**Finding all the Faces**

The first step in our pipeline is *face detection*. Obviously we need to locate the faces in a photograph before we can try to tell them apart!

Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a [way to detect faces](https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework) that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We’re going to use [a method invented in 2005](http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf) called Histogram of Oriented Gradients — or just ***HOG*** for short.

To find faces in an image, we’ll start by making our image black and white because we don’t need color data to find faces:

A person taking a selfie

Description automatically generated

Then we’ll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:

A greyscale photo of a black and white square

Description automatically generated

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:

A black and white pixelated background

Description automatically generated

Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right.

If you repeat that process for **every single pixel** in the image, you end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image.

This might seem like a random thing to do, but there’s a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the *direction* that brightness changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve!

But saving the gradient for every single pixel gives us way too much detail. We end up [missing the forest for the trees](https://en.wiktionary.org/wiki/see_the_forest_for_the_trees). It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

To do this, we’ll break up the image into small squares of 16x16 pixels each. In each square, we’ll count up how many gradients point in each major direction (how many point up, point up-right, point right, etc…). Then we’ll replace that square in the image with the arrow directions that were the strongest.

The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way:

A black background with a black square

Description automatically generated

The original image is turned into a HOG representation that captures the major features of the image regardless of image brightnesss.

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces: