# Project 3: Tracking Image Movement using Convolution

**Due on November 16th, 2020 at 23:59**

## Introduction

Computer vision is a cross-disciplinary field at the intersection of electrical engineering, computer science and machine learning. A few applications of computer vision are self-driving vehicles, where it is used to identify and avoid obstacles, automation, where it allows robots to interact with their environment autonomously or medicine where it has been used to measure heart rates and even identify heart defects.

In this project you will see how many of insights of image processing were derived from signal processing. In particular, the one-dimensional 'time' signal that are the primary interest of this course can be easily generalized to higher dimensions. In this way, an image can be viewed as a two-dimensional signal and many of the techniques from this course can work on images as well.

In particular, the operation of convolution, and the related operation of *cross correlation*, turn out to be particularly important in the computer vision field. In fact, many common image operations such as blurring, sharpening and edge detection can be implemented using just convolution or cross correlation with a particular 'convolutional kernel'.

In the programming component of this assignment, you will use cross correlation to characterize the direction of movement in a video sweeping over a landscape. This is similar to the techniques used by by cartographers to stich together the video feed of a landscape into a single map. Although it is easy for humans to discern the movement of a video, it is surprisingly difficult to create a heuristic for computers to do the same task. However, if the frames of the video are convolved with an intelligently chosen kernel, you will see it becomes much easier to create such a heuristic.

## Task 1 (3 marks): Interpreting Averaging as a Convolution

Consider the system $S : [\mathbb{Z} \to \mathbb{R}] \to [\mathbb{N}_0 \to \mathbb{R}]$ whose output is the average of its last $M$ inputs:

$$y = S(x) \implies \forall n \in \mathbb{N}_0,\ y(n) = \frac{1}{M} \sum_{k=0}^{M-1} x(n-k).$$

1. Find the signal $h \in [\mathbb{Z} \to \mathbb{R}]$ such that

   $$y = S(x) \implies y(n) = (h * x)(n) \text{ for all } n \geq 0,$$

   where the symbol $*$ denotes convolution.

2. Let $y$ be defined as above. For what values of $n \geq 0$ is $y(n)$ independent of $x(k)$ for all $k < 0$?

## Task 2 (5 marks): Cross Correlation

Another operation that acts on two signals to produce a new signal is cross correlation. The cross correlation of two signals is defined by the relationship

$$y = g \star x \implies \forall n \in \mathbb{Z},\ y(n) = \sum_{k=-\infty}^{\infty} g(k-n) \cdot x(k).$$

This operator is very similar to convolution but has a more straightforward interpretation. Notice that if $y = g \star x$, then $y(0)$ is simply the sum of the product of the values of $g$ and $x$ at each index. This is analogous to the dot product of two vectors. Similarly to the dot product, this value is a measure of the closeness of the two signals $g$ and $x$. For values of $n$ that are larger than zero, $y(n)$ is simply the dot product of the signal $x$ with a copy of $g$ that is shifted by $n$ units in time. Therefore, $y(n)$ is simply of measure of how similar $x$ is to a $n$-shifted copy of $g$.

3. Suppose $g \star x = h * x$. Based on the above definition for the cross correlation, find the relationship between $g(n)$ and $h(n)$ for all $n \in \mathbb{Z}$.
4. Recall the system $S$ from Question 1. What is the signal $g \in [\mathbb{Z} \to \mathbb{R}]$ such that $S(x) = y \implies y = g \star x$?

You should notice that in both Question 1 and Question 3, the signal that is either convolved or cross correlated with the input to create the averaging system is of finite support (i.e., the number of non-zero elements is finite). Therefore, it is possible to store this signal on a device with finite physical memory like a computer. In the context of cross correlation, this signal would be called a *convolutional kernel.* By convention, convolutional kernels are usually the left argument to a cross correlation operation. This may be confusing since the term convolutional kernel is more commonly associated with cross correlation than convolution however it is common machine learning and digital signal processing lingo.

5. Suppose $x \in [\mathbb{Z} \to \mathbb{R}]$ is an aperiodic signal. Let $y = x \star D_d(x)$, where

   $$D_d : [\mathbb{Z} \to \mathbb{R}] \to [\mathbb{Z} \to \mathbb{R}], \quad z = D_d(x) \implies \forall n \in \mathbb{Z},\ z(n) = x(n-d),$$

   i.e., $D_d$ is a delay system by $d$ units in time. Determine for what value $n$ is $y(n)$ greatest? Show your steps. (Hint: Consider the Cauchy–Schwartz inequality.)

## Task 3 (3 marks): Cross Correlation of Images

In the programming task in the next section, you will use cross correlation of images to perform a classification task. Therefore we must now generalize the definition and intuition of cross correction developed above such that the definition will work with images. An image can be viewed as a two-dimensional signal and luckily it is straightforward to make the jump to operating on two-dimensional signals using cross correlation.

Just as single dimensional cross correlation takes in two single dimensional signals and returns a single dimensional signal, the two-dimensional generalization of cross correlation should operate on a pair of two-dimensional signals and return another two-dimensional signal. Furthermore, just as the single dimensional cross correlation measures the similarity of two signals when one is shifted in time by $n$ time steps, the two-dimensional output of the two-dimensional cross correlation should measure how similar the first input is to the second input after the first input has been shifted both vertically and horizontally by $m$ and $n$ units respectively.

This suggests the following definition

$$y = g \star x \implies \forall m, n \in \mathbb{Z},\ y(m,n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} g(i-m, j-n) \cdot x(i,j). \tag{1}$$

Notice that the sums in the above equation are infinite, even though images do not usually have infinite extent as it is often assumed for signals. The most common way to deal with this issue is to assume some values for the "missing values" outside the image. Some common options are treating the signal representation of the image as periodic outside of the defined pixel values or assuming all values in the signal outside of the defined pixel values are equal to zero.

For the latter case, considering a grey-scale image signal as

$$x_{\text{img}} \in [\{0, 1, \ldots, d_{\text{vert}} - 1\} \times \{0, 1, \ldots, d_{\text{hori}} - 1\} \to [0, 1]]$$

the signal $x$ used in the correlation (1) would be obtained as

$$x(i, j) = \begin{cases} x_{\text{img}}(i, j), & 0 \leq i < d_{\text{vert}}, 0 \leq j < d_{\text{hori}} \\ 0, & \text{otherwise} \end{cases} . \tag{2}$$

In this assignment we will use a different approach. We will only store the results of the cross correlation that are independent of any values outside of the defined range of the images. The indices of the output that are considered will depend on the sizes of both two-dimensional input images and you should expect the output to have a smaller (or equal) size compared to the larger of the two inputs.

6. Suppose $x_{\text{img}} \in [\{0, 1, \ldots, 99\}^2 \to [0, 1]]$ is the signal representation of an image that is 100 pixels wide and 100 pixels tall. Let $y$ be the cross correlation of $x$ obtained from $x_{\text{img}}$ as in (2) with a convolutional kernel $g$ where $g(i, j) = 0$ if $i < 0$, $i \geq 3$, $j < 0$ or $j \geq 3$. For what values of $m$ and $n$ is $y(m, n)$ independent of $x(i, j)$ for all $i < 0$ or $i \geq 100$ and $j < 0$ or $j \geq 100$?

   We chose to pad $x$ with zeros for indices outside the domain of $x_{\text{img}}$ but for these values of $m$ and $n$ we could have put anything there and got the same result for $y(m, n)$.

**Submit your written response for these written questions via PDF on Canvas.**

## Task 4 (8 marks): Programming Task: Finding the Direction of a Video

This portion of the project is available via Github Classroom at 'https://classroom.github.com/a/gAacwET-'. Use Git to clone the project directory from the Github repository created by Github Classroom. To complete the assignment, implement the functions within the given function skeletons and use Git commands to push your changes to Github.

You must use the Python libraries NumPy and Imageio to complete this project while you may also use the SciPy library. To install these dependencies you can run `python -m pip install numpy scipy imageio` in your Git Bash terminal.

**Only your last commit pushed to Github before the deadline will be marked.**

Imagine that you have a video that sweeps across a landscape in a particular path, for example the video feed coming in from a flying drone. In this project, your goal will be to decipher the path of movement in such a video, therefore recovering the path that the drone originally followed.

You will be provided two videos, the video feed from a drone travelling a triangular path and the video feed from a drone travelling a square path. In order to decipher these two paths, you will use cross-correlation to compare consecutive frames in the incoming video feeds.

The first function you must implement is the `correlate_adjacent_frames` function whose skeleton is given in the `implementation.py` file within the project repository. This function takes in a pair of two-dimensional Numpy arrays with dimension 160 by 160. Each element of this two-dimensional array is an integer value between 0 and 255.

Your function should first perform the following image clean-up tasks:

1. Generate two new arrays (one for each input) from the two input arrays with floating point valued elements between 0 and 1, i.e., 0 should map to 0 and 255 should map to 1.
2. Calculate the average of all elements in both input arrays and subtract this value from each element of both input arrays.

Your function should then extract a sub-array from the second (transformed) input of size 110 by 110 by discarding the first and last 25 elements of the array in each dimension. Next your function should use the array of size 110 by 110 as a convolutional kernel and calculate the cross correlation with the first (transformed) input. You should use the method described in the last paragraph of Task 3 to decide the bounds of this cross correlation operation (hint: the range of cross correlation output pixels that do not depend on any values of the input outside the given range should be smaller than *both* inputs). The Scipy Python library will be available for you to use if needed.

Lastly, your function should transform the elements of the output of the cross correlation to the range of integers between 0 and 255 and return the array. The smallest (most negative) value in your array should be mapped to 0 while the largest is mapped to 255, all other values should interpolate linearly between 0 and 255. To get an integer valued output you should first interpolate and then you may use the `uint8` function within the Numpy package to remove the decimal part of the result.

The second function you must implement is the `make_correlation_video` function whose skeleton is also given in the `implementation.py` file. This function takes in an input filename and an output file name, and returns nothing.

Your function should use the `mimread` function from the imageio Python library to load a .gif video from the input filename as a Numpy array (this is represented as an 160 by 160 by $n$ array where $n$ is the number of frames). It should then apply `correlate_adjacent_frames` to each pair of adjacent frames in the resulting array to create an $n-1$ frame output. Finally your function should write this array as a .gif to the output filename using the `mimwrite` function from the imageio Python library. You can use this function to view the output of the cross correlation operations we have asked you to perform and test your intuition about what the output should look like.

The final function you must implement is the `is_triangular_path` function which takes in a single argument, an input filename for a .gif video. Your function should read the .gif into a Numpy array and classify the path traced by the video as triangular or square. You can use any heuristic for this that you wish but using the cross correlation video from above will make your life much easier. The function should return `True` if the path is triangular and `False` otherwise. Your project will be evaluated on whether it can classify square paths from triangular paths in both the videos provided and a testing set of six more videos.