CS 512 Assignment 4: Report

Charu Saxena
Department of Computer Science
Illinois Institute of Technology
19th Nov ,2017

Abstract

In this report we will discuss about various implementation and usage of various opencv functions and python functions using python 3.6. Some of the implementations and usage discussed are basic computer vision and image processing problems. Particularly, **non-coplanar** camera calibration which will include implementation of camera calibration for a noisy image with robust estimation to remove outliers using RANSAC.

1. Problem Statement

a) To write a program to load and display images and perform following operations:

- Extract features points for an noisy image and save them in a file.

b) Reading the features from the image and get corresponding 3-D points.

- Now to implement camera calibration and compute projection matrix with removing outliers using RANSAC with following parameters:
    - Desired probability=0.99
    - Automatic estimation of number of draws(K) and probability data is an inlier(w)
    - Set max value for K

2. Proposed Solutions

For first part of the program:

For here the corners were detected, using cornerHarris() for the synthetic image,
Then after this we localise the corners using cornerSubPix().
After this we use mgrid().reshape to generate 3d points for the 2d points we got in above.
And these are saved in the file named “**point_generated.txt**”. This file generated is then used in the rest of the program2 as an input through command line.

For second part of the program:

I implement two thing:

a) Camera Calibration
b) RANSAC

a)Camera Calibration-

- Here I used the world and image points from the file we saved in part one.
- Then we used made a matrix A from it where:
  - For each point we define rows for the matrix A

    Row1 = X,Y,Z,1,0,0,0,0,-xi* X, -xi *Y,-xi*Z,-xi

    Row2 =0,0,0,0, X,Y,Z,1,-yi* X, -yi *Y,-yi*Z,-yii

$$\underset{\substack{\mathbf{A}\\ \mathbf{2n\times 12}}}{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix}} \quad \underset{\substack{\mathbf{m}\\ \mathbf{12}}}{} \begin{bmatrix} m_{00} \\ m_{10} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \underset{\substack{\mathbf{0}\\ \mathbf{2n}}}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}$$

  - there for making 12 values in each row for the matrix, number of columns here will be each to (2*number of total world/image points we got from the features in file) this is calculated using numpy.array() and list
  - After finding the matrix A, I use **np.linalg.svd()** to find the svd for the matrix A.
  - The last column for V.transpose will provide the values for the projection matrix
  - I then calculate projection matrix by using:

    ProjectionMatrix = V.T[:,-1].reshape(3,4)

  - After finding the projection matrix , I multiplied this by the world points, this will generate corresponding 2DH image points which can be converted to image points, and will save them in a array.

$$p = K\begin{bmatrix} R & t \end{bmatrix} P = MP$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- For a projection matrix we can describe both intrinsic and extrinsic parameters:

$$P = \overbrace{K}^{\text{Intrinsic Matrix}} \times \overbrace{[R \mid \mathbf{t}]}^{\text{Extrinsic Matrix}}$$

$$= \overbrace{\underbrace{\begin{pmatrix}1&0&x_0\\0&1&y_0\\0&0&1\end{pmatrix}}_{\text{2D Translation}} \times \underbrace{\begin{pmatrix}f_x&0&0\\0&f_y&0\\0&0&1\end{pmatrix}}_{\text{2D Scaling}} \times \underbrace{\begin{pmatrix}1&s/f_x&0\\0&1&0\\0&0&1\end{pmatrix}}_{\text{2D Shear}}}^{\text{Intrinsic Matrix}} \times \overbrace{\underbrace{\left(\, I \mid \mathbf{t} \,\right)}_{\text{3D Translation}} \times \underbrace{\left(\begin{array}{c|c}R&0\\\hline 0&1\end{array}\right)}_{\text{3D Rotation}}}^{\text{Extrinsic Matrix}}$$

b) Ransac

After getting the features in a file, and calculating the corresponding world and image coordinates , RANSAC is done for robust estimation and removal of outliers in the noisy data.
The following parameters are considered:

a) The parameters are initialized:number of trial(K), number of points drawn(N) = 8, threshold(t), number of inliers(d) =6, p = 0.99, probability data is an inlier initial w=0.5

b) For Kth iteration with a max_k set:

   i) Randomly select N points from the file with replacements

   ii) For these points do camera calibration to fit the model i.e to find the projection matrix using steps discussed in above part.

   iii) FInd all the inliers such that, their deviation from the model is below some threshold t.This threshold is t = 1.5*(median(dist(inliers,model)))

   iv) Here the probability data is an inlier(w) is updated during each iteration as:
   w=(number of inliers)/(total points)
   Then K is also updated as:
   $K= \log(1-p)/\log(1-w^n)$

   v) For these inliers if the number of inliers are more than d to say it is a good model we will recompute the model(i.e in this case recompute the camera calibration)

c) After this calculation we see the inliers set and the set with highest number of inliers is chosen as the best solution. Here the model using these inliers we calculate the projection matrix and calculate all the intrinsic and extrinsic parameters.

For comparison, we use the final projection matrix to calculate the estimated image points whose mean square error was calculated with the actual image points.
Using

**((math.pow(expected_U-observed_U,2)+math.pow(expected_V-observed_V,2)))
/len(data)**

3. Implementation

The implementation is done in python 3.6 and spyder

Problems faced and solutions:

- Every Time I was calculating the world or image point I had to change it to float as reading from file will give type ‘str’ and I was not able to perform operations, so for this typecast float() in the starting to prevent from error on data type.
- Appending the points to the matrix A was a problem as all were saving as a single array, sor for this I used list of lists having information of each point(world and image points for each)in each sub-list.
- Sometimes while random.sample we get certain values such that the “wi” 2d-H coordinate for the image came out to be zero, so while calculating the estimated image points it is throwing error “divide by zero.”
- In implementation of RANSAC, value for w is not initially known , so we took w=0.5 and n=8, and so K =log(1-0.99)/log(1-(0.5)^8), then with each iteration it will change

Instructions for the program:

Both the programs will run on spyder IDE or any other python supported IDEs

- For program 1 input image, this will make a text file and save 3d, 2d points in that and match those points on image. As a result a data file is generated with 2d and 3d points the file name generated is **points_generated.txt**

  Run the file using command line:

  ```
  $python program1.py <image>.jpg
  ```

- For program 2 input a text file having noisy data(3d and 2d points) , this will return intrinsic and extrinsic parameters and mean squared error(**program2.py**), for here text file generated in part 1 is passed through command line.
  This has to have two input files:

  File with 3d and 2d points as in format : X Y Z xi yi

In one line for each point.

And second file RANSAC.config: three parameters are input in the format:

```
10000
6
0.99
```

Here first denotes- k_max, second is value of d(number of inliers) and third is ‘p’ probability

Run the file using command line:

```
$python assignmnet4opencv.py <filename>.txt RANSAC.config
```

- For Program 3 is for testing purpose, if you enter text file with 3d and 2d points in this order, <X,Y,Z,xi,yi> then it will produce the corresponding projection matrix and intrinsic and extrinsic parameters without RANSAC robust estimation. (**program3test.py**)
  For here text file generated in part 1 is passed through command line.

Run the file using command line:

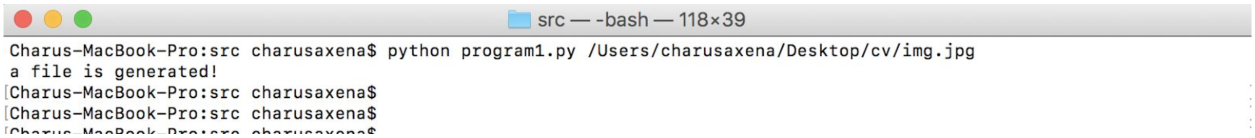
```
$python assignmnt4program3test.py <filename>.txt
```

4. Result and discussion

a) Synthetic image testing

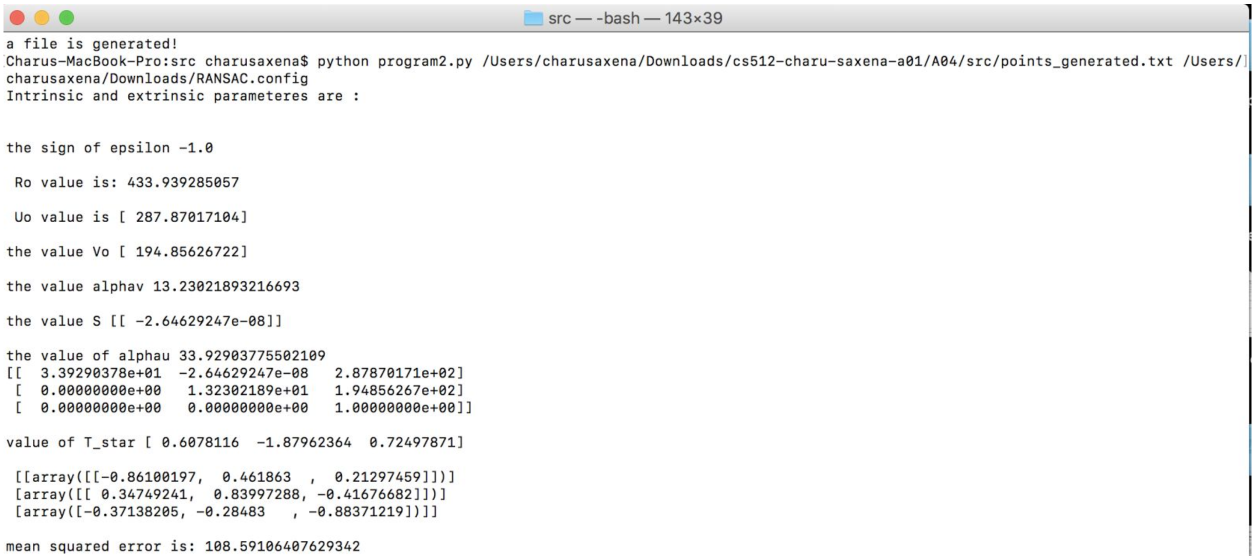
- The image selected was chessboard image for testing the code synthetic image:

When program 1 is run,



```
Charus-MacBook-Pro:src charusaxena$ python program1.py /Users/charusaxena/Desktop/cv/img.jpg
a file is generated!
[Charus-MacBook-Pro:src charusaxena$
[Charus-MacBook-Pro:src charusaxena$
[Charus-MacBook-Pro:src charusaxena$
```

Then program 2 to see the extrinsic and extrinsic parameters and intrinsic parameters,



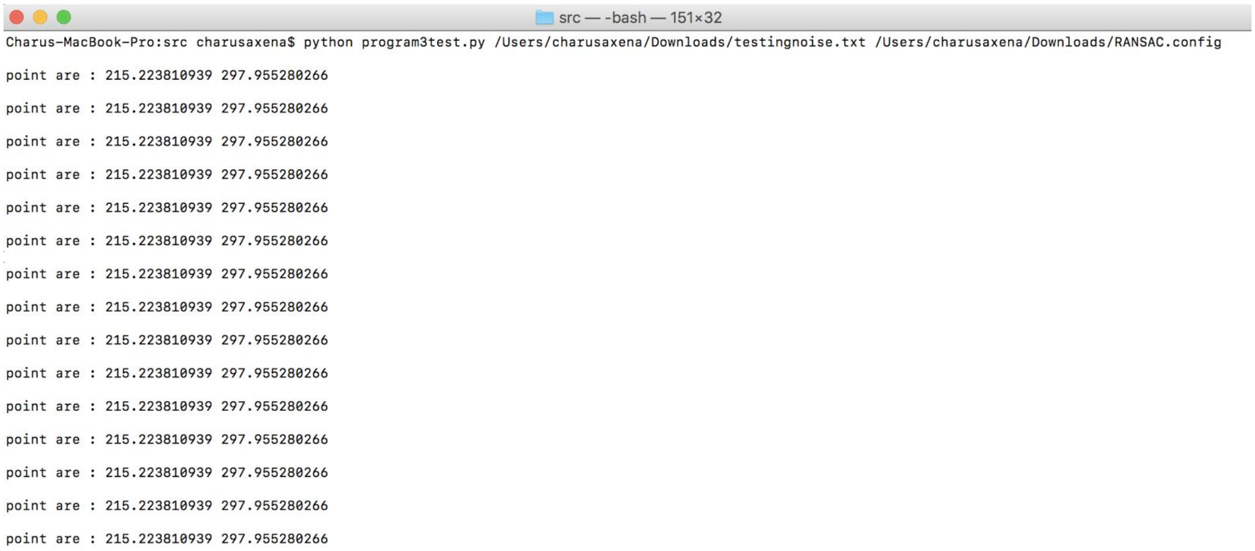
```
a file is generated!
Charus-MacBook-Pro:src charusaxena$ python program2.py /Users/charusaxena/Downloads/cs512-charu-saxena-a01/A04/src/points_generated.txt /Users/]
charusaxena/Downloads/RANSAC.config
Intrinsic and extrinsic parameteres are :


the sign of epsilon -1.0

 Ro value is: 433.939285057

 Uo value is [ 287.87017104]

the value Vo [ 194.85626722]

the value alphav 13.23021893216693

the value S [[ -2.64629247e-08]]

the value of alphau 33.92903775502109
[[  3.39290378e+01  -2.64629247e-08   2.87870171e+02]
 [  0.00000000e+00   1.32302189e+01   1.94856267e+02]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]

value of T_star [ 0.6078116  -1.87962364  0.72497871]

 [[array([[-0.86100197,  0.461863   ,  0.21297459]])]
 [array([[ 0.34749241,  0.83997288, -0.41676682]])]
 [array([-0.37138205, -0.28483   , -0.88371219])]]

mean squared error is: 108.59106407629342
```

In some random sampling we saw that error in that was 108.59 and other intrinsic and extrinsic parameters were obtained.
The possible reason for this high error may be because the 3-d points generated were synthetically produced and were not very real.

<u>b) Validations</u>

- The M matrix was tested on the individual world point and yielded correct result. The world points used were used http://www.cs.iit.edu/~agam/cs512/data/calibration/index.html

i)Noisy Data
Using coplanar ransac on noisy
World point - 200.00 0.00 0.00
Expected image point - 214.9064 298.4516
Observed value:
<u>Running program 3 for test</u>
Using the noisy data cross-checked from data used in cs512 website generated following on terminal.

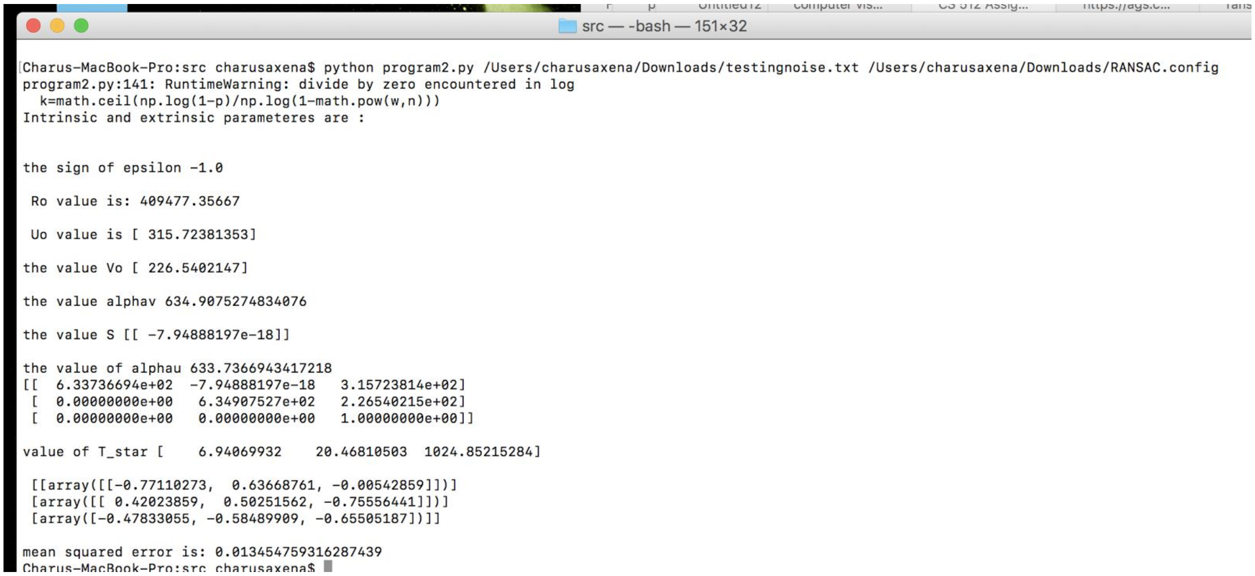


```
Charus-MacBook-Pro:src charusaxena$ python program3test.py /Users/charusaxena/Downloads/testingnoise.txt /Users/charusaxena/Downloads/RANSAC.config

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266

point are : 215.223810939 297.955280266
```

<u>Running program 2</u>

on noisy data entered: for evaluation used noisy data present on cs512 course website got following result:

Showing various extrinsic and extrinsic parameters and mean squared error:



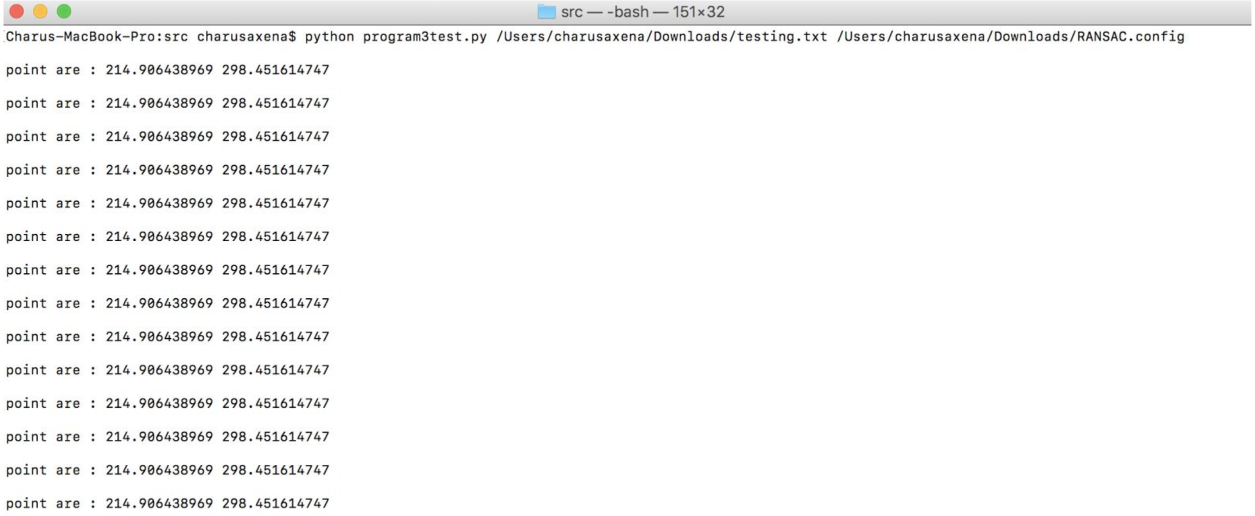
```
Charus-MacBook-Pro:src charusaxena$ python program2.py /Users/charusaxena/Downloads/testingnoise.txt /Users/charusaxena/Downloads/RANSAC.config
program2.py:141: RuntimeWarning: divide by zero encountered in log
  k=math.ceil(np.log(1-p)/np.log(1-math.pow(w,n)))
Intrinsic and extrinsic parameteres are :


the sign of epsilon -1.0

 Ro value is: 409477.35667

 Uo value is [ 315.72381353]

the value Vo [ 226.5402147]

the value alphav 634.9075274834076

the value S [[ -7.94888197e-18]]

the value of alphau 633.7366943417218
[[  6.33736694e+02  -7.94888197e-18   3.15723814e+02]
 [  0.00000000e+00   6.34907527e+02   2.26540215e+02]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]

value of T_star [    6.94069932    20.46810503  1024.85215284]

 [[array([[-0.77110273,  0.63668761, -0.00542859]])]
 [array([[ 0.42023859,  0.50251562, -0.75556441]])]
 [array([[-0.47833055, -0.58489909, -0.65505187]])]]

mean squared error is: 0.013454759316287439
Charus-MacBook-Pro:src charusaxena$
```

ii) **For non-noisy data**

<u>Running program 3 for test</u>

Using the noisy data cross-checked from data used in cs512 website generated following on terminal. Showing various extrinsic and extrinsic parameters and mean squared error

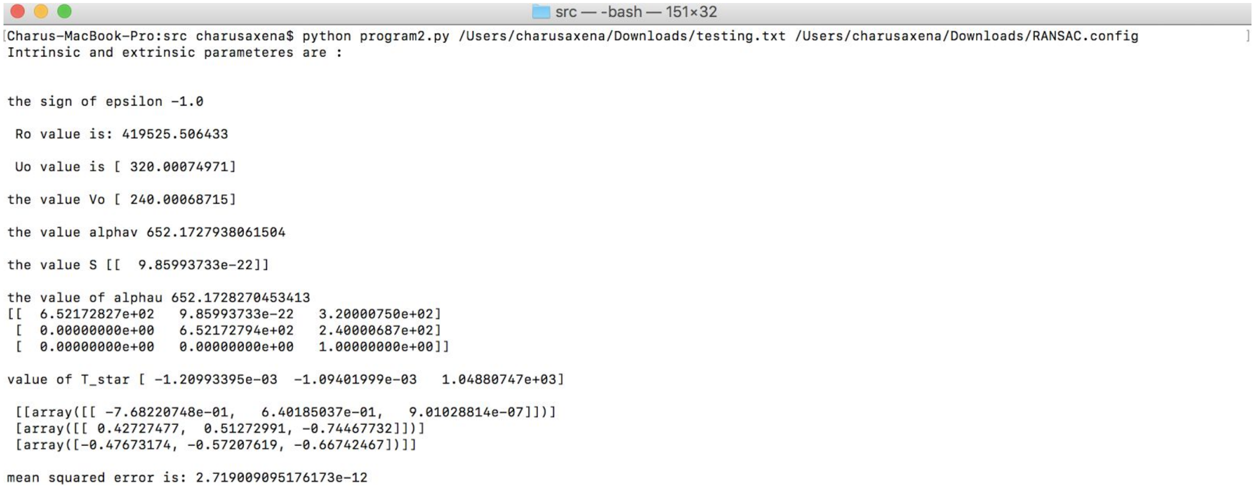
We observe that the values are exact same with the observed value:



```
Charus-MacBook-Pro:src charusaxena$ python program3test.py /Users/charusaxena/Downloads/testing.txt /Users/charusaxena/Downloads/RANSAC.config

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747

point are : 214.906438969 298.451614747
```

Running program 2 for test

Using the noisy data cross-checked from data used in cs512 website generated following on terminal.



```
[Charus-MacBook-Pro:src charusaxena$ python program2.py /Users/charusaxena/Downloads/testing.txt /Users/charusaxena/Downloads/RANSAC.config        ]
 Intrinsic and extrinsic parameteres are :


 the sign of epsilon -1.0

  Ro value is: 419525.506433

  Uo value is [ 320.00074971]

 the value Vo [ 240.00068715]

 the value alphav 652.1727938061504

 the value S [[  9.85993733e-22]]

 the value of alphau 652.1728270453413
 [[  6.52172827e+02   9.85993733e-22   3.20000750e+02]
  [  0.00000000e+00   6.52172794e+02   2.40000687e+02]
  [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]

 value of T_star [ -1.20993395e-03  -1.09401999e-03   1.04880747e+03]

  [[array([[ -7.68220748e-01,   6.40185037e-01,   9.01028814e-07]])]
  [array([[ 0.42727477,  0.51272991, -0.74467732]])]
  [array([-0.47673174, -0.57207619, -0.66742467])]]

 mean squared error is: 2.719009095176173e-12
```

Here we see that error is close to zero, thus this validates the result as error is close to zero and we are getting correct 2d points from 3d world points

The two conditions where ransac fail are:
When the sample size was more- then
And when w is big

5. Code

Program1:

#!/usr/bin/env python3
# -- coding: utf-8 -*-*
"""

```
Created on Sun Oct 22 17:34:22 2017

@author: charusaxena
"""

import cv2

import numpy as np
import math
import sys


#get image and thn gray it and float32 convertion
infile = sys.argv[1]
img = cv2.imread(infile)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)

#detection of corners
distance = cv2.cornerHarris(gray,2,3,0.04)
distance = cv2.dilate(distance,None)
ret, distance = cv2.threshold(distance,0.01*distance.max(),255,0)
dst = np.uint8(distance)

ret, l, s, c = cv2.connectedComponentsWithStats(dst)

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
detected_corners = cv2.cornerSubPix(gray,np.float32(c),(5,5),(-1,-1),criteria)

rows ,cols = detected_corners.shape
sqrt1 = math.ceil(math.pow(rows,1/3))

obj = np.zeros((sqrt1 *sqrt1*sqrt1,3), np.float32)
obj[:,:3] = np.mgrid[0:sqrt1,0:sqrt1,0:sqrt1].T.reshape(-1,3)

infile = open('points_generated.txt','w')

for i,j in zip(obj,detected_corners):
        X,Y,Z = i
        xi,yi = j.ravel()
        t = str(X) +" " + str(Y) + " "+str(Z) +" " + str(xi) + " " +str(yi)
        infile.write(t+'\n')
```

```
print("a file is generated!")
infile.close()
cv2.destroyAllWindows()
```

Program 2: program2.py

```
Created on Sat Nov 18 17:01:16 2017

@author: charusaxena
"""

import random
import numpy as np
import math
import sys

with open(sys.argv[1], 'r') as f:
    content = f.readlines()
content = [x.strip() for x in content]
data = content

with open(sys.argv[2], 'r') as q:
ransac_params = q.readlines()
ransac_params = [x.strip() for x in ransac_params]
max_k = float(ransac_params[0])
n=int(ransac_params[1])
p=float(ransac_params[2])




def make_A_matrix(data):
    data =np.array(data)
    total_matrix = []
    world_point=[]
    image_point=[]
    for i in data:
        X,Y,Z,xi,yi = i.split(' ')
        X = float(X)
        Y = float(Y)
        Z = float(Z)
        xi = float(xi)
        yi = float(yi)
        a= X,Y,Z,1,0,0,0,0,-xi*X,-xi*Y,-xi*Z,-xi
        b = 0,0,0,0, X,Y,Z,1,-yi*X,-yi*Y,-yi*Z,-yi
        c= X,Y,Z,1


        world_point.append(c)
        d= xi,yi
        image_point.append(d)
        matrix= []
        matrix.extend(a)
        total_matrix.append(matrix)
        matrix=[]
        matrix.extend(b)
        total_matrix.append(matrix)
    return total_matrix,world_point,image_point

def projection(matrix):
    U, s, V = np.linalg.svd(matrix)
    m = V.T[:,-1].reshape(3,4)
    return m

def thres(projection_matrix,test):
    a,b,c = make_A_matrix(test)
    distance=[]
    for i in range(len(test)):

        ui,vi,wi=np.matmul(projection_matrix,b[i])
        u=ui/wi
        v=vi/wi
        d,e = c[i]
        dist = math.sqrt(math.pow(d-u,2)+ math.pow(e-v,2))
        distance.append(dist)

    return (1.5*(np.median(distance)))



def inliers(projectionMatrix,data,thres):
    a,world_point,image_point = make_A_matrix(data)

    myinliers=[]
    for i in range(len(data)):

        ui,vi,wi=np.matmul(projection_matrix,world_point[i])
        u=ui/wi
        v=vi/wi
        a,b = image_point[i]
        dist = math.sqrt(math.pow(a-u,2)+ math.pow(b-v,2))
        if dist < thres:
            myinliers.append(data[i])

    return myinliers,len(myinliers)

def print_parameters(projection_matrix):
    KR = projection_matrix[:3,:3]
    KT = projection_matrix[:,-1]
    print("Intrinsic and extrinsic parameters are :\n")
    ro = 1/np.linalg.norm(KR[-1])

    print("\nThe value of roh is",ro)
    Uo = ((ro)**2)*np.dot(KR[:1],KR[-1])
    print("\nthe value Uo",Uo)
    Vo = ((ro)**2)*np.dot(KR[1:2],KR[-1])
    print("\nthe value Vo",Vo)
    alphav =math.sqrt( (((ro)**2)*np.dot(KR[1:2],KR[1:2].T)) -((Vo)**2))
    print("\nthe value alphav",alphav)
        s=(alphav*np.dot(np.cross(KR[:1],KR[-1]),np.cross(KR[1:2],KR[-1]).T))
        print("\nthe value S",s)
 alphau = math.sqrt((math.pow(ro,2)*np.dot(KR[:1],KR[:1].T)) - math.pow(s,2) - math.pow(Uo,2))
    print("\nthe value of alphau",alphau)
    K_str = np.array([[alphau,s,Uo],[0,alphav,Vo],[0,0,1]])
    print(K_str)
    print("\nthe sign of epsilon",np.sign(KT[-1]))
    T_star = np.sign(KT[-1])*ro*(np.matmul((np.linalg.inv(K_str)),KT))
    print("\nvalue of T_star",T_star)
    R3 = np.sign(KT[-1])*ro*KR[-1]
    R1= math.pow(Ro,2)/alphav*(np.cross(KR[1:2],KR[-1]))
    R2 = np.cross(R3,R1)
    R_matrix = np.array([[R1],[R2],[R3]])
    print("\n",R_matrix)


#Ransac Implementation
max_k =10000
n=6
count=0
all_S=[]
```

```
all_len=[]
k=math.ceil(np.log(1-0.99)/np.log(1-math.pow(0.5,n)))
#ransac
while count<k:
    count=count+1
    if (count>max_k):
        break
    else:
        test = random.sample(content,n)
        mat,worldpoint,imagepoint = make_A_matrix(test)

        projection_matrix=projection(mat)
        threshold  = thres(projection_matrix,test)

        new_s,length = inliers(projection_matrix,data,threshold)

        w = len(new_s)/len(content)
        k=math.ceil(np.log(1-0.99)/np.log(1-math.pow(w,n)))

        if length > 6:
            all_S.append(new_s)
            all_len.append(length)

b = all_S[np.argmax(all_len)]
matt,worldpoint,imagepoint = make_A_matrix(all_S[np.argmax(all_len)])
final_projection_matrix=projection(matt)
print_parameters(final_projection_matrix)

##mean square error
g,world,image = make_A_matrix(data)
estimates_image_point=[]
for i in range(len(data)):
        uei,vei,wei=np.matmul(final_projection_matrix,world[i])
        ue=uei/wei
        ve=vei/wei
        de,ee = image[i]
        error = ((math.pow(de-ue,2)+ math.pow(ee-ve,2)))/len(data)

print("\nmean squared error is:",error)
```

Program3test.py

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 19 11:38:20 2017

@author: charusaxena
"""

import numpy as np
import math
import sys

with open(sys.argv[1], 'r') as f:
    content = f.readlines()
content = [x.strip() for x in content]
data = content



def make_A_matrix(data):
    data =np.array(data)
    total_matrix = []
    world_point=[]
    image_point=[]
    for i in data:
        X,Y,Z,w,xi,yi = i.split(' ')
        X = float(X)
        Y = float(Y)
        Z = float(Z)
        xi = float(xi)
        yi = float(yi)
        a= X,Y,Z,1,0,0,0,0,-xi*X,-xi*Y,-xi*Z,-xi
        b = 0,0,0,0, X,Y,Z,1,-yi*X,-yi*Y,-yi*Z,-yi
        c= X,Y,Z,1
        world_point.append(c)
        d= xi,yi
        image_point.append(d)
        matrix= []
        matrix.extend(a)
        total_matrix.append(matrix)
        matrix=[]
        matrix.extend(b)
        total_matrix.append(matrix)
    return total_matrix,world_point,image_point
```

```
def projection(matrix):
    U, s, V = np.linalg.svd(matrix)
    m = V.T[:,-1].reshape(3,4)
    return m

mat,worldpoint,imagepoint = make_A_matrix(data)
projection_matrix=projection(mat)

for i in range(len(data)):
    ui,vi,wi=np.matmul(projection_matrix,worldpoint[0])
    u=ui/wi
    v=vi/wi
    print("\npoint are :",u,v)
```

Ref:
https://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/slides/CS4495-07-Calibration.pdf