

```
In [1]: 1 import cv2
        2 import pickle
        3 import camutils
        4 from camutils import Camera
        5 from camutils import triangulate
        6 from camutils import calibratePose
        7 from camutils import reconstruct
        8 import meshutils
        9 import visutils
       10 import numpy as np
       11 import matplotlib.pyplot as plt
       12 import trimesh
       13
```

```
In [2]: 1 # load in the intrinsic camera parameters from 'calibration.pickle'
        2 file = open('calibration.pickle','rb')
        3 object_file = pickle.load(file)
        4 fx,fy,cx,cy,dist = object_file['fx'],object_file['fy'],object_file['cx'],object_file['cy'],ob
        5
        6
```

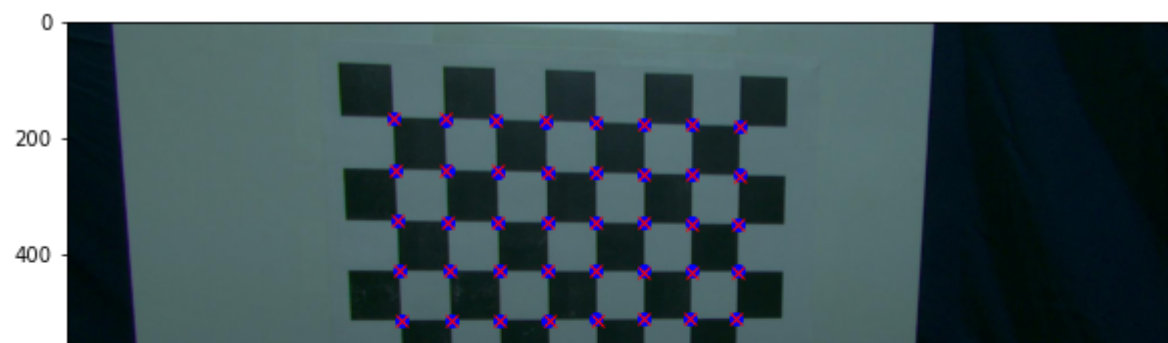
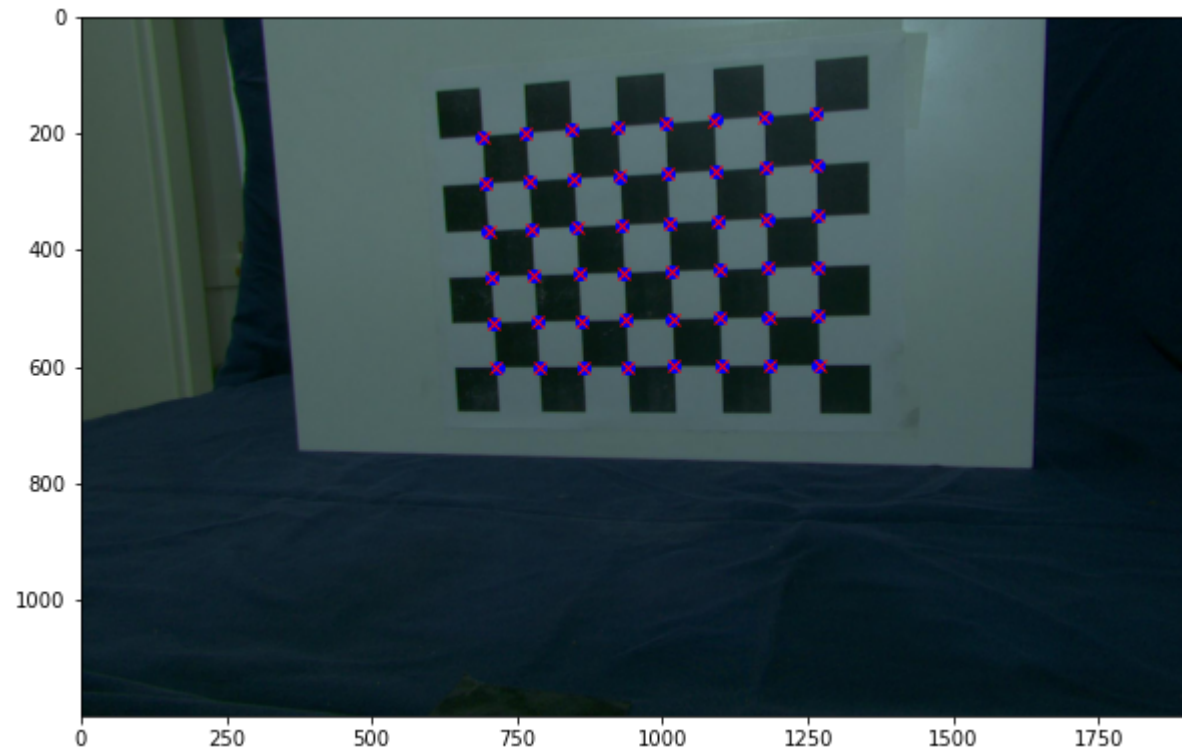
```
In [3]: 1 c = np.array([[cx],[cy]])
        2 f = (fx+fy)/2
        3
        4 # create Camera objects representing the left and right cameras
        5 # use the known intrinsic parameters you loaded in.
        6 R = np.zeros((3,3))
        7 t = np.zeros((3,1))
        8 camL = Camera(f,c,R,t)
        9 camR = Camera(f,c,R,t)
       10
       11 # load in the left and right images and find the coordinates of
       12 # the chessboard corners using OpenCV
       13 imgL = plt.imread('calib_jpg_u/frame_C0_01.jpg')
       14 ret, cornersL = cv2.findChessboardCorners(imgL, (8,6), None)
       15 pts2L = cornersL.squeeze().T
       16
       17 imgR = plt.imread('calib_jpg_u/frame_C1_01.jpg')
       18 gray = cv2.cvtColor(imgR, cv2.COLOR_BGR2GRAY)
       19 ret, cornersR = cv2.findChessboardCorners(gray, (8,6), None)
       20 pts2R = cornersR.squeeze().T
```

```

21
22 # generate the known 3D point coordinates of points on the checkerboard in cm
23 pts3 = np.zeros((3,6*8))
24 yy,xx = np.meshgrid(np.arange(8),np.arange(6))
25 pts3[0,:] = 2.8*xx.reshape(1,-1)
26 pts3[1,:] = 2.8*yy.reshape(1,-1)
27
28 # Now use your calibratePose function to get the extrinsic parameters
29 # for the two images. You may need to experiment with the initialization
30 # in order to get a good result
31
32 #calibratePose(pts3,pts2,cam,params_init)
33 params_init = np.array([0,0,0,0,1,-1])
34 camL = calibratePose(pts3,pts2L,camL,params_init)
35 camR = calibratePose(pts3,pts2R,camR,params_init)
36
37 print(camL)
38 print(camR)
39
40 # As a final test, triangulate the corners of the checkerboard to get back there 3D locations
41 pts3r = triangulate(pts2L,camL,pts2R,camR)
42
43 # Display the reprojected points overlayed on the images to make
44 # sure they line up
45 plt.rcParams['figure.figsize']=[10,10]
46 pts2Lp = camL.project(pts3)
47 plt.imshow(imgL)
48 plt.plot(pts2Lp[0,:],pts2Lp[1,:],'bo')
49 plt.plot(pts2L[0,:],pts2L[1,:],'rx')
50 plt.show()
51
52 pts2Rp = camR.project(pts3)
53 plt.imshow(imgR)
54 plt.plot(pts2Rp[0,:],pts2Rp[1,:],'bo')
55 plt.plot(pts2R[0,:],pts2R[1,:],'rx')
56 plt.show()
57
Camera :
f=1404.6009664593485
c=[[962.16736834 590.91595778]]
R=[[ 0.03843674  0.98947411  0.13951198]
 [ 0.9773577  -0.00815434 -0.2114366 ]

```

```
[-0.20807341  0.14448005 -0.96738357]]  
t = [[ 6.86588564 19.5234718  47.34419111]]  
Camera :  
f=1404.6009664593485  
c=[962.16736834 590.91595778]]  
R=[[-0.00259871  0.99096865  0.13406856]  
[ 0.99277874 -0.01352252  0.11919527]  
[ 0.11993172  0.13341017 -0.98377747]]  
+ - [[ 7.50010574  7.20025006 47.7610520 11
```



In [4]:

```
1000
1 #
2 # first view
3 #
4 imprefixC0 = 'couple/grab_0_u/frame_C0_'
5 imprefixC1 = 'couple/grab_0_u/frame_C1_'
6 colors = ['couple/grab_0_u/color_C0_00.png',
7           'couple/grab_0_u/color_C0_01.png',
8           'couple/grab_0_u/color_C1_00.png',
9           'couple/grab_0_u/color_C1_01.png']
10 threshold = 0.025
11 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
12 boxlimits = np.array([-12,18,2,17,-30,-3])
13 trithresh = 2
14 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R, pts3, cpix)
15 pts3 = meshutils.smoothIt(tri, pts3,4)
16 #meshutils.writeply(pts3,cpix,tri,'first.ply')
17
18 #
19 # second view
20 #
21 imprefixC0 = 'couple/grab_1_u/frame_C0_'
22 imprefixC1 = 'couple/grab_1_u/frame_C1_'
23 colors = ['couple/grab_1_u/color_C0_00.png',
24           'couple/grab_1_u/color_C0_01.png',
25           'couple/grab_1_u/color_C1_00.png',
26           'couple/grab_1_u/color_C1_01.png']
27 threshold = 0.025
28 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
29 boxlimits = np.array([-25,22,0,25,-35,10])
30 trithresh = 5
31 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R, pts3, cpix)
32 pts3 = meshutils.smoothIt(tri, pts3,4)
33 meshutils.writeply(pts3,cpix,tri,'second.ply')
34 #
35 # third view
36 #
37 imprefixC0 = 'couple/grab_2_u/frame_C0_'
38 imprefixC1 = 'couple/grab_2_u/frame_C1_'
39 colors = ['couple/grab_2_u/color_C0_00.png',
40           'couple/grab_2_u/color_C0_01.png',
```

```

41         'couple/grab_2_u/color_C1_00.png',
42         'couple/grab_2_u/color_C1_01.png']
43 threshold = 0.025
44
45 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
46 boxlimits = np.array([-10,23,-10,22,-30,-10])
47 trithresh = 4
48 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)
49 pts3 = meshutils.smoothIt(tri, pts3,4)
50 meshutils.writeply(pts3,cpix,tri,'third.ply')
51 #
52 # forth view
53 #
54 imprefixC0 = 'couple/grab_3_u/frame_C0_'
55 imprefixC1 = 'couple/grab_3_u/frame_C1_'
56 colors = ['couple/grab_3_u/color_C0_00.png',
57           'couple/grab_3_u/color_C0_01.png',
58           'couple/grab_3_u/color_C1_00.png',
59           'couple/grab_3_u/color_C1_01.png']
60 threshold = 0.025
61
62 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
63 boxlimits = np.array([-15,21,0,18,-30,-8])
64 trithresh = 4
65 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)
66 pts3 = meshutils.smoothIt(tri, pts3,4)
67 meshutils.writeply(pts3,cpix,tri,'forth.ply')
68 #
69 # fifth view
70 #
71 imprefixC0 = 'couple/grab_4_u/frame_C0_'
72 imprefixC1 = 'couple/grab_4_u/frame_C1_'
73 colors = ['couple/grab_4_u/color_C0_00.png',
74           'couple/grab_4_u/color_C0_01.png',
75           'couple/grab_4_u/color_C1_00.png',
76           'couple/grab_4_u/color_C1_01.png']
77 threshold = 0.025
78
79 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
80 boxlimits = np.array([-8,21,0,18,-30,-8])
81 trithresh = 4
82 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)

```

```

83 pts3 = meshutils.smoothIt(tri, pts3,4)
84 meshutils.writeply(pts3,cpix,tri,'fifth.ply')
85 #
86 # sixth view
87 #
88 imprefixC0 = 'couple/grab_5_u/frame_C0_'
89 imprefixC1 = 'couple/grab_5_u/frame_C1_'
90 colors = ['couple/grab_5_u/color_C0_00.png',
91           'couple/grab_5_u/color_C0_01.png',
92           'couple/grab_5_u/color_C1_00.png',
93           'couple/grab_5_u/color_C1_01.png']
94 threshold = 0.025
95
96 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
97 boxlimits = np.array([-15,21,3,35,-30,-8])
98 trithresh = 4
99 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)
100 pts3 = meshutils.smoothIt(tri, pts3,4)
101 meshutils.writeply(pts3,cpix,tri,'sixth.ply')
102 #
103 # seventh view
104 #
105 imprefixC0 = 'couple/grab_6_u/frame_C0_'
106 imprefixC1 = 'couple/grab_6_u/frame_C1_'
107 colors = ['couple/grab_6_u/color_C0_00.png',
108           'couple/grab_6_u/color_C0_01.png',
109           'couple/grab_6_u/color_C1_00.png',
110           'couple/grab_6_u/color_C1_01.png']
111 threshold = 0.025
112
113 pts2L,pts2R,pts3,cpix = reconstruct(imprefixC0,imprefixC1,threshold,camL,camR,colors)
114 boxlimits = np.array([13,22,-10,25,-30,-1])
115 trithresh = 2.5
116 tri, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)
117 pts3 = meshutils.smoothIt(tri, pts3, 4)
118 meshutils.writeply(pts3,cpix,tri,'seventh.ply')
loading( 0 1 )( 2 3 )( 4 5 )( 6 7 )( 8 9 )( 10 11 )( 12 13 )( 14 15 )( 16 17 )( 18 19 )

loading( 20 21 )( 22 23 )( 24 25 )( 26 27 )( 28 29 )( 30 31 )( 32 33 )( 34 35 )( 36 37 )( 38 39 )

loading( 0 1 )( 2 3 )( 4 5 )( 6 7 )( 8 9 )( 10 11 )( 12 13 )( 14 15 )( 16 17 )( 18 19 )

```

[illegible]

loading(0 1)(2 3)(4 5)(6 7)(8 9)(10 11)(12 13)(14 15)(16 17)(18 19)

loading(20 21)(22 23)(24 25)(26 27)(28 29)(30 31)(32 33)(34 35)(36 37)(38 39)

loading(0 1)(2 3)(4 5)(6 7)(8 9)(10 11)(12 13)(14 15)(16 17)(18 19)

loading(20 21)(22 23)(24 25)(26 27)(28 29)(30 31)(32 33)(34 35)(36 37)(38 39)