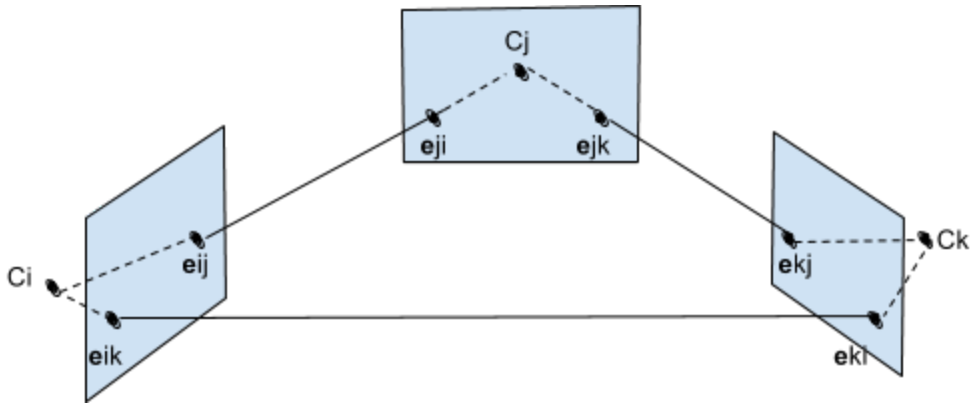
1.1

- The points on this plane will be imaged exactly as if we had a projective camera. The intuition is that since our affine camera center is at infinity, we are infinitely far from the plane at the world origin, and we know that perspective effects diminish as the camera moves farther away from the object.
- The projection of the centroid of the world points is also the centroid of the imaged world points.
- We know that parallel lines intersect at infinity. Also, affine cameras preserve the plane at infinity. So points at infinity in the world will also be imaged at infinity by an affine camera. Thus, parallel lines in the world will also remain parallel in the image.
- Let $p1$ and $p2$ be points on one line and $p3$ and $p4$ be points on the second line. Now we can write (if we think of the points as vectors, the subtraction of them gives the vector representing the line segment between them) $\frac{||p1-p2||}{||p3-p4||} = r$ or equivalently $||p1-p2|| = r*||p3-p4||$ and also $(p1-p2) = r*(p3-p4)$. Now, we apply the affine transformation:

  $$A(p1-p2) = A(r*(p3-p4))$$
  $$||A(p1-p2)|| = ||A(r*(p3-p4))||$$
  $$||A(p1-p2)|| = r*||A(p3-p4)||$$

  Thus, the ratio is preserved.
- If we backproject 2 different points on the image plane, their rays (in the world) will be parallel since affine camera means parallel projection. We showed earlier that parallel world lines will map to parallel image lines, so that means the epipolar lines will also be parallel. Thus, the epipolar planes (plane for which the intersection with the image plane is the epipolar line) will also be parallel.
- This is the analog of how in the projective case, our camera matrices are determined up to a projective ambiguity (that is, we can multiply our matrices by an arbitrary 4x4 projective transformation, and the fundamental matrix will be invariant). Similarly, we can use this argument for the affine case. The other intuition we have is that since our affine cameras are located at infinity, an affine transformation of the coordinate system leaves them unchanged, and thus the imaged points should be unchanged as well.

1.2

- 

- So we want the epipolar line of e_{ik} in image j. Geometrically, we see that backprojecting e_{ik} gives the point Ck in the world. By inspection, we see that Ck will project to e_{jk} in image j. Additionally, the other endpoint on the ray is Ci. We see that Ci is projected as e_{ji} in image j. Thus, the epipolar line is the line containing $e_{jk}\ and\ e_{ji}$, so we have shown that $F_{ij}e_{ik} = e_{jk}\ x\ \ e_{ji}$. Similarly, this holds for the other triplets.
- From the above result, we can write 3 relations:

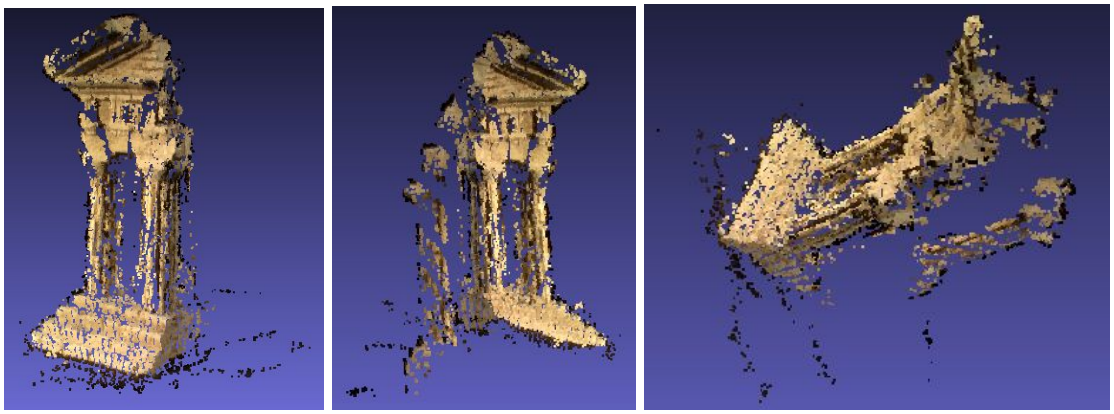
$$e_{jk}^{T}F_{ij}e_{ik} = e_{ij}^{T}F_{ik}e_{kj} = e_{ki}^{T}F_{jk}e_{ji} = 0$$

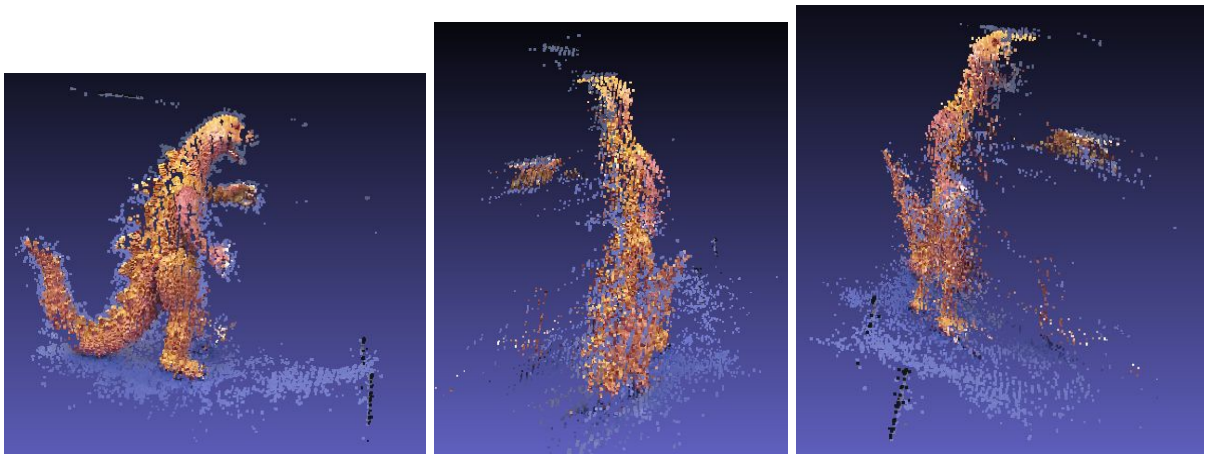
Three fundamental matrices would give us 21 parameters, but since we have these 3 relations, we are reduced to 18 degrees of freedom.

2.1

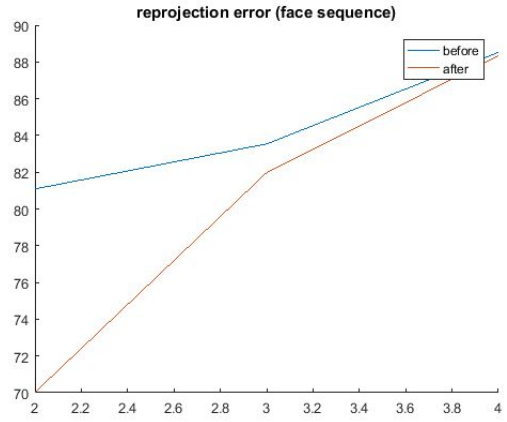
- 
- Major Steps
  - Intrinsics
    - It tries to extract the focal length from EXIF data. If this is not possible, we have to manually specify the focal length of our images.
  - Pairwise-Image SFM
    - Extract features using SIFT descriptor and match between images
    - Estimate fundamental matrix using point correspondences
    - Extract relative pose between cameras from essential matrix
    - Triangulate points and then refine with bundle adjustment
  - Graph Merging
    - Sequentially merge graphs and refine with bundle adjustment in between each merge.
  - Dense Reconstruction
    - Use ZNCC (zero mean normalized cross correlation) to find initial matches in the image. Maintain a priority queue with the best matches.

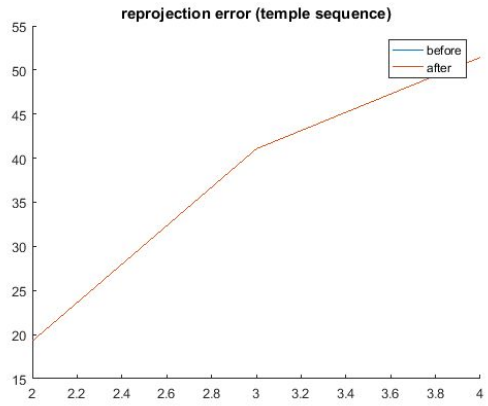
        - Throughout the iterations, take the best matches as well as initializing potential good matches located in the same spatial area as the best matches.
        - Finally, triangulate all the matches found.
- The assumption is that (in normalized coordinates), the principal point is at the center of the image. This assumption would get violated if we are working with cropped images.
- Graph
    - f: ‘f’ is the focal length. Since we are assuming the principal point is at the center of the image, ‘f’ essentially gives us our intrinsic matrix
    - Mot: This represents the relative pose of each camera. (:,:,i) is the relative pose of camera ‘i’
    - Str:This represents the 3D location of the triangulated points
    - ObsVal: This stores the locations of feature points in all images.
    - ObsIdx: This matrix indexes the feature points in the ObsVal structure to each image. Row ‘i’ indicates all observed feature points in frame ‘i’. If there is a zero, it means that the feature is not observed in that frame. ObsVal(:, ObsIdx(i,j)) gives the image coordinates of feature ‘j’ in image ‘i’.
- Reconstruction Results
    - I had to choose images that had a relatively small motion between images so as to ensure that there would be good feature detection overlap between frames. Additionally, I had to make sure that the object of interest could have a good amount of salient features could be detected.
    - I would recommend to the user that there should not be a lot of motion between the different images. The user should also choose a target object that has a lot of interesting features on its surface. Finally, the lighting conditions or the appearance of the surface of the object should remain fairly consistent so that the same features can be detected across all images.
    - Based on the recommendations, it’s clear that the most unstable step is the feature matching. If we don’t have good feature correspondences, then our whole estimation pipeline will produce poor results.

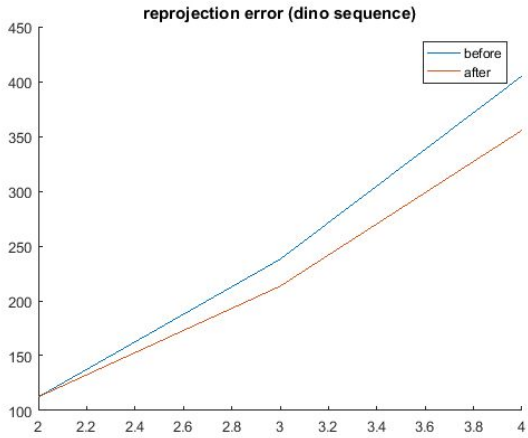


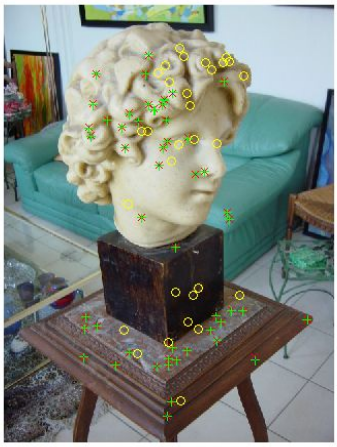


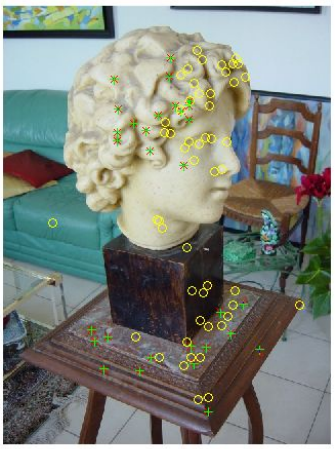
- Reprojection Errors
    - Errors shown are during the graph merging sequence, which is why they are increasing. In the face and dino sequence, we see that the error decreases whereas in the temple sequence it remains the same.

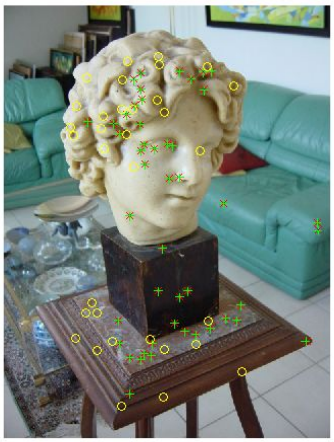


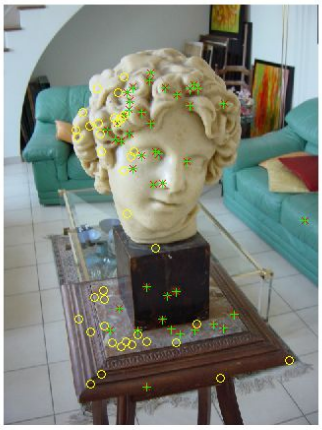


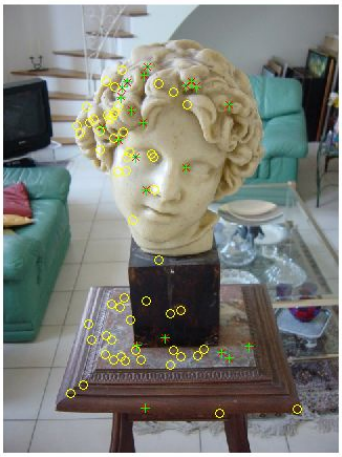


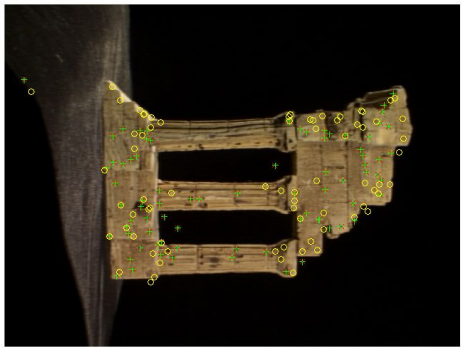


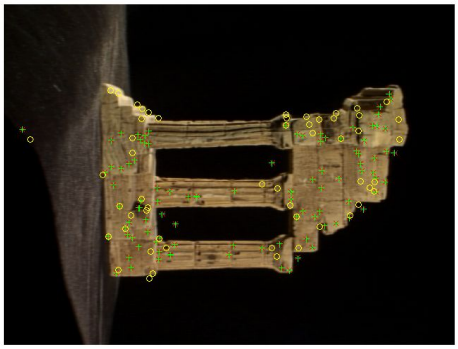


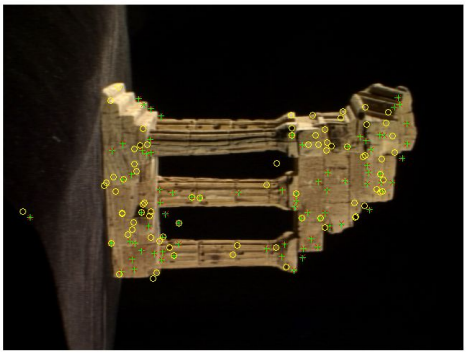


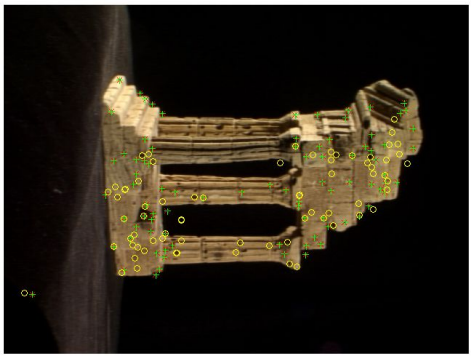


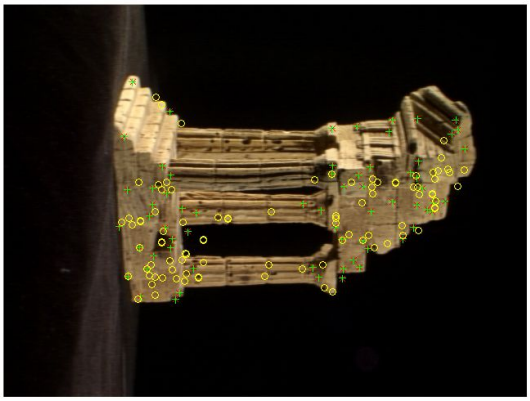


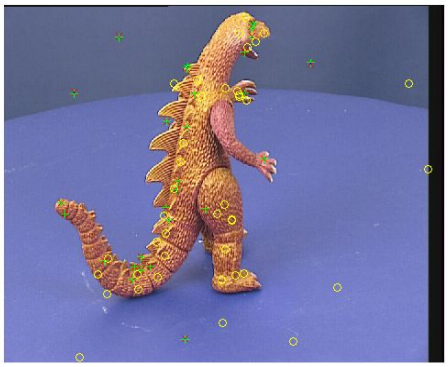


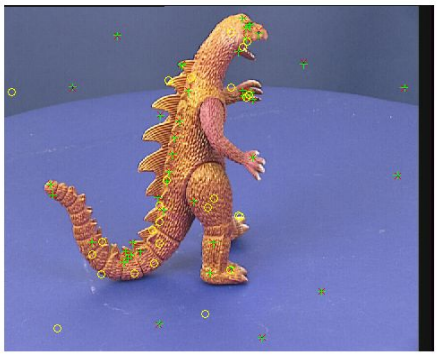


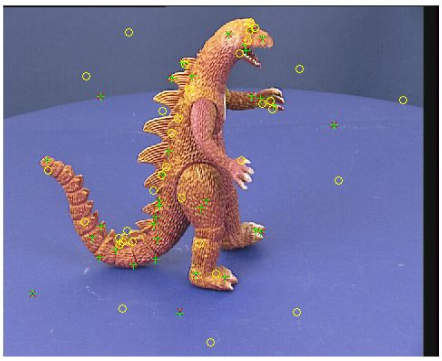


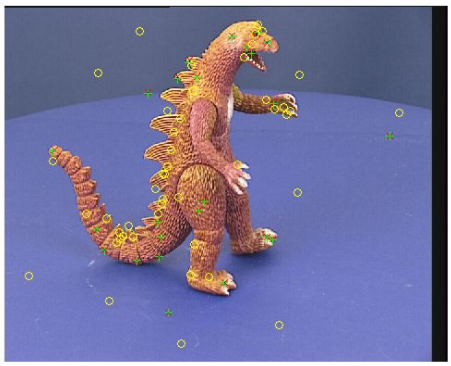












- Levenberg-Marquardt
    - We are trying to minimize reprojection error. Specifically, we are taking the argmin over the 3D points P and the motion M

$$\sum_{i,j} d\left(\mathbf{p}_{ij}, \mathbf{M}_i \mathbf{P}_j\right)^2 = \sum_{i,j} \left(\mathbf{p}_{ij}^1 - \frac{\mathbf{m}_i^{1T} \mathbf{P}_j}{\mathbf{m}_i^{3T} \mathbf{P}_j} \right)^2 + \left(\mathbf{p}_{ij}^2 - \frac{\mathbf{m}_i^{2T} \mathbf{P}_j}{\mathbf{m}_i^{3T} \mathbf{P}_j} \right)^2$$

    - MATLAB's standard nonlinear least squares solver explicitly forms J^TJ (where J is the Jacobian matrix) before computing the conjugate gradient. Therefore, a row of J with many nonzeros, which results in a nearly dense product J^TJ, can lead to a costly solution process for large problems. If we have many observations (points), then this will become a problem.
- Motion Adjustment
    - We are optimizing the same reprojection error equation as above, except we are now only taking the argmin over the motion M
    - For the face image sequence, the reprojection error goes up to 1390 from 88 when we optimize both motion and structure.

- Structure Adjustment
    - We are optimizing the same reprojection error equation as above, except we are now only taking the argmin over the structure P
    - For the face image sequence, the reprojection error goes up to 1427 from 88.
    - The drawback is that the optimization is just a black box and does not exploit additional information, such as our image size and camera matrices. In the

provided routine for triangulation, the triangulated points are better parameterized by utilizing this information.

- Smarter Graph Merging
    - For the face image sequence, the original order was already pretty close the order given by maximal correspondences. In this case, the order changed from [1, 2, 3, 4] to [2, 1, 3, 4]. The reprojection error dropped a bit from 88 to 78.

- Intrinsics
    - From optimizing the intrinsics, I obtained a final K of:

745.6286	0	-28.9999
0	745.6286	88.2580
0	0	1.0000