image processing

by using matrix multiplication, addition and subtraction

slash @ https://github.com/KvN1027/la_project_image_processing

outline

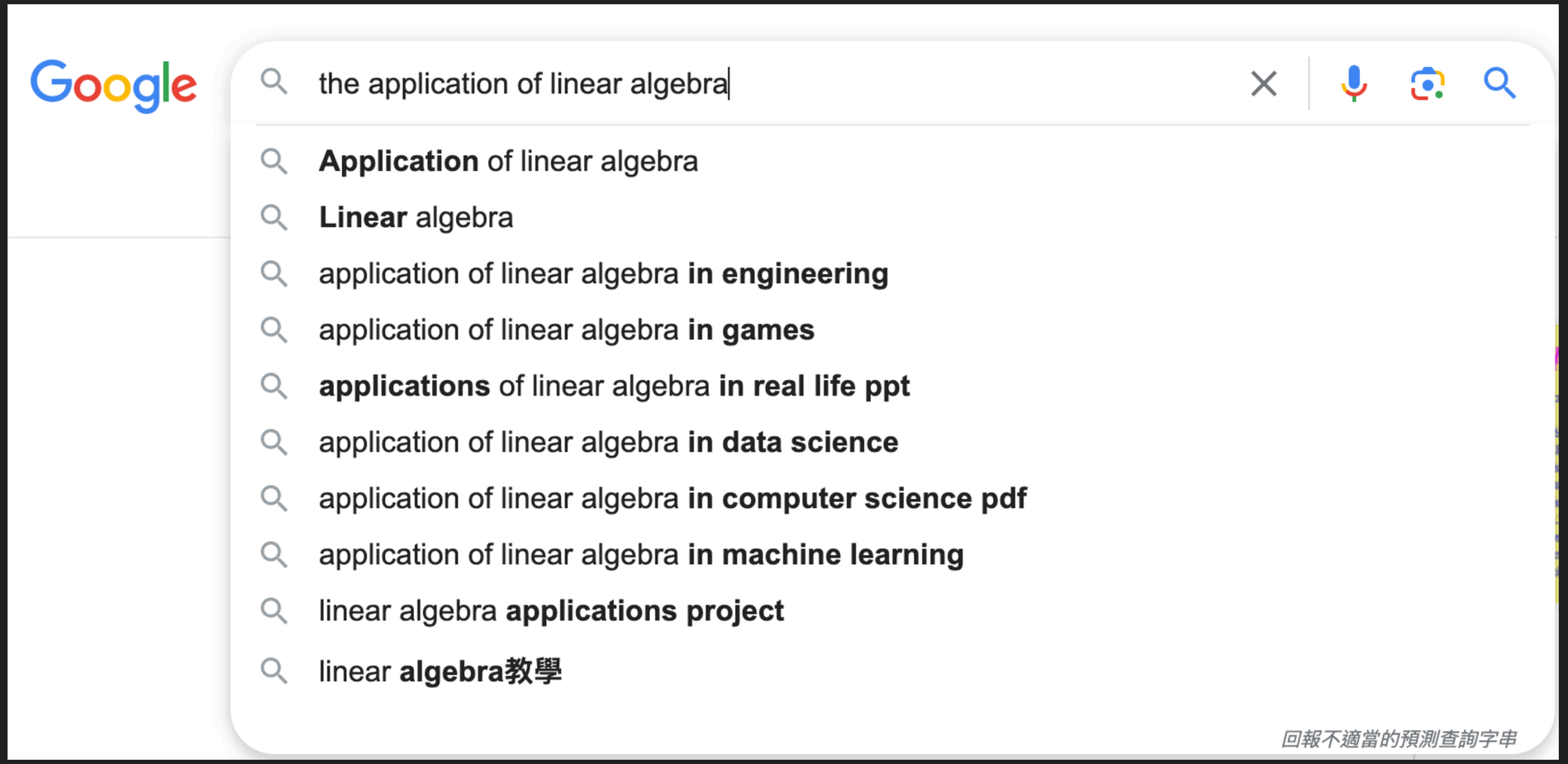
image processing

1. intro
2. based knowledge
3. blur
4. edge sketching
5. notetime
6. conclusion

slash @ https://github.com/KvN1027/la_project_image_processing

intro

slash @ https://github.com/KvN1027/la_project_image_processing



slash @ https://github.com/KvN1027/la_project_image_processing

Applications of linear algebra

- 1)The geometry of the surrounding space. ...
- 2)Analyse of functionality. ...
- 3)Research into more complicated systems. ...
- 4)Scientific computation. ...
- 5)Cryptography. ...
- 6)The theory of games.

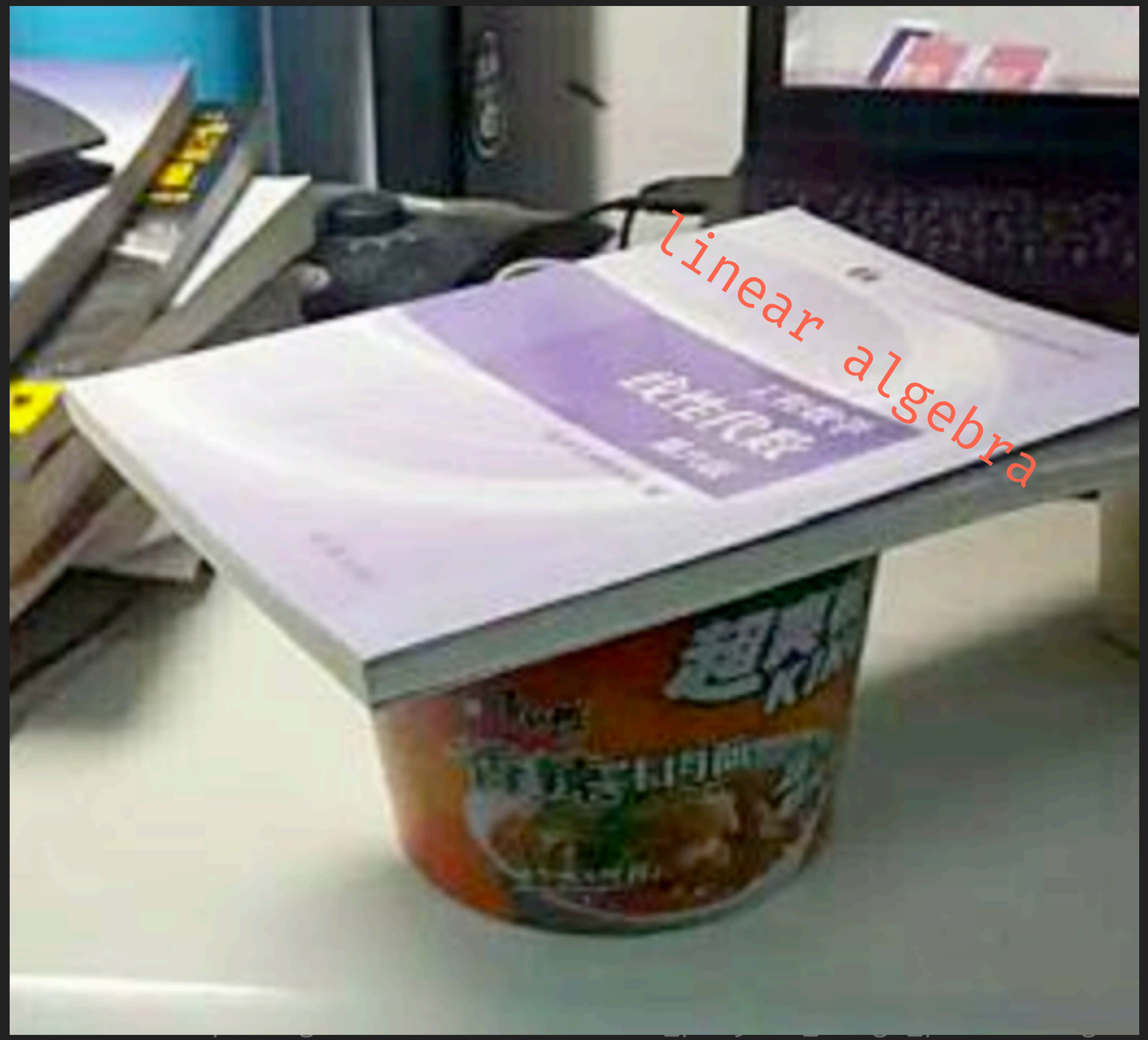
更多項目...

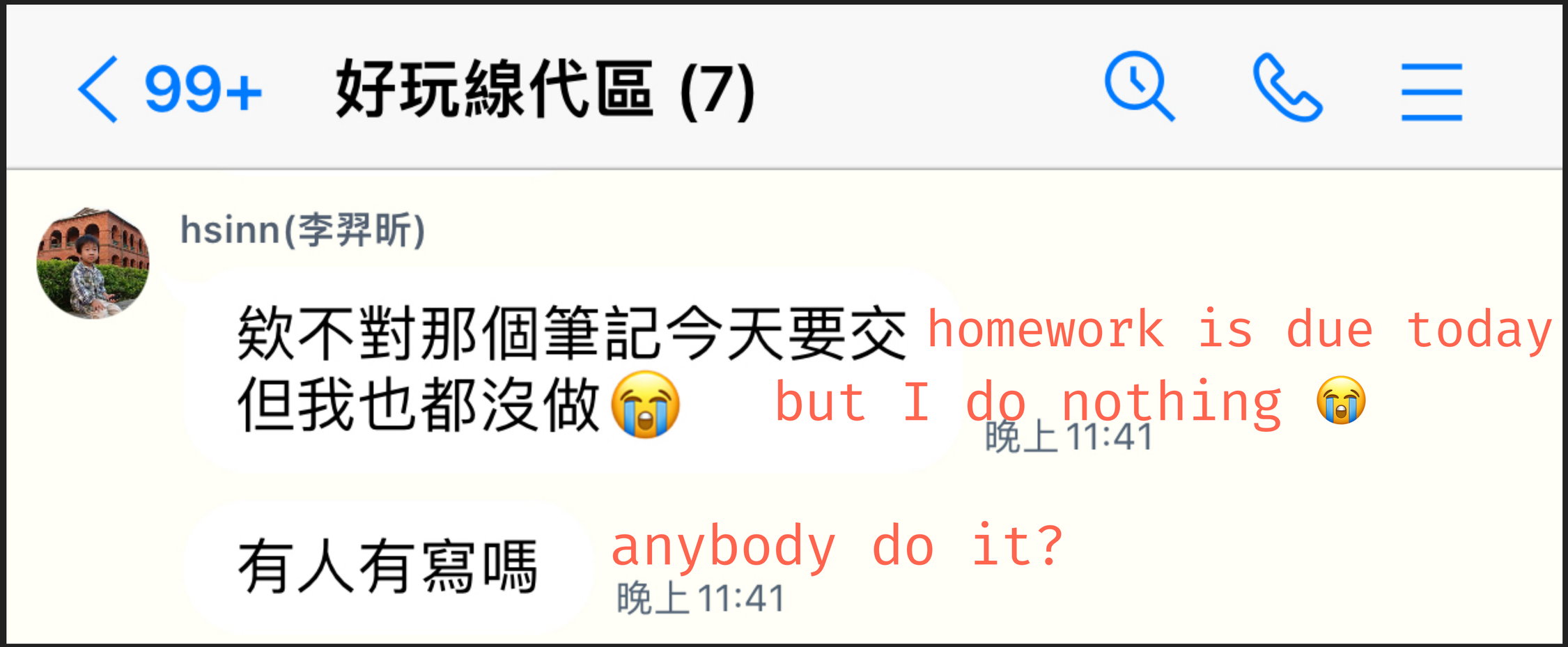
Unacademy

https://unacademy.com › ... › Mathematical Sciences

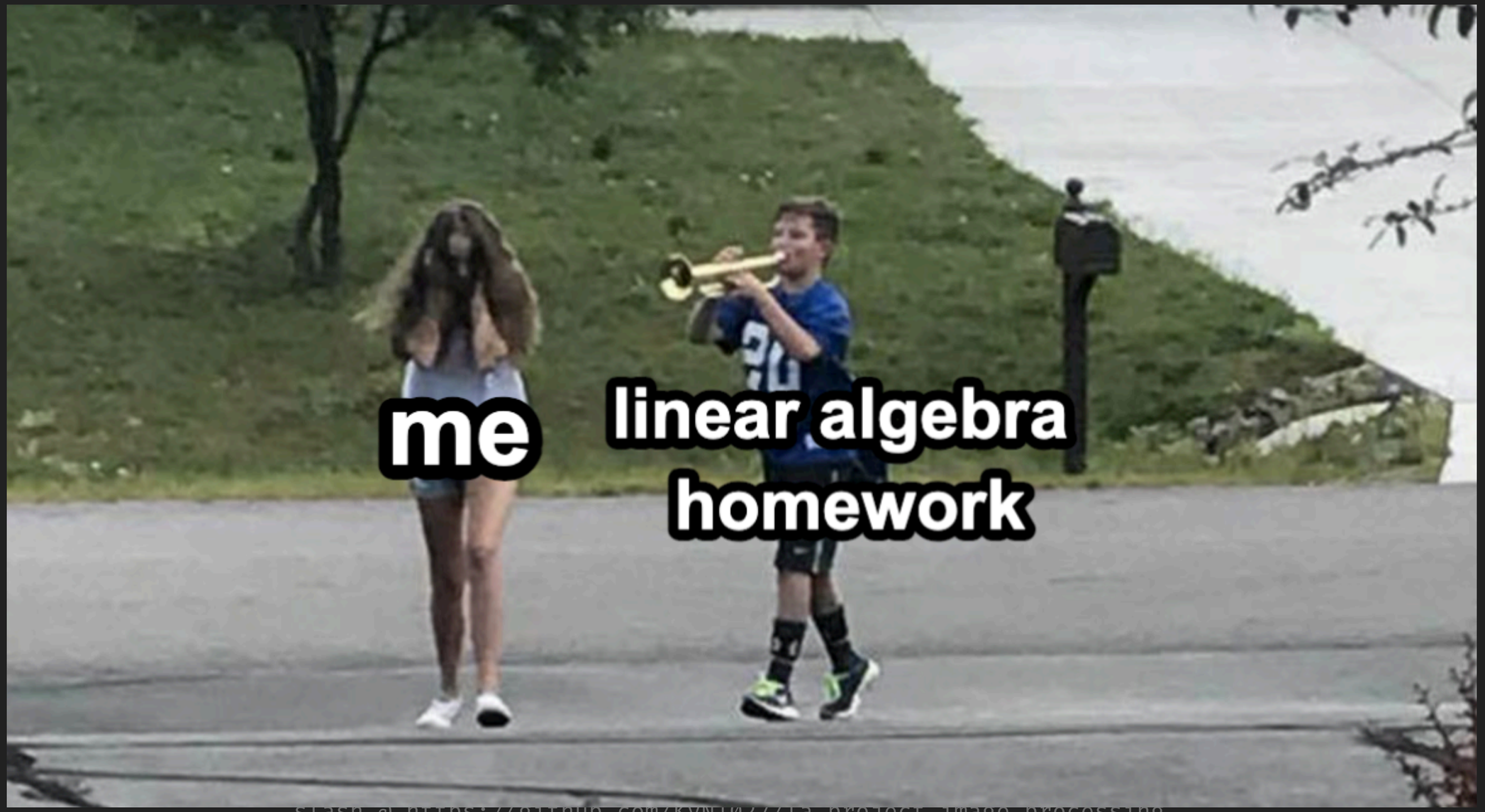
Small Note on Linear Algebra Applications - Unacademy

slash @ https://github.com/KvN1027/la_project_image_processing


linear algebra



slash @ https://github.com/KvN1027/la_project_image_processing



stash @ https://github.com/KVN1027/la_project_image_processing

Physics 12 - September 2018 | An electron is fired into a parall

https://www.enotes.com/homework-help/an-electron-fired-into-parallel-plate-moving-1-5-x-336861

Charge, e is = 1.602 x 10^(-19) C

Mass m...

Unlock All Answers Now

Start your **48-hour free trial** to unlock this answer and thousands more, and enjoy eNotes **ad-free**.

Start Free Trial to Unlock

Already a member? Log in here.

2 EDUCATOR ANSWERS

How do you calculate the of neutrons?

1 EDUCATOR ANSWER

What is the order of colors rainbow?

7 EDUCATOR ANSWERS

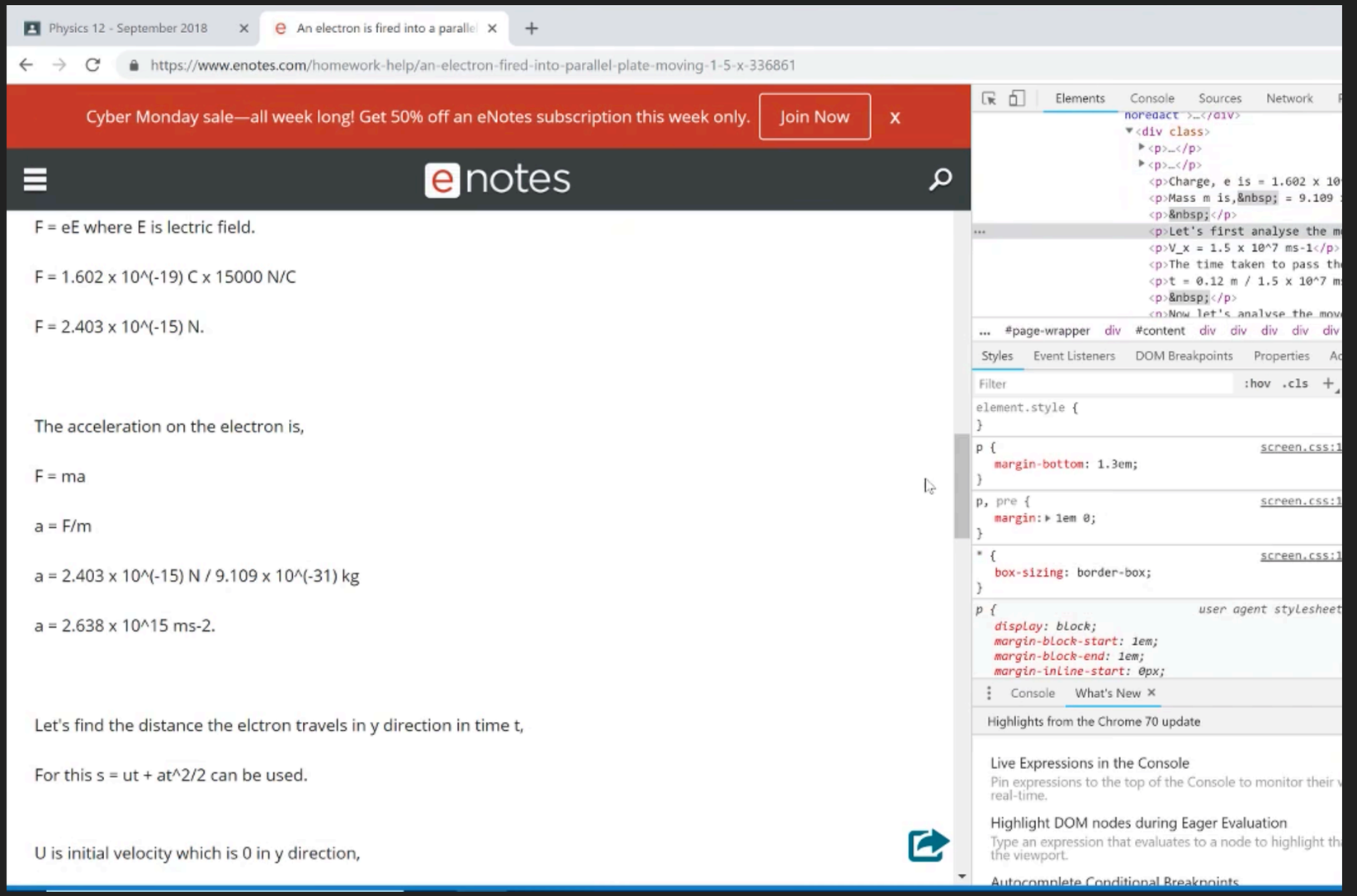
What is the difference bet smooth endoplasmic retic rough endoplasmic reticul

1 EDUCATOR ANSWER

What are ten examples of that you might find in you

2 EDUCATOR ANSWERS

MORE SCIENCE QU



slash @ https://github.com/KvN1027/la_project_image_processing

original image

slash @ https://github.com/KvN1027/la_project_image_processing

image process

original image

blurred image

slash @ https://github.com/KvN1027/la_project_image_processing

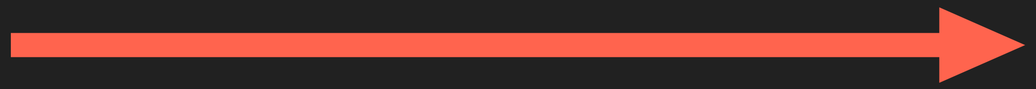


image process

slash @ https://github.com/KvN1027/la_project_image_processing

based knowledge

slash @ https://github.com/KvN1027/la_project_image_processing

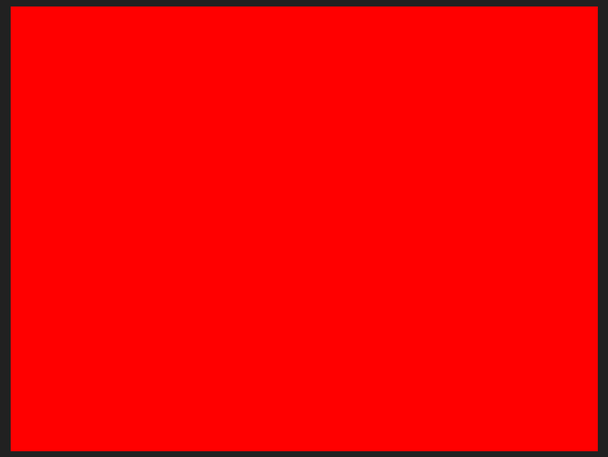
RGB

based knowledge

#ff0000

RGB

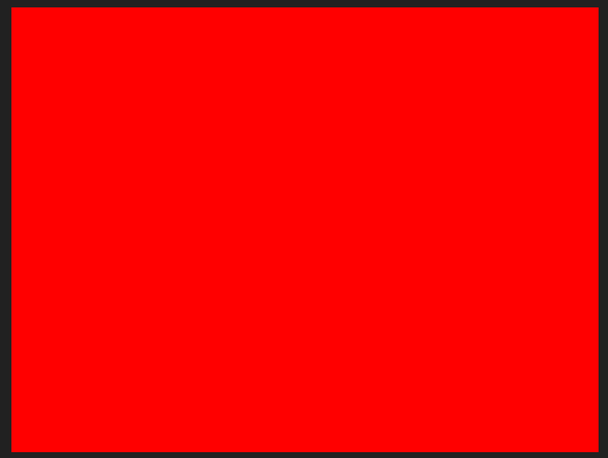
based knowledge



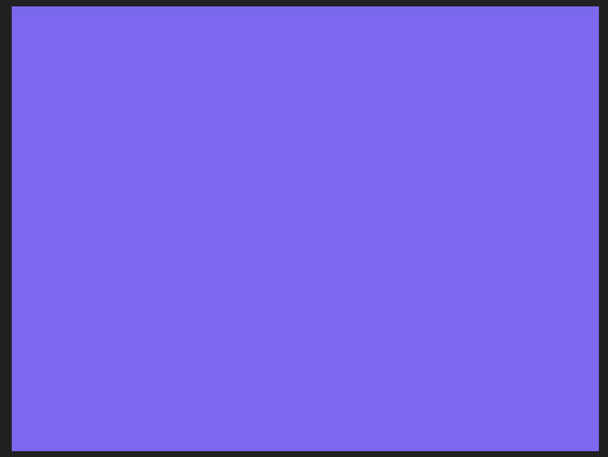
#ff0000

RGB

based knowledge



#ff0000

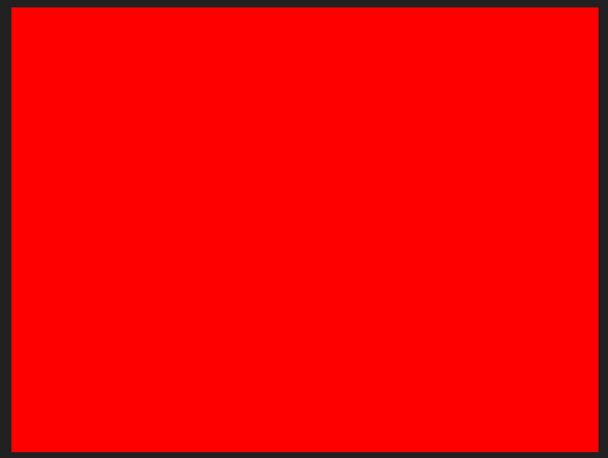


#7b68ee

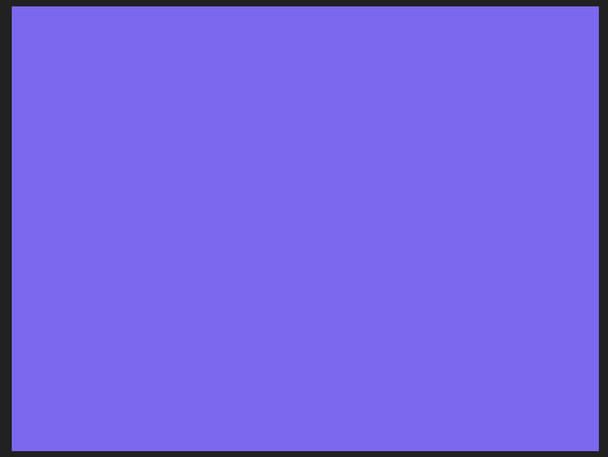
slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge



#ff0000



#7b68ee

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#7b68ee

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#7b68ee

hex

RGB

based knowledge

#7b68ee

2 hex digits range=16*16

hex

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#7b68ee

single color range = 0~255

hex

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#7b68ee

134 204 238

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#000000

0 0 0

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#000000

#ffffff

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

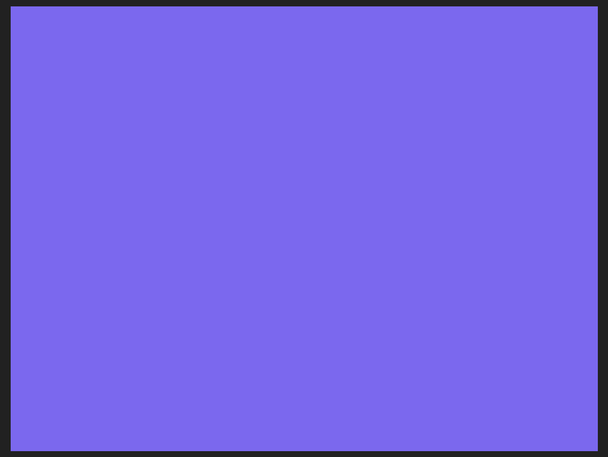
#7b68ee

134 204 238

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge



#7b68ee

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

1080

1920

slash @ https://github.com/KvM1027/1a_project_image_processing

RGB

based knowledge

slash @ https://github.com/KvM1027/1a_project_image_processing

RGB

based knowledge

RGB

#2C0E43	#2C0E43	#2C0E43
#2C0E43	#2C0E43	#2C0E43
#2C0E43	#2C0E43	#2C0E43

RGB

based knowledge

RGB

#2C C0E43

#2C C0E43

#2C C0E43

RGB

based knowledge

#2C0E43	#2C0E43	#2C0E43
#2C0E43	#2C0E43	#2C0E43
#2C0E43	#2C0E43	#2C0E43

dude that’s matrix

slash @ https://github.com/KvN1027/la_project_image_processing

RGB

based knowledge

#2C0E43 #2C0E43 #2C0E43

dude that’s matrix

that’s how image process works

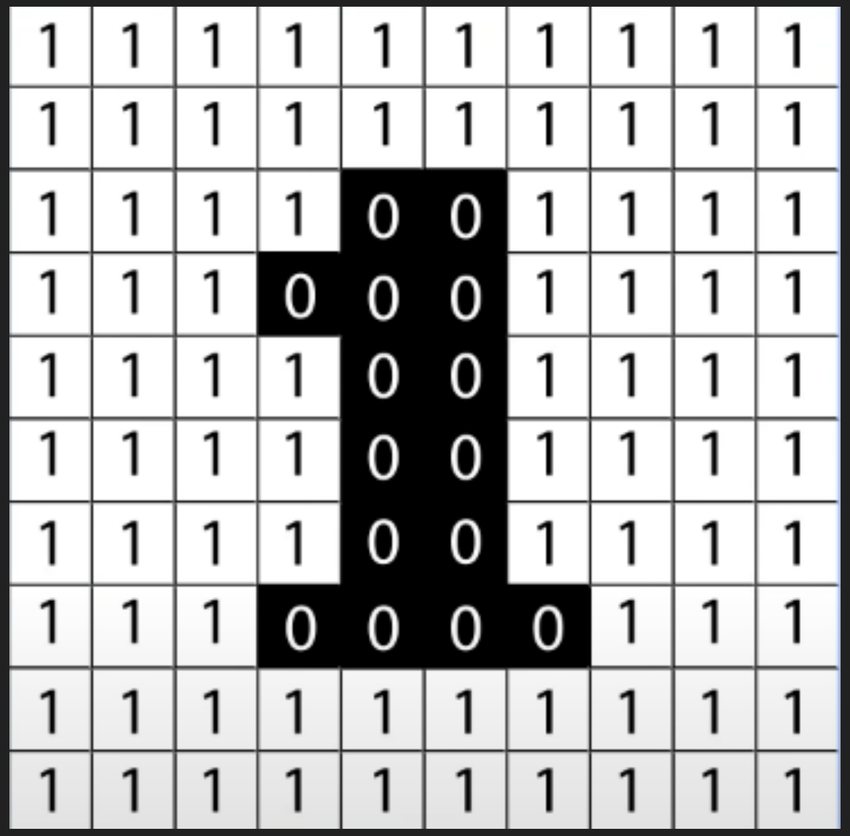
#2C0E43 #2C0E43 #2C0E43

slash @ https://github.com/KvN1027/la_project_image_processing

blur

how to blur

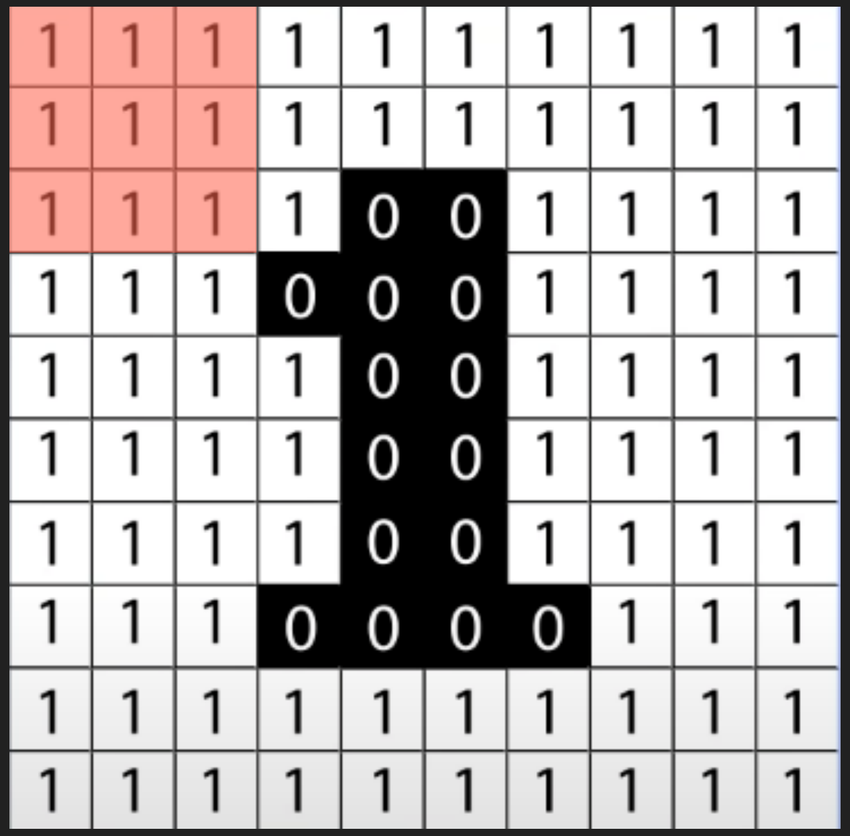
blur



slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

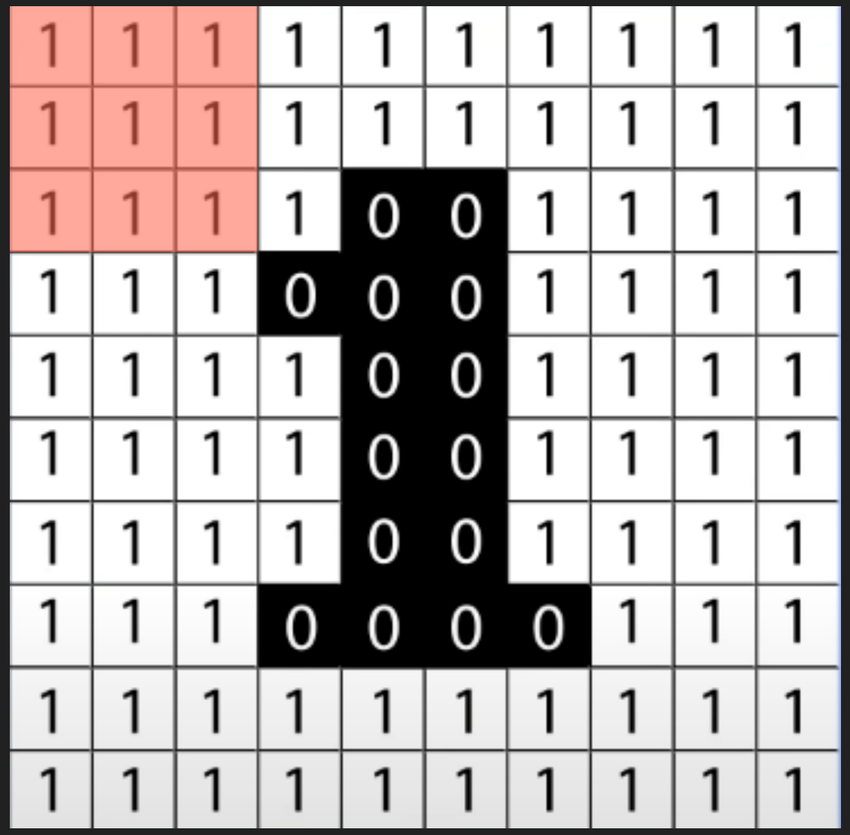
blur



slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

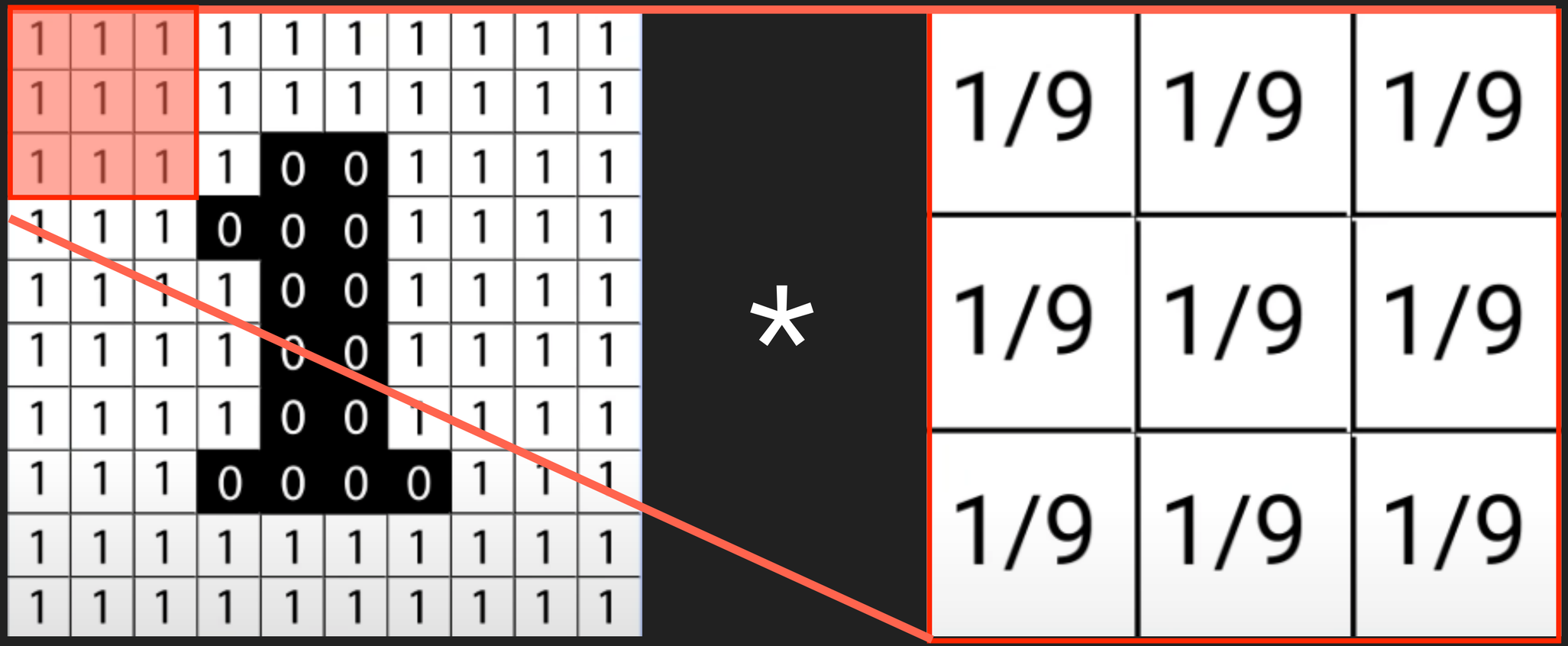


*

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

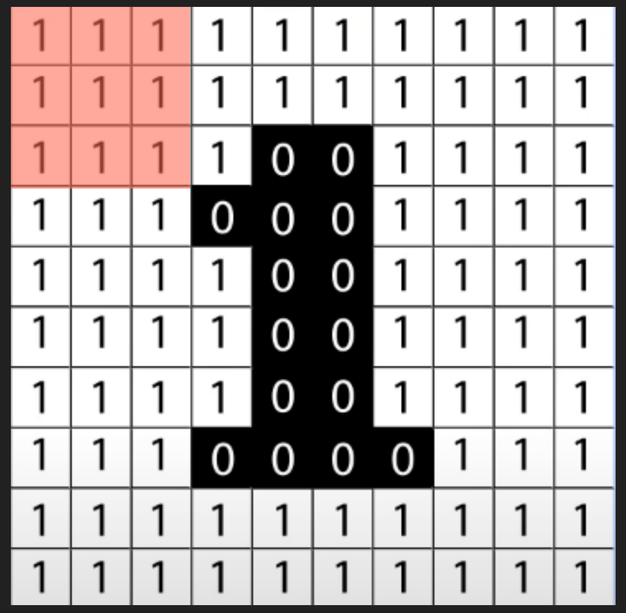
blur



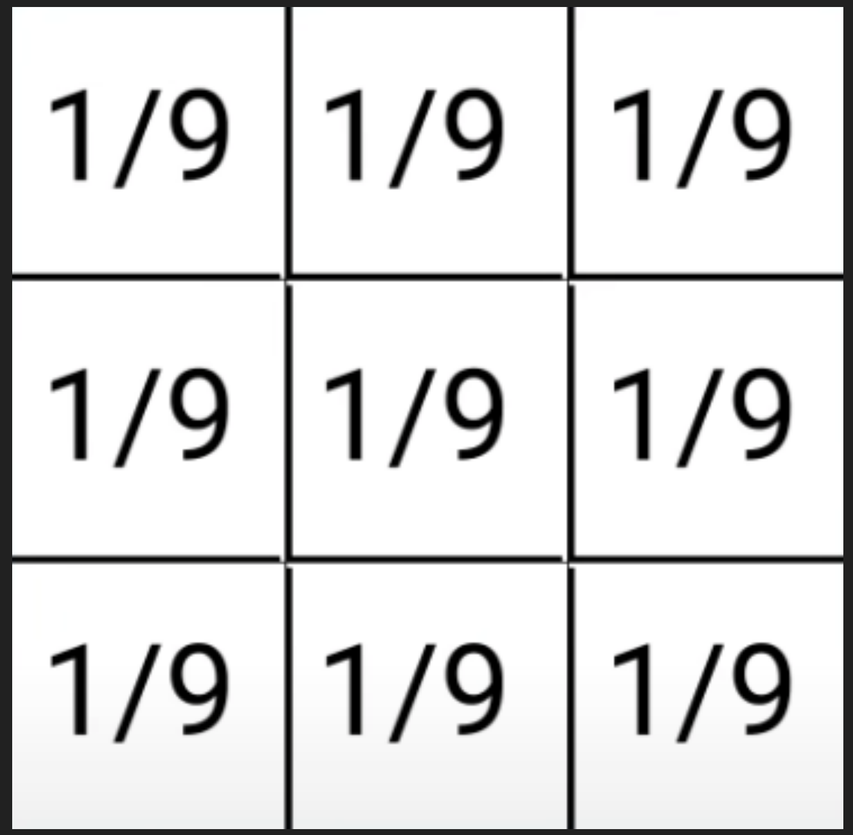
slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur



*

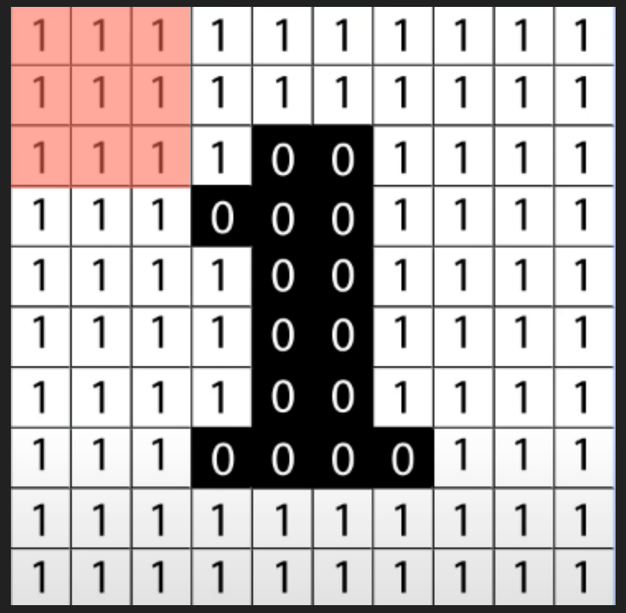


kernel

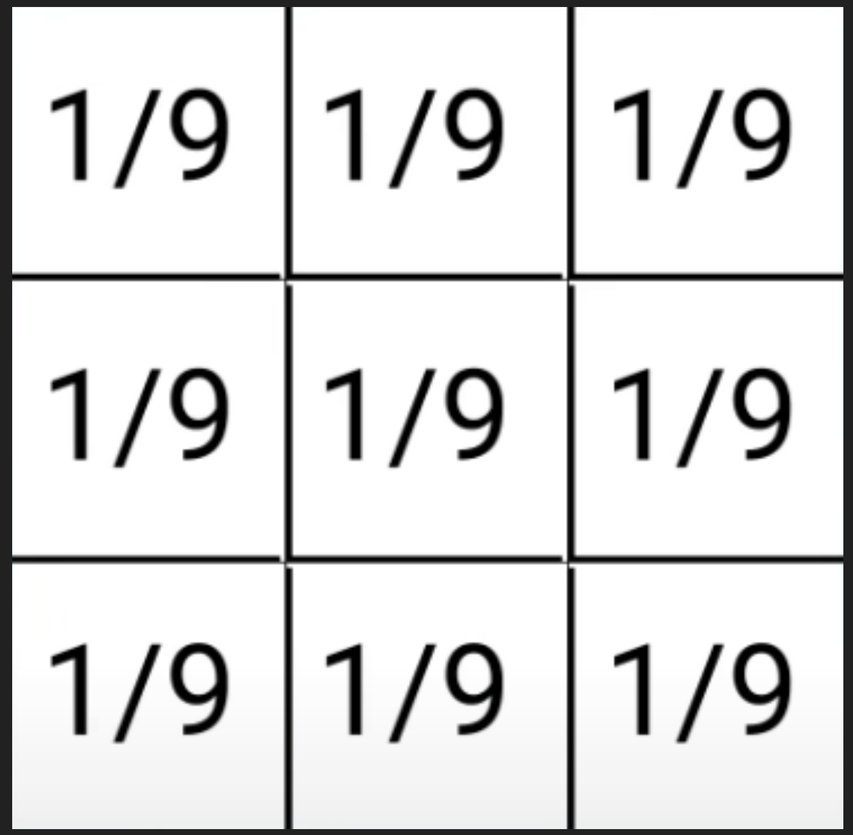
slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur



*



kernel

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

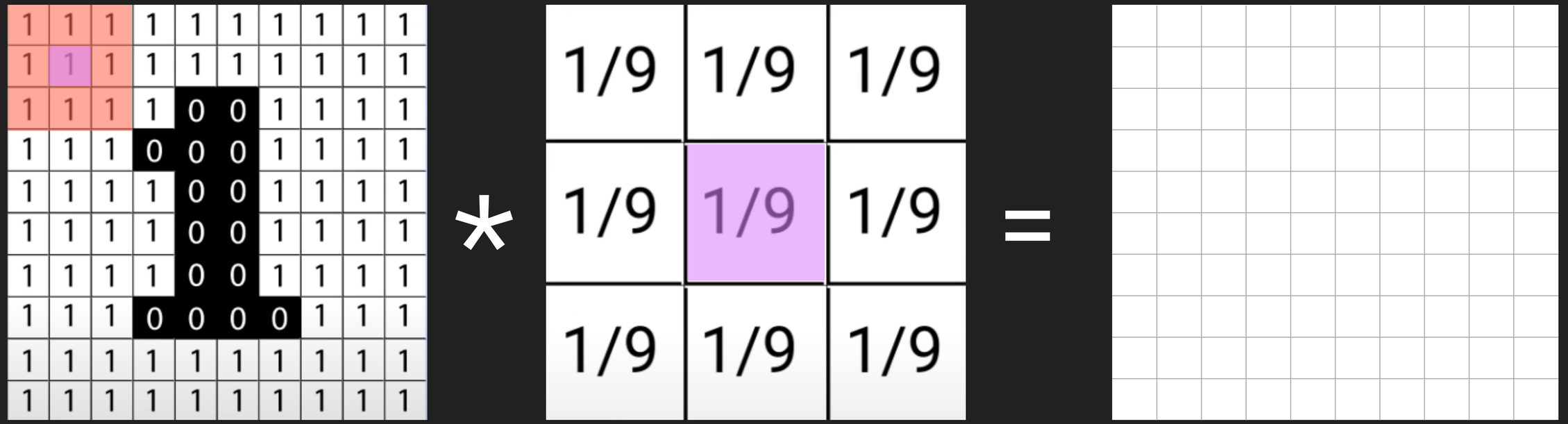
kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur



kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

find the average of the pixel values

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

avg: 1

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

avg: 1

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

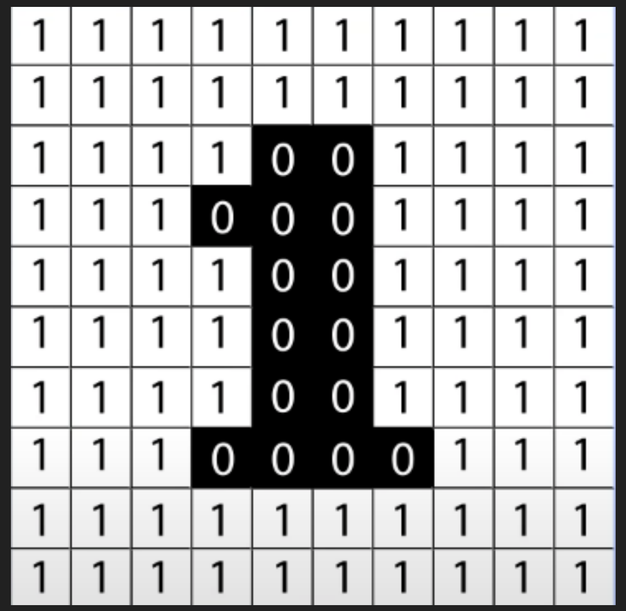
kernel

blur image

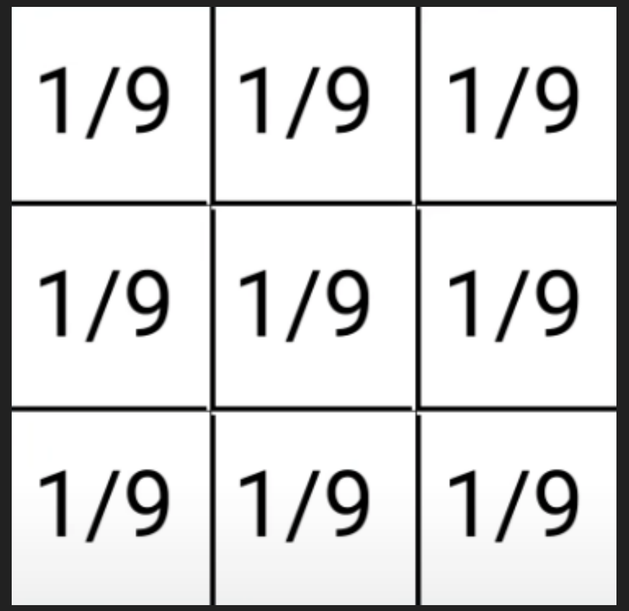
slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

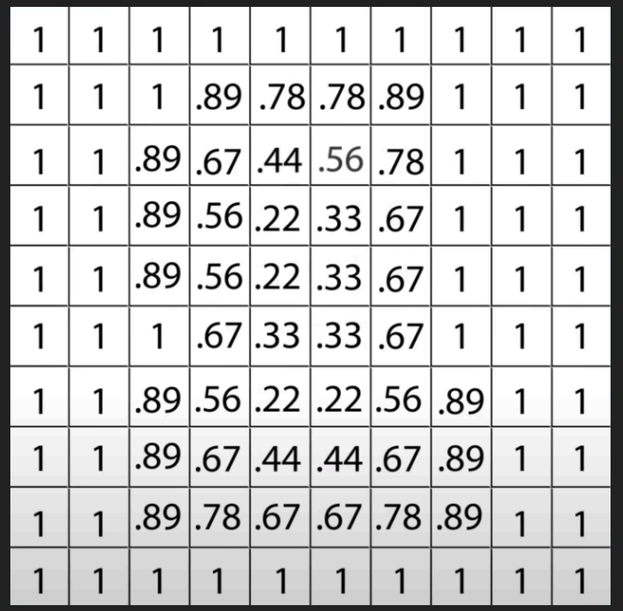
blur



*



=



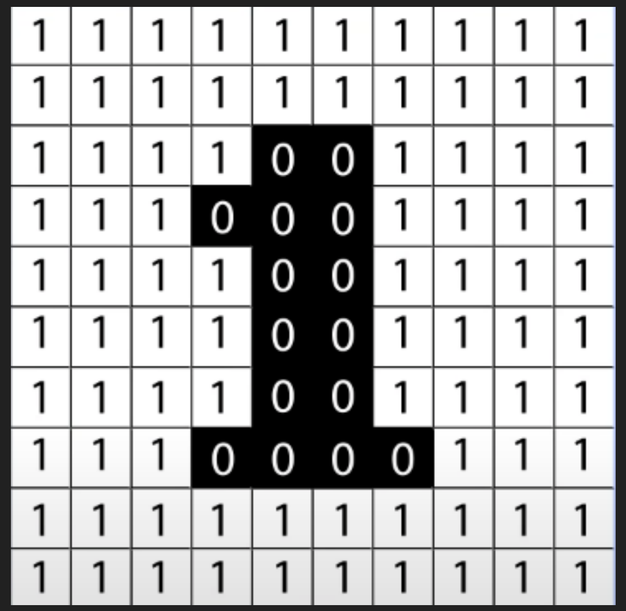
kernel

blur image

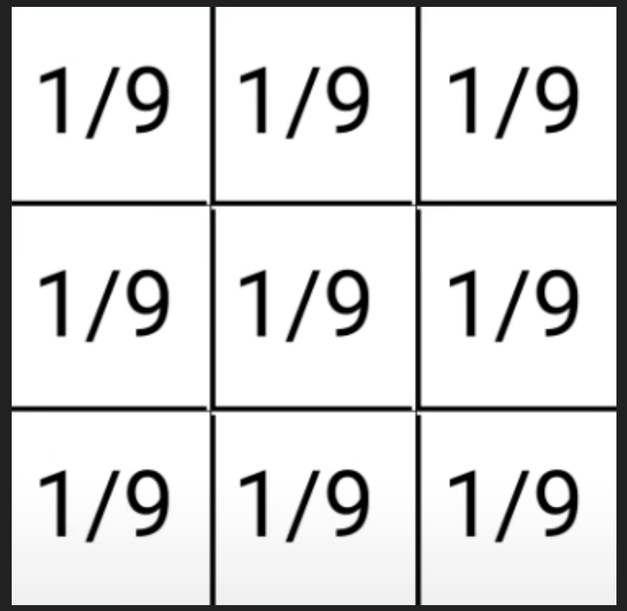
slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

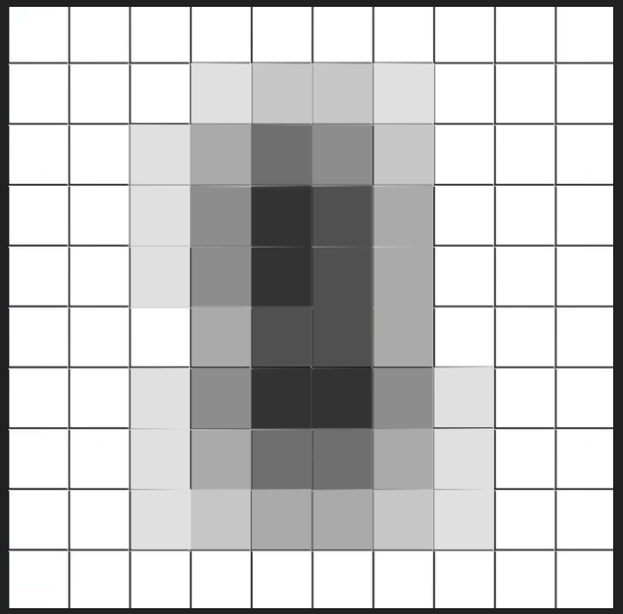
blur



*



=



kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

$$
* \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} =
$$

box blur

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

*

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

=

gaussian blur

slash @ https://github.com/KvN1027/la_project_image_processing

how to blur

blur

box blur

gaussian blur

slash @ https://github.com/KvN1027/la_project_image_processing

gaussian blur

blur

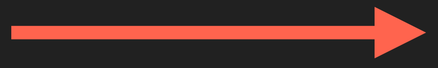
```python
# Set the size of the Gaussian kernel and standard deviation
kernel_size = 50  # A larger kernel size
sigma = 10  # A higher standard deviation for more blur

# Create Gaussian filter
ax = np.linspace(-(kernel_size - 1) / 2., (kernel_size - 1) / 2., kernel_size)
x, y = np.meshgrid(ax, ax)
gaussian_kernel = np.exp(-0.5 * (np.square(x) + np.square(y)) / np.square(sigma))
gaussian_kernel = gaussian_kernel / np.sum(gaussian_kernel)  # Normalization

# Read the original image
image_path = 'meowmeow.jpg'  # Replace with your image path
image = mpimg.imread(image_path)

# Save the original image without axis
fig, ax = plt.subplots()
ax.imshow(image)
ax.axis('off')  # Turn off the axis
plt.savefig('original_image.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory

# Apply Gaussian filter (convolution) to each channel
red = signal.convolve2d(image[:,:,0], gaussian_kernel, boundary='symm', mode='same')
green = signal.convolve2d(image[:,:,1], gaussian_kernel, boundary='symm', mode='same')
blue = signal.convolve2d(image[:,:,2], gaussian_kernel, boundary='symm', mode='same')

# Stack the channels back into a 3D array
blurred_image = np.stack([red, green, blue], axis=2)

# Ensure the pixel values are properly scaled
blurred_image = np.clip(blurred_image, 0, 255).astype(np.uint8)

# Save the blurred image without axis
fig, ax = plt.subplots()
ax.imshow(blurred_image)
ax.axis('off')  # Turn off the axis
plt.savefig('enhanced_blurred_image-less.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory
```

edge sketching

slash @ https://github.com/KvN1027/la_project_image_processing

flows

edge sketching



original image

grayscale

slash @ https://github.com/KvN1027/la_project_image_processing

flows

edge sketching

Grayscale Value=0.299×Red+0.587×Green+0.114×Blue

grayscale

slash @ https://github.com/KvN1027/la_project_image_processing

flows

edge sketching

*

-1	-1	-1
-1	8	-1
-1	-1	-1

=

edge detection

slash @ https://github.com/KvN1027/la_project_image_processing

flows

edge sketching

edge detection matrix →

```python
# Laplacian edge detection kernel
laplacian_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])

# Read the original image
image_path = 'meowmeow.jpg'  # Replace with your image path
image = mpimg.imread(image_path)
if image.ndim == 3 and image.shape[2] == 3:
    # Convert image to grayscale if it's color
    image = np.dot(image[...,:3], [0.2989, 0.5870, 0.1140])

# Apply Laplacian kernel to the grayscale image for edge detection
edges = convolve(image, laplacian_kernel)

# Ensure the pixel values are properly scaled
edges = np.clip(edges, 0, 255).astype(np.uint8)

# Save the edge-detected image without axis
fig, ax = plt.subplots()
ax.imshow(edges, cmap='gray')
ax.axis('off')  # Turn off the axis
plt.savefig('edge_detected_image.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory
```

slash @ https://git

flows

edge sketching

turn it to grayscale →

```python
# Laplacian edge detection kernel
laplacian_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])

# Read the original image
image_path = 'meowmeow.jpg'  # Replace with your image path
image = mpimg.imread(image_path)
if image.ndim == 3 and image.shape[2] == 3:
    # Convert image to grayscale if it's color
    image = np.dot(image[...,:3], [0.2989, 0.5870, 0.1140])

# Apply Laplacian kernel to the grayscale image for edge detection
edges = convolve(image, laplacian_kernel)

# Ensure the pixel values are properly scaled
edges = np.clip(edges, 0, 255).astype(np.uint8)

# Save the edge-detected image without axis
fig, ax = plt.subplots()
ax.imshow(edges, cmap='gray')
ax.axis('off')  # Turn off the axis
plt.savefig('edge_detected_image.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory
```

flows

edge sketching

apply the kernel →

```python
# Laplacian edge detection kernel
laplacian_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])

# Read the original image
image_path = 'meowmeow.jpg'  # Replace with your image path
image = mpimg.imread(image_path)
if image.ndim == 3 and image.shape[2] == 3:
    # Convert image to grayscale if it's color
    image = np.dot(image[...,:3], [0.2989, 0.5870, 0.1140])

# Apply Laplacian kernel to the grayscale image for edge detection
edges = convolve(image, laplacian_kernel)

# Ensure the pixel values are properly scaled
edges = np.clip(edges, 0, 255).astype(np.uint8)

# Save the edge-detected image without axis
fig, ax = plt.subplots()
ax.imshow(edges, cmap='gray')
ax.axis('off')  # Turn off the axis
plt.savefig('edge_detected_image.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory
```

slash @ https://git

flows

edge sketching

final result ✅

slash @ https://git

```python
# Laplacian edge detection kernel
laplacian_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])

# Read the original image
image_path = 'meowmeow.jpg'  # Replace with your image path
image = mpimg.imread(image_path)
if image.ndim == 3 and image.shape[2] == 3:
    # Convert image to grayscale if it's color
    image = np.dot(image[...,:3], [0.2989, 0.5870, 0.1140])

# Apply Laplacian kernel to the grayscale image for edge detection
edges = convolve(image, laplacian_kernel)

# Ensure the pixel values are properly scaled
edges = np.clip(edges, 0, 255).astype(np.uint8)

# Save the edge-detected image without axis
fig, ax = plt.subplots()
ax.imshow(edges, cmap='gray')
ax.axis('off')  # Turn off the axis
plt.savefig('edge_detected_image.png', bbox_inches='tight', pad_inches=0)
plt.close(fig)  # Close the figure to free up memory
```

note time!

(100% handmade)

slash @ https://github.com/KvN1027/la_project_image_processing

kernel

blur image

slash @ https://github.com/KvN1027/la_project_image_processing

the original picture is an $n \times n$ matrix P_0 and the one time blured pic is P_1

n times blured pic is P_n

$\Rightarrow P' = APB$

A is row combination (vertical blurring)

B is column combination (horizontal blurring)

$$A = \frac{1}{3}\begin{bmatrix} 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ & 1 & 1 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 1 & 1 \\ & & & & 1 & 1 \end{bmatrix} = \frac{1}{3}A_0$$

$$B = \frac{1}{3}\begin{bmatrix} 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ & 1 & 1 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 1 & 1 \\ & & & & 1 & 1 \end{bmatrix} = \frac{1}{3}A_0$$

which $A = B$

So that

$$P_1 = \left(\frac{1}{3}A_0\right)P_0\left(\frac{1}{3}A_0\right) = \frac{1}{9}A_0P_0A_0$$

$$P_2 = \left(\frac{1}{3}A_0\right)P_1\left(\frac{1}{3}A_0\right) = \left(\frac{1}{9}\right)^2A_0^2P_0A_0^2$$

$$\vdots$$

So, $$P_n = \left(\frac{1}{3}A_0\right)P_{n-1}\left(\frac{1}{3}A_0\right) = \left(\frac{1}{9}\right)^nA_0^nP_0A_0^n$$

and how to find A_0^n ?

for $n \times n$

$$A_0 = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}_{n \times n}$$

$$= I_{n \times n} + \begin{bmatrix} 0 & 1 & & \\ & 0 & 1 & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}_{n \times n}$$

suppose that $R = \begin{bmatrix} 0 & 1 & & 0 \\ & 0 & 1 & \\ & & \ddots & 1 \\ 0 & & & 0 \end{bmatrix}_{n \times n}$

we get $A_0^n = (R+I)^n$

we can use binomial theorem,

so $A_0^n = R^n + C_1^n R^{n-1} I + \cdots$

$$+ C_{n-1}^n R^1 [I]^{n-1} + C_n^n I^n$$

and that we found that

$R, R^1, R^2, R^3 \cdots R^n$

has some speeial regularity

for example, an $R_{4\times4}$ regularity

$$R = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$R^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$R^3 = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

$$R^4 = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 0 & 5 & 0 & 3 \\ 3 & 0 & 5 & 0 \\ 0 & 3 & 0 & 2 \end{bmatrix}$$

$$R^5 = \begin{bmatrix} 0 & 5 & 0 & 3 \\ 5 & 0 & 8 & 0 \\ 0 & 8 & 0 & 5 \\ 3 & 0 & 5 & 0 \end{bmatrix}$$

$$R^6 = \begin{bmatrix} 4 & 0 & 8 & 0 \\ 0 & 13 & 0 & 8 \\ 8 & 0 & 13 & 0 \\ 0 & 8 & 0 & 5 \end{bmatrix}$$

1 1 2 2 5 5 ...

0 1 1 3 3 8 ...

how to blur the picture but with less distortion

→ put more weight on the center pixel

for example:

Guassian filter matrix

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

after Guassian blurred n times, (P_0 turns into P_n)

$$P_n = \left(\frac{1}{16}\right)^n A^n P_0 A^n \text{, where } A = \begin{bmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & 1 & 2 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 2 \end{bmatrix}$$

conclusion

(We got through it!)

slash @ https://github.com/KvN1027/la_project_image_processing

Matrix !

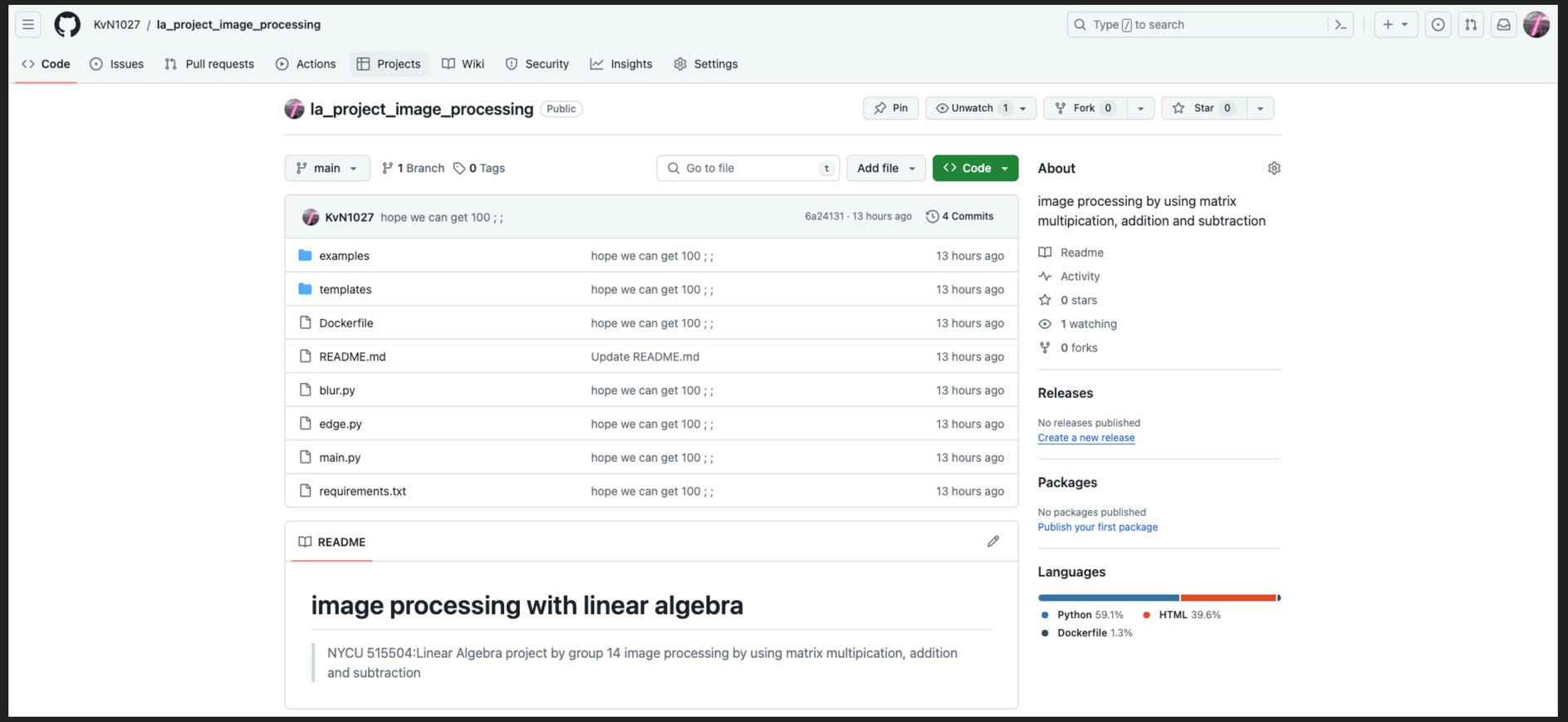
conclusion

- We did
  - Kernel Generation
  - Convolution Operation
  - Real-Time Preview (On Your Phone!)

slash @ https://github.com/KvN1027/la_project_image_processing

Matrix !

conclusion



slash @ https://github.com/KvN1027/la_project_image_processing