imge1 = cv.imread('notredom\_test.jpg')

img1 = cv.cvtColor(imge1, cv.COLOR\_BGR2GRAY) # queryImage

imge2 = cv.imread('notredom\_train.jpg')

img2 = cv.cvtColor(imge2, cv.COLOR\_BGR2GRAY) # trainImage

# creating SIFT feature detector object, (Scale-Invariant Feature Transform)

# https://docs.opencv.org/master/da/df5/tutorial\_py\_sift\_intro.html

sift = cv.xfeatures2d.SIFT\_create()

# find the keypoints and descriptors with SIFT

kp1, des1 = sift.detectAndCompute(img1,None)

kp2, des2 = sift.detectAndCompute(img2,None)

# BFMatcher with default params

bf = cv.BFMatcher()

matches = bf.knnMatch(des1,des2,k=2)

# Apply ratio test

good = []

for m,n in matches:

if m.distance < 0.75\*n.distance:

good.append([m])

# cv.drawMatchesKnn expects list of lists as matches.

img3 = cv.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=cv.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

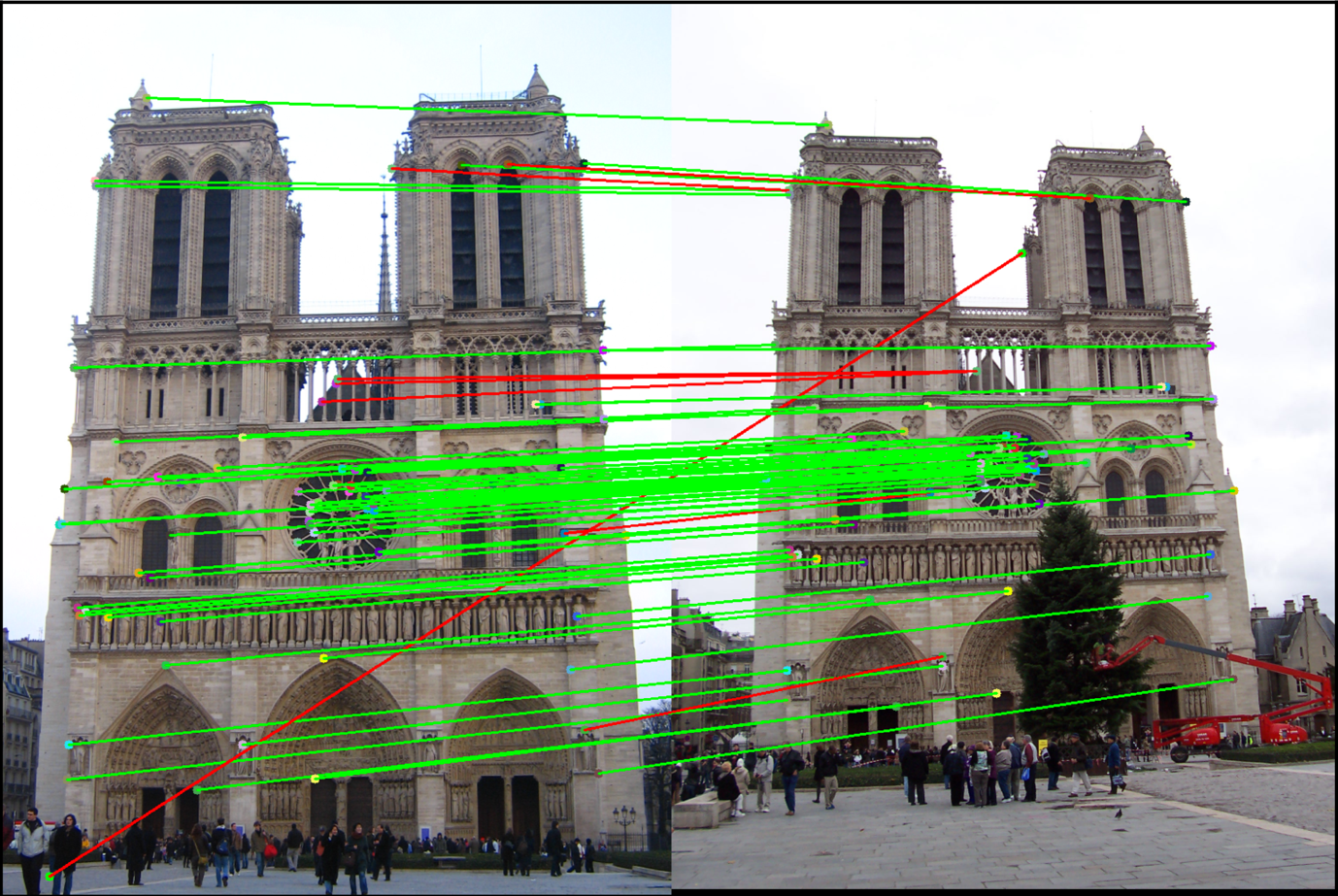
plt.imshow(img3),plt.show()

Some Definitions

1. cv.xfeatures2d.SIFT\_create() = Creating SIFT Object which is in xfeatures2d module. SIFT = Scale Invariant Feature Transform, It consists of various modules: Scale Space Extrema Detection, Keypoint Localization, Orientation Assignment, Keypoint Descriptor, Keypoint Matching.
2. Sift.DetectAndCompute = Detect and computes the Descriptors from the keypoints.
3. BFMatcher = BruteForce Descriptor matcher. For each descriptor in the first set, this matcher finds the closest descriptor in the second set by trying each one. It has KNN algorithm residing in it which actually performs the matches in both the images. Distance taken is 0.75

Storing all the points in the good array and then applying them in the images. The implementation works like this:

1. First import the images and assign them variables
2. Convert them into grey vectors.
3. Use the xFeatures2D library and use SIFT, object for computing pixels. It transforms the image to features (Descriptors) and pixels (Key points). Extract keypoints and calculate descriptors. Keypoints are 2D points / pixels and Descriptors are features.
4. Using BF Matcher we import the algorithms libraries for performing descriptor matching. The points which are similar in descriptors have to be similar in k nearest neighbour.
5. KNN will result in the matching vectors. From train and test images. Once we check if test image descriptors have similarity 75% with train image then add to our good array.
6. And finally we plot those points in image.



Graphs G1 : {V1, E1} and G2 : {V2, E2} There is a mapper function which maps the node V of G1 to the nodes of V of G2 such that the matching or similarity of the nodes remain highest. G2=m(G1) where G2 is the test criteria and G1 is the training criteria. The idea is to evaluate all the mapper functions of such nodes, such that an overall mapping of both is found which results in a maximum similarity.