Color Histogram Normalization using Matlab and Applications in CBIR

László Csink, Szabolcs Sergyán
Budapest Tech
SSIP’05, Szeged
Outline

- Introduction
- Demonstration of the algorithm
- Mathematical background
- Computational background: Matlab
- Presentation of the algorithm
- Evaluation of the test
- Conclusion
Introduction (1)

- Retrieval in large image databases
  - Textual key based
    - Key generation is subjective and manual
    - Retrieval is errorless
  - Content Based Image Retrieval (CBIR)
    - Key generation is automatic (possibly time consuming)
    - Noise is unavoidable
Introduction (2)

- If a new key is introduced, the whole database has to be reindexed.
- In textual key based retrieval the reindexing requires a lot of human work, but in CBIR case it requires only a lot of computing.
Introduction (3)

n CBIR
  n Low level
    n Color, shape, texture
  n High level
    n Image interpretation
    n Image understanding
Introduction (4)

- Typical task of low level CBIR
  - Search for a given object using similarity distance based on content keys
- One way of defining similarity distance is to use color histograms – we concentrate on this approach in the present talk
Demonstration of the algorithm
Image versions and their histograms
Two images and their histograms
Similarity distance between two image histograms

\[ d_{\text{Euclid}}(h_x, h_y) = \sqrt{\sum_{r=1}^{4} \sum_{g=1}^{4} \sum_{b=1}^{4} (h_{x\text{rgb}} - h_{y\text{rgb}})^2} \]
Different illuminations
Normalized versions
Normalization may change image outlook
Normalization may change image outlook
Mathematical background
Color cluster analysis

\[
f = \begin{bmatrix} f_{ij} \end{bmatrix} \quad i = 1, K, M; \quad j = 1, K, N
\]

1. Compute the cluster center of all pixels \( f \) by 
   \( m = E[f]. m \) is a vector which points to the 
   center of gravity.

2. \( C = E[(f - m)(f - m)^T] \)
   
   The eigenvalues \((\lambda_1, \lambda_2, \lambda_3)\) and 
   eigenvectors of \( C \) are computed directly.

3. Denote the eigenvector belonging to the 
   largest eigenvalue by \( v = (a, b, c)^T \).
Rodrigues formula

Rotating \( \mathbf{v} \) along \( \mathbf{n} \) (\( \mathbf{n} \) is a unit vector) by \( \theta \): \( R \mathbf{v} \), where

\[
R = I + U(n) \sin \theta + U(n)^2 (1 - \cos \theta)
\]

\[
U(n) = \begin{pmatrix}
0 & -n_3 & n_2 \\
n_3 & 0 & -n_1 \\
-n_2 & n_1 & 0
\end{pmatrix}
\]
Color rotation in RGB-space
Color rotation in RGB-space

\[ n' = (a, b, c)^T \times \frac{1}{\sqrt{3}} (1,1,1)^T \]

\[ \cos \theta' = (a, b, c)^T \cdot \frac{1}{\sqrt{3}} (1,1,1)^T \]
Color rotation in RGB-space

4. Use the Rodrigues formula in order to rotate with \( \theta' \) around \( n' \).

5. Shift the image along the main axis of the RGB-cube by \((128, 128, 128)^T\).

6. Clip the overflows above 255 and the underflows under 0.
Computational background

Fundamentals of MATLAB
Presentation of the algorithm
MATLAB code

function color_normalization(FILENAME, OUTPUT);

inp_image = imread(FILENAME);  % read input image
[m,n,d]=size(inp_image);  % get size of input image
f=double(inp_image);  % double needed for computations
M=zeros(m*n,3);
z=1;
mv=mean(mean(f));  % a vector containing the mean r,g and b value
v1=[mv(1),mv(2),mv(3)];  % means in red, green and blue
MATLAB code

for i=1:m
    for j=1:n
        v=[f(i,j,1),f(i,j,2),f(i,j,3)];  % image pixel at i,j
        M(z,:) = v - v1;  % image normed to mean zero
        z = z + 1;
    end
end
C = cov(M);  % covariance computed using Matlab cov function
MATLAB code

%find eigenvalues and eigenvectors of C.

[V,D]=eig(C); % computes the eigenvectors(V) and eigenvalues (diagonal elements of D) of the color cluster C

%get the max. eigenvalue meig and the corresponding eigenvector ev0.

meig = max(max(D)); % computes the maximum eigenvalue of C.
    Could also be norm(C)

if(meig==D(1,1)), ev0=V(:,1);, end
if(meig==D(2,2)), ev0=V(:,2);, end
if(meig==D(3,3)), ev0=V(:,3);, end

% selects the eigenvector belonging to the greatest eigenvalue
MATLAB code

\begin{verbatim}
Idmat = eye(3);  % identity matrix of dimension 3
wbaxis = [1;1;1]/sqrt(3);  % unit vector pointing from origin along the main
diagonal
nvec = cross(ev0, wbaxis);  % rotation axis, cross(A,B)=A\times B
cosphi = dot(ev0, wbaxis)  % dot product, i.e. sum((ev0.*wbaxis))
sinphi = norm(nvec);  % sinphi is the length of the cross product of two
unit vectors
nvec = nvec/sinphi;  % normalize nvec
\end{verbatim}
if(cosphi>0.99)
    f=uint8(f);
    imwrite(f,OUTPUT); %in this case we dont normalize, output is input etc.
else
    % we normalize
    n3 = nvec(3); n2 = nvec(2); n1 = nvec(1);
    % remember: this is a unit vector along the rotation axis
    U = [[ 0  -n3  n2]; [ n3  0  -n1]; [ -n2 n1  0]]; %
    U2 = U*U;
    Rphi = Idmat + (U*sinphi) + (U2*(1-cosphi));
MATLAB code

n0   = [0 0 0]';
n255 = [255 255 255]';
for i=1:m
    for j=1:n
        s(1)= f(i,j,1)-mv(1); % compute vector s of normalized image at i,j
        s(2)= f(i,j,2)-mv(2);
        s(3)= f(i,j,3)-mv(3);
        t = Rphi*s'; % s transposed, as s is row vector, then rotated
        tt = floor(t + [128 128 128]'); % shift to middle of cube and make it integer
MATLAB code

```
tt = max(tt,n0);  % handling underflow
tt = min(tt,n255);  % handling overflow

g(i,j,:)=tt;
end
end

g=uint8(g);
imwrite(g,OUTPUT);
end  % end of normalization
```
Evaluation of the test
Test databases

- 5 objects
- Alternative color cubes
- 3 illuminations
- 3 background
- 90 images
Some test results (without normalization)
Some test results (with normalization)
Conclusions
References


Thank you for your attention!

Csink.laszlo@nik.bmf.hu       Sergyan.szabolcs@nik.bmf.hu