Lab 2 - Camera Calibration

Luca Scattolaro

1 April 2021

1 Report

1.1 Implementation choices

I decided to use the checkerboard images shared by the professor in order to being able to do a comparison between my result and the one obtained by the professor.

1.1.1 Initialization

I use these vectors:
- **objpoints**: vectors of 3D points for each checkerboard image containing corners position.
- **imgpoints**: vectors of 2D points for each checkerboard image for the extracted corners in pixels.

1.1.2 Find the Corners of Checkerboard

To find the corners of the checkerboard I used $cv :: findChessboardCorners()$.
Then I used $cv :: cornerSubPix()$ to find more exact corner positions in order to have a more reliable mean reprojection error.
Then I used $cv :: drawChessboardCorners()$ to show the corners found by the previous method.

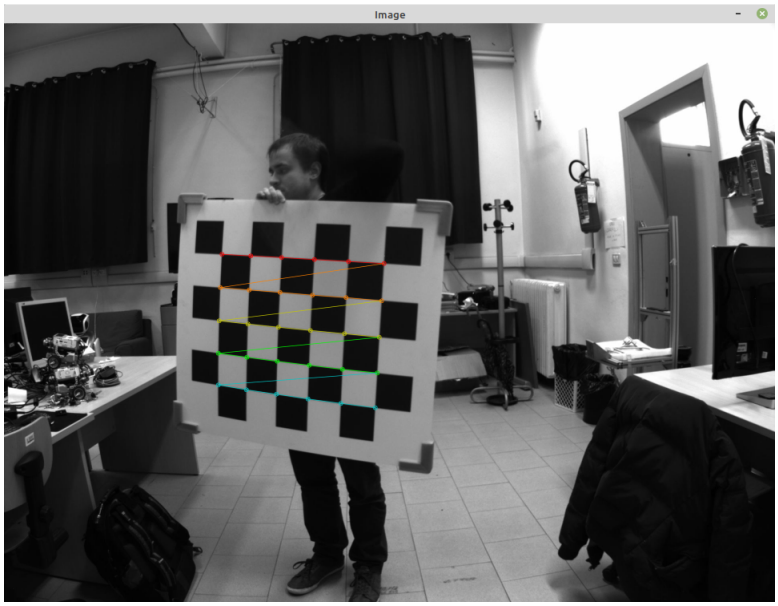

Figure 1: Example

1.1.3 Camera Calibration

To calibrate the camera I need to find the calibration Parameters.
To do that, I applied the method $cv :: calibrateCamera()$ and I get the values of:

- Camera Matrix
- Distorsion Coefficients
- Rotation vector
- Translation vector

```
Intrinsic Parameters :
[1249.509337730245, 0, 973.8414768297581;
 0, 1248.394838512299, 684.0219129841656;
 0, 0, 1]

Distorsion Coefficients :
 [-0.3085227042011033, 0.1391640838018809, 8.170728800678957e-06, 0.0003887339777297575, -0.03631232045609377]
```

Figure 2: Intrinsic Parameters and Distortion Coefficients

1.1.4 reprojection error

I compute the the mean reprojection error using the function *cv :: norm*. As we can see on the image below I tried in 2 different way to compute it but then by mathematically checking the results I found Which was the right way to compute it. (so then I commented the Wrong way)

```
for (int i = 0; i < objpoints.size(); ++i) {
    projectPoints(objpoints[i], rvec.row(i), tvec.row(i), cameraMatrix, distCoeffs, projectedPoints);
    //calculate reprojection Errors
    // RIGHT WAY
    double dist=0;
    for (int j = 0; j < objpoints[i].size(); ++j)
        dist = dist + norm(imgpoints[i][j] - projectedPoints[j]);
    mean_error+=reprojectionErrors[i]=dist/projectedPoints.size();

    // WRONG WAY
    mean_error+=reprojectionErrors[i]=norm(imgpoints[i], projectedPoints, normType: NORM_L2)/(projectedPoints.size());
```

Figure 3: Reprojection Error

In the point 7) of the Lab we had to print the name of the image for which the calibration performs best and the name of the image for which it performs worst.
To perform this choice I use the reprojection error calculated in the point 6) of the lab.
Best Image: minimum reprojection error (0039_color.png).
Worst Image: maximum reprojection error (0027_color.png).

```
MEAN TOTAL ERROR 0.321945
BEST IMAGE ../data/checkerboard_images/0039_color.png   0.0718993
WORST IMAGE ../data/checkerboard_images/0027_color.png   2.06902
```

Figure 4: Mean Total Error Error

1.2 Result

Using the calibration parameters I create an undistorted version of the test image.
On the image below we can see the original image on the left and the rectify one on the right.

Figure 5: Result Obtained

1.3 Difficulties

To do this Laboratory I follow primarily the opencv online guide and the theoretical slides of the lab.
On the slide there are some hints about the way to proceed and the functions to use to get the right results and to understand the meaning of all of them and also to use them properly I read the guide.
I have encountered some difficulties on point 5 (Computes the mean reprojection error) and on point 8 (Compare the result in a split view using the highgui module).
For the point 5 I searched for some pre-build function to calculate the mean reprojection error and I found norm() function but initially I used it in the wrong way.
To see what was the right way to use it I calculate the mean reprojection error explicitly and compared it with the norm function.
In the point 8 instead I search a lot on internet if there was something about "split view".
But I didn't find a lot of functions.
So at the end I decided to concatenate horizontally the two images (the test image and its Undistorted version) using the function hconcat.