Preprocess.py

1. Detect_keypoints():
   a. Given the image file path, read the image.
   b. Using cv2.SIFT, get the keypoints of the image.
2. Create_feature_matches():
   a. Using cv2.BFMatcher, feature match descriptors from each image using k_nearest_neighbour value of 2.
   b. Filter the feature matches using the lowe ratio test using a ratio of 0.6.
3. Create_ransac_matches():
   a. Using cv2.findEssentialMatrix, calculate the essential matrix and the inlier mask to differentiate inliers and outliers.
4. Create_scene_graph():
   a. For each image pair, if the number of inlier feature matches exceeds the minimum number of inliers threshold, we add an edge between those 2 images' nodes.

Sfm.py

1. Get_init_image_ids():
   a. For each pair of images, select the image pair that has the highest number of feature matches.
2. Get_init_extrinsics():
   a. Using cv2.recoverPose, recover the rotation and translation vectors from essential matrix.
   b. Using cv2.rodrigues, convert rotation vector to the rotation matrix.
   c. Create and return the extrinsic matrix by appending translation vector to rotation matrix [R|t].
3. Get_next_pair():
   a. For each image id in registered_ids, check each of its adjacent nodes (which is not registered) and select and return the pair with the highest number of feature match inliers.
4. Solve_pnp():
   a. Using cv2.solvePnP, calculate the rotation and translation vectors from the camera intrinsic and selected 3d/2d points.
   b. Using cv2.rodrigues, convert rotation vector to the rotation matrix.
   c. Use get_reprojection_residuals() to calculate the reprojection residuals
      i) Get_reprojection_residuals():
         (1) Create the homography using intrinsic and extrinsic matrices.
         (2) For each point in points3d, reproject it onto the 2d plane using homography matrix.
         (3) Calculate the Euclidean distance between reprojected 2d point and original 2d point.
         (4) Return the residuals once all points3d have been reprojected.
5. Add_points3d():
   a. Given the unregistered feature matches between 2 images, triangulate these corresponding points to create new 3d points.

Bundle_adjustment.py

1. Compute_ba_residuals():
   a. Calculate the different homography matrices from the intrinsic and extrinsic matrices.
   b. Homogenise the 3d points given by appending a $[1,\ldots..]^T$ vector to the 3d points.
   c. Retrieve the corresponding 3d points using points3d[points3d_idxs].
   d. Reproject the corresponding 3d points using the homography matrices calculated before and normalise the new reprojected 2d points.
   e. Calculate the difference between the (x-axis values of the reprojected points and initial 2d points) squared.

f. Calculate the difference between the (y-axis values of the reprojected points and initial 2d points) squared.
g. Square root the x-axis/y-axis differences to get the Euclidean distances between initial and reprojected 2d points across the different camera poses.
h. Sum the Euclidean distances across the different camera poses to get the ba residuals.

Results

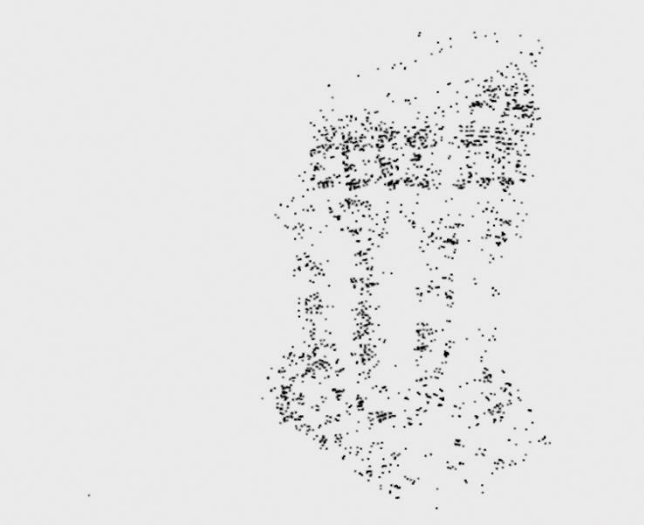

Figure 1 Mini-temple w/o BA

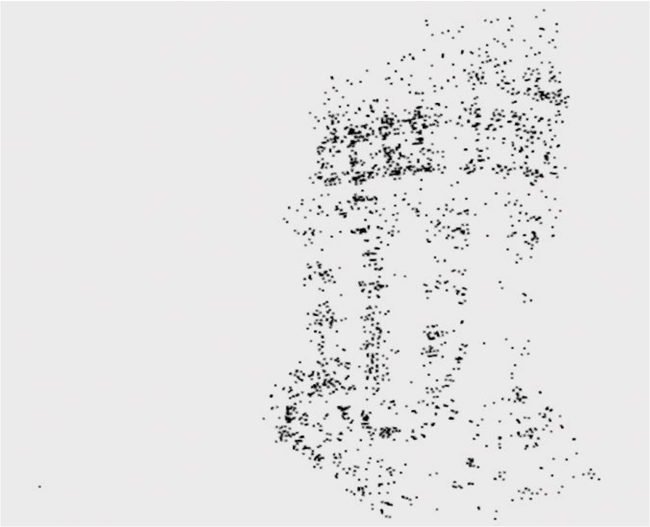

Figure 1 TA's Mini-temple w/o BA

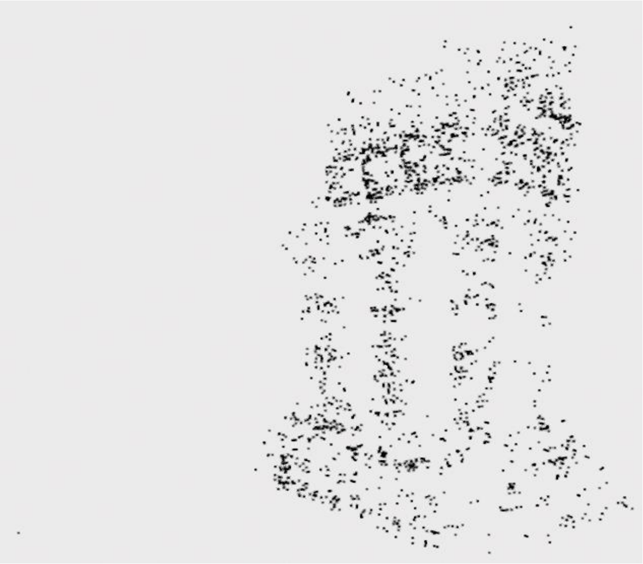

Figure 3 Mini-temple with BA

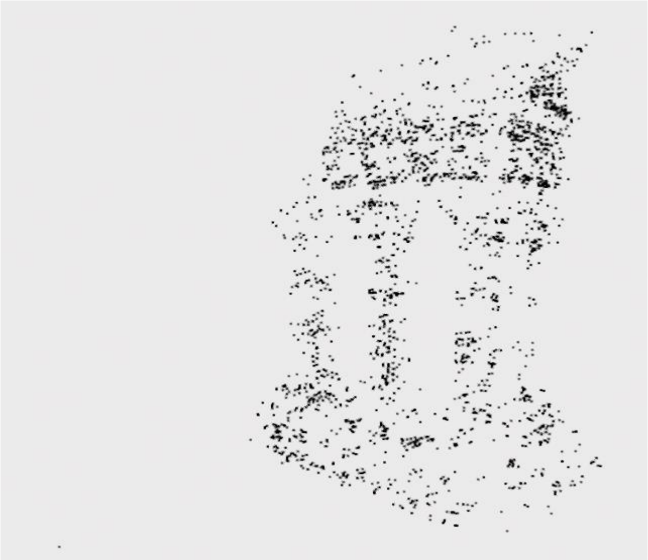

Figure 4 TA's Mini-temple with BA

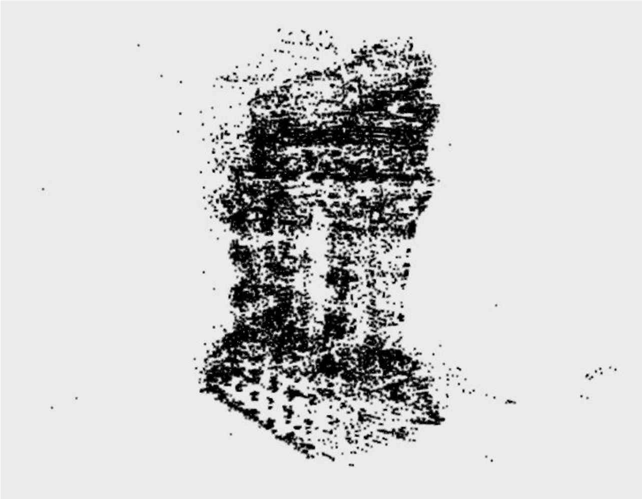

Figure 5 Temple w/o BA

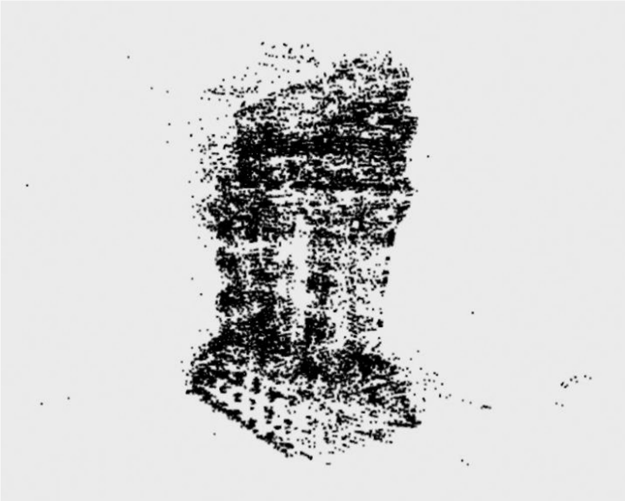

Figure 6 TA's Temple w/o BA