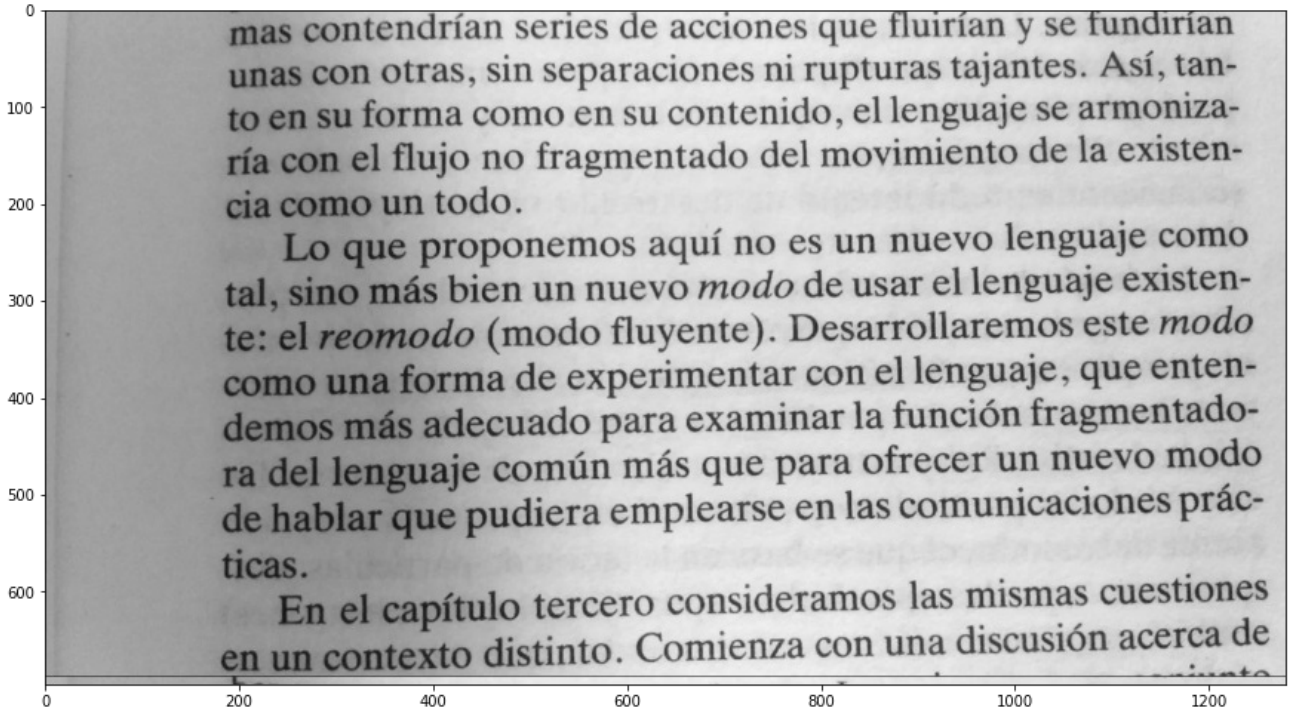
CHALLENGE 3

Xabier Oyanguren Asua 1456628

We first import the image we will work with grayscaled:

In [226]:

```
import cv2
from skimage.measure import label, regionprops
import matplotlib.pyplot as plt
import numpy as np

image = cv2.imread("bohm_resol.jpeg", cv2.IMREAD_GRAYSCALE)
def plot(im):
    fig, ax = plt.subplots(1,1, figsize=(15,10))
    ax.imshow(im, cmap='gray')
    plt.show()

plot(image)
print(image.dtype, image.shape)
```

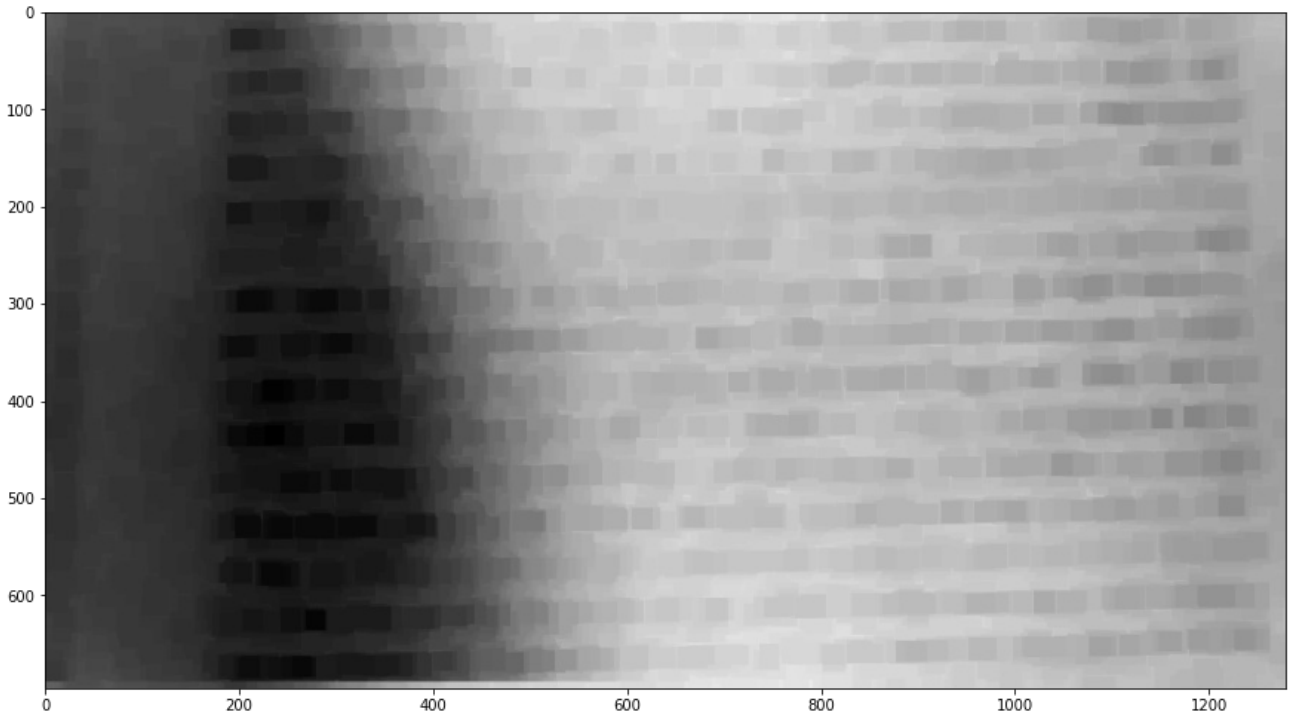



```
uint8 (696, 1280)
```

We first dilate the image (erode the text) until no text is present, then dilate to get an estimate of the background of the text.

In [227]:

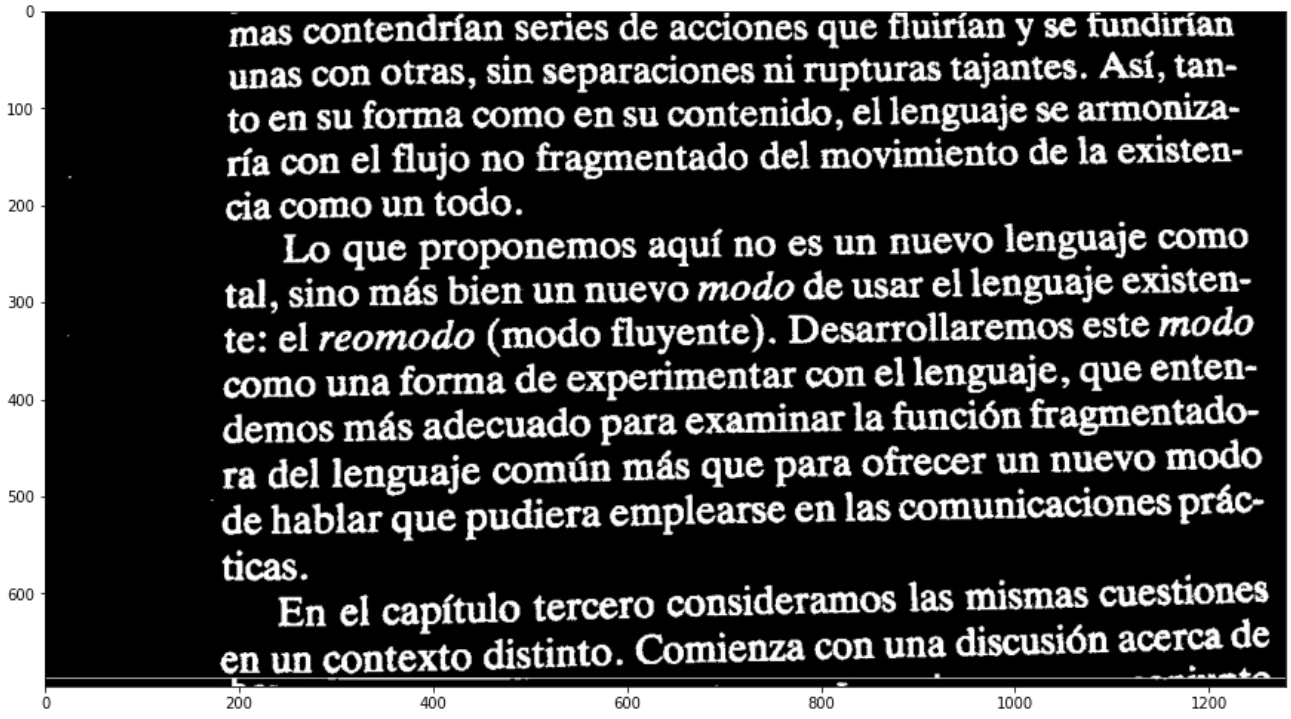
```
backgr = image.copy()
#for i in range(10,1,-1):
kernel = np.ones((20,20),dtype=np.uint8) # structuring element
backgr = cv2.erode(cv2.dilate(backgr, kernel), kernel)

plot(backgr)
```



A binarized background subtraction will yield a more isolated view of what the text is:

In [228]:

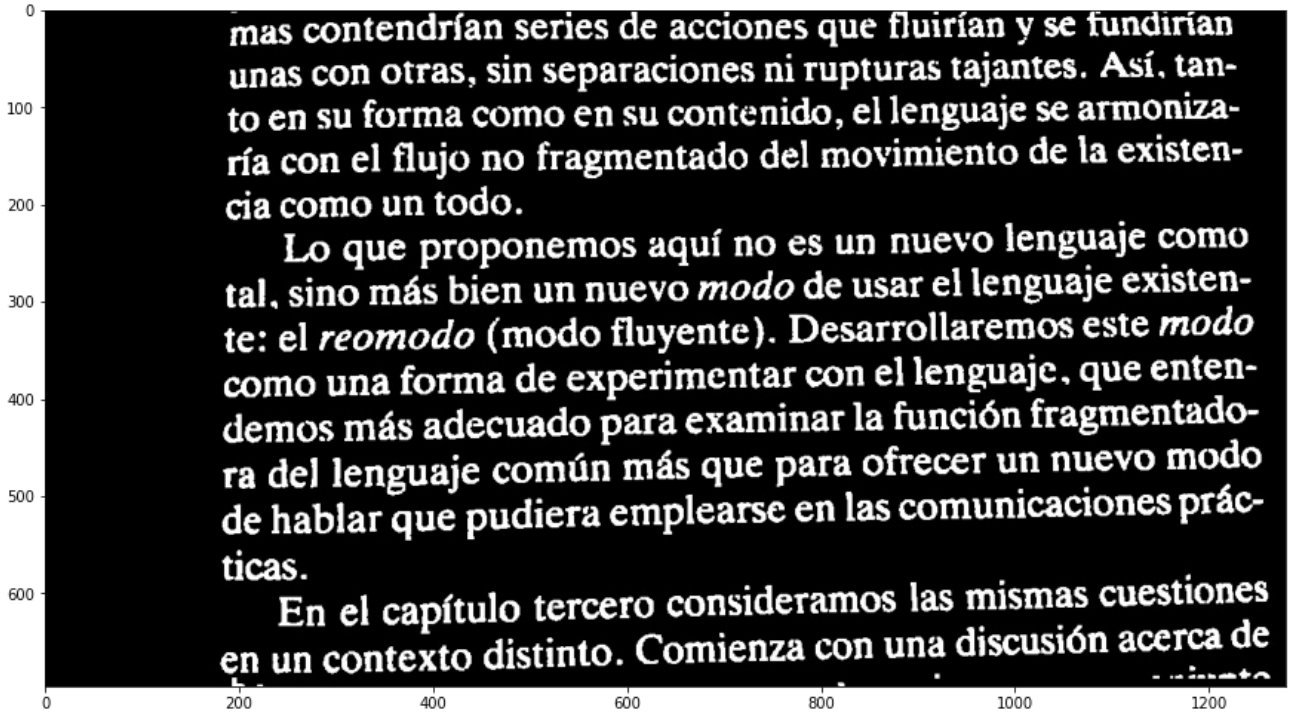
```
bimage = np.abs(image.astype(np.float64)-backgr.astype(np.float64))
bimage = (((bimage/bimage.max())>0.15)*255).astype(np.uint8)
plot(bimage)
```



We can see the image has small dot-like imperfections (in the left top there is one clear). We can remove them easilby using for example an erode and dilate with a small structuring element or using a small kernel median filter.

In [262]:

```
kernel = np.ones((3,3),dtype=np.uint8) # structuring element
bimage_cor = cv2.dilate(cv2.erode(bimage, kernel), kernel)

plot(bimage_cor)
```



In fact, this last step has allowed us to make the individual characters get separated, which is fine for the following.

We should now generate three images, where:

- **Photo A**: In one, each letter is a single connected component (say the dots of the i should be connected to its body). We will use this image to count the number of characters in the photo.
- **Photo B**: In a second image we should have each word as a single connected component, so that we can use it to count the number of words in the image.
- **Photo C**: In a third image we should have each line as a single connected component, so that we can use it to count the number of rows of text in the photo.

We can consecutively obtain them using the fact that the orthogonal distance between the set of parts of a same character is smaller than their distance with parts of other characters, allowing us to use a simple dilatation. Then using the fact that the distance between words is bigger than the distance between the characters of a same word, we can isolate the words in single connected componeent using a dilatation with a bigger structural component. Lastly, using the fact that the separation between words is smaller than the separation between lines, we can englobe the lines in separate connected componenets using a yet bigger structural element.

Photo A

In [269]:

```
# first separate the letters horizontally with an erosion
kernel = np.ones((1,3),dtype=np.uint8) # structuring element
char_im = cv2.erode(bimage_cor, kernel)

# then try to join the letter parts (eg dots of i-s)
kernel = np.ones((9,1),dtype=np.uint8) # structuring element
char_im = cv2.dilate(char_im, kernel)
#bimage_cor = cv2.erode(bimage_cor, kernel)

# then separate more the letters horizontally with an erosion
kernel = np.ones((1,2),dtype=np.uint8) # structuring element
char_im = cv2.erode(char_im, kernel)

plot(char_im)
```

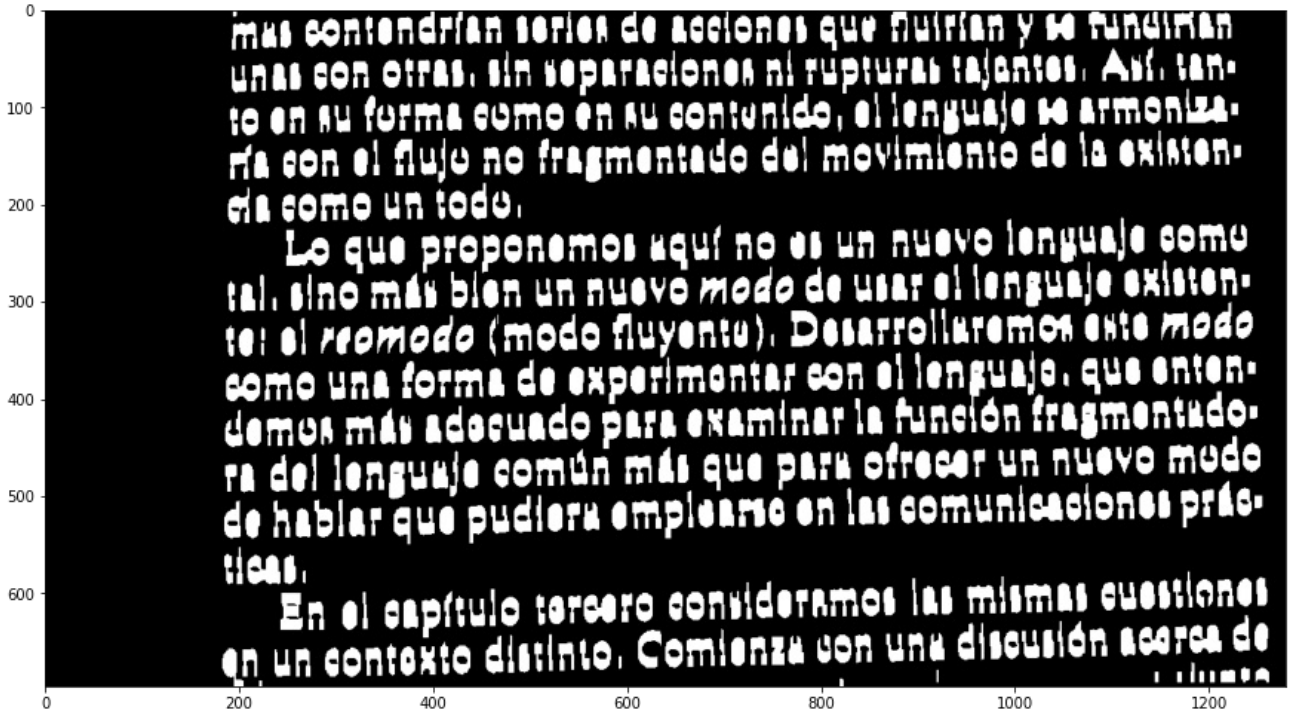


Photo B

In [315]:

```
# now try to join the word parts (eg dots of i-s)
kernel = np.ones((1,12),dtype=np.uint8) # structuring element
word_im = cv2.dilate(char_im, kernel)
# separate more the words
kernel = np.ones((1, 14),dtype=np.uint8) # structuring element
word_im = cv2.erode(word_im, kernel)
# separate more the lines
kernel = np.ones((5, 1),dtype=np.uint8) # structuring element
word_im = cv2.erode(word_im, kernel)
plot(word_im)
```

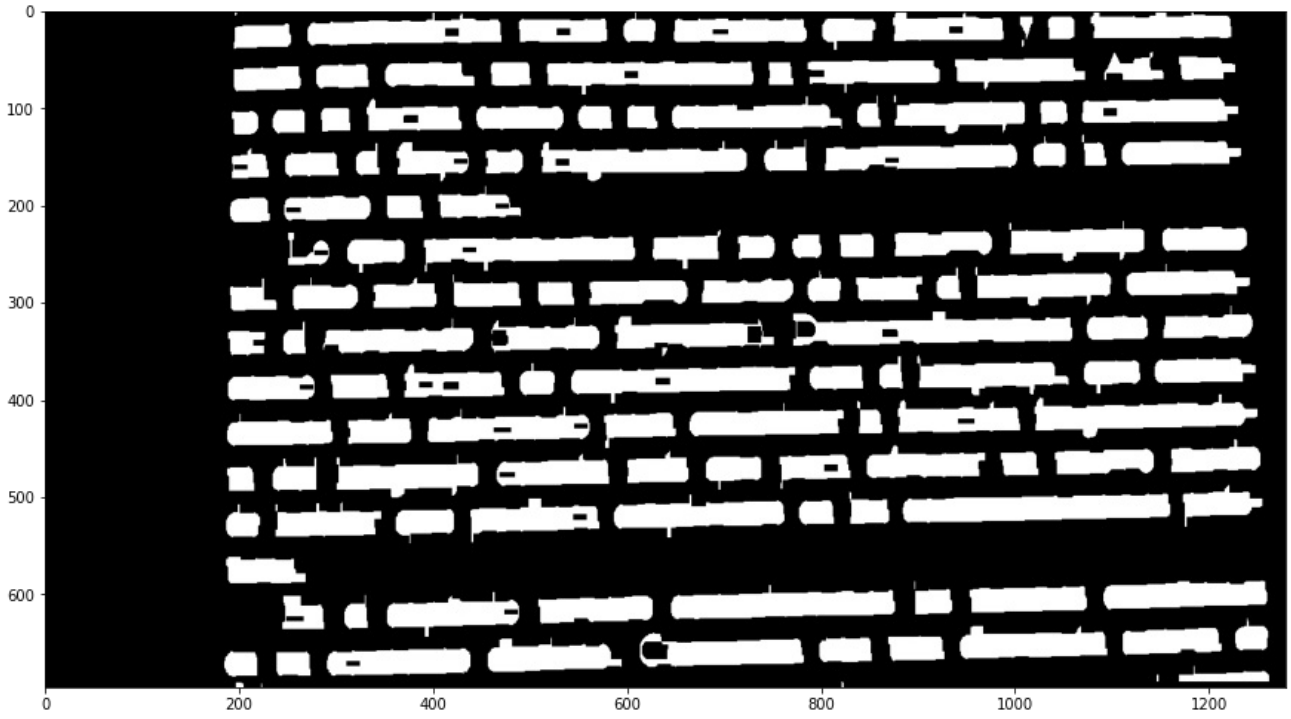
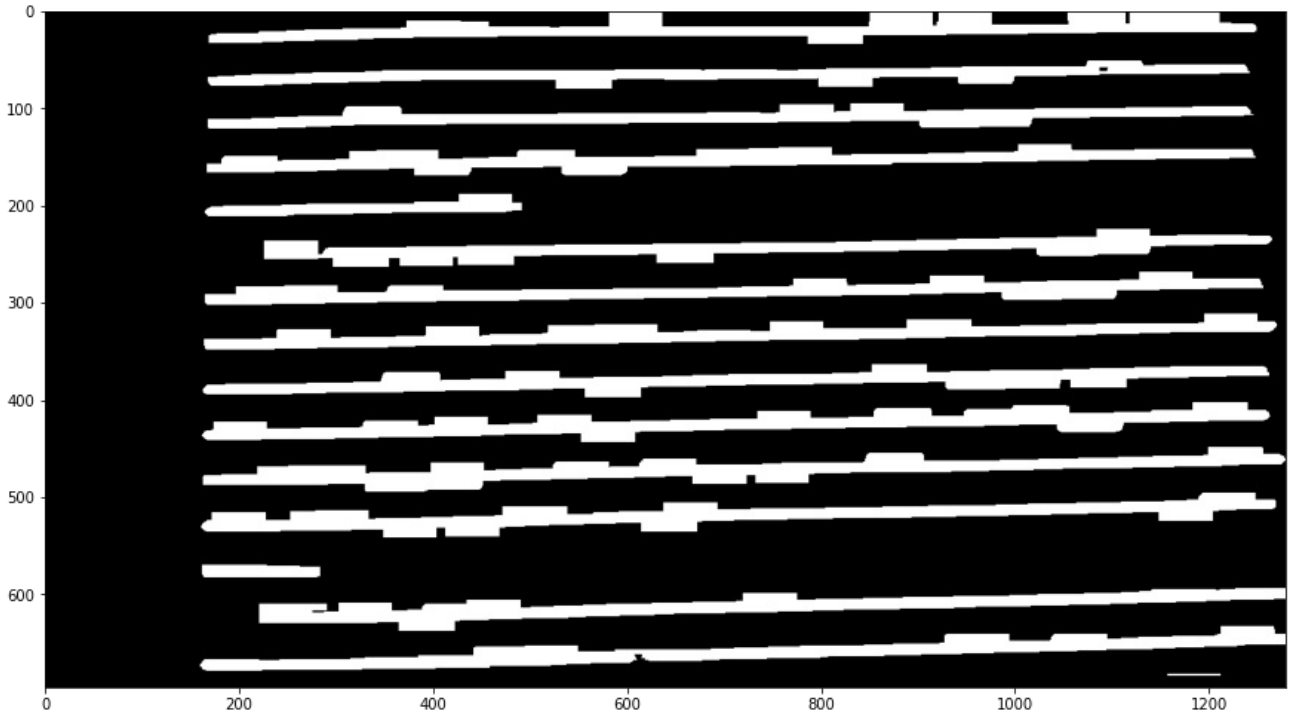


Photo C

In [325]:

```
# separate more the lines
kernel = np.ones((2, 1),dtype=np.uint8) # structuring element
line_im = cv2.erode(word_im, kernel)
# now try to join the line parts
kernel = np.ones((1,55),dtype=np.uint8) # structuring element
line_im = cv2.dilate(line_im, kernel)

plot(line_im)
```



To count the number of connected components in each photo we will use two different approaches:

- Count the number of connected components directly
- Filter the connected components with significant area, in case there are some parts of the letters or words that were not correctly joined (or in case there is still some noise, or in the case of lines, the line that is detected in the border of the image).

We will provide both estimates.

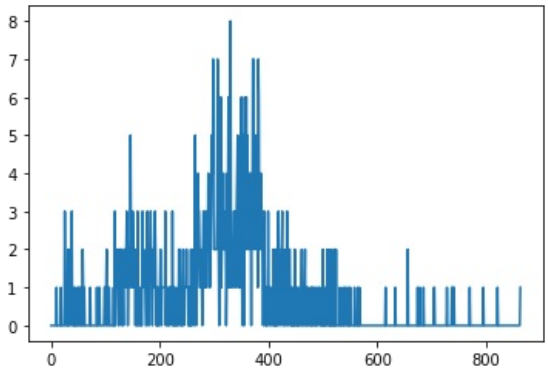
In [337]:

```
def compute_histogram(im, max_bins):
    h=np.zeros((max_bins), dtype=np.float64)
    for I in im.flatten():
        h[I]+=1
    return h
```

Number of Characters

In [350]:

```
char_label_image = label(char_im)
areas_ch = np.array([region.area for region in regionprops(char_label_image)])
h = compute_histogram(areas_ch, areas_ch.max()+1)
plt.plot(h)
plt.show()

print(f"Raw estimate as number of connected components for number of characters is: {len(areas_ch)}\n\
The estimate using the accumulation point of area {200} in the histogram is {len([region.label for region in regi
onprops(char_label_image) if region.area>=200])}")
```

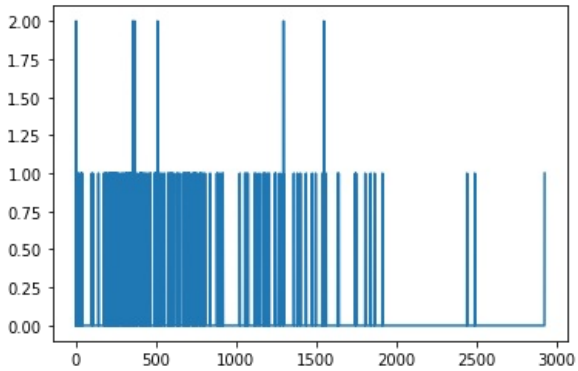



```
Raw estimate as number of connected components for number of characters is: 659
The estimate using the accumulation point in the histogram is 519
```

Number of Words

In [361]:

```
word_label_image = label(word_im)
areas_w = np.array([region.area for region in regionprops(word_label_image)])
h = compute_histogram(areas_w, areas_w.max()+1)
plt.plot(h)
plt.show()

print(f"Raw estimate as number of connected components for number of words is: {len(areas_w)}\n\
The estimate using the accumulation point of area {100} in the histogram is {len([region.label for region in regi
onprops(word_label_image) if region.area>=100])}")
```

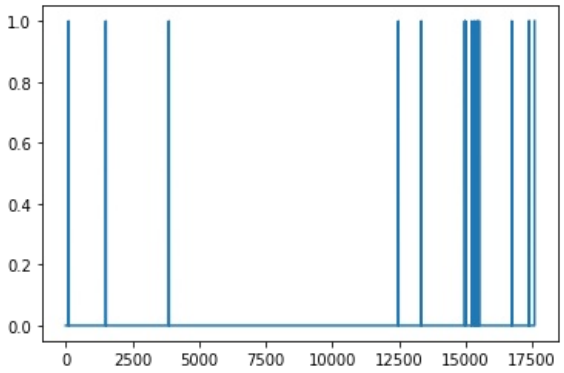



```
Raw estimate as number of connected components for number of words is: 164
The estimate using the accumulation point of area 100 in the histogram is 152
```

Number of Lines

In [355]:

```
lines_label_image = label(line_im)
areas_l = np.array([region.area for region in regionprops(lines_label_image)])
h = compute_histogram(areas_l, areas_l.max()+1)
plt.plot(h)
plt.show()

print(f"Raw estimate as number of connected components for number of lines is: {len(areas_l)}\n\
The estimate using the histogram is {len([region.label for region in regionprops(lines_label_image) if region.area>=200])}")
```




```
Raw estimate as number of connected components for number of lines is: 16
The estimate using the histogram is 15
```