Blurring

Blurring is used to remove noise from an image. It involves computing the average of pixel values in the image, replacing every pixel with the average of all its neighboring pixels.

Functions

1. **blur()**
2. **GaussianBlur()**
3. **medianBlur()**
4. **bilateralFilter()**

---

blur()

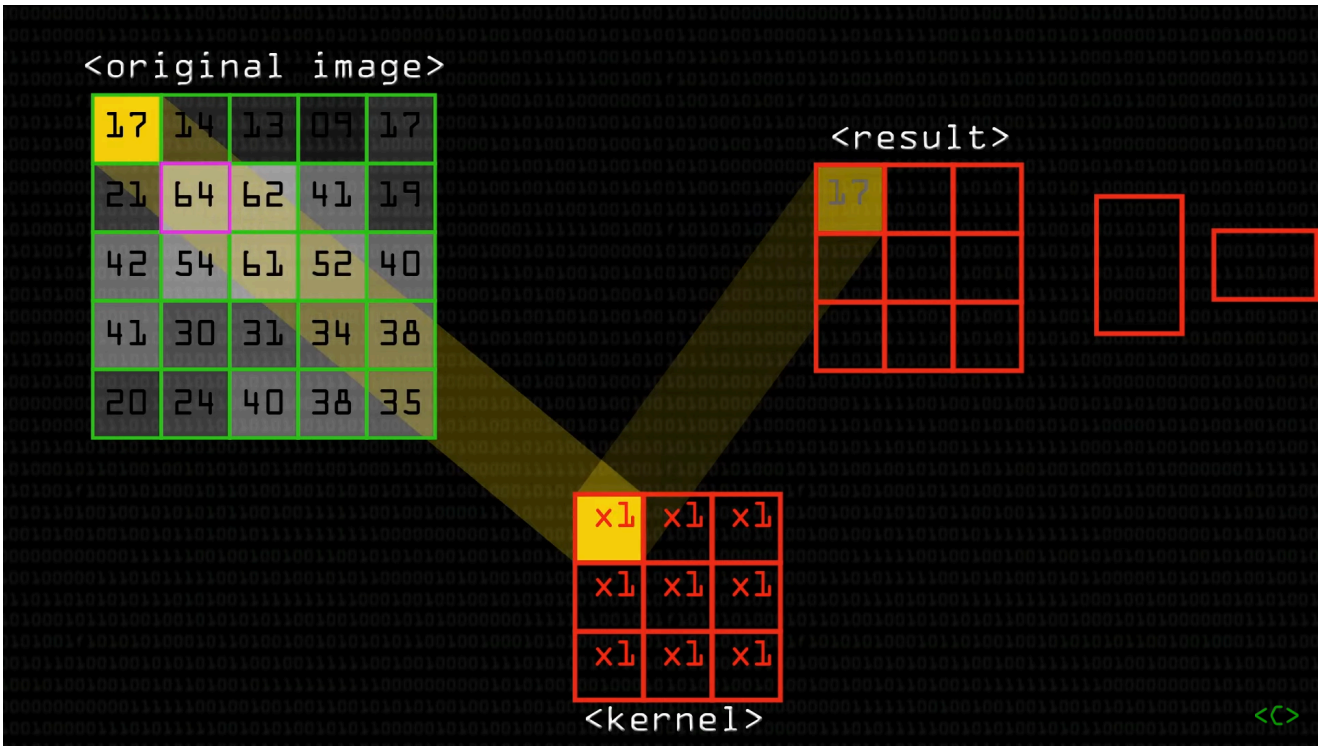
Classic Blur

```
k_size = 7  # 7x7 neighborhood will be considered for averaging
# Apply the blur using the defined kernel size (k_size x k_size)
blur = cv2.blur(img, (k_size, k_size))  # Each pixel will be replaced with
the average of its 7x7 neighbors
cv2.imshow("Blur", blur)
cv2.waitKey(0)
```

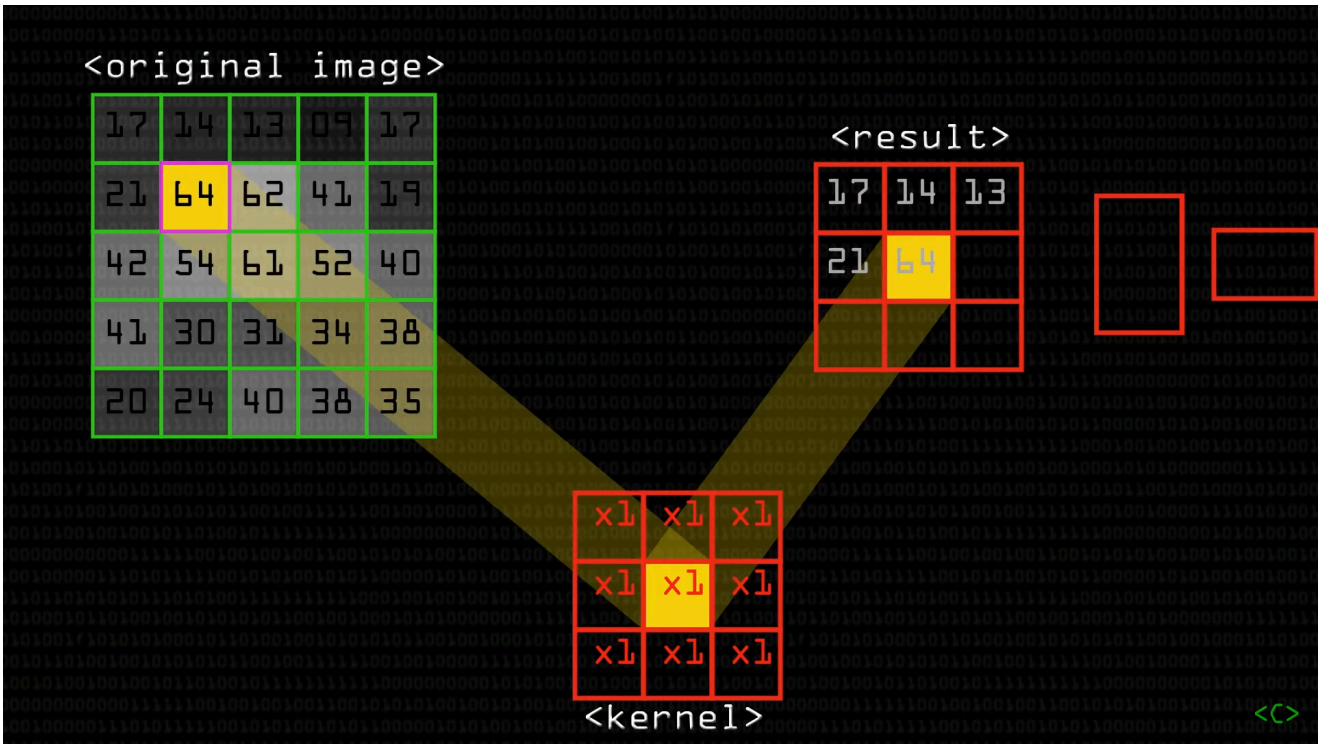
What is a kernel?

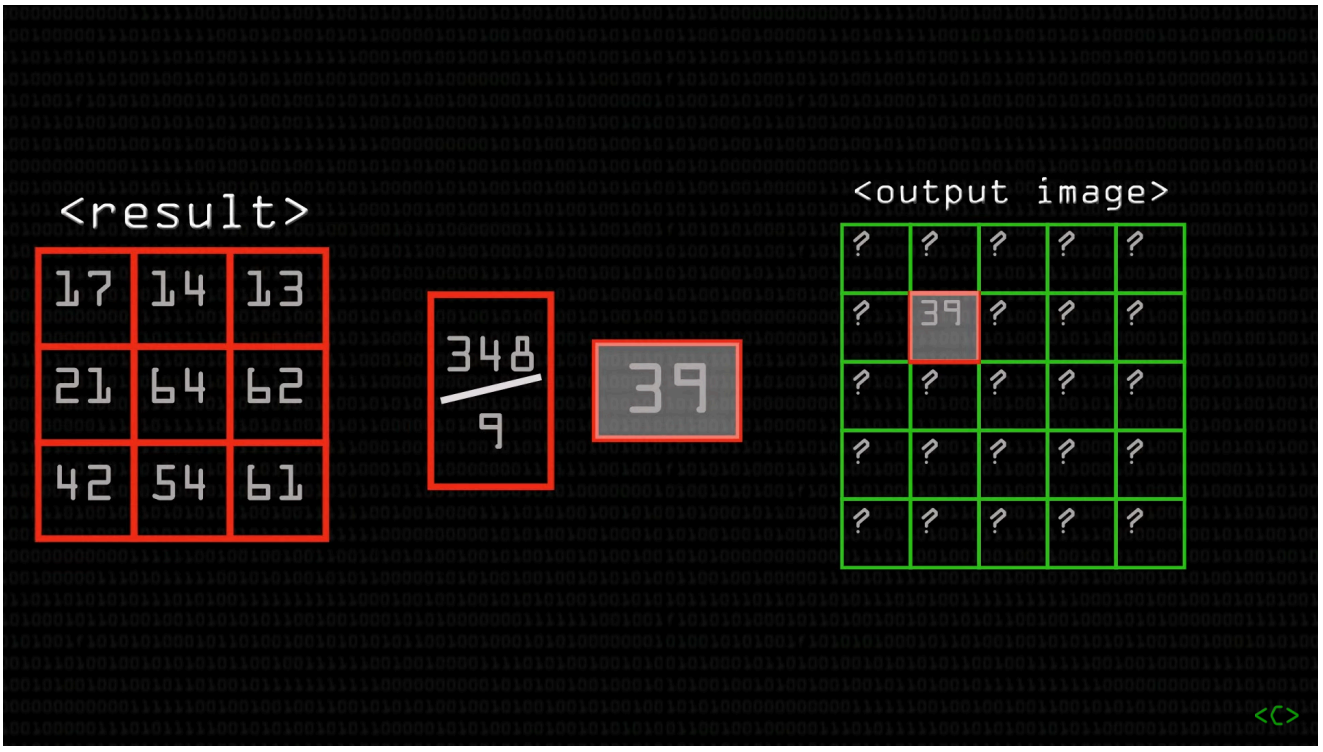
- A **kernel** is a small grid that moves over the image pixel by pixel.
  - A `1x1` kernel includes only the pixel itself.
  - A `3x3` kernel includes the pixel and its 8 neighbors (9 pixels total).
  - A `7x7` kernel includes the pixel and all pixels in a 7x7 grid (49 pixels total).

> **Note:** Kernel size is always an odd number to ensure the central pixel is well-defined.



[k_size =3, so for 64 pixel, kernel neighborhood is operated]





[Average is taken and 64 is replaced by 39. similarly all pixels will be replaced]

GaussianBlur():

```
k_size = 7
img_gaussian_blur = cv2.GaussianBlur(img, (k_size, k_size), 3)  # Standard deviation
cv2.imshow("GaussianBlur", img_gaussian_blur)
cv2.waitKey(0)
```

Definition

Gaussian blur produces a smoother, more natural blur effect compared to classic blur. It maintains natural-looking transitions between blurred areas and sharp edges.

Explanation

- **Effect**: The result is a soft blur with smooth falloff near edges, making it less noticeable.
- **Why it's preferred**: It is often used in image processing tasks like noise reduction, smoothing, and simulating lens blur.

Role of Standard Deviation (σ) in Gaussian Blur

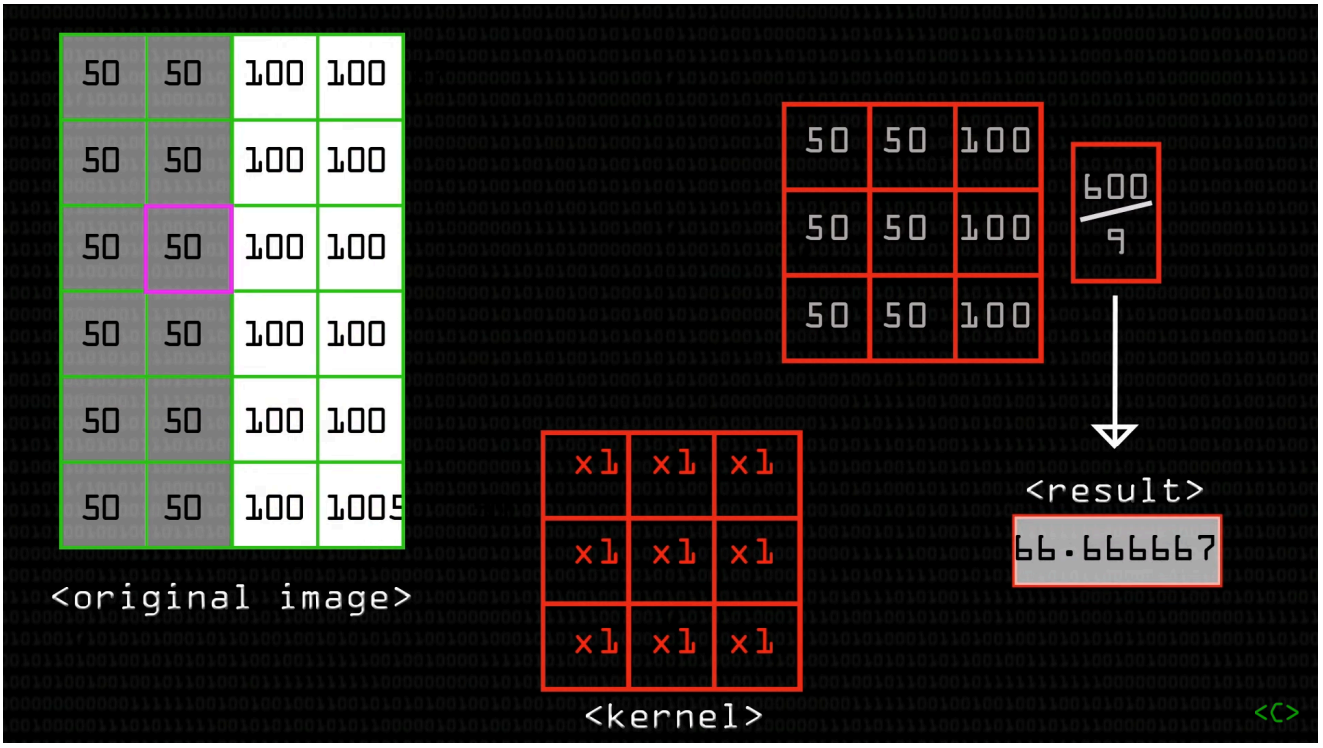
- **Why σ is important**: Controls how "wide" the blur effect is, influencing the radius of pixel influence.
- **Effect of σ**:

- **Small σ**: Creates a sharper, tighter blur where only nearby pixels contribute significantly.
- **Large σ**: Produces a wider blur where distant pixels contribute more, resulting in a pronounced blur effect.

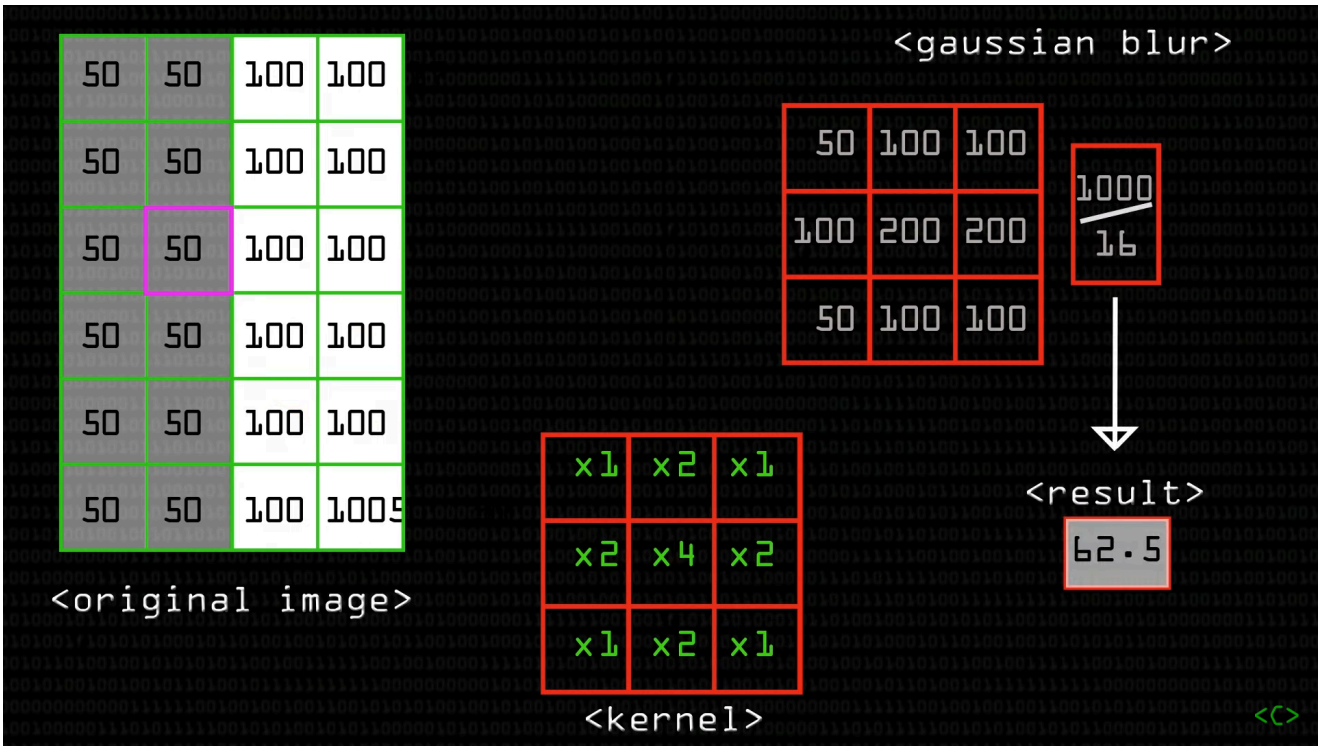
- **Visual Difference**:
  - Small σ: Slight softening of edges.
  - Large σ: Blurred image with edges blending into one another.

Difference between results in Normal Blur and Gaussian Blur :

[Normal Blur with 3x3 kernel]



[Gaussian Blur with 3x3 kernel]



medianBlur()

Median blur is similar to normal blur but uses the **median** value instead of the mean (average).

python

Copy code

```
k_size = 3
img_median_blur = cv2.medianBlur(img, k_size)
cv2.imshow("Median Blur", img_median_blur)
cv2.waitKey(0)
```

Comparison of Blur Types

Feature	Normal Blur	Gaussian Blur	Median Blur
Kernel Weights	All pixels equal weight	Weighted by Gaussian distribution	All pixels contribute equally, but uses median
Effect on Image	Uniform smoothing, blocky edges	Smooth, natural transitions	Removes noise, sharper edges
Edge Preservation	Poor	Better due to weighted averaging	Good: Reduces noise while maintaining edges

Feature	Normal Blur	Gaussian Blur	Median Blur
Control Over Blur	Limited	Fine control via σ	Limited: Kernel size only
Use Case	General-purpose blur	Sophisticated noise reduction, editing	Removing salt-and-pepper noise

---

What is Salt-and-Pepper Noise?

Salt-and-pepper noise looks like random black and white dots scattered across an image. It's caused by issues like sensor errors or transmission problems.

📜 Notes By: Naveen V

✍️ *"Turning thoughts into insights."*