In [5]:

```
import imutils
import numpy as np
import cv2

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [6]:

```
image_1 = mpimg.imread("../input/pond-dataset/DJI_0570.JPG")
image_2 = mpimg.imread("../input/pond-dataset/DJI_0571.JPG")
image_3 = mpimg.imread("../input/pond-dataset/DJI_0572.JPG")
image_4 = mpimg.imread("../input/pond-dataset/DJI_0573.JPG")
image_5 = mpimg.imread("../input/pond-dataset/DJI_0574.JPG")
image_6 = mpimg.imread("../input/pond-dataset/DJI_0575.JPG")
image_7 = mpimg.imread("../input/pond-dataset/DJI_0576.JPG")
image_8 = mpimg.imread("../input/pond-dataset/DJI_0577.JPG")
image_9 = mpimg.imread("../input/pond-dataset/DJI_0578.JPG")
image_10 = mpimg.imread("../input/pond-dataset/DJI_0579.JPG")
image_11 = mpimg.imread("../input/pond-dataset/DJI_0580.JPG")
image_12 = mpimg.imread("../input/pond-dataset/DJI_0581.JPG")
image_13 = mpimg.imread("../input/pond-dataset/DJI_0582.JPG")
image_14 = mpimg.imread("../input/pond-dataset/DJI_0583.JPG")
image_15 = mpimg.imread("../input/pond-dataset/DJI_0584.JPG")
image_16 = mpimg.imread("../input/pond-dataset/DJI_0585.JPG")
image_17 = mpimg.imread("../input/pond-dataset/DJI_0586.JPG")
image_18 = mpimg.imread("../input/pond-dataset/DJI_0587.JPG")
image_19 = mpimg.imread("../input/pond-dataset/DJI_0588.JPG")
image_20 = mpimg.imread("../input/pond-dataset/DJI_0589.JPG")
image_21 = mpimg.imread("../input/pond-dataset/DJI_0590.JPG")
image_22 = mpimg.imread("../input/pond-dataset/DJI_0591.JPG")
image_23 = mpimg.imread("../input/pond-dataset/DJI_0592.JPG")
image_24 = mpimg.imread("../input/pond-dataset/DJI_0593.JPG")
image_25 = mpimg.imread("../input/pond-dataset/DJI_0594.JPG")
image_26 = mpimg.imread("../input/pond-dataset/DJI_0595.JPG")
image_27 = mpimg.imread("../input/pond-dataset/DJI_0596.JPG")
image_28 = mpimg.imread("../input/pond-dataset/DJI_0597.JPG")
image_29 = mpimg.imread("../input/pond-dataset/DJI_0598.JPG")
image_30 = mpimg.imread("../input/pond-dataset/DJI_0599.JPG")
image_31 = mpimg.imread("../input/pond-dataset/DJI_0600.JPG")
image_32 = mpimg.imread("../input/pond-dataset/DJI_0601.JPG")
image_33 = mpimg.imread("../input/pond-dataset/DJI_0602.JPG")
image_34 = mpimg.imread("../input/pond-dataset/DJI_0603.JPG")
image_35 = mpimg.imread("../input/pond-dataset/DJI_0604.JPG")
image_36 = mpimg.imread("../input/pond-dataset/DJI_0605.JPG")
image_37 = mpimg.imread("../input/pond-dataset/DJI_0606.JPG")
```

```
image_38 = mpimg.imread("../input/pond-dataset/DJI_0607.JPG")
image_39 = mpimg.imread("../input/pond-dataset/DJI_0608.JPG")
image_40 = mpimg.imread("../input/pond-dataset/DJI_0609.JPG")
image_41 = mpimg.imread("../input/pond-dataset/DJI_0610.JPG")
image_42 = mpimg.imread("../input/pond-dataset/DJI_0611.JPG")
image_43 = mpimg.imread("../input/pond-dataset/DJI_0612.JPG")
image_44 = mpimg.imread("../input/pond-dataset/DJI_0613.JPG")
image_45 = mpimg.imread("../input/pond-dataset/DJI_0614.JPG")
image_46 = mpimg.imread("../input/pond-dataset/DJI_0615.JPG")
image_47 = mpimg.imread("../input/pond-dataset/DJI_0616.JPG")
image_48 = mpimg.imread("../input/pond-dataset/DJI_0617.JPG")
image_49 = mpimg.imread("../input/pond-dataset/DJI_0618.JPG")
image_50 = mpimg.imread("../input/pond-dataset/DJI_0619.JPG")
```

In [7]:

```
images = [image_1,image_5,image_9,image_13, image_17, image_21, image_24, image_8,
image_9, image_10, image_11, image_12, image_13, image_14, image_15, image_16,
image_17, image_18, image_19,image_20, image_21, image_22, image_23, image_24,
image_25, image_26, image_27, image_28, image_29, image_30, image_31, image_32,
image_33, image_34, image_35, image_36, image_37, image_38, image_39, image_40,
image_41, image_42, image_43, image_44, image_45, image_46, image_47, image_48,
image_49, image_50]
```

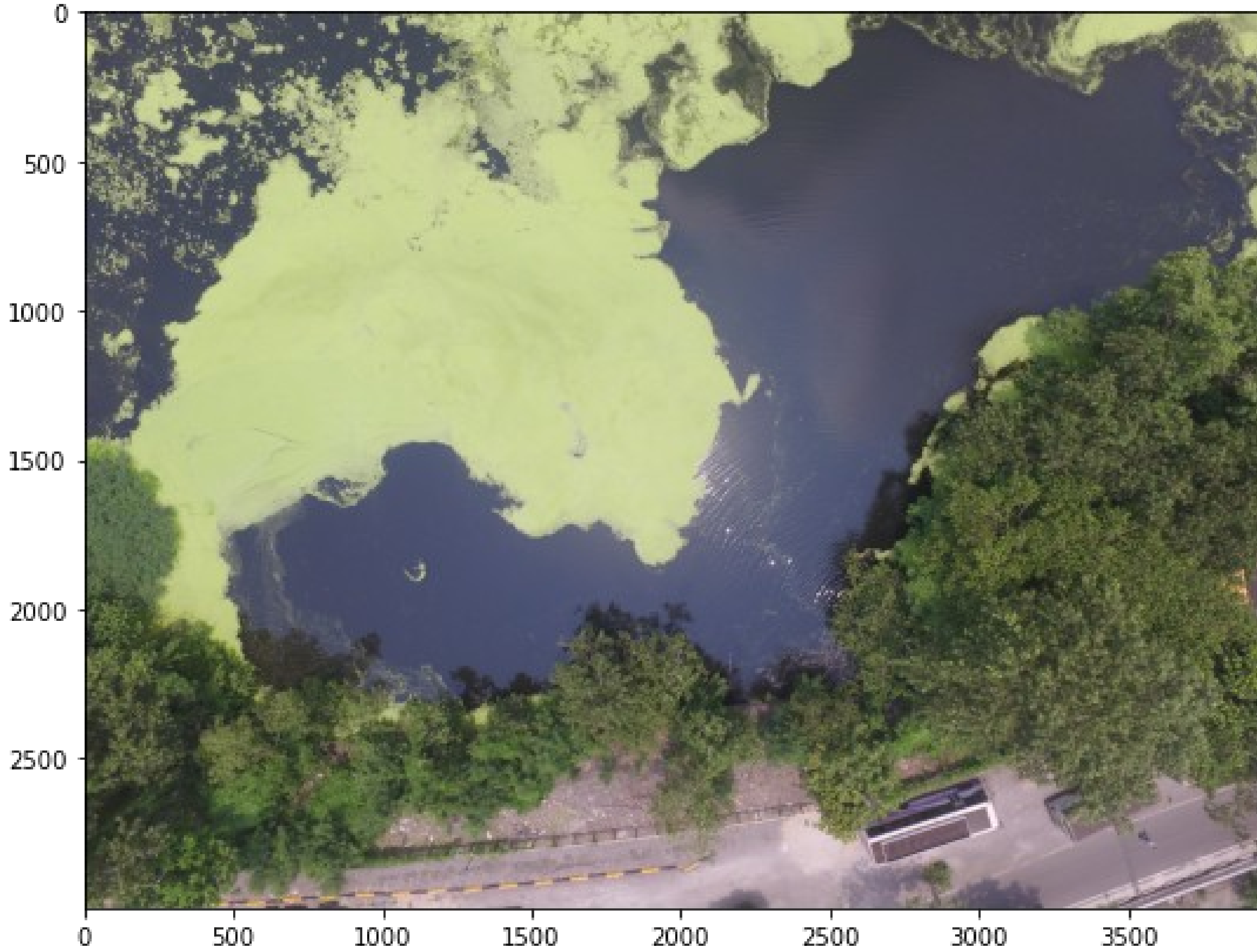
In [8]:

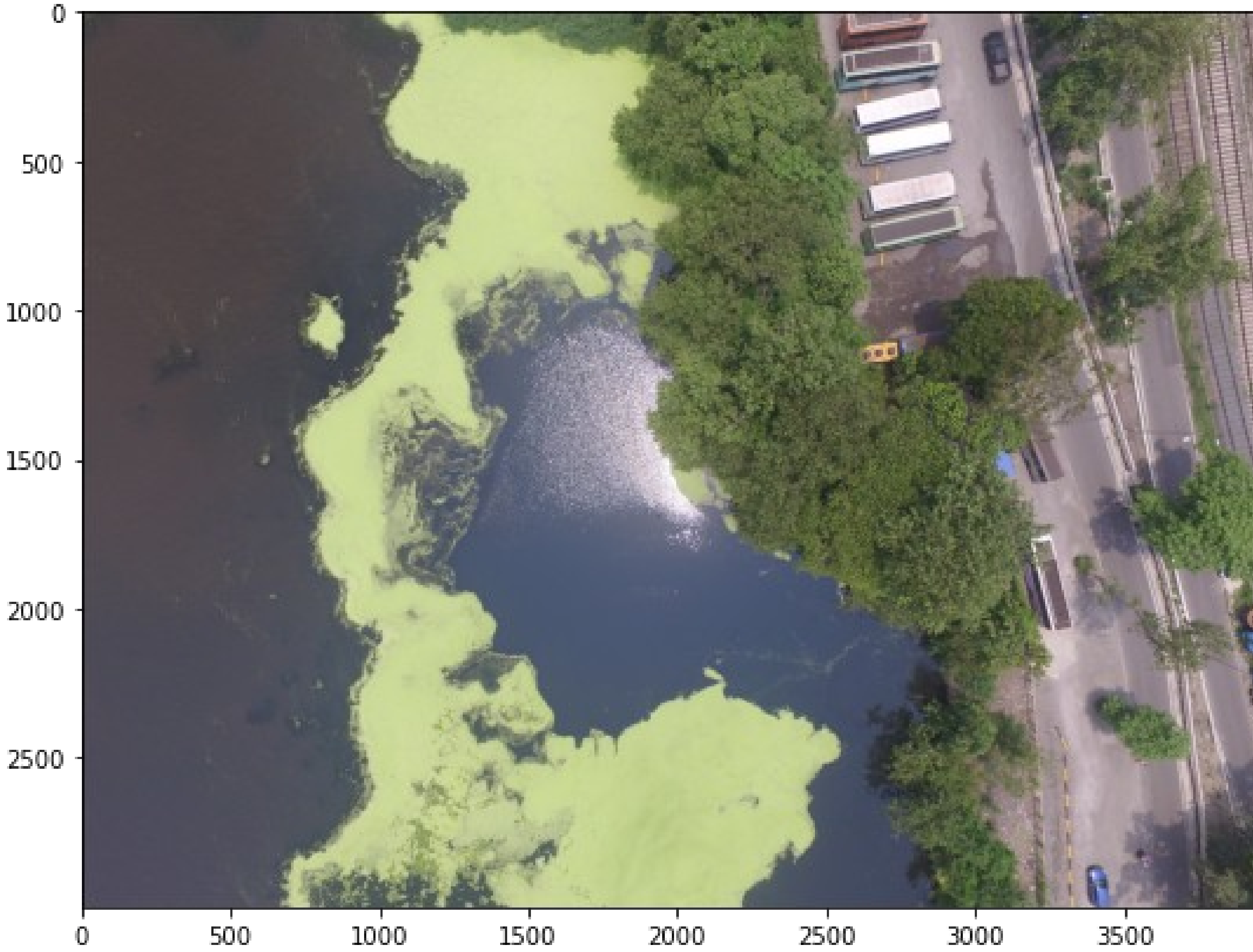
```
#to display the first four images

for i in range(4):
    plt.figure(figsize=(10,7))
    plt.imshow( images[i])
```

0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000
3500

0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000
3500


0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000
3500



In [9]:

```
# stitching:
# initialize OpenCV's image sticher object and then perform the image
print("stitching images...")
stitcher = cv2.Stitcher_create(0)
(status, stitched) = stitcher.stitch(images)
```

```
stitching images...
```

In [10]:

```
if status == 0:
    print("stitching successful")
```

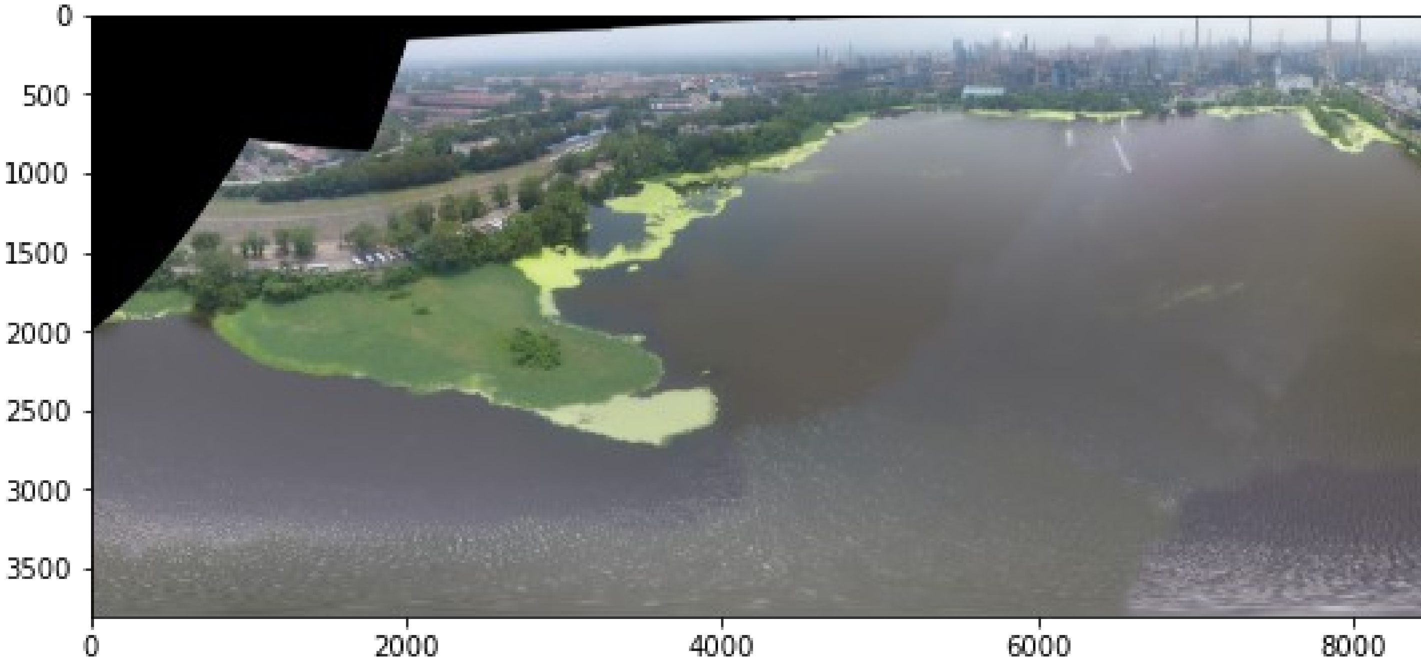
```
stitching successful
```

In [11]:

```
# display  the current result:
plt.figure(figsize=(15,15))
plt.imshow( stitched)
```

Out[11]:

```
<matplotlib.image.AxesImage at 0x7f31c01b79d0>
```



In [12]:

```
# create a 10 pixel border surrounding the stitched image
print("cropping...")
stitched = cv2.copyMakeBorder(stitched, 10, 10, 10, 10,cv2.BORDER_CONSTANT, (0, 0,
0))
```

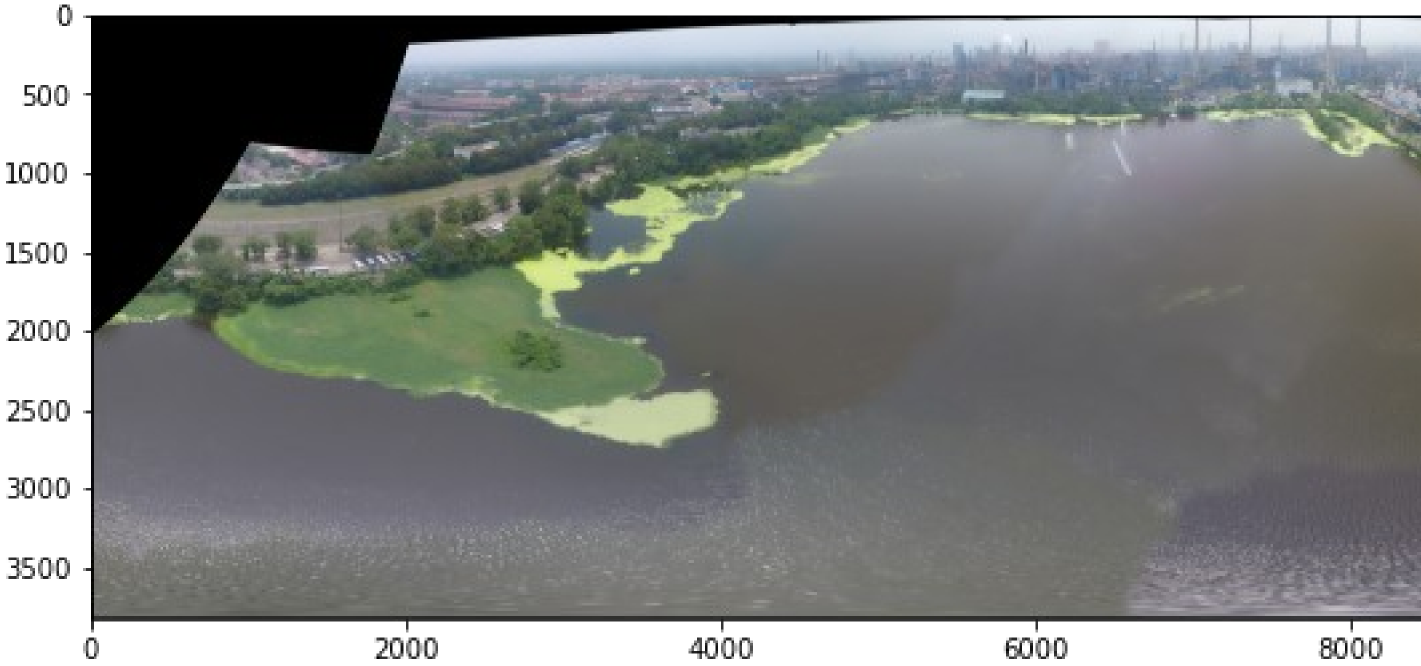
```
cropping...
```

In [13]:

```
# display  the current result:
plt.figure(figsize=(15,15))
plt.imshow( stitched)
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x7f31c0131350>
```



In [14]:

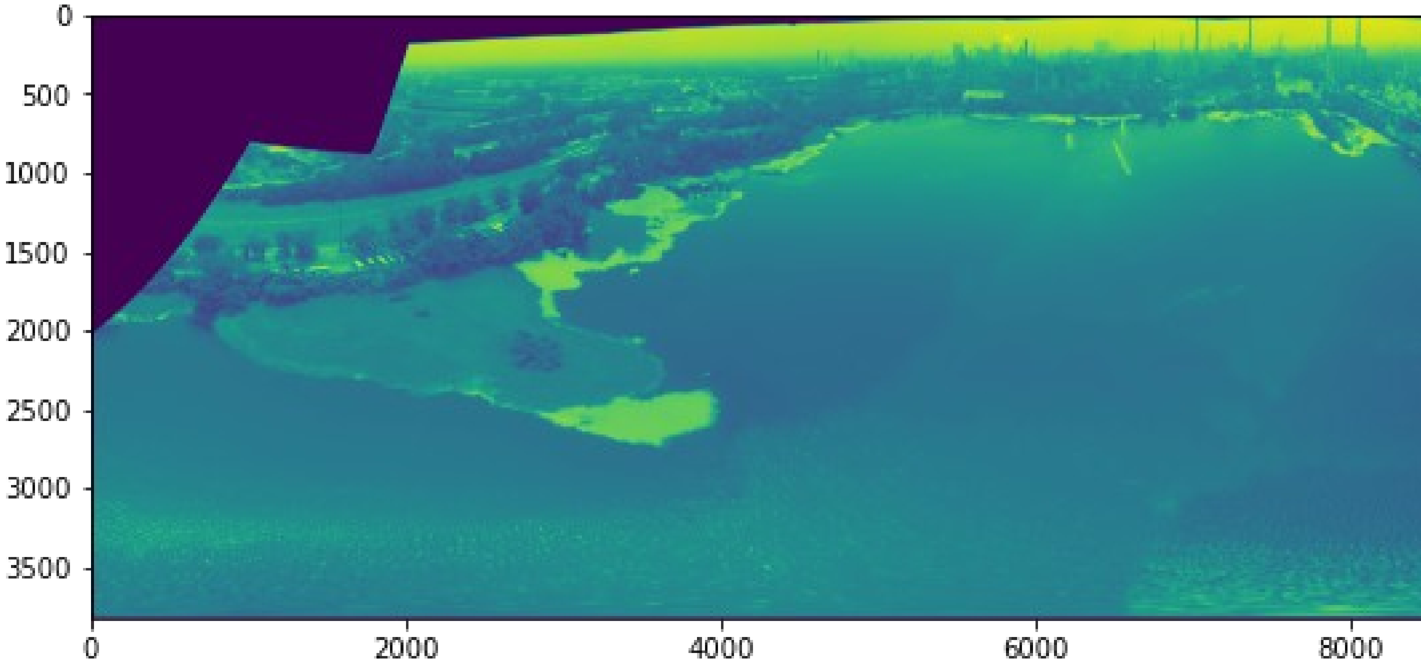
```
# convert the stitched image to grayscale and threshold it such that all pixels
greater than zero are set to 255
# (foreground) while all others remain 0 (background)
gray = cv2.cvtColor(stitched, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY)[1]
```

In [15]:

```
plt.figure(figsize=(15,15))
plt.imshow( gray)
```

Out[15]:

```
<matplotlib.image.AxesImage at 0x7f31c00a1c90>
```

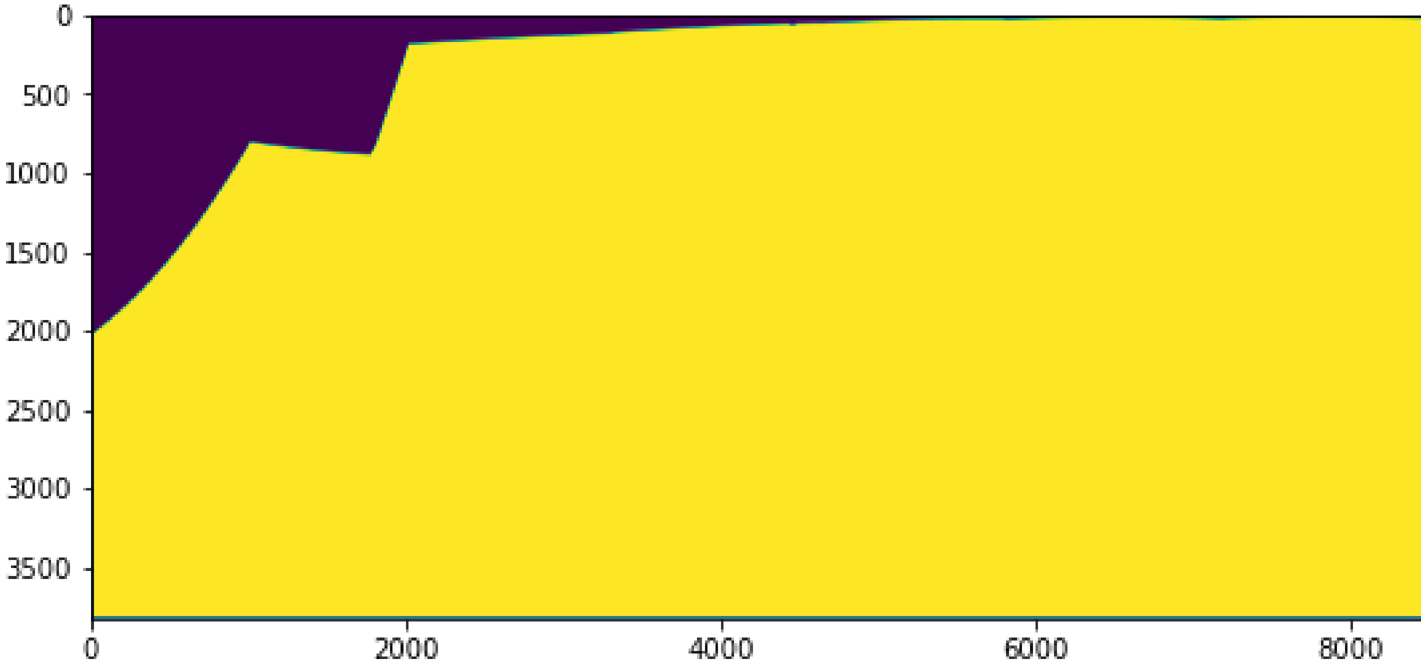


In [16]:

```
plt.figure(figsize=(15,15))
plt.imshow(thresh)
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x7f31c00233d0>
```



In [17]:

```
# find all external contours in the threshold image
# then find the *largest* contour which will be the contour/outline of the stitched
image
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
c = max(cnts, key=cv2.contourArea)
```

In [18]:

```
# allocate memory for the mask which will contain the rectangular bounding box of
the stitched image region
mask = np.zeros(thresh.shape, dtype="uint8")
(x, y, w, h) = cv2.boundingRect(c)
cv2.rectangle(mask, (x, y), (x + w, y + h), 255, -1)
```

Out[18]:

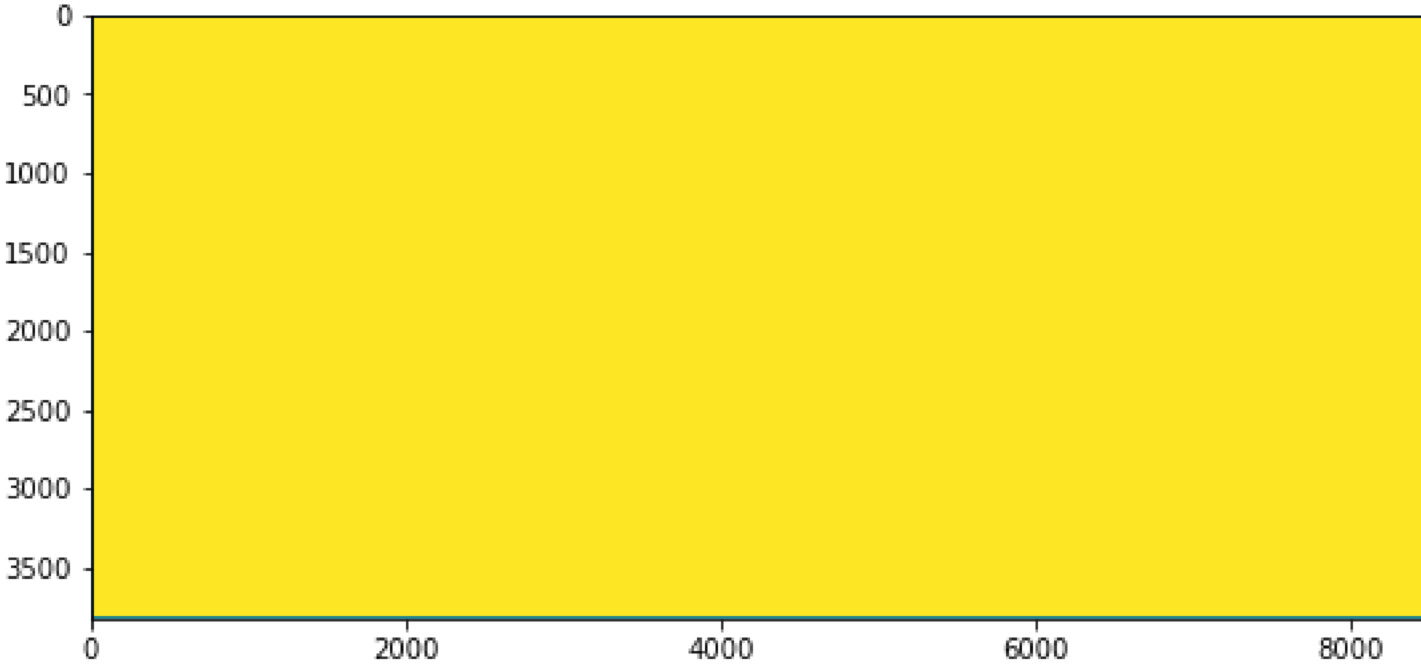
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [19]:

```
plt.figure(figsize=(15,15))
plt.imshow( mask)
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x7f31b5ba30d0>
```



In [20]:

```
# create two copies of the mask: one to serve as our actual minimum rectangular region
# and another to serve as a counter for how many pixels need to be removed to form the minimum rectangular region
minRect = mask.copy()
sub = mask.copy()
```

In [21]:

```
# keep looping until there are no non-zero pixels left in the subtracted image
while cv2.countNonZero(sub) > 0:
# erode the minimum rectangular mask and then subtract the thresholded image from the minimum rectangular mask
# so we can count if there are any non-zero pixels left

    minRect = cv2.erode(minRect, None)
    sub = cv2.subtract(minRect, thresh)
# display and write the current result:
```

In [22]:

```
# find contours in the minimum rectangular mask and then extract the bounding box (x, y)-coordinates
cnts = cv2.findContours(minRect.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
c = max(cnts, key=cv2.contourArea)
(x, y, w, h) = cv2.boundingRect(c)
```

In [23]:

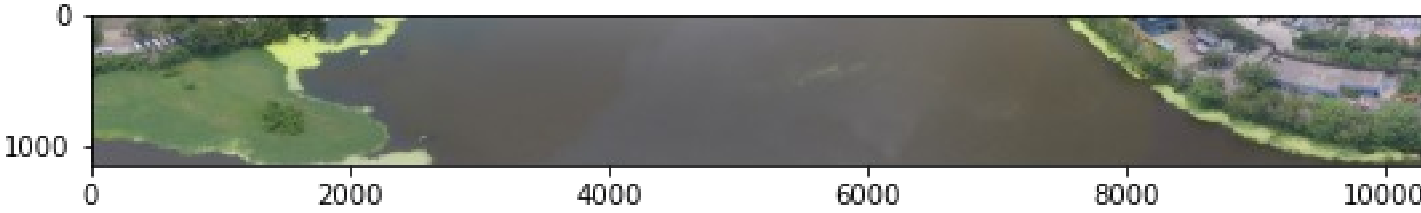
```
# use the bounding box coordinates to extract the our final
# stitched image
stitched = stitched[y:y + h, x:x + w]
```

In [24]:

```
# display the output stitched image to our screen
#final Image
plt.figure(figsize=(10,7))
plt.imshow(stitched)
```

Out[24]:

```
<matplotlib.image.AxesImage at 0x7f31b5b1d310>
```



In [ ]: