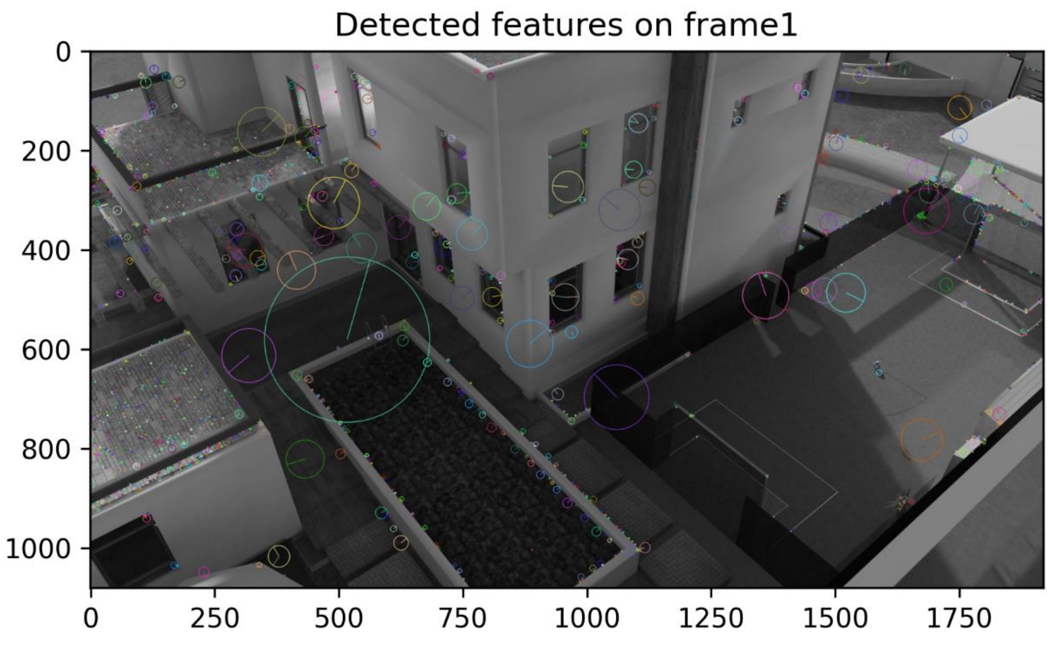
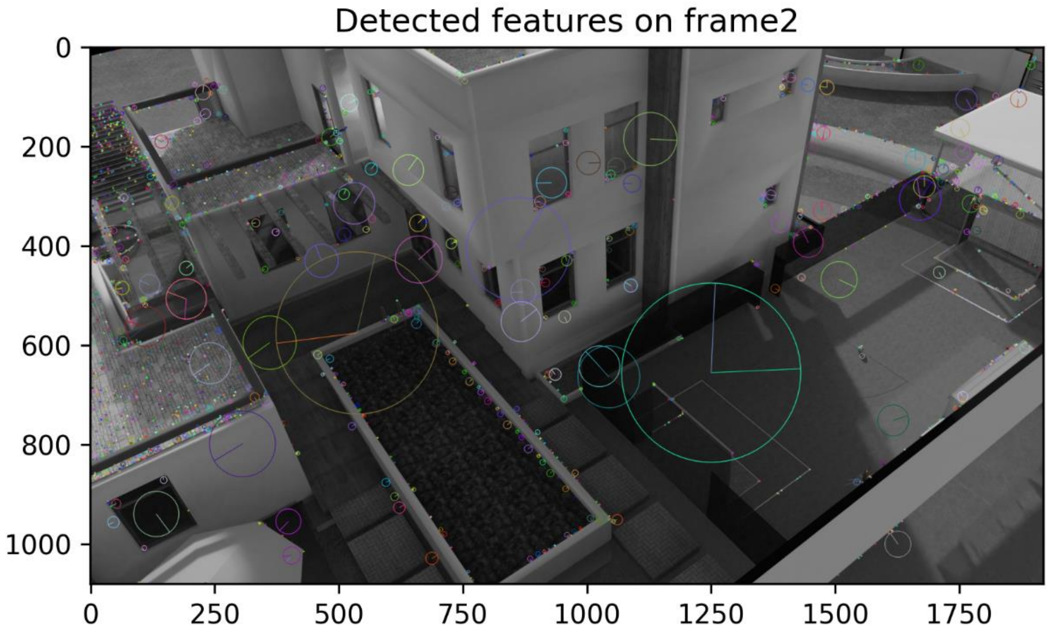
1.
SIFT is a suitable technique to detect salient features, and I will focus on features that have good locality, invariance, repeatability, and distinctiveness. In the given frames, they will be corners, e.g., corners of windows; blobs, e.g., footballs on the ground; distinctive patterns on building; building edges that have strong contrast, these features are stable and immune to geometrical changes during frames while providing good information. SIFT has 4 steps, scale-space extrema detection, keypoint localization, orientation estimation and keypoint descriptor. By doing the first step, SIFT achieves scale-invariant, keypoint localization removes some edges and features detected earlier which don't have enough contrast based on DoG Value, orientation estimation makes it orientation-invariant, and using gradient orientations makes it robust to intensity value changes. And it's faster than a traditional Harris-Laplace detector because it uses DoG instead of LoG. These properties make SIFT a good choice to detect salient features.

2. To match the detected salient features, I use **Nearest Neighbor Distance Ratio** test and brute-force matcher. To do this, I first use BFMatcher to find the nearest and second nearest neighbors according to the **Euclidean distance** on detected salient features. Then, I apply the NNDR with ratio threshold **0.4** on the distance of nearest neighbor d1 and distance of second nearest neighbor d2 to reject ambiguous matching, which only record match when $d1/d2 < 0.4$ to reduce the false positive rate. Regarding how I arrived at the threshold, I started testing the threshold from 0.8 and the initial output graph contained many obviously incorrect matches, when I lowered the threshold to 0.4 most of the matches appeared to be correct and the number of matches was still high, so I chose to use 0.4.

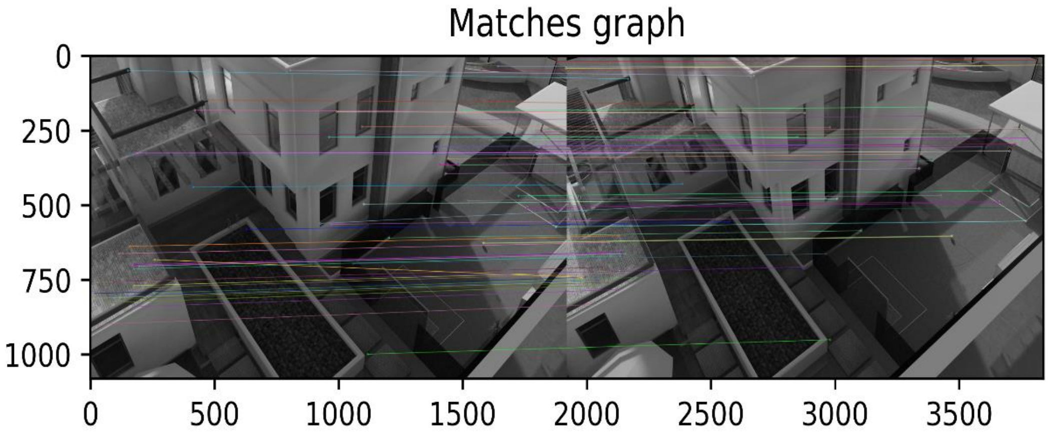
3. a. The detected features are show in figures below, each feature is surrounded by a circle, the circle size represents the scale, and the line represents the orientation.
(1)



a (2)



3.b In the figure below, each match is connected by a line.



3.c

By using the matched features, I applied the RANSAC algorithm for estimation and get:

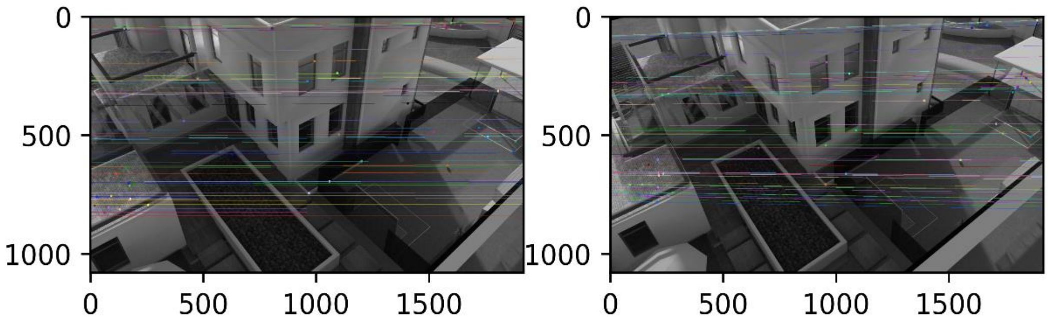
```
array([[ 6.27421664e-08, -7.53153926e-07, -9.89594893e-04],
       [ 1.87201826e-06, -5.78501221e-07, -2.46870627e-02],
       [ 2.98522134e-04,  2.22422573e-02,  1.00000000e+00]])
```

I used $F = [K't]_{\times} K'RK^{-1}$ on extrinsic and intrinsic camera parameters, and get:

```
array([[-8.94716958e-04, -1.95802211e-01,  1.10989237e+02],
       [ 5.60156937e-02,  3.04586526e-03,  2.93130697e+03],
       [-4.19999288e+01, -2.58338899e+03, -1.17061260e+05]])
```

The method that using extrinsic and intrinsic camera parameters is more accurate, since the camera parameters are provided and the position of cameras are consistent, compare with the fundamental matrix estimated from those matched features we picked from SIFT detector and NNDR test with self-picked ratio threshold, the probability of having inaccurate calibration is way less than having incorrect matches. To improve the least accurate method, we should increase the matching accuracy. For example, in this case we have enough matches, so we can apply a stricter NNDR threshold to make the detected matches' false positive rate lower while still have enough matches to make the resulted ·fundamental matrix more accurate.

3.d. Below is the figure of matches that meet the epipolar constraint using the fundamental matrix that estimated by matched features above. Here I use the fundamental matrix computed from extrinsic and intrinsic camera parameters because it is more accurate. A fundamental matrix should satisfies $x'^T F x = 0$ on match pair (x,x'). And since we need to ensure epipolar constraint, here we use points from RANSAC that already removed outliers to ensure uniqueness, ordering and smoothness constraint. To visualize the matching, since $l' = e' \times (P'P^{+})x$ and $F = e' \times (P'P^{+})$, we can use matrix F and point from left frame to draw the epipolar line on the right frame, on the computed epipolar line, we can find the matched point on the epipolar line. For the other frame, just use F^T instead of F and do the same thing.



3. e

I choose to estimate the area and length by 3D surface reconstruction.

First, we manually check the coordinates of required points on the 2d images. We get:

3 corner points of swimming pool and 2 corner points of football field on different frames. Then we update the intrinsic and extrinsic camera parameters by applying stereo image rectification to re-projecting image planes to align the images by cv.stereoRectify. Then we use cv.triangulatePoints on the rectified points to get the real-world coordinate. After that we can use cv.convertPointsFromHomogeneous to converts the points to Euclidean space, then we can do the calculation on it to get the result. And the figure below is the result.

```
Area of swimming pool: 36.39632316452956.
Length of football field: 14.180912321023175.
```