CameraMotion

October 5, 2021

$x_i = \text{pinhole}[w_i, \Lambda, \Omega, \tau]$

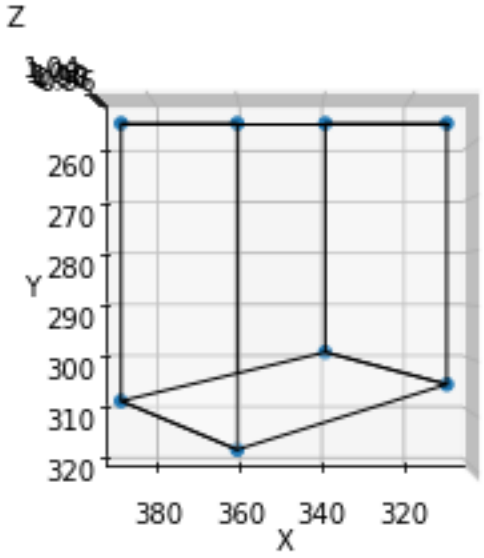
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
```

1 Using previous values

[2]:
```
camera1deg = 35

camera2deg = -35

#skew factor
gamma = 0

#focal distances
phi = np.array([200,200])

#Principal point
delta = np.array([256,256])

#cube coordinates
C = np.array([[0, 50, 50, 0, 0, 50, 50, 0],
              [0, 0, 50, 50, 0, 0, 50, 50],
              [200, 200, 200, 200, 250, 250, 250, 250]])

#make the world coordinates homogenous
homogenousW = np.vstack((C, np.ones([1,8])))


tau_1 = np.array([[200,0,0]])

tau_2 = np.array([[-200,0,0]])

#rotation
omegav = np.array([[np.cos(35 * np.pi/180), 0, -np.sin(35 * np.pi/180)],
                   [0, 1, 0],
```

```
                    [np.sin(35 * np.pi/180), 0 , np.cos(35 * np.pi/180)]])



#rotation and translation
omegatau = np.hstack([omegav,tau_1.T])

#intrinsic matrix
Lambda = np.array([[phi[0], gamma, delta[0]],
                   [0, phi[1], delta[1]],
                   [0, 0, 1]])


print(Lambda.shape)
print(omegatau.shape)
print(C.shape)


box = Lambda @ omegatau @ homogenousW

#divide by to the z values in order to make it 1
box = box/box[2]

box = box.T


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

print(box)


ax.scatter3D(box[:, 0], box[: ,1], box[:, 2])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')


verts = [[box[0],box[1],box[2],box[3]], [box[4],box[5],box[6],box[7]],␣
 ↪[box[0],box[1],box[5],box[4]], [box[3],box[2],box[6],box[7]]]
        #[box[1],box[2],box[6],box[5]], [box[1],box[2],box[5],box[7]]]

# plot sides
ax.add_collection3d(Poly3DCollection(verts, facecolors='w', linewidths=1,␣
 ↪edgecolors='k', alpha=.25))
ax.view_init(90,90)
```

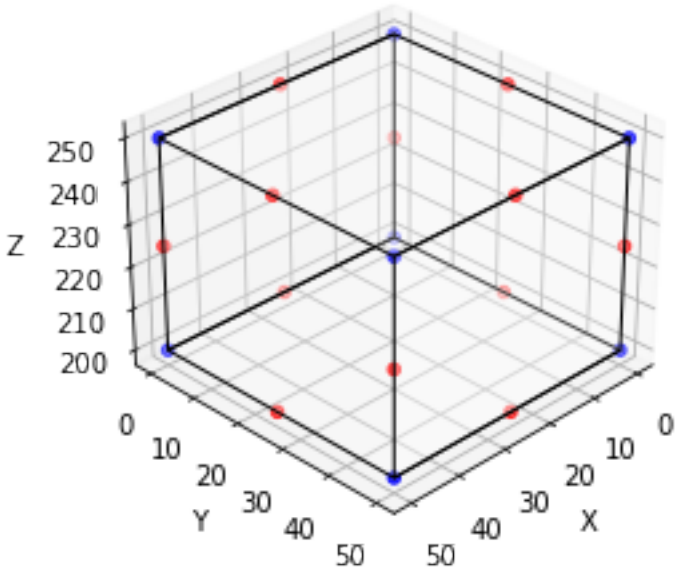
```
(3, 3)
(3, 4)
(3, 8)
[[360.11341011 256.           1.        ]
 [387.15455763 256.           1.        ]
 [387.15455763 307.94556108   1.        ]
 [360.11341011 317.03872944   1.        ]
 [311.28242656 256.           1.        ]
 [339.57803284 256.           1.        ]
 [339.57803284 298.83263655   1.        ]
 [311.28242656 304.83098355   1.        ]]
```



2 Add additional points

[3]:
```
threedbox = C.T

additionalpoints = np.array([[25, 50, 25, 0, 0, 50, 50, 0, 25, 50, 25, 0],
                             [0, 0, 0, 0, 25, 25, 25, 25, 50, 50, 50, 50],
                             [200, 225, 250, 225, 200, 200, 250, 250, 200, 225,␣
 ↪250, 225]])

newpoints = additionalpoints.T

fignew = plt.figure()
ax = fignew.add_subplot(111, projection='3d')
```

```
threedverts = [[threedbox[0],threedbox[1],threedbox[2],threedbox[3]],
               [threedbox[4],threedbox[5],threedbox[6],threedbox[7]],
               [threedbox[0],threedbox[1],threedbox[5],threedbox[4]],
               [threedbox[3],threedbox[2],threedbox[6],threedbox[7]]]

ax.scatter3D(threedbox[:, 0], threedbox[: ,1], threedbox[:, 2], c='blue')
ax.scatter3D(newpoints[:, 0], newpoints[:, 1], newpoints[:, 2], c='red')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')


ax.add_collection3d(Poly3DCollection(threedverts, facecolors='w', linewidths=1,␣
 ↪edgecolors='k', alpha=.25))
ax.view_init(30, 45)

# print(threebox)
```



3 Finding normalized image coordinates by using homogenous world points, intrinsic matrix

[4]:
```
synthethicW = np.concatenate((C, additionalpoints), axis = 1)
synthethicW = np.vstack((synthethicW, np.ones([1,20])))
# print(np.concatenate((homogenousW, additionalpoints), axis = 1).shape)
```

```
print(synthethicW.T)

print(Lambda)

print(omegatau)

#equation 14.26
X =  Lambda @ omegatau @ synthethicW
# X = Lambda @ omegatau @ homogenousW
# X_ = omegatau @ synthethicW

X = X/X[2]

print("\nX:\n", X)


#equation 14.27
X_ = np.linalg.inv(Lambda) @ X


#Divide by lambda to get the x and y values

#X_[2] is lambda
X_ = X_/X_[2]

print("\nX\':\n", X_.T)
```

```
[[  0.   0. 200.   1.]
 [ 50.   0. 200.   1.]
 [ 50.  50. 200.   1.]
 [  0.  50. 200.   1.]
 [  0.   0. 250.   1.]
 [ 50.   0. 250.   1.]
 [ 50.  50. 250.   1.]
 [  0.  50. 250.   1.]
 [ 25.   0. 200.   1.]
 [ 50.   0. 225.   1.]
 [ 25.   0. 250.   1.]
 [  0.   0. 225.   1.]
 [  0.  25. 200.   1.]
 [ 50.  25. 200.   1.]
 [ 50.  25. 250.   1.]
 [  0.  25. 250.   1.]
 [ 25.  50. 200.   1.]
 [ 50.  50. 225.   1.]
 [ 25.  50. 250.   1.]
 [  0.  50. 225.   1.]]
[[200   0 256]
```

```
 [  0 200 256]
 [  0   0   1]]
[[  0.81915204   0.          -0.57357644 200.        ]
 [  0.           1.           0.           0.        ]
 [  0.57357644   0.           0.81915204   0.        ]]

X:
 [[360.11341011 387.15455763 387.15455763 360.11341011 311.28242656
  339.57803284 339.57803284 311.28242656 374.7221427  361.0790536
  326.35604337 332.98508592 360.11341011 387.15455763 339.57803284
  311.28242656 374.7221427  361.0790536  326.35604337 332.98508592]
 [256.         256.         307.94556108 317.03872944 256.
  256.         298.83263655 304.83098355 256.         256.
  256.         256.         286.51936472 281.97278054 277.41631828
  280.41549178 312.12622841 302.95099493 301.63554807 310.25664839]
 [  1.           1.           1.           1.           1.
    1.           1.           1.           1.           1.
    1.           1.           1.           1.           1.
    1.           1.           1.           1.           1.        ]]

X':
 [[0.52056705 0.         1.        ]
 [0.65577279 0.         1.        ]
 [0.65577279 0.25972781 1.        ]
 [0.52056705 0.30519365 1.        ]
 [0.27641213 0.         1.        ]
 [0.41789016 0.         1.        ]
 [0.41789016 0.21416318 1.        ]
 [0.27641213 0.24415492 1.        ]
 [0.59361071 0.         1.        ]
 [0.52539527 0.         1.        ]
 [0.35178022 0.         1.        ]
 [0.38492543 0.         1.        ]
 [0.52056705 0.15259682 1.        ]
 [0.65577279 0.1298639  1.        ]
 [0.41789016 0.10708159 1.        ]
 [0.27641213 0.12207746 1.        ]
 [0.59361071 0.28063114 1.        ]
 [0.52539527 0.23475497 1.        ]
 [0.35178022 0.22817774 1.        ]
 [0.38492543 0.27128324 1.        ]]
```

4 Funciton to construct system of equation

[5]:
```
def constructA(W, X_):
    #make empty array to be able to populate
#     print(W.shape)
    A = np.zeros((W.shape[0]*2, 12))
    print(A.shape)
    i = 0
    for w, x_ in zip(W, X_):
        # print("\nw: ",w)
        # print("x: ",x_)
        A[i] = np.array([w[0], w[1], w[2], 1, 0 ,0 ,0 , -w[0]*x_[0],␣
↪-w[1]*x_[0], -w[2]*x_[0], -x_[0]])
        A[i+1] =  np.array([0, 0, 0 ,0 ,w[0], w[1], w[2], 1 , -w[0]*x_[1],␣
↪-w[1]*x_[1], -w[2]*x_[1], -x_[1]])
        # print(A[i])
        # print(A[i+1])
        i += 2

    print("iteration:",i)
    return A
```

5 Using SVD to find b and solve for Tau_hat

[6]:
```
# w = np.concatenate((C, additionalpoints), axis = 1)

# w = C
# print(w.T)
# print("SynthethicW:\n", synthethicW.shape, "\n X_:\n", X_.shape)


# print("SynthethicW:\n", synthethicW.T, "\n\n X_:\n", X_.T)

A = constructA(synthethicW.T, X_.T)
# A = constructA(homogenousW.T, X_.T)
# print(A)

U,L,V = np.linalg.svd(A)

# print(U)
# print(L)
# print(V)
#set b hat equal to last column
b_ = V.T[:,-1]
print("\nb hat : \n", b_)
```

```
Omega = np.array([[b_[0],b_[1],b_[2]],
                  [b_[4],b_[5],b_[6]],
                  [b_[8],b_[9],b_[10]]])

print("Original omega:\n", omegav)

print("Omega:\n",Omega)

Tau = np.array([b_[3], b_[7], b_[11]])

print("Tau: \n:", Tau)
#every fourth point is tau
U_, L_, V_ = np.linalg.svd(Omega)

Omega_hat = -(U_@V_)

print("Omega hat \n:", Omega_hat)

Tau_hat = np.array([np.sum(Omega_hat)/np.sum(Omega)*Tau])

# Tau_hat = Tau_hat *-1

# Tau_hat = Tau_hat*10

print("Tau hat: \n", Tau_hat)
```

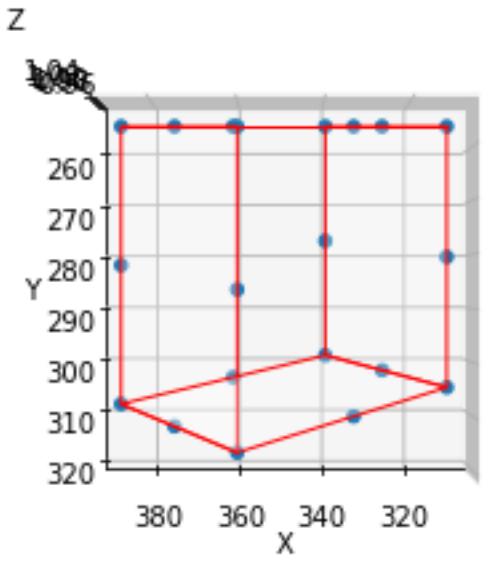
```
(40, 12)
iteration: 40

b hat :
 [-4.09560664e-03 -1.52168738e-17  2.86777464e-03 -9.99962502e-01
 -1.47162674e-16 -4.99981251e-03  2.08537291e-16 -2.49582420e-15
 -2.86777464e-03  1.51238371e-17 -4.09560664e-03  1.84133995e-15]
Original omega:
 [[ 0.81915204  0.         -0.57357644]
 [ 0.          1.          0.        ]
 [ 0.57357644  0.          0.81915204]]
Omega:
 [[-4.09560664e-03 -1.52168738e-17  2.86777464e-03]
 [-1.47162674e-16 -4.99981251e-03  2.08537291e-16]
 [-2.86777464e-03  1.51238371e-17 -4.09560664e-03]]
Tau:
: [-9.99962502e-01 -2.49582420e-15  1.84133995e-15]
Omega hat
: [[ 8.19152044e-01 -2.24867053e-14 -5.73576436e-01]
 [ 1.42972505e-14  1.00000000e+00 -1.85922062e-14]
 [ 5.73576436e-01  7.28583650e-15  8.19152044e-01]]
```

```
Tau hat:
 [[ 2.00000000e+02  4.99183558e-13 -3.68281799e-13]]
```

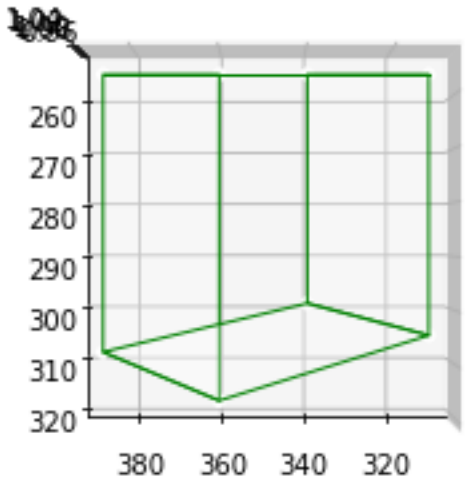
[11]:
```
print(Omega.shape)
print(Tau_hat.shape)
OmegaTau = np.hstack([Omega_hat, Tau_hat.T])

# print(homogenousW)

projectedbox = Lambda @ OmegaTau @ synthethicW

#divide by to the z values in order to make it 1
projectedbox = projectedbox/projectedbox[2]

projectedbox = projectedbox.T


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

print(projectedbox)


ax.scatter3D(projectedbox[:, 0], projectedbox[: ,1], projectedbox[:, 2])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')


projverts = [[projectedbox[0],projectedbox[1],projectedbox[2],projectedbox[3]],
         [projectedbox[4],projectedbox[5],projectedbox[6],projectedbox[7]],
         [projectedbox[0],projectedbox[1],projectedbox[5],projectedbox[4]],
         [projectedbox[3],projectedbox[2],projectedbox[6],projectedbox[7]]]
         #[box[1],box[2],box[6],box[5]], [box[1],box[2],box[5],box[7]]]

# plot sides
ax.add_collection3d(Poly3DCollection(projverts, facecolors='w', linewidths=1,␣
 ↪edgecolors='r', alpha=.25))
ax.view_init(90,90)
```

```
(3, 3)
(1, 3)
[[360.11341011 256.           1.        ]
 [387.15455763 256.           1.        ]
 [387.15455763 307.94556108   1.        ]
 [360.11341011 317.03872944   1.        ]
 [311.28242656 256.           1.        ]
 [339.57803284 256.           1.        ]
```

```
[339.57803284 298.83263655   1.        ]
[311.28242656 304.83098355   1.        ]
[374.7221427  256.           1.        ]
[361.0790536  256.           1.        ]
[326.35604337 256.           1.        ]
[332.98508592 256.           1.        ]
[360.11341011 286.51936472   1.        ]
[387.15455763 281.97278054   1.        ]
[339.57803284 277.41631828   1.        ]
[311.28242656 280.41549178   1.        ]
[374.7221427  312.12622841   1.        ]
[361.0790536  302.95099493   1.        ]
[326.35604337 301.63554807   1.        ]
[332.98508592 310.25664839   1.        ]]
```



[16]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter3D(box[:, 0], box[: ,1], box[:, 2], c="w")

verts = [[box[0],box[1],box[2],box[3]], [box[4],box[5],box[6],box[7]],␣
 ↪[box[0],box[1],box[5],box[4]], [box[3],box[2],box[6],box[7]]]
        #[box[1],box[2],box[6],box[5]], [box[1],box[2],box[5],box[7]]]

# plot sides
ax.add_collection3d(Poly3DCollection(verts, facecolors='w', linewidths=1,␣
 ↪edgecolors='g', alpha=.25))
ax.view_init(90,90)
```



Projected image with estimation seems to be working fine with

[8]:
```
# print(projectedbox[:8] - box)


# error = np.sqrt(projectedbox[:8][0]**2 + box **2)

# print(error)
```

[ ]:
```

```