After your finish the assignment, remember to run all cells and save the note book to your local machine as a PDF for gradescope submission by pressing Ctrl-P or Cmd-P. Make sure images are not split between pages; insert Text blocks to make sure this is the case before printing to PDF!

List your collaborators here:

16720 HW 4: 3D Reconstruction

Problem 1: Theory

1.1

See pdf for the question.

===== your answer here for 1.1! =====

Given according to the diagram the two points x and x' are $(0, 0)$. We know that we can write the epipolar constraints as:

$$x_2^T F x_1 = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Thus,

$$f_{33} = 0$$

===== end of your answer for 1.1 =====

1.2

See pdf for the question.

===== your answer here for 1.2! =====

At time i, we have R_i and t_i

At time $i+1$, we have R_{i+1} and t_{i+1}

Relative rotation and translation between frames i and $i+1$:

At time i, point in camera coordinate frame X_i using world coordinate frame X_w

$$X_i = R_i X_w + t_i$$

We can write the above equation for world frame coordinates as:

$$X_w = R_i^{-1}(X_i - t_i)$$

Since R_i is orthonormal matrices the inverse is equal to transpose so we can write the above equation as:

$$X_w = R_i^T(X_i - t_i)$$

At time i, point in camera coordinate frame X_{i+1} using world coordinate frame X_w

$$X_{i+1} = R_{i+1} X_w + t_{i+1}$$

Substituting the world coordinate expression in the above equation:

$$X_{i+1} = R_{i+1} R_i^T (X_i - t_i) + t_{i+1}$$

$$X_{i+1} = (R_{i+1} R_i^T) X_i + (t_{i+1} - R_{i+1} R_i^T t_i)$$

From above equation,

$$R_{rel} = R_{i+1} R_i^T$$

$$t_{rel} = t_{i+1} - R_{rel} t_i$$

We know that,

$$E = \hat{T} R$$

and

$$F = K_2^{-1} \hat{T} R K_1$$

Thus for relative motion we can write this as:

$$E = \hat{T_{rel}} R_{rel}$$

Since the camera is same

$$F = K^{-1}\hat{T_{rel}}R_{rel}K$$

===== end of your answer for 1.2 =====

Coding

Initialization

Run the following code, which imports the modules you'll need and defines helper functions you may need to use later in your implementations.

In [39]:
```
import os
import numpy as np
import scipy
import scipy.optimize
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from google.colab.patches import cv2_imshow
import cv2


connections_3d = [[0,1], [1,3], [2,3], [2,0], [4,5], [6,7], [8,9], [9,11], [10,11],
                  [1,5], [5,9], [2,6], [6,10], [3,7], [7,11]]
color_links = [(255,0,0),(255,0,0),(255,0,0),(255,0,0),(0,0,255),(255,0,255),(0,255
colors = ['blue','blue','blue','blue','red','magenta','green','green','green','gree


def visualize_keypoints(image, pts, Threshold=100):
    '''
    This function visualizes the 2d keypoint pairs in connections_3d
    (as define above) whose match score lies above a given Threshold
    in an OpenCV GUI frame, against an image background.

    :param image: image as a numpy array, of shape (height, width, 3) where 3 is th
    :param pts: np.array of shape (num_points, 3)
    '''
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    for i in range(12):
        cx, cy = pts[i][0:2]
        if pts[i][2]>Threshold:
            cv2.circle(image,(int(cx),int(cy)),5,(0,255,255),5)

    for i in range(len(connections_3d)):
        idx0, idx1 = connections_3d[i]
        if pts[idx0][2]>Threshold and pts[idx1][2]>Threshold:
            x0, y0 = pts[idx0][0:2]
            x1, y1 = pts[idx1][0:2]
```

```
            cv2.line(image, (int(x0), int(y0)), (int(x1), int(y1)), color_links[i],

    cv2_imshow(image)

    return image


def plot_3d_keypoint(pts_3d):
    '''
    this function visualizes 3d keypoints on a matplotlib 3d axes

    :param pts_3d: np.array of shape (num_points, 3)
    '''
    fig = plt.figure()
    num_points = pts_3d.shape[0]
    ax = fig.add_subplot(111, projection='3d')
    for j in range(len(connections_3d)):
        index0, index1 = connections_3d[j]
        xline = [pts_3d[index0,0], pts_3d[index1,0]]
        yline = [pts_3d[index0,1], pts_3d[index1,1]]
        zline = [pts_3d[index0,2], pts_3d[index1,2]]
        ax.plot(xline, yline, zline, color=colors[j])
    np.set_printoptions(threshold=1e6, suppress=True)
    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')
    plt.show()


def calc_epi_error(pts1_homo, pts2_homo, F):
    '''
    Helper function to calcualte the sum of squared distance between the
    corresponding points and the estimated epipolar lines.

    pts1_homo \dot F.T \dot pts2_homo = 0

    :param pts1_homo: of shape (num_points, 3); in homogeneous coordinates, not nor
    :param pts2_homo: same specification as to pts1_homo.
    :param F: Fundamental matrix
    '''

    line1s = pts1_homo.dot(F.T)
    dist1 = np.square(np.divide(np.sum(np.multiply(
        line1s, pts2_homo), axis=1), np.linalg.norm(line1s[:, :2], axis=1)))

    line2s = pts2_homo.dot(F)
    dist2 = np.square(np.divide(np.sum(np.multiply(
        line2s, pts1_homo), axis=1), np.linalg.norm(line2s[:, :2], axis=1)))

    ress = (dist1 + dist2).flatten()
    return ress


def toHomogenous(pts):
    """
    Adds a stack of ones at the end, to turn a set of points into a set of
```

```
    homogeneous points.

    :params pts: in shape (num_points, 2).
    """
    return np.vstack([pts[:,0],pts[:,1],np.ones(pts.shape[0])]).T.copy()


def _epipoles(E):
    """
    gets the epipoles from the Essential Matrix.

    :params E: Essential matrix.
    """
    U, S, V = np.linalg.svd(E)
    e1 = V[-1, :]
    U, S, V = np.linalg.svd(E.T)
    e2 = V[-1, :]
    return e1, e2


def displayEpipolarF(I1, I2, F, points):
    """
    GUI interface you may use to help you verify your calculated fundamental
    matrix F. Select a point I1 in one view, and it should correctly correspond
    to the displayed point in the second view.
    """
    e1, e2 = _epipoles(F)

    sy, sx, _ = I2.shape

    f, [ax1, ax2] = plt.subplots(1, 2, figsize=(12, 9))
    ax1.imshow(I1)
    ax1.set_title('The point you selected:')
    ax2.imshow(I2)
    ax2.set_title('Verify that the corresponding point \n is on the epipolar line i

    plt.sca(ax1)

    colors = ['r','g','b','y','m','k']
    for i, out in enumerate(points):
      x, y = out #[0]

      xc = x
      yc = y
      v = np.array([xc, yc, 1])
      l = F.dot(v)
      s = np.sqrt(l[0]**2+l[1]**2)

      if s==0:
          print('Zero line vector in displayEpipolar')

      l = l/s

      if l[0] != 0:
          ye = sy-1
          ys = 0
```

```
            xe = -(l[1] * ye + l[2])/l[0]
            xs = -(l[1] * ys + l[2])/l[0]
        else:
            xe = sx-1
            xs = 0
            ye = -(l[0] * xe + l[2])/l[1]
            ys = -(l[0] * xs + l[2])/l[1]

        # plt.plot(x,y, '*', 'MarkerSize', 6, 'LineWidth', 2);
        ax1.plot(x, y, '*', markersize=6, linewidth=2, color=colors[i%len(colors)])
        ax2.plot([xs, xe], [ys, ye], linewidth=2, color=colors[i%len(colors)])
    plt.draw()


def _singularize(F):
    U, S, V = np.linalg.svd(F)
    S[-1] = 0
    F = U.dot(np.diag(S).dot(V))
    return F

def _objective_F(f, pts1, pts2):
    F = _singularize(f.reshape([3, 3]))
    num_points = pts1.shape[0]
    hpts1 = np.concatenate([pts1, np.ones([num_points, 1])], axis=1)
    hpts2 = np.concatenate([pts2, np.ones([num_points, 1])], axis=1)
    Fp1 = F.dot(hpts1.T)
    FTp2 = F.T.dot(hpts2.T)

    r = 0
    for fp1, fp2, hp2 in zip(Fp1.T, FTp2.T, hpts2):
        r += (hp2.dot(fp1))**2 * (1/(fp1[0]**2 + fp1[1]**2) + 1/(fp2[0]**2 + fp2[1]
    return r

def refineF(F, pts1, pts2):
    f = scipy.optimize.fmin_powell(
        lambda x: _objective_F(x, pts1, pts2), F.reshape([-1]),
        maxiter=100000,
        maxfun=10000,
        disp=False
    )
    return _singularize(f.reshape([3, 3]))


# Used in 4.2 Epipolar Correspondence
def epipolarMatchGUI(I1, I2, F, points, epipolarCorrespondence):
    e1, e2 = _epipoles(F)

    sy, sx, _ = I2.shape

    f, [ax1, ax2] = plt.subplots(1, 2, figsize=(12, 9))
    ax1.imshow(I1)
    ax1.set_title('The point you selected:')
    ax2.imshow(I2)
    ax2.set_title('Verify that the corresponding point \n is on the epipolar line i

    plt.sca(ax1)
```

```
    colors = ['r','g','b','y','m','k']

    for i, out in enumerate(points):
      x, y = out

      xc = int(x)
      yc = int(y)
      v = np.array([xc, yc, 1])
      l = F.dot(v)
      s = np.sqrt(l[0]**2+l[1]**2)

      if s==0:
          print('Zero line vector in displayEpipolar')

      l = l/s

      if l[0] != 0:
          ye = sy-1
          ys = 0
          xe = -(l[1] * ye + l[2])/l[0]
          xs = -(l[1] * ys + l[2])/l[0]
      else:
          xe = sx-1
          xs = 0
          ye = -(l[0] * xe + l[2])/l[1]
          ys = -(l[0] * xs + l[2])/l[1]

      ax1.plot(x, y, '*', markersize=6, linewidth=2, color=colors[i%len(colors)])
      ax2.plot([xs, xe], [ys, ye], linewidth=2, color=colors[i%len(colors)])

      # draw points
      x2, y2 = epipolarCorrespondence(I1, I2, F, xc, yc)
      ax2.plot(x2, y2, 'ro', markersize=8, linewidth=2)
      plt.draw()
```

```
<>:71: SyntaxWarning: invalid escape sequence '\d'
<>:71: SyntaxWarning: invalid escape sequence '\d'
/tmp/ipython-input-1707180375.py:71: SyntaxWarning: invalid escape sequence '\d'
  pts1_homo \dot F.T \dot pts2_homo = 0
```

Set up data

In this section, we will download the test case image views, camera intrinsics, and point correnspondences, which you will use for testing your implementations.

In [40]:
```
if not os.path.exists('data'):
  !wget https://www.andrew.cmu.edu/user/eweng/data.zip -O data.zip
  !unzip -qq "data.zip"
  print("downloaded and unzipped data")
```

```
--2025-10-30 01:42:22--  https://www.andrew.cmu.edu/user/eweng/data.zip
Resolving www.andrew.cmu.edu (www.andrew.cmu.edu)... 128.2.42.53
Connecting to www.andrew.cmu.edu (www.andrew.cmu.edu)|128.2.42.53|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21314971 (20M) [application/zip]
Saving to: ‘data.zip’

data.zip            100%[===================>]  20.33M   580KB/s    in 38s

2025-10-30 01:43:01 (549 KB/s) - ‘data.zip’ saved [21314971/21314971]

downloaded and unzipped data
```

Problem 2: Estimating the Fundamental Matrix with the Eight-point Algorithm

In this part, implement the 8-point algorithm you learned in class, which estimates the fundamental matrix from corresponding points in two images.

In [41]:
```python
def eightpoint(pts1, pts2, M):
    '''
    Q2.1: Eight Point Algorithm
    Input:  pts1, Nx2 Matrix
            pts2, Nx2 Matrix
            M, a scalar parameter computed as max(imwidth, imheight)
    Output: F, the fundamental matrix

    HINTS:
    (1) Normalize the input pts1 and pts2 using the matrix T.
    (2) Setup the eight point algorithm's equation.
    (3) Solve for the least square solution using SVD.
    (4) Use the function `_singularize` (provided in the helper functions above) to
    (5) Use the function `refineF` (provided in the helper functions above) to refi
        (Remember to use the normalized points instead of the original points)
    (6) Unscale the fundamental matrix by the lower right corner element
    '''



    F = None
    N = pts1.shape[0]
    # print(N)

    # ===== your code here! =====
    T = np.array([[1.0/M, 0,     0],
                  [0,     1.0/M, 0],
                  [0,     0,     1]])

    ones = np.ones((N,1))
    # print(pts1.shape)
    normalised_pts1 = pts1 / M
    normalised_pts2 = pts2 / M
    # print(normalised_pts2.shape)
```

```
    x = normalised_pts1[:, 0]
    x_dash = normalised_pts2[:, 0]

    y = normalised_pts1[:, 1]
    y_dash = normalised_pts2[:, 1]

    A = np.column_stack([x_dash * x, x_dash * y, x_dash,
                         y_dash * x, y_dash * y, y_dash,
                         x, y, np.ones(N)])
    # print(A.shape)

    U, S, Vt = np.linalg.svd(A)
    F = Vt[-1, :]
    F = F.reshape(3,3)

    F_normalized = _singularize(F)
    F_normalized = refineF(F_normalized, normalised_pts1[:, :2], normalised_pts2[:,
    F = T.T @ F_normalized @ T
    F = F / F[2, 2]

    # ==== end of code ====

    return F
```

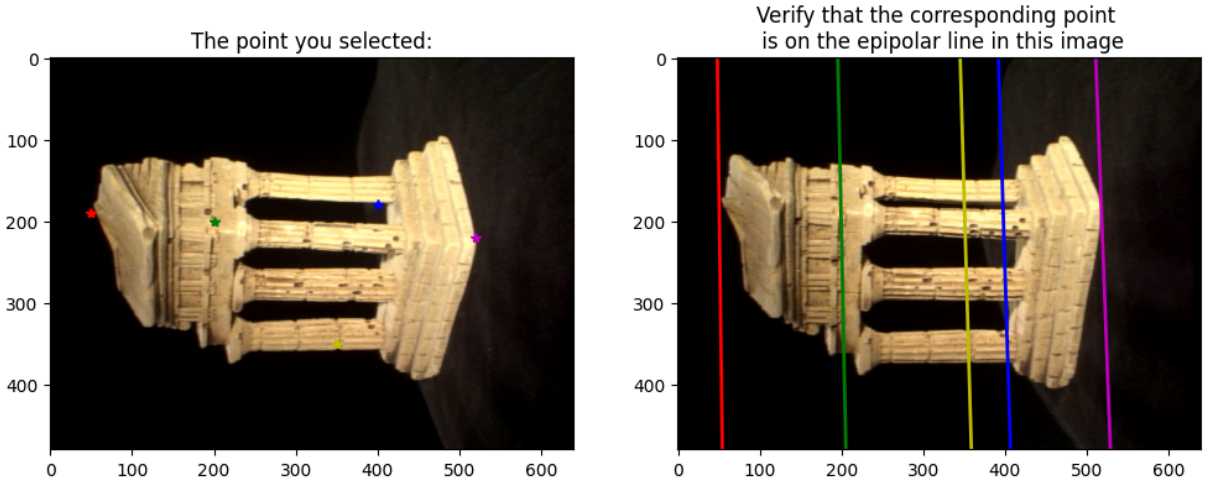
Run this code to test your implementation of the 8-point algorithm. Your code should pass all the assert statements at the end.

In [42]:
```
correspondence = np.load('data/some_corresp.npz') # Loading correspondences
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')

F = eightpoint(pts1, pts2, M=np.max([*im1.shape, *im2.shape]))
print(f'recovered F:\n{F.round(4)}')

# Simple Tests to verify your implementation:
pts1_homogenous, pts2_homogenous = toHomogenous(pts1), toHomogenous(pts2)

assert F.shape == (3, 3), "F is wrong shape"
assert F[2, 2] == 1, "F_33 != 1"
assert np.linalg.matrix_rank(F) == 2, "F should have rank 2"
assert np.mean(calc_epi_error(pts1_homogenous, pts2_homogenous, F)) < 1, "F error i
```

```
recovered F:
[[-0.      0.     -0.2519]
 [ 0.     -0.      0.0026]
 [ 0.2422 -0.0068  1.    ]]
```

The following tool may help you debug. You may specify a point in im1, and view the corresponding epipolar line in im2 based on the F you found. In your submission, make sure you include the debug picture below, with at least five epipolar point-line correspondences taht show that your calculation of F is correct.

In [43]:
```
# the points in im1, whose correnponding epipolar line in im2 you'd like to verify
point = [(50,190),(200, 200), (400,180), (350,350), (520, 220)]
# feel free to change these point, to verify different point correspondences
displayEpipolarF(im1, im2, F, point)
```



Problem 3: Metric Reconstruction

3.1 Essential Matrix

In [44]:
```
def essentialMatrix(F, K1, K2):
    '''
    Q3.1: Compute the essential matrix E.
        Input:  F, fundamental matrix
                K1, internal camera calibration matrix of camera 1
                K2, internal camera calibration matrix of camera 2
        Output: E, the essential matrix
    '''


    # ----- TODO -----
    ### BEGIN SOLUTION
    # print(K2.T.shape, F.shape, K1.shape)
    E = K2.T @ F @ K1
    E = E / E[2,2]

    ### END SOLUTION
    return E
```

Run the following code to check your implementation.

In [45]:
```
correspondence = np.load('data/some_corresp.npz') # Loading correspondences
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')
```

```
F = eightpoint(pts1, pts2, M=np.max([*im1.shape, *im2.shape]))
E = essentialMatrix(F, K1, K2)
print(f'recovered E:\n{E.round(4)}')

# Simple Tests to verify your implementation:
assert(E[2, 2] == 1)
assert(np.linalg.matrix_rank(E) == 2)
```

```
recovered E:
[[   -3.3716   456.6158 -2473.8947]
 [  197.6042   -10.2903    64.3966]
 [ 2480.7427    19.8564     1.    ]]
```

3.2 Triangulation

In [56]:

```
def triangulate(C1, pts1, C2, pts2):
  '''
  Q3.2: Triangulate a set of 2D coordinates in the image to a set of 3D points.
  Input:  C1, the 3x4 camera matrix
          pts1, the Nx2 matrix with the 2D image coordinates per row
          C2, the 3x4 camera matrix
          pts2, the Nx2 matrix with the 2D image coordinates per row
  Output: P, the Nx3 matrix with the corresponding 3D points per row
          err, the reprojection error.

  Hints:
  (1) For every input point, form A using the corresponding points from pts1 & pts2
  (2) Solve for the least square solution using np.linalg.svd
  (3) Calculate the reprojection error using the calculated 3D points and C1 & C2 (
      homogeneous coordinates to non-homogeneous ones)
  (4) Keep track of the 3D points and projection error, and continue to next point
  (5) You do not need to follow the exact procedure above.
  '''

  # ----- TODO -----
  ### BEGIN SOLUTION
  N = pts1.shape[0]
  P = np.zeros((N, 3))
  err = 0

  for i in range(N):

    x1 = pts1[i][0]
    y1 = pts1[i][1]

    x2 = pts2[i][0]
    y2 = pts2[i][1]

    A = np.array([
                x1 * C1[2] - C1[0],
                y1 * C1[2] - C1[1],
                x2 * C2[2] - C2[0],
                y2 * C2[2] - C2[1]
            ])
```

```
    U, S, Vt = np.linalg.svd(A)
    w = Vt[-1]
    # print(w.shape)
    w = w/w[3]
    # print(w)
    P[i] = w[:3]

    reprojected_pts1 = C1 @ w
    reprojected_pts1 = reprojected_pts1[:2] / reprojected_pts1[2]

    reprojected_pts2 = C2 @ w
    reprojected_pts2 = reprojected_pts2[2] / reprojected_pts2[2]
    err += np.sum((pts1[i] - reprojected_pts1)**2) + np.sum((pts2[i] - reprojected_



  ### END SOLUTION

  return P, err
```

3.3 Find M2

In [57]:

```
def camera2(E):
  """helper function to find the 4 possibile M2 matrices"""
  U,S,V = np.linalg.svd(E)
  m = S[:2].mean()
  E = U.dot(np.array([[m,0,0], [0,m,0], [0,0,0]])).dot(V)
  U,S,V = np.linalg.svd(E)
  W = np.array([[0,-1,0], [1,0,0], [0,0,1]])

  if np.linalg.det(U.dot(W).dot(V))<0:
      W = -W

  M2s = np.zeros([3,4,4])
  M2s[:,:,0] = np.concatenate([U.dot(W).dot(V), U[:,2].reshape([-1, 1])/abs(U[:,2])
  M2s[:,:,1] = np.concatenate([U.dot(W).dot(V), -U[:,2].reshape([-1, 1])/abs(U[:,2]
  M2s[:,:,2] = np.concatenate([U.dot(W.T).dot(V), U[:,2].reshape([-1, 1])/abs(U[:,2
  M2s[:,:,3] = np.concatenate([U.dot(W.T).dot(V), -U[:,2].reshape([-1, 1])/abs(U[:,
  return M2s


def findM2(F, pts1, pts2, intrinsics):
    '''
    Q3.3: Function to find camera2's projective matrix given correspondences
        Input:  F, the pre-computed fundamental matrix
                pts1, the Nx2 matrix with the 2D image coordinates per row
                pts2, the Nx2 matrix with the 2D image coordinates per row
                intrinsics, the intrinsics of the cameras, load from the .npz file
                filename, the filename to store results
        Output: [M2, C2, P] the computed M2 (3x4) camera projective matrix, C2 (3x4

    ***
    Hints:
```

```
    (1) Loop through the 'M2s' and use triangulate to calculate the 3D points and p
        of the projection error through best_error and retain the best one.
    (2) Remember to take a look at camera2 to see how to correctly reterive the M2

    '''

    K1, K2 = intrinsics['K1'], intrinsics['K2']

    # ----- TODO -----
    ### BEGIN SOLUTION
    E = K2.T @ F @ K1
    M2s = camera2(E)

    M1 = np.hstack([np.eye(3), np.zeros((3, 1))])
    C1 = K1 @ M1

    best_error = np.inf
    best_M2 = None
    best_C2 = None
    best_P = None

    for i in range(4):
        M2_candidate = M2s[:, :, i]
        C2_candidate = K2 @ M2_candidate
        P_candidate, err = triangulate(C1, pts1, C2_candidate, pts2)

        num_positive_cam1 = np.sum(P_candidate[:, 2] > 0)

        R2 = M2_candidate[:, :3]
        t2 = M2_candidate[:, 3]
        P_cam2 = (R2 @ P_candidate.T).T + t2
        num_positive_cam2 = np.sum(P_cam2[:, 2] > 0)

        if num_positive_cam1 == pts1.shape[0] and num_positive_cam2 == pts1.shape[0
            if err < best_error:
                best_error = err
                best_M2 = M2_candidate
                best_C2 = C2_candidate
                best_P = P_candidate
    M2 = best_M2
    C2 = best_C2
    P = best_P

    ### END SOLUTION

    return M2, C2, P
```

Run the following code to check your implementation of triangulation and findM2.

In [58]:
```
correspondence = np.load('data/some_corresp.npz') # Loading correspondences
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')
```

```
F = eightpoint(pts1, pts2, M=np.max([*im1.shape, *im2.shape]))

M2, C2, P = findM2(F, pts1, pts2, intrinsics)

# Simple Tests to verify your implementation:
M1 = np.hstack((np.identity(3), np.zeros(3)[:,np.newaxis]))
C1 = K1.dot(M1)
C2 = K2.dot(M2)
P_test, err = triangulate(C1, pts1, C2, pts2)
print(err)
assert(err < 500)
```

```
351.8979675350623
```

Problem 4: 3D Visualization

In [59]:

```
def epipolarCorrespondence(im1, im2, F, x1, y1):
    '''
    Q4.1: 3D visualization of the temple images.
    Input:  im1, the first image
            im2, the second image
            F, the fundamental matrix
            x1, x-coordinates of a pixel on im1
            y1, y-coordinates of a pixel on im1
    Output: x2, x-coordinates of the pixel on im2
            y2, y-coordinates of the pixel on im2

    Hints:
    (1) Given input [x1, x2], use the fundamental matrix to recover the correspondi
    (2) Search along this line to check nearby pixel intensity (you can define a se
        find the best matches
    (3) Use gaussian weighting to weight the pixel simlairty

    '''
    # ----- TODO -----
    # YOUR CODE HERE

    h1, w1 = im1.shape[:2]
    h2, w2 = im2.shape[:2]

    point1 = np.array([x1, y1, 1.0])
    # print(point1.shape)
    # print(F.shape)
    epipolar_line = F @ point1.T
    norm = np.linalg.norm(epipolar_line[:2])
    epipolar_line = epipolar_line / norm

    window_size = 5
    search_range = 50
    half_window = window_size // 2

    patch_image1 = im1[y1 - half_window:y1 + half_window + 1, x1 - half_window:x1 +
    patch_image1 = scipy.ndimage.gaussian_filter(patch_image1, sigma = 1)
```

```
    y_start = max(0, y1 - search_range)
    y_end = min(h2, y1 + search_range)

    min_error = float('inf')
    best_x2 = None
    best_y2 = None

    for y2 in range(y_start, y_end + 1):
        x2 = -(epipolar_line[1] * y2 + epipolar_line[2]) / epipolar_line[0]
        x2 = int(round(x2))

        if x2 - half_window < 0 or x2 + half_window >= w2 or y2 - half_window < 0 o
            continue

        patch_image2 = im2[y2 - half_window:y2 + half_window + 1, x2 - half_window:
        patch_image2 = scipy.ndimage.gaussian_filter(patch_image2, sigma = 1)

        error = np.sum((patch_image1 - patch_image2) ** 2)
        if error < min_error:
            min_error = error
            best_x2 = x2
            best_y2 = y2


    x2, y2 = best_x2, best_y2

    # END YOUR CODE
    return x2, y2
```

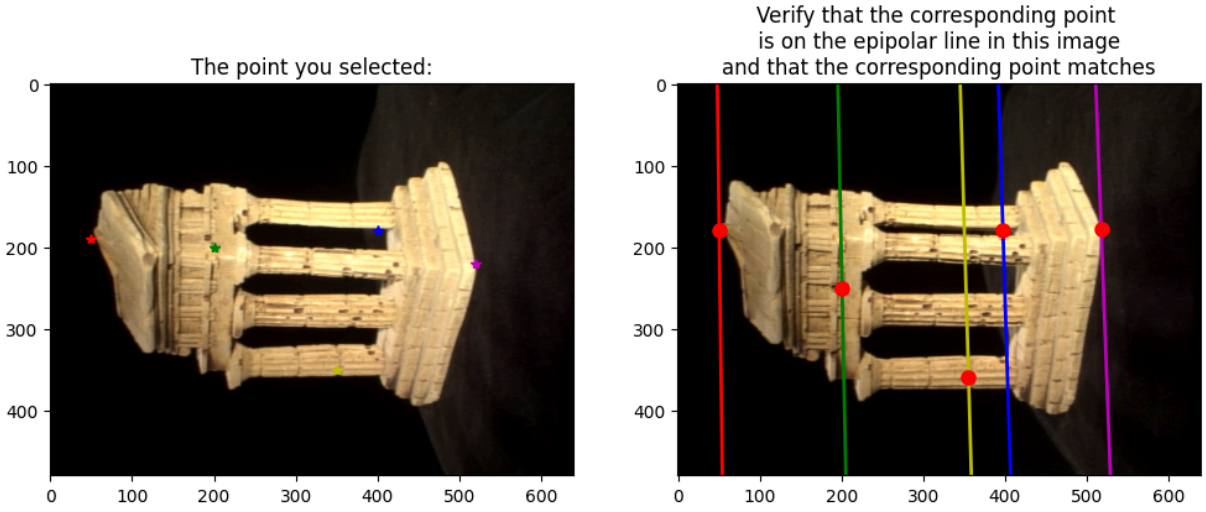
Run the following code to check your implementation.

In [60]:
```
correspondence = np.load('data/some_corresp.npz') # Loading correspondences
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')

F = eightpoint(pts1, pts2, M=np.max([*im1.shape, *im2.shape]))
# print(F)

# Simple Tests to verify your implementation:
x2, y2 = epipolarCorrespondence(im1, im2, F, 119, 217)
print(x2, y2)
assert(np.linalg.norm(np.array([x2, y2]) - np.array([118, 181])) < 10)
```

```
118 182
```

Use the below tool to debug your code.

In [61]:
```
# the points in im1 whose correnponding epipolar line in im2 you'd like to verify
points = [(50,190), (200, 200), (400,180), (350,350), (520, 220)]
# feel free to change these points to verify different point correspondences
epipolarMatchGUI(im1, im2, F, points, epipolarCorrespondence)
```



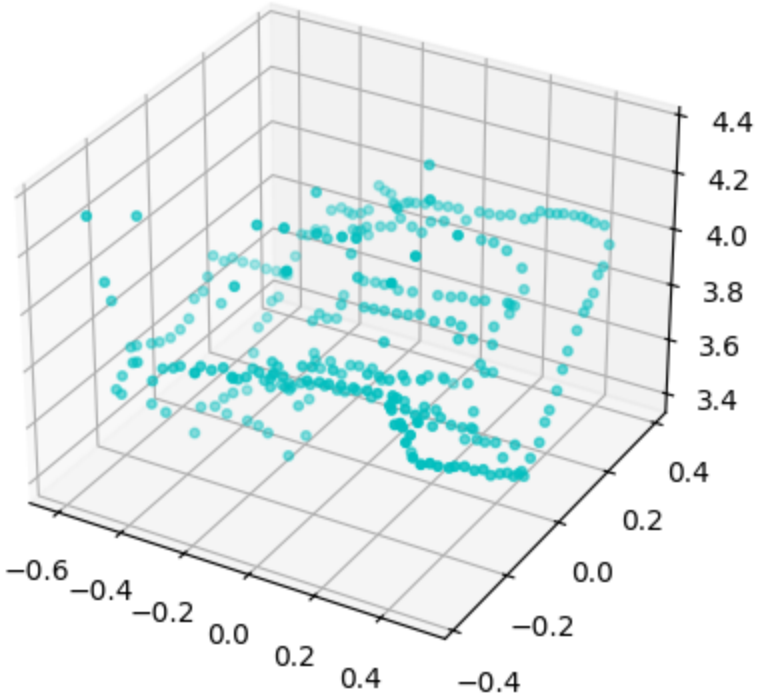
4.2 Temple Visualization

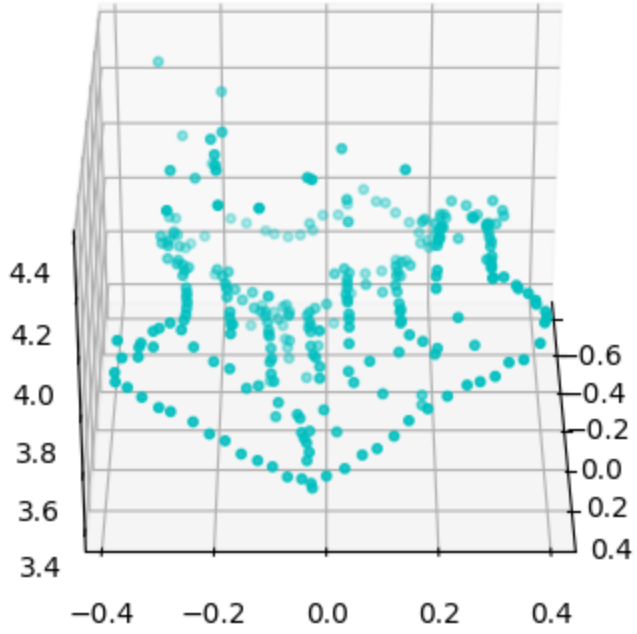
In [62]:
```
def compute3D_pts(temple_pts1, intrinsics, F, im1, im2):
    '''
    Q4.2: Finding the 3D position of given points based on epipolar correspondence
    Input:  temple_pts1, chosen points from im1
            intrinsics, the intrinsics dictionary for calling epipolarCorrespondenc
            F, the fundamental matrix
            im1, the first image
            im2, the second image
    Output: P (Nx3) the recovered 3D points

    Hints:
    (1) Use epipolarCorrespondence to find the corresponding point for [x1 y1] (fin
    (2) Now you have a set of corresponding points [x1, y1] and [x2, y2], you can c
        matrix and use triangulate to find the 3D points.
    (3) Use the function findM2 to find the 3D points P (do not recalculate fundame
    (4) As a reference, our solution's best error is around ~2200 on the 3D points.
    '''
    # ----- TODO -----
    # YOUR CODE HERE

    correspondences = []
    for i in range(len(temple_pts1)):
        x1, y1 = temple_pts1[i][0], temple_pts1[i][1]
        x2, y2 = epipolarCorrespondence(im1, im2, F, x1, y1)
        correspondences.append([x2, y2])
    correspondences = np.array(correspondences)

    M2, C2, P = findM2(F, temple_pts1, correspondences, intrinsics)


    return P
    # END YOUR CODE
```

Below, integrate everything together. The provided starter code loads in the temple data found at `data/templeCoords.npz`, which contains 288 hand-selected points from im1

saved in the variables x1 and y1. Then, get the 3d points from the 2d point point correspondences by calling the function you just implemented, as well as other necessary function. Finally, visualize the 3D reconstruction using matplotlib or plotly 3d scatter plot.

In [63]:
```
temple_coords = np.load('data/templeCoords.npz') # Loading temple coordinates
correspondence = np.load('data/some_corresp.npz') # Loading correspondences
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')

# ----- TODO -----
# Call eightpoint to get the F matrix
# Call compute3D_pts to get the 3D points and visualize using matplotlib scatter
# hint: you can change the viewpoint of a matplotlib 3d axes using
# `ax.view_init(azim, elev)` where azim is the rotation around the vertical z
# axis, and elev is the angle of elevation from the x-y plane

temple_pts1 = np.hstack([temple_coords['x1'], temple_coords['y1']])
# print(temple_pts1.shape)

# YOUR CODE HERE
F = eightpoint(pts1, pts2, M = np.max([*im1.shape, *im2.shape]))
P = compute3D_pts(temple_pts1, intrinsics, F, im1, im2)

# END YOUR CODE

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(P[:, 0], P[:, 1], P[:, 2], s=10, c='c', depthshade=True)
plt.draw()

# also show a different viewpoint
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(P[:, 0], P[:, 1], P[:, 2], s=10, c='c', depthshade=True)
ax.view_init(30, 0)
plt.draw()
```





Problem 5: Bundle Adjustment

Below is the implementation of RANSAC for Fundamental Matrix Recovery.

In [64]:
```
def ransacF(pts1, pts2, M, nIters=100, tol=10):
    '''
```

```
    Input:  pts1, Nx2 Matrix
            pts2, Nx2 Matrix
            M, a scaler parameter
            nIters, Number of iterations of the Ransac
            tol, tolerence for inliers
    Output: F, the fundamental matrix
            inliers, Nx1 bool vector set to true for inliers

    '''
    N = pts1.shape[0]
    pts1_homo, pts2_homo = toHomogenous(pts1), toHomogenous(pts2)
    best_inlier = 0
    inlier_curr = None

    for i in range(nIters):
        choice = np.random.choice(range(pts1.shape[0]), 8)
        pts1_choice = pts1[choice, :]
        pts2_choice = pts2[choice, :]
        F = eightpoint(pts1_choice, pts2_choice, M)
        ress = calc_epi_error(pts1_homo, pts2_homo, F)
        curr_num_inliner = np.sum(ress < tol)
        if curr_num_inliner > best_inlier:
            F_curr = F
            inlier_curr = (ress < tol)
            best_inlier = curr_num_inliner
    inlier_curr = inlier_curr.reshape(inlier_curr.shape[0], 1)
    indixing_array = inlier_curr.flatten()
    pts1_inlier = pts1[indixing_array]
    pts2_inlier = pts2[indixing_array]
    F = eightpoint(pts1_inlier, pts2_inlier, M)
    return F, inlier_curr
```

Below is the implementation of Rodrigues and Inverse Rodrigues Formulas. See the pdf for the detailed explanation of the functions.

In [65]:
```
def rodrigues(r):
    '''
        Input:  r, a 3x1 vector
        Output: R, a rotation matrix
    '''

    r = np.array(r).flatten()
    I = np.eye(3)
    theta = np.linalg.norm(r)
    if theta == 0:
        return I
    else:
        U = (r/theta)[:, np.newaxis]
        Ux, Uy, Uz = r/theta
        K = np.array([[0, -Uz, Uy], [Uz, 0, -Ux], [-Uy, Ux, 0]])
        R = I * np.cos(theta) + np.sin(theta) * K + \
            (1 - np.cos(theta)) * np.matmul(U, U.T)
    return R
```

```
def invRodrigues(R):
    '''
    Input:  R, a rotation matrix
    Output: r, a 3x1 vector
    '''

    def s_half(r):
        r1, r2, r3 = r
        if np.linalg.norm(r) == np.pi and (r1 == r2 and r1 == 0 and r2 == 0 and r3 <
            return -r
        else:
            return r

    A = (R - R.T)/2
    ro = [A[2, 1], A[0, 2], A[1, 0]]
    s = np.linalg.norm(ro)
    c = (np.sum(np.matrix(R).diagonal()) - 1)/2
    if s == 0 and c == 1:
        r = np.zeros(3)
    elif s == 0 and c == -1:
        col = np.eye(3) + R
        col_idx = np.nonzero(
            np.array(np.sum(col != 0, axis=0)).flatten())[0][0]
        v = col[:, col_idx]
        u = v/np.linalg.norm(v)
        r = s_half(u * np.pi)
    else:
        u = ro/s
        theta = np.arctan2(s, c)
        r = u * theta

    return r
```

Rodrigues Residual objective function

In [66]:

```
def rodriguesResidual(K1, M1, p1, K2, p2, x):
    '''
    Q5.1: Rodrigues residual.
        Input:  K1, the intrinsics of camera 1
                M1, the extrinsics of camera 1
                p1, the 2D coordinates of points in image 1
                K2, the intrinsics of camera 2
                p2, the 2D coordinates of points in image 2
                x, the flattened concatenationg of P, r2, and t2.
        Output: residuals, 4N x 1 vector, the difference between original and estim
    '''
    N = p1.shape[0]
    # ----- TODO -----
    ### BEGIN SOLUTION
    P = x[: 3 * N].reshape(N, 3)
    rotation_cam2 = x[3 * N : 3 * N + 3]
    translation_cam2 = x[3 * N + 3 :]

    R2 = rodrigues(rotation_cam2)
    M2 = np.hstack((R2, translation_cam2.reshape(3, 1)))
```

```
    C1 = K1 @ M1
    C2 = K2 @ M2

    p_homogenous = np.hstack([P, np.ones((N, 1))])
    p1_homogenous = (C1 @ p_homogenous.T).T
    p2_homogenous = (C2 @ p_homogenous.T).T

    p1_proj = p1_homogenous[:, :2] / p1_homogenous[:, 2:3]
    p2_proj = p2_homogenous[:, :2] / p2_homogenous[:, 2:3]

    residuals1 = (p1 - p1_proj).flatten()
    residuals2 = (p2 - p2_proj).flatten()

    residuals = np.concatenate([residuals1, residuals2])

    ### END SOLUTION
    return residuals
```

Bundle Adjustment

In [67]:

```
def bundleAdjustment(K1, M1, p1, K2, M2_init, p2, P_init):
    '''
    Q5.2 Bundle adjustment.
    Input:  K1, the intrinsics of camera 1
            M1, the extrinsics of camera 1
            p1, the 2D coordinates of points in image 1
            K2,  the intrinsics of camera 2
            M2_init, the initial extrinsics of camera 1
            p2, the 2D coordinates of points in image 2
            P_init, the initial 3D coordinates of points
    Output: M2, the optimized extrinsics of camera 1
            P2, the optimized 3D coordinates of points
            o1, the starting objective function value with the initial input
            o2, the ending objective function value after bundle adjustment

    Hints:
    (1) Use the scipy.optimize.minimize function to minimize the objective function
        You can try different (method='..') in scipy.optimize.minimize for best res
    '''
    obj_start = obj_end = 0
    # ----- TODO -----
    ### BEGIN SOLUTION
    N = p1.shape[0]

    R2_init = M2_init[:3, :3]
    t2_init = M2_init[:3, 3]

    r2_init = invRodrigues(R2_init)

    x0 = np.hstack([P_init.flatten(), r2_init.flatten(), t2_init.flatten()])

    def objective(x):
        residuals = rodriguesResidual(K1, M1, p1, K2, p2, x)
        return np.sum(residuals**2)
```

```
    obj_start = objective(x0)
    result = scipy.optimize.minimize(objective, x0, method='BFGS', options={'maxite

    x_opt = result.x
    P = x_opt[: 3 * N].reshape(N, 3)
    r2_opt = x_opt[3 * N : 3 * N + 3]
    t2_opt = x_opt[3 * N + 3 :]

    R2_opt = rodrigues(r2_opt)

    M2 = np.hstack([R2_opt, t2_opt.reshape(3, 1)])
    obj_end = result.fun


    ### END SOLUTION
    return M2, P, obj_start, obj_end
```

Put it all together

1. Call the ransacF function to find the fundamental matrix
2. Call the findM2 function to find the extrinsics of the second camera
3. Call the bundleAdjustment function to optimize the extrinsics and 3D points
4. Plot the 3D points before and after bundle adjustment using the plot_3D_dual function

On the given temple data, bundle adjustment can take up to 2 min to run.

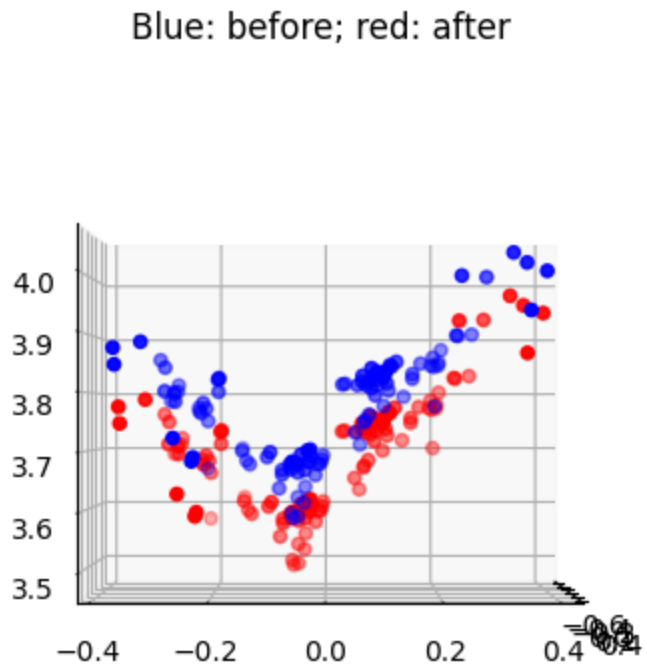
In [68]:
```
# Visualization:
np.random.seed(1)
correspondence = np.load('data/some_corresp_noisy.npz') # Loading noisy corresponde
intrinsics = np.load('data/intrinsics.npz') # Loading the intrinscis of the camera
K1, K2 = intrinsics['K1'], intrinsics['K2']
pts1, pts2 = correspondence['pts1'], correspondence['pts2']
im1 = plt.imread('data/im1.png')
im2 = plt.imread('data/im2.png')
M=np.max([*im1.shape, *im2.shape])

# YOUR CODE HERE
'''
Call the ransacF function to find the fundamental matrix
Call the findM2 function to find the extrinsics of the second camera
Call the bundleAdjustment function to optimize the extrinsics and 3D points
'''
F, inliers = ransacF(pts1, pts2, M)
pts1_inliers = pts1[inliers.flatten()]
pts2_inliers = pts2[inliers.flatten()]

M1 = np.hstack((np.identity(3), np.zeros((3,1))))
M2_init, C2, P_init = findM2(F, pts1_inliers, pts2_inliers, intrinsics)

M2, P_final, obj_start, obj_end = bundleAdjustment(K1, M1, pts1_inliers, K2, M2_ini

# END YOUR CODE
print(f"Before reprojection error: {obj_start}, After: {obj_end}")
```

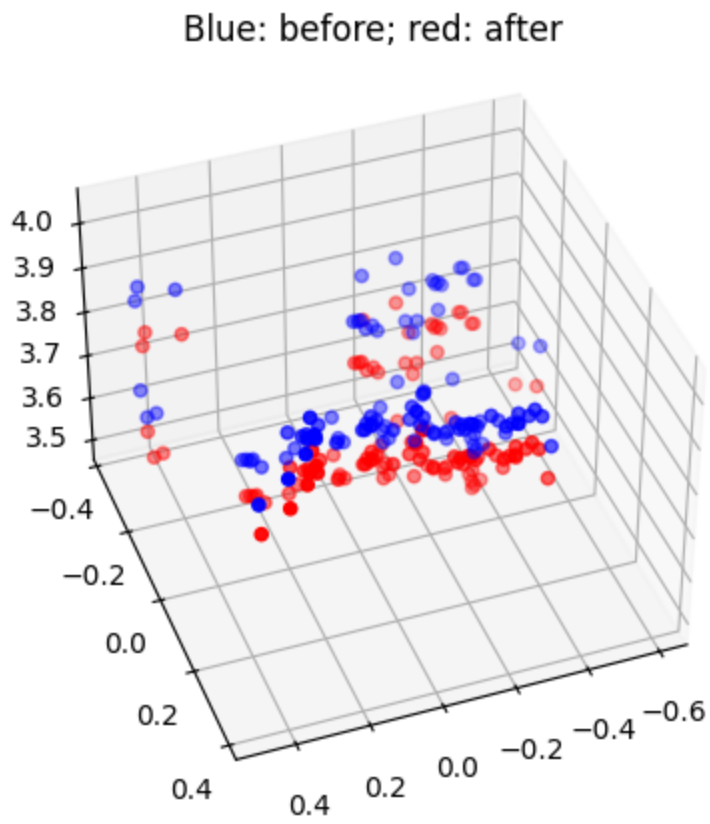
```
/tmp/ipython-input-3573617480.py:37: OptimizeWarning: Unknown solver options: ftol
  result = scipy.optimize.minimize(objective, x0, method='BFGS', options={'maxiter':
1000, 'ftol': 1e-9})
Before reprojection error: 352.8418819006491, After: 10.886760047115335
```

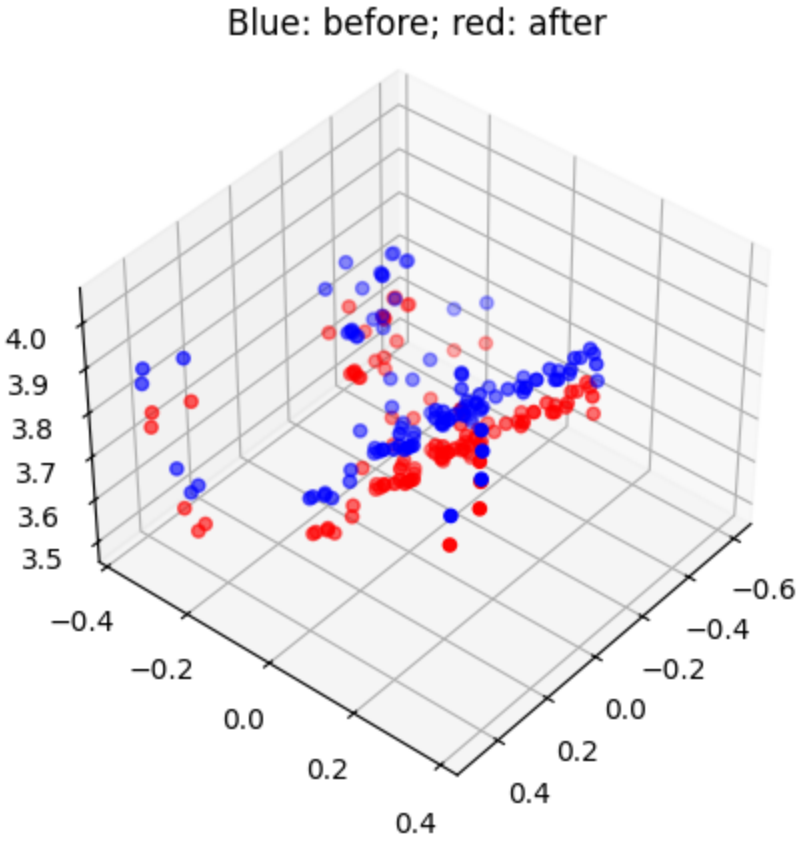
In [84]:
```
P_final.shape
```

Out[84]: (110, 3)

In [69]:
```
# helper function for visualization
def plot_3D_dual(P_before, P_after, azim=70, elev=45):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.set_title("Blue: before; red: after")
    ax.scatter(P_before[:,0], P_before[:,1], P_before[:,2], c = 'blue')
    ax.scatter(P_after[:,0], P_after[:,1], P_after[:,2], c='red')
    ax.view_init(azim=azim, elev=elev)
    plt.draw()

# plots the 3d points before and after BA from different viewpoints
plot_3D_dual(P_init, P_final, azim=0, elev=0)
plot_3D_dual(P_init, P_final, azim=70, elev=40)
plot_3D_dual(P_init, P_final, azim=40, elev=40)
```







(Extra Credit) Problem 6: Multiview Keypoint Reconstruction

6 Multi-View Reconstruction of keypoints

In [22]:
```
def MultiviewReconstruction(C1, pts1, C2, pts2, C3, pts3, Thres = 100):
    '''
    Q6.1 Multi-View Reconstruction of keypoints.
        Input:  C1, the 3x4 camera matrix
                pts1, the Nx3 matrix with the 2D image coordinates and confidence p
                C2, the 3x4 camera matrix
                pts2, the Nx3 matrix with the 2D image coordinates and confidence p
                C3, the 3x4 camera matrix
                pts3, the Nx3 matrix with the 2D image coordinates and confidence p
        Output: P, the Nx3 matrix with the corresponding 3D points for each keypoin
                err, the reprojection error.
    '''

    # Replace pass with your implementation
    # ------- TODO -------
    # YOUR CODE HERE
    def _project(C, X_h):
        u = C @ X_h
        return u[:2] / u[2]

    def _triangulate_linear(views):
        A = []
        for C, (x, y) in views:
            A.append(x * C[2, :] - C[0, :])
            A.append(y * C[2, :] - C[1, :])
        A = np.stack(A, axis=0)
        _, _, Vt = np.linalg.svd(A)
        X_h = Vt[-1]
        X_h = X_h / X_h[3]
        return X_h

    cams = [C1, C2, C3]
    dets = [pts1, pts2, pts3]

    N = pts1.shape[0]
    P = np.full((N, 3), np.nan, dtype=np.float64)
    per_point_err = np.full(N, np.nan, dtype=np.float64)

    for i in range(N):
        views = []
        for C, pts in zip(cams, dets):
            x, y, conf = pts[i]
            if conf >= Thres:
                views.append((C, (float(x), float(y))))

        if len(views) < 2:
            continue

        X_h = _triangulate_linear(views)
        P[i] = X_h[:3]
```

```
        errs = []
        for C, (x, y) in views:
            u_pred = _project(C, X_h)
            errs.append(np.linalg.norm(u_pred - np.array([x, y])))
        per_point_err[i] = float(np.mean(errs))

    valid = ~np.isnan(per_point_err)
    err = float(np.mean(per_point_err[valid])) if np.any(valid) else np.nan


    return P, err
    # END YOUR CODE
```

Plot Spatio-temporal (3D) keypoints

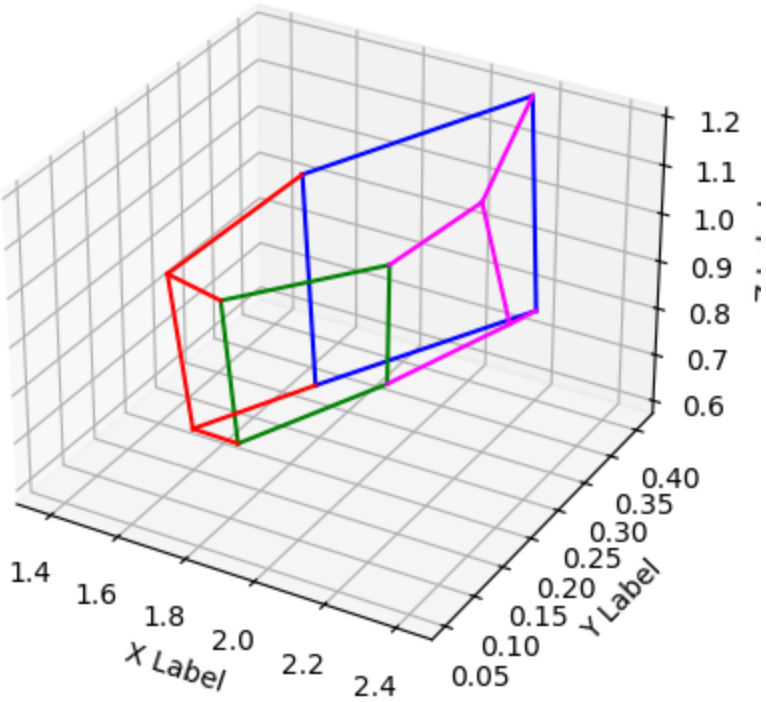
In [23]:
```
def plot_3d_keypoint_video(pts_3d_video):
  '''
  Plot Spatio-temporal (3D) keypoints
      :param car_points: np.array points * 3
  '''

  fig = plt.figure()
  ax = fig.add_subplot(111, projection='3d')
  for pts_3d in pts_3d_video:
      num_points = pts_3d.shape[1]
      for j in range(len(connections_3d)):
          index0, index1 = connections_3d[j]
          xline = [pts_3d[index0,0], pts_3d[index1,0]]
          yline = [pts_3d[index0,1], pts_3d[index1,1]]
          zline = [pts_3d[index0,2], pts_3d[index1,2]]
          ax.plot(xline, yline, zline, color=colors[j])
  np.set_printoptions(threshold=1e6, suppress=True)
  ax.set_xlabel('X Label')
  ax.set_ylabel('Y Label')
  ax.set_zlabel('Z Label')
  plt.show()
```

Put it all together for all 10 timesteps.

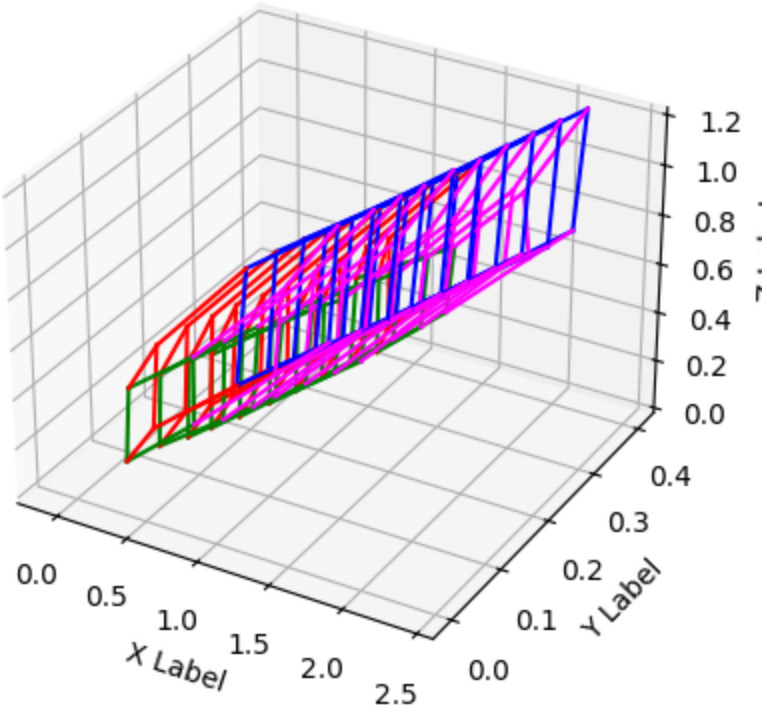
In [24]:
```
pts_3d_video = []
for loop in range(10):
    print(f"processing time frame - {loop}")

    data_path = os.path.join('data/q6/','time'+str(loop)+'.npz')
    image1_path = os.path.join('data/q6/','cam1_time'+str(loop)+'.jpg')
    image2_path = os.path.join('data/q6/','cam2_time'+str(loop)+'.jpg')
    image3_path = os.path.join('data/q6/','cam3_time'+str(loop)+'.jpg')

    im1 = plt.imread(image1_path)
    im2 = plt.imread(image2_path)
    im3 = plt.imread(image3_path)

    data = np.load(data_path)
    pts1 = data['pts1']
```

```
    pts2 = data['pts2']
    pts3 = data['pts3']

    K1 = data['K1']
    K2 = data['K2']
    K3 = data['K3']

    M1 = data['M1']
    M2 = data['M2']
    M3 = data['M3']

    if loop == 0 or loop==9:  # feel free to modify to visualize keypoints at other
        img = visualize_keypoints(im2, pts2)

    # YOUR CODE HERE
    C1 = K1 @ M1
    C2 = K2 @ M2
    C3 = K3 @ M3

    # Compute 3D reconstruction
    pts_3d, err = MultiviewReconstruction(C1, pts1, C2, pts2, C3, pts3, Thres=100)
    pts_3d_video.append(pts_3d)

    # END YOUR CODE

    if loop == 0:
        plot_3d_keypoint(pts_3d)

plot_3d_keypoint_video(pts_3d_video)
```

```
processing time frame - 0
```




```
processing time frame - 1
processing time frame - 2
processing time frame - 3
processing time frame - 4
processing time frame - 5
processing time frame - 6
processing time frame - 7
processing time frame - 8
processing time frame - 9
```



Problem 6.5: Camera Pose Estimation in Dynamic Scenes (Extra Credit)

6.2

In [25]:
```
!git clone --recursive https://github.com/naver/dust3r
```

```
Cloning into 'dust3r'...
remote: Enumerating objects: 611, done.
remote: Counting objects: 100% (445/445), done.
remote: Compressing objects: 100% (236/236), done.
remote: Total 611 (delta 328), reused 209 (delta 209), pack-reused 166 (from 3)
Receiving objects: 100% (611/611), 748.00 KiB | 27.70 MiB/s, done.
Resolving deltas: 100% (358/358), done.
Submodule 'croco' (https://github.com/naver/croco) registered for path 'croco'
Cloning into '/content/dust3r/croco'...
remote: Enumerating objects: 198, done.
remote: Counting objects: 100% (89/89), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 198 (delta 56), reused 34 (delta 34), pack-reused 109 (from 1)
Receiving objects: 100% (198/198), 403.77 KiB | 28.84 MiB/s, done.
Resolving deltas: 100% (95/95), done.
Submodule path 'croco': checked out 'd7de0705845239092414480bd829228723bf20de'
```

In [26]:
```
%cd dust3r
```

```
/content/dust3r
```

In [27]:
```
!git submodule update --init --recursive
```

In [28]:
```
!ls
```

```
assets                docker          NOTICE                     train.py
croco                 dust3r          README.md                  visloc.py
datasets_preprocess   dust3r_visloc   requirements_optional.txt
demo.py               LICENSE         requirements.txt
```

In [29]:
```
!pip install -r requirements.txt
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (fro
m -r requirements.txt (line 1)) (2.8.0+cu126)
Requirement already satisfied: torchvision in /usr/local/lib/python3.12/dist-package
s (from -r requirements.txt (line 2)) (0.23.0+cu126)
Collecting roma (from -r requirements.txt (line 3))
  Downloading roma-1.5.4-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: gradio in /usr/local/lib/python3.12/dist-packages (fr
om -r requirements.txt (line 4)) (5.49.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages
(from -r requirements.txt (line 5)) (3.10.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from
-r requirements.txt (line 6)) (4.67.1)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.12/dist-packa
ges (from -r requirements.txt (line 7)) (4.12.0.88)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (fro
m -r requirements.txt (line 8)) (1.16.2)
Requirement already satisfied: einops in /usr/local/lib/python3.12/dist-packages (fr
om -r requirements.txt (line 9)) (0.8.1)
Collecting trimesh (from -r requirements.txt (line 10))
  Downloading trimesh-4.9.0-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.12/dist-package
s (from -r requirements.txt (line 11)) (2.19.0)
Collecting pyglet<2 (from -r requirements.txt (line 12))
  Downloading pyglet-1.5.31-py3-none-any.whl.metadata (7.6 kB)
Requirement already satisfied: huggingface-hub>=0.22 in /usr/local/lib/python3.12/di
st-packages (from huggingface-hub[torch]>=0.22->-r requirements.txt (line 13)) (0.3
6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages
(from torch->-r requirements.txt (line 1)) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.1
2/dist-packages (from torch->-r requirements.txt (line 1)) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages
(from torch->-r requirements.txt (line 1)) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packa
ges (from torch->-r requirements.txt (line 1)) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages
(from torch->-r requirements.txt (line 1)) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (fr
om torch->-r requirements.txt (line 1)) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (fr
om torch->-r requirements.txt (line 1)) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/pyt
hon3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/p
ython3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/pyt
hon3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python
3.12/dist-packages (from torch->-r requirements.txt (line 1)) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python
3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python
3.12/dist-packages (from torch->-r requirements.txt (line 1)) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/pytho
n3.12/dist-packages (from torch->-r requirements.txt (line 1)) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/pyth
```

```
on3.12/dist-packages (from torch->-r requirements.txt (line 1)) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/pyth
on3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/pytho
n3.12/dist-packages (from torch->-r requirements.txt (line 1)) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.1
2/dist-packages (from torch->-r requirements.txt (line 1)) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.1
2/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/pyth
on3.12/dist-packages (from torch->-r requirements.txt (line 1)) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python
3.12/dist-packages (from torch->-r requirements.txt (line 1)) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packa
ges (from torch->-r requirements.txt (line 1)) (3.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (fro
m torchvision->-r requirements.txt (line 2)) (2.0.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.12/di
st-packages (from torchvision->-r requirements.txt (line 2)) (11.3.0)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.12/dis
t-packages (from gradio->-r requirements.txt (line 4)) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.12/dist-pac
kages (from gradio->-r requirements.txt (line 4)) (4.11.0)
Requirement already satisfied: brotli>=1.1.0 in /usr/local/lib/python3.12/dist-packa
ges (from gradio->-r requirements.txt (line 4)) (1.1.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in /usr/local/lib/python3.12/di
st-packages (from gradio->-r requirements.txt (line 4)) (0.120.0)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.12/dist-packages (fro
m gradio->-r requirements.txt (line 4)) (0.6.4)
Requirement already satisfied: gradio-client==1.13.3 in /usr/local/lib/python3.12/di
st-packages (from gradio->-r requirements.txt (line 4)) (1.13.3)
Requirement already satisfied: groovy~=0.1 in /usr/local/lib/python3.12/dist-package
s (from gradio->-r requirements.txt (line 4)) (0.1.2)
Requirement already satisfied: httpx<1.0,>=0.24.1 in /usr/local/lib/python3.12/dist-
packages (from gradio->-r requirements.txt (line 4)) (0.28.1)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.12/dis
t-packages (from gradio->-r requirements.txt (line 4)) (3.0.3)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.12/dist-package
s (from gradio->-r requirements.txt (line 4)) (3.11.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages
(from gradio->-r requirements.txt (line 4)) (25.0)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.12/dist-pa
ckages (from gradio->-r requirements.txt (line 4)) (2.2.2)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.12/dist
-packages (from gradio->-r requirements.txt (line 4)) (2.11.10)
Requirement already satisfied: pydub in /usr/local/lib/python3.12/dist-packages (fro
m gradio->-r requirements.txt (line 4)) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in /usr/local/lib/python3.1
2/dist-packages (from gradio->-r requirements.txt (line 4)) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.12/dist-pa
ckages (from gradio->-r requirements.txt (line 4)) (6.0.3)
Requirement already satisfied: ruff>=0.9.3 in /usr/local/lib/python3.12/dist-package
s (from gradio->-r requirements.txt (line 4)) (0.14.2)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in /usr/local/lib/python3.12/
dist-packages (from gradio->-r requirements.txt (line 4)) (0.1.7)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.12/di
```

```
st-packages (from gradio->-r requirements.txt (line 4)) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in /usr/local/lib/python3.12/d
ist-packages (from gradio->-r requirements.txt (line 4)) (0.48.0)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in /usr/local/lib/python3.12/
dist-packages (from gradio->-r requirements.txt (line 4)) (0.13.3)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.12/dist-pa
ckages (from gradio->-r requirements.txt (line 4)) (0.20.0)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.12/dist-pac
kages (from gradio->-r requirements.txt (line 4)) (0.38.0)
Requirement already satisfied: websockets<16.0,>=13.0 in /usr/local/lib/python3.12/d
ist-packages (from gradio-client==1.13.3->gradio->-r requirements.txt (line 4)) (15.
0.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-pa
ckages (from matplotlib->-r requirements.txt (line 5)) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packag
es (from matplotlib->-r requirements.txt (line 5)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-p
ackages (from matplotlib->-r requirements.txt (line 5)) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-p
ackages (from matplotlib->-r requirements.txt (line 5)) (1.4.9)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-pa
ckages (from matplotlib->-r requirements.txt (line 5)) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dis
t-packages (from matplotlib->-r requirements.txt (line 5)) (2.9.0.post0)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.12/dist-packag
es (from tensorboard->-r requirements.txt (line 11)) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.12/dist-pack
ages (from tensorboard->-r requirements.txt (line 11)) (1.76.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-pac
kages (from tensorboard->-r requirements.txt (line 11)) (3.9)
Requirement already satisfied: protobuf!=4.24.0,>=3.19.6 in /usr/local/lib/python3.1
2/dist-packages (from tensorboard->-r requirements.txt (line 11)) (5.29.5)
Requirement already satisfied: six>1.9 in /usr/local/lib/python3.12/dist-packages (f
rom tensorboard->-r requirements.txt (line 11)) (1.17.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/l
ib/python3.12/dist-packages (from tensorboard->-r requirements.txt (line 11)) (0.7.
2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-pac
kages (from tensorboard->-r requirements.txt (line 11)) (3.1.3)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages
(from huggingface-hub>=0.22->huggingface-hub[torch]>=0.22->-r requirements.txt (line
13)) (2.32.4)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dis
t-packages (from huggingface-hub>=0.22->huggingface-hub[torch]>=0.22->-r requirement
s.txt (line 13)) (1.2.0)
Requirement already satisfied: safetensors[torch] in /usr/local/lib/python3.12/dist-
packages (from huggingface-hub[torch]>=0.22->-r requirements.txt (line 13)) (0.6.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-packages
(from anyio<5.0,>=3.0->gradio->-r requirements.txt (line 4)) (3.11)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packag
es (from anyio<5.0,>=3.0->gradio->-r requirements.txt (line 4)) (1.3.1)
Requirement already satisfied: annotated-doc>=0.0.2 in /usr/local/lib/python3.12/dis
t-packages (from fastapi<1.0,>=0.115.2->gradio->-r requirements.txt (line 4)) (0.0.
3)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (f
rom httpx<1.0,>=0.24.1->gradio->-r requirements.txt (line 4)) (2025.10.5)
```

```
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packa
ges (from httpx<1.0,>=0.24.1->gradio->-r requirements.txt (line 4)) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages
(from httpcore==1.*->httpx<1.0,>=0.24.1->gradio->-r requirements.txt (line 4)) (0.1
6.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packag
es (from pandas<3.0,>=1.0->gradio->-r requirements.txt (line 4)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-pack
ages (from pandas<3.0,>=1.0->gradio->-r requirements.txt (line 4)) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/d
ist-packages (from pydantic<2.12,>=2.0->gradio->-r requirements.txt (line 4)) (0.7.
0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/di
st-packages (from pydantic<2.12,>=2.0->gradio->-r requirements.txt (line 4)) (2.33.
2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.1
2/dist-packages (from pydantic<2.12,>=2.0->gradio->-r requirements.txt (line 4)) (0.
4.2)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-
packages (from sympy>=1.13.3->torch->-r requirements.txt (line 1)) (1.3.0)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packag
es (from typer<1.0,>=0.12->gradio->-r requirements.txt (line 4)) (8.3.0)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.12/dist-
packages (from typer<1.0,>=0.12->gradio->-r requirements.txt (line 4)) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.12/dist-packa
ges (from typer<1.0,>=0.12->gradio->-r requirements.txt (line 4)) (13.9.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.1
2/dist-packages (from requests->huggingface-hub>=0.22->huggingface-hub[torch]>=0.22-
>-r requirements.txt (line 13)) (3.4.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-
packages (from requests->huggingface-hub>=0.22->huggingface-hub[torch]>=0.22->-r req
uirements.txt (line 13)) (2.5.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/di
st-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio->-r requirements.txt (line
4)) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/
dist-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio->-r requirements.txt (li
ne 4)) (2.19.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages
(from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio->-r requirement
s.txt (line 4)) (0.1.2)
Downloading roma-1.5.4-py3-none-any.whl (25 kB)
Downloading trimesh-4.9.0-py3-none-any.whl (736 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 736.5/736.5 kB 60.6 MB/s eta 0:00:00
Downloading pyglet-1.5.31-py3-none-any.whl (1.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 82.7 MB/s eta 0:00:00
Installing collected packages: pyglet, trimesh, roma
Successfully installed pyglet-1.5.31 roma-1.5.4 trimesh-4.9.0
```

In [30]:
```
!mkdir -p checkpoints/
```

In [31]:
```
!wget https://download.europe.naverlabs.com/ComputerVision/DUSt3R/DUSt3R_ViTLarge_B
```

```
--2025-10-30 01:21:16--  https://download.europe.naverlabs.com/ComputerVision/DUSt3
R/DUSt3R_ViTLarge_BaseDecoder_512_dpt.pth
Resolving download.europe.naverlabs.com (download.europe.naverlabs.com)... 110.234.5
6.25
Connecting to download.europe.naverlabs.com (download.europe.naverlabs.com)|110.234.
56.25|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2285005731 (2.1G)
Saving to: ‘checkpoints/DUSt3R_ViTLarge_BaseDecoder_512_dpt.pth’

DUSt3R_ViTLarge_Bas 100%[===================>]   2.13G  16.5MB/s    in 2m 19s

2025-10-30 01:23:36 (15.7 MB/s) - ‘checkpoints/DUSt3R_ViTLarge_BaseDecoder_512_dpt.p
th’ saved [2285005731/2285005731]
```

In [82]:
```
!python3 demo.py --model_name DUSt3R_ViTLarge_BaseDecoder_512_dpt
```

```
Warning, cannot find cuda-compiled version of RoPE2D, using a slow pytorch version i
nstead
/content/dust3r/dust3r/cloud_opt/base_opt.py:275: FutureWarning: `torch.cuda.amp.aut
ocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` inst
ead.
  @torch.cuda.amp.autocast(enabled=False)
[2025-10-30 02:34:42] Outputing stuff in /tmp/tmp9fgs_n77dust3r_gradio_demo
[2025-10-30 02:34:42] * Running on local URL:  http://127.0.0.1:7860
[2025-10-30 02:34:44] * Running on public URL: https://d205e7f9262505af90.gradio.liv
e
[2025-10-30 02:34:44]
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run
`gradio deploy` from the terminal in the working directory to deploy to Hugging Face
Spaces (https://huggingface.co/spaces)
ERROR:    Exception in ASGI application
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/uvicorn/protocols/http/h11_impl.py",
line 403, in run_asgi
    result = await app(  # type: ignore[func-returns-value]
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/uvicorn/middleware/proxy_headers.p
y", line 60, in __call__
    return await self.app(scope, receive, send)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/fastapi/applications.py", line 1134,
in __call__
    await super().__call__(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/applications.py", line 11
3, in __call__
    await self.middleware_stack(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/middleware/errors.py", lin
e 186, in __call__
    raise exc
  File "/usr/local/lib/python3.12/dist-packages/starlette/middleware/errors.py", lin
e 164, in __call__
    await self.app(scope, receive, _send)
  File "/usr/local/lib/python3.12/dist-packages/gradio/brotli_middleware.py", line 7
4, in __call__
    return await self.app(scope, receive, send)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/route_utils.py", line 888, in
__call__
    await self.simple_response(scope, receive, send, request_headers=headers)
  File "/usr/local/lib/python3.12/dist-packages/gradio/route_utils.py", line 904, in
simple_response
    await self.app(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/middleware/exceptions.py",
line 63, in __call__
    await wrap_app_handling_exceptions(self.app, conn)(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/_exception_handler.py", li
ne 53, in wrapped_app
    raise exc
  File "/usr/local/lib/python3.12/dist-packages/starlette/_exception_handler.py", li
ne 42, in wrapped_app
    await app(scope, receive, sender)
  File "/usr/local/lib/python3.12/dist-packages/fastapi/middleware/asyncexitstack.p
```

```
y", line 18, in __call__
    await self.app(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/routing.py", line 716, in
__call__
    await self.middleware_stack(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/routing.py", line 736, in
app
    await route.handle(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/routing.py", line 290, in
handle
    await self.app(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/fastapi/routing.py", line 124, in ap
p
    await wrap_app_handling_exceptions(app, request)(scope, receive, send)
  File "/usr/local/lib/python3.12/dist-packages/starlette/_exception_handler.py", li
ne 53, in wrapped_app
    raise exc
  File "/usr/local/lib/python3.12/dist-packages/starlette/_exception_handler.py", li
ne 42, in wrapped_app
    await app(scope, receive, sender)
  File "/usr/local/lib/python3.12/dist-packages/fastapi/routing.py", line 110, in ap
p
    response = await f(request)
               ^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/fastapi/routing.py", line 390, in ap
p
    raw_response = await run_endpoint_function(
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/fastapi/routing.py", line 289, in ru
n_endpoint_function
    return await dependant.call(**values)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/routes.py", line 1708, in upl
oad_file
    form = await multipart_parser.parse()
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/route_utils.py", line 742, in
parse
    async for chunk in self.stream:
  File "/usr/local/lib/python3.12/dist-packages/starlette/requests.py", line 236, in
stream
    raise ClientDisconnect()
starlette.requests.ClientDisconnect
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/gradio/queueing.py", line 759, in pr
ocess_events
    response = await route_utils.call_process_api(
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/route_utils.py", line 354, in
call_process_api
    output = await app.get_blocks().process_api(
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/blocks.py", line 2112, in pro
cess_api
    inputs = await self.preprocess_data(
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
  File "/usr/local/lib/python3.12/dist-packages/gradio/blocks.py", line 1774, in pre
process_data
    processed_input.append(block.preprocess(inputs_cached))
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gradio/components/slider.py", line 1
55, in preprocess
    Number.raise_if_out_of_bounds(payload, self.minimum, self.maximum)
  File "/usr/local/lib/python3.12/dist-packages/gradio/components/number.py", line 1
31, in raise_if_out_of_bounds
    raise Error(f"Value {num} is greater than maximum value {maximum}.")
gradio.exceptions.Error: 'Value 2 is greater than maximum value 1.'
[2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:
30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 0
2:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-
30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025
-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30]
[2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:
30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 02:36:30] [2025-10-30 0
2:36:30] [2025-10-30 02:37:42] [2025-10-30 02:37:42] [2025-10-30 02:37:42] [2025-10-
30 02:37:42] [2025-10-30 02:37:42] [2025-10-30 02:37:42] [2025-10-30 02:37:42] [2025
-10-30 02:37:42] [2025-10-30 02:38:44] >> Loading a list of 3 images
[2025-10-30 02:38:44]  - adding /tmp/gradio/c94c422d951ac2d6b3fb00aa1ecf0345ba6a903e
44931e9c20f15f7623397c3d/cam1_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:38:44]  - adding /tmp/gradio/07eda461d65a1132c8533c9a04b27449d6fa709c
d18fc6309fab6015a516efa9/cam2_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:38:44]  - adding /tmp/gradio/bf2d32ad3af6112b429ad5ec899e920aeee1ef0c
31ae1c4bf957fef7acff6de9/cam3_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:38:44]  (Found 3 images)
[2025-10-30 02:38:44] >> Inference with model on 6 image pairs
  0% 0/6 [00:00<?, ?it/s]/content/dust3r/dust3r/inference.py:44: FutureWarning: `tor
ch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda',
args...)` instead.
  with torch.cuda.amp.autocast(enabled=bool(use_amp)):
/content/dust3r/dust3r/model.py:206: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
/content/dust3r/dust3r/inference.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
100% 6/6 [00:04<00:00,  1.40it/s]
[2025-10-30 02:38:49]  init edge (2*,1*) score=np.float64(6.499037742614746)
[2025-10-30 02:38:49]  init edge (2,0*) score=np.float64(2.5505120754241943)
[2025-10-30 02:38:49]  init loss = 0.007280420046299696
[2025-10-30 02:38:49] Global alignement - optimizing for:
[2025-10-30 02:38:49] ['pw_poses', 'im_depthmaps', 'im_poses', 'im_focals']
  0% 0/300 [00:00<?, ?it/s]/content/dust3r/dust3r/cloud_opt/base_opt.py:366: UserWar
ning: Converting a tensor with requires_grad=True to a scalar may lead to unexpected
behavior.
Consider using tensor.detach() first. (Triggered internally at /pytorch/torch/csrc/a
utograd/generated/python_variable_methods.cpp:835.)
  return float(loss), lr
100% 300/300 [00:11<00:00, 26.75it/s, lr=3.433e-05 loss=0.00197967]
[2025-10-30 02:39:01] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 02:39:05] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
```

```
[2025-10-30 02:39:07] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 02:39:23] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 02:39:27] >> Loading a list of 3 images
[2025-10-30 02:39:27]  - adding /tmp/gradio/c94c422d951ac2d6b3fb00aa1ecf0345ba6a903e
44931e9c20f15f7623397c3d/cam1_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:39:27]  - adding /tmp/gradio/07eda461d65a1132c8533c9a04b27449d6fa709c
d18fc6309fab6015a516efa9/cam2_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:39:27]  - adding /tmp/gradio/bf2d32ad3af6112b429ad5ec899e920aeee1ef0c
31ae1c4bf957fef7acff6de9/cam3_time0.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:39:27]  (Found 3 images)
[2025-10-30 02:39:27] >> Inference with model on 6 image pairs
  0% 0/6 [00:00<?, ?it/s]/content/dust3r/dust3r/inference.py:44: FutureWarning: `tor
ch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda',
args...)` instead.
  with torch.cuda.amp.autocast(enabled=bool(use_amp)):
/content/dust3r/dust3r/model.py:206: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
/content/dust3r/dust3r/inference.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
100% 6/6 [00:02<00:00,  2.10it/s]
[2025-10-30 02:39:30]  init edge (2*,1*) score=np.float64(6.499037742614746)
[2025-10-30 02:39:30]  init edge (2,0*) score=np.float64(2.5505120754241943)
[2025-10-30 02:39:30]  init loss = 0.007280420046299696
[2025-10-30 02:39:30] Global alignement - optimizing for:
[2025-10-30 02:39:30] ['pw_poses', 'im_depthmaps', 'im_poses', 'im_focals']
100% 300/300 [00:10<00:00, 27.34it/s, lr=3.433e-05 loss=0.00197967]
[2025-10-30 02:39:41] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 02:40:06] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 02:51:17] >> Loading a list of 3 images
[2025-10-30 02:51:17]  - adding /tmp/gradio/ef54d72151f34e5e0f3d4d5030e9533db83044e8
f791d7fe188db8c0e0cfe4ee/cam1_time1.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:51:17]  - adding /tmp/gradio/b540f7e5303d8f5c2a401e890f288d951ad8d305
5f54c0a8e17c0e87d1998d0f/cam2_time1.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:51:17]  - adding /tmp/gradio/6f313bd585d8440923db887a67fd64e10c9cdf68
99db6530e319444229d3b04f/cam3_time1.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 02:51:17]  (Found 3 images)
[2025-10-30 02:51:17] >> Inference with model on 6 image pairs
  0% 0/6 [00:00<?, ?it/s]/content/dust3r/dust3r/inference.py:44: FutureWarning: `tor
ch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda',
args...)` instead.
  with torch.cuda.amp.autocast(enabled=bool(use_amp)):
/content/dust3r/dust3r/model.py:206: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
/content/dust3r/dust3r/inference.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
100% 6/6 [00:02<00:00,  2.07it/s]
[2025-10-30 02:51:20]  init edge (2*,1*) score=np.float64(8.518674850463867)
[2025-10-30 02:51:20]  init edge (2,0*) score=np.float64(2.8294925689697266)
```

```
[2025-10-30 02:51:20]  init loss = 0.0067738126963377
[2025-10-30 02:51:20] Global alignement - optimizing for:
[2025-10-30 02:51:20] ['pw_poses', 'im_depthmaps', 'im_poses', 'im_focals']
100% 300/300 [00:10<00:00, 27.49it/s, lr=3.433e-05 loss=0.00182599]
[2025-10-30 02:51:31] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 03:01:02] >> Loading a list of 3 images
[2025-10-30 03:01:02]  - adding /tmp/gradio/3368e774975b425a56256e8cc6a105aafc633dac
abf50221a0fc2c02720b121e/cam1_time2.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:01:02]  - adding /tmp/gradio/76ea0459b5c3114a56aefefbdc1794f78e7fc9d1
cfc1cf5dad9e3297afd00d6e/cam2_time2.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:01:02]  - adding /tmp/gradio/cfbc6162a468ec9a67f92a7bad767d1bef179d82
7036b09358b672b5f9a131b5/cam3_time2.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:01:02]  (Found 3 images)
[2025-10-30 03:01:02] >> Inference with model on 6 image pairs
  0% 0/6 [00:00<?, ?it/s]/content/dust3r/dust3r/inference.py:44: FutureWarning: `tor
ch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda',
args...)` instead.
  with torch.cuda.amp.autocast(enabled=bool(use_amp)):
/content/dust3r/dust3r/model.py:206: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
/content/dust3r/dust3r/inference.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
100% 6/6 [00:02<00:00,  2.05it/s]
[2025-10-30 03:01:05]  init edge (2*,1*) score=np.float64(7.443448543548584)
[2025-10-30 03:01:05]  init edge (2,0*) score=np.float64(3.3372232913970947)
[2025-10-30 03:01:05]  init loss = 0.007330354303121567
[2025-10-30 03:01:05] Global alignement - optimizing for:
[2025-10-30 03:01:05] ['pw_poses', 'im_depthmaps', 'im_poses', 'im_focals']
100% 300/300 [00:10<00:00, 27.41it/s, lr=3.433e-05 loss=0.00199635]
[2025-10-30 03:01:16] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 03:01:50] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 03:01:52] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 03:04:59] >> Loading a list of 3 images
[2025-10-30 03:04:59]  - adding /tmp/gradio/9a39c772062046a81f7018c2cdef91930b8b537a
3424201fb30a6fe560b132b8/cam1_time3.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:04:59]  - adding /tmp/gradio/7a2d9eafe7038ec37eec16ab20e41598517150d4
db7ebf197e8dafcf95677321/cam2_time3.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:05:00]  - adding /tmp/gradio/483b6261ab61d726470c066038bf434d22c23a85
92f2c754c95acf733b3e7825/cam3_time3.jpg with resolution 1920x1080 --> 512x288
[2025-10-30 03:05:00]  (Found 3 images)
[2025-10-30 03:05:00] >> Inference with model on 6 image pairs
  0% 0/6 [00:00<?, ?it/s]/content/dust3r/dust3r/inference.py:44: FutureWarning: `tor
ch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autocast('cuda',
args...)` instead.
  with torch.cuda.amp.autocast(enabled=bool(use_amp)):
/content/dust3r/dust3r/model.py:206: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
  with torch.cuda.amp.autocast(enabled=False):
/content/dust3r/dust3r/inference.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
s...)` is deprecated. Please use `torch.amp.autocast('cuda', args...)` instead.
```

```
  with torch.cuda.amp.autocast(enabled=False):
100% 6/6 [00:02<00:00,  2.07it/s]
[2025-10-30 03:05:03]  init edge (2*,1*) score=np.float64(10.071928977966309)
[2025-10-30 03:05:03]  init edge (2,0*) score=np.float64(3.9187803268432617)
[2025-10-30 03:05:03]  init loss = 0.019163010641932487
[2025-10-30 03:05:03] Global alignement - optimizing for:
[2025-10-30 03:05:03] ['pw_poses', 'im_depthmaps', 'im_poses', 'im_focals']
100% 300/300 [00:10<00:00, 27.55it/s, lr=3.433e-05 loss=0.011311]
[2025-10-30 03:05:14] (exporting 3D scene to /tmp/tmp9fgs_n77dust3r_gradio_demo/scen
e.glb )
[2025-10-30 03:11:18] Keyboard interruption in main thread... closing server.
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/gradio/blocks.py", line 2958, in blo
ck_thread
    time.sleep(0.1)
KeyboardInterrupt

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/content/dust3r/demo.py", line 45, in <module>
    main_demo(tmpdirname, model, args.device, args.image_size, server_name, args.ser
ver_port, silent=args.silent)
  File "/content/dust3r/dust3r/demo.py", line 287, in main_demo
    demo.launch(share=True, server_name=server_name, server_port=server_port)
  File "/usr/local/lib/python3.12/dist-packages/gradio/blocks.py", line 2865, in lau
nch
    self.block_thread()
  File "/usr/local/lib/python3.12/dist-packages/gradio/blocks.py", line 2960, in blo
ck_thread
    print("Keyboard interruption in main thread... closing server.")
  File "/content/dust3r/dust3r/demo.py", line 61, in print_with_timestamp
    builtin_print(*args, **kwargs)
KeyboardInterrupt
[2025-10-30 03:11:18] Killing tunnel 127.0.0.1:7860 <> https://d205e7f9262505af90.gr
adio.live
```

16-820: Advanced Computer Vision

HW-3

Akanksha Singal (asingal2)

October 30, 2025

Part 1 Theory

Q1.1

Given according to the diagram the two points x and x' are $(0, 0)$. We know that we can write the epipolar constraints as:

$$x_2^T F x_1 = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Thus,

$$f_{33} = 0$$

Q1.2

At time i, we have R_i and t_i At time $i + 1$, we have R_{i+1} and t_{i+1}

Relative rotation and translation between frames i and $i+1$: At time i, point in camera coordinate frame X_i using world coordinate frame X_w

$$X_i = R_i X_w + t_i$$

We can write the above equation for world frame coordinates as:

$$X_w = R_i^{-1}(X_i - t_i)$$

Since R_i is orthonormal matrices the inverse is equal to transpose so we can write the above equation as:

$$X_w = R_i^T(X_i - t_i)$$

At time i, point in camera coordinate frame X_{i+1} using world coordinate frame X_w

$$X_{i+1} = R_{i+1} X_w + t_{i+1}$$

Substituting the world coordinate expression in the above equation:

$$X_{i+1} = R_{i+1} R_i^T (X_i - t_i) + t_{i+1}$$

$$X_{i+1} = (R_{i+1} R_i^T) X_i + (t_{i+1} - R_{i+1} R_i^T t_i)$$

From above equation,

$$R_{rel} = R_{i+1} R_i^T$$

$$t_{rel} = t_{i+1} - R_{rel} t_i$$

We know that,

$$E = \hat{T} R$$

and

$$F = K_2^{-1} \hat{T} R K_1$$

Thus for relative motion we can write this as:

$$E = \hat{T_{rel}} R_{rel}$$

Since the camera is same

$$F = K^{-1} \hat{T_{rel}} R_{rel} K$$

Part 2

Problem 6

0.0.1 6.2

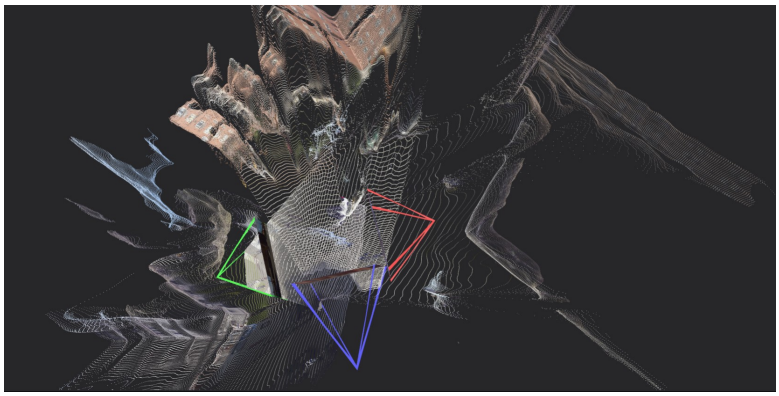


Figure 1: Time 0

Figure 2: Time 0 RGB and confidence

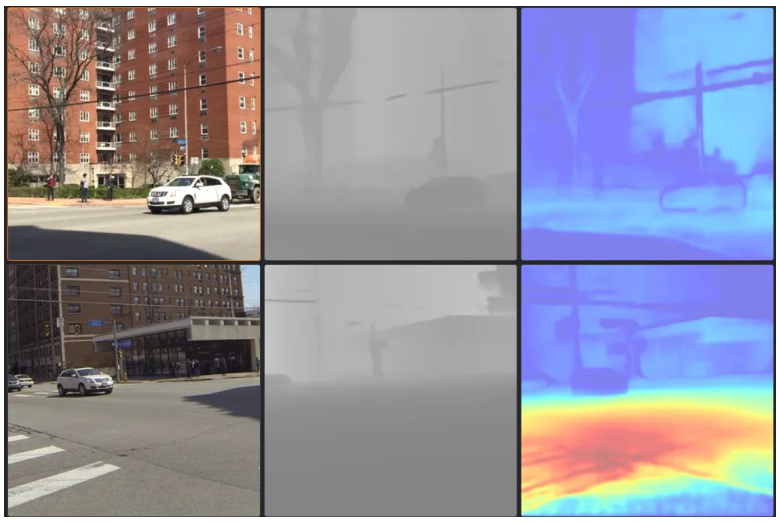


Figure 3: Time 0 RGB and confidence

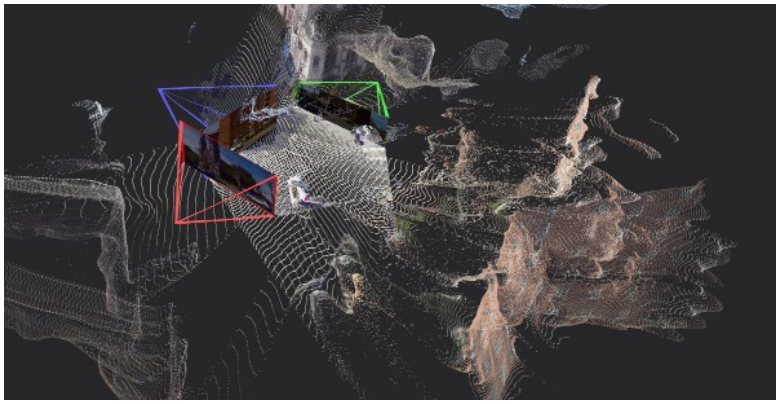


Figure 4: Time 1

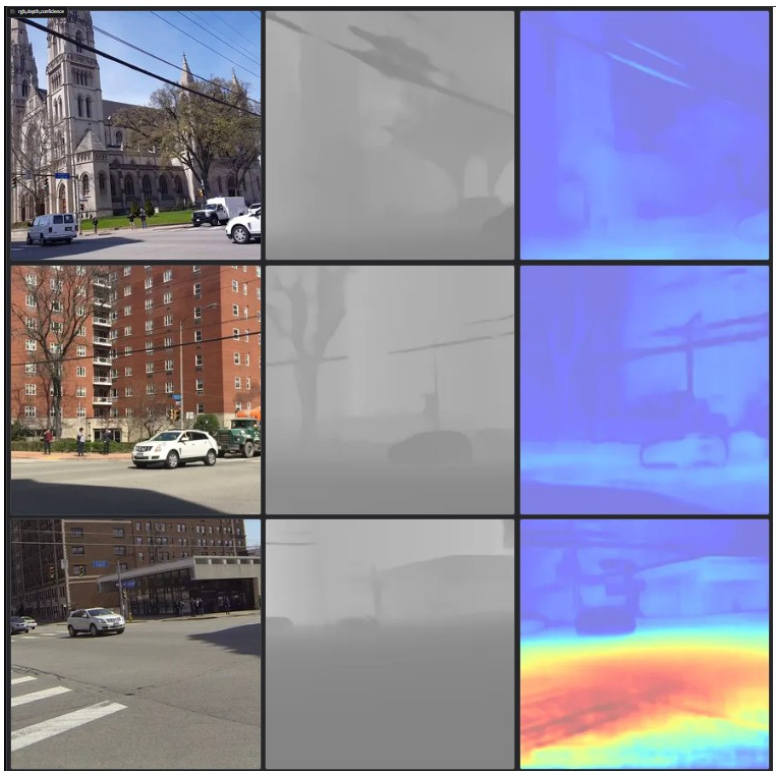


Figure 5: Time 1 RGB and confidence

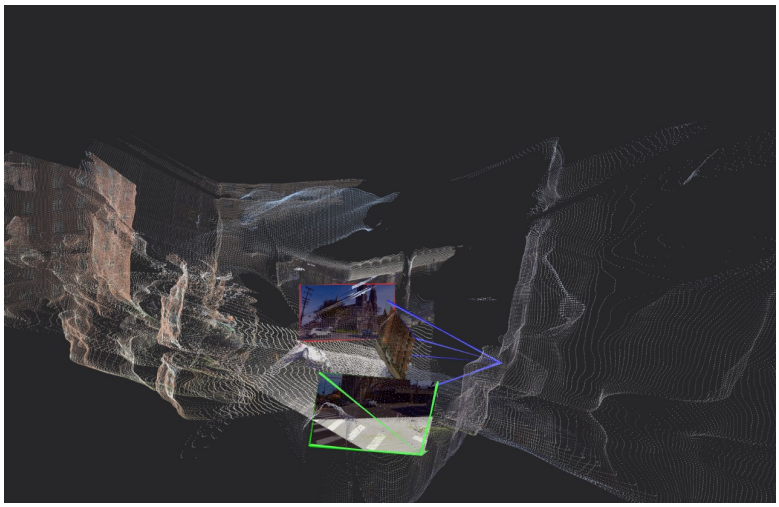


Figure 6: Time 3

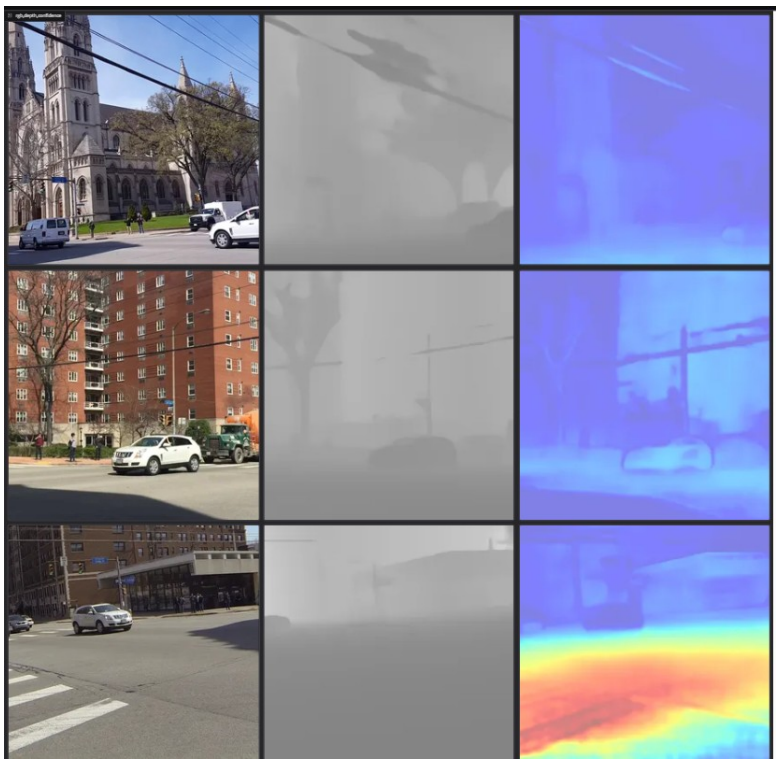


Figure 7: Time 2 RGB and confidence

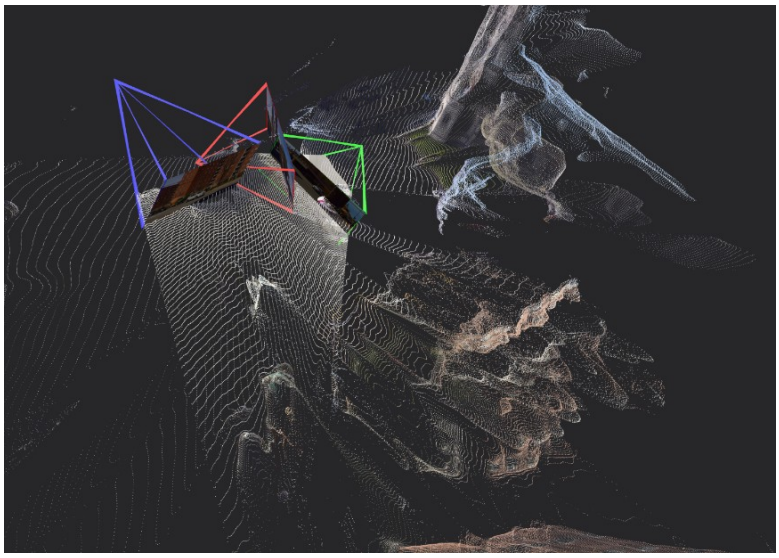


Figure 8: Time 3

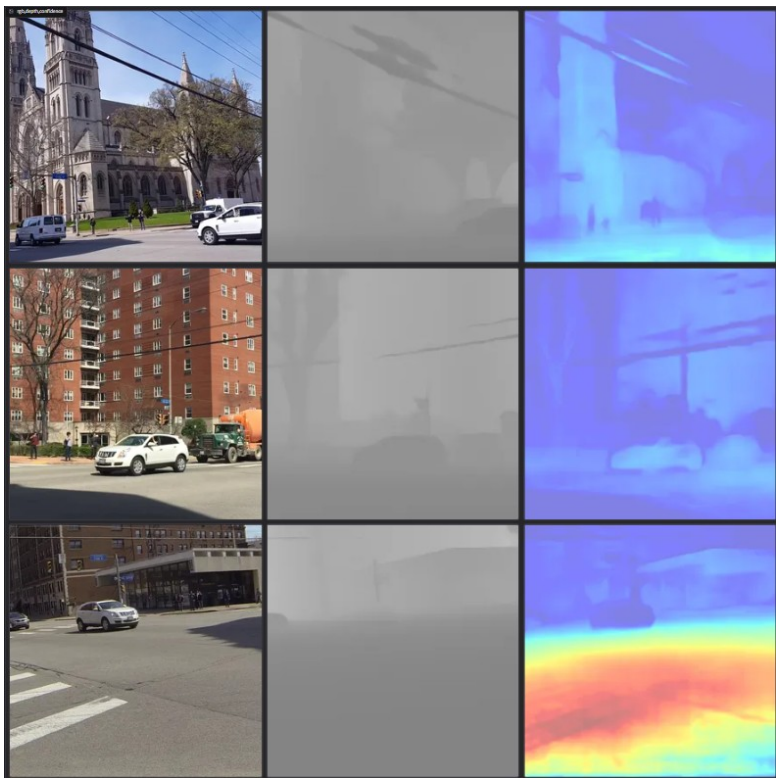


Figure 9: Time 3 RGB and confidence

Dust3r doesn't work in dynamic case as it we see that the car is not reconstructed properly but the background was constructed properly. Dust3r performs well on sets of images with sufficient overlap, distinct textures, and consistent viewpoints, as it leverages dense matching and geometric consistency to recover accurate camera poses and 3D point clouds. If the scene contains static backgrounds and minimal occlusions or motion blur, DUSt3R is likely to give reliable reconstructions. However, it dynamic elements, repetitive patterns, large viewpoint changes, or poorly textured areas, it leads to incorrect matches and noisy 3D reconstructions as we can see from the results. However, the advantage of Dust3r is that we dont need to estimate or know camera intrinsics to estimate. And we also get the camera positions from dust3r. On static backgrounds, DUSt3R typically performs robustly, as consistent features across views enhance correspondence estimation and depth triangulation.