Instructor: Andy Mina
Grade Level and Subject: 12th Grade - Introduction to Computer Vision
Topic: Introduction to Convolutions
Lesson: 06_edge_detection

NYS Computer Science and Digital Fluency Learning Standards	9-12.CT.2 - Collect and evaluate data from multiple sources for use in a computational artifact. 9-12.CT.3 - Refine and visualize complex data sets to tell different stories with the same data set 9-12.CT.5 - Modify a function or procedure in a program to perform its computation in a different way over the same inputs, while preserving the result of the overall program.
Content Objective	Students will be able to: ● Understand how we use derivatives to detect edges ● Construct a kernel that calculates the derivative in the x/y direction
Scaffolding Needed	Students should be able to: ● Understand what the derivative of a function represents ● Comfortably use the Pythagorean Theorem ● Use the block blur operator to take the average value of pixels surrounding a target pixel
Key Vocabulary	**Edge**: a rapid change in pixel intensity over a small area **Derivative**: a function that measures the rate of change of another function
Assessments	Roll for Confidence (Formative) Students will be asked to "roll for confidence" and respond by showing the instructor a number from 1 to 5 on one of their hands. Their confidence is representative of how comfortable they feel in continuing to explore and compare other sorting algorithms on their own. Scores represent the following: 1. **Not confident.** Needs a re-explanation or summary of the lesson with emphasis on key points. 2. **Pretty shaky.** Needs a brief recap and some teacher-guided practice to solidify concepts and understanding. 3. **Okay.** Needs some peer-guided practice and some more time to let things sink in. Ideal rating after

	the lesson. 4. **Pretty confident.** Needs some peer-guided practice for more challenging algorithms, but is self-sufficient for what's covered in class. Ideal rating before a unit test. 5. **Extremely confident.** Needs little to no guidance and can tackle problems of exceptional difficulty with relative ease. Indicative of an under-challenged student. These checks shouldn't take any longer than one minute.
Materials	06_edge_detection slides

Lesson Component	**Description or Execution of Lesson Component (w/ scripting when appropriate)**
Essential Question	How can we detect edges with convolution?
Do Now	**S1, 4m** Briefly let students ask and answer. Afterward, go over any unanswered questions as a class.
Presentation of Content	**S2, 3m** "Here's the solution to the homework from last night. Remember that the Gaussian kernel is a two-dimensional function. This means the current row/col we're on directly corresponds to the weight of the pixel at that point. You'll notice that the normalization factor from the original formula (1/sigma * sqrt(2 *pi)) isn't anywhere in this code. As we talked about in class, the important part is that the values add up to 1. However you do it is up to you. In this code, we're normalizing by dividing each element by the sum of all elements, i.e. creating a percentage." **S3-5, 3m** Go through slides with students and put special emphasis on the definition of an edge. Prompt students if they know how to measure the rate of change in a function.

	S6, 4m Give a brief introduction/refresher on derivatives to students according to the slides. “The derivative is a function that measures the rate of change of another function. Here, the derivative is a tool for us to measure the rate of change of pixel brightness/intensity. We’re not going to go too deep into the math but take a look at the limit definition of the derivative. Given that we’re taking the difference across h values of f(x) and then dividing it by h, this should remind you of something. We’ll talk more about that in a couple of slides.” **S7, 4m** Read over the slide for students. “Look at the graphs on the right. Imagine the function f(x) is a function representing the intensity of pixel X across a line. As we scan the pixel brightness from left to right, when we find an edge the intensity will go up, remain at the top for a little while we’re inside the edge, and then go back down when we exit the edge. The second graph represents the derivative of the first graph. When we cross into the edge on the first graph, there’s a lot of change going which results in a sharp peak in the derivative. When we cross out of the edge, the derivative has a sharp drop/trough. For now, we’re only concerned with the strength of the edge, so we’ll take the absolute value which is what the third graph displays. However, in the future, there may be a situation where want to find the direction of the edge (strong/weak positive/negative edge) so we won’t take the absolute value. As you might’ve already asked yourself, ‘How do I apply the derivative in two dimensions?’” **S8, 3m** Go through the slides with students. “To get a two-dimensional derivative to measure the rate of change, we’ll measure the x derivative and y derivative and then combine to get the two-dimensional result. Tomorrow we’ll apply this process, but today

	we're focusing on the theory behind edge detection" **S9, 3m** Stop and answer questions as needed. Roll for confidence in: ● Understanding how the derivative helps us detect edge ● How to find the derivative in two-dimensions **S10, 2m** Explain to students that the limit definition of the derivative is essentially an average of h number of values of f(x). We already know how to take the average with block blur kernel, we just need to modify it to take the average in one direction at a time. **S11-14, 6m** Don't stop for questions, but take this slowly. Work with students to discover how to modify the block blur kernel to find the x-derivative. Note to students that to emphasize the difference between pixels and measure the rate of change we need negative weights in some places. Also note that we want to find the rate of change surrounding a specific pixel, but not including that pixel. We're removing pixels by just setting the weight to 0. **S15, 6m** Introduce students to the Prewitt kernels which calculate the derivative in the x and y directions. "We'll work with this Prewitt kernel and another edge detection kernel tomorrow. For now, make sure to understand why we're taking the derivative and how the kernel weights are letting us do that." Go back and review the recreation of the Prewitt kernel if necessary. **S16, 3m** "Since the derivative is measuring the rate of change, noise is especially distracting in edge detection. Noise adds a bunch of mini-changes everywhere so it's hard to differentiate between what's a real edge and what's an edge only because of noise. As we spoke about yesterday, a Gaussian kernel can save us here."

	Go through the slide with students.
Homework	n/a