Computer Vision

Exercise 1 report

Amrollah Seifoddini

* The images are taken from Rabeeh Karimi, as all my efforts for connecting my mobile camera to the laptop failed.
* All the data and images are available in the attached zip file.
* The results of bouget’s toolbox are available as .mat file in the attachment.

# Part 1) DLT:

Error: 0.6012

K = [4142.76483968464 90.8717322544668 512.236301383930

0 3838.75395072845 776.611944924064

0 0 1]

R = [-0.638551202776751 0.762524982409825 0.103961592106222

-0.611268834602189 -0.420473548765596 -0.670486693851972

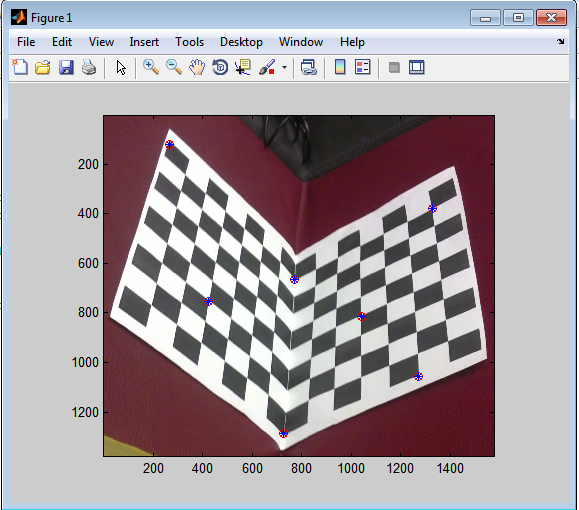
-0.467549754867272 -0.491688566055142 0.734601647652789]

t = [1.56101540709755

4.26043521781777

32.0342557109819]

For the implementation, I followed the exercise slides guide. First, normalization and then forming the matrix equation system is obvious. Afterwards, by using the SVD, I got V matrix and extract P out of it.



\*Red circles are the original points and blue crosses are re-projected ones.

# Part 2) Gold Standard Algorithm:

Error: 0.5984

K = [4139.84780964289 90.7837266099746 515.218190235403

0 3835.41490600405 779.112684068189

0 0 1]

R = [-0.638326013589452 0.762772761825715 0.103525910726592

-0.610956517479677 -0.420217724726278 -0.670931589340505

-0.468264918741629 -0.491522916701390 0.734256895257548]

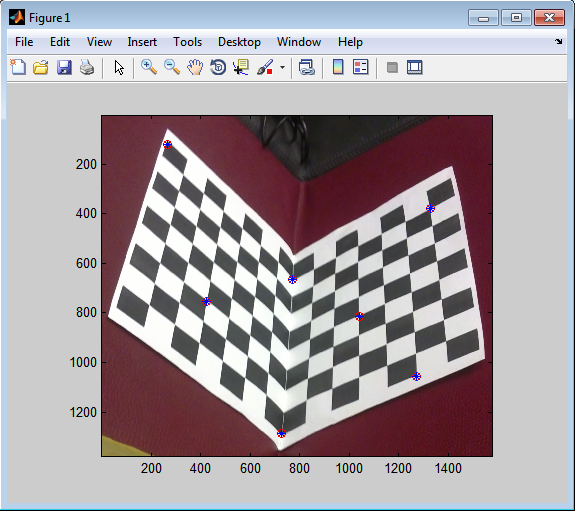
t = [1.53831516387587

4.23965491846563

32.0097919390351]

For this part, I also followed the slides of the course and exercise guides. After normalization and initialization with DLT, I used an iterative loop for finding the best P^.

As we can see, we have an improvement (although little!) compared to DLT. This comes from our iterative approach.



# Part 3) Bouget’s toolbox:

The result are as follows for 14 images that I used.

Focal Length: fc = [ 3527.27973 3604.87629 ] ± [ 148.79011 167.75616 ]

Principal point: cc = [ 1307.38494 651.10609 ] ± [ 175.67058 181.79677 ]

Skew: alpha\_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc = [ 0.08274 -0.56439 -0.01693 0.00463 0.00000 ] ± [ 0.17530 0.56595 0.01703 0.02165 0.00000 ]

Pixel error: err = [ 5.28786 7.02424 ]

It can be seen that the error rate is much higher than DLT and Gold algorithms. I interpret this as a bad data issue. Because I used some steep angles and did not use the optional parameters that the toolbox offers, this is not a big surprise to see that much error. I mean the size of each square and window size and etc.

Some of the results are shown as pictures in the following.

