take 3 views and reconstruct 3d

1. Compute SIFT and Match images -> Use opencv

```
-- If your python SIFT is not working, I have created sift matches for you.
-- use scipy.io.loadmat() to load the .mat file
import scipy.io as io
data = io.loadmat(‘sift_matches.mat’)
q11 = data[‘p11’]

-- p11, p12 -> image1 points and image2 points
-- p22, p23 -> image2 points and image3 points
-- p31, p33 -> image3 points and image1 points


sift = cv2.SIFT()
kp, des = sift.detectAndCompute(gray,None)


IF YOU ARE USING SIFT
https://stackoverflow.com/questions/46607647/sift-feature-matching-point-coordinates

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html
```

- Match image 1 to 2
- Match image 2 to 3
- Match image 3 to 1

  Normalize points with K^{-1} => see lecture
  (u v 1) = K^-1 * (u’ v’ 1)
  K is the calibration matrix in intrinsics.txt

2. Implement RANSAC for the Essential matrix

Pseudocode:

```
For i = 1 to N_iterations
    Select random 8 points
    Get the Essential matrix
    Compute residual of the epipolar equation and threshold
    Count number of inliers
    iF inliers > previousMax
        Save inlier index
        Save Essential Matrix
Repeat
```

3. Compute E_12, E_23, E_31 using RANSAC and remove outliers

4. Make sure that E’s are rank 2.. (Hint: Use SVD)

5. Use the function cv2.recoverpose() to get R_12, t_12, R_23, t_23, R_23, R_31, t_31

6. Solve for the scale:
t_12 + a t_23 + b t_31 = 0

7. Find t's

8. Triangulate all points and transfer to common reference frame
Draw a figure of the cameras and poses in a note and see how you can transfer points.

9. Visualize 3D

# Extras

Generating 3D is a long process so there are other steps you have to consider many things. So these are only optional things.

1. Find common matches between all three using the SIFT descriptors.
2. Remove duplicate points in 3D coming from common matches.
3. Formulate an optimization problem where you minimize sum of reprojection errors. The parameters you can change are R, t, p