Theoretical understanding

Convolution

Mathematical operation that consists of a weighted sum of the intensities of pixels in the surrounding area of the original image . The weights are stored in a filter kernel , which has entries , where and being the height and the width (on pixels) of the image. This is convolving with the filter , where two-dimensional convolution is defined as where is the new value for the pixel at our output image.

It is important to notice that is flipped in both directions, this is done to meet the commutative mathematical property of convolution, ensuring that the order of functions does not affect the outcome. This is needed due to the kernel being moved over the image, aligning with a specific set of image pixels at each position. This led us to the need of ensuring that the positions of the elements in the kernel match the corresponding positions in the image; the flipping compensates for the shifting effect and aligns the kernel’s elements properly with the image’s pixel values. It will also allow us to apply efficient computational methods, such as Fast Fourier Transform.

When applying the kernel without further modification we will run into problems at the borders of the images since we would be trying to access points that are outside the image. To deal with this we could apply different types of padding, including rows above and below and columns at both sides of our images with standard values allowing us to increase to fully process the image. Padding could be achieved using one of the most common padding techniques:

* Zero padding: including or assuming the values of is 0 outside the defined image region.
* Constant padding: including or assuming the values of is outside the defined image region, where is a constant.
* Reflective padding: mirroring the pixels along the borders of the image, replicating them in a mirrored fashion.

Finally, for achieving a clean computational implementation of convolution, we usually consider odd-sized kernels to have a center where we can centralize our operation. However, it is possible to apply convolution with even-sized or not-squared kernels.

Linear Filtering

Linear filtering is the result of applying different kernels to an image using convolution operation. Doing linear filtering we want to remove unwanted sources of variation and keep the information relevant for whatever task we need to solve, therefore, the kernel used will entirely depend on the expected result we want to obtain from the process. Some of the most used kernels in linear filtering are:

* Blur
  + Average box: Low-pass filter that smooths the image by making each output pixel the average of the surrounding ones, removing details, noise and edges from images.
  + Gaussian: By convolving an image with a 2D Gaussian defined as each pixel in the resulting image is a weighted sum of the surrounding pixels, where the weights depend on the Gaussian profile: nearer pixels contribute relatively more to the final output. This process blurs the image, where the degree of blurring is dependent on the standard deviation of the Gaussian filter.
* Edges
  + Laplacian: discrete two-dimensional approximation to the Laplacian operator given by . This will result in a response of high magnitude where the image is changing, regardless of the direction of that change: the response is zero in regions that are flat and significant where edges occur in the image. It is hence invariant to constant additive changes in luminance and useful for identifying interesting regions of the image.
  + Prewitt: Searching for places in the image where the intensity changes abruptly we try taking the first derivative of the image along the rows or the columns. This will give us a flat result when there are no changes across a direction in our image, negative when the image values are increasing and positive if they are decreasing. The kernels for each direction (horizontal and vertical) will have the shape and
  + Sobel: The same intuition than Prewitt filters case, Sobel filters use the first derivative identifying abrupt changes on the image intensities but increasing more (when compared to Prewitt’s) the final intensity of the edge, leading to shapes and horizontal and vertical respectively.

It is important to remark that any of these kernels could be increased in size following specific relations between the values contained on it; however, it will not be always a useful approach in some of those, especially the edges detection ones.

Template Matching

Different from convolution, cross-correlation do not flip the kernel, apart from that, the process is the same. Then, we use Normalized Cross-Correlation (NCC) to find in an image the given template by searching for the maximum values (representing the best matches of the template across the image) given by the NCC formula. The NCC formula recalls on the existing difference between and the current region of the , this is , since the squared difference give us two constant terms, we don’t care about them only considering the variable one resulting as . Since we are searching for a maximum value, to obtain a result that will not be affected by the difference of intensities at different regions on the image when applying sliding window process, we will normalize (as the name of the formula says) constraining the obtained values to vary between -1 and 1, this normalization results in the NCC formula . Our algorithm should calculate for every point in storing and coordinates from the maximum values finding the best matches of our template on .

Fourier Transform

Results & discussion

A collage of images

Description automatically generated

A screenshot of a computer

Description automatically generatedA purple and yellow rectangular shapes with white text

Description automatically generated with medium confidence