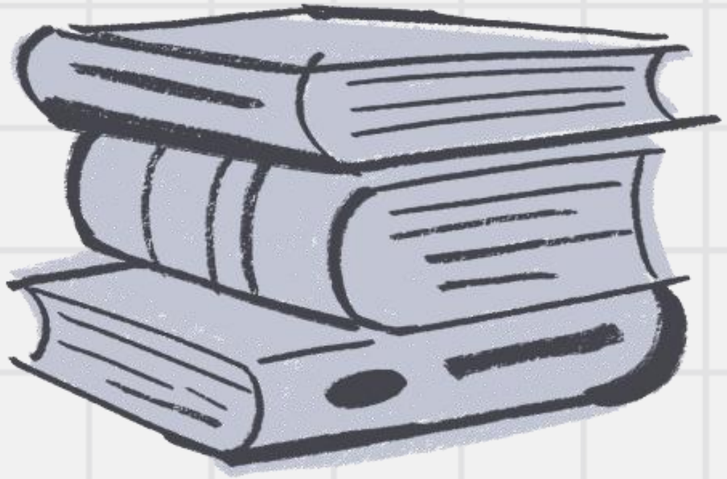
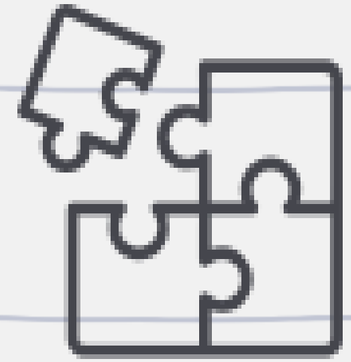
COMPUTER VISION

Table of Contents



01

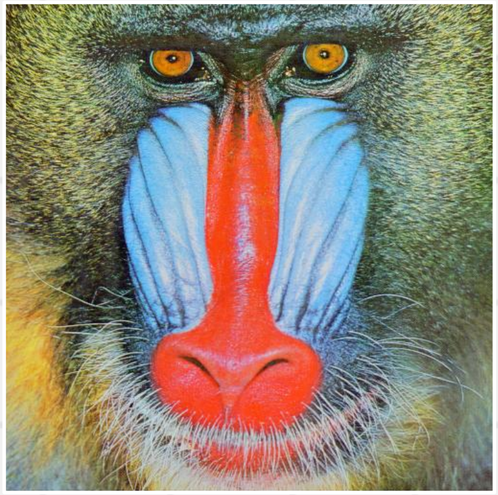


Introduction

Introduction

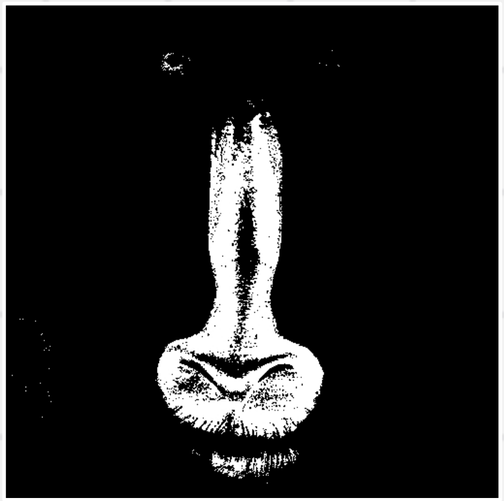
Last time, we finished our talk with a question. How could we reduce noise? 🤔💭
Because after we used HSV format to make color threshold, we found that the result still had noise. 📺

Original Image



RGB Threshold

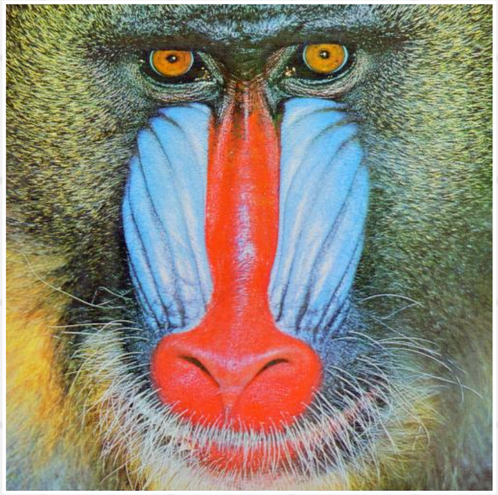
HSV Threshold



Introduction

The answer is that we need to apply an image processing technique called filtering. Its job is to remove noise from the image, and the specific filter used for this task is called a blurring filter. However, to understand how filters work, we first need to review an important mathematical concept called convolution. Later in this session, we will explore convolution theory in computer vision. 🔎

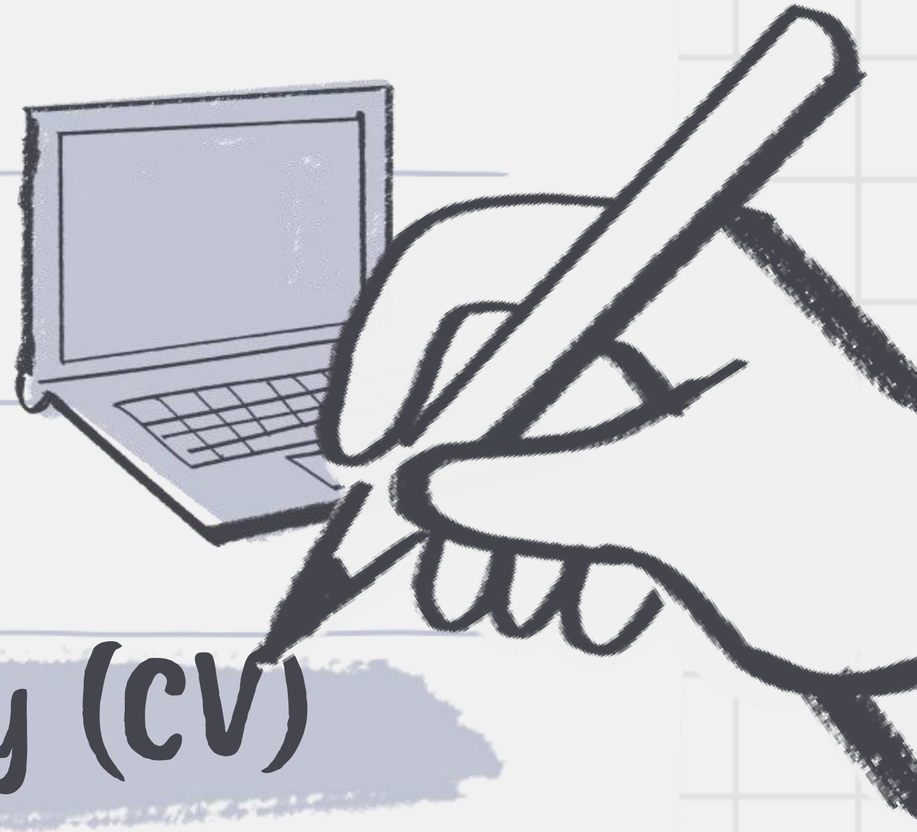
Original Image



RGB Threshold

HSV Threshold

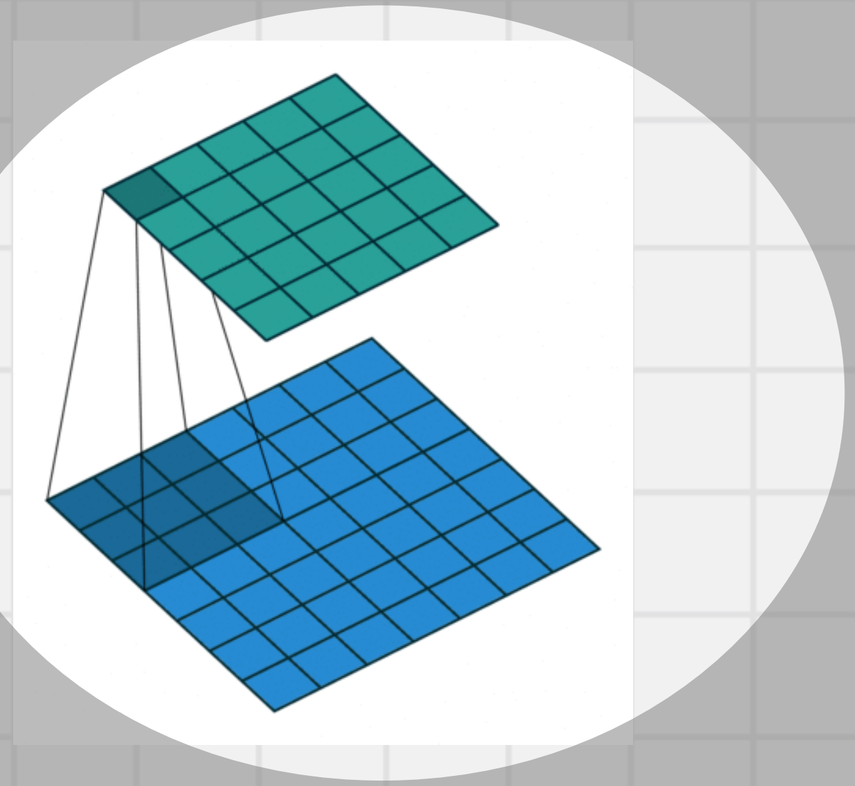
02



Convolution Theory (CV)

Convolution Theory (CV)

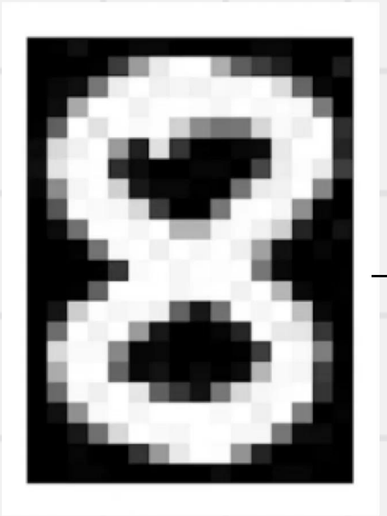
In Computer Vision, convolution is generally used to extract or create a feature map (with the help of kernels) out of the input image.

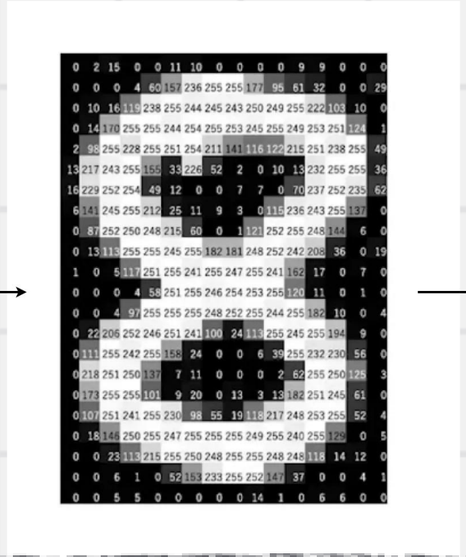


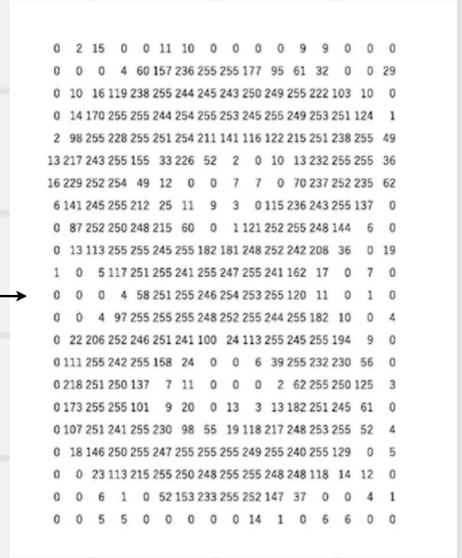
In this image, the blue matrix is the input, and the green matrix is the output. Whereas we have a kernel moving through the input matrix to get/extract the features. So, let's first understand the input matrix.

Convolution Theory (CV)

Input Matrix: An image is made up of pixels, where each pixel is in the inclusive range of [0, 255]. So we can represent an image, in terms of a matrix, where every position represents a pixel. The pixel value represents how bright it is, i.e. pixel -> 0 is black and pixel -> 255 is white (highest brightness). A grayscale image has a single matrix of pixels, i.e. it doesn't have any colour, whereas a coloured image (RGB) has 3 channels, and each channel represents its colour density.



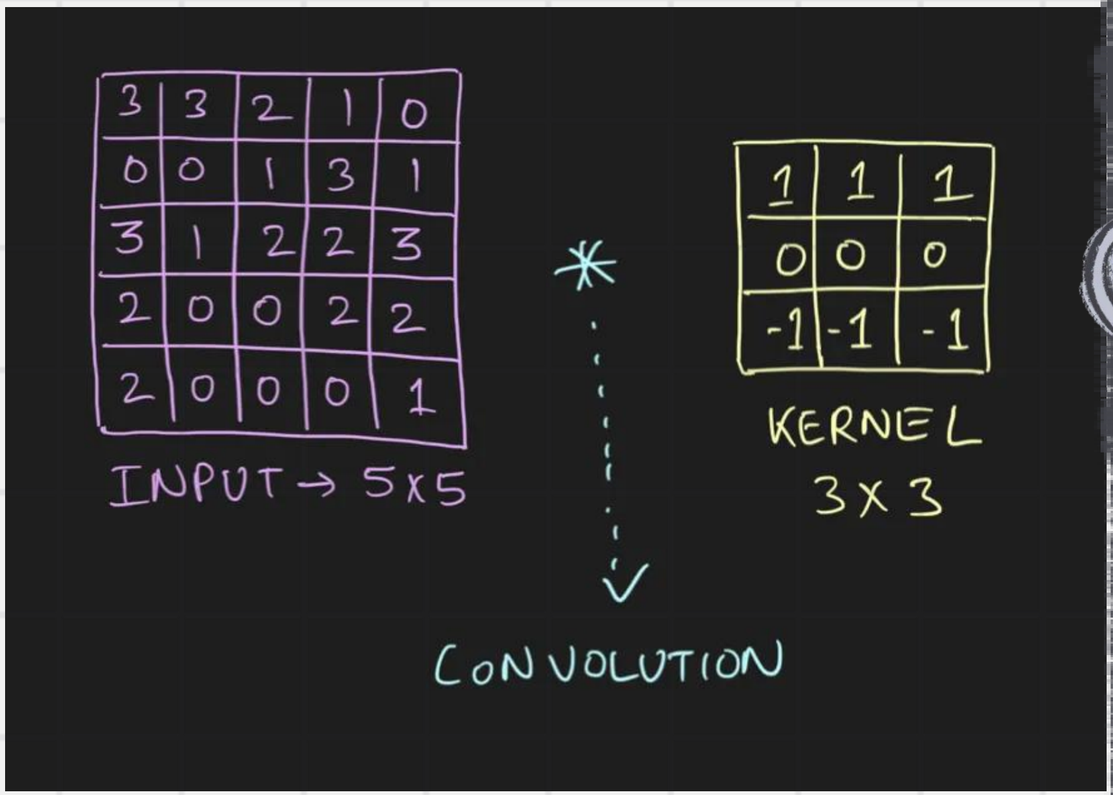




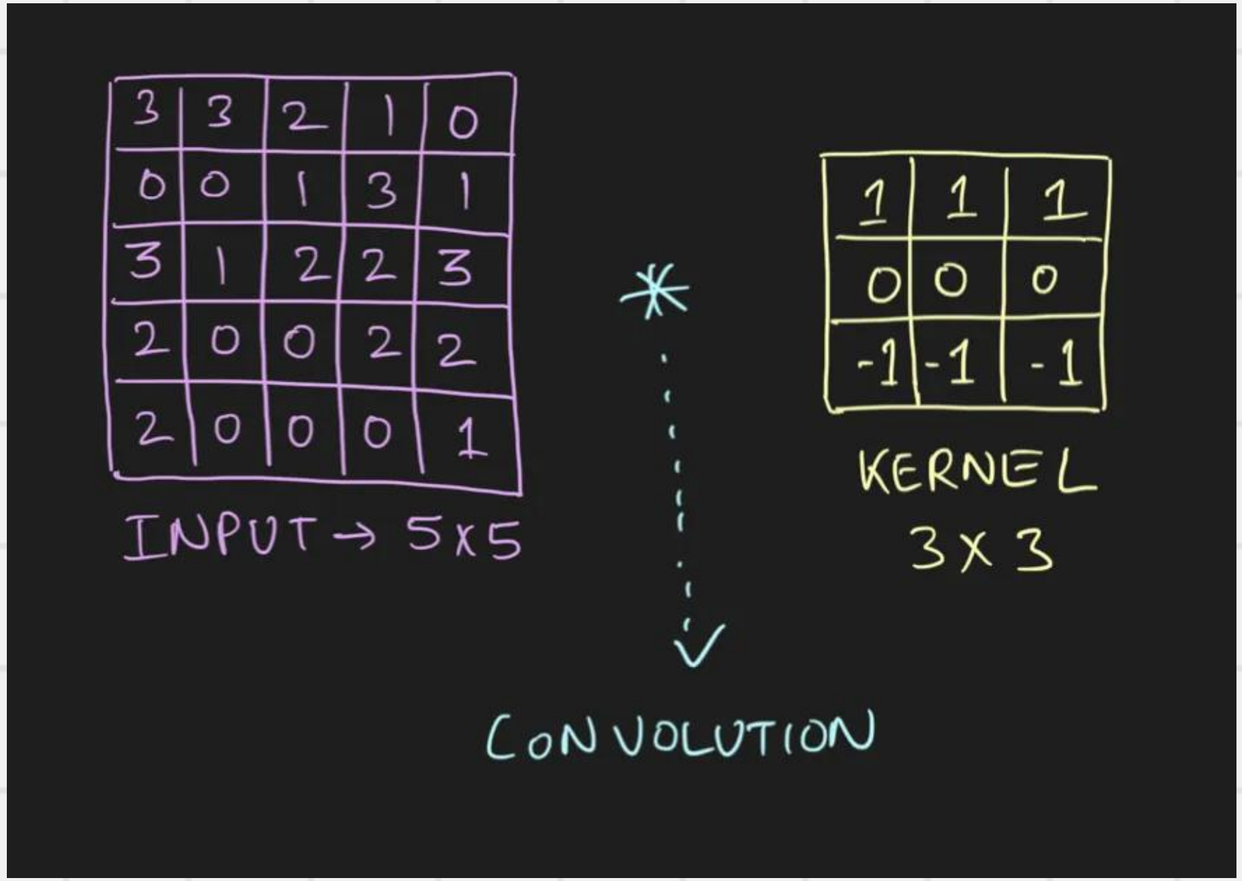
Convolution Theory (CV)

Now, we know what is the first input to the convolution operator, but how to transform this input and get the output feature matrix. Here comes the term ‘kernel’ which acts on the input image to get the required output.

Kernel: In an image, we can say that a pixel surrounding another pixel has similar values. So, to harness this property of the image we have kernels. A kernel is a small matrix that uses the power of localisation to extract the required features from the given image (input). Generally, a kernel is much smaller than the input image. We have different kernels for different tasks like blurring, sharpening or edge detection.



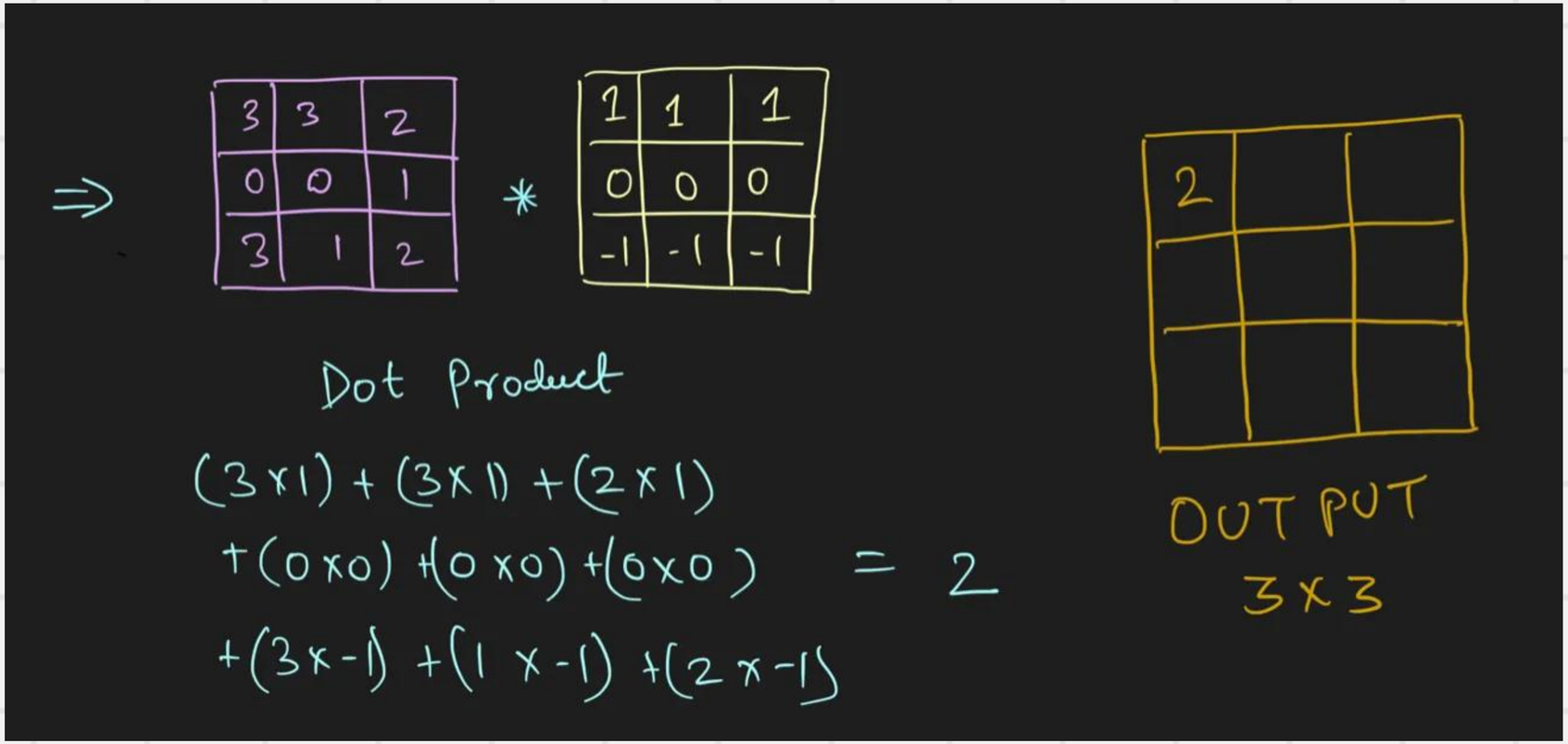
Convolution Theory (CV)



In the above image, for the first convolution, we will select a 3x3 region in the image (sequential order) and do a dot product with the kernel. Ta-da! This is the first convolution we did and we will move the region of interest, pixel by pixel(stride)..

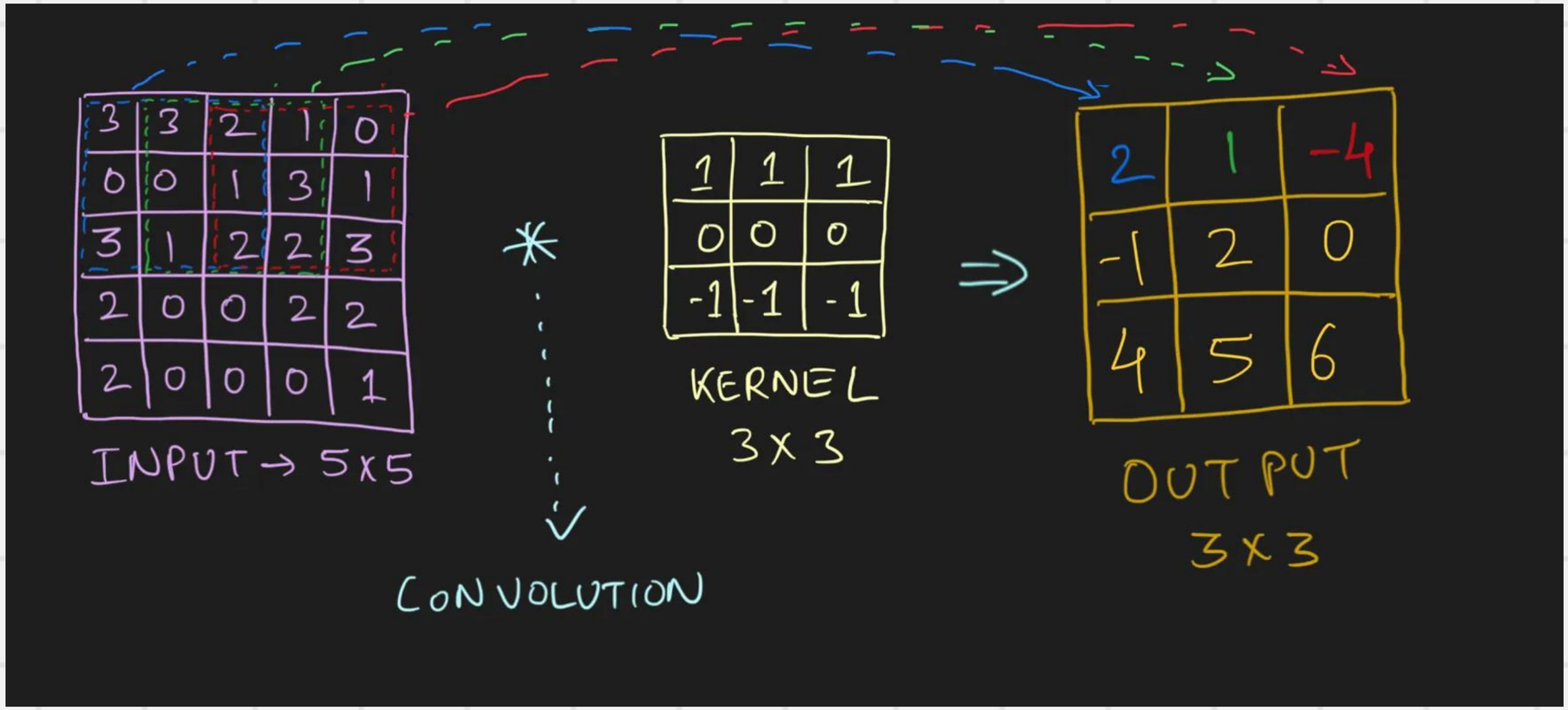
Convolution Theory (CV)

The dimension of the region of interest (ROI) is always equal to the kernel's dimension. We move this ROI after every dot product and continue to do so until we have the complete output matrix.



Convolution Theory (CV)

So by now, we know how to perform a convolution, what exactly is the input, and what the kernel looks like. But after every dot product, we slide the ROI by some pixels (can skip 1, 2, 3 ... pixels). This functionality is controlled by a parameter called stride.



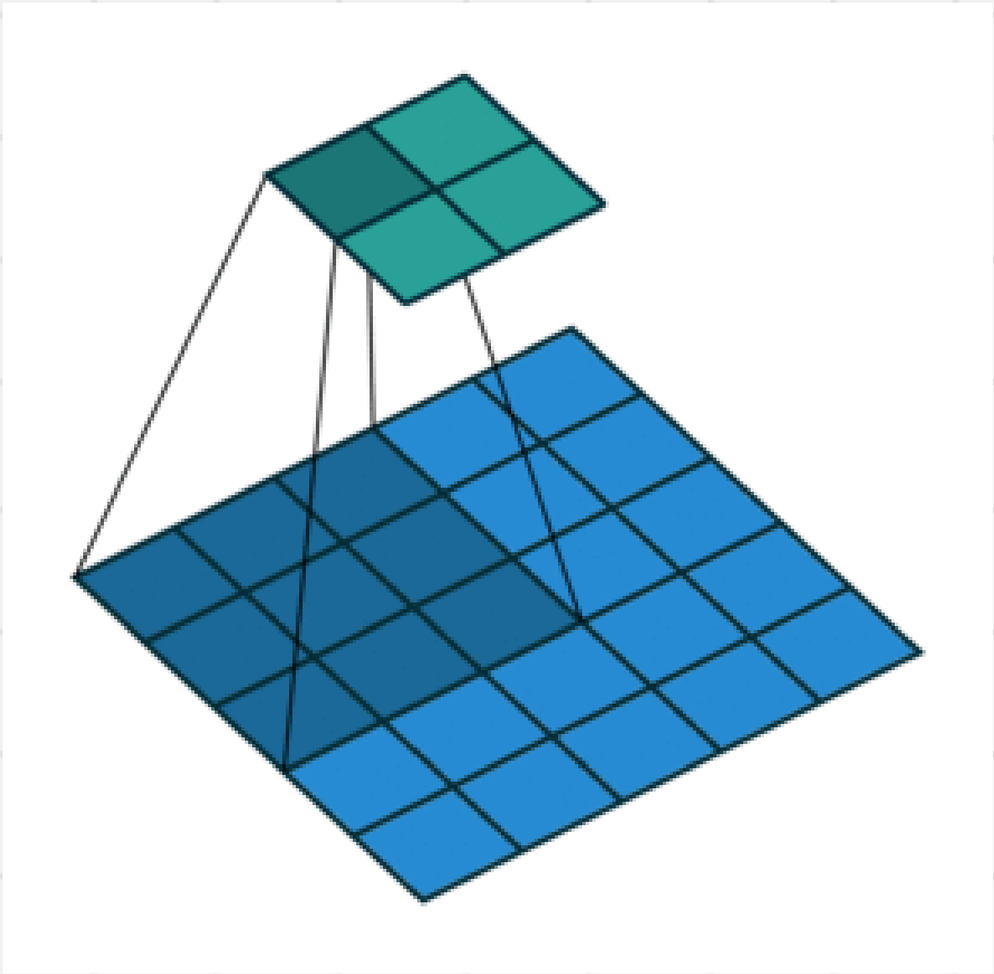
Convolution Theory (CV)

Stride: It is a parameter which controls/modifies the amount of movement of the ROI in the image. The stride of greater than 1 is used to decrease the output dimension. Intuitively, it skips the overlap of a few pixels in every dot product, which leads to a decrease in the final shape of the output.

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolution Theory (CV)



In the previous image, we always move the ROI by 1 pixel and perform the dot product with the kernel. But if we increase the stride, let's say stride = 2, then the output matrix would be of dimension -> 2 x 2.

Convolution Theory (CV)

📌 Source & References

The content in the previous section is directly taken from:

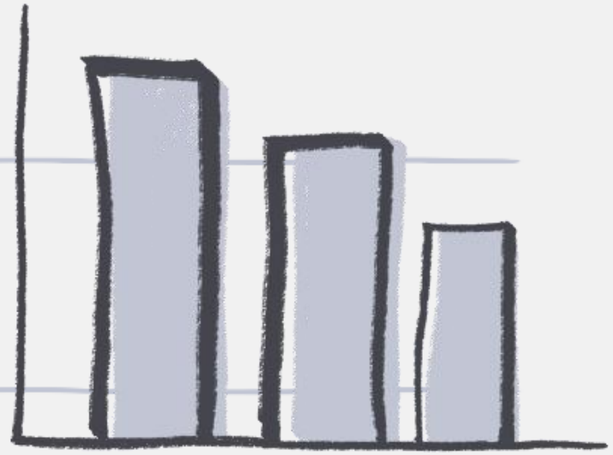
🔹 Medium - Computer Vision: Convolution Basics

Author: Harsh Yadav

Published on: Jul 5, 2022

All text and explanations in this part are copied from the above article without modifications.

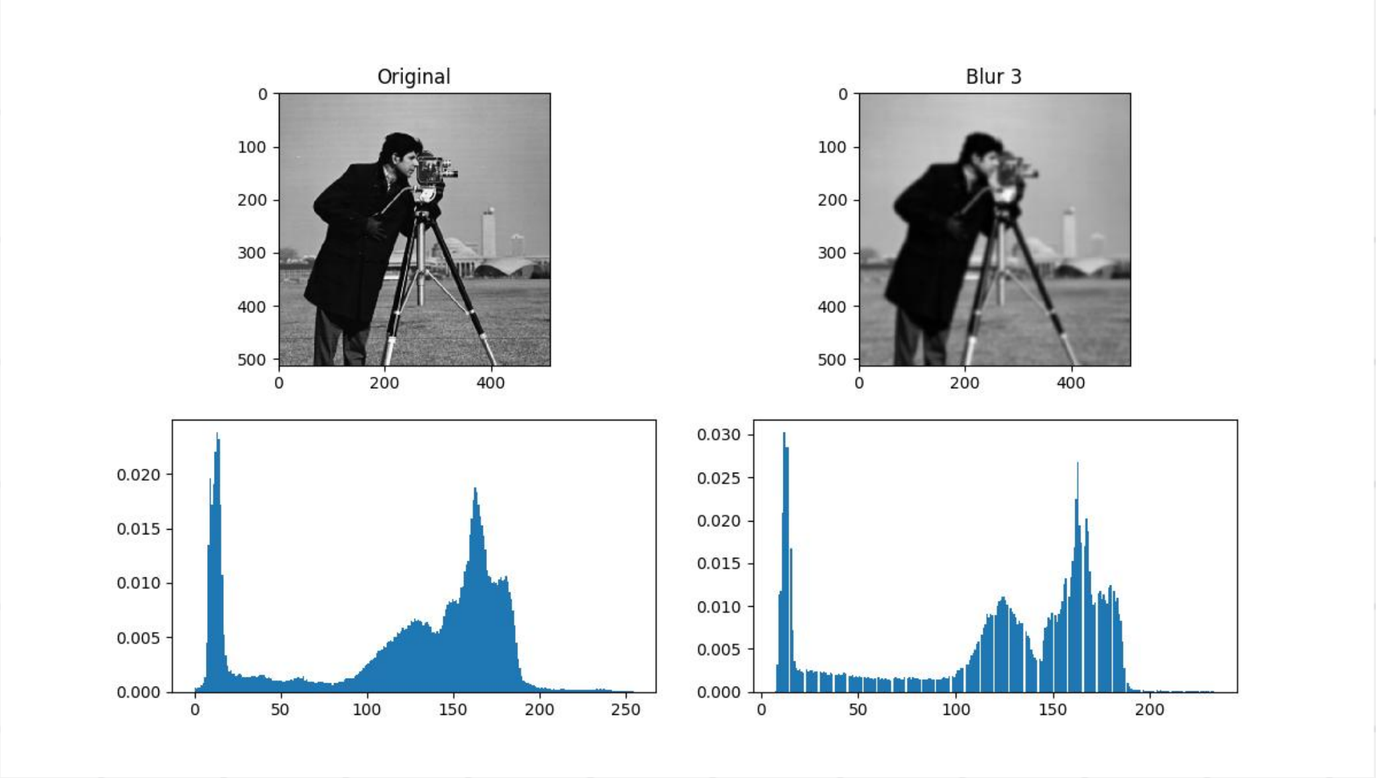
03



Blurring Filters

Blurring Filters

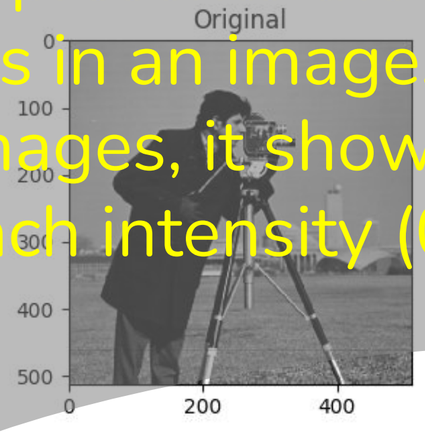
Blurring filters are image processing techniques that apply a convolution operation to reduce high-frequency details, smoothing intensity variations and minimizing noise while preserving the overall structure of the image.

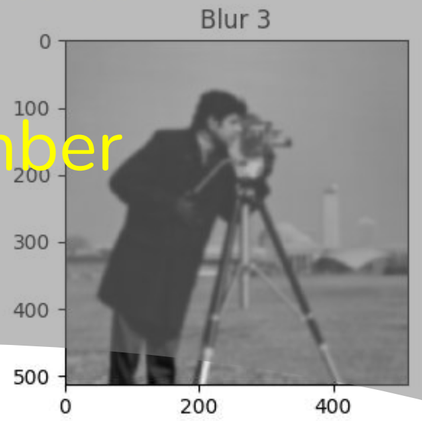


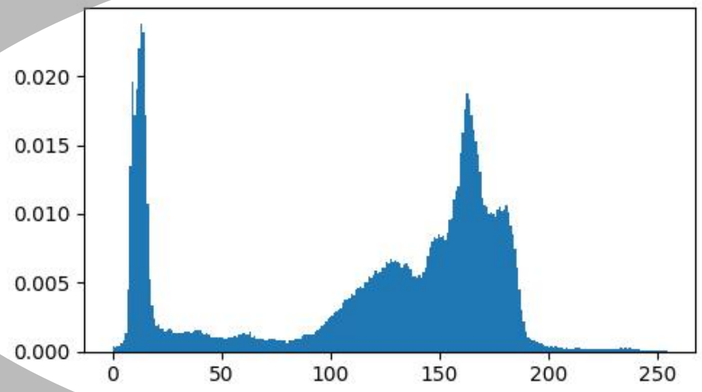
Blurring Filters

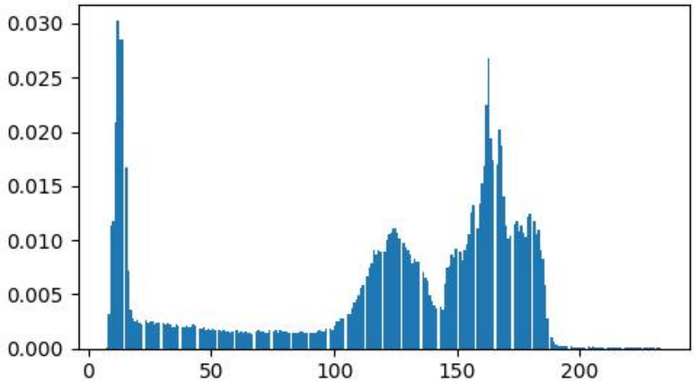
These 2 plots called histogram.

- A histogram represents the distribution of pixel intensities in an image.
- In grayscale images, it shows the number of pixels for each intensity (0-255).



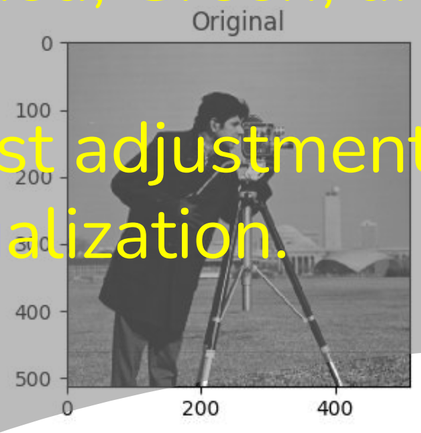


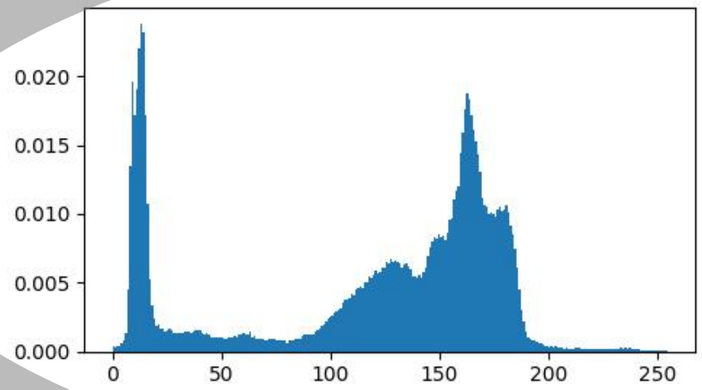


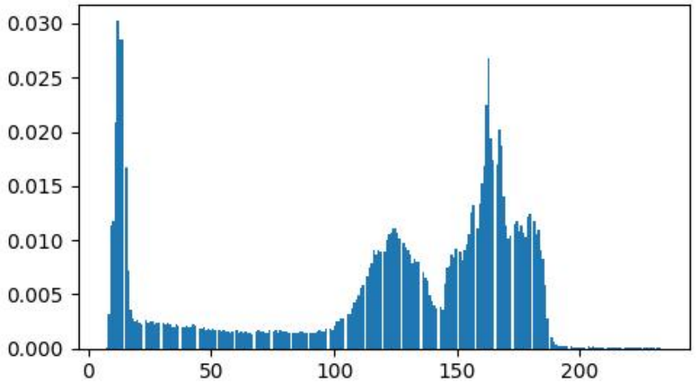


Blurring Filters

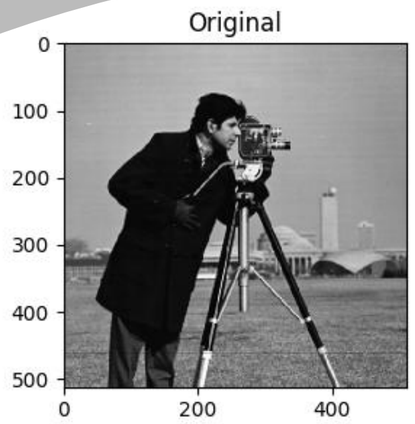
- In color images, separate histograms can be drawn for Red, Green, and Blue channels.
- Used in contrast adjustment, thresholding, and image equalization.

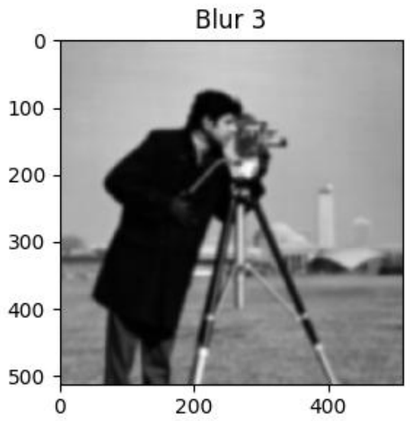


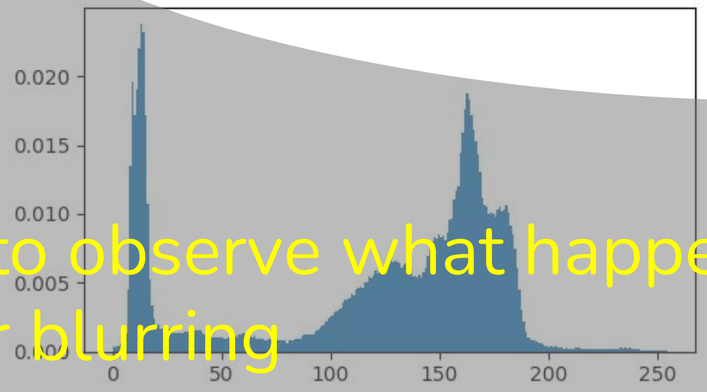


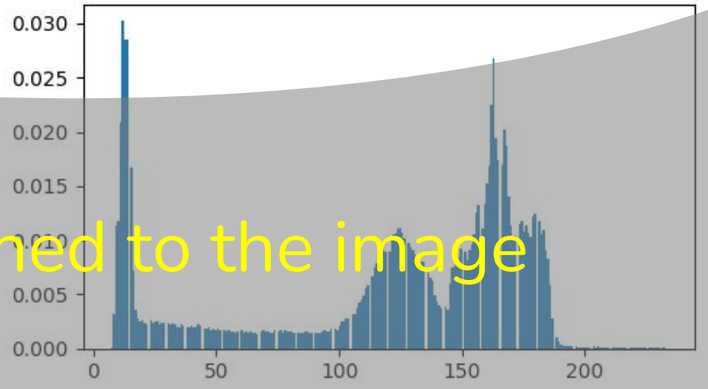


Blurring Filters



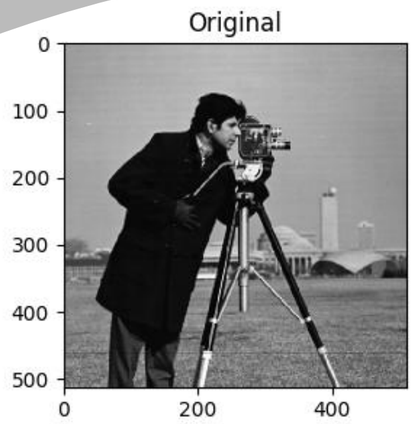


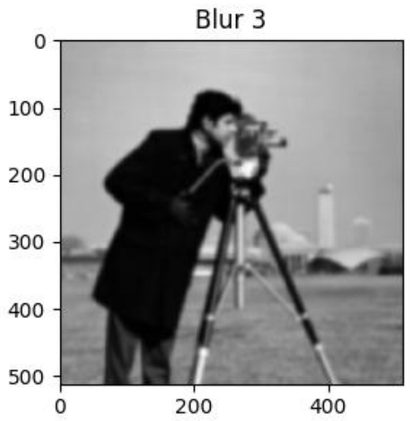


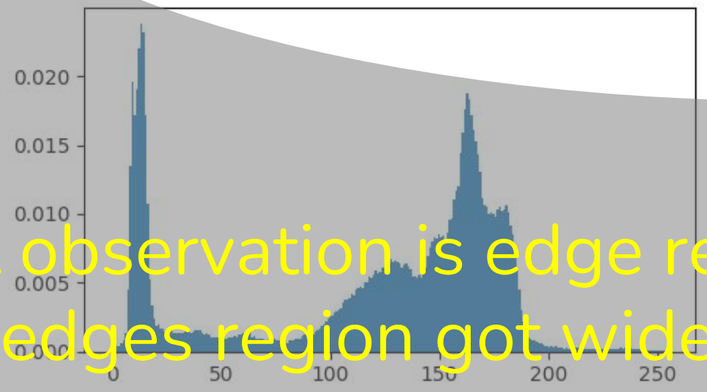


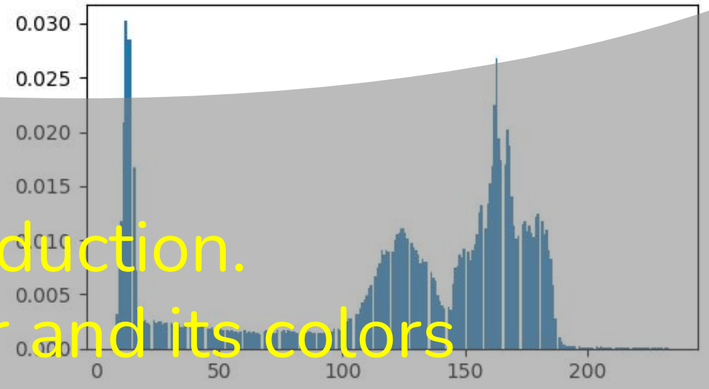
Try to observe what happened to the image after blurring

Blurring Filters



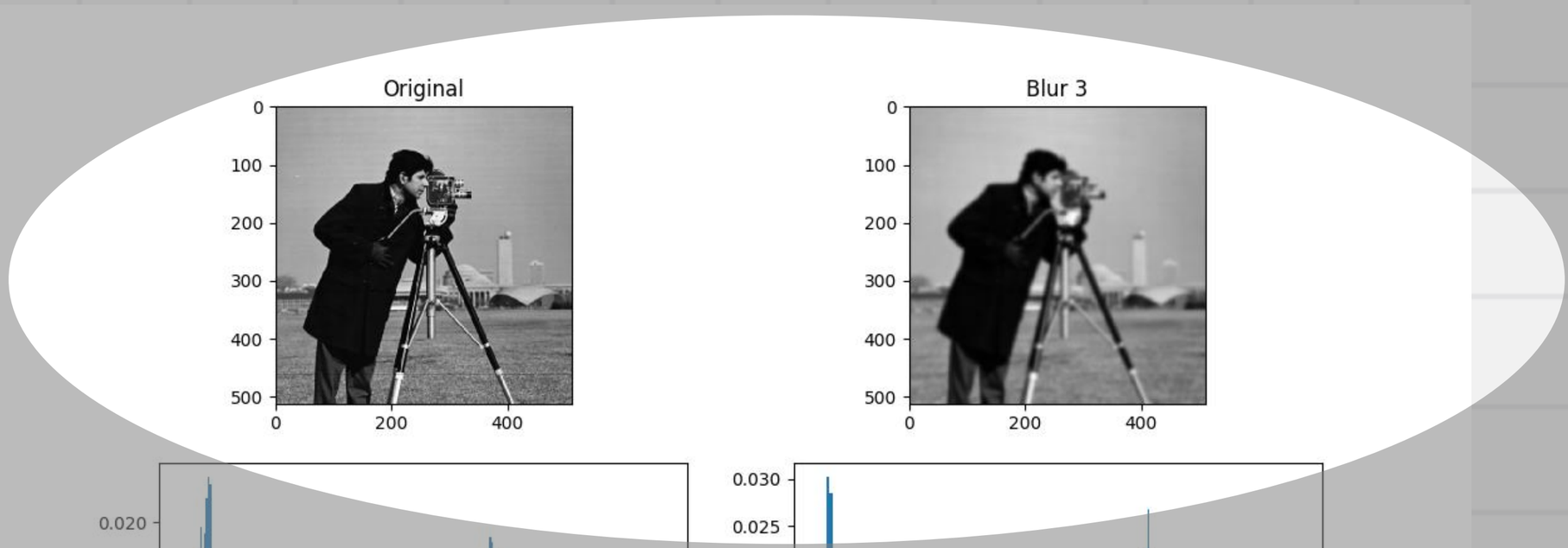






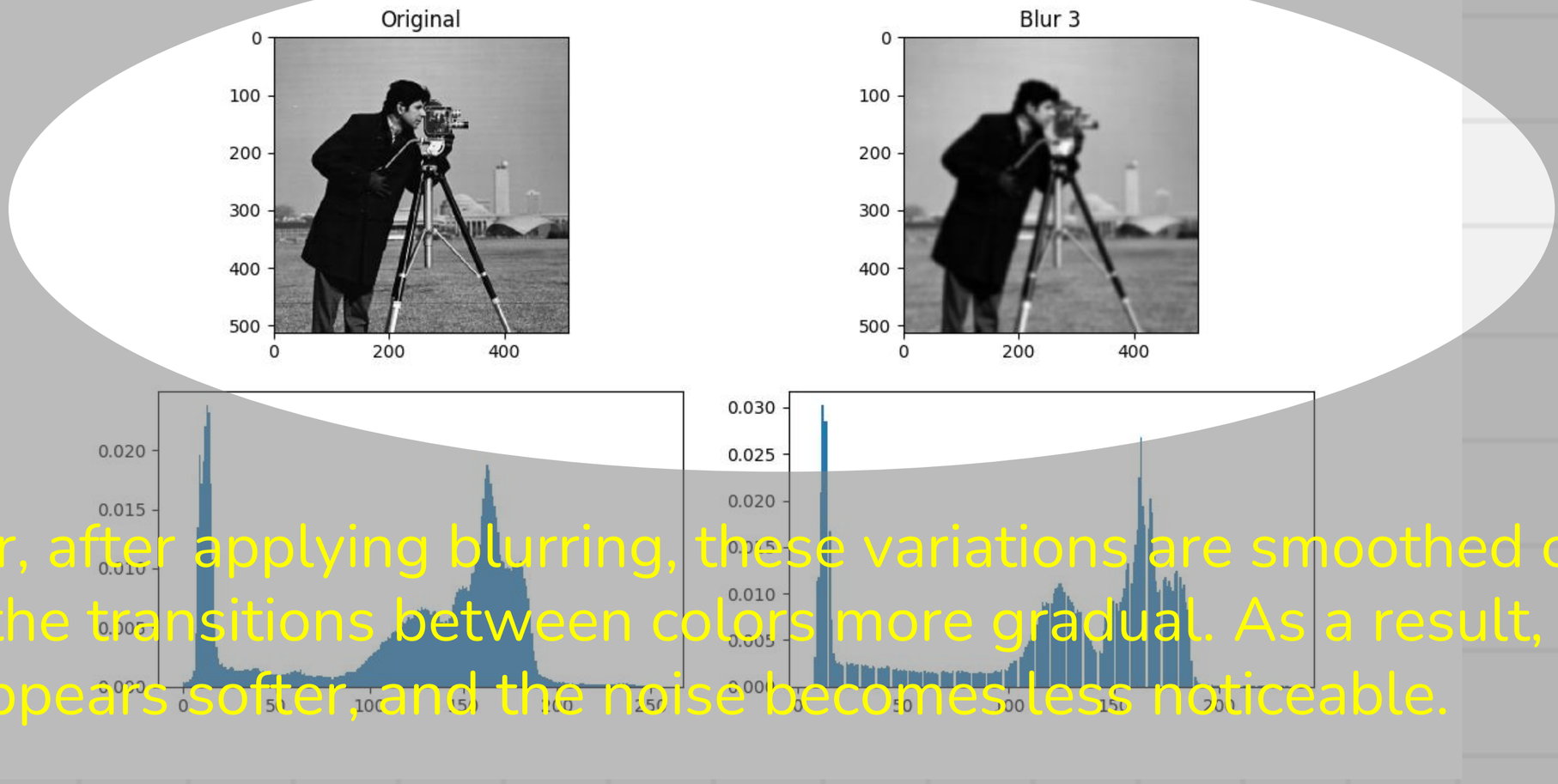
First observation is edge reduction.
The edges region got wider and its colors brighter.

Blurring Filters



The second important observation is denoising. If we zoom in on the original image, especially in the sky region, we can notice the roughness of pixels, where small variations in brightness make the color differences between pixels easily visible.

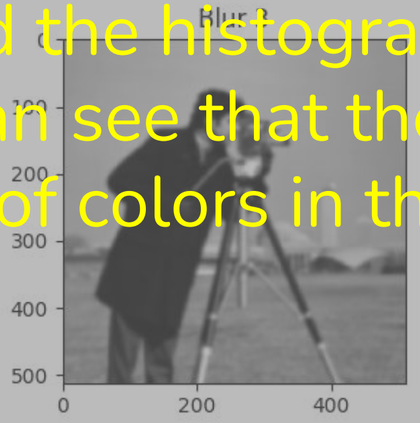
Blurring Filters

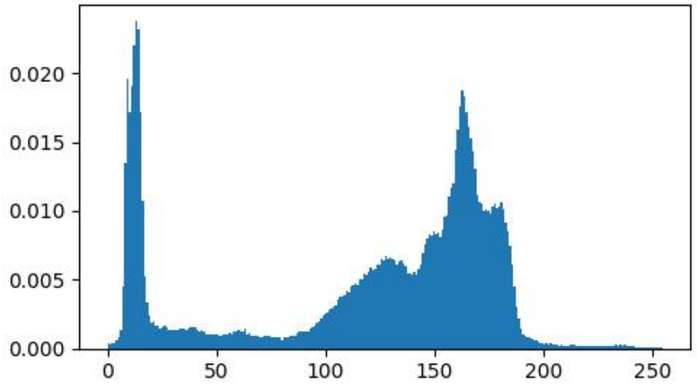


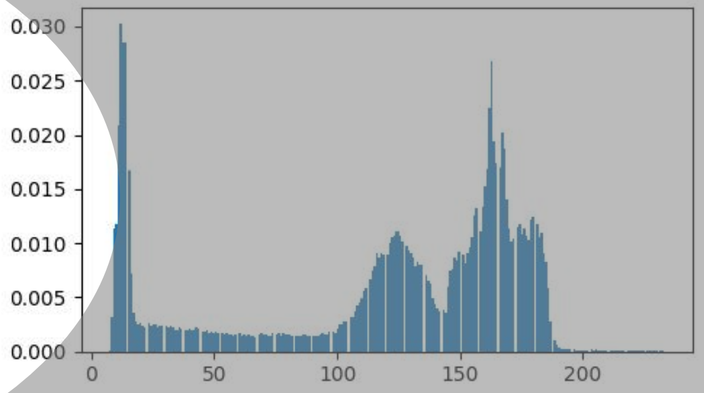
However, after applying blurring, these variations are smoothed out, making the transitions between colors more gradual. As a result, the image appears softer, and the noise becomes less noticeable.

Blurring Filters

Now, let's examine how this affected the histogram.
First, in the original histogram, we can see that the bars are closely connected, indicating a wide variety of colors in the image.

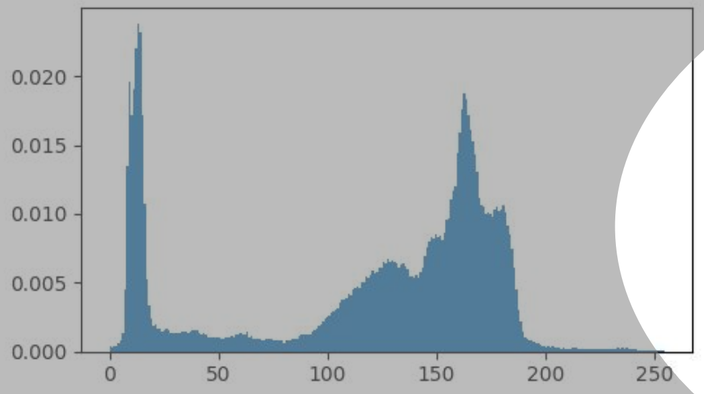


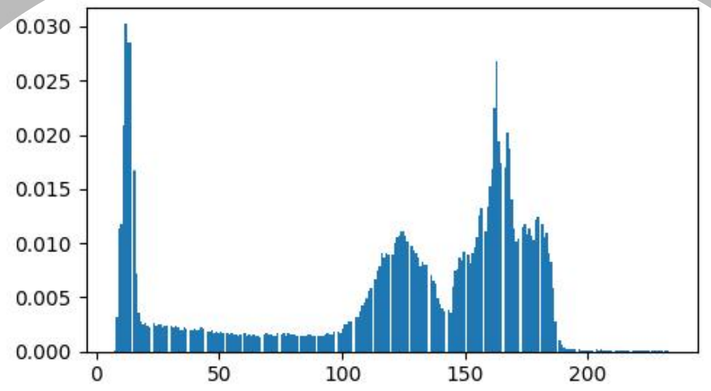




Blurring Filters

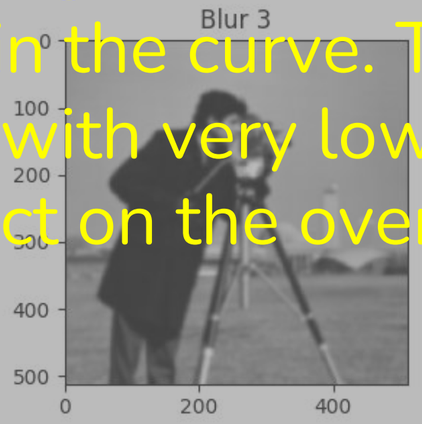
After applying the filter, the bars become more separated, which means the color variations have been reduced. This reduction in color diversity contributes to the denoising effect, making the image appear smoother.

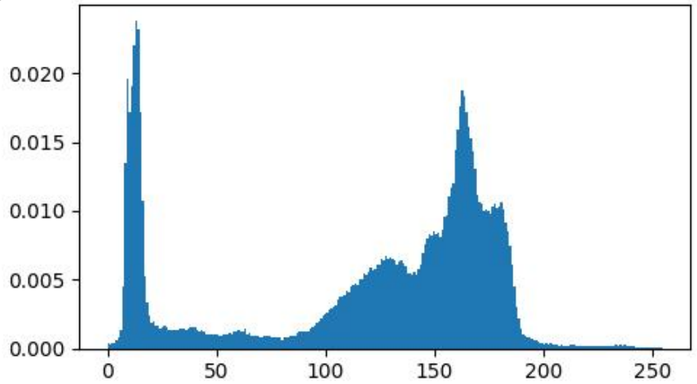


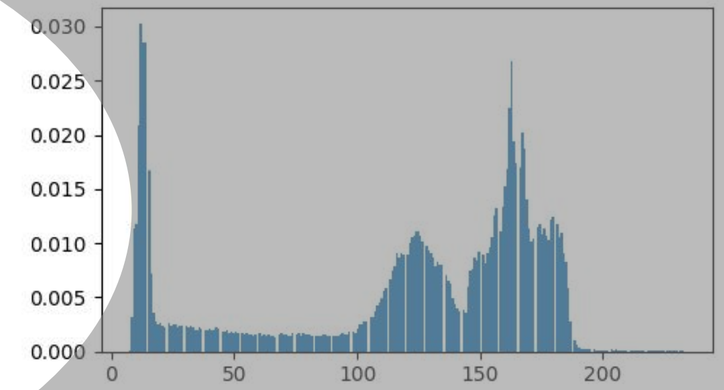


Blurring Filters

Secondly, if you examine the original histogram in the range of (20–90), you will notice rough, uneven variations in the curve. This indicates the presence of insignificant color variations with very low relative frequency, meaning they have little impact on the overall appearance of the image.

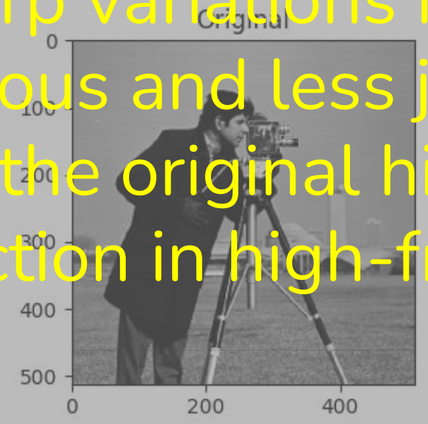


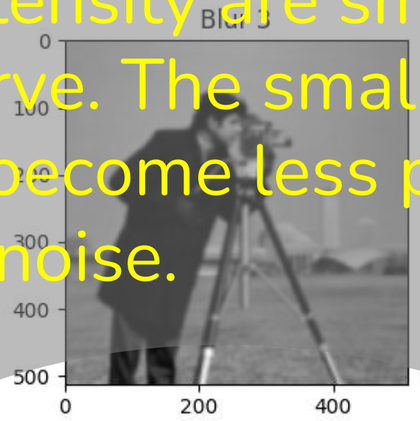


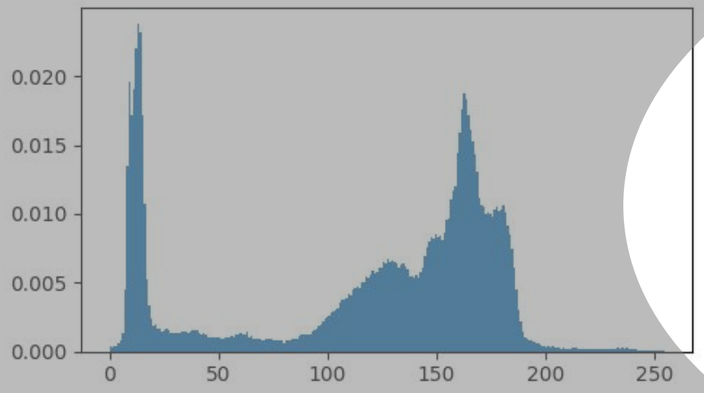


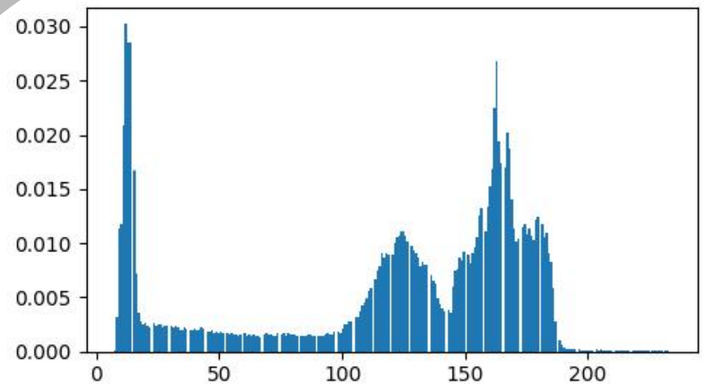
Blurring Filters

After applying the blurring filter, the histogram undergoes noticeable changes. The sharp variations in pixel intensity are smoothed out, resulting in a more continuous and less jagged curve. The small gaps and rough bends present in the original histogram become less pronounced, indicating a reduction in high-frequency noise.



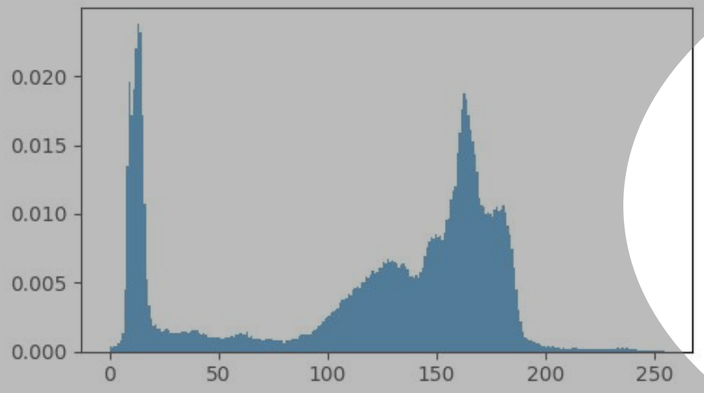


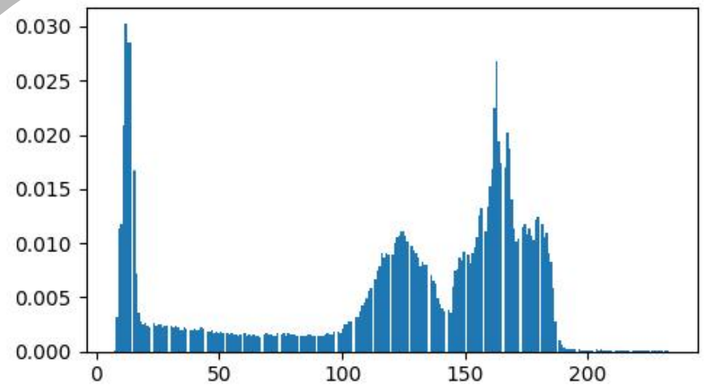




Blurring Filters

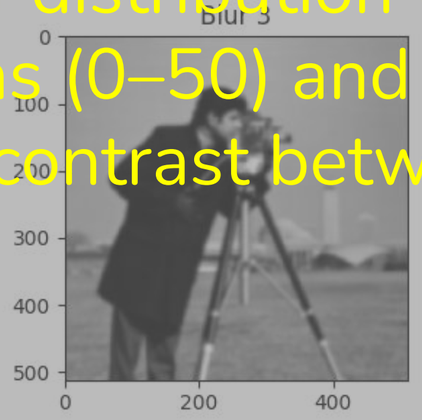
Additionally, the overall distribution becomes more compact, as extreme dark and bright values blend with their neighboring intensities. This effect reduces contrast slightly but helps create a smoother, less noisy image.

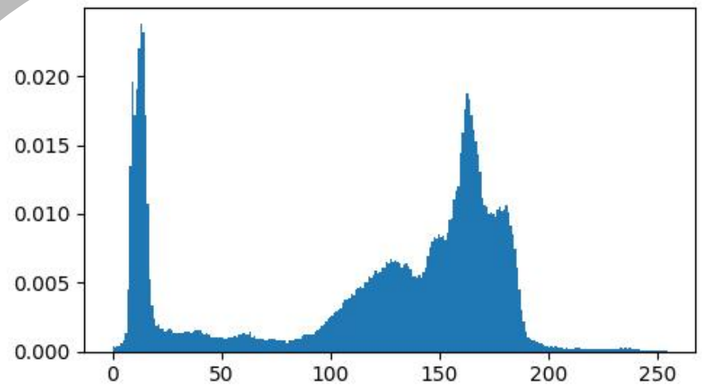


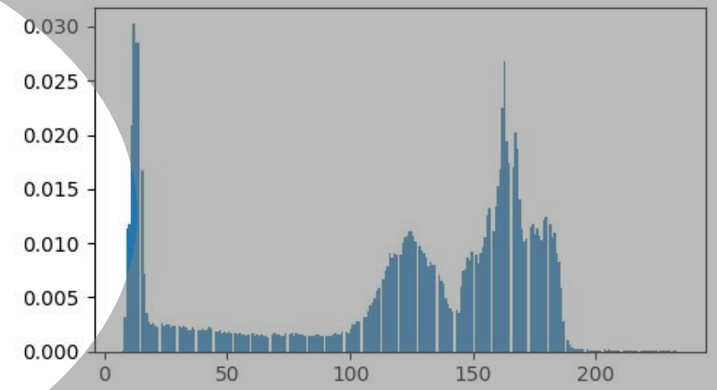


Blurring Filters

In the original histogram, we see a wider distribution with distinct peaks and valleys, especially in the dark regions (0–50) and bright regions (200–255). These peaks represent high contrast between different parts of the image.

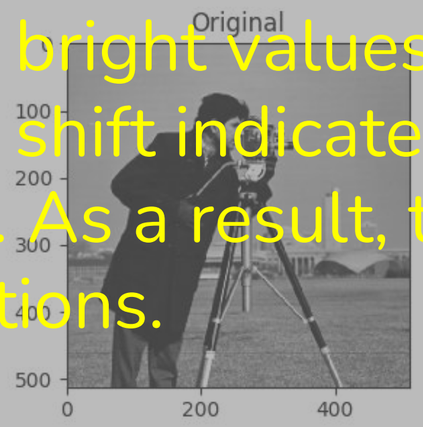


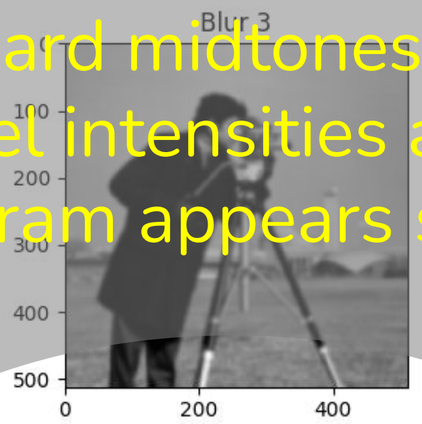


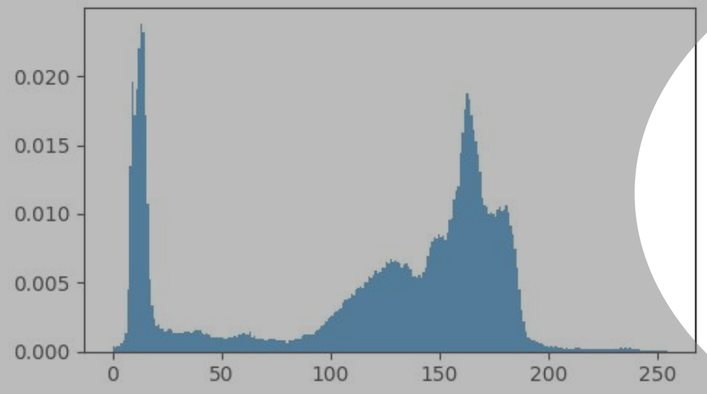


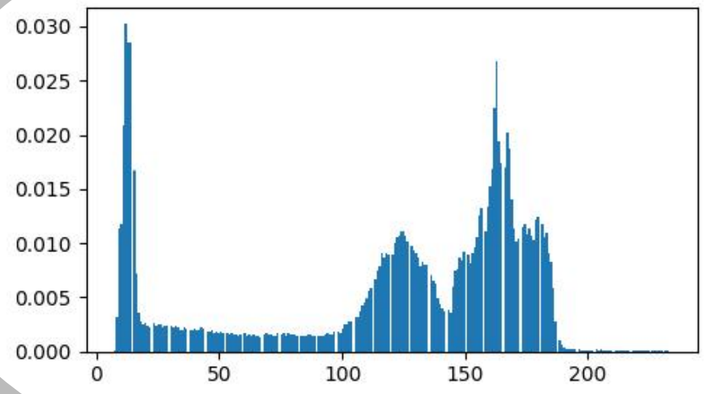
Blurring Filters

However, after blurring, the histogram becomes more compressed, as extreme dark and bright values shift toward midtones (around 100–180). This inward shift indicates that pixel intensities are averaging out, reducing contrast. As a result, the histogram appears smoother, with fewer sharp variations.









Blurring Filters

Now, we want to understand the math which is behind the scene of blurring filters.

We will know 2 types of blurring filters:

1.Mean Filter

2.Gaussian Filter

Mean Filter

The Mean Blur (sometimes referred to as Box or Average Blur), looks at each pixel of an image and replaces its value with the average value of all of its surrounding pixels.
In our example, we will be using a simple 3x3 matrix kernel where all values are 1.

The box blur kernel:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Mean Filter

While the above kernel is incredibly simple — it's going to help us understand how convolutions work. In this convolution, because the matrix values are all 1, the result of each window is the average value of the center pixel and its neighbors. Since we are using a sliding window, we cannot update the pixel value in-line (otherwise it would impact the next time the window shifts). We update a new image with the average value at the position corresponding to the center of the kernel.

1. Multiply the 1st neighbor pixel by the 1st value in the Kernel

125	213	98	203	202	170
104	145	161	204	201	157
72	8	209	202	194	144
73	9	202	201	194	156
81	15	189	185	181	144
15	189	185	194	227	158

Original Image

1	1	1
1	1	1
1	1	1

3x3 Box
Blur Kernel

Mean Filter

Using a sliding window, the convolution will process all of the pixels in the image. Once the convolution is completed, our new image will appear blurred.

Box Blur Convolution

125	213	98	203	202	170
104	145	161	204	201	157
72	8	209	202	194	144
73	9	202	201	194	156
81	15	189	185	181	144
15	189	185	194	227	158

Original Image

Blurred Image

Mean Filter

In this animation, notice the pixels with the values of 8 and 9. These two pixels, in particular, are outliers compared to all the other pixels in the images. These pixels stand out from the rest of the pixels. After the convolution, these pixel values are replaced with 109 and 95 bringing those values more in alignment with the neighbors.

Box Blur Convolution

125	213	98	203	202	170
104	145	161	204	201	157
72	8	209	202	194	144
73	9	202	201	194	156
81	15	189	185	181	144
15	189	185	194	227	158

Original Image

Blurred Image

Mean Filter

As we can see, the result of the box blur is to “smooth” out the image by making pixels more like their neighbors — resulting in a blurred image.

Original Image

Image with Box Blur

Gaussian Filter

1	2	1
2	4	2
1	2	1

Another blur kernel worth discussing is the Gaussian Blur kernel. Instead of a direct average of all of the values surrounding the pixel, we give each of the pixels in the kernel a different weight. We give the center pixel the most priority and pixels furthest away from the center less priority. As a result, blurs using a Gaussian Kernel have a tendency to appear less “Boxy”. Here is one example of a simple and easy-to-read 3x3 Gaussian Kernel:

Gaussian Filter

The convolution of a Gaussian kernel is identical to the Box Blur kernel. Think of the box blur as dividing the sum of the kernel values, which totals 9. In the Gaussian kernel illustrated above, the sum of the kernel values is 16.

1. Multiply the 1st neighbor pixel by the 1st value in the Kernel

125	213	98	203	202	170
104	145	161	204	201	157
72	8	209	202	194	144
73	9	202	201	194	156
81	15	189	185	181	144
15	189	185	194	227	158

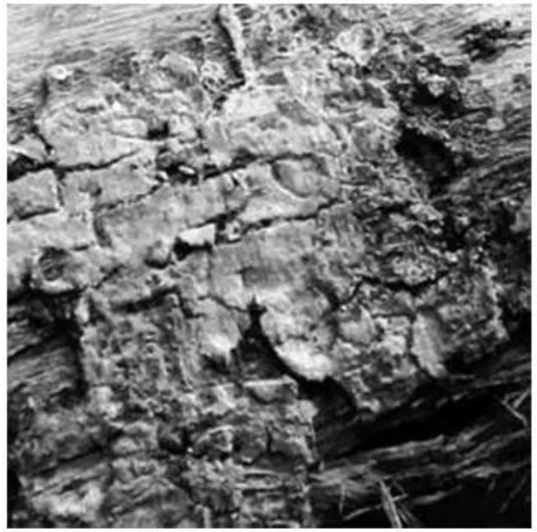
Original Image

1	2	1
2	4	2
1	2	1

3x3
Gaussian Kernel

Gaussian Filter

Notice the difference between Mean blur and gaussian



Original Image

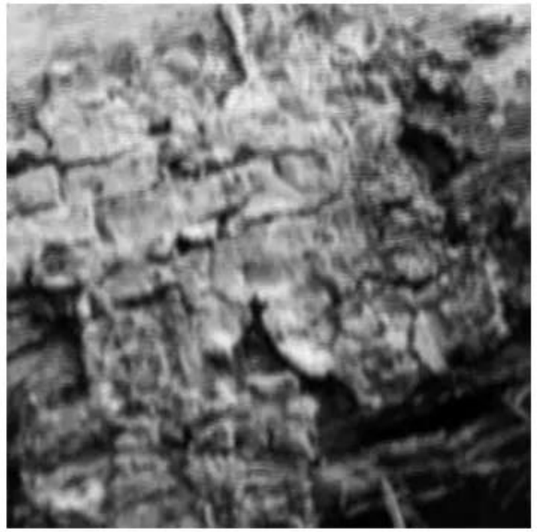


Image with
Box Blur

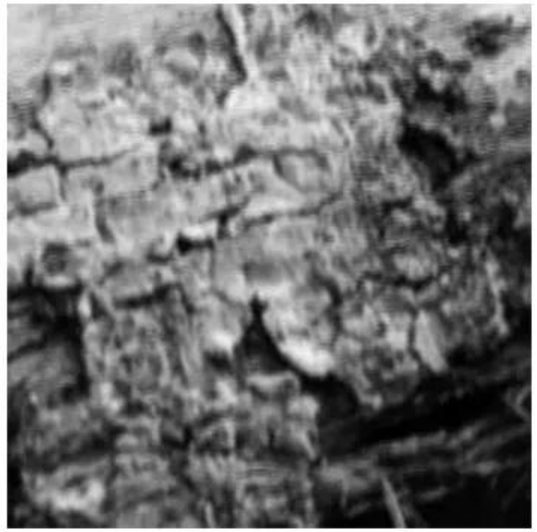
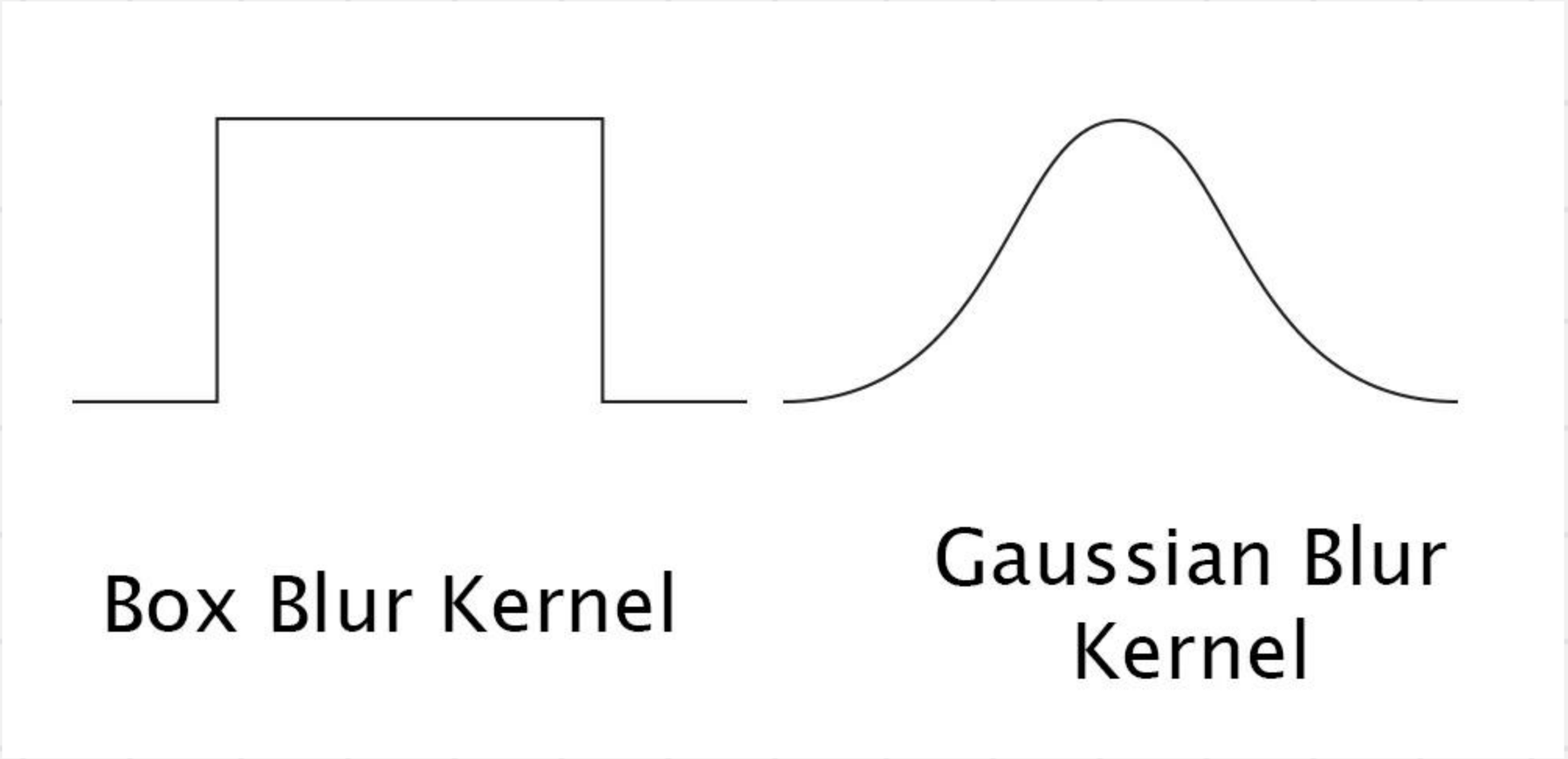


Image with
Gaussian Blur

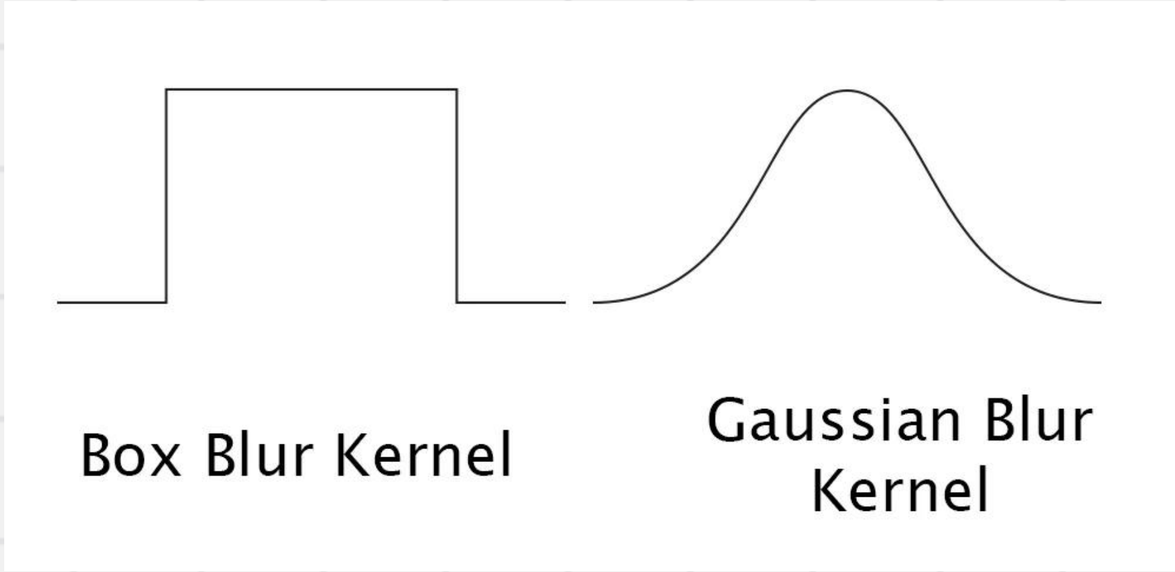
Gaussian Filter

As a quick visualization exercise, let's also compare what the Box Blur and Gaussian Blur kernels would look like if we were to plot them in a 2D graph.



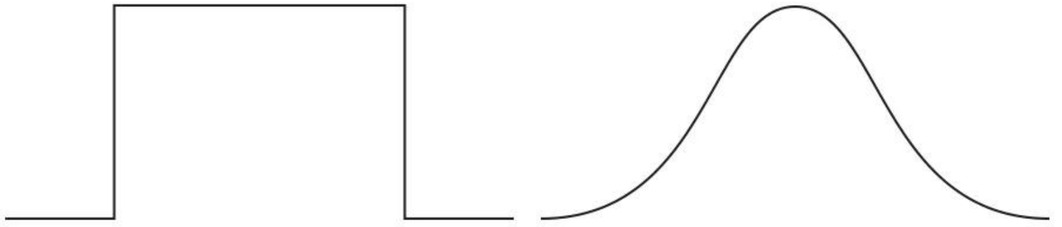
Gaussian Filter

The Gaussian kernel illustrated above is a very simple kernel with simple whole numbers. The specific weights of a Gaussian Kernel can be calculated based on the size (the width of the curve) and the strength (the height of the curve) of the Kernel. While the specific math of how we get to the size and values of a Gaussian kernel is a bit beyond the intention of these articles, it's important to point out that the calculations of the Gaussian kernel are dependent on the sigma value — a number that describes the smoothness of the Gaussian curve (Standard Deviation, for the Math Nerds).



Gaussian Filter

That is, the sigma represents the relationship between the numbers in the center of the kernel to the edges of the kernel. We will revisit the importance of sigma when we discuss the Differences of Gaussian feature detector later. For now, consider that the sigma will represent how much the Gaussian kernel smooths out an image. The larger the sigma, the stronger the blur.



Box Blur Kernel

Gaussian Blur Kernel

Blurring Filters

📌 Source & References

The content in Mean and Gaussian filters section is directly taken from:

🔹 Medium - Computer Vision for Busy Developers

Author: Vinny DaSilva

Published on: Jun 19, 2019

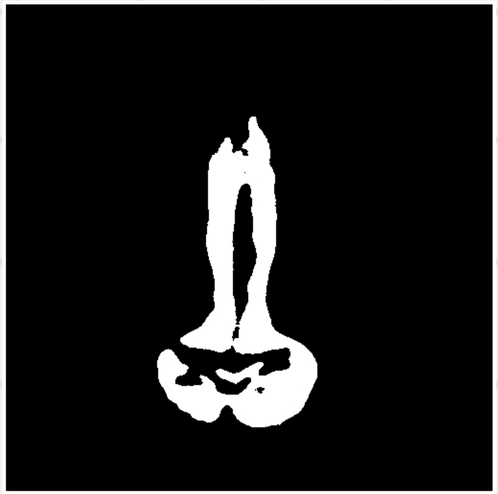
All text and explanations in this part are copied from the above article without modifications.

Blurring Filters

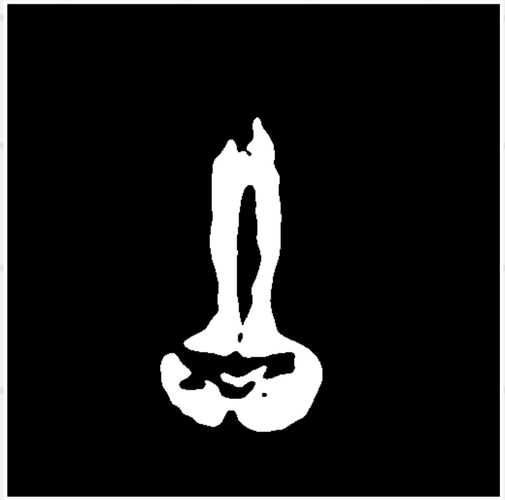
Now, let’s look on chimpanzee’s nose after applying Blurring Filters

HSV Threshold

Mean Blur



Gaussian Blur



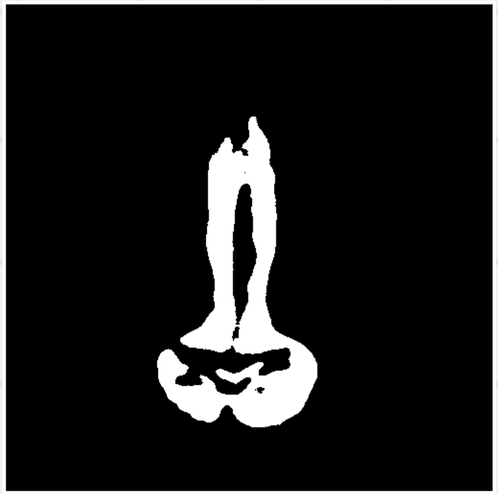
Blurring Filters

Now the noise is almost done. But we still have two problems need to be answered.

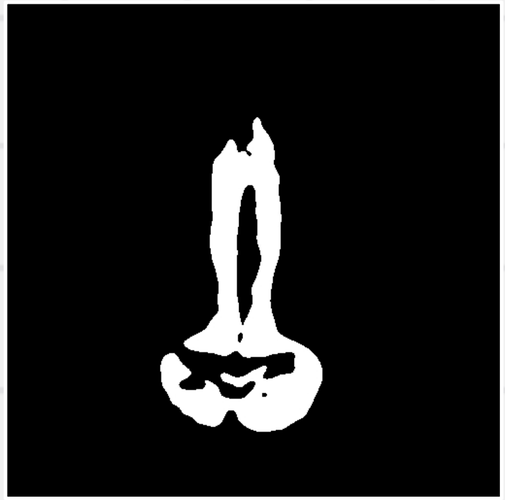
1. Do we have better solution for filtering the image from noise?
2. How would we fill the gaps of nose?

HSV Threshold

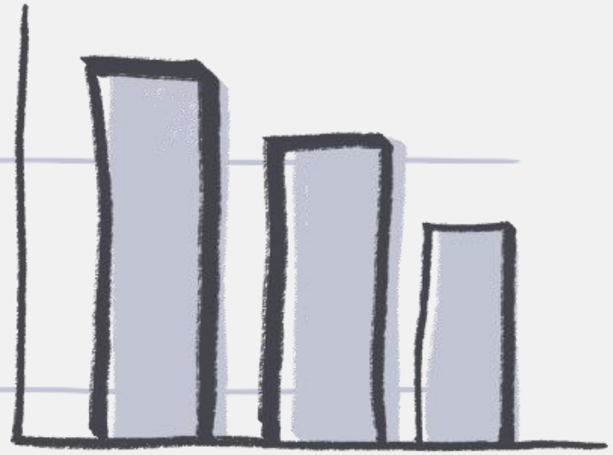
Mean Blur



Gaussian Blur

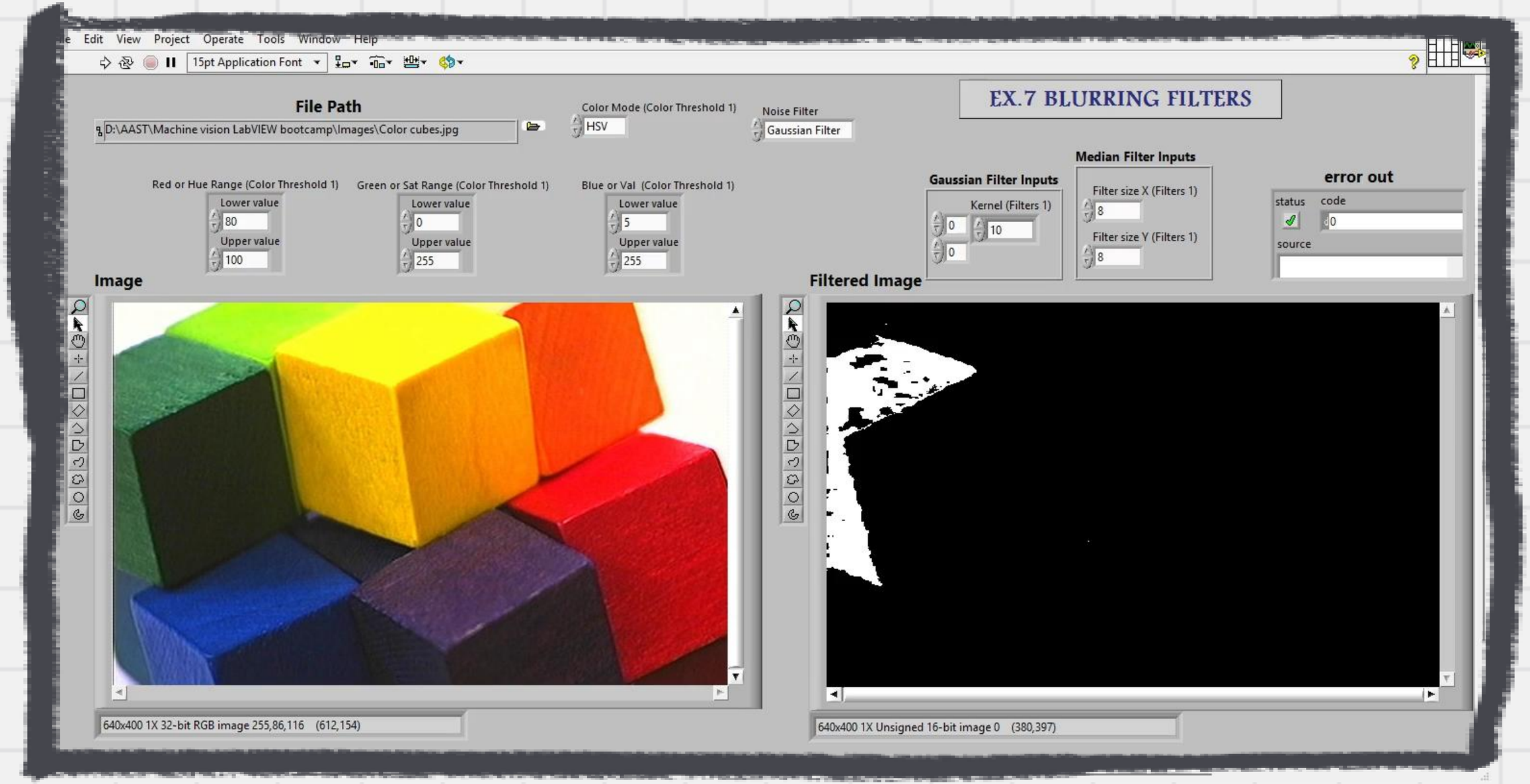


04

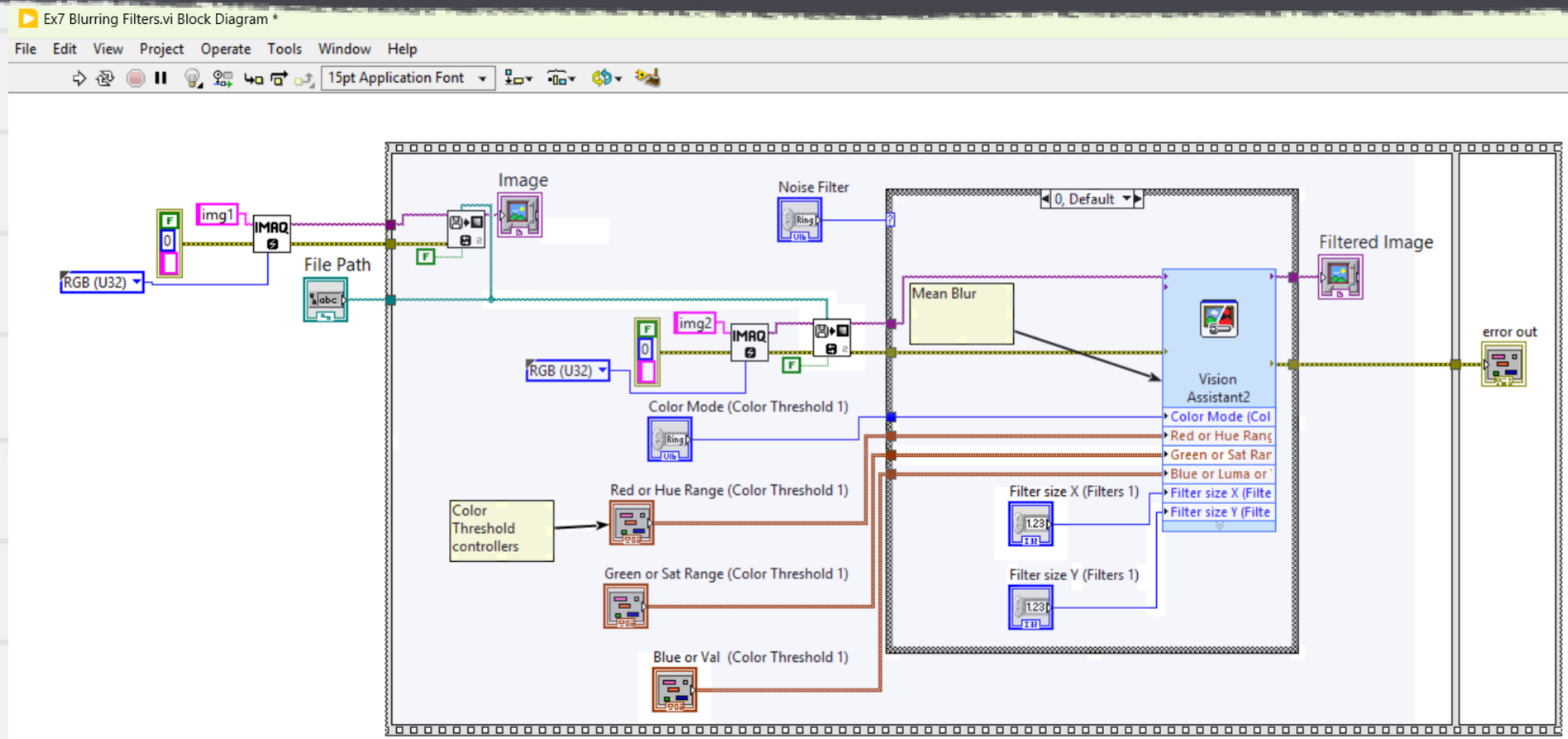


LabVIEW

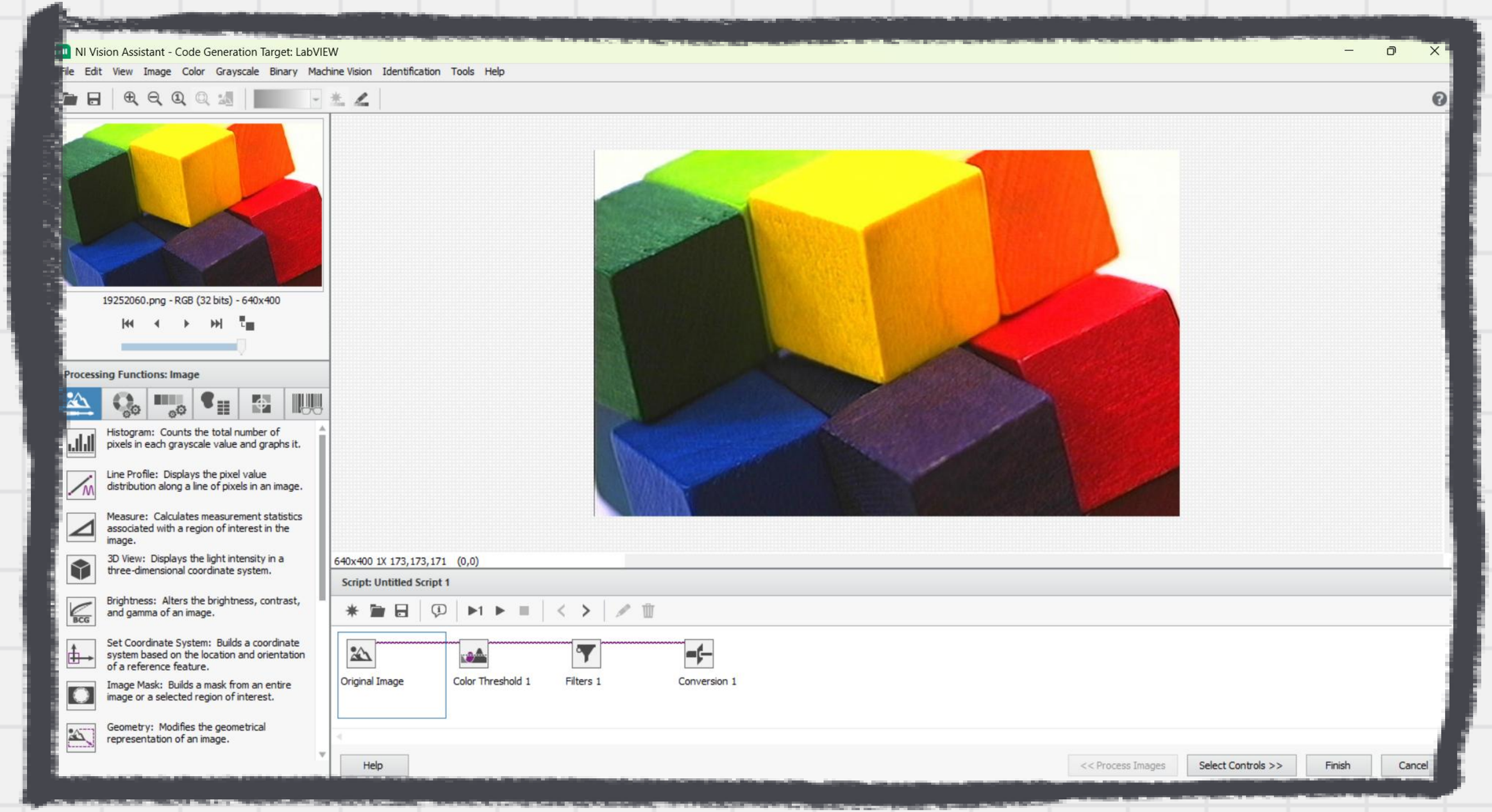
EX7. Blurring Filters



EX7. Blurring Filters



EX7. Blurring Filters




THANK
YOU