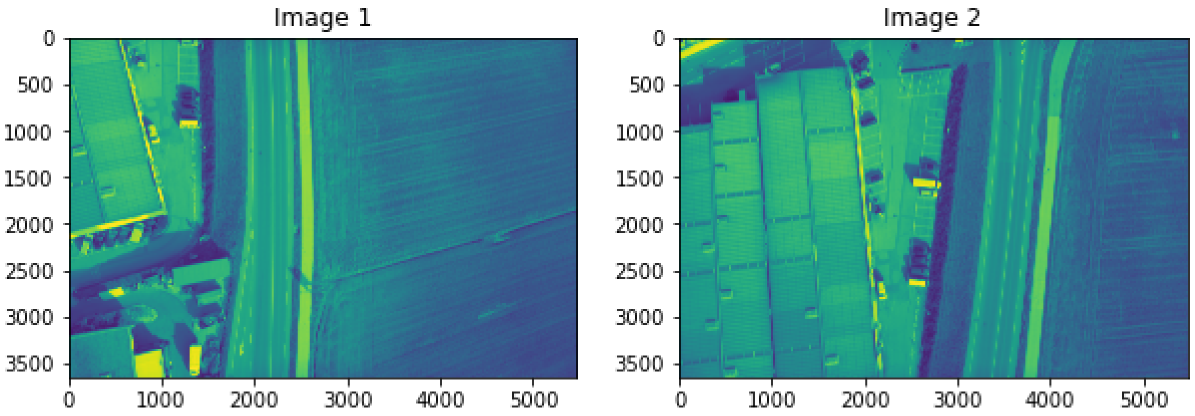
In [1]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```

```
c:\users\saloni\appdata\local\programs\python\python37\lib\site-package
s\numpy\_distributor_init.py:32: UserWarning: loaded more than 1 DLL fr
om .libs:
c:\users\saloni\appdata\local\programs\python\python37\lib\site-package
s\numpy\.libs\libopenblas.GK7GX5KEQ4F6UYO3P26ULGBQYHGQO7J4.gfortran-win
_amd64.dll
c:\users\saloni\appdata\local\programs\python\python37\lib\site-package
s\numpy\.libs\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win
_amd64.dll
  stacklevel=1)
```

In [2]:
```
img_1 = cv.imread('EP-00-00012_0119_0001.JPG', cv.IMREAD_GRAYSCALE)
img_2 = cv.imread('EP-00-00012_0119_0002.JPG', cv.IMREAD_GRAYSCALE)
```

In [3]:
```
plt.figure(figsize=[10,5])
plt.subplot(1,2,1)
plt.title('Image 1')
plt.imshow(img_1)
plt.subplot(1,2,2)
plt.imshow(img_2)
plt.title('Image 2')
```

Out[3]:
```
Text(0.5, 1.0, 'Image 2')
```



In [4]:

```
# Initiate AKAZE detector
akaze = cv.AKAZE_create()
```

In [5]:

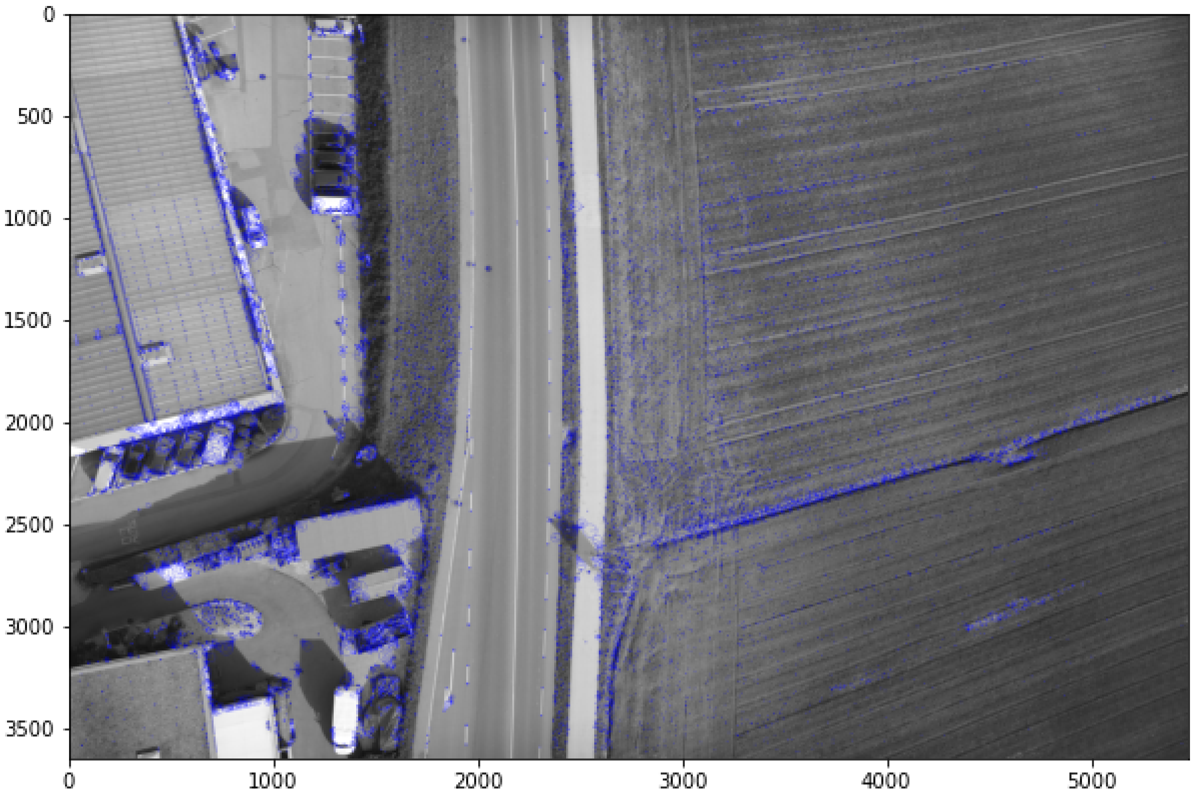
```
# Find the keypoints and descriptors with SIFT
kp1, des1 = akaze.detectAndCompute(img_1, None)
kp2, des2 = akaze.detectAndCompute(img_2, None)
```

In [6]:

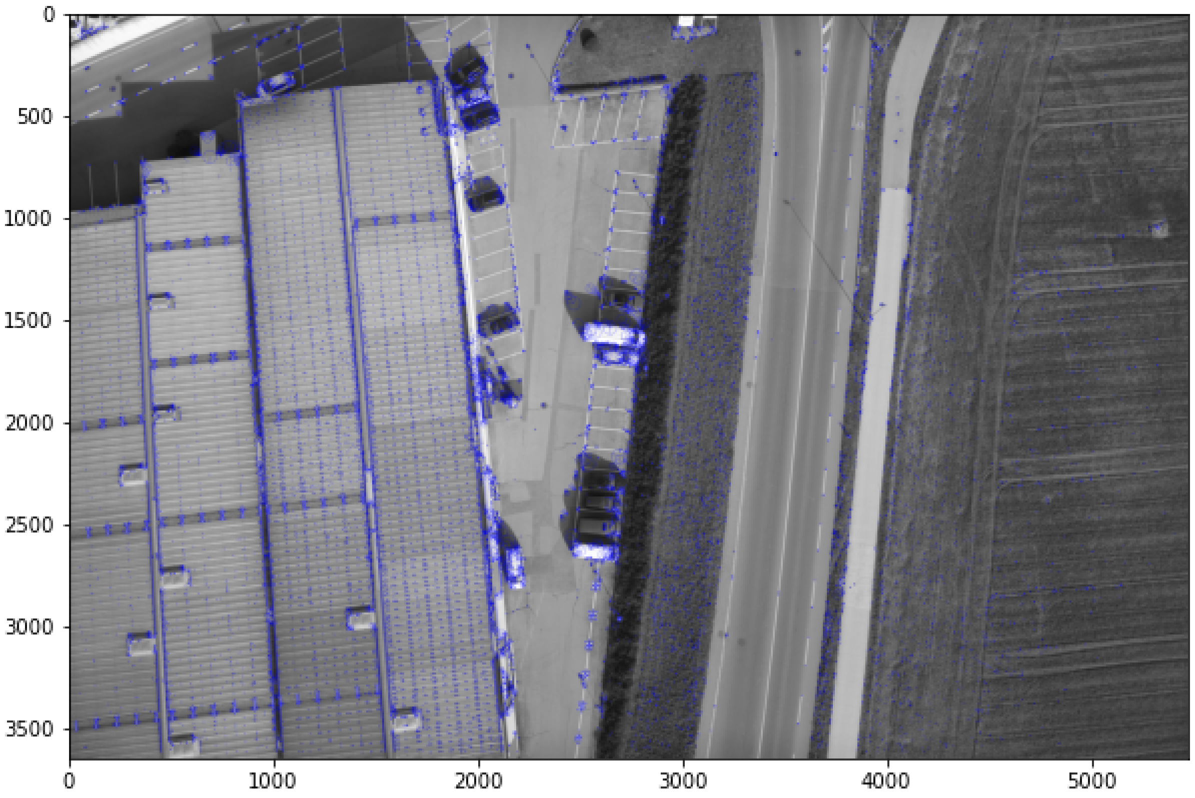
```
img1 = cv.imread('EP-00-00012_0119_0001.JPG')
img2 = cv.imread('EP-00-00012_0119_0002.JPG')
```

In [7]:

```
img1_key=cv.drawKeypoints(img_1, kp1,img1,(0, 0, 255),cv.DRAW_MATCHES_F
LAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite('keypoints.jpg', img1_key)
plt.figure(figsize=(10,10))
plt.imshow(img1_key)
plt.show()
```



In [8]:
```
img2_key=cv.drawKeypoints(img_2, kp2,img2,(0, 0, 255),cv.DRAW_MATCHES_F
LAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite('keypoints2.jpg', img2_key)
plt.figure(figsize=(10,10))
plt.imshow(img2_key)
plt.show()
```



In [9]:

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

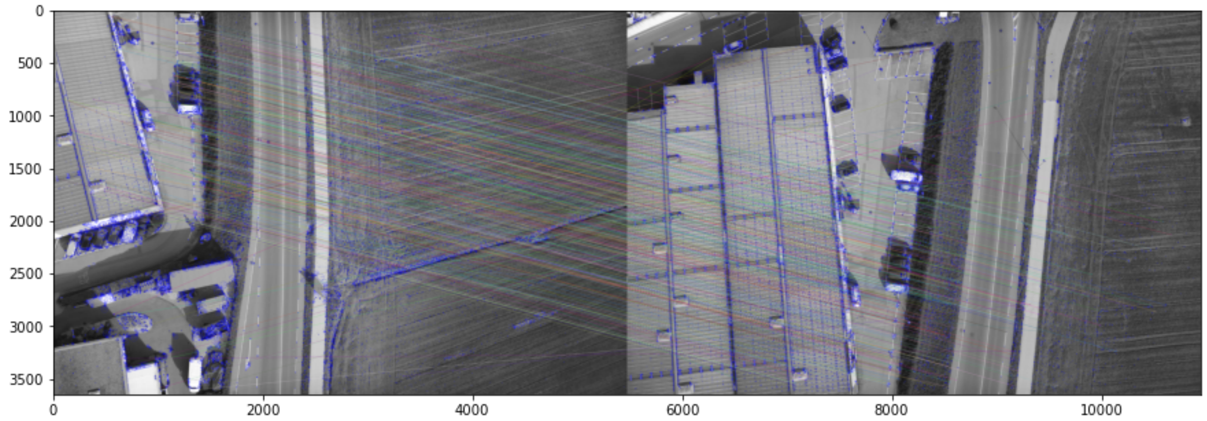
In [11]:

```
# Apply ratio test
good_matches = []
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good_matches.append([m])
```

In [12]:

```
# Draw matches
img4 = cv.drawMatchesKnn(img1,kp1,img2,kp2,good_matches,None,flags=cv.D
rawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

```
cv.imwrite('matches.jpg', img4)
plt.figure(figsize=(15,15))
plt.imshow(img4)
plt.show()
```



In [13]:

```
# Select good matched keypoints
ref_matched_kpts = np.float32([kp1[m[0].queryIdx].pt for m in good_matc
hes]).reshape(-1,1,2)
sensed_matched_kpts = np.float32([kp2[m[0].trainIdx].pt for m in good_m
atches]).reshape(-1,1,2)
```

In [14]:

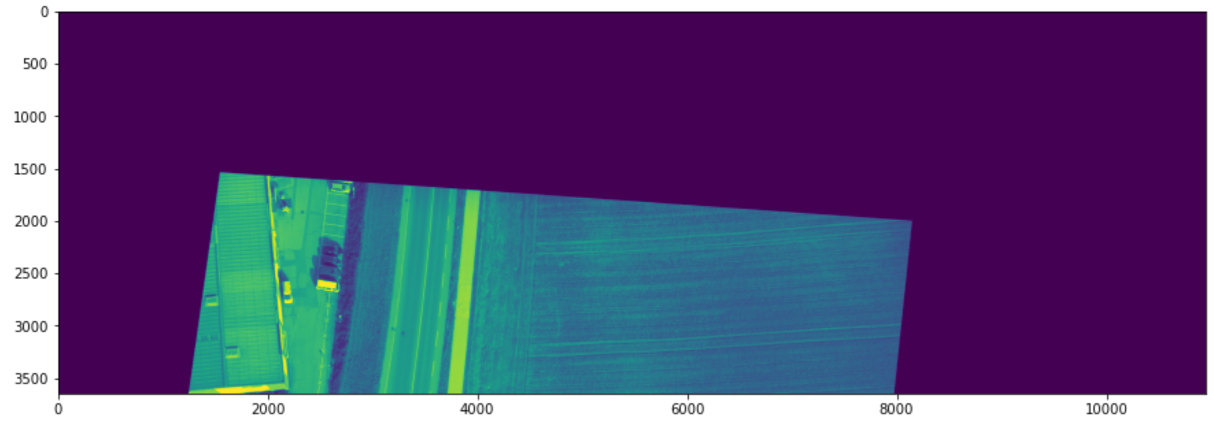
```
# Compute homography
H, status = cv.findHomography(ref_matched_kpts, sensed_matched_kpts, cv
.RANSAC,5.0)
```

In [15]:

```
# Warp image
h=img_1.shape[1]+img_2.shape[1]
w=img_1.shape[0]
warped_image = cv.warpPerspective(img_1, H, (h,w))
```

In [16]:

```
cv.imwrite('warped.jpg', warped_image)
plt.figure(figsize=(15,15))
plt.imshow(warped_image)
plt.show()
```

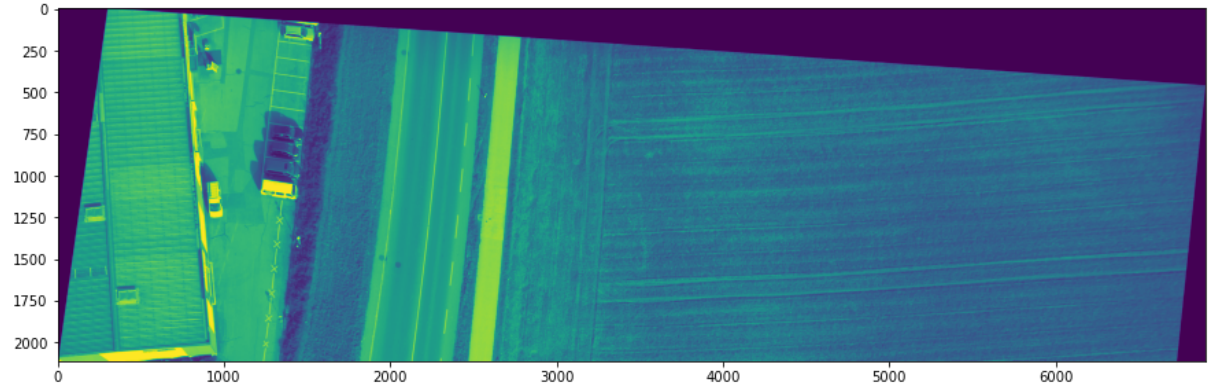


In [18]:

```python
def crop(img):
    y,x=np.nonzero(img)
    return img[np.min(y):np.max(y),np.min(x):np.max(x)]
```

In [19]:

```python
cv.imwrite('cropped.jpg', crop(warped_image))
plt.figure(figsize=(15,15))
plt.imshow(crop(warped_image))
plt.show()
```

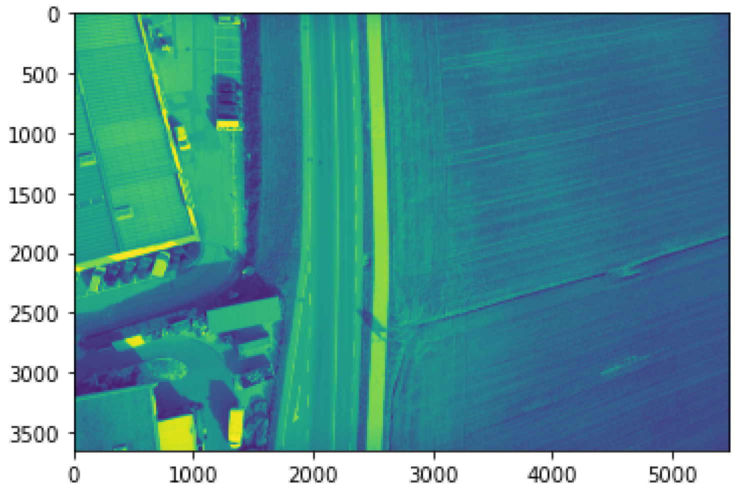


In [23]:

```python
img_3 = cv.imread('EP-00-00012_0119_0001.JPG', cv.IMREAD_GRAYSCALE)
img3 = cv.imread('EP-00-00012_0119_0002.JPG')
```

```
img_4=cv.imread('cropped.jpg')
plt.imshow(img_3)
```

Out[23]: `<matplotlib.image.AxesImage at 0x2cd0407cb70>`



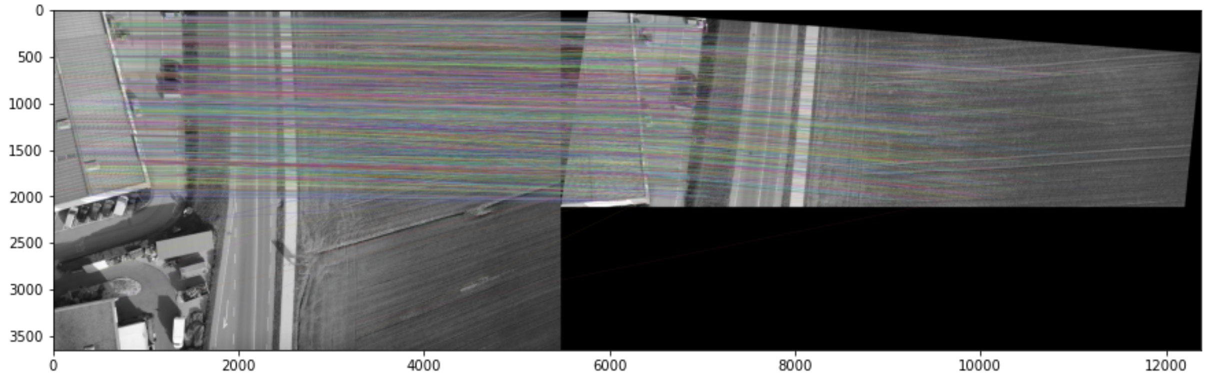
In [25]:
```
#Initiate AKAZE detector
akaze = cv.AKAZE_create()
# Find the keypoints and descriptors with SIFT
kp1, des1 = akaze.detectAndCompute(img_3, None)
kp2, des2 = akaze.detectAndCompute(img_4, None)
```

In [26]:
```
# BFMatcher with default params
bf = cv.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)
```

In [28]:
```
#Apply ratio test
good_matches1 = []
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good_matches1.append([m])
```

In [29]:
```
# Draw matches
```

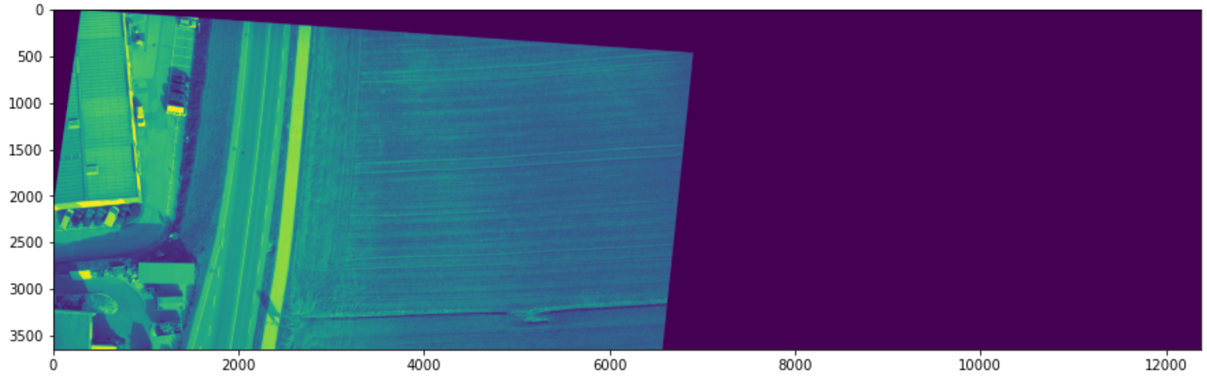
```
img5 = cv.drawMatchesKnn(img_3,kp1,img_4,kp2,good_matches1,None,flags=c
v.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
cv.imwrite('matches1.jpg', img5)
plt.figure(figsize=(15,15))
plt.imshow(img5)
plt.show()
```



In [30]:
```
# Select good matched keypoints
ref_matched_kpts = np.float32([kp1[m[0].queryIdx].pt for m in good_matc
hes1]).reshape(-1,1,2)
sensed_matched_kpts = np.float32([kp2[m[0].trainIdx].pt for m in good_m
atches1]).reshape(-1,1,2)
```

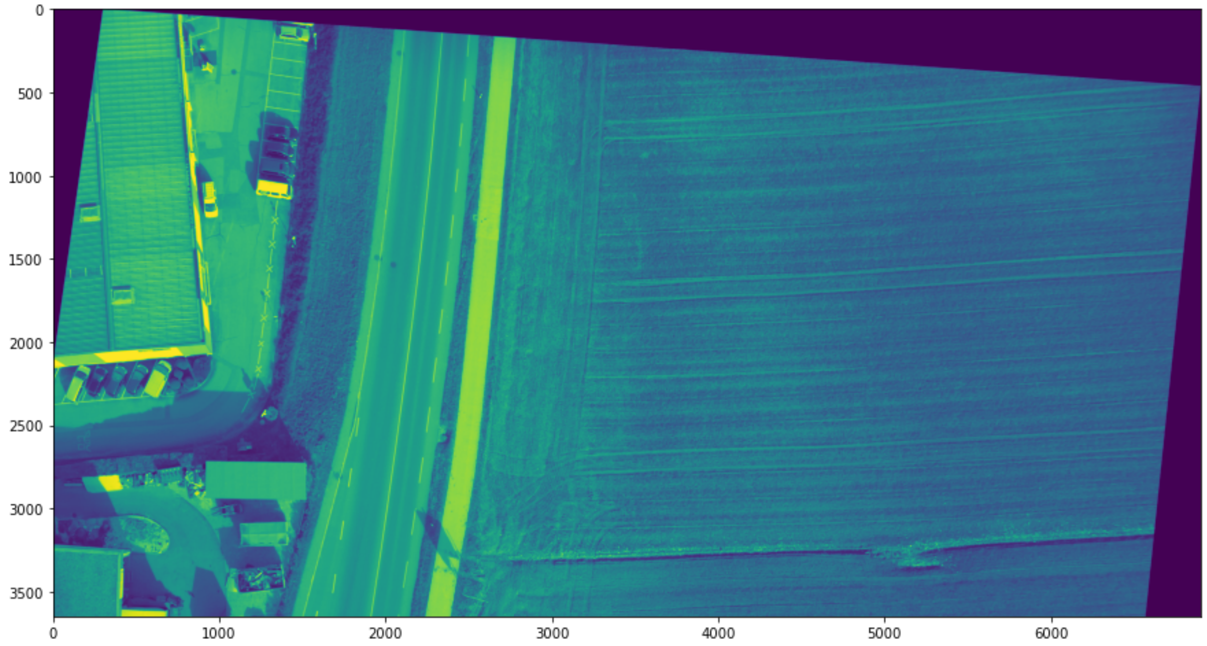
In [31]:
```
# Compute homography
H, status = cv.findHomography(ref_matched_kpts, sensed_matched_kpts, cv
.RANSAC,5.0)
```

In [32]:
```
# Warp image
h=img_3.shape[1]+img_4.shape[1]
w=img_3.shape[0]
warped_image1 = cv.warpPerspective(img_3, H, (h,w))
cv.imwrite('warped1.jpg', warped_image1)
plt.figure(figsize=(15,15))
plt.imshow(warped_image1)
plt.show()
```



In [33]:

```
cv.imwrite('cropped1.jpg', crop(warped_image1))
plt.figure(figsize=(15,15))
plt.show(crop(warped_image1))
plt.show()
```



In [ ]:

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```