University of Michigan
EECS 442: Computer Vision
Fall 2025 Instructors: Jun Gao and Bernadette Bucher

Problem Set 1: Image filtering

Posted: Monday, August 25, 2025 **Due:** Wednesday, September 10, 2025

Download the **starter code** for the assignment **here**.
Final assignment submission is done through **Canvas.**
Problem 1.1: submit written solution as a .pdf file.
Problem 1.2: submit notebook solution as a .ipynb AND .pdf file.

Problem 1.0 `numpy` *review (optional)*

This Colab notebook contains a brief **introduction to `numpy`**. If you are new to `numpy` and numerical computing, we encourage you to work through these examples. They should cover the background that you need to complete this problem set.

Problem 1.1 *Properties of convolution*

Recall that 1D convolution between two signals $f, g \in \mathbb{R}^N$ is defined:

$$(f * g)[n] = \sum_{k=0}^{N-1} f[n-k]g[k]. \tag{1}$$

(a) Construct a matrix multiplication that produces the same result as a 1D convolution with a given filter. In other words, given a filter f, construct a matrix H such that $f * g = Hg$ for any input g. Here Hg denotes matrix multiplication between the matrix H and vector g.
(2 points)

Note: One option is to define H in "bracket" notation using "..." symbols, e.g., $H = \begin{bmatrix} 0 & 1 & \dots & N-1 \\ N-1 & N-2 & \dots & 1 \end{bmatrix}$. Another option is to explicitly define the entries H_{ij}.

(b) In class, we showed that convolution with a 2D Gaussian filter can be performed efficiently as a sequence of convolutions with 1D Gaussian filters. This idea also works with other kinds of filters. We say that a 2D filter $F \in \mathbb{R}^{N \times N}$ is separable if $F = uv^\top$ for some $u, v \in \mathbb{R}^N$, i.e., F is the *outer product* of u and v. Show that if F is separable, then the 2D convolution $G * F$ can be computed as a sequence of two one-dimensional convolutions **(2 points)**.

(c) (*optional*) Show that convolution is commutative, i.e., $f * g = g * f$. Assume circular padding (e.g., $f[-1] = f[N-1]$), zero padding, or whatever is convenient. *Note: we will not grade this problem, and you will* ***not*** *get bonus points for completing it.*

Hint: Write the equation for $f * g$, and figure out how to "rename" the variable used in the summation to arrive at $g * f$.

(d) (*optional*) Show that convolution is associative, i.e. $(f * g) * h = f * (g * h)$.

Hint: Start by writing down the expression for $G * F$ and then, inside the double summation, write F in terms of u and v.

(e) (*optional*) Show that cross-correlation,

$$h[n] = \sum_{k=0}^{N-1} f[n+k]g[k], \tag{2}$$

is *not* commutative.

Problem 1.2 *Sponsored problem: pet edge detection*

As you know, this course is largely funded by grants from Petsco, Inc.™ Unfortunately, one of the strings attached to this funding is that we must occasionally assign *sponsored problems* that cover topics with significant business implications for our sponsor[1]. Our partners at Petsco see an opportunity to use computer vision to pull ahead of their bitter rivals, Petsmart™. To avoid painstakingly marking the edges of dogs and cats in pictures by hand, the current industry best practice, they have turned to us for an automated solution.

Figure 1: The Petco / EECS 442 partnership, illustrated with DALL-E 2.

(a) Apply the horizontal and vertical gradient filters [1 -1] and $[1 \; -1]^\top$ to the picture of the provided pet[2] producing filter responses I_x and I_y. Write a function `convolve(im, h)` that

[1] While these problems, sadly, are not very educational, we are thankful for Petsco's sponsorship. Recall that in previous semesters, the whole class shared a single GPU, which the course staff shuttled between their homes in a rented van.

[2] These convolution filters are 1×2 and 2×1 matrices respectively. Even though these filters are not square, your convolution function should have no problem using them, since it should be able to handle filters of any dimension.

(a) An input image

(b) "Fluffy coat" failure case

Figure 3: (a) A pet photo helpfully delivered by our sponsor. (b) A failure case for a simple edge detector. These images are provided in the starter code. Photo credits can be found here.

takes a grayscale image and a 2D filter as input, and returns the result of the convolution. Please do not use any "black box" filtering functions for this, such as the ones in `scipy`[3]. You may use `numpy.dot`, but it is not necessary. Instead, implement the convolution as a series of nested `for` loops. Compute the *edge strength* as $I_x^2 + I_y^2 f$. After that, create visualizations of I_x, I_y, and the edge strength, following the sample code.

The filter response that your function returns should be the same dimensions as the input image. Please use *zero padding*, i.e. assume that out-of-bounds pixels in the image are zero. Also please make sure to implement convolution, *not* cross-correlation. Note that this simple filtering method will have a fairly high error rate — there will be true object boundaries it misses and spurious edges that it erroneously detects. The team at Petsco, thankfully, has volunteered to fix these errors by hand **(4 points)**.

Figure 2: Photo from the meeting between the provost and a Petsco representative that resulted in this generous funding relationship. Made using DALL-E 2.

(b) ***(Optional)*** This method detects edges fairly well on one of the dogs, but not the other. Our sponsor would like us to investigate why this is happening. First, convert your gradient outputs to edge detections using a hand-chosen threshold τ (i.e., set strength values at most τ to 0 and those above to 1). Point out 2 errors in the resulting edge map — that is, edge detections that do not correspond to the boundary of an object — and explain what causes these errors.

(c) While the edge detector you submitted works well on some pets, engineers are reporting a large number of failures, and the Petsco execs are *not* happy. Worrisomely, it has been failing on dogs with fluffy coats, such as "doodle" mixes — a lucrative market for our sponsor (Figure 3b). The source of this error, Petsco's team has concluded, is that the gradient filter is often firing on small, spurious edges in highly textured regions.

[3]Our sponsor has been prohibited from from using these functions as part of the terms of an lawsuit that has been rumored to involve a dog bite and a key member of the `scipy` development team.

Please kindly address this problem by creating an edge detector that only responds to edges at larger spatial scales. To do this, blur the image with a Gaussian filter prior to computing gradients. Implement your Gaussian filter on your own[4]. Please do not use any "black box" Gaussian filtering functions for this, such as `scipy.ndimage.gaussian_filter`. Apply both the Gaussian filter and the gradient filter using a black box convolution function `scipy.ndimage.convolve` on the image, rather than your hand-crafted solution.

(i) Compute the edges without blurring, so we can look at the before-and-after results.

(ii) Compute the blurred image using $\sigma = 2$ and an 11×11 Gaussian filter.

(iii) Instead of blurring the image with a Gaussian filter, use a box filter (i.e., set each of the 11×11 filter values to $1/11^2$).

(iv) Compute edges on the two blurred images.

(v) ***(Optional)*** Do you see artifacts in the box-filtered result? Describe how the two results differ. Include your written response in the notebook.

Visualize the edges in the same manner as Problem 1.2(a). Your notebook should contain the edges that were computed: (i) using no blurring, (ii) using Gaussian blurring, and (iii) box filtering. Please also show the (iv) Gaussian blurred image, (v) the box-filtered image **(2 points)**.

(d) Instead of convolving the image with two filters to compute I_x (i.e., a Gaussian blur filter followed by a gradient filter), create a *single* filter Gx that yields the same response. You can reuse the function in part (c). Visualize this filter using the provided code **(2 points)**.

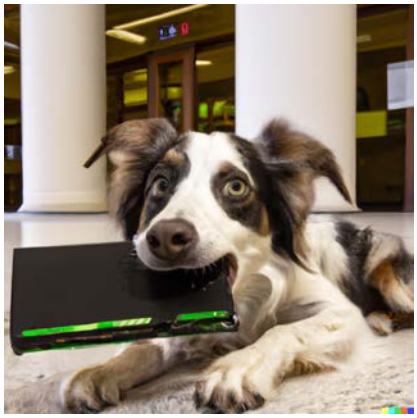

Figure 4: A member of our dog-based GPU delivery fleet, made using DALL-E 2.

(e) Good news. Petsco execs are pleased with your work and have renewed our funding for several more problem sets. At our last meeting, however, they made an additional request. Their competitors at Petsmart are now computing *oriented* edges of their pets. Rather than estimating the only horizontal or the vertical gradients, they now can provide gradients for an arbitrary angle, θ. Petsmart's method for doing this, however, is computationally expensive: they construct a new filter for each angle, and filter the image with it. Petsco sees an opportunity to pull ahead of their rivals, using the class's knowledge of steerable filters.

Write a function `oriented_grad`(I_x, I_y, θ) that returns the gradient steered in the direction θ, given the horizontal and vertical gradients I_x and I_y. Use this function to compute your gradients on a blurred version of the input image at $\theta \in \{\frac{1}{4}\pi, \frac{1}{2}\pi, \frac{3}{4}\pi\}$, using the same Gaussian blur kernel as (c). Visualize these results in the same manner as the gradients in Problem 1.2 (a) **(2 points)**.

(f) Petsco would also like to offer the ability to enhance low quality pet photos. Given a noisy image of a pet, please remove the noise by applying the box and Gaussian filters that you have

[4]Hint: Make sure that the kernel is properly centered.

developed in (c). Then, implement a function `median(im, k)` that performs *median filtering* using a $k \times k$ window. More specifically, for each pixel of the noisy image, you will compute the median intensity value of the other pixels within a $k \times k$ window. Your code should be very similar to your code for `convolve`, but instead of computing a weighted average in the inner loop, you will compute the median. You may use a built-in function for computing the median of a list or array, such as `np.median`. Include the results for $k = 3$ and $k = 7$.

Your solution should display the following 5 images: (i) the noisy image itself, (ii) the box filtered noisy image, (iii) the Gaussian filtered noisy image, (iv) median filtered noisy image with $k = 3$, (v) median filtered image with $k = 7$. **(3 points)**