cs512 Assignment 3: Report

Arun Rajagopalan
Graduate student, A20360689
Department of Computer Science
Illinois Institute of Technology

November 8, 2015

Abstract:
The problem or the task is to compute the camera parameters precisely from a set of world and image points, using our own implementation of camera calibration algorithm, alongside using other OpenCV library functions as necessary. The aim is to understand the algorithm and implement it in software. The inputs for the algorithm, effect of noise in the input, method for noise remediation and parameters for noise remediation algorithm are also to be studied, documented and analyzed.

1 Problem statement

To find camera parameters from the given world-image point correspondence, using own implementation of camera calibration algorithm. The names of the files that have the world and image points (with path, if the images are in some other folder) will be specified as arguments while starting the program from command line; the files should be read and parameters should be computed using them. After computing the parameters, verification should be done by using re-projecting world points using the camera parameters and finding deviation between the original and computed image points. If the image points are noisy, user will indicate so and in that case, RANSAC algorithm must be used for robust camera parameter estimation. RANSAC parameters should be read from a file called RANSAC.config. The program should also be tested with own data gathered from images we take. Solving this problem, helps to understand the camera calibration concept and familiarizes us with the software implementation details of the same.

2 Proposed solution

The solution starts with a check on the number of parameters being passed to the program while starting. Based on parameter count, either normal calibration or calibration with RANSAC is done. If only file-names (world point file-name followed by image point file-name) are given, normal calibration is performed and if a flag '-n' is specified after file-names, then calibration with RANSAC is to be performed. The program should never terminate unless the user intends and instructs the program to do so. While providing for camera parameter estimation, the program should also allow users to experiment with parameters (for RANSAC, in case of calibration with noisy points), which can be done by altering the RANSAC.config file. This is the basic outline of the solution. Deeper implementation details follow in the next paragraph.

3 Implementation details

To implement the solution, python 2.7.3 and OpenCV 3.0.0 was used. Main packages used include 'cv2' (for core image processing methods), 'numpy' (for image storing and array operation methods) and 'sys' (to get the arguments from command line). Source code (hw3_main.py) is placed in the 'code' folder under the parent directory 'arun_rajagopalan_as_3'. Files used to test the code are placed in 'data' folder under the same parent directory.

The calibration method employed is non-coplanar calibration.

At first, as discussed above, the program checks for the number of arguments passed on startup. If it is 3, usual calibration procedure is used to compute camera parameters. If it is 4 with the last being '-n', calibration with RANSAC (to eliminate the effect of outliers in the world and image points) is used. In case of invalid path or invalid number of arguments, an error message is thrown and program exits gracefully. A screen shot of the actual program behaviour on startup is shown in Illustration 1.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/world.txt
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
EXITED with ERROR: Incorrect argument count or flag.
Usage:
If data is not noisy: python hw3.py worldfilename imagefilename
If data is noisy:     python hw3.py worldfilename imagefilename -n
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/world.txt /home/rasuishere/arun_rajagopalan_as_
3/data/image.txt -m
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
EXITED with ERROR: Incorrect argument count or flag.
Usage:
If data is not noisy: python hw3.py worldfilename imagefilename
If data is noisy:     python hw3.py worldfilename imagefilename -n
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/world.txt /home/rasuishere/arun_rajagopalan_as_
3/data/image.t -n
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading and processing the noisy image points from the given path...
EXITED with ERROR: Unable to read/process file from the path!
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 1: Startup Cases

Camera parameter estimation, re-projection and error calculation

On start, with the correct world and image file-names (optionally the flag '-n') given as input, the program computes and displays the camera parameters and the deviation error right away. In case of noisy image and world points, the flag '-n' is included in command line and hence RANSAC is used while parameter computation.

First with the world and image points, the A matrix is formed in the format of [[Pw, 0, -xPw],[0,Pw, -yPw],.....] where Pw is the homogeneous world point and x and y are co-ordinates of the corresponding image point. Each element is of 4X1 size and hence the A matrix is of size 2mX12 where m is the number of world/image points.

By using svd function and finding singular value decomposition of A matrix, we get a V matrix in which the last column represents the eigen vector belonging to zero eigen value. This last column of V matrix is our M matrix of 4X4 size and it is subdivided into a1, a2, a3 and b. 'b' is the last column of M matrix and a1, a2 and a3 are transposes of rows of M matrix up until last column.

From the a1, a2, a3 and b values, camera parameters are found by the below formulae.

$$
\begin{aligned}
|\rho| &= 1/|a_3| \\
u_0 &= |\rho|^2 a_1 \cdot a_3 \\
v_0 &= |\rho|^2 a_2 \cdot a_3 \\
\alpha_v &= \sqrt{|\rho|^2 a_2 \cdot a_2 - v_0^2} \\
s &= |\rho|^4 / \alpha_v (a_1 \times a_3) \cdot (a_2 \times a_3) \\
\alpha_u &= \sqrt{|\rho|^2 a_1 \cdot a_1 - s^2 - u_0^2} \\
K^* &= \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\
\epsilon &= \mathrm{sgn}(b_3) \\
T^* &= \epsilon |\rho| (K^*)^{-1} b \\
r_3 &= \epsilon |\rho| a_3 \\
r_1 &= |\rho|^2 / \alpha_v a_2 \times a_3 \\
r_2 &= r_3 \times r_1 \\
R^* &= [r_1^T \;\; r_2^T \;\; r_3^T]^T
\end{aligned}
$$

RANSAC algorithm, which is used in case of noisy image points, is usually employed to get rid of outliers in any dataset and is used in computer vision applications widely. It is implemented in the below fashion.

First a set of random points are taken and a model is fit to those points. This model is then used to fit all the points and using a pre-defined threshold, inliers are identified. Now with the inliers, a new model is formed and fitted on all points. This process runs for a calculated K number of times or a maximum count, whichever occurs earlier. The best model is identified by minimum objective or maximum consensus and is used to determine the camera parameters. K is calculated for every iteration by the below formula.

$$K = \log(1-P)/\log(1-W^n)$$

Here, W is the ratio between number of inliers and total number of points for each iteration. P is the probability that at least one of the draws is free from outliers. The threshold, number of points to be drawn at each iteration and the maximum count are specified by the users in RANSAC.config file.

After the camera parameters are identified, the world points are projected using the parameters and the error between the original and computed image points are calculated to verify our parameters.

Error is less than zero for noiseless files in our experiment. Error is tolerable in case of second set of noisy points, computed with RANSAC. In case of first set, RANSAC does not work because a single satisfiable model cannot be achieved from these points. The outliers are very aberrant that they disturb the models formed with random samples.

4 Results and discussion

Output of the program for noise less world and image point files (world.txt and image.txt present in 'data' folder) is shown below in Illustration 2. The parameters calculated are precisely correct and the deviation is less than 0 (rounded in the program).

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/world.txt /home/rasuishere/arun_rajagopalan_as_
3/data/image.txt
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading the image points from the given path...

Intrinsic parameters are as follows.
u0 = 320.000170, v0 = 239.999971, alphaU = 652.174069, alphaV = 652.174075, s =
-0.000000
Extrinsic parameters are as follows.
T* matrix is as below.
[ -2.57805273e-04   3.26847737e-05   1.04880905e+03]
R* matrix is as below.
[ -7.68221190e-01   6.40184508e-01   1.46391093e-07] [ 0.4272743   0.51272918 -0
.74467809] [-0.47673145 -0.57207743 -0.66742381]

Mean error between the known and computed image points = 0.000000

rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 2: Parameter estimation for world.txt and image.txt

The outputs for the input pairs worldn.txt-imagen.txt and worldn1.txt-imagen1.txt, all available in the 'data' folder, are as below.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/worldn1.txt /home/rasuishere/arun_rajagopalan_a
s_3/data/imagen1.txt -n
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading and processing the noisy image points from the given path...

Intrinsic parameters are as follows.
u0 = 319.999468, v0 = 239.998112, alphaU = 652.171708, alphaV = 652.171967, s =
-0.000000
Extrinsic parameters are as follows.
T* matrix is as below.
[  8.56717290e-04   2.73672145e-03   1.04880610e+03]
R* matrix is as below.
[ -7.68221589e-01   6.40184029e-01  -5.70838619e-07] [ 0.4272736   0.51272816 -0
.7446792 ] [-0.47673144 -0.57207888 -0.66742258]

Mean error between the known and computed image points = 1.901176

rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 3: Parameter estimation for worldn1.txt and imagen1.txt

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/worldn.txt /home/rasuishere/arun_rajagopalan_as
_3/data/imagen.txt -n
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading and processing the noisy image points from the given path...
hw3_main.py:33: RuntimeWarning: divide by zero encountered in double_scalars
  s = math.pow(mp,4)/av*np.dot(np.cross(a1,a3),np.cross(a2,a3))
hw3_main.py:34: RuntimeWarning: invalid value encountered in sqrt
  au = np.sqrt((mp*mp*np.dot(a1,a1))-s*s-u0*u0)
EXCEPTION HANDLED: Current instance will be skipped, operation continues...

Intrinsic parameters are as follows.
u0 = 332.069025, v0 = 193.922755, alphaU = 499.672029, alphaV = 520.084556, s =
-1.000000
Extrinsic parameters are as follows.
T* matrix is as below.
[ -28.2438418    65.54916055  871.93011384]
R* matrix is as below.
[-0.7366518   0.67568806  0.02810287] [ 0.41609493  0.48561229 -0.76879497] [-0.
53311268 -0.55464074 -0.63887755]

Mean error between the known and computed image points = 3.416707

rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 4: Parameter estimation for worldn.txt and imagen.txt

The output for the own input pair worldc.txt-imagec.txt without using RANSAC is as below.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/worldc.txt /home/rasuishere/arun_rajagopalan_as
_3/data/imagec.txt
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading the image points from the given path...

Intrinsic parameters are as follows.
u0 = 349.855807, v0 = 256.622974, alphaU = 453.552397, alphaV = 450.100012, s =
-5.000000
Extrinsic parameters are as follows.
T* matrix is as below.
[ -45.60750203  -21.27090442  808.3205459 ]
R* matrix is as below.
[-0.74865925  0.66249291  0.02474832] [ 0.43422765  0.51822938 -0.73680707] [-0.
50095476 -0.54087102 -0.67564996]

Mean error between the known and computed image points = 4.000000

rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 5: Parameter estimation for worldc.txt and imagec.txt without RANSAC

The output for the own input pair worldc.txt-imagec.txt with RANSAC is as below.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python hw3_main.py /home/rasui
shere/arun_rajagopalan_as_3/data/worldc.txt /home/rasuishere/arun_rajagopalan_as
_3/data/imagec.txt -n
**Starting Basic Camera Calibrator (Non-Coplanar) v1.0**
Reading and processing the noisy image points from the given path...
EXCEPTION HANDLED: Current instance will be skipped, operation continues...

Intrinsic parameters are as follows.
u0 = 319.998588, v0 = 239.997908, alphaU = 652.174755, alphaV = 652.175102, s =
-0.000000
Extrinsic parameters are as follows.
T* matrix is as below.
[  2.10848300e-03   3.16154603e-03   1.04881003e+03]
R* matrix is as below.
[ -7.68221858e-01   6.40183706e-01  -1.23403106e-06] [ 0.42727331  0.51272748 -0
.74467983] [-0.47673126 -0.57207985 -0.66742187]

Mean error between the known and computed image points = 1.213421

rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$
```

Illustration 6: Parameter estimation for worldc.txt and imagec.txt with RANSAC

The output of the ChessCornerDetector program, which is used to extract points from own images, with left01.jpg image as input is as below. The points are also written to files and stored in the current folder.

```
rasuishere@ARajago6:~/arun_rajagopalan_as_3/code$ python ChessCornerDetector.py
/home/rasuishere/arun_rajagopalan_as_3/data/left01.jpg
**Starting Basic Chess Corner Detector v1.0**
[array([[ 0.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 2.,  0.,  0.],
       [ 3.,  0.,  0.],
       [ 4.,  0.,  0.],
       [ 5.,  0.,  0.],
       [ 6.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 1.,  1.,  0.],
       [ 2.,  1.,  0.],
       [ 3.,  1.,  0.],
       [ 4.,  1.,  0.],
       [ 5.,  1.,  0.],
       [ 6.,  1.,  0.],
       [ 0.,  2.,  0.],
       [ 1.,  2.,  0.],
       [ 2.,  2.,  0.],
       [ 3.,  2.,  0.],
       [ 4.,  2.,  0.],
       [ 5.,  2.,  0.],
       [ 6.,  2.,  0.],
       [ 0.,  3.,  0.],
```

Illustration 7: Output of ChessCornerDetector.py program

5 References

https://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_tutorials.html
http://docs.opencv.org/
https://docs.python.org/3/tutorial/
http://stackoverflow.com/questions/tagged/python-2.7
http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html