Feature matching and extraction AKAZE

In [3]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving IX-11-01917_0004_0001.JPG to IX-11-01917_0004_0001.JPG
Saving IX-11-01917_0004_0002.JPG to IX-11-01917_0004_0002.JPG
Saving IX-11-01917_0004_0003.JPG to IX-11-01917_0004_0003.JPG
Saving IX-11-01917_0004_0004.JPG to IX-11-01917_0004_0004.JPG
Saving IX-11-01917_0004_0005.JPG to IX-11-01917_0004_0005.JPG
Saving IX-11-01917_0004_0006.JPG to IX-11-01917_0004_0006.JPG
Saving IX-11-01917_0004_0007.JPG to IX-11-01917_0004_0007.JPG
Saving IX-11-01917_0004_0008.JPG to IX-11-01917_0004_0008.JPG
Saving IX-11-01917_0004_0009.JPG to IX-11-01917_0004_0009.JPG
Saving IX-11-01917_0004_0010.JPG to IX-11-01917_0004_0010.JPG
```

In [5]:
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

# Open and convert the input and training-set image from BGR to GRAYSCALE
image1 = cv.imread(filename = 'IX-11-01917_0004_0009.JPG',
                   flags = cv.IMREAD_GRAYSCALE)

image2 = cv.imread(filename = 'IX-11-01917_0004_0010.JPG',
                   flags = cv.IMREAD_GRAYSCALE)
```

In [6]:
```
# Initiate A-KAZE descriptor
AKAZE = cv.AKAZE_create()

# Find the keypoints and compute the descriptors for input and training-set image
keypoints1, descriptors1 = AKAZE.detectAndCompute(image1, None)
keypoints2, descriptors2 = AKAZE.detectAndCompute(image2, None)
```

In [7]:
```
FLANN_INDEX_KDTREE = 1
```

```
index_params = dict(algorithm = FLANN_INDEX_KDTREE,
                    trees = 5)

search_params = dict(checks = 50)

# Convert to float32
descriptors1 = np.float32(descriptors1)
descriptors2 = np.float32(descriptors2)

# Create FLANN object
FLANN = cv.FlannBasedMatcher(indexParams = index_params,
                             searchParams = search_params)

# Matching descriptor vectors using FLANN Matcher
matches = FLANN.knnMatch(queryDescriptors = descriptors1,
                         trainDescriptors = descriptors2,
                         k = 2)

# Lowe's ratio test
ratio_thresh = 0.7

# "Good" matches
good_matches = []

# Filter matches
for m, n in matches:
    if m.distance < ratio_thresh * n.distance:
        good_matches.append(m)

# Draw only "good" matches
output = cv.drawMatches(img1 = image1,
                        keypoints1 = keypoints1,
                        img2 = image2,
                        keypoints2 = keypoints2,
                        matches1to2 = good_matches,
                        outImg = None,
                        flags = cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

plt.imshow(output)
plt.show()
```