Mr. Mina
12th Grade Computer Vision
9-12.CT.2, 9-12.CT.3, 9-12.CT.5, 9-12.CT.6

Lesson 06 - Comparing Kernels

EQ: What kernels can/should we use to detect edges?

Do Now

Review L5, S11-13 on how we derived the Prewitt edge detector from the block blur kernel. Using a similar train of thought, can you “invent” any new **edge detector** kernels?

Remember, we want to calculate the change *surrounding* the target pixel, not including it.

(edge detected!!!)

Pros vs. Cons: Prewitt kernel

Pros

- Good localization (doesn’t let far pixels influence)

Cons

- Diagonal pixels are just as important as adjacent pixels
- Very sensitive to noise since
  - weights are equal
  - small kernel size
- Usually have to smooth image before to reduce impact of noise

Prewitt x kernel

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

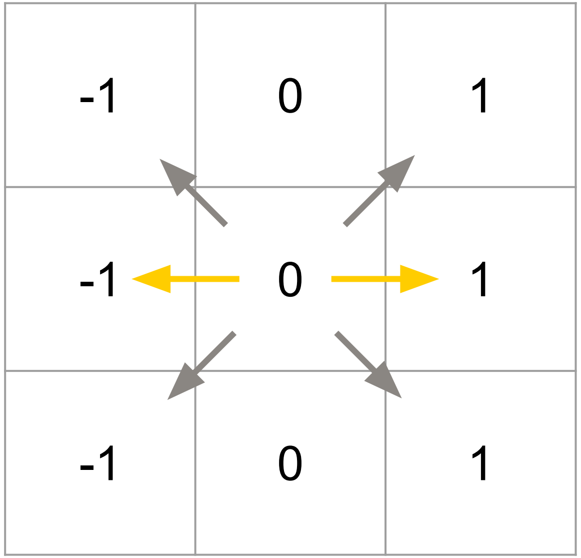
Prewitt y kernel

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

Prewitt x kernel

- All surrounding pixels are weighed evenly.

What would happen if we gave adjacent pixels more weight since they're closer than the diagonals?



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

Sobel kernel

- Usually **n=2**, but n=3–5 also yields decent results
- Puts extra weight for adjacent pixels
- Most common kernel for simple edge detection

Sobel x kernel

-1	0	1
-n	0	n
-1	0	1

Sobel y kernel

1	n	1
0	0	0
-1	-n	-1

Another Sobel derivation!

With the Prewitt operator, we have to smooth the image with a Gaussian first to reduce noise. What if we applied a Gaussian to our Prewitt operator instead?

With your partner, convolve our x derivative Prewitt with a vertical Gaussian to get a 3x3 kernel. Compute all points of intersection!

Prewitt x derivative

$$\begin{bmatrix}-1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} \text{Gaussian} = \;??$$

Prewitt ∗ Gaussian = Sobel

$$\underset{\text{Prewitt x derivative}}{\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}} * \overset{\text{Gaussian}}{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}} = \overset{\text{All intersections}}{\begin{bmatrix} -1*1 & 0*1 & 1*1 \\ -1*2 & 0*2 & 1*2 \\ -1*1 & 0*1 & 1*1 \end{bmatrix}} = \overset{\text{Sobel! 🎉}}{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}}$$

Pros vs. Cons: Sobel kernel

Pros

- Includes Gaussian smoothing!
- Good localization (doesn’t let far pixels influence)
- Adds more weight to adjacent pixels

Cons

- Hard to find diagonal edges since the weights are small
- Sensitive to noise since kernel is small

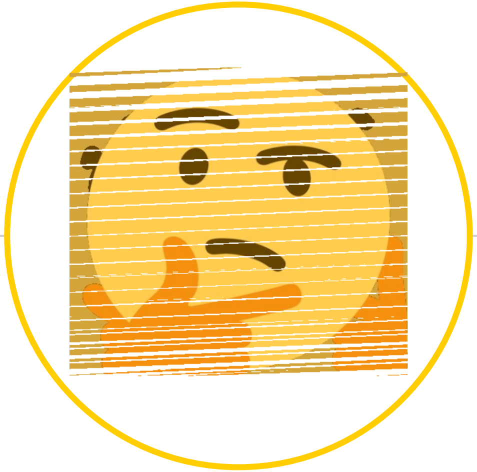
Sobel x kernel

-1	0	1
-2	0	2
-1	0	1

Sobel y kernel

1	2	1
0	0	0
-1	-2	-1

Roll for confidence!

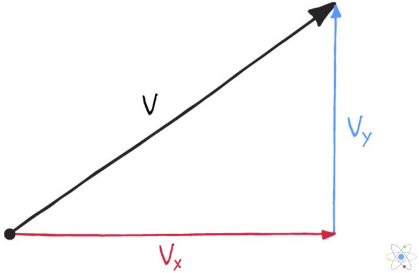


We’ve looked at separate kernels for each direction. How do we combine the results?

Combining directional edge detection

There are two ways to combine results:

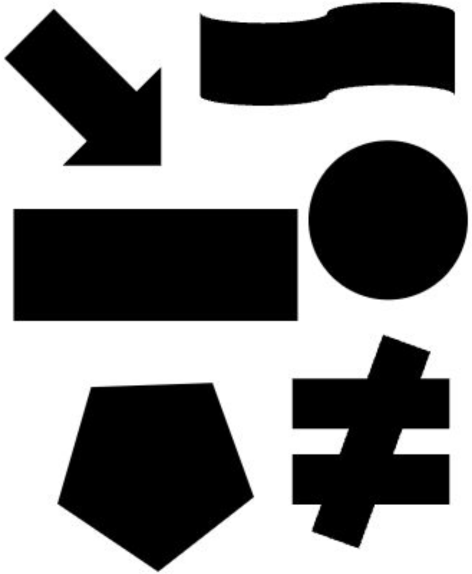
- Average the values from the x/y derivatives to create the merged result
  - Generally fast
  - Inaccurate results
- Use the Pythagorean Theorem to find the magnitude of the split x and y components
  - Slow operation for large images
  - Highly accurate than average



Edge detection examples

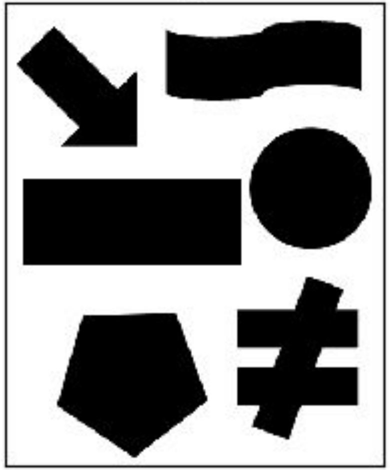
Let’s use the Prewitt and Sobel kernels to find the edges in the x and y directions of these shapes.

For now, we’re only looking for the strength of the edge so we’ll take the absolute value.



Prewitt edge detection

1. First, let’s use Prewitt x kernels to find the edges in the x direction.



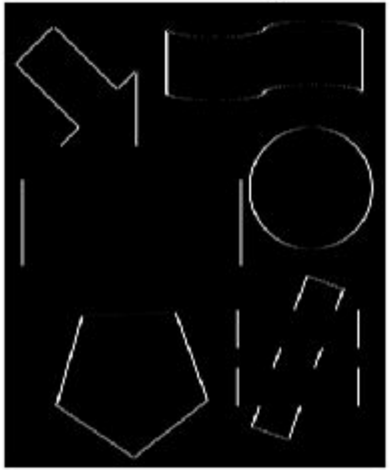
Original

*

Prewitt x kernel

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

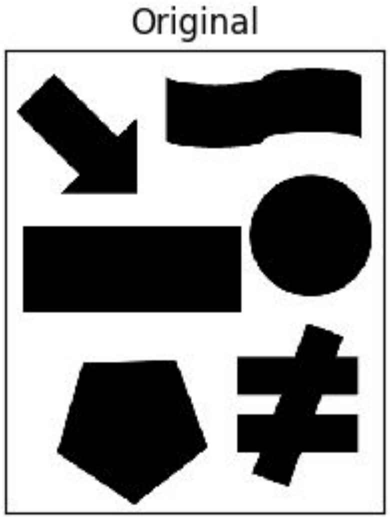
=



Prewitt x edges

Prewitt edge detection

2. Do the same for the y edges.



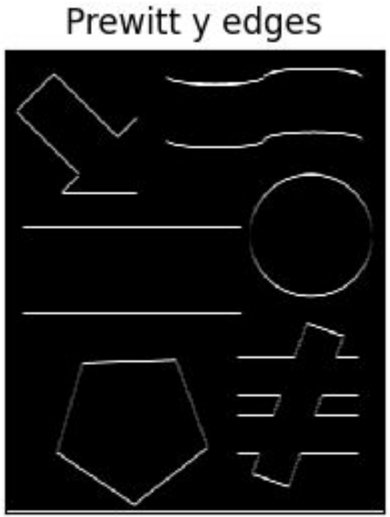
Original

*

Prewitt y kernel

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

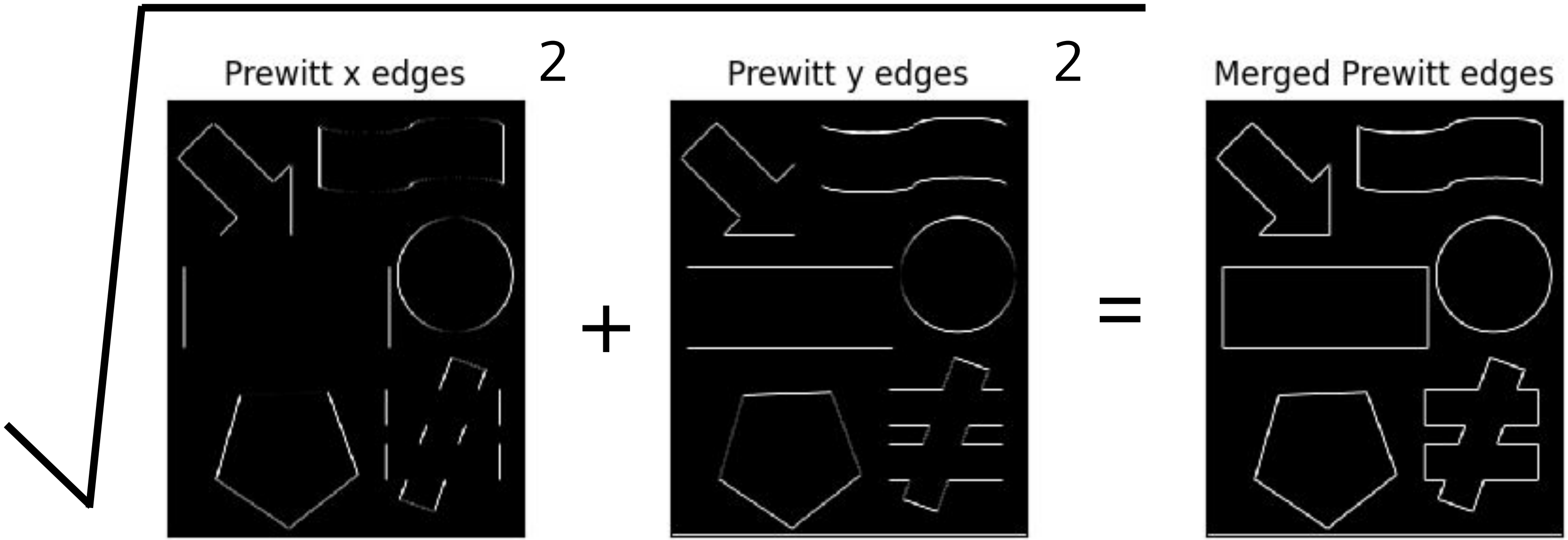
=



Prewitt y edges

Prewitt edge detection

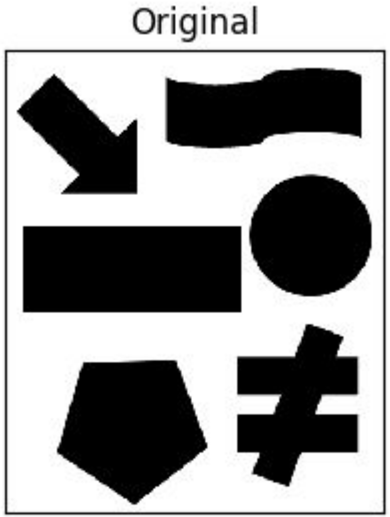
3. Merge the results using the Pythagorean Theorem.



Roll for confidence!

Sobel edge detection

1. First, find the x edges

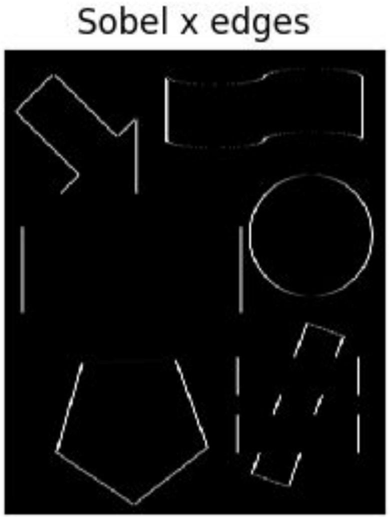

Original

*

Sobel x kernel

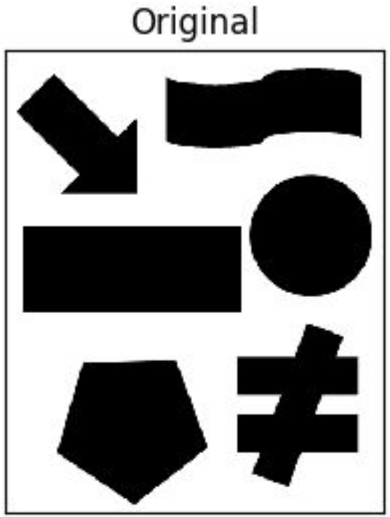
-1	0	1
-2	0	2
-1	0	1

=


Sobel x edges

Sobel edge detection

2. Do the same for the y edges.

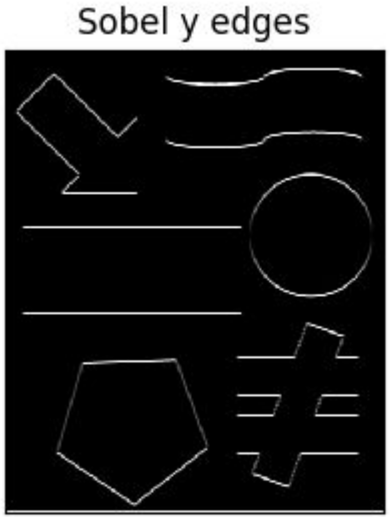


*

Sobel y kernel

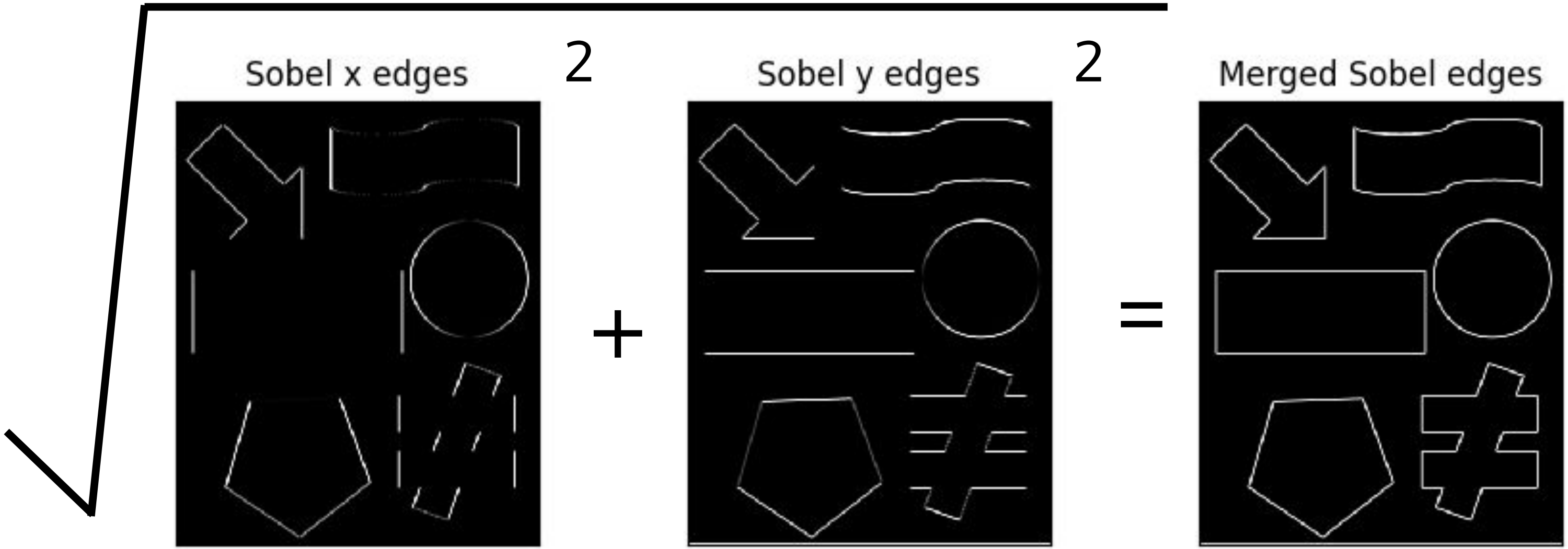
1	1	1
0	0	0
-1	-1	-1

=



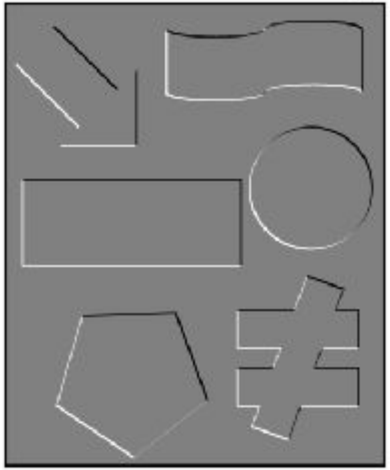
Sobel edge detection

3. Merge the results using the Pythagorean Theorem.

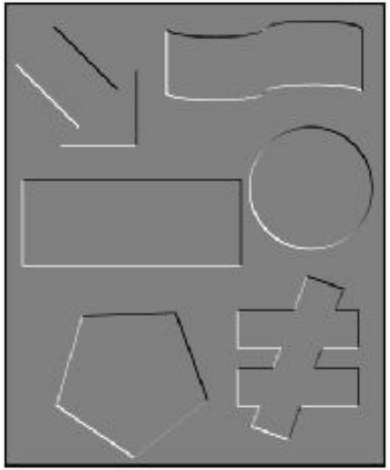


Direction of edges

- **White** = strong edge in the positive direction of the kernel
- **Gray** = either no edge or equal in both directions
- **Black** = strong edge in the negative direction of the kernel



Direction of Prewitt edges



Direction of Sobel edges