Computer Vision

Computer vision is a scientific field, often drawing from artificial intelligence and machine learning, that aims to teach computers to “see” by getting the computer to understand and appropriately respond to information gathered from the content of digital images.

Techniques

Colors

Color Spaces

RGB is red-green-blue color space. You can think of this as a 3D space, in this case a cube, where any color can be represented by a 3D coordinate of R, G, and B values. For example, white has the coordinate (255, 255, 255), which has the maximum value for red, green, and blue.

HSV color space (hue, saturation, and value)

HLS space (hue, lightness, and saturation)

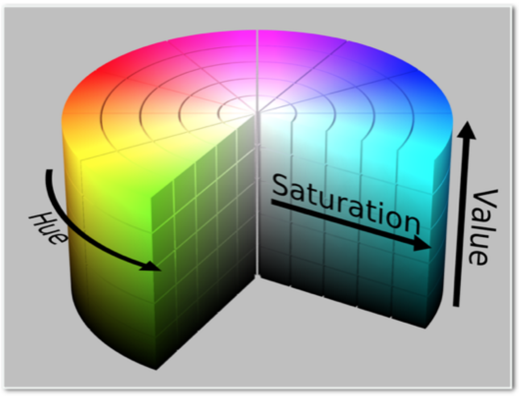
To get some intuition about these color spaces, you can generally think of Hue as the value that represents color independent of any change in brightness. So if you imagine a basic red paint color, then add some white to it or some black to make that color lighter or darker -- the underlying color remains the same and the hue for all of these colors will be the same.

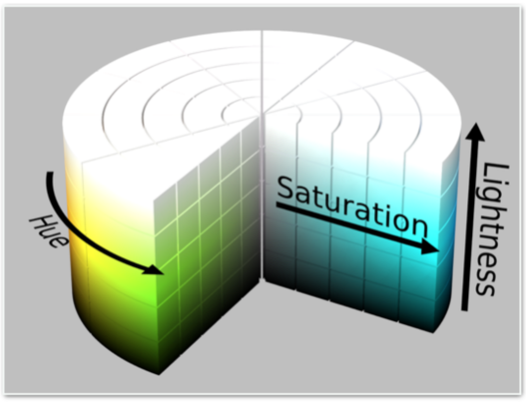
On the other hand, Lightness and Value represent different ways to measure the relative lightness or darkness of a color. For example, a dark red will have a similar hue but much lower value for lightness than a light red.

Saturation is a measurement of colorfulness. So, as colors get lighter and closer to white, they have a lower saturation value, whereas colors that are the most intense, like a bright primary color (imagine a bright red, blue, or yellow), have a high saturation value. You can get a better idea of these values by looking at the 3D color spaces pictured below.

Color selection and region masking

It consists in selecting pixels according to their value per color channel (e.g. RGB). The region masking technique determines a region of interest (ROI) so that we can apply a specific treatment to this region.





Gradient Threshold (magnitude and direction)

Canny edge detection (magnitude)

Canny edge detection (Open_CV)is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It identifies edges of an image by computing gradients at every single pixel of an image.

Canny Edge Detection is a popular edge detection algorithm.It was developed by John F. Canny in1986. It is a multi-stage algorithm:

- **Noise Reduction** using a 5x5 Gaussian filter
- **Finding Intensity Gradient**: Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel
- **Non-maximum Suppression**: a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient.
- **Hysteresis Thresholding**: This stage decides which are all edges are really edges and which are not. Note that "Hysteresis" is the lagging of an effect—a kind of inertia. In the context of thresholding, it means that areas above some low threshold are considered to be above the threshold if they are also connected to areas above a higher, more stringent, threshold.

We'll also include Gaussian smoothing, before running Canny, which is essentially a way of suppressing noise and spurious gradients by averaging (check out the OpenCV docs for GaussianBlur). cv2.Canny() actually applies Gaussian smoothing internally, but we include it here because you can get a different result by applying further smoothing (and it's not a changeable parameter within cv2.Canny()!).

You can choose the kernel_size for Gaussian smoothing to be any odd number. A larger kernel_size implies averaging, or smoothing, over a larger area. The example in the previous lesson was kernel_size = 3.

Reference: OpenCV Canny Edge Detector

Sobel operator

The Sobel operator is at the heart of the Canny edge detection algorithm. Applying the Sobel operator to an image is a way of taking the derivative of the image in the x or y direction. The operators for Sobelx and Sobely, respectively, look like this

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$S_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

These are examples of Sobel operators with a kernel size of 3 (implying a 3 x 3 operator in each case). This is the minimum size, but the kernel size can be any odd number. A larger kernel implies taking the gradient over a larger region of the image, or, in other words, a smoother gradient.

To understand how these operators take the derivative, you can think of overlaying either one on a 3 x 3 region of an image. If the image is flat across that region (i.e., there is little change in values across the given region), then the result (summing the element-wise product of the operator and corresponding image pixels) will be zero.

- Taking the gradient in the x direction emphasizes edges closer to vertical.
- Alternatively, taking the gradient in the y direction emphasizes edges closer to horizontal.

In the case of lane lines, we're interested only in edges of a particular orientation. So now we will explore the direction, or orientation, of the gradient.

The direction of the gradient is simply the inverse tangent (arctangent) of the y gradient divided by the x gradient: arctan(sobely /sobelx), Each pixel of the resulting image contains a value for the angle of the gradient away from horizontal in units of radians, covering a range of $-\pi/2$ to $\pi/2$. An orientation of 0 implies a vertical line and orientations of $+/-\pi/2$ imply horizontal lines.

Hough Transform

In image space, a line is plotted as x vs. y, but in 1962, Paul Hough devised a method for representing lines in parameter space, which we will call “Hough space” in his honor.

In Hough space, I can represent my "x vs. y" line as a point in "m vs. b" instead. The Hough Transform is just the conversion from image space to Hough space. So, the characterization of a line in image space will be a single point at the position (m, b) in Hough space.

Application:

- detection of road lane lines

Distortion correction (camera image)

Perspective Transform

Histogram peak and sliding windows