In [24]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from skimage.io import imread, imshow
```

In [25]:
```
img1=imread('EP-00-00012_0119_0001.JPG')
plt.figure(figsize=(7,7))
imshow(img1)
img2=imread('EP-00-00012_0119_0001.JPG')
plt.figure(figsize=(7,7))
imshow(img2)
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x1b002081a20>
```

In [26]:
```
# Convert it to grayscale
grey_img1_bw = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(7,7))
imshow(grey_img1_bw)
grey_img2_bw = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(7,7))
imshow(grey_img2_bw)
```

Out[26]:
```
<matplotlib.image.AxesImage at 0x1b0020e7c18>
```

In [27]:
```
# Ini
orb
```

In [28]:
```
img1_
img2_
```

In [29]:
```
img_1_keyp = cv2.drawKeypoints(img1, img1_keyp, img1, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS, color=(0,255,0)) # Draw circles.
plt.figure(figsize=(16, 16))
plt.title('ORB Interest Points')
plt.imshow(img_1_keyp);
plt.show()
```

ORB Interest Points

Brute-Force Matching with ORB Descriptors

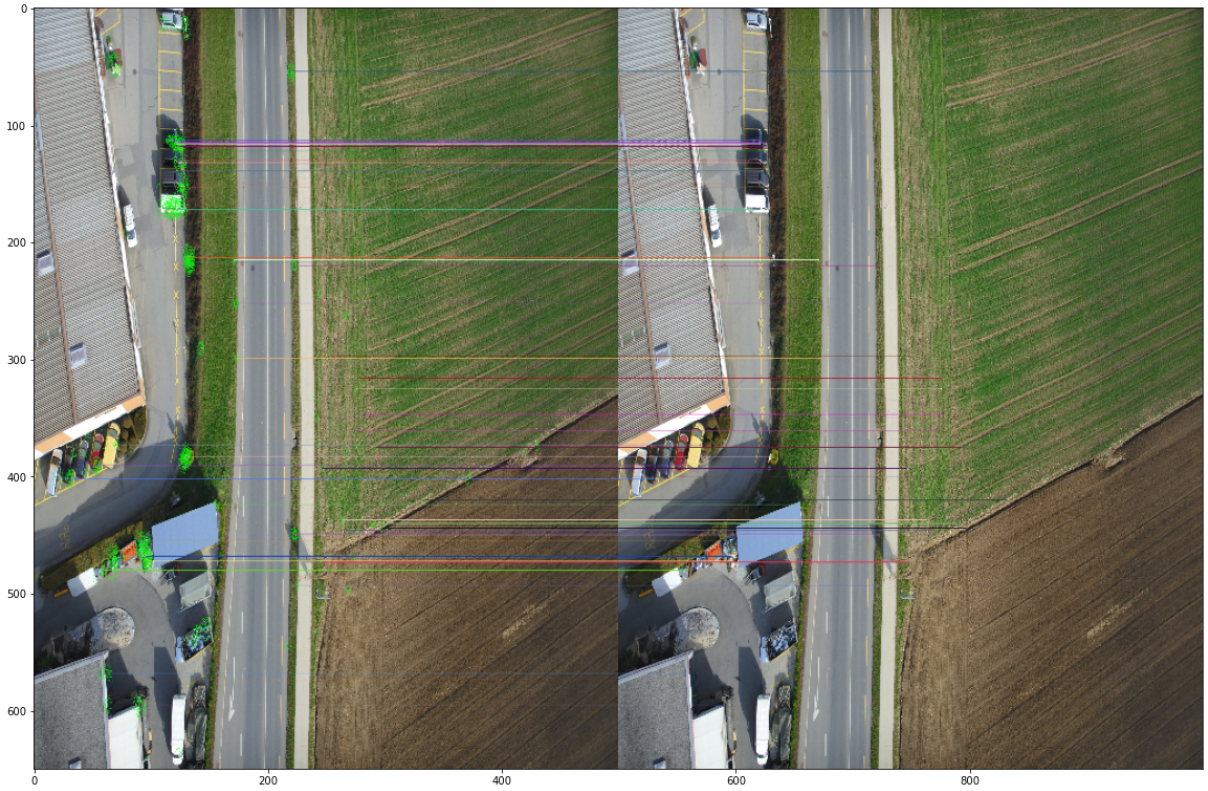
In [30]:
```
matcher = cv2.BFMatcher()
matches = matcher.match(img1_Desp,img2_Desp)
```

In [31]:
```
final_img = cv2.drawMatches(img1, img1_Keyp, img2, img2_Keyp, matches[:
100], None)
```

In [32]:
```
final_img = cv2.resize(final_img, (1000,650))
```

In [33]:
```
plt.figure(figsize=(20,20))
plt.imshow(final_img)
```

Out[33]: <matplotlib.figure.AxesImage at 0x1b002157f60>



In [34]:
```
img_1 = imread('EP-00-00012_0119_0001.JPG')
img_2 = imread('EP-00-00012_0119_0001.JPG')
```

In [35]:
```
# Convert it to grayscale
greyimg1 = cv2.cvtColor(img_1,cv2.COLOR_BGR2GRAY)
greyimg2 = cv2.cvtColor(img_2, cv2.COLOR_BGR2GRAY)
```

In [36]:
```
plt.figure(figsize=(20,20))
plt.subplot(121)
imshow(img_1)
plt.subplot(122)
imshow(greyimg1)
```

Out[36]: <matplotlib.image.AxesImage at 0x1b0021290f0>

In [37]:
```
plt.figure(figsize=(20,20))
plt.subplot(121)
imshow(img_2)
plt.subplot(122)
imshow(greyimg2)
```

Out[37]: <matplotlib.image.AxesImage at 0x1b001d99da0>

In [38]:
```
img1Keyp, img1Desc = orb.detectAndCompute(greyimg1,None)
img2Keyp, img2Desc = orb.detectAndCompute(greyimg2,None)
```

In [39]:
```
a_keyp = cv2.drawKeypoints(img_1, img1_Keyp, img_1,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS, color=(0,255,0)) # Draw circles.
plt.figure(figsize=(16, 16))
plt.title('ORB Interest Points')
plt.imshow(a_keyp); plt.show()
```

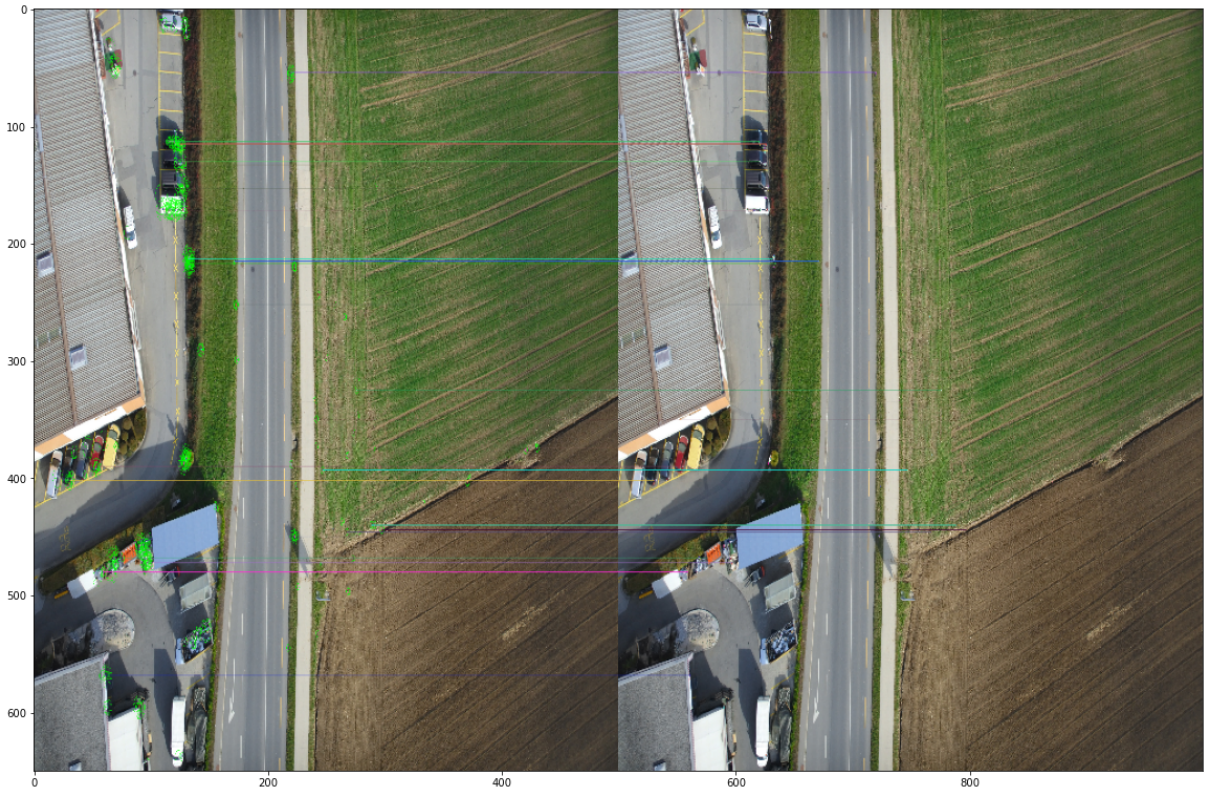
In [40]:
```
matcher = cv2.BFMatcher()
matches = matcher.match(img1Desc,img2Desc)
```

In [42]:
```
final_image = cv2.drawMatches(img_1, img1_Keyp, img_2, img2_Keyp, matches[:50], None)
```

In [43]:
```
final_image = cv2.resize(final_image, (1000,650))
```

In [44]:
```
plt.figure(figsize=(20,20))
plt.imshow(final_image)
```

Out[44]: <matplotlib.image.AxesImage at 0x1b0026c70f0>



https://towardsdatascience.com/image-feature-extraction-using-pytorch-e3b327c3607a

https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774