UNIVERSITY OF AVEIRO
DEPARTAMENT OF ELECTRONICS, TELECOMUNICATIONS AND INFORMATICS

Computer Vision (2015/2016)

Lecture 04

Camera calibration

- Chessboard calibration
- Projection of 3D points in image with camera model
- Calibrating your camera
- External calibration

1. Chessboard calibration

Compile and test the file `chessboard.cpp`. This code detects corners in a chessboard pattern using openCV functions and shows the results of the detection for a series of images.
Use the available code to calibrate the camera used in the provided images (left01.jpg to left13.jpg). You need to use the function `calibrateCamera`. Help on this function can be found in http://docs.opencv.org/modules/calib3d/doc/calib3d.html. Last parameter should be 0 (not using previous information for calibration), and do not specify a termination criteria (not using last parameter).

You need to create the following variables to receive the results of the camera calibration process.

```
cv::Mat intrinsic = cv::Mat(3, 3, CV_32FC1);
cv::Mat distCoeffs;
std::vector<cv::Mat> rvecs;
std::vector<cv::Mat> tvecs;
```

You might show in the standard output the calibration results using:

```
std::cout << std::endl << "Intrinsics =  "<< std::endl << " "  << intrinsic_matrix << std::endl
<< std::endl;
std::cout << std::endl << "Distortion = "<< std::endl << " "  << distortion_coeffs << std::endl
<< std::endl;


std::cout << std::endl << "Translations = "<< std::endl ;
for (i=0;i<n_boards;i++)
std::cout << std::endl << translation_vectors.at(i);


std::cout << std::endl << "Rotations= "<< std::endl ;
for (i=0;i<n_boards;i++)
       std::cout << std::endl << rotation_vectors.at(i);
```

After a successful calibration, save the intrinsic and distortion matrices in an xml file using `FileStorage` as in the following code.

```
cv::FileStorage fs("../CamParams.xml", cv::FileStorage::WRITE);
fs << "cameraMatrix" << intrinsic_matrix << "distCoeffs" << distortion_coeffs;
fs.release();
```

Analyze the code for filling the 3D coordinates of the pattern. Imagine that you use another pattern with a different size what should you do to get the distance correctly evaluated?

Optional

You might improve the precision of the corner detection by using the `cornerSubPix` function after the call to `FindChessboardCorners`.

2. Projection of 3D points in the image

Use the function `cvProjectPoints2()` to project a line orthogonal (normal) to the chessboard (or a wireframe cube) in the provide images into each of the chessboard images using the rotation and translation vectors from the calibration.

3. Using camera on computer

Modify the code to use the camera from your computer to process the chessboard (comment the code for reading the provided images to allow switching between camera and provided images). Calibrate you camera with several chessboard images (use `cvWaitKey()` to move the chessboard position and a pre-defined number of images, for example 10). Be careful to check if the available chessboard is similar to the one in the provided images. If not, modify the code accordingly. If you want real metric distances, you need to update the code with the real distances of the used chessboard. Save the calibration parameters to a file.

Sample code for accessing an image is openCV as follow:

```
cv::Mat image;

cv::VideoCapture cap(0); // open the video camera no. 0

// Access Video
if (!cap.isOpened())  // if not success, exit program
  {
  std::cout << "Cannot open the video file" << std::endl;
  getchar();
  return -1;
  }

 for(int i=0; i<10; i++)
    {
    cap >> image; // get a new frame from camera
    imshow("cam", image);
    if(cv::waitKey(30) >= 0) break;
    }
```

4. External calibration

Calibrate a camera (using the given images or using your computer camera) and save the camera parameter file with another name.

Modify the previous examples to read the intrinsic and distortion parameters from the file and perform external parameters calibration (using function `solvePnP`) for a single image with the calibration pattern.

Read the file using the following code:

```
// load Matrixes
cv::Mat intrinsic_matrix;
cv::Mat distortion_coeffs;

cv::FileStorage fs("../CamParams.xml", cv::FileStorage::READ);


if (!fs.isOpened())
{
  std::cerr << "Failed to open " << filename << std::endl;
  return 1;
}


fs["cameraMatrix"] >>  intrinsic_matrix;
fs["distCoeffs"] >>  distortion_coeffs;
fs.release();
```

Remember that in this case, the external calibration should be performed for each single image returning its position (rotation and orientation).

5. Report

Write a report following the DETI journal template about the experiences done in this class. It should contain an example of the images displayed in each exercise, as well your comments about them. All the exercises must be repeated with images you acquired and the parameters of calibration of the camera you will be using must appear in the report.