Computer Engineering Master Degree

Computer Vision

# Report about Project

Raffaele Di Nardo Di Maio 1204879

September 8, 2020

## 1 Project details

The goal of the experience was to estimate in an image the best match from a series of views of an object that we are trying to localize.

1. **Computation of $\gamma$-transform**
   this transform was applied to increase the constrast and stretch very high values in the images. In fact the values of the gamma values are all greater than 1. This transformation was applied only to views because of very strong brightness in them.

2. **Computation of Canny detection**
   Canny detector is applied to test images and also to $\gamma$-transformed views.

3. **Matching phase**
   this phase is performed using two different approaches, depending on specification of optional argument `-dist` (see Section 2.1). These approaches are:

   - **Template Matching with distance transform**
     The distance transformed is applied to each test image and then the function `cv::matchTemplate()` is applied to the result of the transform using Canny detected view as template. Then I search the max value in the resulting matrix and using the position of that pixel as the position of the template for the match. With this approach I take only the ten best matches for each image, as specified in the assignment. The score value used to establish the best matches is the value contained in the max entry of result matrix.
   - **Template Matching with histogram refinement**
     The matching is performed applying `cv::matchTemplate()` on the test image with detected edges and using view image with detected edges as template. To improve the results obtained, I don't take only the 10 best results but the 50 best results and then apply a refinement algorithm based on histogram, established using the same score of previous method. For each match previously computed a subwindow, related to the position of found match and with size equal to the related view, is computed and filtered with the mask related to the view of the match. This RGB subwindow of the original test image is then converted to the HSV space and its histogram of the Hue channel is computed. The resulting histogram is then compared with the Hue histogram of the original view image using `cv::compareHist()`. The final score of a match is the following:

     $$final\_score = \alpha * match\_score + \beta * hist\_score$$

     where $match\_score$ is the score obtained by applying only `cv::matchTemplate()` and $hist\_score$ is the score obtained by `cv::compareHist()`. Hence from the set of 50 best matches only 10 best matches survive.

4. **Generation of result txt files and result images**
   for each mask of 10 best matches, its egdes are highlighted in red and printed in position evaluated before over the image. For each dataset the matches are written in the corresponding result txt file.

### 1.1 Performance

### 1.2 Experimental results

I tried also to equalize images to increase contrast in the image but the results were worst than obtained ones without it.

## 2 Code

### 2.1 Command line parameters

The program needs to have, as command line arguments, the following ones with the specified format:

```
-i input_path [-r results_path] [-o output_path] [-h] [-dist]
```

**-i input_path** *Mandatory argument*
input_path is the folder that contains the subfolders of the three datasets(`can/`, `driller/` and `duck/`). Each one of this folders must contains also, as specified in the assignment of the project, the two subfolders `models/` and `test_images/`.

**-r results_path** *Optional argument*
results_path is the folder that will contain the txt files describing the ten best matches for each image (`can_results.txt`, `driller_results.txt` and `duck_results.txt`).
If this argument isn't specified by the user, the txt files are going to be stored in "default" path `../../../`

**-o output_path** *Optional argument*
output_path represents the path in which the program will save all the test_images, modified by printing the edges detected through Canny of the mask of the 10 best matches.
These are written on the image using red color and the name of the related model is shown in the high left corner of the image.
If this argument isn't specified, the result images with highlighted matches are not going to be store on disk.

**-h** *Optional argument*
if this argument is specified, even if there are other arguments, the program exits printing the description of possible command line arguments.

**-dist** *Optional argument*
if this argument is specified, distance transform method will be used in the computation of best matches, otherwise it will be used histogram refinement.

### 2.2 Code Organization

The program is organized in 13 files that we can organized, looking to their functionalities, into the following sets:

- **Template matching**
  It's composed by `TemplateMatching.hpp` and `TemplateMatching.cpp` files that implement the two methods used in the program to estimate the best matches.
- **Gamma transform**
  It's composed by `GammaTransform.hpp` and `GammaTransform.cpp` files that implement the gamma transform, used in the program to change the contrast of test images.
- **Menu**
  It's composed by `Menu.hpp` and `Menu.cpp` files that implement the parser of the command line arguments and manage the lines to be shown on the screen.
- **Main activity**
  It's composed by `Project.hpp` and `Project.cpp` that manage threads that perform Template Matching.
- **Canny detection**
  It's composed by `CannyDetector.hpp` and `CannyDetector.cpp` that compute Canny Detection for a specified image.
- **Storing of best matches**
  It's composed by `BestResults.hpp` and `BestResults.cpp` that implement the classes:

  - `Result`
    it constructs the entity of result of match and implements the methods needed to access to its fields.
  - `BestResults`
    it constructs the buffer of results and manage the update of best matches for each image.

- **Utility constants**
  It's composed by `Utility.hpp` file and implements some usefull values:
  - Color used in log printing
  - Color used in Menu
  - Color used for printing the mask name and the edges of the matched mask in the result image
  - Gamma values for Gamma Transform
  - Threshold values for Canny Detection
  - Threshold values for Canny Detection
  - Name of subfolders needed to load images of datasets provided by user