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

In [2]:
```
def load(path):
    img=cv.imread(path)
    #opencv reads the image in BGR, thus we have to turn it to RGB
    img=cv.cvtColor(img,cv.COLOR_BGR2RGB)
    return img
```

In [3]:
```
def display(img1,cmap="gray"):
    fig=plt.figure(figsize=(12,12))
    ax=fig.add_subplot()
    ax.imshow(img1,cmap="gray")
```

In [5]:
```
# Create SIFT Object
sift = cv.xfeatures2d.SIFT_create()
MIN_MATCH_COUNT=10

query=cv.imread("/Users/mehradhq/Computer_Vision/Research_2/dataset/train/Prohibition_Signs/12.jpeg")
query=cv.cvtColor(query, cv.COLOR_BGR2RGB)

target=cv.imread("/Users/mehradhq/Computer_Vision/Research_2/dataset/train/Prohibition_Signs/17.jpeg")
target=cv.cvtColor(target, cv.COLOR_BGR2RGB)

# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(query,None)
kp2, des2 = sift.detectAndCompute(target,None)

# BFMatcher with default params
bf = cv.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

# Apply ratio test
good = []
for match1,match2 in matches:
    if match1.distance < 0.75*match2.distance:
        good.append([match1])

good = []
for match1,match2 in matches:
    if match1.distance < 0.9*match2.distance:
        good.append([match1])

# cv2.drawMatchesKnn expects list of lists as matches.
sift_matches = cv.drawMatchesKnn(query,kp1, target,kp2,good,None,flags=2)
display(sift_matches)
```

In [6]:

```
# Create SIFT Object
sift = cv.xfeatures2d.SIFT_create()

#query image
query=cv.imread("/Users/mehradhq/Computer_Vision/Research_2/dataset/train/Prohibition_Signs/38.jpeg")
query=cv.cvtColor(query, cv.COLOR_BGR2RGB)

#target image
target=cv.imread("/Users/mehradhq/Computer_Vision/Research_2/dataset/train/Prohibition_Signs/41.jpeg")
target=cv.cvtColor(target, cv.COLOR_BGR2RGB)

# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(query,None)
kp2, des2 = sift.detectAndCompute(target,None)

# BFMatcher with default params
bf = cv.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

# Apply ratio test
good = []
for match1,match2 in matches:
    if match1.distance < 0.9*match2.distance:
        good.append([match1])
```

```
# cv2.drawMatchesKnn expects list of lists as matches.
sift_matches = cv.drawMatchesKnn(query,kp1, target,kp2,good,None,flags=2)
display(sift_matches)
```