Recognition of a picture of a species

Huhn-Kie Lee

I. Introduction

Assume you take a picture of a butterfly in the field with a digital camera. You save the digital picture in your computer as a JPEG file. Now, you want to know what species the butterfly is, and find detailed information about the butterfly. The first software package you need is the "species recognizer" program. Second, the recognizer will analyze your butterfly picture and pass the analysis result to the "picture matcher" program, which will compare the analysis result with "butterfly database" to find the matching butterfly in the database.

Similarly, if you want to know the species of a plant, you can cut off its leaf or flower and take a picture. You can also photograph the contour (or branching pattern) of a tree or a cactus.

II. Four "apparent" difficulties of recognizing a picture by color composition

At first sight, the problem looks difficult. Maybe you can carefully take a picture so that there is no "partial" shadow that "partially" overhang the wings of a butterfly. Say, if there is such a partial shadow, you can block the sunlight so that the shadow completely and uniformly overhang the wings of a butterfly. So the picture of the butterfly does not have any partial shadow now. The problem now is the "brightness" of the picture: the brightness will depend on how bright the day was. This is the first problem and I call it "brightness problem". Second, the pictures of the same butterfly may contain different sizes: this is the "size problem". Third, the picture may photographed the butterfly in different rotation angles, like upside down: this is the "rotation problem." Fourth, butterfly moves its wings, so the "tilted angle" of the wings will differ in different pictures: this is the "angle" problem.

In sum, my recognizer algorithm solves all these four problems.

III. Preliminary: representation of a picture of a species

So you took a picture of a species that does not have any partial shadows over it. Assume your camera is not digital, but a traditional one. You get the picture from the photo shop. You take a scissors and cut off the exact "body" of the species of the picture, and you scan it to your computer to save it as JPEG file. The JPEG file will store the picture as a set of matrices, whose entry corresponds to each pixel of the picture. One color pixel is in RGB format (red, green, blue). So you need 3 matrices for red, green, blue: I call the matrices R, G, B. Consider a pixel in position (i, j) in the picture. Each entry can have value from 0 to 16. If the R(i, j)=16 and G(i, j)=16 and B(i, j)=16, then the pixel at

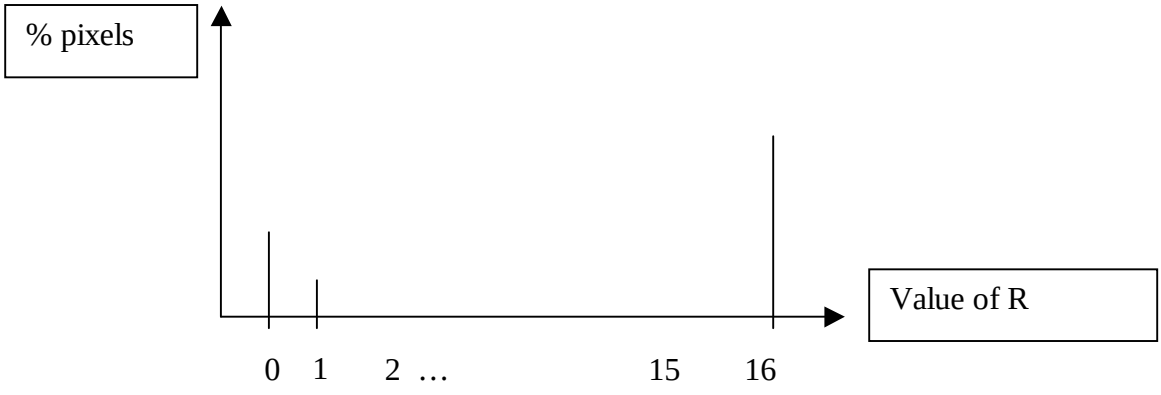
R(i, j) is black. If the R(i, j)=16 and G(i, j)=16 and B(i, j)=0, then the pixel at R(i, j) is yellow, and so on.

IV. The purpose of the recognizer algorithm

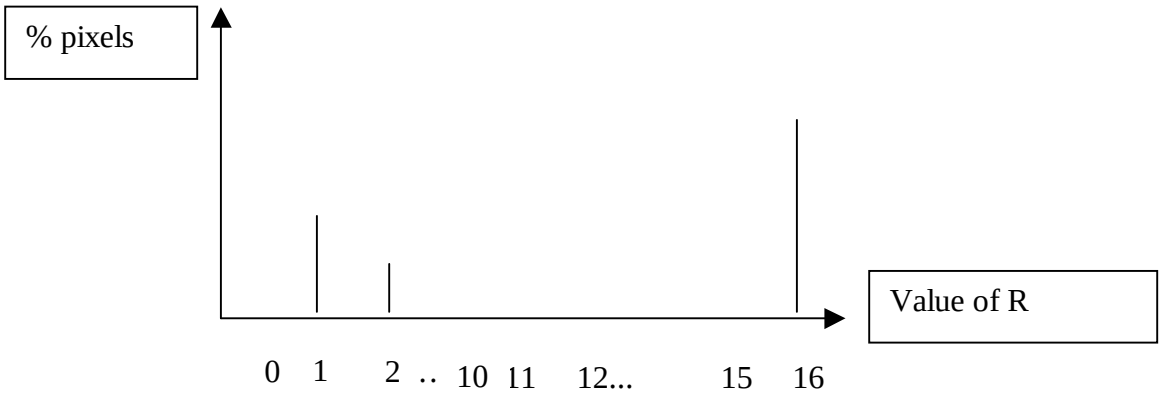
Note that the purpose of my recognizer algorithm is not to give you the exact answer, but to narrow down possibility. Say the butterfly database contains 10,000 species. To the query processor of the database, you say you took the picture of butterfly in Israel-Jordan border. The query processor now narrowed down the number of possible species into 1,000 butterflies. Now, my recognizer will analyze your picture and narrow the number down into 10 butterflies. This is small enough: you can eyeball the 10 butterfly pictures and pick one yourself !

V. The mechanism of recognizer algorithm

To solve the "rotation", "angle", "size" problem, you need to represent the picture as a set of "percentage". Say, a wing of a butterfly has yellow background and black stripes. The "percentage" of yellow and black does not change even if you enlarge the picture, rotate the picture, or tilt the angle of the wing when you take the picture. So three problems are solved. Now the brightness problem. Remember that my algorithm's job is to "narrow down" the number of possibility. The input of the algorithm is the three matrices R, G, B that represent the pixel values of the picture. Here, the "picture" means the exact "body" of the butterfly, and not the background: we cut the background out with the scissors. Now, consider matrix R. For the matrix R, the recognizer counts the number of entries that has the value of 0. The recognizer counts the number of entries that has the value of 1. Do this all the way up to value 16. Divide these numbers by the total number of the pixels of the picture to get the percentage. The plot will look like:



Now, do the same for G, B matrices as well. Now, assume you took the picture on a darker day then above. Then the new plot will look like:



But we want to recognize the above two pictures as the same. Hence, the recognizer simply sort the vertical bars above by length. (Maybe sorting is too insensitive method because it does not reflect the preserved order. Then simply push the vertical bars all the way to the left.) Do the same for matrices G, B. And these three sorted lists of R, G, B values are the new representation of the picture. This solves the brightness problem.