my class ObjectMatching, used to manage the aim of the program. It contains also the function frequently_color(), used to refine the selection of the best matches.
- utilities.h: it contains the declaration of the class and the function implemented in the corresponding source file.

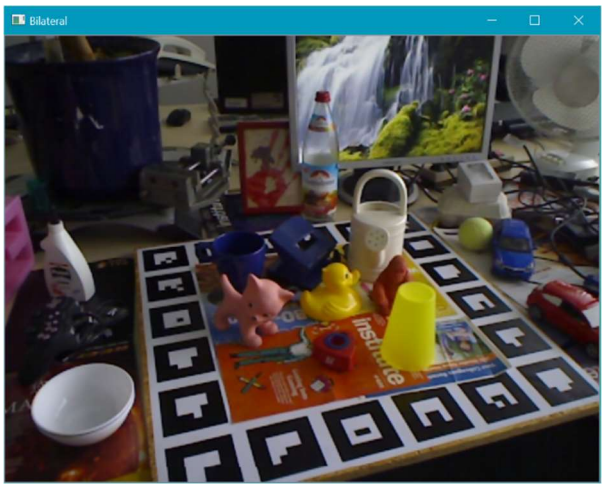
## Approach:

To obtain the correct matching between the views and the test image, I applied a template matching with a refinement based on the appearance of the objects. First, I applied the Bilateral Filter only to the test images to help the Canny algorithm to detect useful edges. I decided to use different parameters for the Bilateral function for each set of test images. I made this choice because the three objects that we are looking for (the can, the driller and the duck) are very different one from the other. In the following figures are reported the results of the Bilateral filter



in one image for each set (test_image 7 for the can set, test_image 5 for the driller set and test_image 1 for the duck set).




Then, to highlight the boundaries of the object in the picture, I modified the colours in the test and template images multiplying the values for each channel (in the RGB space) and for each pixel, for a constant (different for each set of images). In this way I obtained images darker or lighter than the original (depending on the value of the constant). I report the values for each object set:

| | | |
|---|---|---|
| Can: | `alpha_test = 0.5` | `alpha_template = 1` |
| Driller: | `alpha_test = 2.5` | `alpha_template = 1` |
| Duck: | `alpha_test = 1` | `alpha_template = 4` |

Finally, I computed the edge detection, using the Canny algorithm function, on the resulting test and template image. Thanks to the pre-processing of the images, the algorithm is able to detect all the edges that I need to correctly compute the template matching. The presence of not too many details helps the procedure to correctly identify each object and its positions.

The 2 following images are an example of the results of the Canny algorithm on a test image (img 2 of the driller set) and a template view (not the correct one with respect to the test image).


Using the edges images, I computed the template matching, in order to find the 60 best matches.

To refine the results of the template matching and choose the 10 best views, with their corresponding position, I decided to use the most frequent colours. I checked the colour frequency in the selected template view and in its position on the test image. I saved the 2 most frequent colours for each image and I computed the difference between them in the test image and in the template view. To do this in the best way, I identified the colours with the H channel of the HSV space. Then I recomputed the score of each match adding also the colours frequency information and I select the 10 best matches.

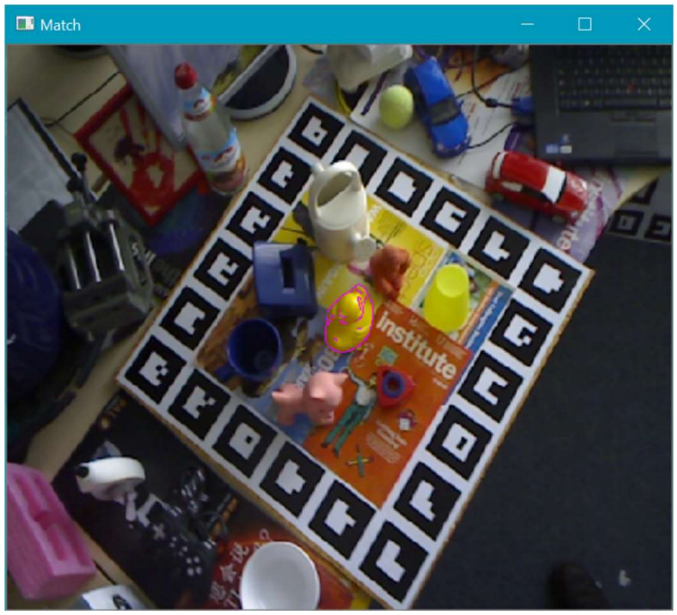
## Results:

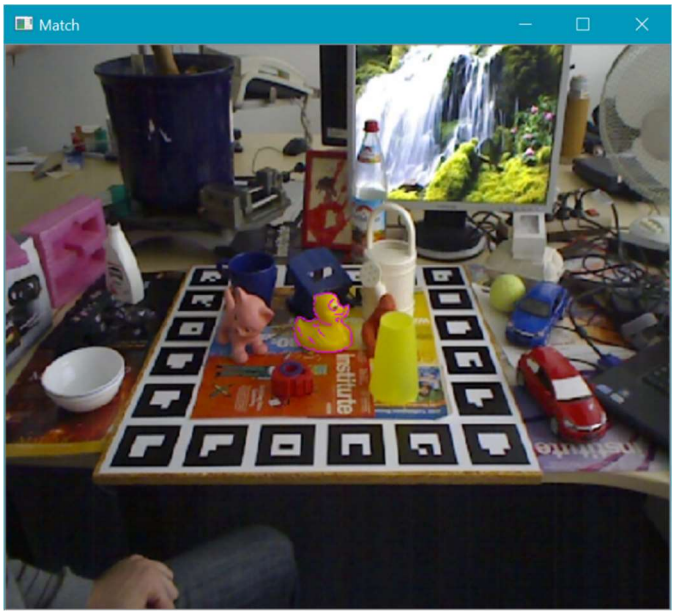
Using the approach described in the previous section, I obtained good results. In all the 30 test images, my program detects the correct view of the looking for object with one of the 10 best matches.

The more complex object to detect was the duck. I needed to change different methods before to finding the correct one. I think that the problems are caused by the easy shape of the duck by some point of view, that looks like a circle. Because of this, very often the templates matched with a high score with wrong object and so the results were not correct. This situation suggested me to use the general appearance, and so the frequency colours, of the template to do the refinement of the matches.

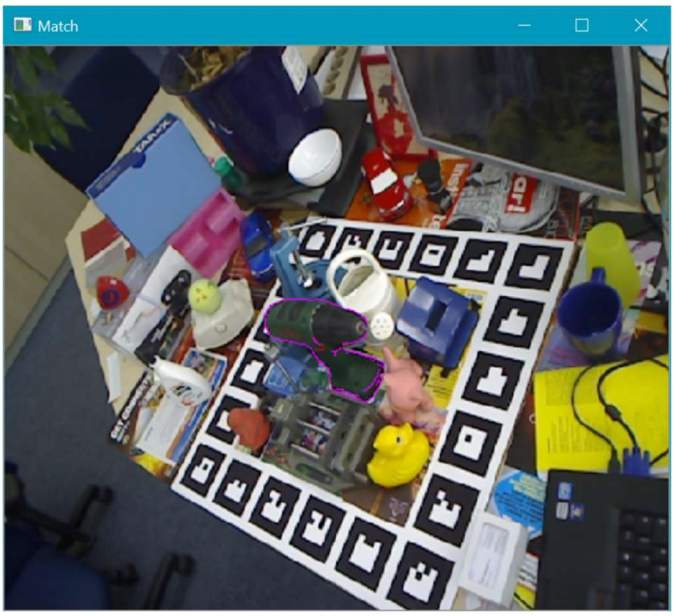
In particular, the duck in the test image number 8 has been the more difficult to detect. This happens probably because from that point of view the object have not well-defined details, useful to detect the correct template view.

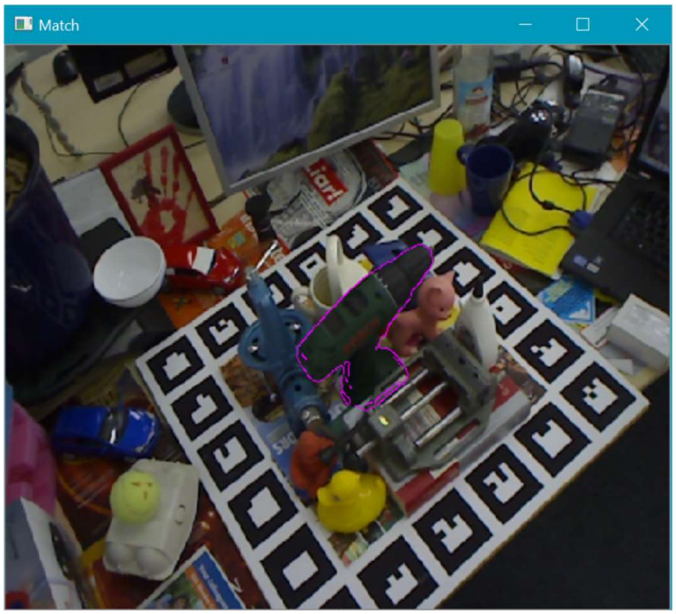
Here I report some results obtained with the duck set and in the next page with the other set.



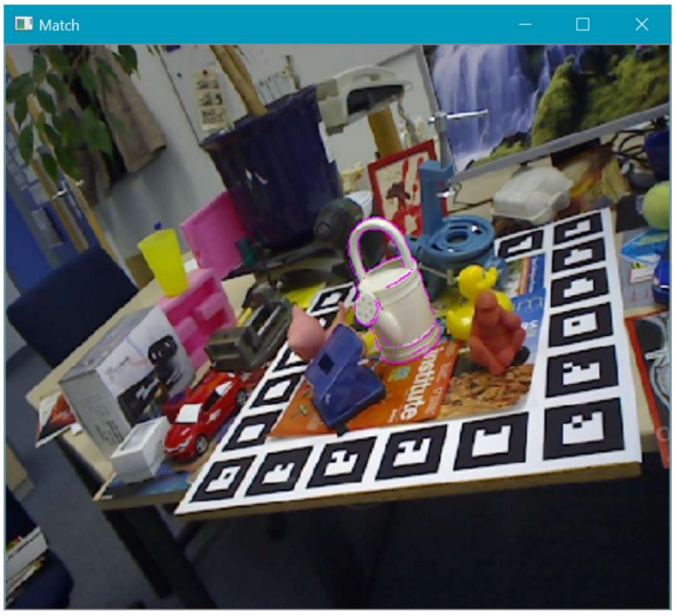



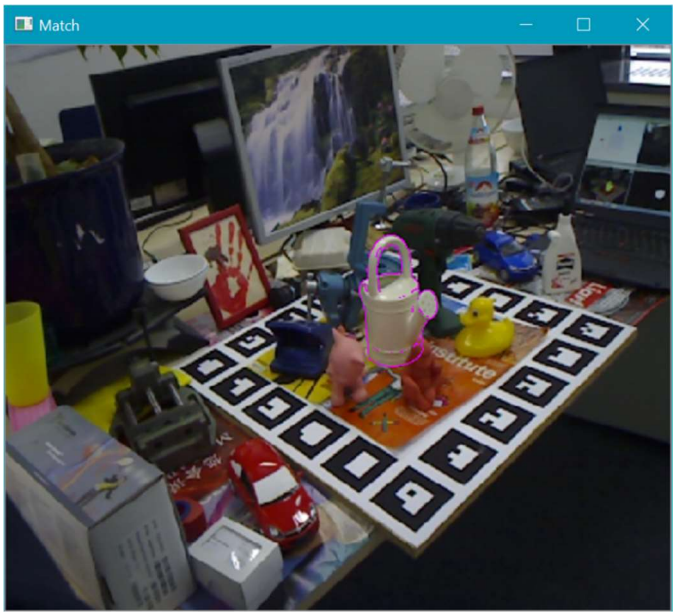
Another very problematic object to detect was the driller in the last image of its set, the number 9. Its position confuses a little bit my program, maybe because the base of the driller has a small part occluded by another object near it. The other views of the driller have been easier to correct match.




The can was very visible in the edges image, so it didn't cause problem to correct matches in all the test images with the correct template view.



Overall, I noticed that not always the correct match between a template view and the object position in the test image is the match with the highest score. Sometimes it is the fourth or fifth, this probably because in the test images there are a lot of object that can be approximated with the shape of the looking for objects' views. There are also some regions with the same frequent colours of the templates. Their presence can grow the score of wrong matches and put them in the set of the best matches during the refinement. This phenomenon happens in particular with the can set of images.