CMSC 426: Image Processing
Using Gaussians for Color Segmentation

Nitin J. Sanket

September 1, 2016

Score me on color!

Now you have segmented out the image into different colors using hard thresholds or ideal band pass filtering. Often in life you are not so certain to take a decision about which color a pixel is. For eg., which shade of red is an apple? In reality, an apple has multiple shades of red and we need some mathematical formulation which captures this variation and assigns a score as to how red each pixel is.

One of the simplest models/mathematical formulation to this is a Gaussian given below:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

Don't worry if the equation looks too scary, I'll break it down for you. The equation just mentions that the probability of a pixel x being red is given by the above equation. I am not going to derive the complex math behind why is it so, take it for granted or read up online. The two other terms which we haven't discussed yet are μ and σ. μ tells us where the distribution is centred and *sigma* tells us what is the distribution's spread.

Now, you might be wondering what makes this classify 'red' as 'red' and 'yellow' as 'yellow'? That's where the magic of the parameters comes in, you have to set the parameters μ and σ such that it only classifies one color. Also, note that here x, μ and σ all have single values or are scalars. This operation is performed recursively on all pixels. Also, you must play around with the values/transform them to get an image where each color (say red, green or blue) can be extracted easily. This can be as simple as normalizing each color like red/(red+green+blue) to obtain read feature map on which the gaussian is applied. Note that the constant before the exponential is just to normalize and need not be actually used if all you care about is if a value is high or low and the absolute value of the probability density is not important which is generally the case and you can save the extra computation. I generally never compute this. But, beware if you are using this in the part of a pipeline to feed into some other probability, you might want to use it.

If you feel generating individual feature maps to extract each color is tedious and time consuming for a lot of colors, say 20, you're right! It's annoying and nobody does it by hand.

The above example was just to introduce you to Gaussians. Now if you really want to be fancy and automatically learn the gaussians to classify red, green, blue, yellow and so on from an RGB Image you can do so by using a multi-dimensional (3D in this case because we have Red, Green and Blue channels) gaussian distribution. The formula used for this is given by

$$p(x_1, ..., x_k) = \frac{1}{\sqrt{(2\pi)^k\,|\Sigma|}} \exp\left(\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Again, the equation looks fancy but is very similar to the previous one. Let's break it down. The probability has many variables because we have many (k) channels. $k = 3$ in this case. The normalization term can be again ignored but I just put it in for the sake of completeness. x and μ are column vectors of size $k \times 1$. In our case they'll be a column vector of red, green and blue values at each pixel. Σ is the covariance matrix of size $k \times k$ (3×3 in our case). Note that the matrix Σ cannot be any arbitrary matrix, it has to be a Positive Semi-Definite (PSD) matrix (look up wikipedia for properties of positive definite-matrix). It is not easy to just get a matrix which is a PSD. How this is done is the interesting part. Just like a human learns by looking at examples, we need to present the computer with examples of how a color looks like. This phase is called the training phase and is described next.

Let us say I want to identify red colored apples. I'll collect multiple examples of how a red apple looks (with slight illumination variations and slight color/shade variations). I take the mean (using the MATLAB function `mean`) of all the values in red,green and blue channels. This is my 3×1 mean color vector or μ. Yes! It's as simple as calling one function. Now, to find the covariance, we use all the values and use the MATLAB function `cov` to find the 3×3 covariance matrix Σ. Yes! I know right! MATLAB makes our life so simple. If you are interested on how this is done look at this link: http://www.itl.nist.gov/div898/handbook/pmc/section5/pmc541.htm. It's pretty simple so try to know it. Now you have your model for classifying red color of apples. Just compute the probability/score for each pixel. Then you can use a threshold like 70% of maximum value is an apple. You're done. You just built your own fancy classifier to classify red colored apples using a 3D Gaussian. Great Job!

If you want to take it one step further to make your model more robust, you can look up Gaussian Mixture Models (GMMs) which is a pretty advanced topic for this class. If you want to know about this come to me during office hours and I'll be delighted to explain it. It is very powerful and used on most of our robots in the lab (including drones and humanoids) due to its implementation simplicity, speed, robustness and relatively quick training times.

If anything was unclear or wrong in this document feel free to e-mail me at nitinsan@terpmail.umd.edu.