Mr. Mina
12th Grade Computer Vision

Lesson 06 - Edge Detection

EQ: How can we detect edges with convolution?

Do Now

Ask and answer: Ask your partner a question you have about Gaussian kernels and answer your partner's question. If both of you can't answer a question, ask another pair! Feel free to ask about last night's homework!

```python
def create_gaussian(size, sigma):
    """
    Create a simple 2D square Gaussian distribution.

    Parameters:
    - size: Integer specifying the size of the output square array (height and width are the same).
    - sigma: Standard deviation of the Gaussian.

    Returns:
    - A 2D NumPy array representing the square Gaussian distribution.
    """
    # Create an empty kernel
    gaussian = np.zeros((size, size))

    # Calculate the center of the kernel.
    # The kernel is square so the center is the same for the row and col
    center = size // 2

    # The Gaussian is a function of the x,y or col, row. Each col,row combination
    # produces the Gaussian value at that point.
    for row in range(size):
        for col in range(size):
            # Squared distance from this point to the center.
            d = (row - center)**2 + (col - center)**2

            # Set up the exponent
            exponent = (- 1 / (2 * sigma**2)) * d

            # G(col, row) = e^(exponent)
            gaussian[row][col] = np.exp(exponent)

    # Normalize to make sure the sum is 1
    return gaussian / np.sum(gaussian)
```

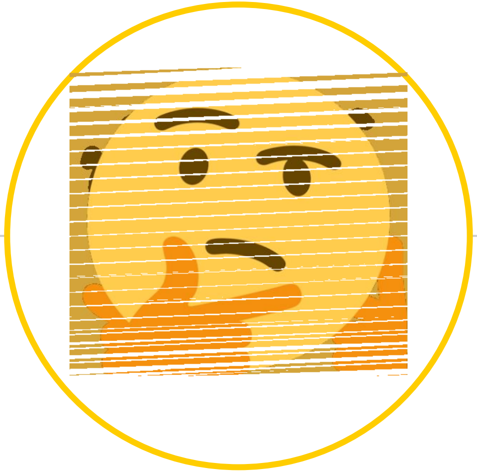
What is an edge?

- How would you describe an edge:
  - in a color image?
  - in a grayscale image?
  - using math?

vs.

What is an edge?

Zoom into
red region

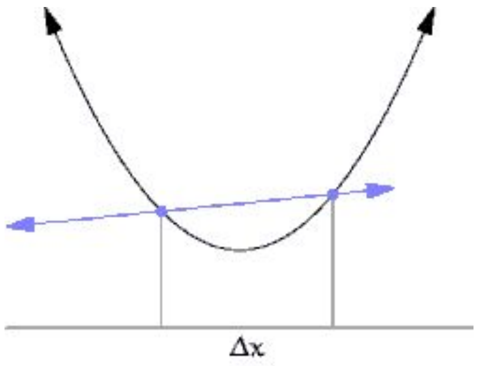


An edge is rapid change in pixel intensity over a small area.

How can we measure rapid change of
values over small areas?
(Think calculus...)

Derivative refresher

- The derivative measures the **instantaneous rate of change** of a function at point
  - A strong/weak slope means the function is changing **quickly**
  - A positive/negative slope indicates in which direction the function is changing



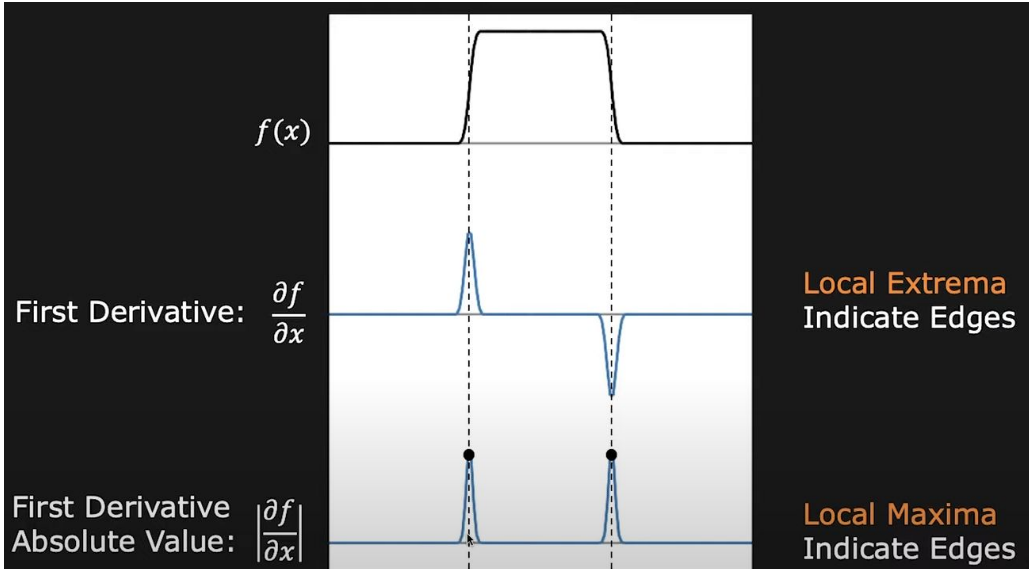
$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Derivative in signal processing

The derivative gives

1. The direction the edge is strongest (maxima vs. minima)
2. The strength of the edge (how intense the change is)

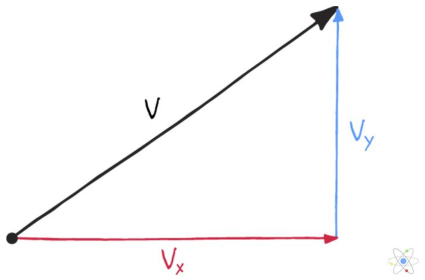
For now, let's take the absolute value to just get the strength.



Credit: Shree Nayar @ Columbia

Finding edges with derivatives

- In physics/geometry, we can break down vectors/lines into their x/y components.
- We can reverse this process to find all edges in a two-dimension context (i.e. an image).
  1. Use the derivative in the x-axis to find horizontal edges
  2. Use the derivative in the y-axis to find vertical edges
  3. Combine the results to get all edges



Roll for confidence!

Calculating the derivative in pixels

- **The limit definition of the derivative is a glorified average!!**
- Block blur kernel computes average in all directions, can we modify it to:
  - Calculate the average only along the x-axis? y-axis?
  - Consider the direction of inputs?
- We can compute the average with a convolution, but how do we make it specific to one direction??

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Block blur kernel →x derivative kernel

- We’ll transform the block blur kernel into a x-derivative kernel
- Let’s use whole numbers in the kernel
  - We’ll have to normalize later to prevent over-saturation. Ignore that for now

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

→

1	1	1
1	1	1
1	1	1

Block blur kernel →x derivative kernel

- We want to calculate the change in the surrounding pixels, not the center pixel. Let's remove (1, 1)
- We only want to measure across the x-axis. Let's remove (0, 1) and (0, 2) since they measure the y-axis (i.e. from top to bottom)

1	1	1
1	1	1
1	1	1

→

1	0	1
1	0	1
1	0	1

Block blur kernel →x derivative kernel

- We want to include the direction and flow in our convolution calculations. Let’s set the left side to negative and the right side positive (like a number line)

1	0	1
1	0	1
1	0	1

→

-1	0	1
-1	0	1
-1	0	1

Congrats!

We’ve recreated the Prewitt x derivative kernel!

Prewitt kernel

- Sometimes called Prewitt operator
- Created in 1970 to calculate the derivative
- Two kernels: one for each x/y derivative

Prewitt x–kernel

-1	0	1
-1	0	1
-1	0	1

Prewitt y–kernel

1	1	1
0	0	0
-1	-1	-1

Gaussian in edge detection

- Noisy images can lead to false positives, but we can smooth with Gaussian to make the derivative less sensitive
  - Rapid changes in brightness have been smoothed out

vs.