CameraMotion

September 30, 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

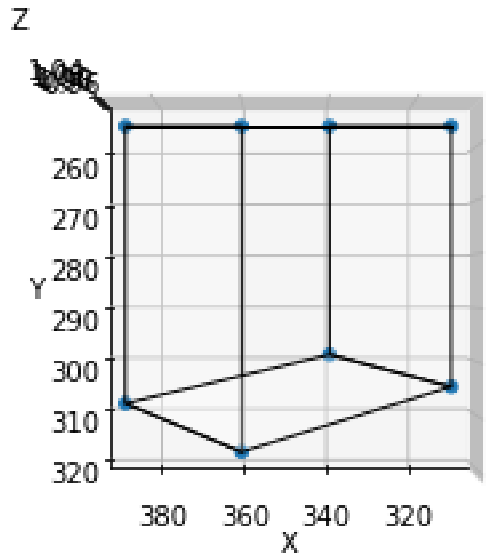
[2]:

```
#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 = 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.T])
```

```python
#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.        ]
```

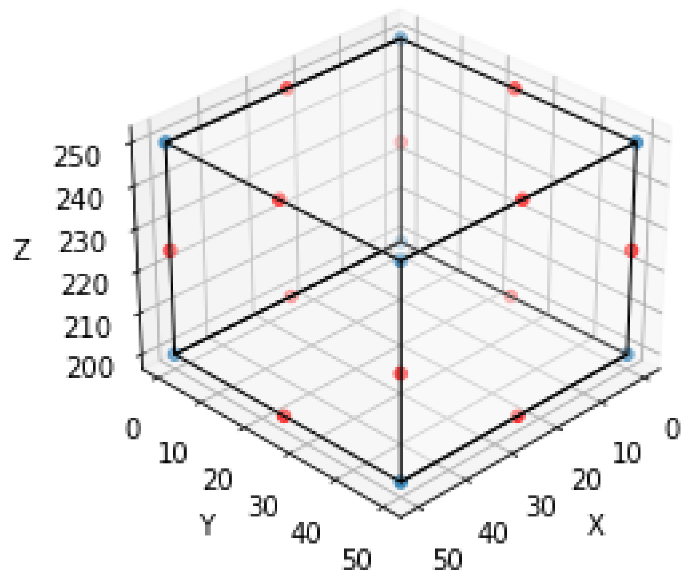
```
 [360.11341011 317.03872944   1.        ]
 [311.28242656 256.           1.        ]
 [339.57803284 256.           1.        ]
 [339.57803284 298.83263655   1.        ]
 [311.28242656 304.83098355   1.        ]]
```



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])
```

```
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)

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

X = X/X[2]
```

```python
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)
```

```
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)
    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 ,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 += 1

    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)

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

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

#set b hat equal to last column
b_ = V.T[:,-1]
print("b hat : ", 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/Omega)*Tau])


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

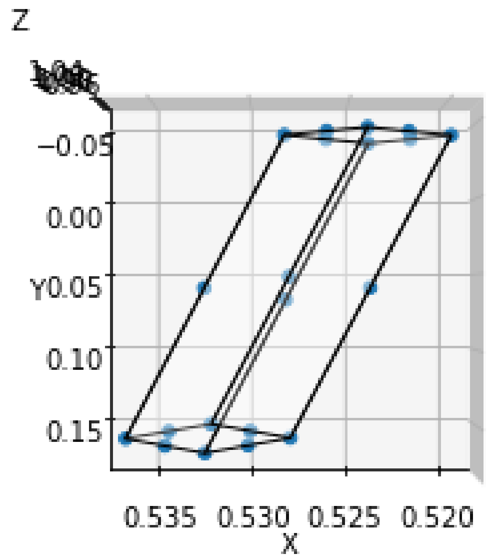
```
b hat :  [ 0.02663327  0.01225753  0.02286972  0.42652162  0.31593962
-0.00989123
 -0.00989123 -0.00989123  0.02623778  0.03598295  0.03098721  0.84477332]
Original omega:
 [[ 0.81915204  0.          -0.57357644]
 [ 0.          1.          0.        ]
 [ 0.57357644  0.          0.81915204]]
Omega:
 [[ 0.02663327  0.01225753  0.02286972]
 [ 0.31593962 -0.00989123 -0.00989123]
 [ 0.02623778  0.03598295  0.03098721]]
Tau:
: [ 0.42652162 -0.00989123  0.84477332]
Omega hat
: [[ 0.09252912 -0.30056531  0.94926227]
 [ 0.99051145 -0.0695064  -0.11855771]
 [ 0.10161414  0.95122519  0.29128201]]
Tau hat:
 [[35.11013493 -0.81422014 69.53951215]]
```

[7]:

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

projectedbox = OmegaTau @ synthethicW

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

```
box = projectedbox.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)
(1, 3)
[[ 0.52397248 -0.03687059  1.        ]
 [ 0.53233432  0.16878273  1.        ]
 [ 0.52796044  0.15865912  1.        ]
 [ 0.51971862 -0.04239352  1.        ]
 [ 0.52826385 -0.04253053  1.        ]
 [ 0.53638922  0.15916334  1.        ]
 [ 0.53200889  0.14945009  1.        ]
 [ 0.5239957  -0.0478165   1.        ]
 [ 0.5281893   0.06683899  1.        ]
 [ 0.53438195  0.16392516  1.        ]
 [ 0.53236073  0.05916511  1.        ]
 [ 0.52613989 -0.03972922  1.        ]
 [ 0.52182059 -0.03966447  1.        ]
 [ 0.53012214  0.16366251  1.        ]
 [ 0.53417427  0.15425176  1.        ]
 [ 0.52610522 -0.04520392  1.        ]
 [ 0.5238741   0.0589761   1.        ]
 [ 0.53000436  0.1540098   1.        ]
 [ 0.52803525  0.05162814  1.        ]
 [ 0.52187832 -0.04513183  1.        ]]
```


Z
−0.05
0.00
Y0.05
0.10
0.15
0.535
0.530
0.525
0.520
X

[ ]:

[ ]: