Otsu

En vision par ordinateur et traitement d’image, la méthode d’Otsu est utilisée pour effectuer un seuillage automatique à partir de la forme de l’histogramme de limage, ou la réduction d’une image à niveaux de gris en une image binaire. L’algorithmique suppose alors que l’image à binariser ne contient que deux classes de pixels,(c’est à dire le premier plan et l’arrière plan) puis calcule le seuil optimal qui sépare ces deux classes afin que leur variance intra-class soit minimale. L’ extension de la méthode originale pour faire du seuillage à plusieurs niveaux est appelée Multi Otsu méthode. Le nom cette méthode provient du nom de son initiateur, Nobuyuki Otsu.

Méthode

Dans la méthode d’Otsu, le seuil qui minimise la variance intra-classe est recherché à partir de tous les seuillage possible:

IMG_256

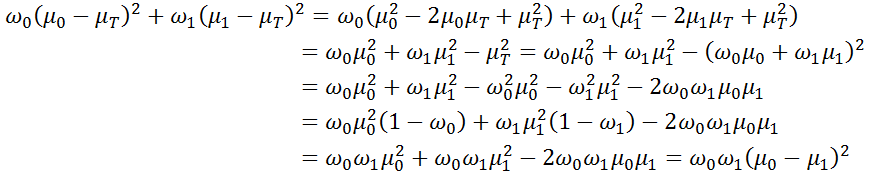
Les poids IMG_257 repensent la probabilité d’être dans la IMG_258ème classe, chacune étant séparée par un seuil t. Finalement, les IMG_260 sont les variances de ces classe(ici on n’a que deux classes, la première plan et l’arrière plan).

Otsu montre que minimiser la variance intra-classe revient à maximiser la variance inter-classe:



Qui est exprimée en termes des probabilité de classe IMG_262 et des moyennes de classes IMG_263 qui à leur tour peuvent être mises à jour iterativement. Cette idée conduit à un algorithme efficace.

On a:



Algorithme sur python：

|  |
| --- |
| function [threshold\_otsu] = Thredsholding\_Otsu( Image)  %Intuition:  %(1)pixels are divided into two groups  %(2)pixels within each group are very similar to each other  % Parameters:  % t : threshold  % r : pixel value ranging from 1 to 255  % q\_L, q\_H : the number of lower and higher group respectively  % sigma : group variance  % miu : group mean  % Author: Lei Wang  % Date : 22/09/2013  % References : Wikepedia,  % This is my original work  nbins = 256;counts = imhist(Image,nbins);p = counts / sum(counts);  for t = 1 : nbins  q\_L = sum(p(1 : t));  q\_H = sum(p(t + 1 : end));  miu\_L = sum(p(1 : t) .\* (1 : t)') / q\_L;  miu\_H = sum(p(t + 1 : end) .\* (t + 1 : nbins)') / q\_H;  sigma\_b(t) = q\_L \* q\_H \* (miu\_L - miu\_H)^2;  end  [~,threshold\_otsu] = max(sigma\_b(:));  end |