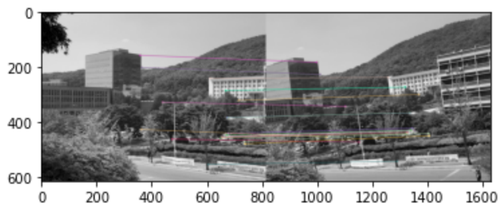
실습2

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

img1 = cv.imread('b1.png',cv.IMREAD_GRAYSCALE)  # queryImage
img2 = cv.imread('b2.png',cv.IMREAD_GRAYSCALE)  # trainImage
# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
```

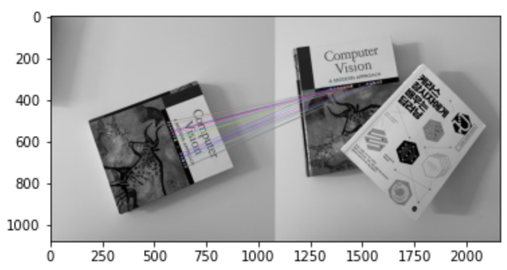
In [2]:
```
# create BFMatcher object
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
# Match descriptors.
matches = bf.match(des1,des2)
# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)
# Draw first 30 matches.
img3 = cv.drawMatches(img1,kp1,img2,kp2,matches[:30],None,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(img3,'gray')
plt.show()
```



실습3

In [3]:
```
import cv2 as cv
img1 = cv.imread('p1.jpg',cv.IMREAD_GRAYSCALE)  # queryImage
img2 = cv.imread('p2.jpg',cv.IMREAD_GRAYSCALE)  # trainImage
# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
```

In [4]:
```
# create BFMatcher object
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
# Match descriptors.
matches = bf.match(des1,des2)
# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)
# Draw first 30 matches.
img3 = cv.drawMatches(img1,kp1,img2,kp2,matches[:30],None,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(img3,'gray')
plt.show()
```



과제

In [5]:
```
pic1 = cv.imread('pic1.jpg',cv.IMREAD_GRAYSCALE)  # queryImage
pic2 = cv.imread('pic2.jpg',cv.IMREAD_GRAYSCALE)        # trainImage
# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(pic1,None)
kp2, des2 = orb.detectAndCompute(pic2,None)
```

In [6]:
```
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
# Match descriptors.
matches = bf.match(des1,des2)
# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)
# Draw first 30 matches.
pic3 = cv.drawMatches(pic1,kp1,pic2,kp2,matches[:50],None,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(pic3,'gray')
plt.show()
```