```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

HSL and HSV color spaces.

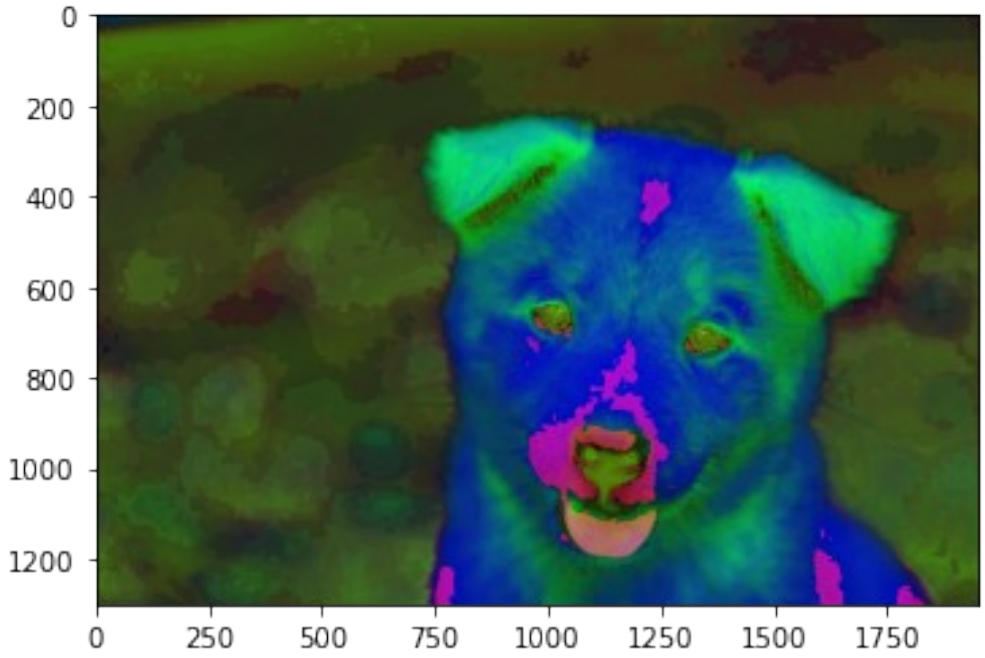
These color spaces are the created alterbnate to the RGB model by computer graphic engineers as this captures the perception of how humans view color.

```
image = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/00-puppy.jpg")
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7fe6390d9c88>
```

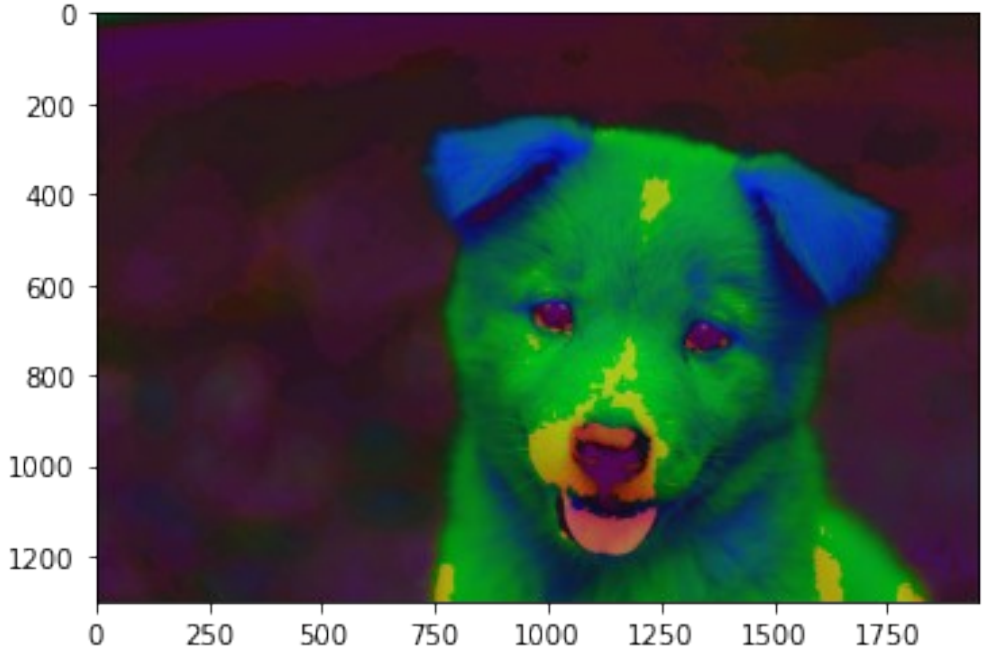

```
image_hsv = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
plt.imshow(image_hsv)
```

```
<matplotlib.image.AxesImage at 0x7fe6398f7320>
```




```
image_hsl = cv2.cvtColor(image,cv2.COLOR_BGR2HLS)
plt.imshow(image_hsl)
```

```
<matplotlib.image.AxesImage at 0x7fe618185f98>
```

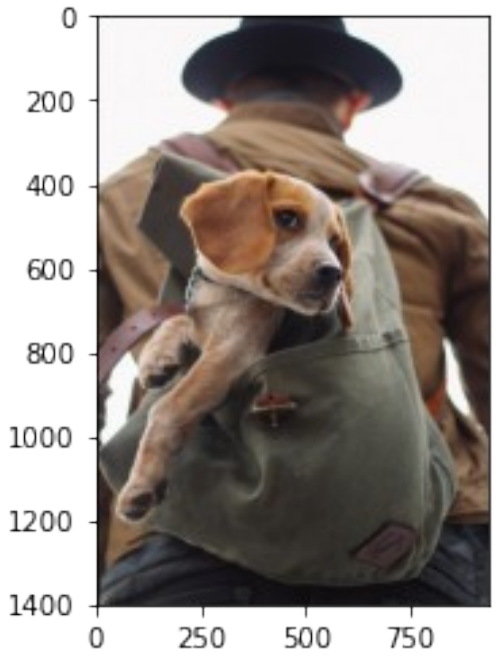


Blending and Pasting Frames

Blending frames is an important tool for blending or merging two images into one. This is predominatly done in video transition to ensure there is weighted representation between the two frames. The formulae for blending is done on the pixel basis: `pixel_blend = alpha * pixel_img1 + beta * pixel_img2 + gamma` where alpha and beta are the weighted representations of each pixel. This is a powerful formula as we can tinker around with the values for more representation of one image over the other. The **`addWeighted`** function is used here.

```
img1 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/dog_backpack.jpg")
img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/watermark_no_copy.png")
img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)

plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7fe6180ea748>
```

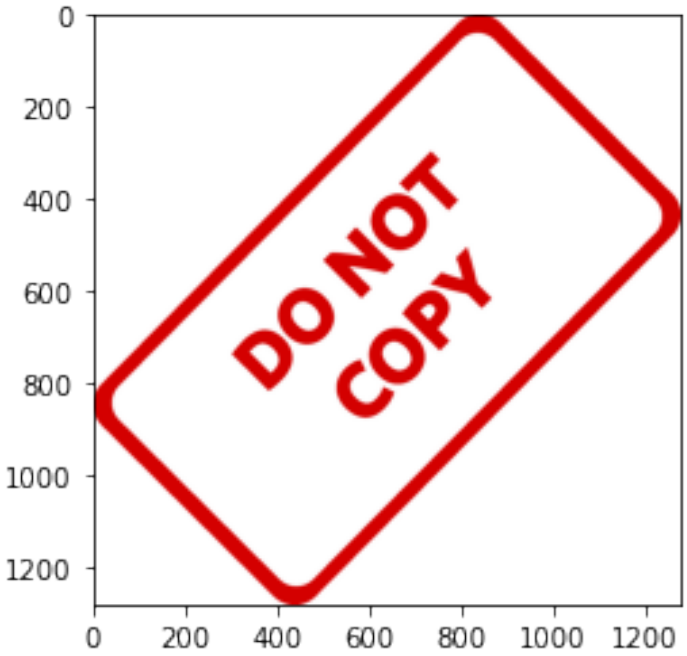



```
img1.shape
```

```
(1401, 934, 3)
```

```
plt.imshow(img2)
```

```
<matplotlib.image.AxesImage at 0x7fe639930240>
```

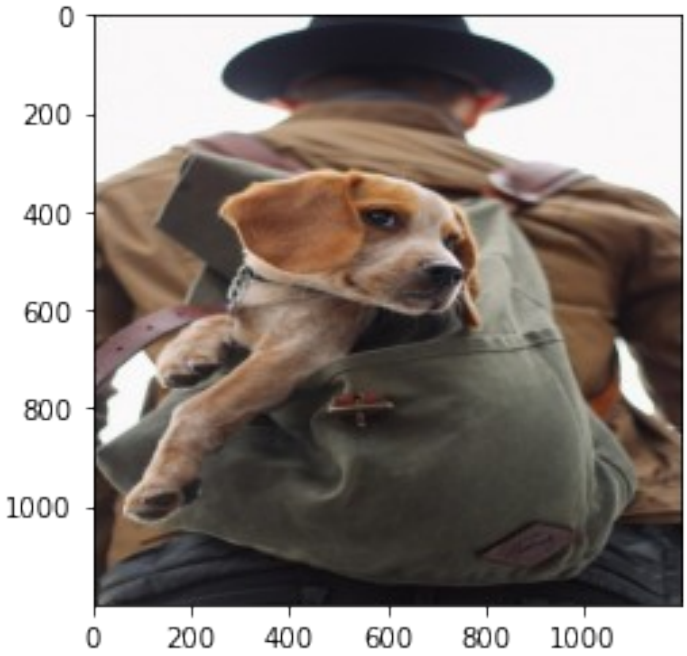



```
img2.shape
```

```
(1280, 1277, 3)
```

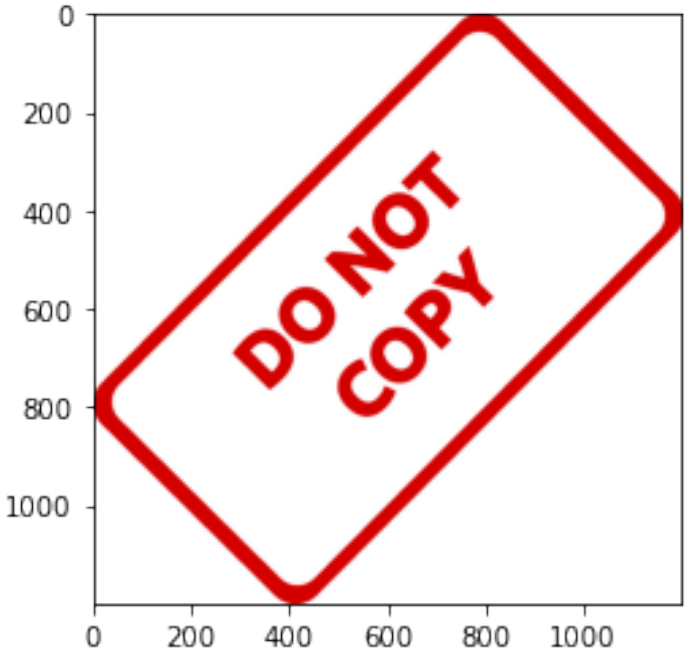
```
img1 = cv2.resize(img1,(1200,1200))
img2 = cv2.resize(img2,(1200,1200))

plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7fe618106780>
```



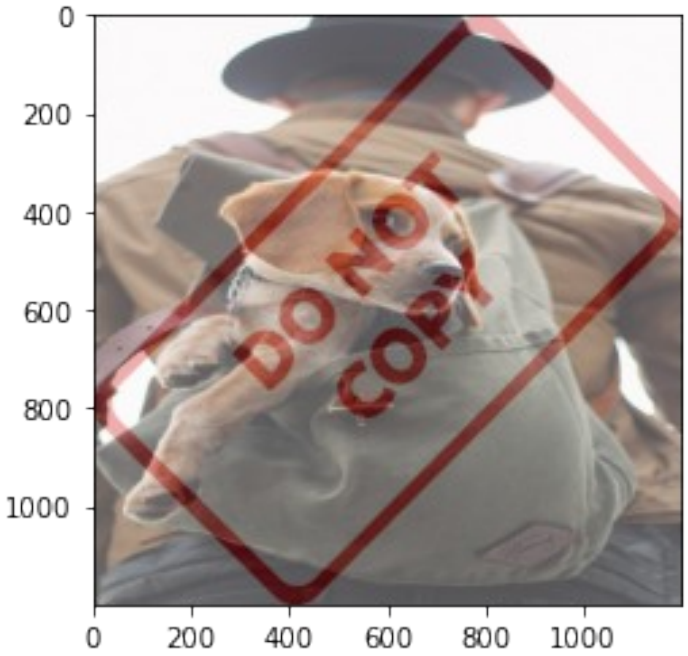

```
plt.imshow(img2)
```

```
<matplotlib.image.AxesImage at 0x7fe6495d39b0>
```




```
blended = cv2.addWeighted(img1,0.7,img2,0.3,gamma=0)
plt.imshow(blended)
```

```
<matplotlib.image.AxesImage at 0x7fe6188862b0>
```



Note that this is only the case if both the images have the **same size**. For different sizes, there will be an error. Hence we will be using the concept of overlay over here where will paste the image of one over the other. This just involves a simple numpy slicing!

```
#We first define the two images and resize the second image
img1 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/dog_backpack.jpg")
img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/watermark_no_copy.png")
img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
img2 = cv2.resize(img2,(600,600))
```

Now we define the offset of where the second image should be placed.

```
#Assigning large and small image
large_image = img1
small_img = img2

x_offset = 0 #x and y corrdinates of the overlayed image is starting from origin (0,0)
y_offset = 0

x_end = 600 #Height and width of the overlayed image
y_end = 600
```

According to numpy, the columns of the the image is read in as the x-axis while the rows are read in as the y-axis. Therefore numpy slicing is done as numpy[:y,:x]

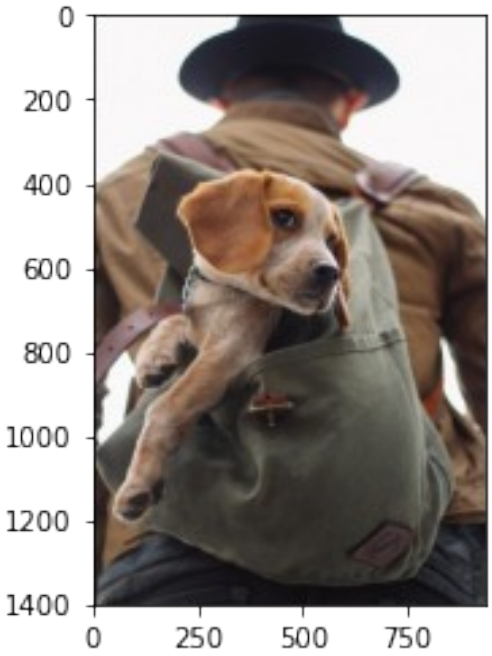
```
large_image[y_offset:y_end,x_offset:x_end] = small_img

plt.imshow(large_image)
```

```
<matplotlib.image.AxesImage at 0x7fe6080ae940>
```

Now we are going to use blending with masks where we would blend a segment or mask of the smaller image, with the larger image.

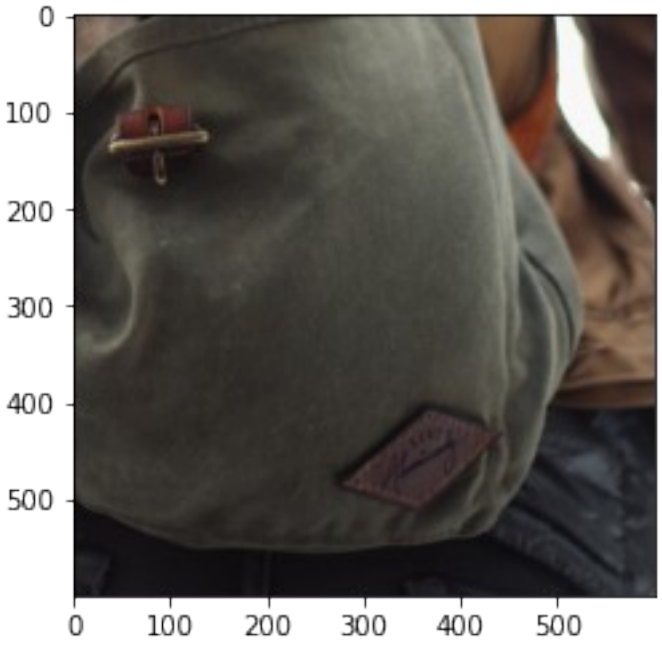
```
#We first define the two images and resize the second image
img1 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/dog_backpack.jpg")
img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2 = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/watermark_no_copy.png")
img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
img2 = cv2.resize(img2,(600,600))

plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7fe6187fe3c8>
```



Since we have resized the small image to 600X600, we need to estimate the bottom location of the roi.In order to do that we need to extract the Region of Interest (ROI) of the section below to place our overlay.

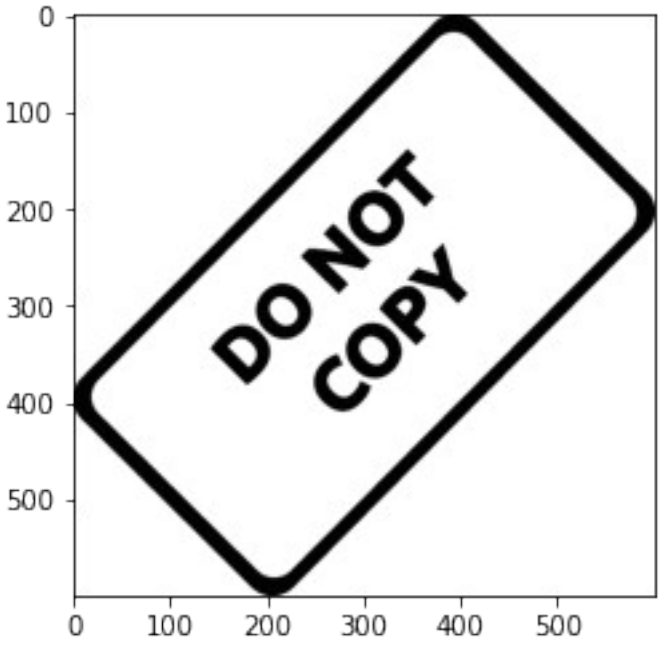
```
rows,columns,channel = img2.shape

img1.shape

(1401, 934, 3)

x_offset = 934 - rows
y_offset = 1401 - columns

roi = img1[y_offset:1401,x_offset:934]

plt.imshow(roi)

<matplotlib.image.AxesImage at 0x7f9d1213eef0>
```



Now that we have extracted the ROI of the image, we move our focus on the second image. We need to create a mask of the image so as to overlay the ROI portion on the original image. Lets first convert the second image to gray scale.

```
mask = cv2.cvtColor(img2,cv2.COLOR_RGB2GRAY)
plt.imshow(mask,cmap = 'gray')
```

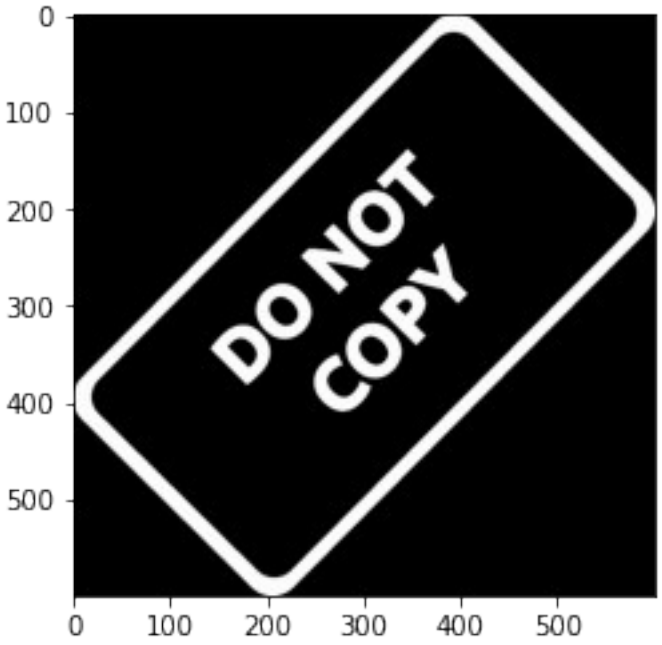
```
<matplotlib.image.AxesImage at 0x7f9d16e21a90>
```



Since the black portion is going to be the part hidden by the mask, we need to inverse this image so as to extract the text from the image. We use the **`bitwise_not`** function to invert the image.

```
mask_inv = cv2.bitwise_not(mask)
plt.imshow(mask_inv,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d42256400>
```



Unfortunately since the image is in gray scale, it has lost the channel refernce input

```
mask_inv.shape
```

```
(600, 600)
```

Hence we need to work on 3 Dimensional image, we need to to a create an array and multiply it by 255. **np.full()** does it in one step

```
white_background = np.full(img2.shape,255,dtype=np.int32)
white_background
```

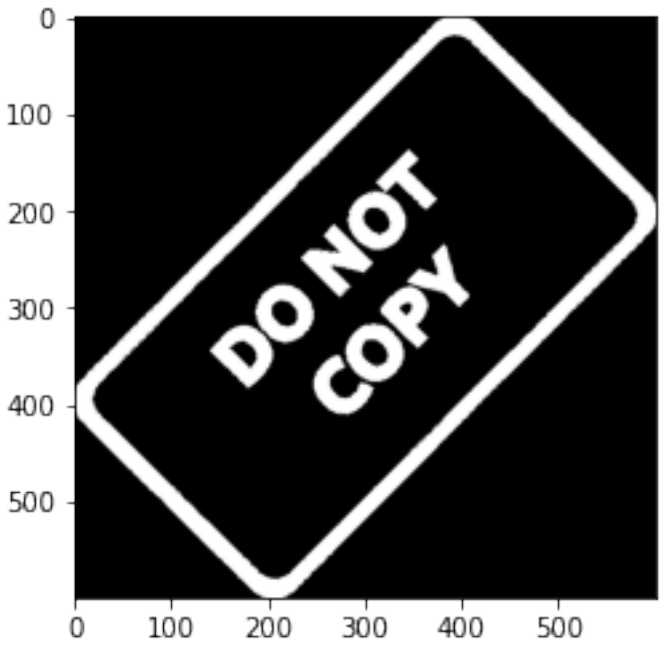
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
```

```
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=int32)
```

This array is a 3 channel white image. Now in order to apply this to the mask, we use the bitwise_or operation, of applying the mask on the 3 dimensional array such that only the higher intensity value pixels are dhown in the image. In short, the mask gets applied to all the channels of the image.

```
bk = cv2.bitwise_or(white_background,white_background,mask=mask_inv)
plt.imshow(bk)
```

```
<matplotlib.image.AxesImage at 0x7f9d16c806d8>
```



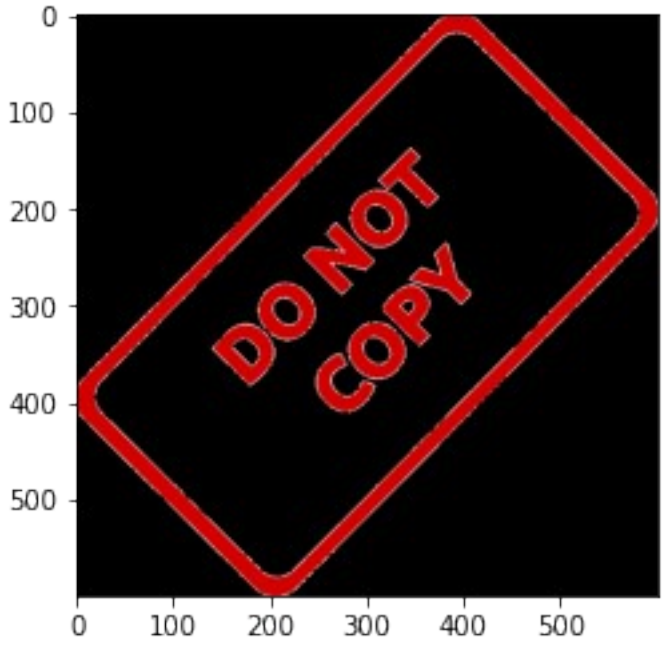

```
bk.shape
```

```
(600, 600, 3)
```

This is how a mask is supposed to be represnted. Lets now apply the mask_inv on the ROI such that we can obtain the red text in the foreground.

```
fg = cv2.bitwise_or(img2,img2,mask=mask_inv)
plt.imshow(fg)
```

```
<matplotlib.image.AxesImage at 0x7f9d21d6b908>
```



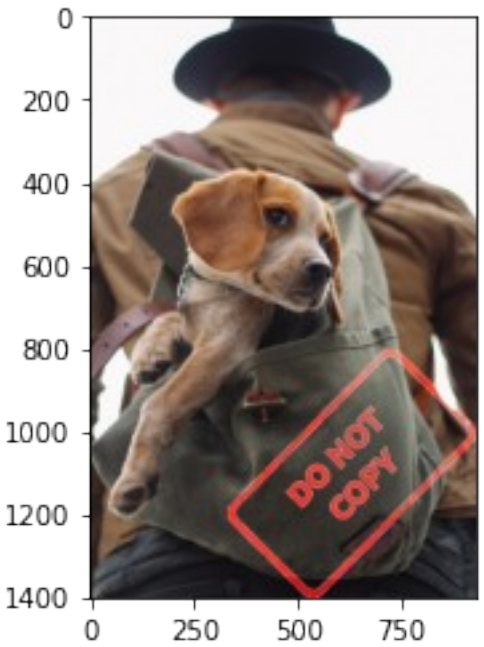
Finally applying it to the final picture, we can finally apply the filtered region of the mask on the the ROI.

```
roi = cv2.bitwise_or(roi,fg)
plt.imshow(roi)
```

```
<matplotlib.image.AxesImage at 0x7f9d16d6ee10>
```

```
img1[y_offset:1401,x_offset:1401] = roi

plt.imshow(img1)
```

```
<matplotlib.image.AxesImage at 0x7f9d440b2ef0>
```

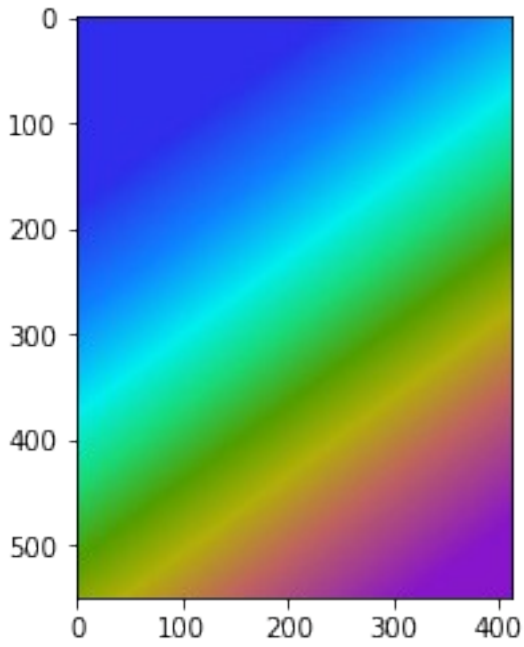


Thresholding

Thresholding is the phenomenon of flooring the intensity values based on the threshold. It is essentially done for segmenting certain objects or enhancing the text on an image for further analysis.

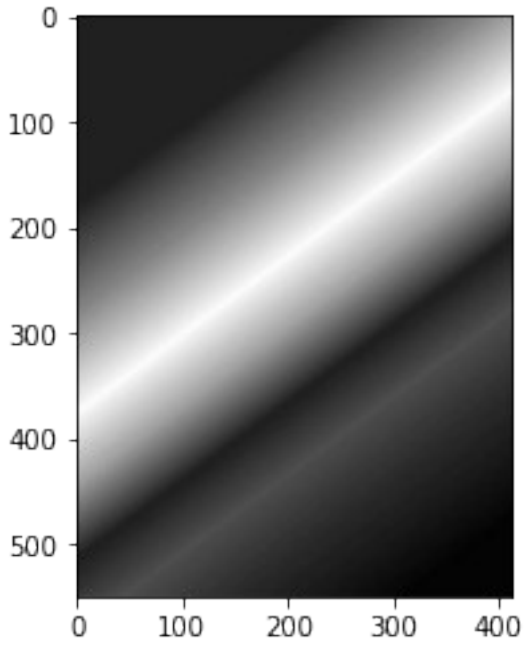
```
img = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/rainbow.jpg")
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f9d17355240>
```




```
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
plt.imshow(img,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d449af748>
```

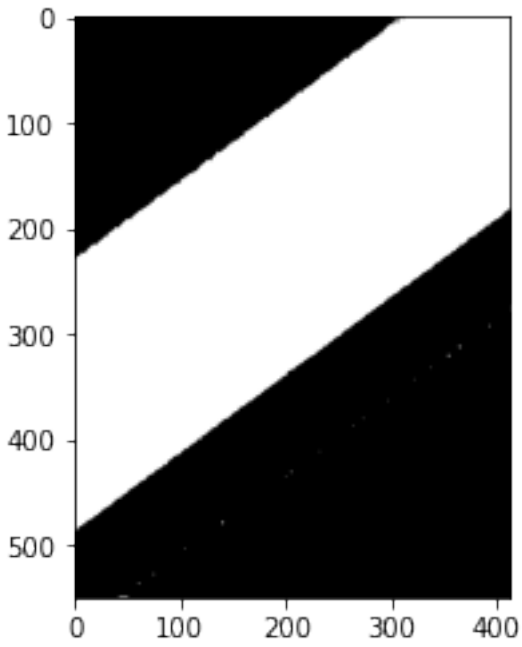



```
img.max()
```

```
214
```

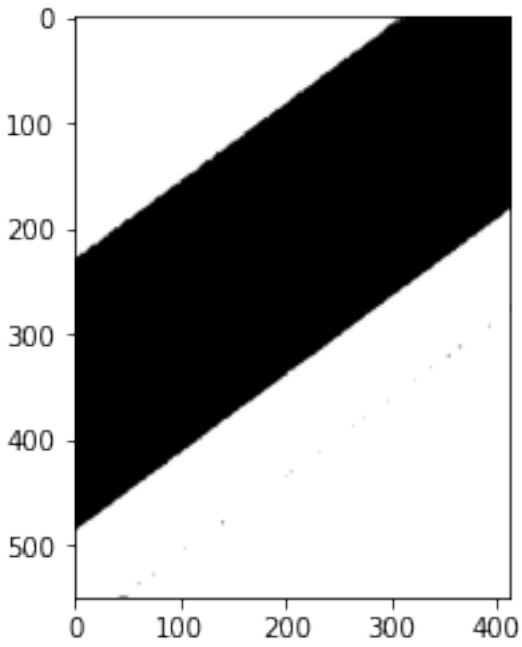
```
ret, img_thresh = cv2.threshold(img,127,255,type=cv2.THRESH_BINARY)
plt.imshow(img_thresh,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d16f46ac8>
```



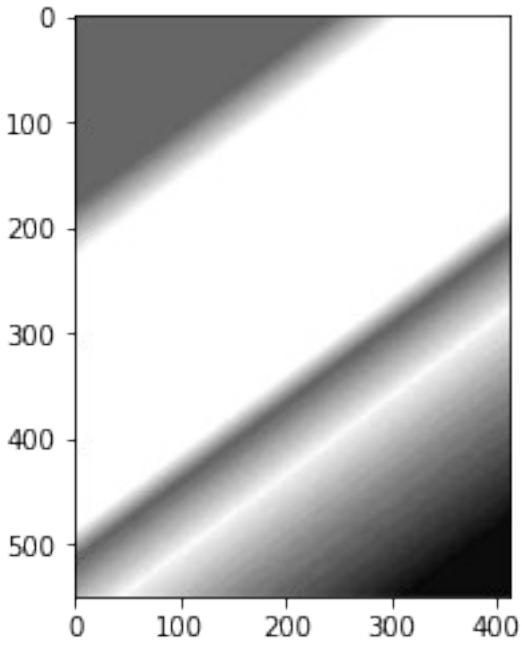

```
ret, img_thresh =
cv2.threshold(img,127,255,type=cv2.THRESH_BINARY_INV)
plt.imshow(img_thresh,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d17215e48>
```



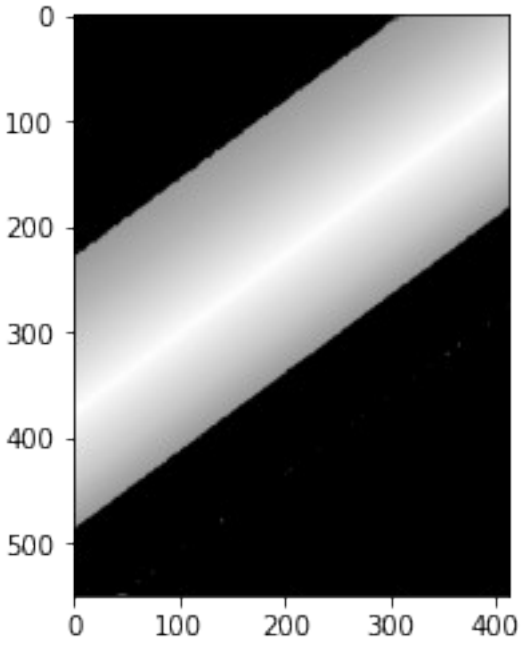

```
ret, img_thresh = cv2.threshold(img,127,255,type=cv2.THRESH_TRUNC)
plt.imshow(img_thresh,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d163d7470>
```

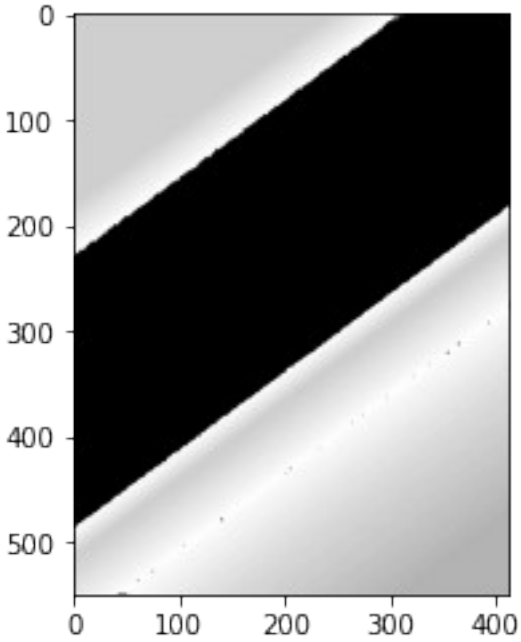



```
ret, img_thresh = cv2.threshold(img,127,255,type=cv2.THRESH_TOZERO)
plt.imshow(img_thresh,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9d17637ef0>
```

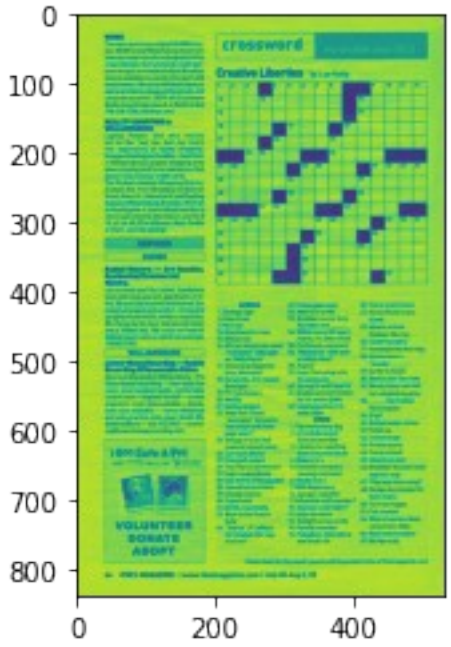



```
ret, img_thresh =
cv2.threshold(img,127,255,type=cv2.THRESH_TOZERO_INV)
plt.imshow(img_thresh,cmap='gray')

<matplotlib.image.AxesImage at 0x7f9d1773b860>
```



```
img = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/crossword.jpg",0)
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f9d17842f28>
```




```
def show_pic(image):
    plt.figure(figsize=(15,15))
    return plt.imshow(image,cmap='gray')

show_pic(img)
```

```
<matplotlib.image.AxesImage at 0x7f9d44c7ee48>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban lifestyle. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of reality shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M// train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park...we'll be waiting!*

SERVICES

DUMBO

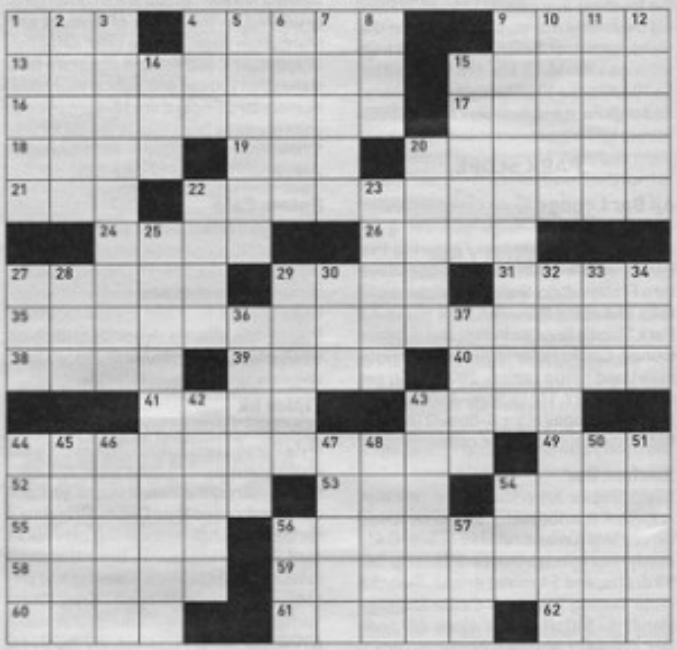
Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee.' We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718 852 2352.*

WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: lock-outs available — some rehearsal and solo practice slots open *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword
we enable your OCD

Creative Liberties by Lee Kelly



ACROSS
1. Bodega sign
4. Exec hires
9. Not me!
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who spat on 1940s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* awardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (var.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. *Twilight* series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Surfer's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____ tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Contaminate
43. Prickly plants
44. Funny schtuff
45. Irked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

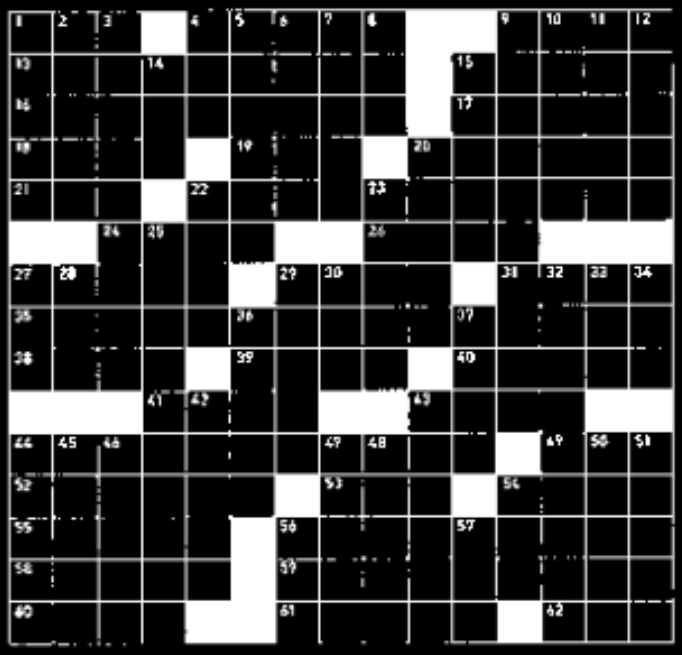
```
ret, img_thresh = cv2.threshold(img,150,255,cv2.THRESH_BINARY_INV)
show_pic(img_thresh)
```

```
<matplotlib.image.AxesImage at 0x7f9cd07c2ac8>
```

crossword

we enable your OCD

Creative Liberties by Lee Kelly



ACROSS
1. Bodega sign
6. Exec hires
9. Not me!
13. Blackbeard's crew
15. Wilchy one
16. African-American novel "translator" who spat on 1940s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 9th reactionary
20. Mellify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Ativan pill output
29. Tina Fey's ex prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* awardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder [war]
7. Milk dispensers
8. Job app. requisite
9. Influential staffmember?
10. Odyssey undertaker?
11. Utter boredom
12. *Tenlight* series scribe
16. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Surfer's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____ the Funkee Homosapien
34. D tip?
36. Bottled water choice
37. Polish up
42. Consummate
43. Prickly plants
44. Funny schtuff
45. Inked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
49. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

```
ret, img_thresh = cv2.threshold(img,180,255,cv2.THRESH_BINARY)
show_pic(img_thresh)
```

```
<matplotlib.image.AxesImage at 0x7f9d007a0cf8>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban lifestyle. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of really shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M/J train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park....we'll be waiting!*

[redacted]

DUMBO

Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee." We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718.852.2352.*

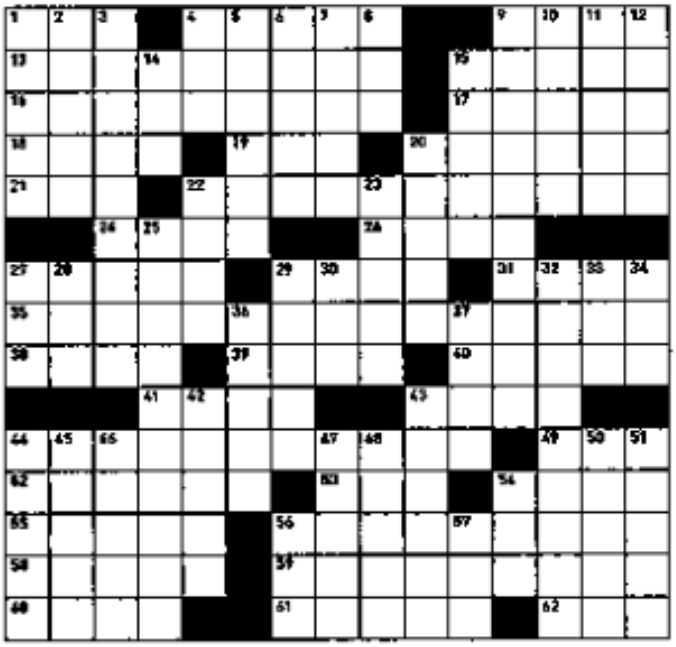
WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: back-outs available — some rehearsal and solo practice slots open. *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword

we enable your OCD

Creative Liberties by Lee Kelly



ACROSS
1. Bodega sign
4. Exec hires
9. Not me!
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who aged on 1960s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* swardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (vs.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. *Twilight* series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Surfer's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____ tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Cantonwale
43. Prickly plants
44. Funny schtuff
45. Irked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

```
ret
```

```
180.0
```

```
img_adap =
cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BI
NARY,11,5)
show_pic(img_adap)
```

```
<matplotlib.image.AxesImage at 0x7f9d32931860>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban lifestyle. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of reality shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M/J train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park....we'll be waiting!*

SERVICES

DUMBO

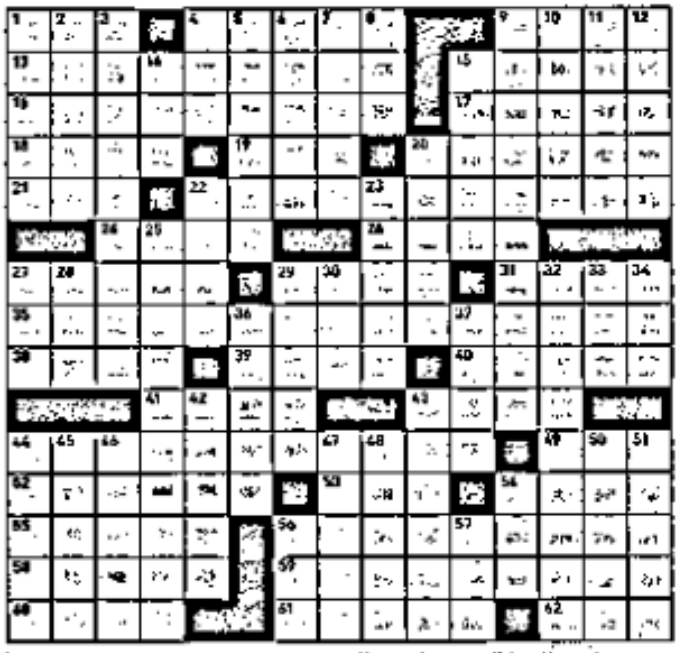
Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee." We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718 852 2352.*

WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: lock-outs available — some rehearsal and solo practice slots open *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword

Creative Liberties by Lee Kelly



ACROSS
1. Bodega sign
6. Exec hires
9. Not met
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who spoke on 1940s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* Bully
44. "Duelist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* swordee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (var.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. Twilight series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boy-_____ friend
23. Atheist activist Madalyn Murray _____
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Sutter's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Contaminate
43. Prickly plants
44. Funny schtuff
45. Inked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

```
blended = cv2.addWeighted(img_thresh,0.6,img_adap,0.4,gamma=0)
show_pic(blended)
```

```
<matplotlib.image.AxesImage at 0x7f9d17e43eb8>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban life style. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of reality shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park....we'll be waiting!*

SERVICES

DUMBO

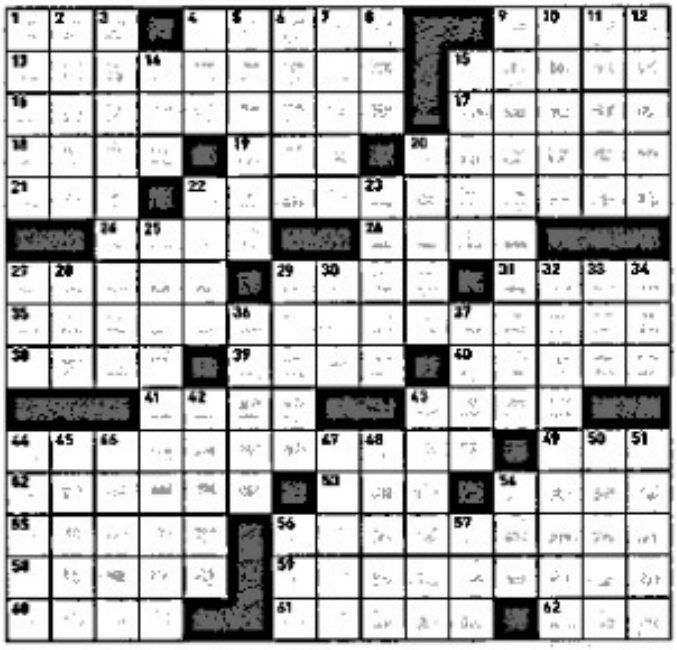
Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee." We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718 852 2352.*

WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: lock-outs available — some rehearsal and solo practice slots open. *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword
we enable your OCD

Creative Liberties
by Lee Kelly



ACROSS
1. Bodega sign
4. Exec hires
9. Not me!
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who spat on 1940s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* swardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (var.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. Twilight series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray _____
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Surfer's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Contaminate
43. Prickly plants
44. Funny schtuff
45. Irked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

```
blended = cv2.addWeighted(img_thresh,0.7,img_adap,0.3,gamma=0)
show_pic(blended)
```

```
<matplotlib.image.AxesImage at 0x7f9d44f2ab38>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban lifestyle. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of reality shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M/J train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park...we'll be waiting!*

SERVICES

DUMBO

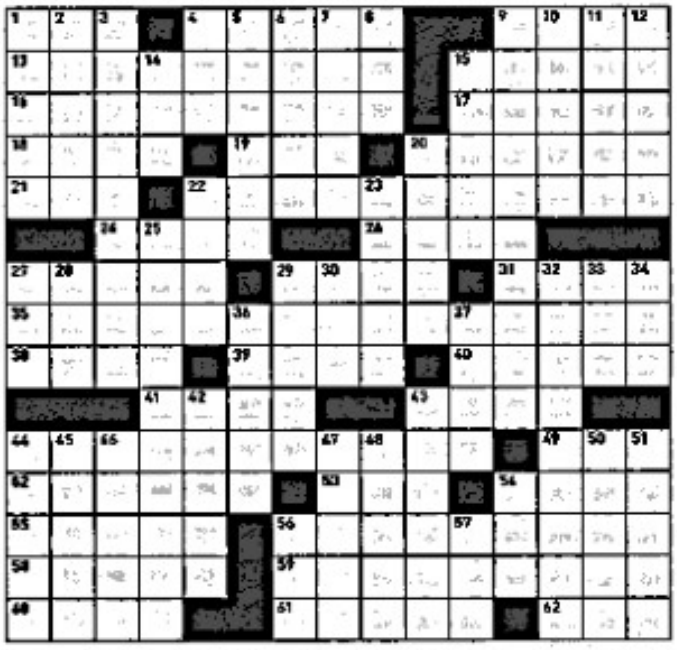
Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee." We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718 852 2352.*

WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: lock-outs available — some rehearsal and solo practice slots open. *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword
we enable your OCD

Creative Liberties
by Lee Kelly



ACROSS
1. Bodega sign
4. Exec hires
9. Not me!
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who aged on 1940s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* swardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (var.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. *Twilight* series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Sutter's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____ tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Contaminate
43. Prickly plants
44. Funny schtuff
45. Irked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

```
ret, img_thresh = cv2.threshold(blended,130,255,cv2.THRESH_OTSU)
show_pic(img_thresh)
```

```
<matplotlib.image.AxesImage at 0x7f9d45b249e8>
```

WONK
The sister store to our original DUMBO location, WONK's new Williamsburg showroom features modern furniture designed to fit the urban lifestyle. Our functional, multi-purpose designs are made locally in Brooklyn and are available in a variety of lacquers and wood veneers. We serve individual clients as well as architects, design professionals, and corporate accounts. *160 N. 4th St, between Bedford and Driggs Aves (L to Bedford Ave) 718-218-7750, wonknyc.com*

REALITY SHOPPING in WILLIAMSBURG
Laptop Potato: One who misses out on the "see me, feel me, touch me" experience of really shopping. Go beyond texting your buddies...meet them in Williamsburg's largest shopping area where buying stuff is an adventure that doesn't max out your credit cards.
The Graham Avenue Shopping District, Graham Ave. from Broadway to Boerum Street, Moore St., Debevoise St. and Flushing Avenue in Williamsburg, Brooklyn. *M/J train to Flushing Ave. L train to Montrose Ave. (a short walk towards the City) or use the B 13, 43, 46, 48, 57 or 60 buses. Walk, Peddle or Park....we'll be waiting!*

[redacted]

DUMBO

Rabbit Movers — Art Handler, Residential/Commercial Movers.
Let us move you! Our smart, handsome team will move your art, apartment, or office. We are fully licensed and insured. Our movers are quick and careful—strong and sprightly visual artists, writers, musicians. We charge by the hour and you will never find a "hidden fee." We cross our hearts! *Rabbitmovers.com, or call for an estimate: 718.852.2352.*

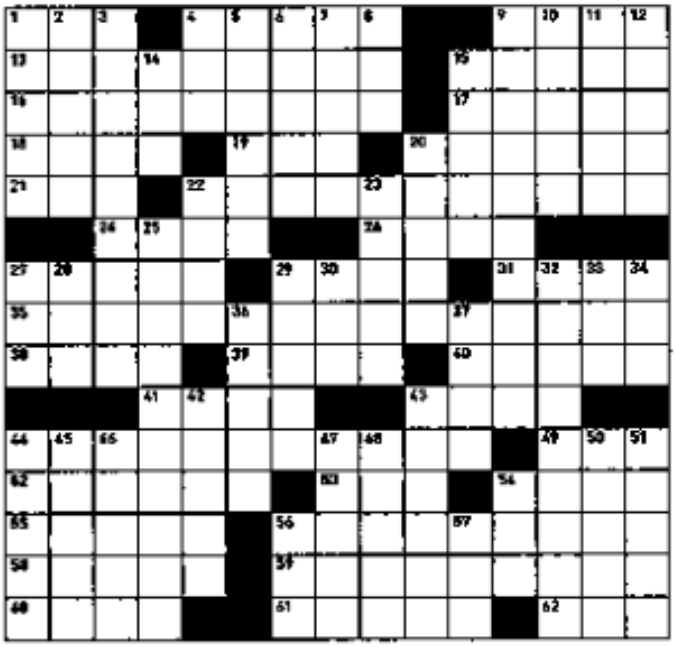
WILLIAMSBURG

Love or Money Recording — Quality Recording @ Reasonable Rates
New recording studio in Williamsburg — Pro Tools-based recording — nice-sized live room, vocal isolation booth, comfortable control room — engineer on staff — outside engineers: room rates available — bands: back-outs available — some rehearsal and solo practice slots open. *South 5th, Bedford/Berry — 646.472.5902 — recording@loveormoneyrecording.com*

crossword

we enable your OCD

Creative Liberties by Lee Kelly



ACROSS
1. Bodega sign
4. Exec hires
9. Not me!
13. Blackbeard's crew
15. Witchy one
16. African-American novel "translator" who aged on 1960s Paris?
17. Tattooed protagonist from *Memento*
18. Kurylenko, the newest Bond girl
19. 911 reactionary
20. Mollify
21. Geisha libation
22. *New York Times* "journalist" famed for more than one front-page story?
24. Vintage trucks that inspired a band name
26. Car wash device
27. Flavorpill output
29. Tina Fey's is prominent
31. Capt.'s subordinate
35. Sub-theme of this puzzle?
38. HahaAhaHahAHaha
39. Change course
40. Coped with
41. AC bill, essentially
43. *Back to the Future* bully
44. "Diarist" JT LeRoy's terminated alter ego of sorts?
49. Cutesy approvals
52. Make the grade
53. Brazilian soccer bros Sócrates and _____
54. Within many NY apartments, it's often small
55. Fictitiously represent
56. "Memoirist" with over a million fans?
58. Aspect
59. Uses Photoshop a bit too zealously
60. Spring St and Canal St
61. Aviation pioneer known for his pedal glider
62. Least populous U.S. state: Abbr.

DOWN
1. Top monastery dog
2. *A Confederacy of Dunces* swardee
3. Libation for washing down a buncha tacos
4. Bakery hrs.
5. Common word processing command
6. Hinder (vs.)
7. Milk dispensers
8. Job app. requisite
9. Influential staff member?
10. Odyssey undertaker?
11. Utter boredom
12. *Twilight* series scribe
14. Familia member
15. Palladium, Danceteria and Studio 54
20. Visit a crack house
22. Nicole Richie's boyfriend
23. Atheist activist Madalyn Murray
25. Celeb hounders
27. Homonym for this mag
28. Nickelodeon's "Invader _____"
29. Surfer's mount
30. Benicio Del Toro role
32. Woody Allen married her adopted daughter
33. _____ tha Funkee Homosapien
34. Q tip?
36. Bottled water choice
37. Polish up
42. Cantonwale
43. Prickly plants
44. Funny schtuff
45. Irked to no end
46. Brooklyn-bound 3 train express stop
47. "Play one more song!"
48. Design duo lionized for their chairs
50. Far from happy
51. Full consent
54. Where humans likely come from: Abbr.
56. Rent check enabler
57. Bill Nye subj.

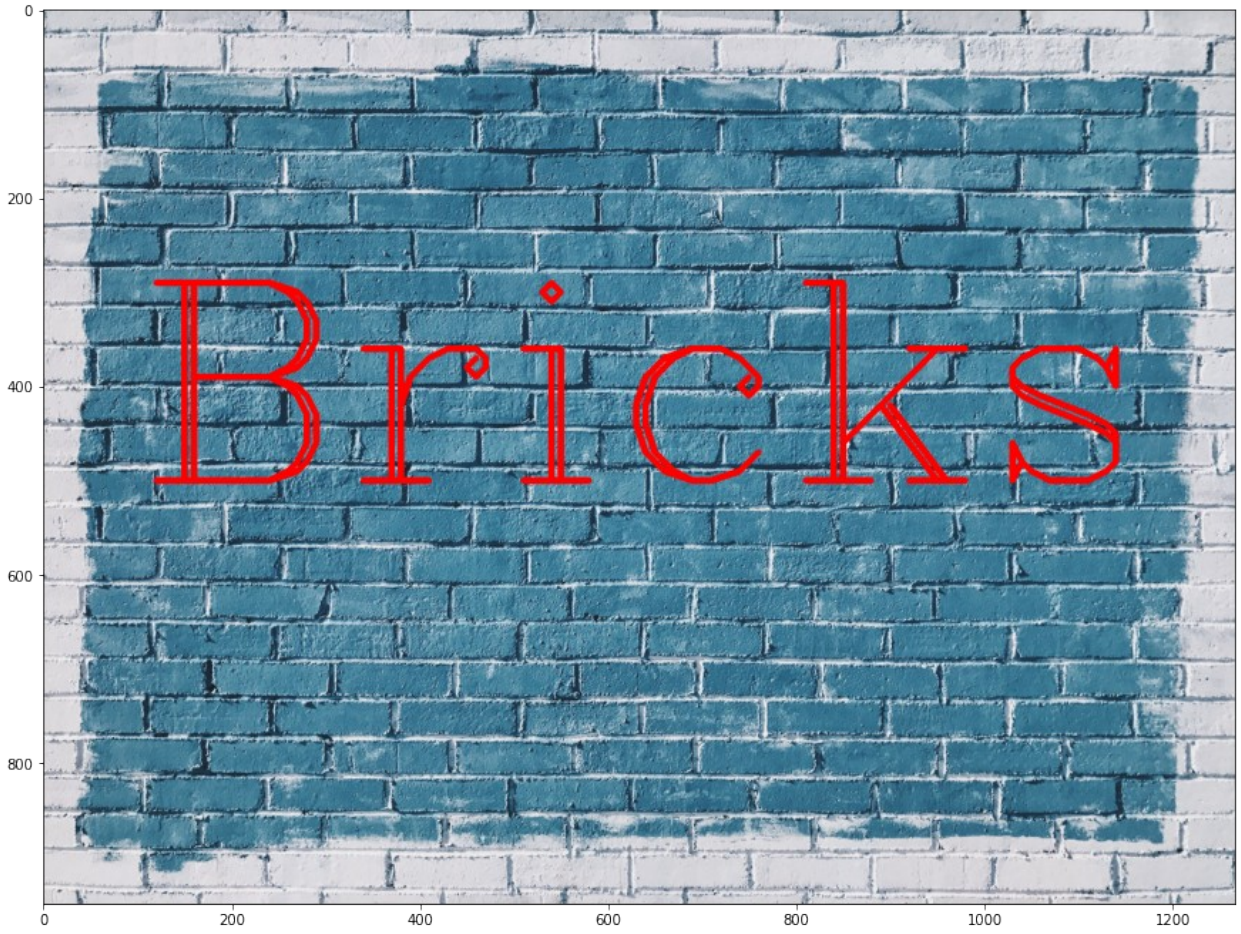
Blurring/Smoothing Images

Blurring and smoothing images can help in reducing the noise in an image and smoothen and enhance the quality of the image. This is truly helpful for edge detection as that can reduce the number of unwanted edges on the image.

```
def load_img():
    img = cv2.imread('/Users/ruben/Desktop/Learning/Computer-Vision-
with-Python/DATA/bricks.jpg').astype(np.float32)/255
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    font = cv2.FONT_HERSHEY_COMPLEX

cv2.putText(img,'Bricks',org=(100,500),fontFace=font,fontScale=10,colo
r=(255,0,0),thickness=5,lineType=cv2.LINE_4)
    return img

def display_pic(img):
    plt.figure(figsize=(15,15))
    return plt.imshow(img)

img = load_img()
display_pic(img)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd2841748>
```

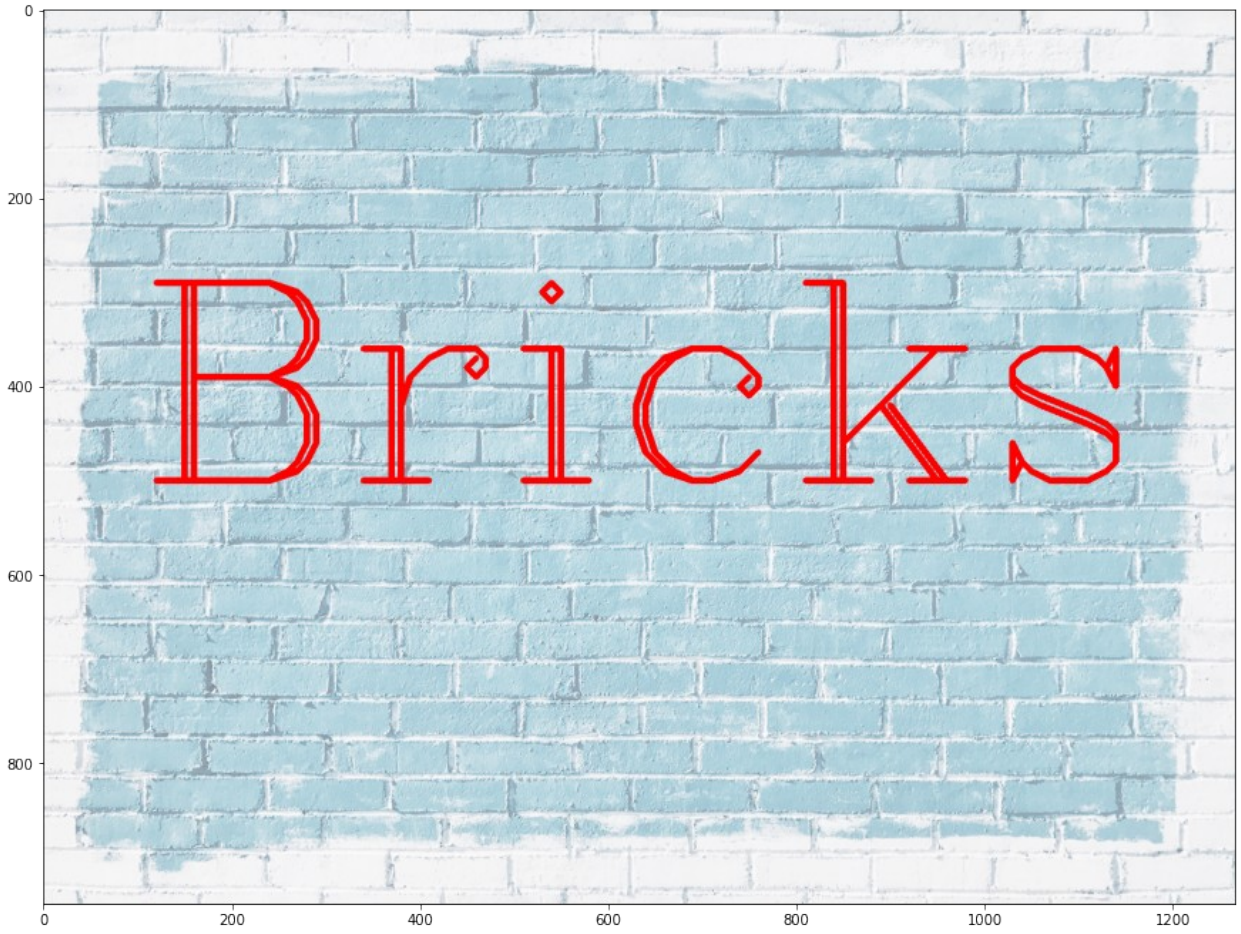


Now that we have obtained the picture, lets introduce the gamma correction filtering of the image. This process just involves making the image lighter or darker. the gamma correction involves just increamenting the intensity values of the image to the power of the gamma factor.

```
gamma=1/4
img_gamma = np.power(img,gamma)
display_pic(img_gamma)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd0e7d400>
```




```
gamma=2
img_gamma = np.power(img,gamma)
display_pic(img_gamma)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd01c4f28>
```

Moving on to kernel based filters, this provides kind of blurring techniques to blur the background. Blurring kernel is matrix by which a matrix is multiplied the origijnal image to recover the final image. The kernel is multiplied element wise by a submatrix and summed up to form the final image pixel.

As in one-dimensional signals, images also can be filtered with various low-pass filters (LPF), high-pass filters (HPF), etc. LPF helps in removing noise, blurring images, etc. HPF filters help in finding edges in images.

OpenCV provides a function cv.filter2D() to convolve a kernel with an image. As an example, we will try an averaging filter on an image. A 5x5 averaging filter kernel will look like the below:

```
kernel = np.ones(shape=(5,5),dtype=np.float32)/25
kernel
```

```
array([[0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04]], dtype=float32)
```

```
img_blur = cv2.filter2D(img,-1,kernel=kernel)
display_pic(img_blur)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).

<matplotlib.image.AxesImage at 0x7f9d45f9a780>
```

As its shown, the color is blurred within. the image, with the shapes looking a little bit less sharp. This is an example of an image being filtered with a low pass filter. Now we will perform image blurring directly with the averaging method. This is done by convolving an image with a normalized box filter. It simply takes the average of all the pixels under the kernel area and replaces the central element.

Note:We take `ddepth` as '-1' because we want the input desired depth to be same as the output desired depth.

```
img = load_img()
display_pic(img)
```

```
img_blur = cv2.blur(img,ksize=(5,5))
display_pic(img_blur)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd2502940>
```

Now there are more blurring techniques which go beyond just blurrring the entire image. The problem with the previous techniques are that blurring is done all the pixels which will modify some important details in the pic. Lets experiment with Gaussian Blur and Median Blur.

```
img = load_img()
print('reset')
```

```
reset
```

```
img_blur = cv2.GaussianBlur(img,(5,5),10)
display_pic(img_blur)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9d3563a518>
```

This has produced results similar to the previous ones. However, **Gaussian filtering is primarily done when resizing an image** as this is a low pass filter before reaampling. This is done in order to nullify any high frequency components in the downsampled image, and removes any sharp edges in the image making a clear image to look at.

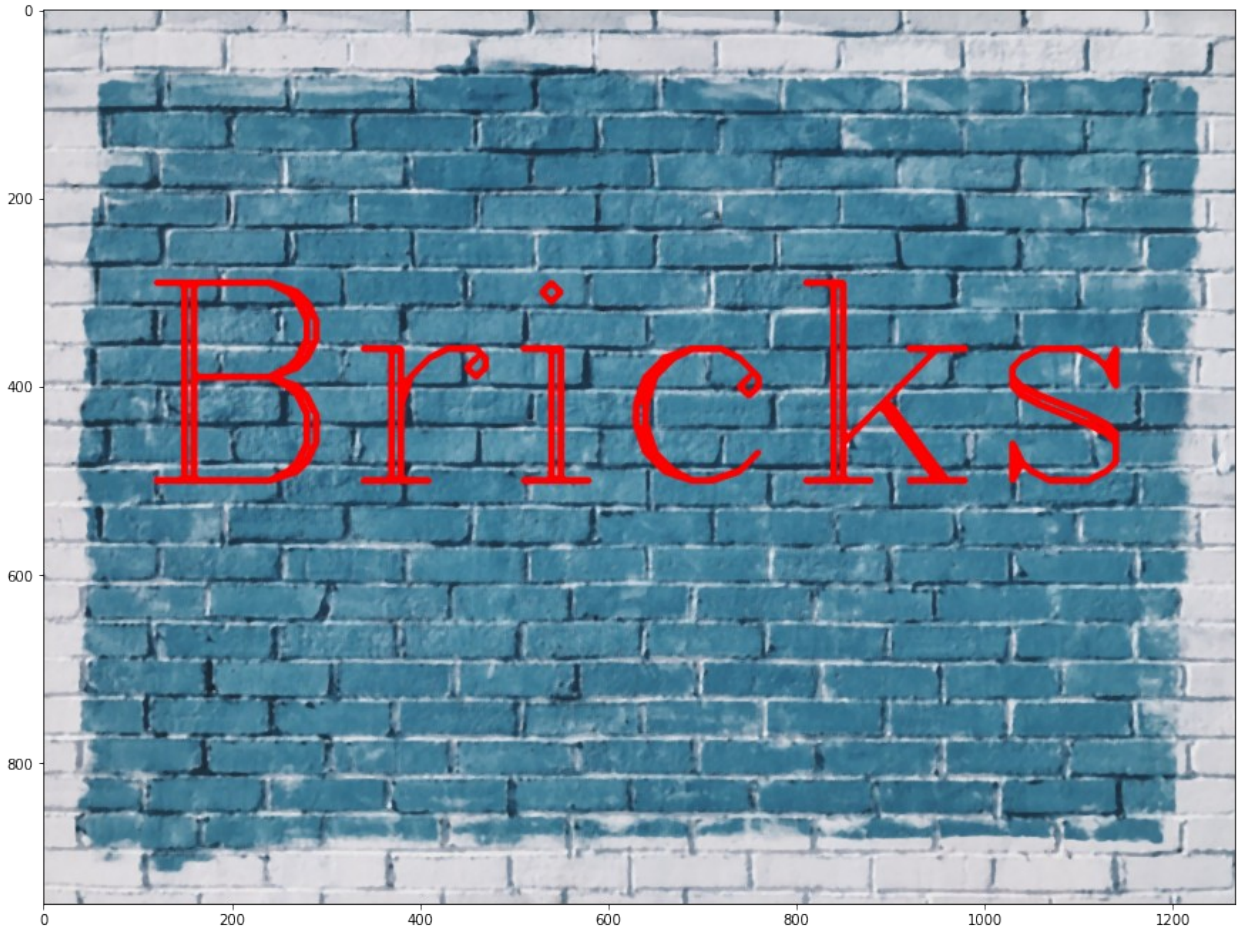
```
img = load_img()
print('reset')
```

```
reset
```

```
img_blur = cv2.medianBlur(img,ksize=5)
display_pic(img_blur)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd7599630>
```



Hence Median blur is the best option for noise filtering from an image as unlike the previous filtering techniques, the pixel values are always changed hence effectively reducing the noise significantly. **Bilateral filtering** also helps reduce noise but takes a lot of computation time.

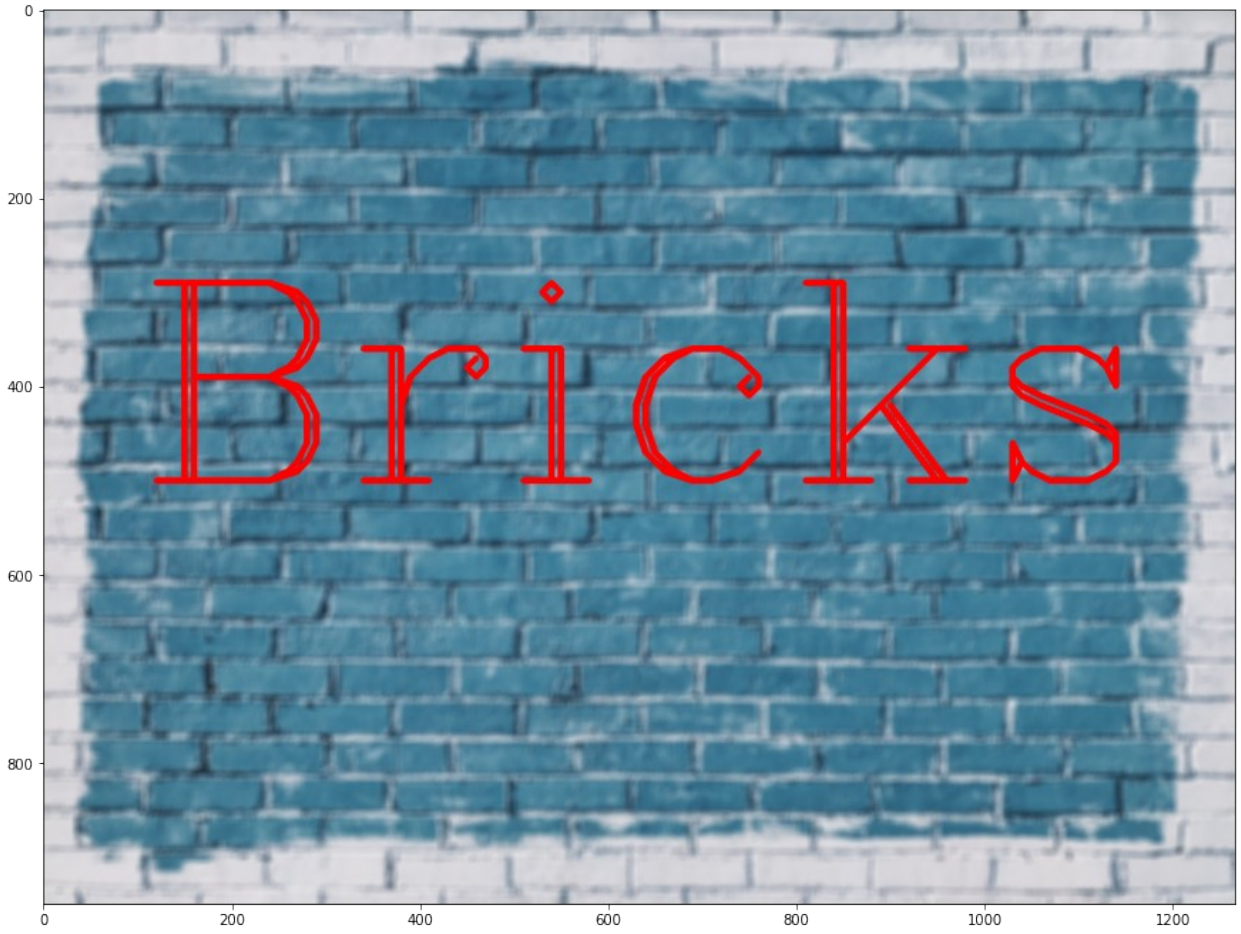
```
img = load_img()
print('reset')
```

```
reset
```

```
img = cv2.bilateralFilter(img,10,25,75)
display_pic(img)
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7f9cd0e2af28>
```



As you can notice, the edges have come out a little more distinct and smoothly while blurring and smoothening the background. What this filter essentially does that it computes two weights for each neighbour, the spatial weight and range weight. The range weight as the previous filters, changes the value the pixel to the weighted average of the neighbourhood pixels while the spatial weight computes the difference between the pixel and the neighbour. Hence any distant pixels are penalized such that slow varying pixels are only smoothened while maintaining the sharp distinct edges.

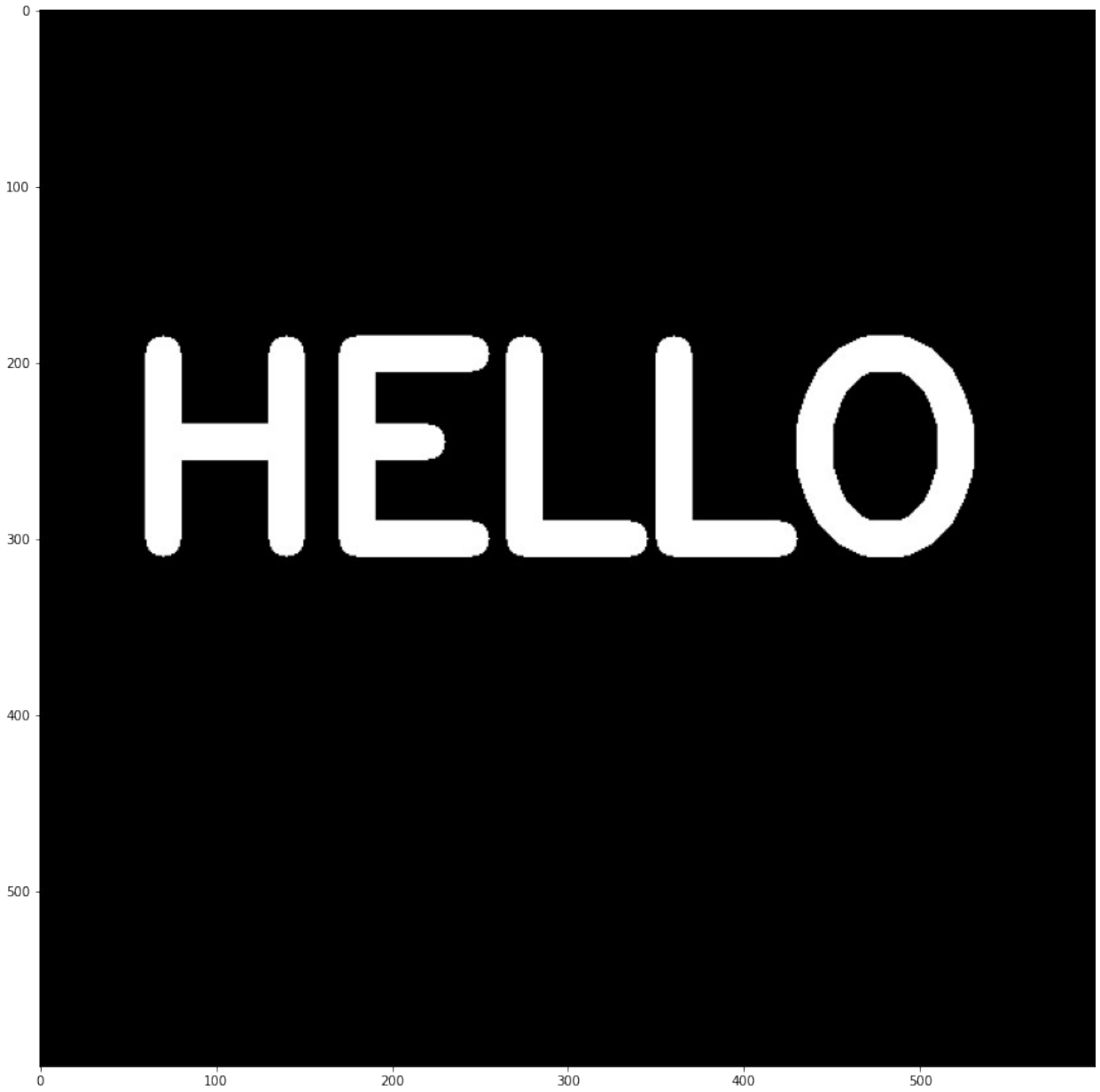
Morphological Operators

These kind of operators provides special effects to the quality of the foreground and background of the image.

```
def load_synth():
    img = np.zeros(shape=(600,600))
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(img,'HELLO',
(50,300),fontFace=font,fontScale=5,color=(255,255,255),thickness=20,li
neType=cv2.LINE_8)
    return img
```

```
def display_pic(img):
    fig = plt.figure(figsize=(15,15))
    ax = fig.add_subplot(111)
    return plt.imshow(img,cmap='gray')

display_pic(load_synth())
```

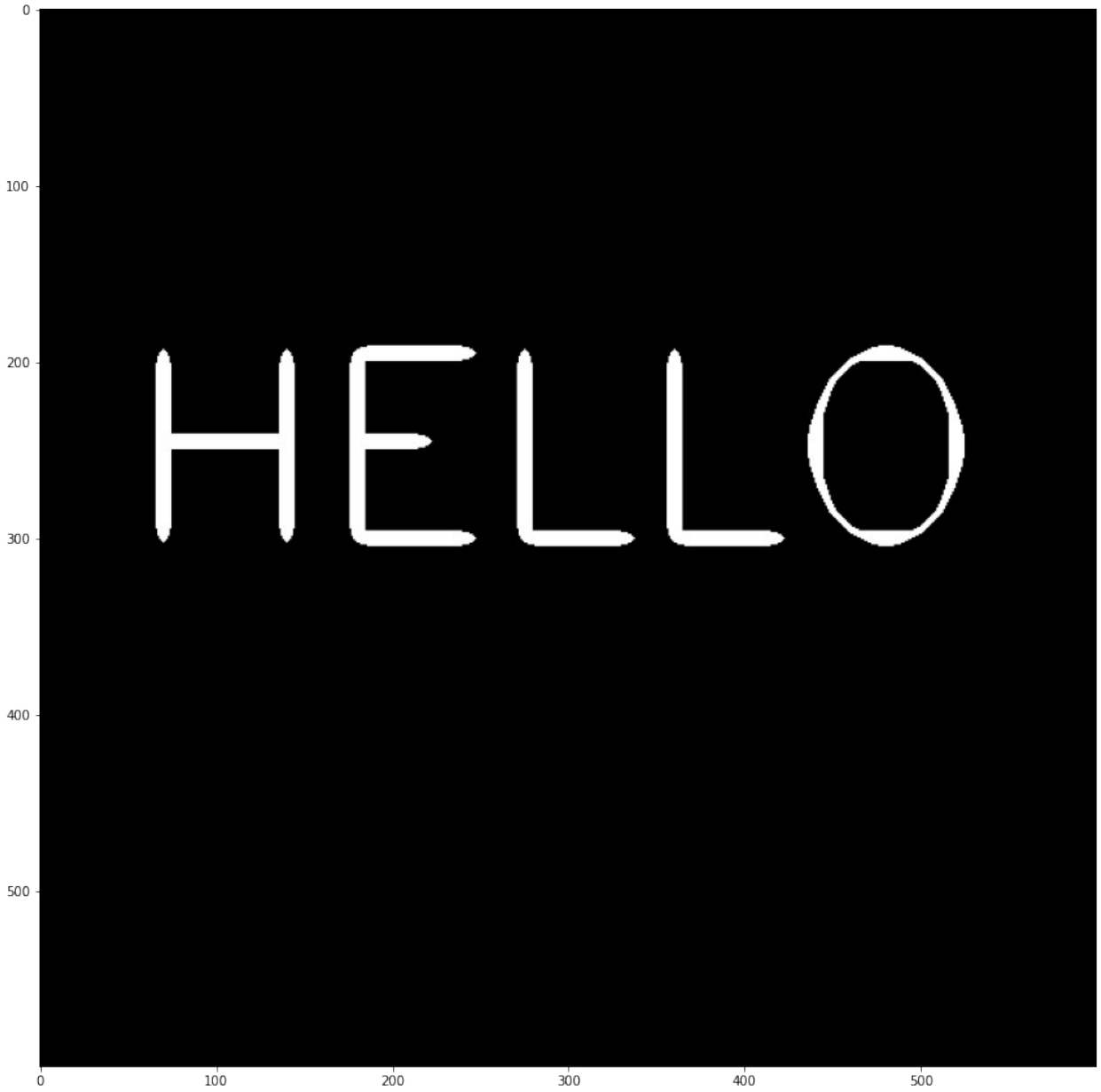
```
<matplotlib.image.AxesImage at 0x7f9cd256ebe0>
```




```
img = load_synth()

kernel = np.ones((5,5),dtype=np.int16)
```

```
img_erode = cv2.erode(img,kernel,iterations=3)
display_pic(img_erode)
```

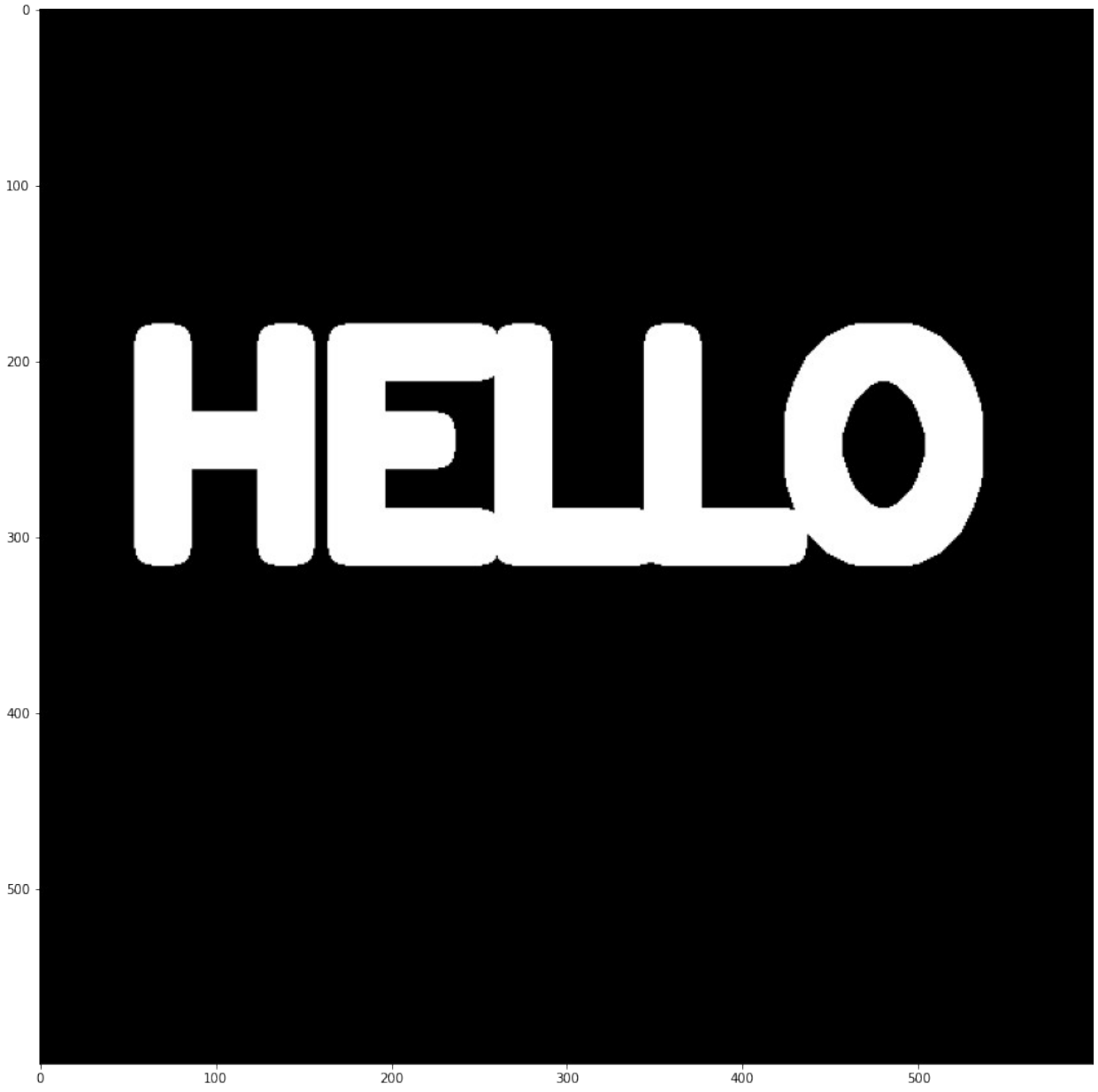
```
<matplotlib.image.AxesImage at 0x7f9cd7583080>
```



The above is called the **Erosion** operator, lets perform **Dilation** which will add more edge thickness to the text

```
img_morph =
cv2.morphologyEx(img,cv2.MORPH_DILATE,kernel=kernel,iterations=3)

display_pic(img_morph)
```

```
<matplotlib.image.AxesImage at 0x7f9ce0017198>
```



As you can see, the Dilation operator has add the thickness along the edges. Now lets check the **Opening** operator which essentially removes background noise from the foreground image.

```
img = load_synth()

white_noise =
np.random.randint(low=0,high=2,size=(600,600),dtype=np.int16)

white_noise
```

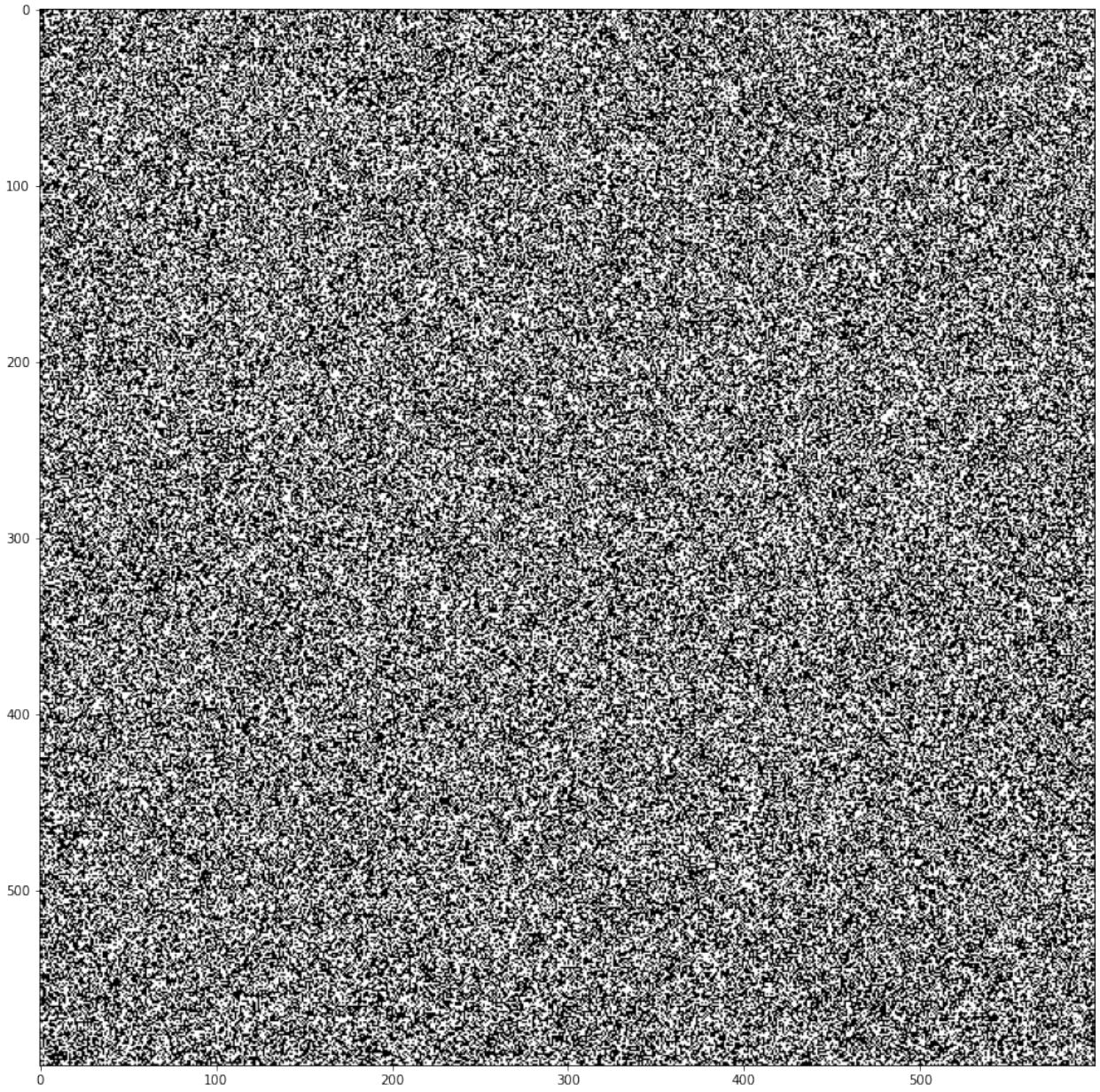
```
array([[0, 0, 0, ..., 1, 0, 1],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 1, 1, 0],
       ...,
       [0, 0, 0, ..., 1, 0, 0],
       [0, 1, 1, ..., 0, 1, 1],
       [0, 1, 0, ..., 1, 0, 1]], dtype=int16)
```

```
img.max()
```

```
255.0
```

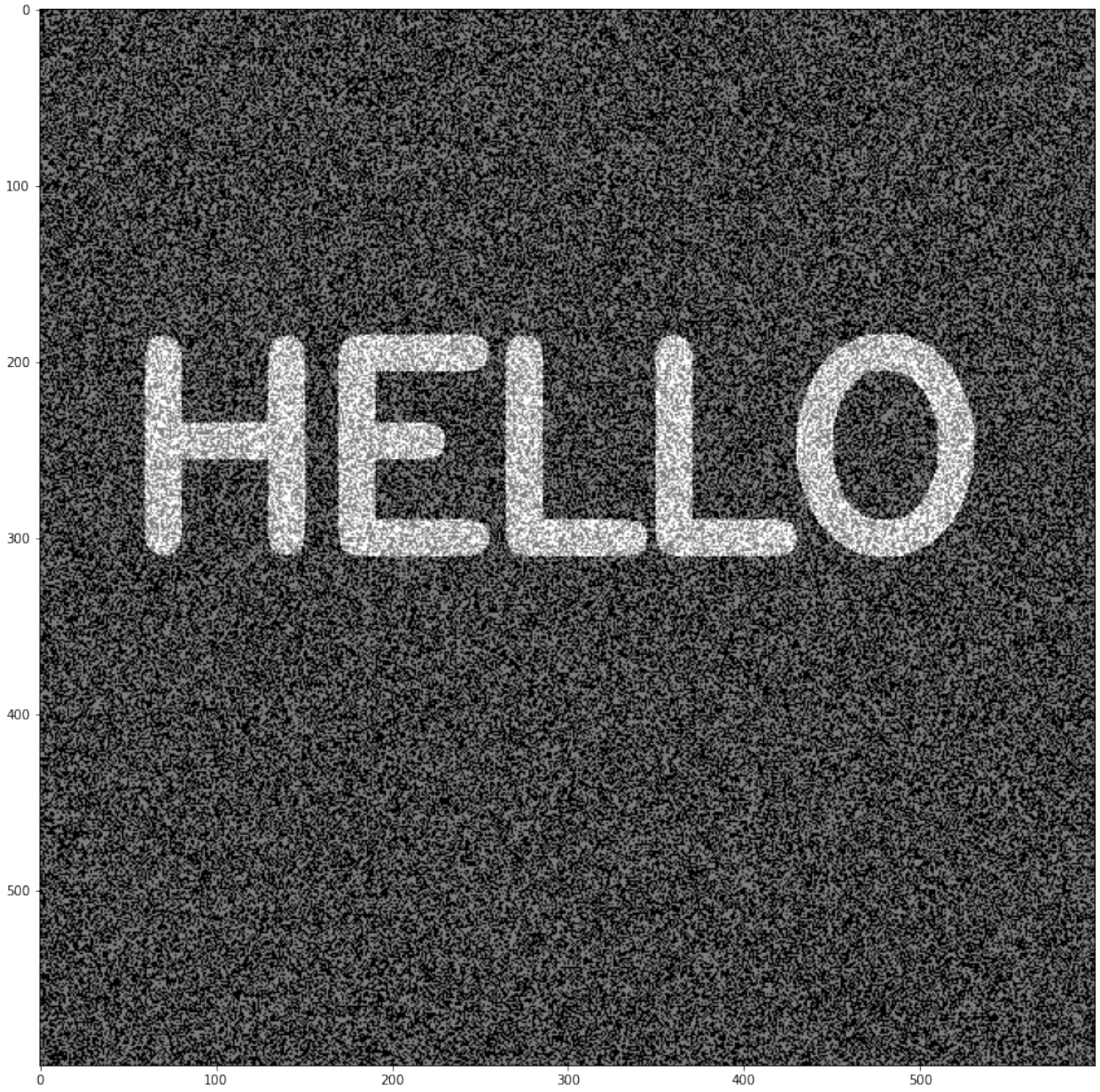
```
white_noise = white_noise * 255
display_pic(white_noise)
```

```
<matplotlib.image.AxesImage at 0x7f9d45b7d208>
```




```
white_noise_img = img + white_noise
display_pic(white_noise_img)
```

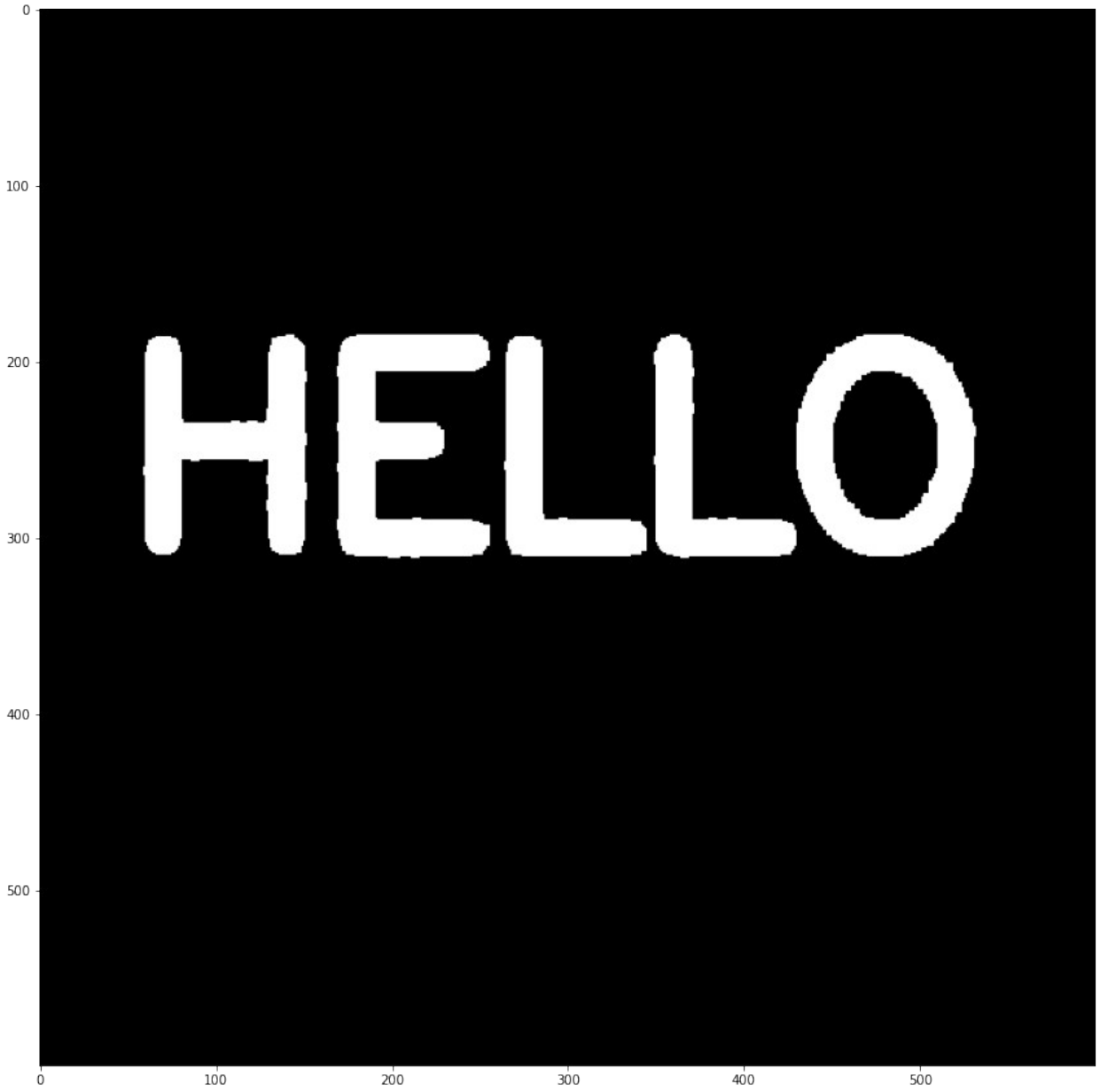
```
<matplotlib.image.AxesImage at 0x7f9d46067518>
```



This is called background noise on the image.

```
img_morph =
cv2.morphologyEx(white_noise_img,cv2.MORPH_OPEN,kernel=kernel)
display_pic(img_morph)
```

```
<matplotlib.image.AxesImage at 0x7f9cd26565c0>
```



This is the **Opening operator**. What this essentially does is that it clears the noise from the background from the foreground and making the text clear and visible. Now lets see the Closing operator. For that we need to create some noise in the **foreground** of the image.

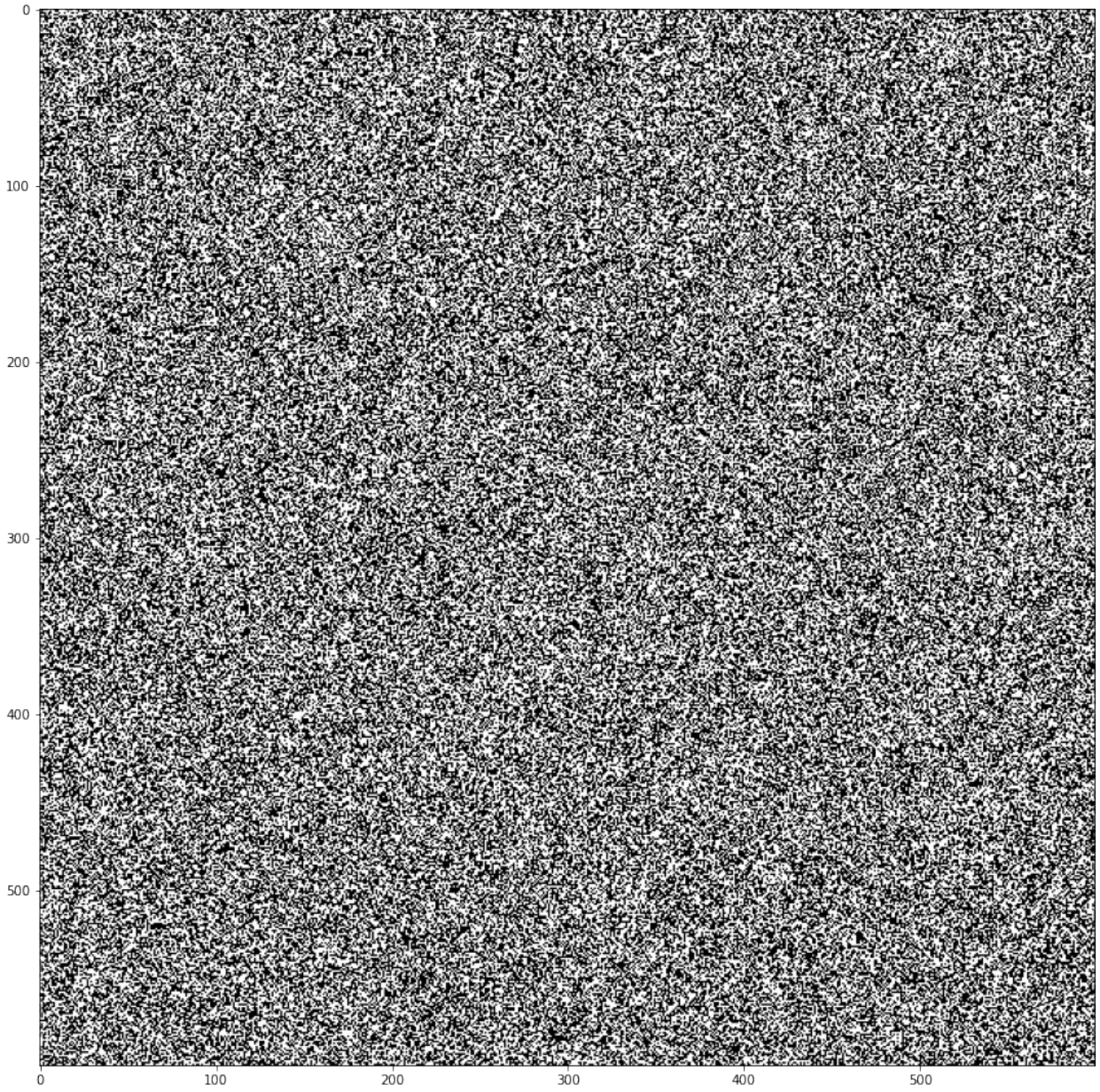
```
img = load_synth()

img
```

```
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.]])

black_noise = np.random.randint(low=0,high=2,size=(600,600))
```

In order to get noise in the foreground alone, we need to multiply the noise array by -255 so as to get the inverse of the noise in the background.

```
black_noise = black_noise * -255

black_noise

array([[   0, -255, -255, ..., -255, -255, -255],
       [   0, -255,    0, ...,    0,    0, -255],
       [-255, -255, -255, ...,    0, -255,    0],
       ...,
       [-255,    0,    0, ..., -255, -255, -255],
       [   0,    0, -255, ..., -255, -255, -255],
       [-255,    0, -255, ...,    0,    0, -255]])

display_pic(black_noise)

<matplotlib.image.AxesImage at 0x7f9cd0ecd0f0>
```



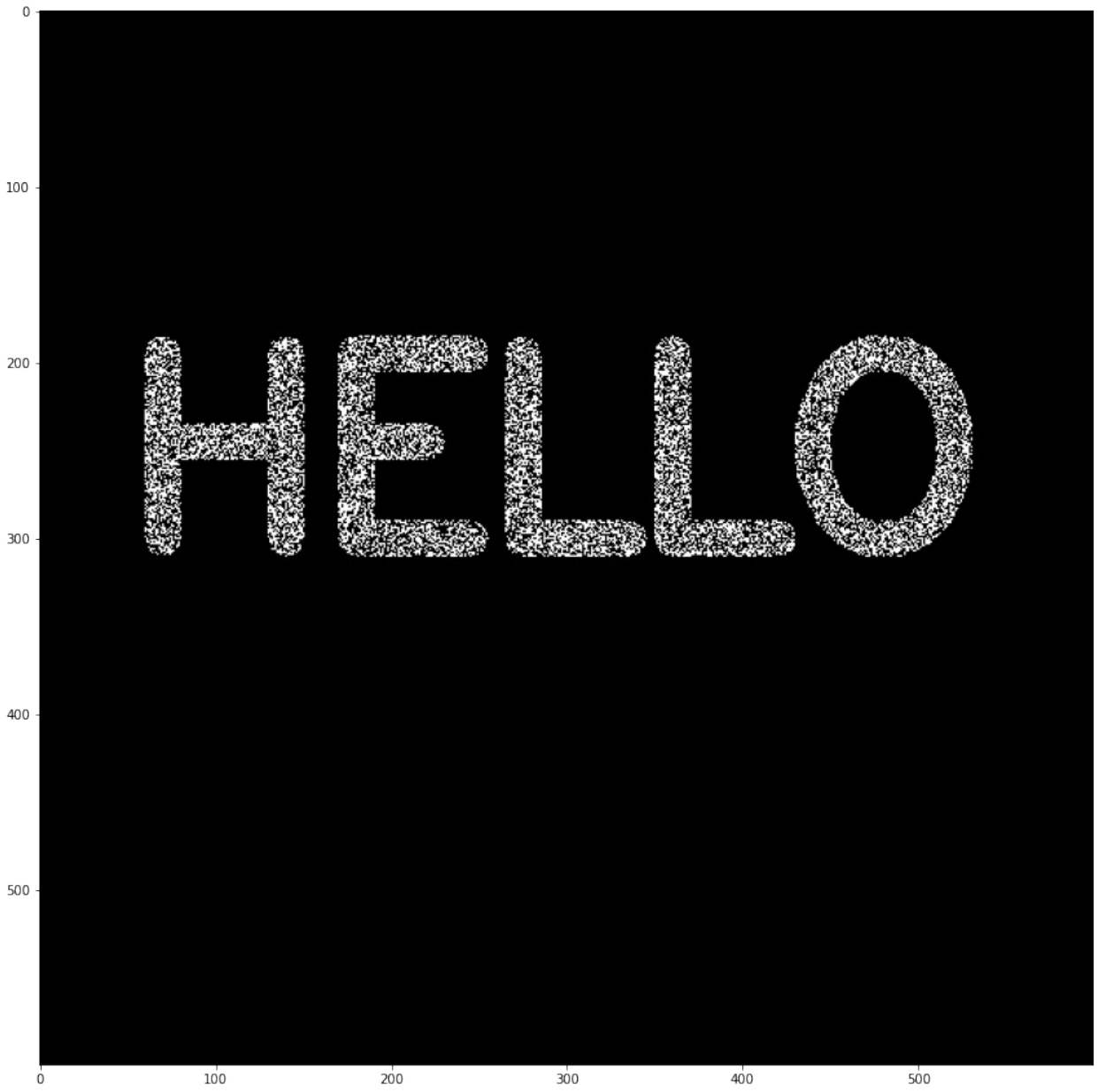

```
black_noise_img = black_noise + img

black_noise_img
```

```
array([[   0., -255., -255., ..., -255., -255., -255.],
       [   0., -255.,    0., ...,    0.,    0., -255.],
       [-255., -255., -255., ...,    0., -255.,    0.],
       ...,
       [-255.,    0.,    0., ..., -255., -255., -255.],
       [   0.,    0., -255., ..., -255., -255., -255.],
       [-255.,    0., -255., ...,    0.,    0., -255.]])
```

Pixel values can never be negative hence they will be floored to 0

```
black_noise_img[black_noise_img==-255]=0

display_pic(black_noise_img)
```

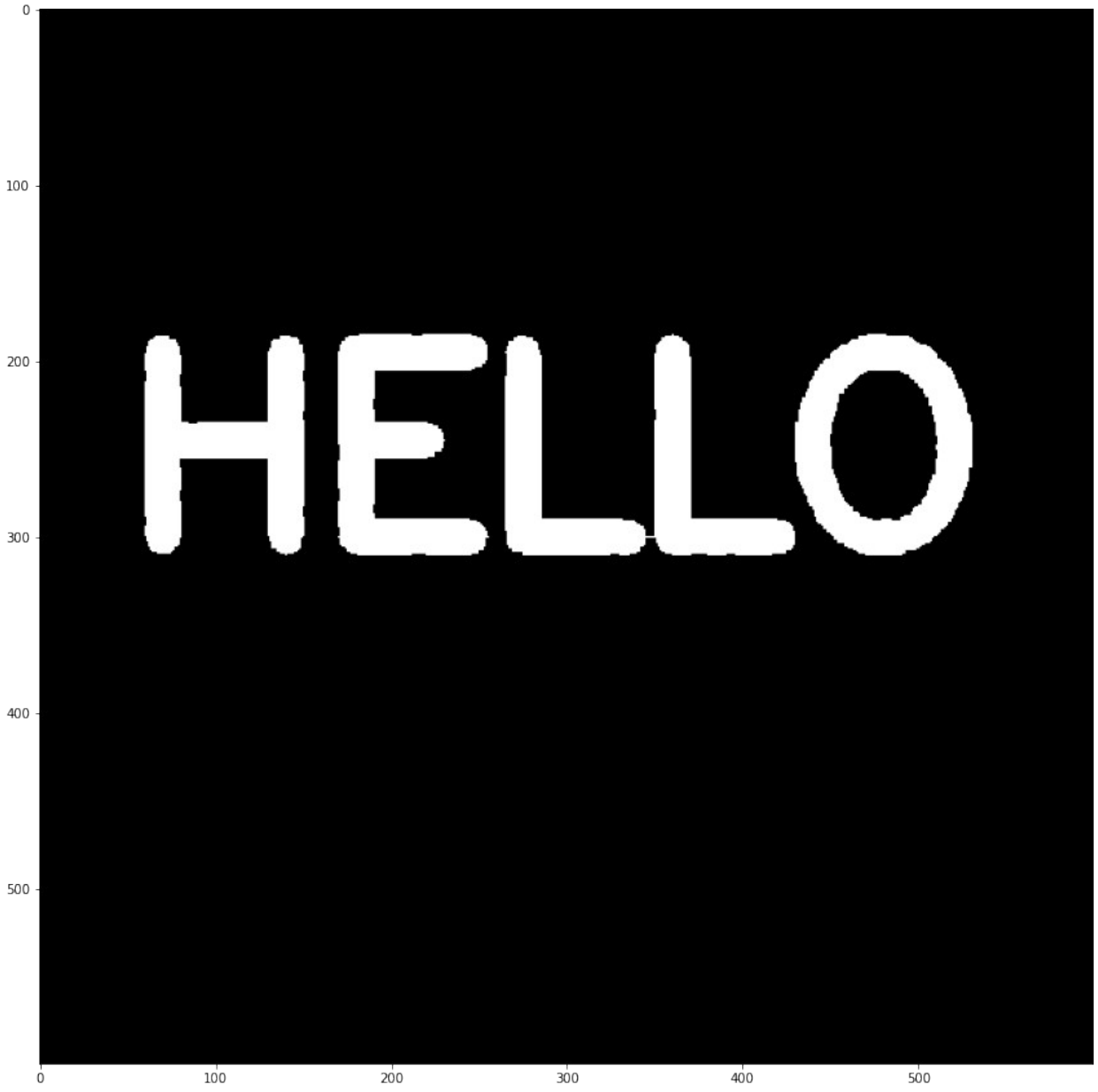
```
<matplotlib.image.AxesImage at 0x7f9cd387dbe0>
```



Now for this, we will be using **Closing operator** which essentially removes the noise from the foreground of the image.

```
img_morph =
cv2.morphologyEx(black_noise_img,cv2.MORPH_CLOSE,kernel=kernel)

display_pic(img_morph)
```

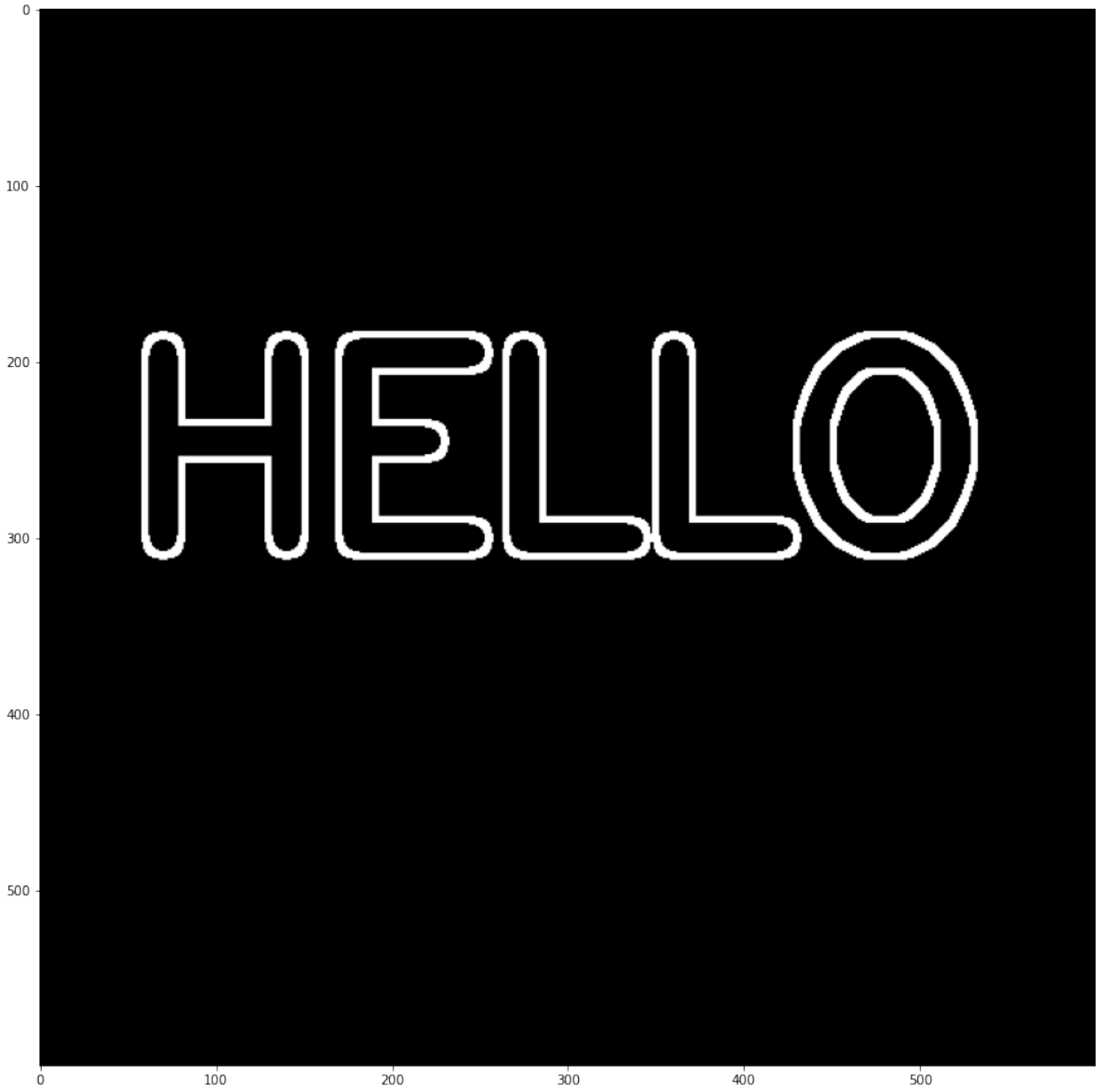
```
<matplotlib.image.AxesImage at 0x7f9cc063e048>
```



Finally we have the **Gradient** operator which is the difference of the Erosion and Dilation operators i.e. it only displays the hollow edges of the text.?

```
img_morph = cv2.morphologyEx(img,cv2.MORPH_GRADIENT,kernel=kernel)
display_pic(img_morph)
```

```
<matplotlib.image.AxesImage at 0x7f9cb432d2b0>
```

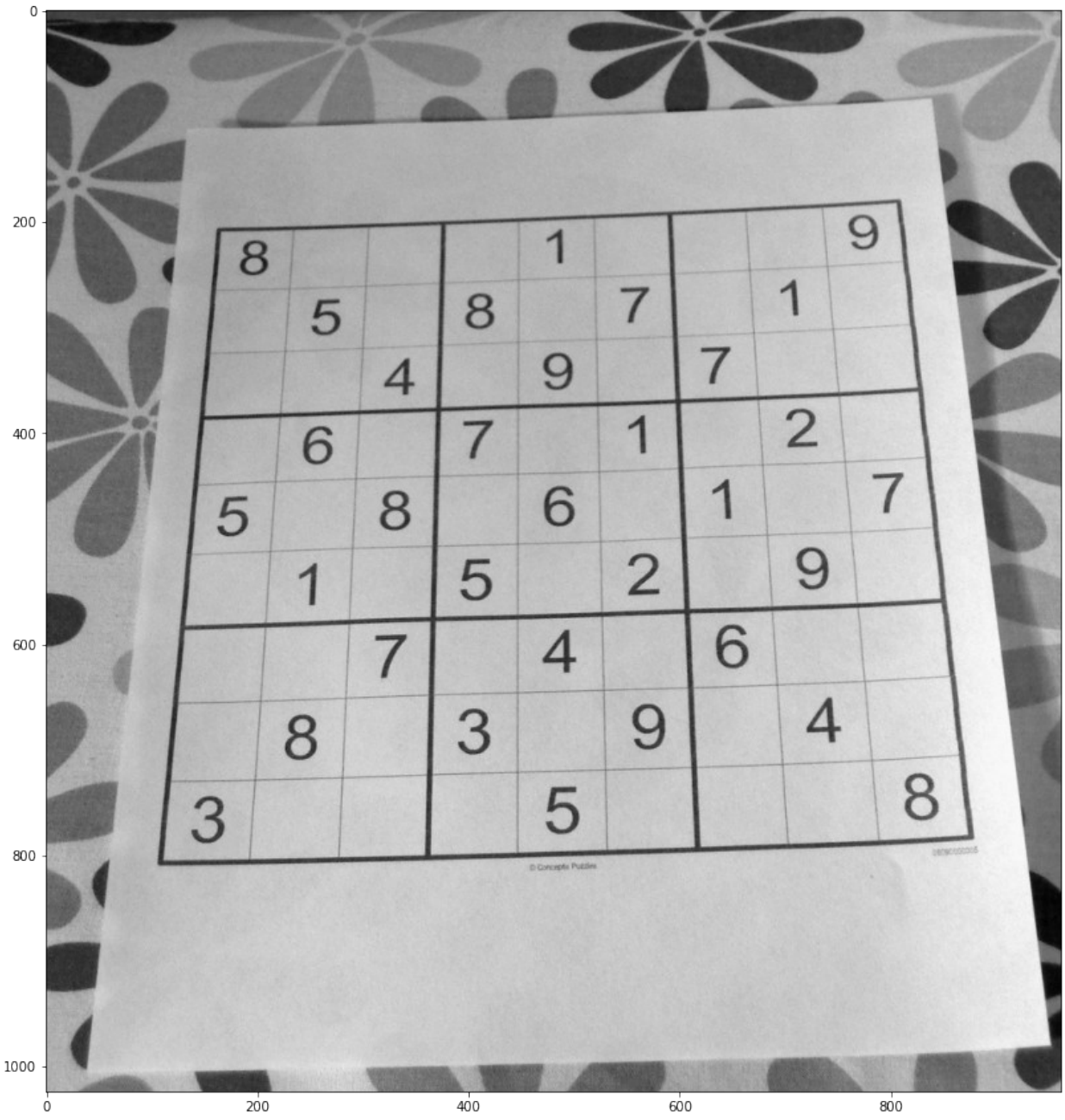


Gradients

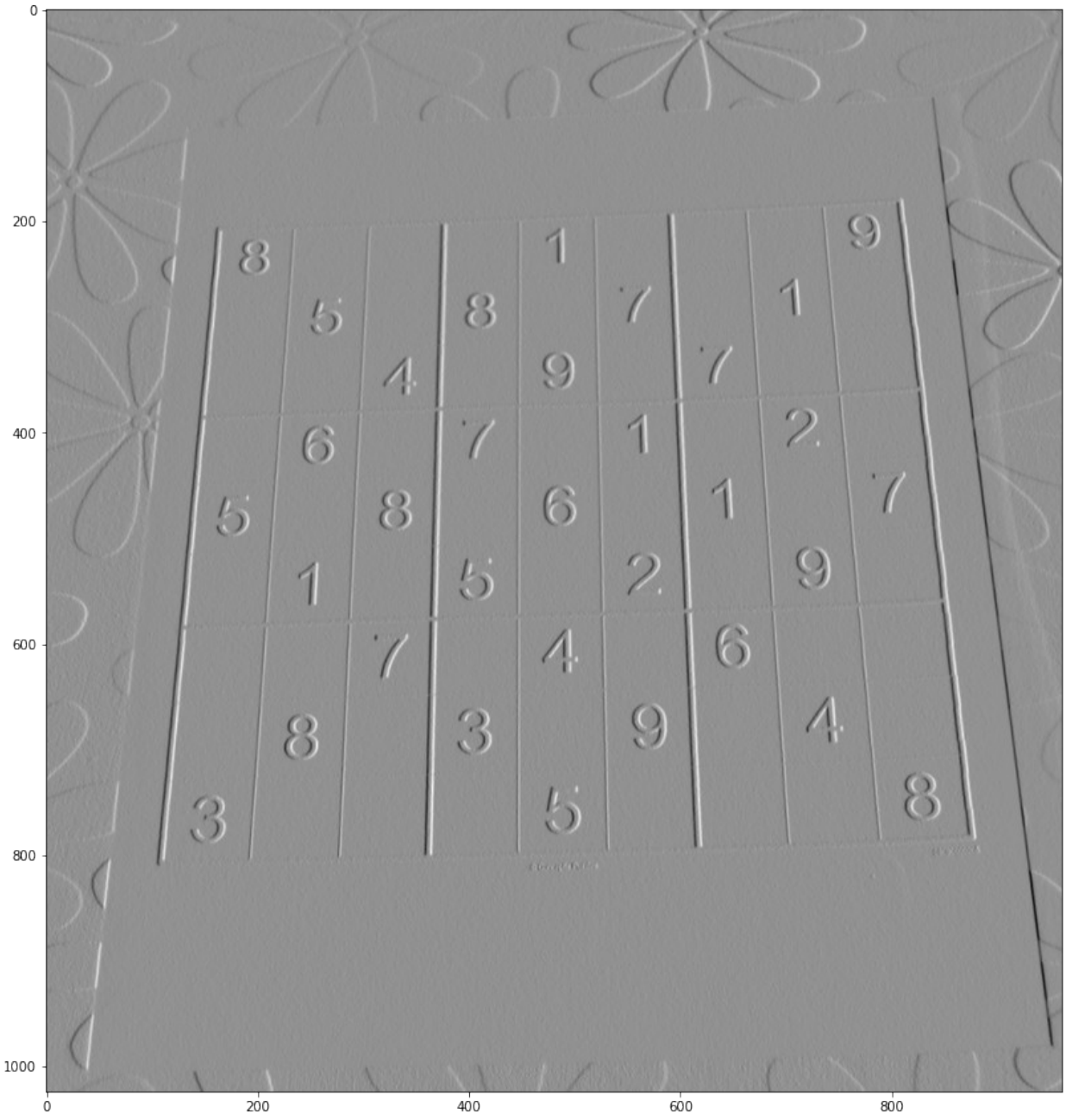
Gradients is the calculation of the rate of change of color intensity in a specific direction. We use the Sobel-Friedman operators to calculate our gradients. Through this method, we can obtain horizontal edges, vertical edges or both which is the foundational stone of edge detection.

```
def display_pic(img):
    """
    Display the image as part of a subplot on a 15x15 canvas.
    """
    fig = plt.figure(figsize=(15,15))
```

```
    ax = fig.add_subplot(111)
    ax.imshow(img,cmap='gray')

img = cv2.imread('/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/sudoku.jpg',0)
display_pic(img)
```

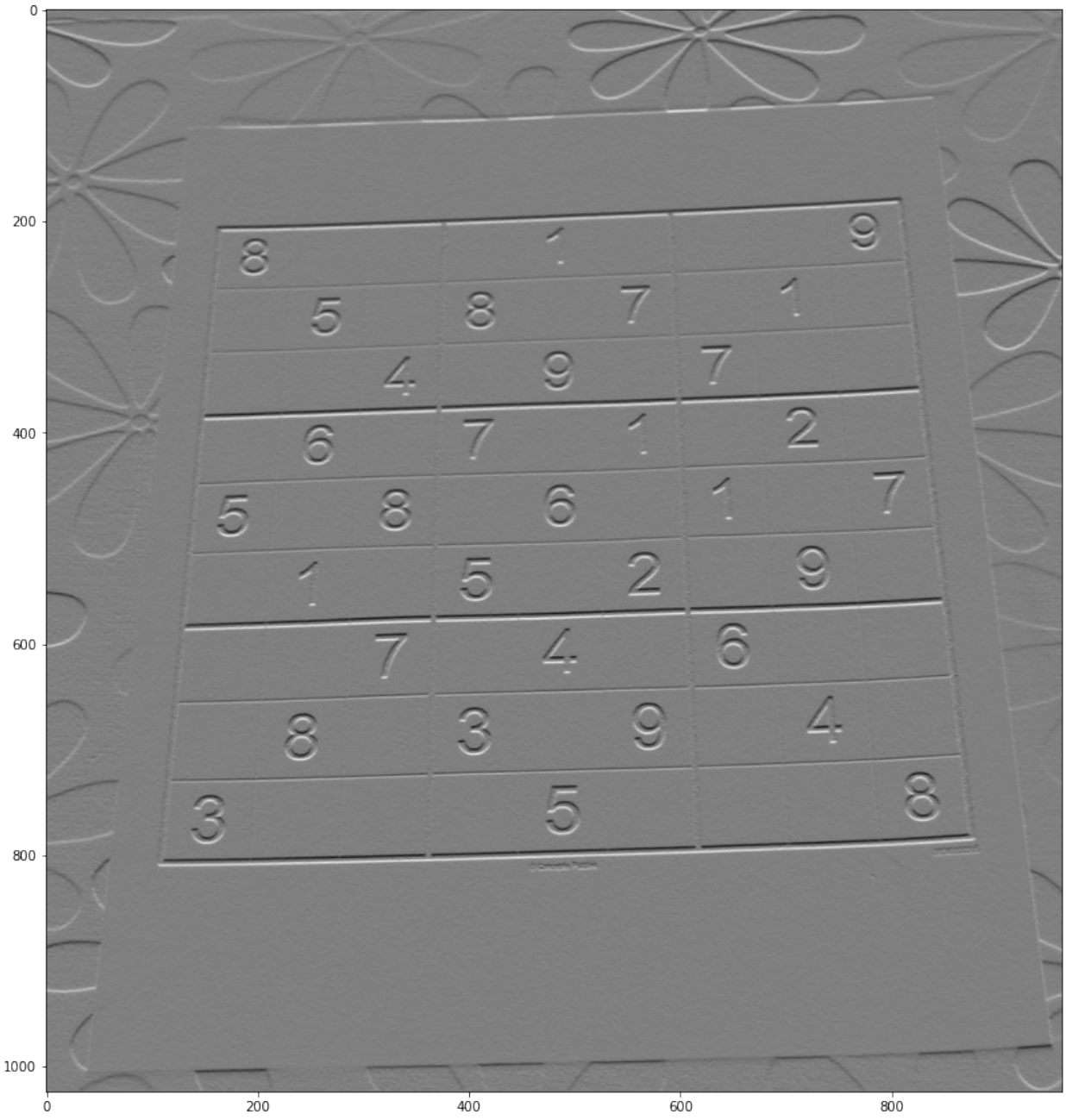



```
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0)
display_pic(sobelx)
```



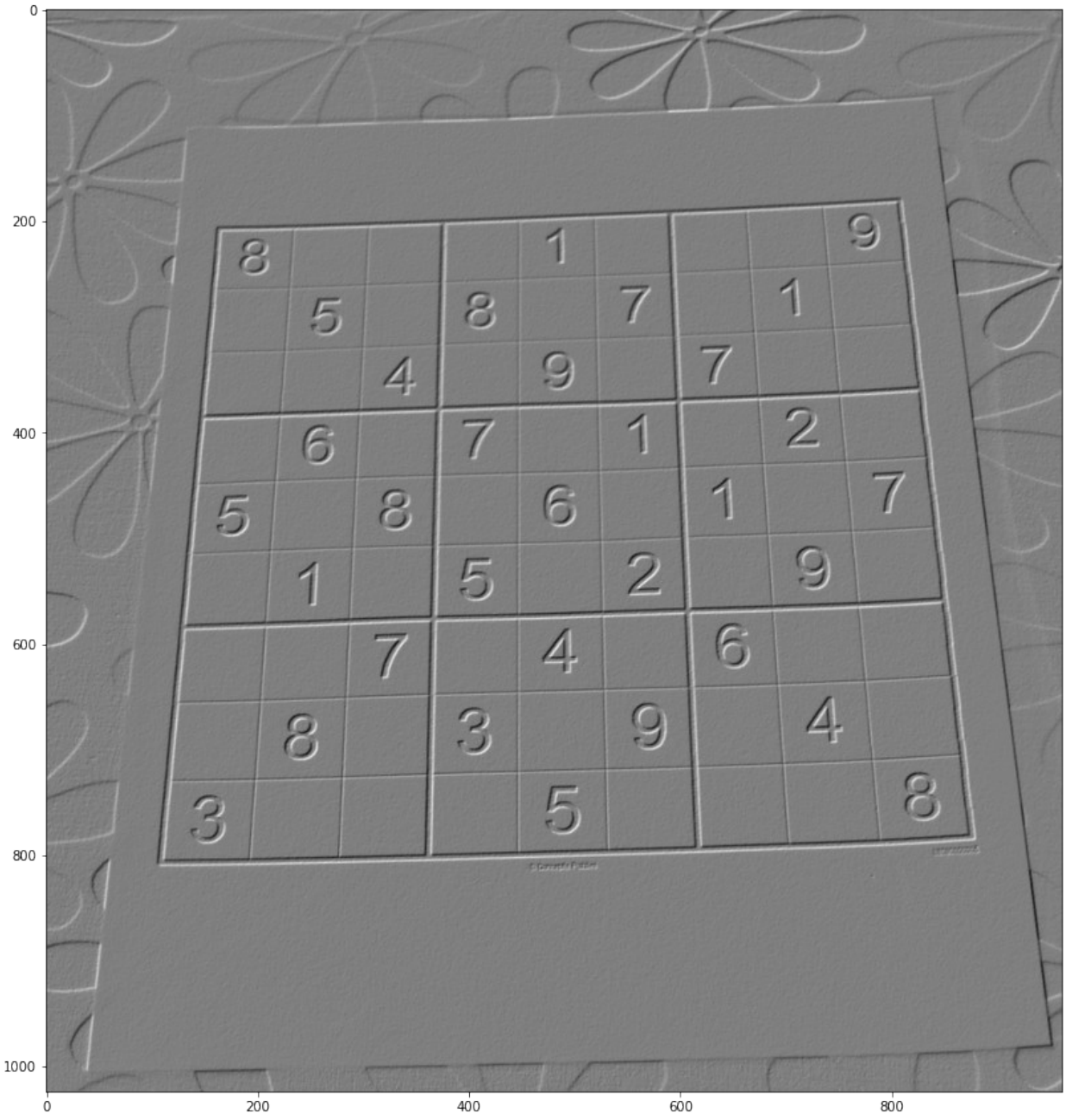
The sobelx operator is used to find the vertical edges of the image. Here as we can see the gradient of the x makes the color intensity lines very low as the derivative of x line is almost 0.

```
sobely = cv2.Sobel(img,cv2.CV_64F,0,1)
display_pic(sobely)
```



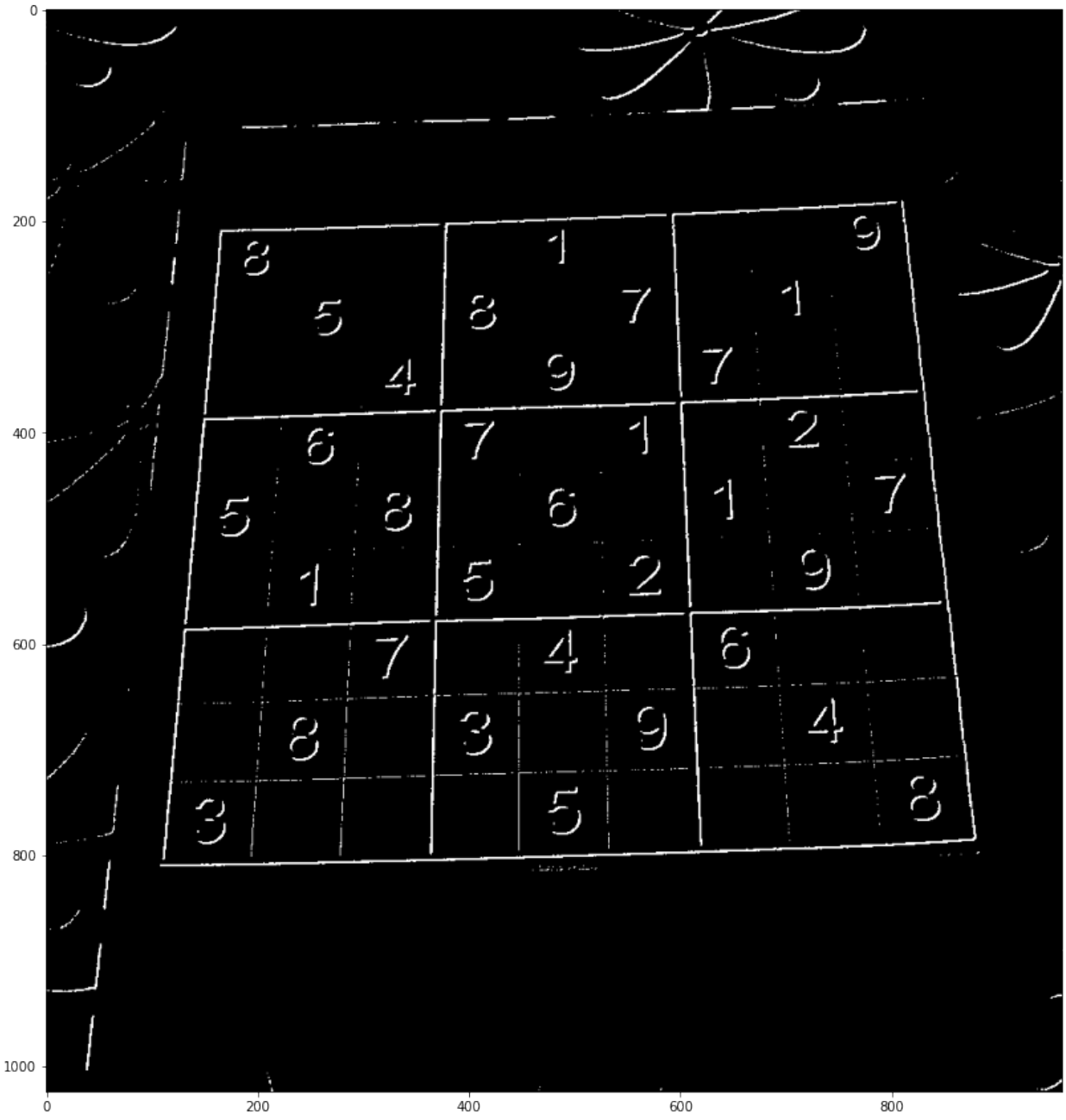
Similarly for the y gradient, the horizontal lines will be visible as the gradient of intensities in the y direction is almost 0. Lets combine both the gradients through the process of blending an image. Here we will give weightage to both the images.

```
blended =
cv2.addWeighted(src1=sobelx,alpha=0.5,src2=sobely,beta=0.5,gamma=0)
display_pic(blended)
```



Perfect!!, we can almost perform edge detection over here. We could further make it better by adding a binary threholding to this blended image.

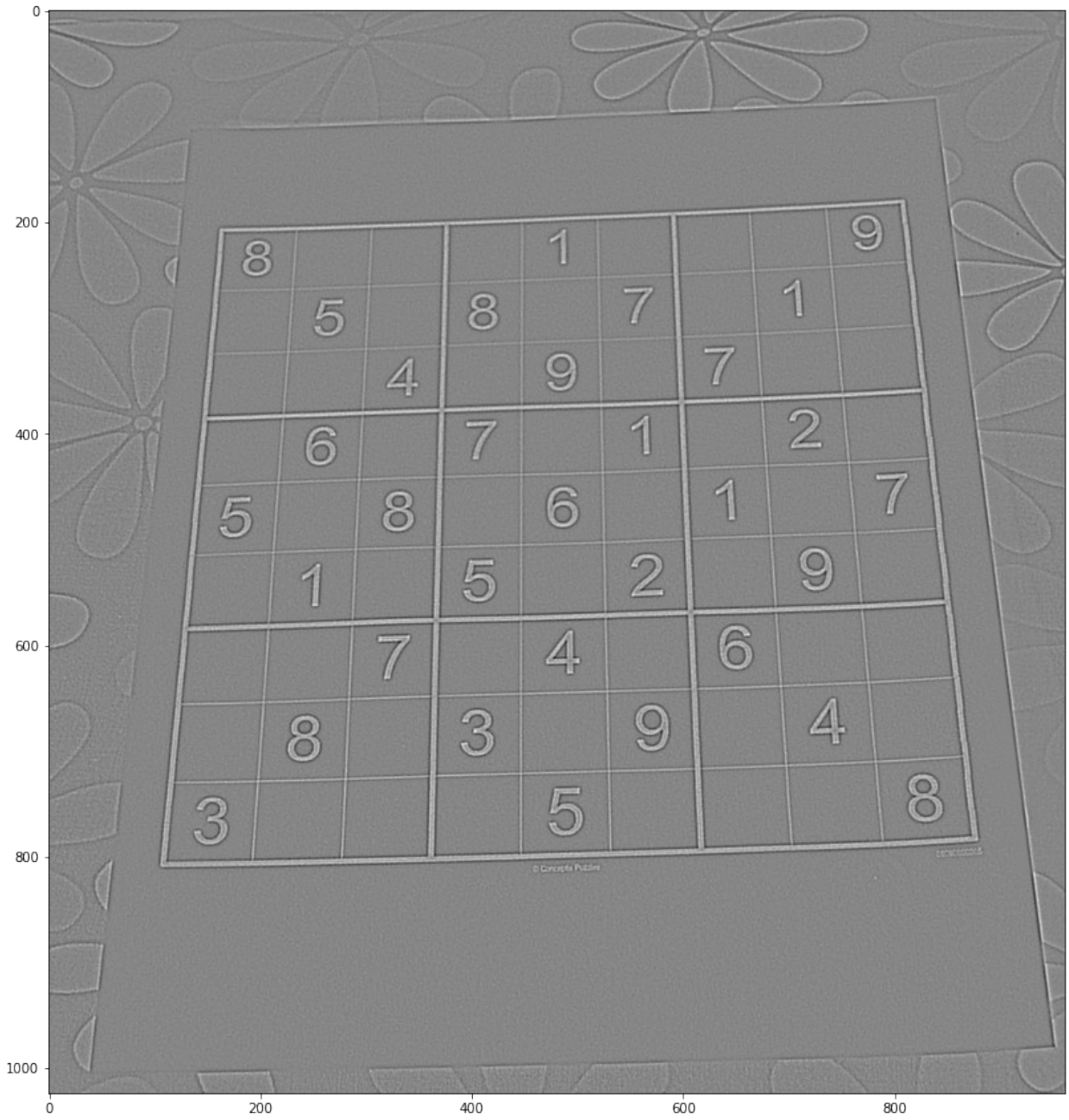
```
ret, blended_thresh = cv2.threshold(blended,100,255,cv2.THRESH_BINARY)
display_pic(blended_thresh)
```




```
#blended = ((blended - blended.min())/(blended.max()-
blended.min()))*255
```

This hasn't done anything great as the inner grid lines are barely visible even considsering a lower threshold. Lets apply a new concept of **Laplacian filters**. The Laplacian operator is a second-order derivative operator that is used to find the zero crossings of the image.The Laplacian of an image highlights regions of rapid intensity change and is based on the fact that in the edge area, the pixel intensity shows a "jump" or a high variation of intensity.

```
laplace = cv2.Laplacian(img,ddepth = cv2.CV_64F,ksize=5)
display_pic(laplace)
```

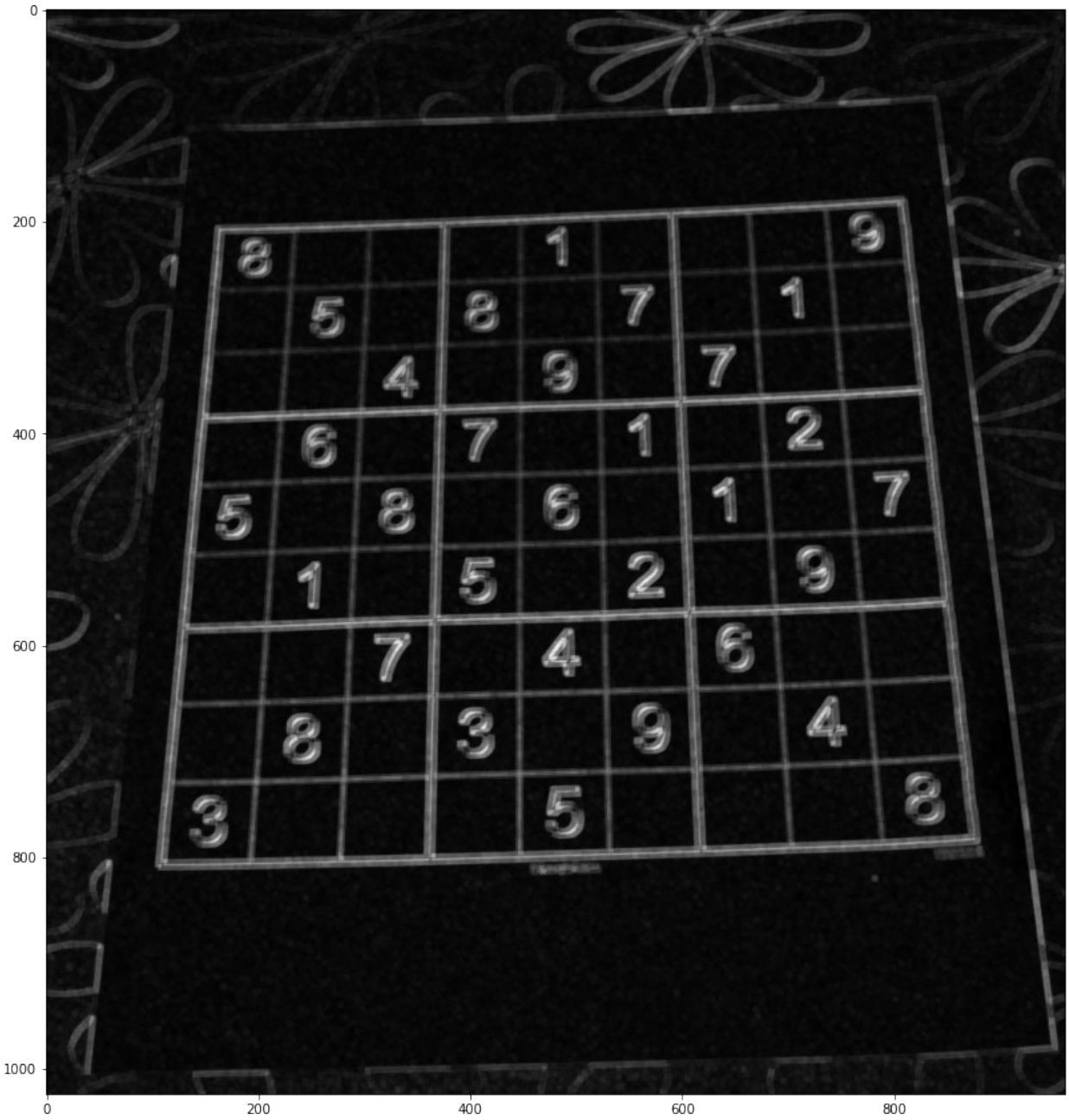


This looks so much better, the reason is because laplacian second derivative is 0 and eliminates any unwanted noise of slow varying intesnities. Laplacian formula is as follows:

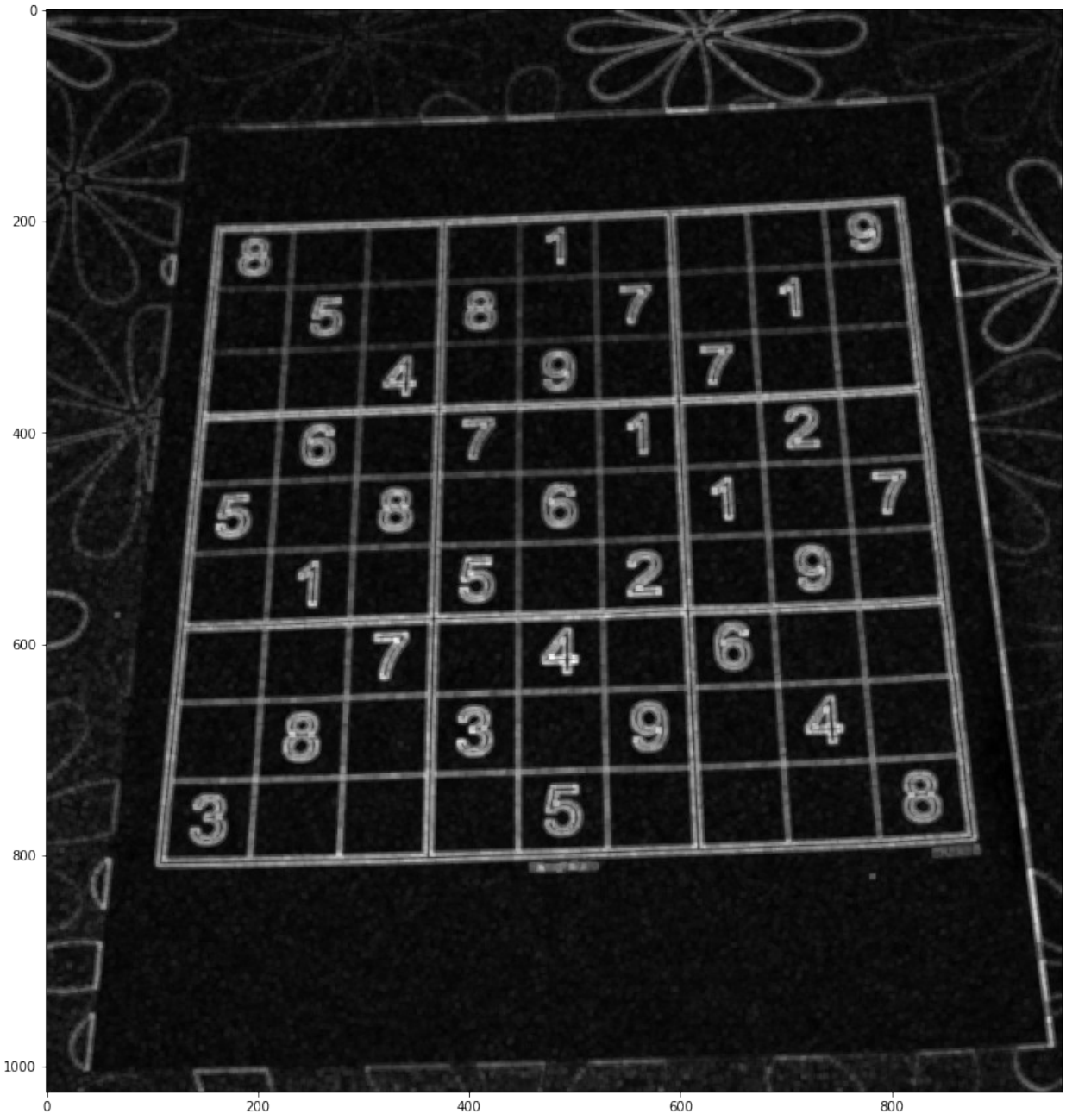
$$L(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian filtering is better suited for detecting edges and other features in an image, while normalized gradients are better suited for detecting edges at different scales.

```
kernel = np.ones(shape=(5,5),dtype=np.uint8)

gradient = cv2.morphologyEx(blended,cv2.MORPH_GRADIENT,kernel=kernel)
display_pic(gradient)
```




```
gradient = cv2.morphologyEx(laplace,cv2.MORPH_GRADIENT,kernel=kernel)
display_pic(gradient)
```



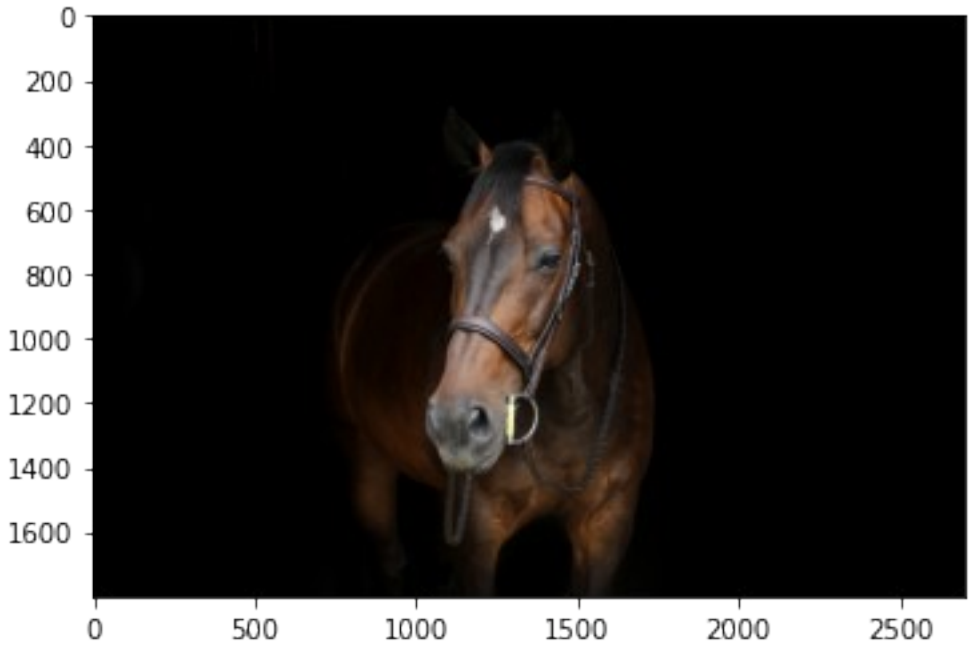
Histograms

Histograms are visual represntations for the frequency of values of a particular ranges into bins. In images, histograms are mostly used as representation of intensity values of the different color intensity values into 3 channels.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

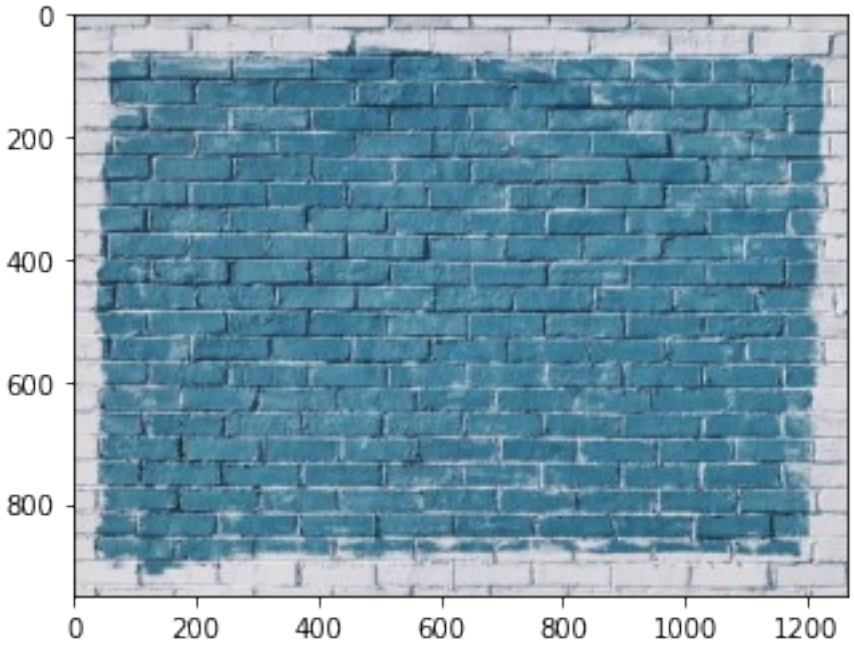
```
dark_horse = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/horse.jpg")
show_horse = cv2.cvtColor(dark_horse,cv2.COLOR_BGR2RGB)

rainbow = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/rainbow.jpg")
show_rainbow = cv2.cvtColor(rainbow,cv2.COLOR_BGR2RGB)

blue_bricks = cv2.imread("/Users/ruben/Desktop/Learning/Computer-Vision-with-Python/DATA/bricks.jpg")
show_bricks = cv2.cvtColor(blue_bricks,cv2.COLOR_BGR2RGB)

plt.imshow(show_horse)
```

```
<matplotlib.image.AxesImage at 0x7f7e41c246a0>
```



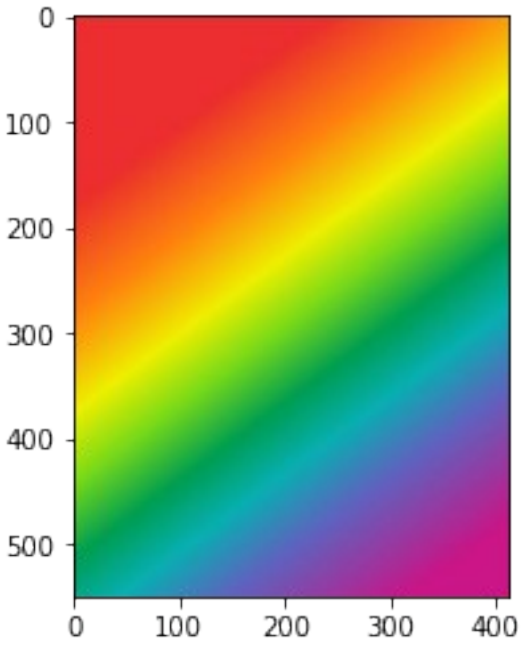

```
plt.imshow(show_bricks)
```

```
<matplotlib.image.AxesImage at 0x7f7e526834e0>
```



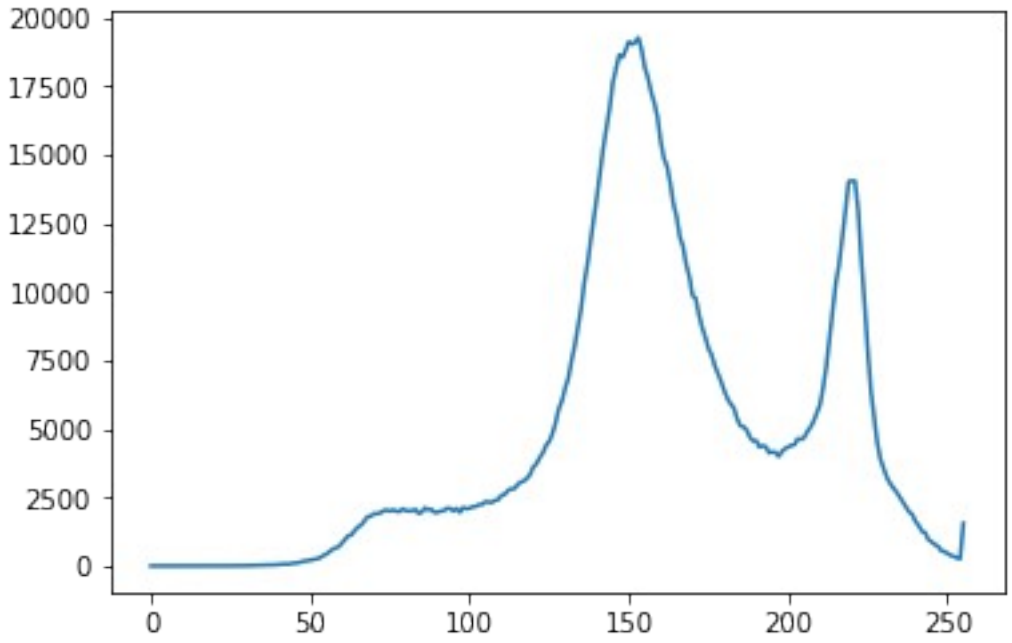

```
plt.imshow(show_rainbow)
```

```
<matplotlib.image.AxesImage at 0x7f7e524ed588>
```

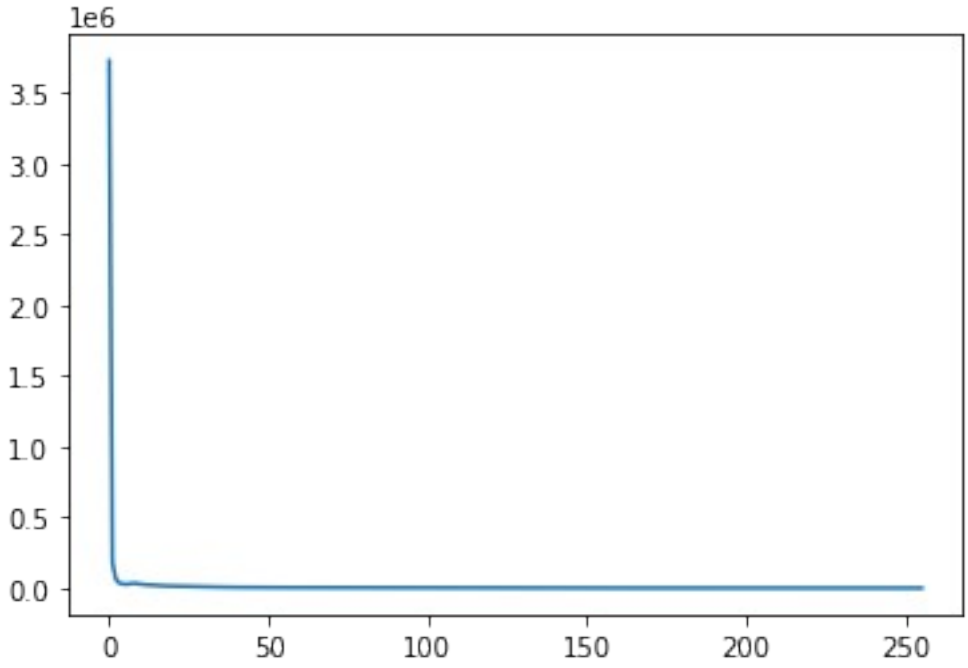



```
#256 since its mutually exlusive
hist = cv2.calcHist([blue_bricks],channels =
[0],mask=None,histSize=[256],ranges=[0,256])
plt.plot(hist)
```

```
[<matplotlib.lines.Line2D at 0x7f7e523b51d0>]
```

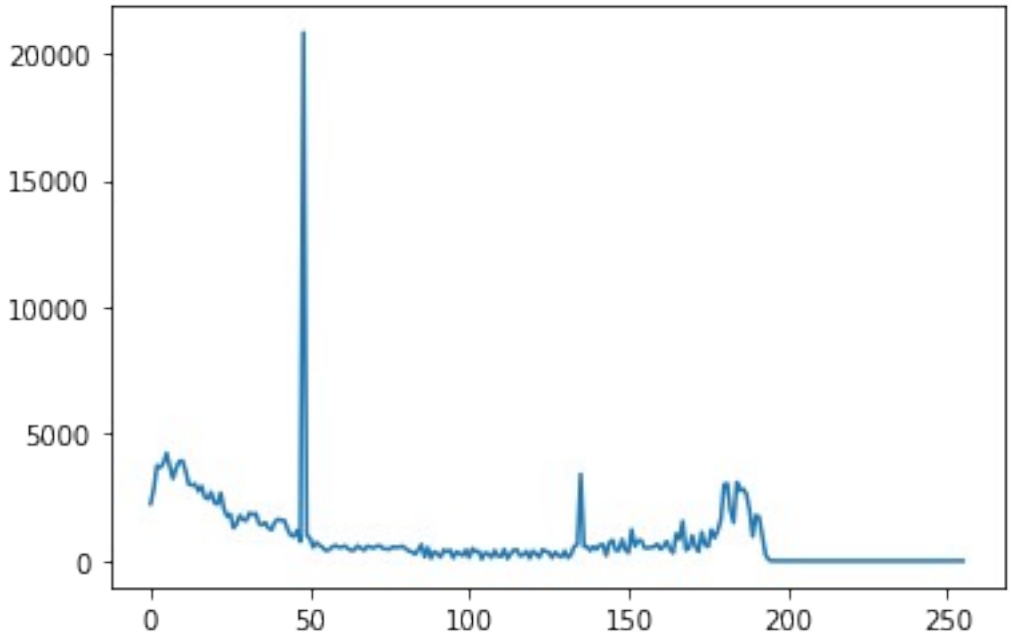



```
#256 since its mutually exlusive
hist = cv2.calcHist([dark_horse],channels =
[0],mask=None,histSize=[256],ranges=[0,256])
plt.plot(hist)

[<matplotlib.lines.Line2D at 0x7f7e52495a90>]
```

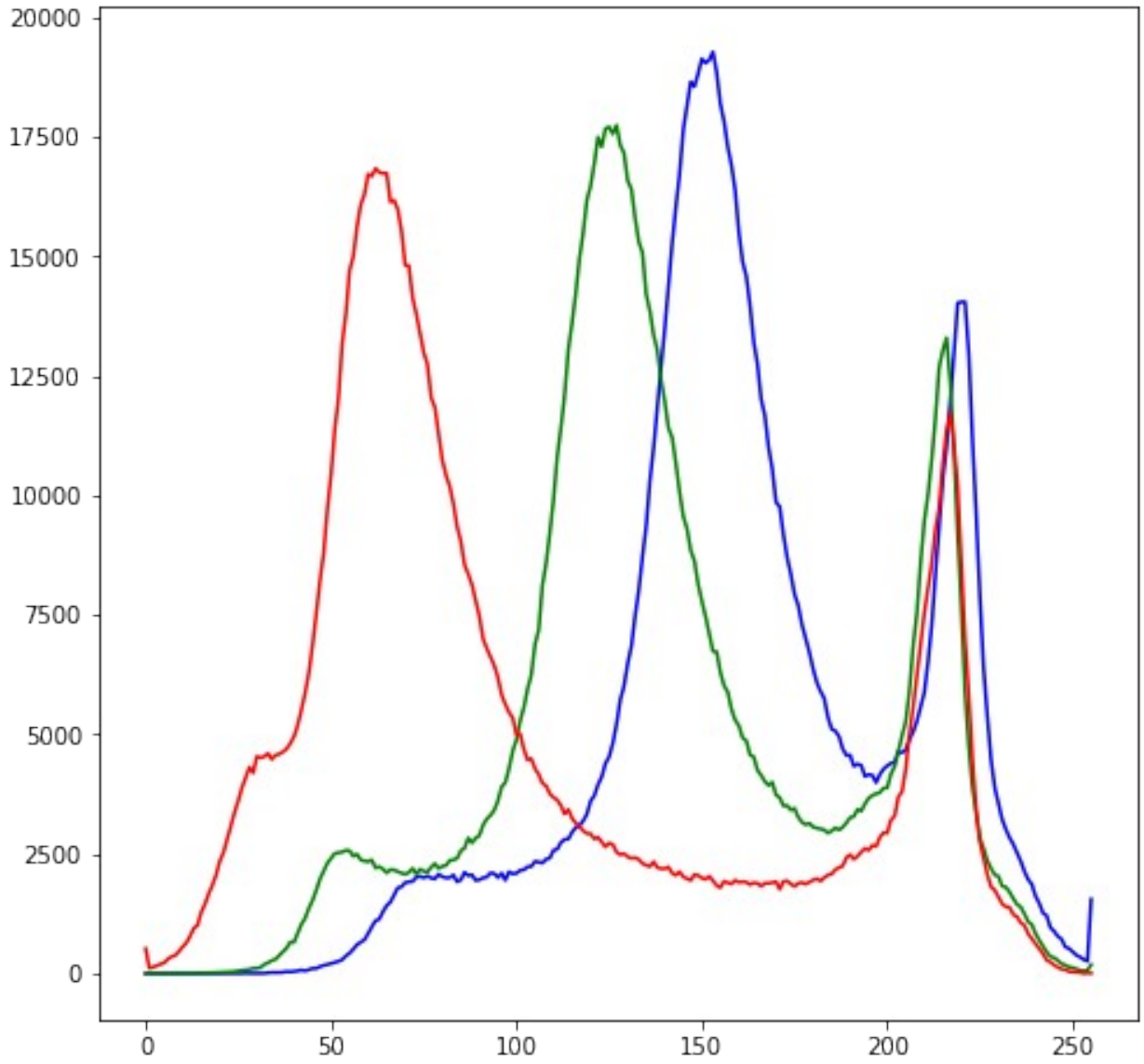


```
#256 since its mutually exlusive
hist = cv2.calcHist([rainbow],channels =
[0],mask=None,histSize=[256],ranges=[0,256])
plt.plot(hist)
```

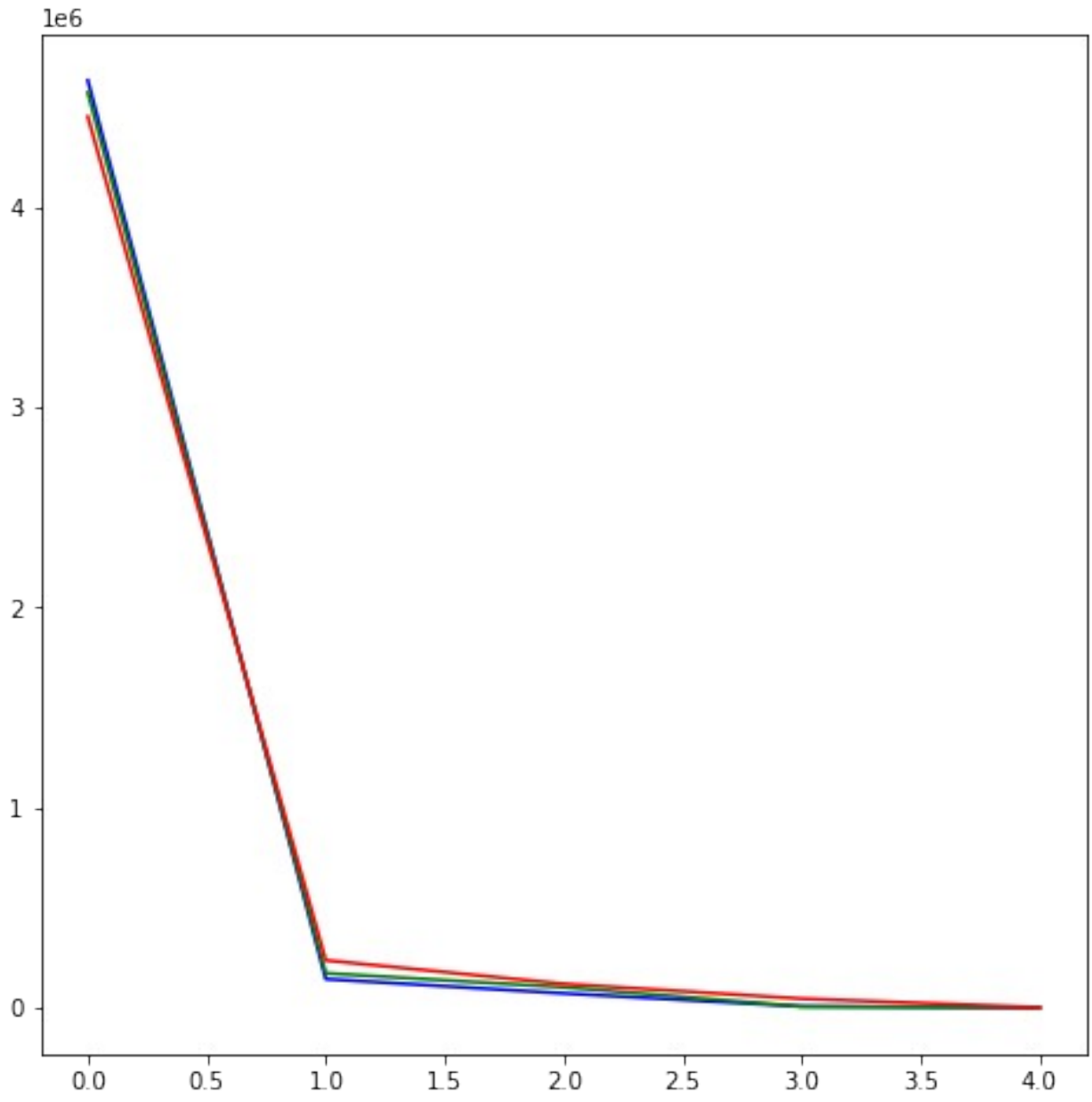
```
[<matplotlib.lines.Line2D at 0x7f7e41c09d68>]
```



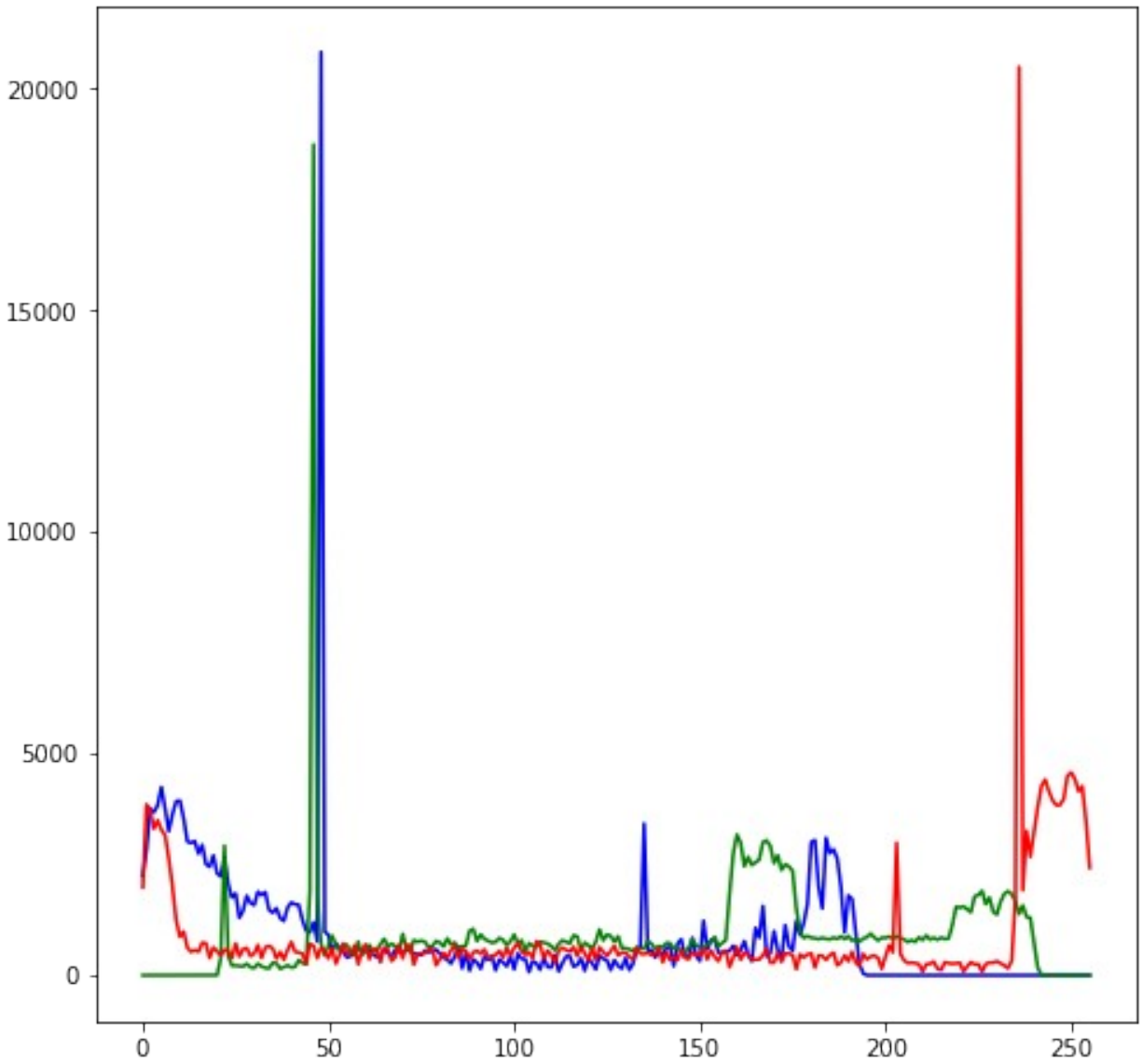

```
color = ('b','g','r')
plt.figure(figsize=(8,8))
for i,col in enumerate(color):
    histc = cv2.calcHist([blue_bricks],
[i],mask=None,histSize=[256],ranges=[0,256])
    plt.plot(histc,col)
```




```
color = ('b','g','r')
plt.figure(figsize=(8,8))
for i,col in enumerate(color):
    histc = cv2.calcHist([dark_horse],
[i],mask=None,histSize=[5],ranges=[0,256])
    plt.plot(histc,col)
```




```
color = ('b','g','r')
plt.figure(figsize=(8,8))
for i,col in enumerate(color):
    histc = cv2.calcHist([rainbow],
[i],mask=None,histSize=[256],ranges=[0,256])
    plt.plot(histc,col)
```



If the image size is very large, the pixel count will distort the histograms. So its important to resize the image to a lower dim,to understand the color histogram.