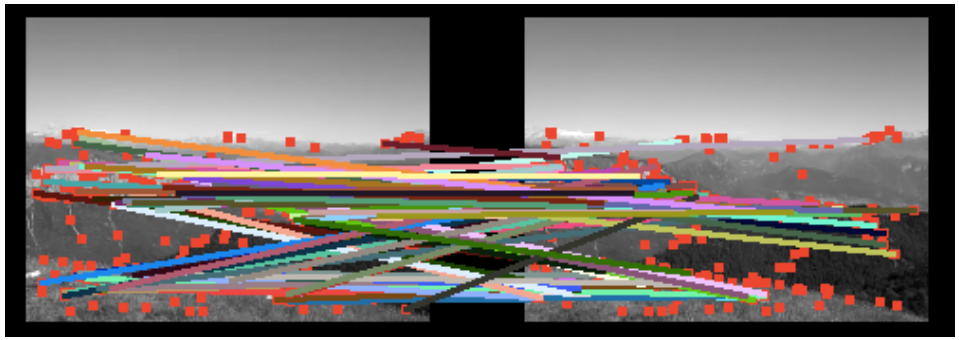
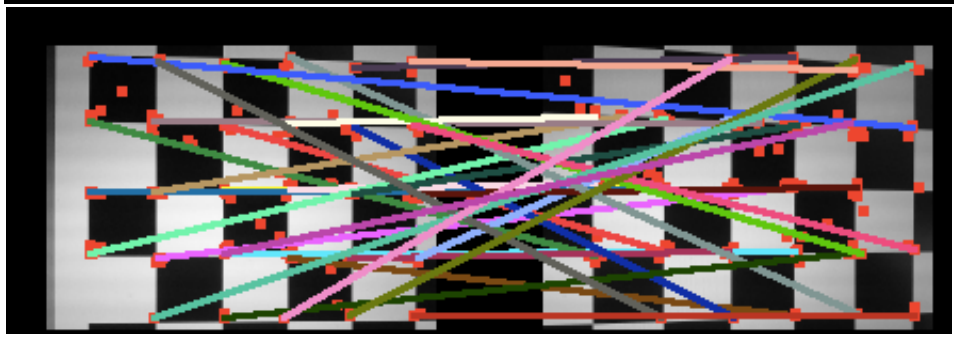
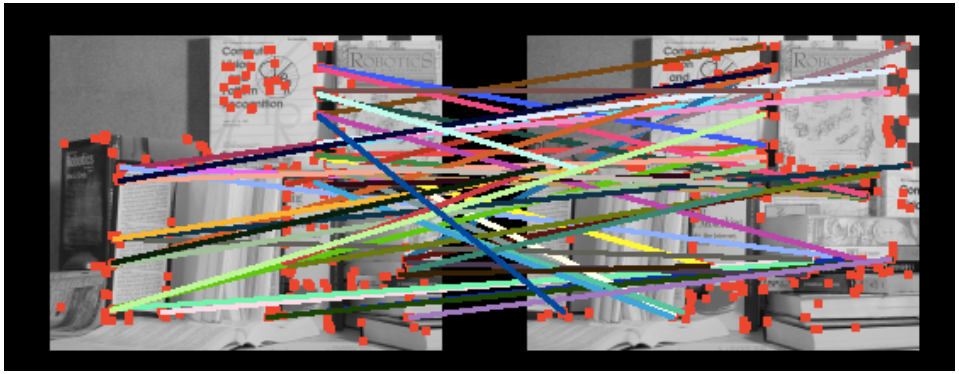
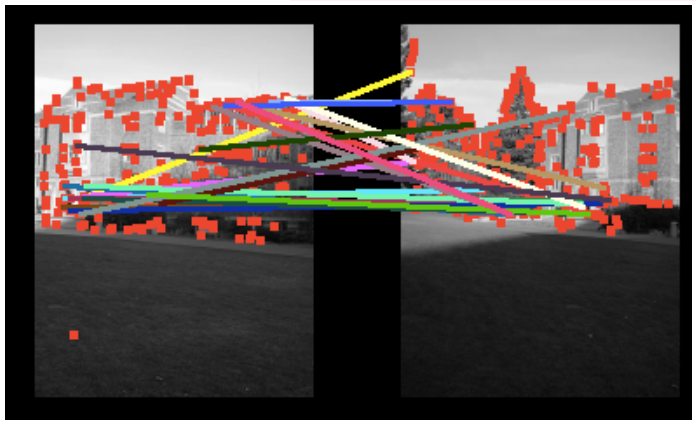
snapshots from ORB features and descriptor matching followed by RANSAC

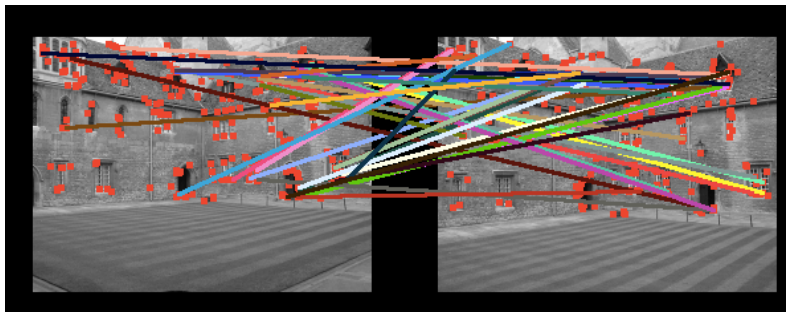
TODO: following up on use of RANSAC. Some of these have > 50% outliers.

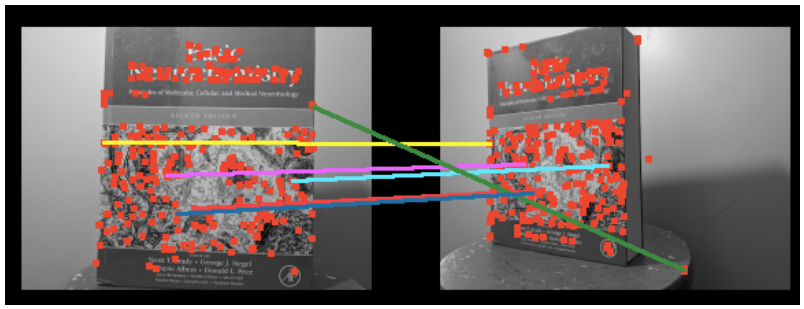




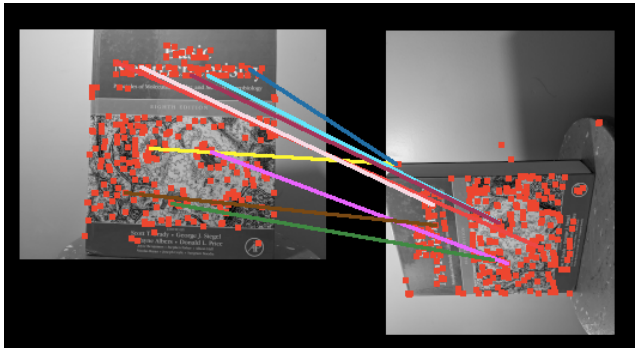


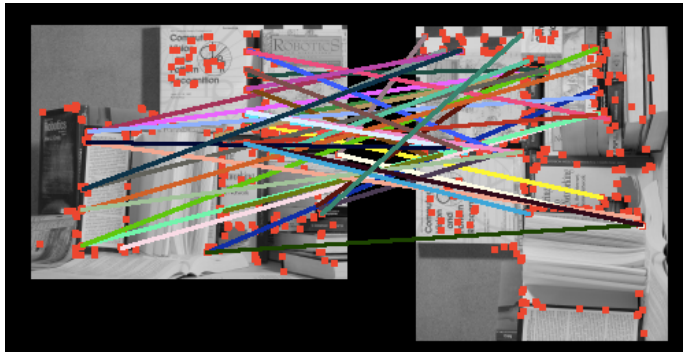


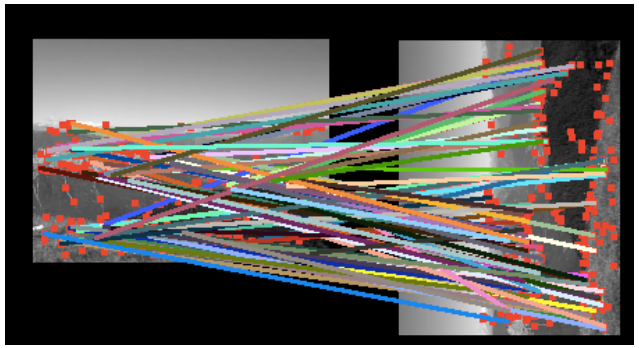


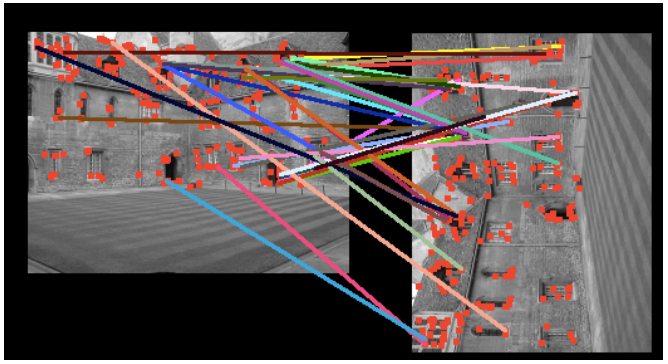


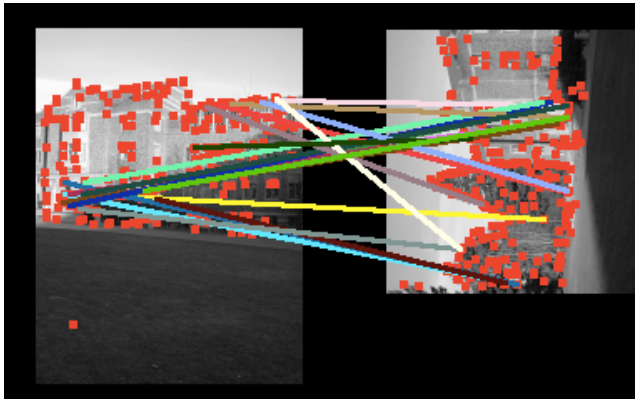
NOTE: alternatively, the ObjectMatcher tends to perform better than the ORBMatcher.
It uses MSER and LCH color space to find objects. HOGs matching of those areas performs well even with rotation.

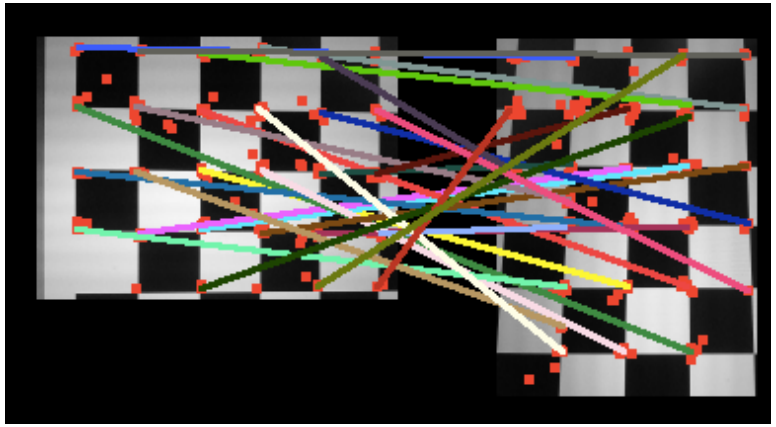












TODO: following up on use of RANSAC.
Some of these have > 50% outliers.

NOTE: alternatively, the ObjectMatcher tends to perform better than the ORBMatcher.
It uses MSER and LCH color space to find objects. HOGs matching of those areas performs well even with rotation.

ORB descriptor matching for nMatches < 7:

(1) need to remove outliers
(2) if can create a good homography, can find more matches by applying it to points in first image and finding closest projections in 2nd image.
(3) if have > 7 correspondence at this stage, can let the ORB matching method continue with use of RANSAC to fir for epipolar geometry.

——

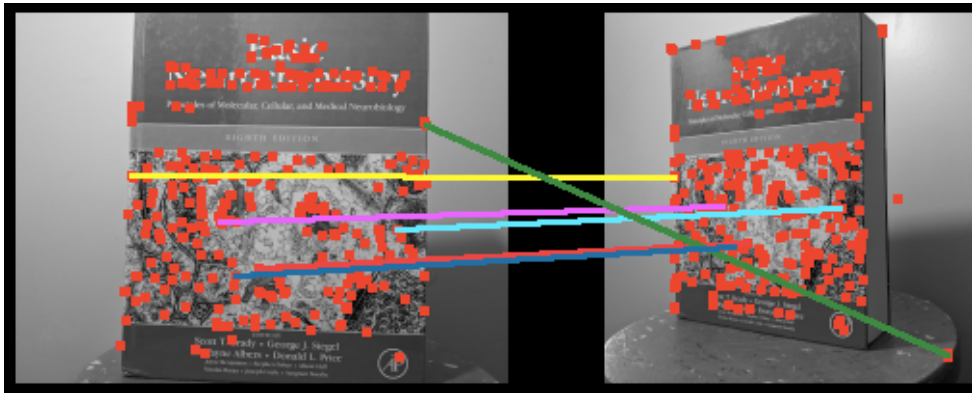
(1) removing outliers:
can use affine projection.
can iterate over subsamples of the input point to remove points from it,
fit and evaluate the affine projection
apply the best affine projection to keypoints1 to fins close matches
in keypoints2 where close match is (x,y) and descriptor cost.
the projection algorithm solves for rotation and real world scene coordinates.
It does not solve for translation,
but one could estimate a rough difference, (not the camera translation
in the camera reference frame coords) if the image scales are the
same by subtracting the 2nd image correspondence points from the
1st image correspondence points rotated.

OrthographicProjectionResults re = Reconstruction.calculateAffineReconstruction(
double[][] x, int mImages).

where OrthographicProjectionResults results = new OrthographicProjectionResults();
results.XW = s;
results.rotationMatrices = rotStack;

Considering the paper
https://www.researchgate.net/publication/221110532_Outlier_Correction_in_Image_Sequences_for_the_Affine_Camera
"Outlier Correction in Image Sequences for the Affine Camera"
by Huartley, and Heydeon 2003
Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV'03)

excerpt from the abstract:
In this paper, we present an outlier correction scheme that
iteratively updates the elements of the image measurement matrix



a quick look at using DLT to solve for simplest homography to get a reprojection error that might be good enough to remove large deviation outliers.

Plotted is the ORB descriptor matches.
correspondence set #4 is the green line that should be easy to see as a large deviation in the reproduction errors.

DLT to solve for homography:
p2 = A * p1 where p1 is (x1, y1) and p2 is (x2, y2)
A = a00 a01, a10 a11

A = P2 * P1^T * (P!*P1^T)^-1

H = 0 -1
1 0 anti-symmetric matrix

alpha * P2 = A * P1

multiply both sides of equation from the left by (P2)^T*H
(P2)^T*H * alpha * P2 = (P2)^T*H * A * P1
note that (P2)^T*H*P2 = 0
then have (P2)^T*H*A*P1 = 0

[x2 y2] * [0 -1] * [a00 a01] * [x1] = 0
[1 0] [a10 a11] [y1]

[y2 -x2] * [a00 a01] * [x1] = 0
[a10 a11] [y1]

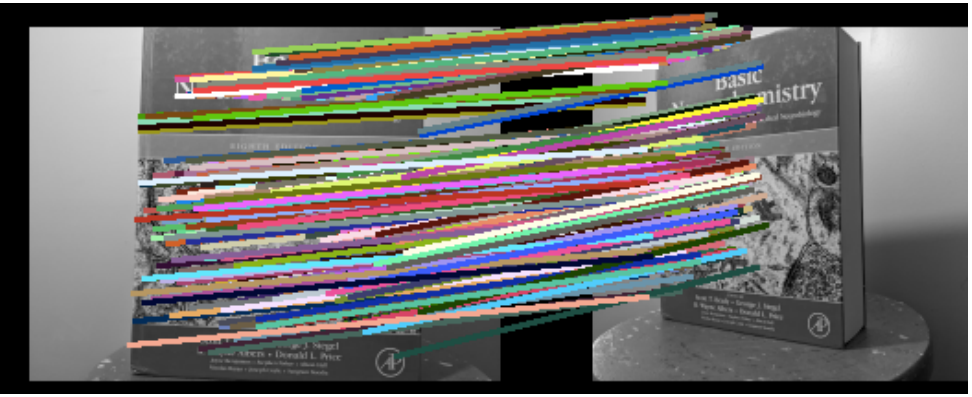
```
[ y2*a00 - x2*a10   y2*a01 - x2*a11 ] * [x1] = 0
                                        [y1]

  a00*y2*x1 - a10*x2*x1 + a01*y2*y1 - a11*x2*y1 = 0

  can be written as 0 = (b)^T * a where b and a are vectors
   b = (y2*x1)  a=(a00)
       (y2*y1)    (a01)
       (-x2*x1)   (a10)
       (-x2*y1)   (a11)

reprojection error

  y2*(a00*x1+a01*y1) + x2*(-a10*x1-a11*y1) = 0
  y2*(a00*x1+a01*y1) = x2*(a10*x1+a11*y1)
  x2/y2 = ( (a00*x1+a01*y1)/(a10*x1+a11*y1) )
```



Plotted is the rough homography reproductions

0 (x1,y1), (x2,y2) = (60,84), (36,85)
xhat_0=3.174e-01, 1.549e-01, -4.686e-02, 9.344e-01
reprojection error=116.000

1 (x1,y1), (x2,y2) = (198,112), (122,101)
xhat_1=3.792e-01, 7.657e-02, -1.635e-01, 9.075e-01
reprojection error=2545.000

2 (x1,y1), (x2,y2) = (106,108), (61,100)
xhat_2=3.717e-01, 1.417e-01, -8.484e-02, 9.136e-01
reprojection error=170.000

3 (x1,y1), (x2,y2) = (126,132), (74,119)
xhat_3=3.820e-01, 1.479e-01, -8.780e-02, 9.080e-01
reprojection error=170.000

4 (x1,y1), (x2,y2) = (213,58), (178,177)
xhat_4=1.863e-01, 2.054e-02, -7.586e-02, 9.793e-01
reprojection error=37813.000

5 (x1,y1), (x2,y2) = (114,136), (67,121)
xhat_5=3.712e-01, 1.593e-01, -7.396e-02, 9.118e-01
reprojection error=52.000

xhat_all=-4.513e-01, 4.895e-01, -4.735e-01, 5.767e-01
reprojection error all/n=6811.000

xhat_w/o#4=4.638e-01, 4.491e-02, -9.300e-02, 8.799e-01
reprojection error w/o#4/n=610.600

so can see that the simple DLT homography can be used to remove outliers but should not be retained as a valid transformation between the cameras.