Trường ĐHSPKT TP.HCM **ĐÁP ÁN MÔN: CHUYÊN ĐỀ 1 – XỬ LÝ ẢNH SỐ**

Khoa Công nghệ Thông tin **MÃ MÔN HỌC:** **1279230**

Bộ môn Tin học cơ sở **Học kỳ I – Năm học 2014-2015**

---------------------- ------------------------------

**Câu 1: (2đ)**

Mục đích của việc cân bằng histogram? Xây dựng và cài đặt thuật toán cân bằng histogram tự động.

**Trả lời:**

Mục đích của cân bằng histogram là làm cho histogram đồng đều. Khi đó ta làm tăng được độ tương phản của ảnh.

Cân bằng histogram được cho bằng phương trình:

s=T(r)=(L-1)

với p*r*(w) : Xác suất xảy ra mức xám *w*

Trong xác suất, tích phân của hàm mật độ là hàm phân phối. Công thức trên có *w* là biến liên tục, ta không thể lập trình nó. Ta phải dùng công thức rời rạc:

sk=T(rk)=(L-1)

với k= 0,1,2,…,L-1

**Code:**

void HistogramEqualization(Mat imgin,Mat imgout)

{

int x, y, M, N;

int h[L];

int r;

M = imgin.size().height;

N = imgin.size().width;

for (r=0; r<L; r++)

h[r] = 0;

for (x=0; x<M; x++)

for (y=0; y<N; y++) {

r = imgin.at<uchar>(x,y);

h[r]++;

}

double p[L];

for (r=0; r<L; r++)

p[r] = 1.0\*h[r]/(M\*N);

double T[L];

int j, k;

for (k=0; k<L; k++) {

T[k] = 0;

for (j=0; j<=k; j++)

T[k] += p[j];

}

double s;

for (x=0; x<M; x++)

for (y=0; y<N; y++) {

r = imgin.at<uchar>(x,y);

s = T[r];

imgout.at<uchar>(x,y) = (uchar)((L-1)\*s);

}

return;

}

**Câu 2: (3đ)**

Trình bày các bước lọc ảnh trong miền tần số (gồm 7 bước).

Trình bày bộ lọc Notch Reject và ứng dụng bộ lọc Notch Reject để xoá nhiễu Moiré.

Xây dựng và cài đặt thuật toán xoá nhiễu Moiré của ảnh sau:

**Trả lời:**

Lọc ảnh trong miền tần số gồm 7 bước sau đây:

Bước 1: Cho ảnh đầu vào f(x,y) có kích thước MxN. Mở rộng ảnh có kích thước là PxQ. OpenCV có hàm xác định kích thước P và Q tối ưu.

Bước 2: Thêm zero vào phần mở rộng, ta được ảnh fp(x,y).

Bước 3: Nhân fp(x,y) với (-1)x+y để dời F(0,0) vào tâm ảnh.

Bước 4: Biến đổi Fourier của ảnh ở Bước 3 ta được F(u,v).

Bước 5: Cho hàm lọc có giá trị thực H(u,v) đối xứng qua tâm (P/2,Q/2). Thực hiện phép nhân

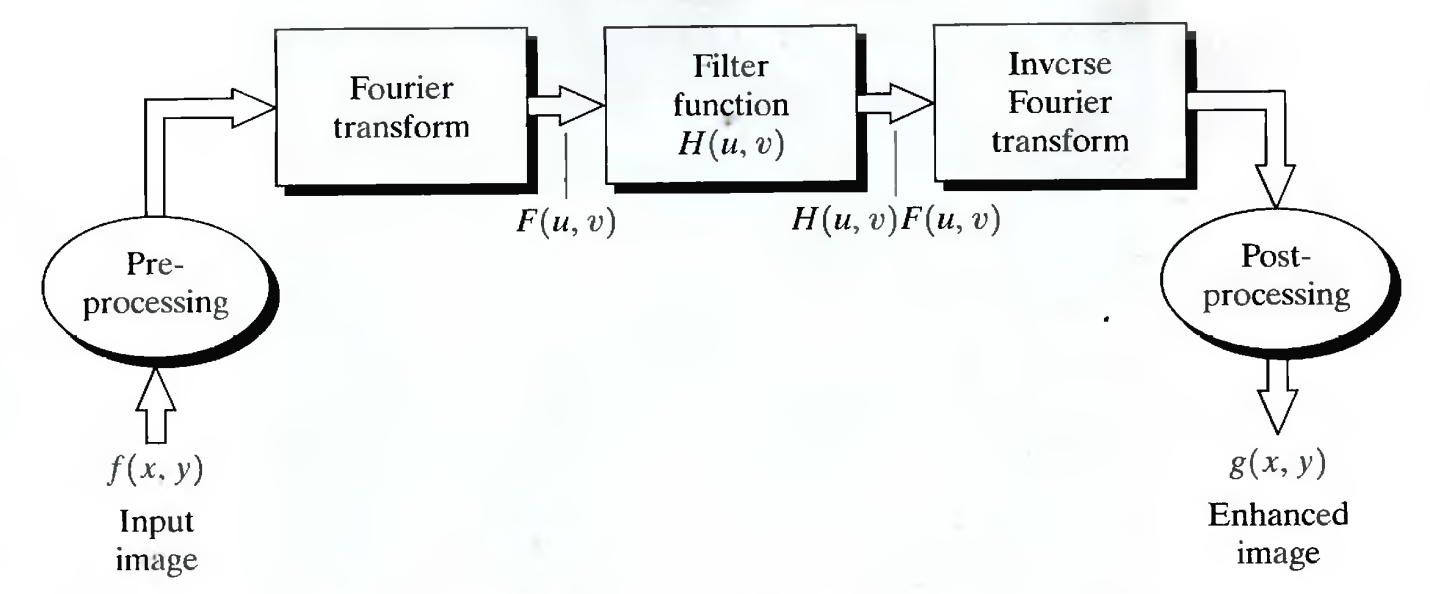
G(u,v) = F(u,v)H(u,v).

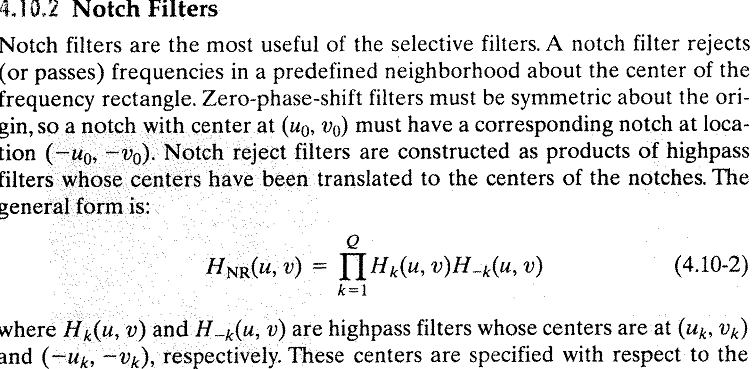
Bước 6: Thu được ảnh đã xử lý bằng biến đổi Fourier ngược, lấy phần thực và dời trở lại gốc tọa độ

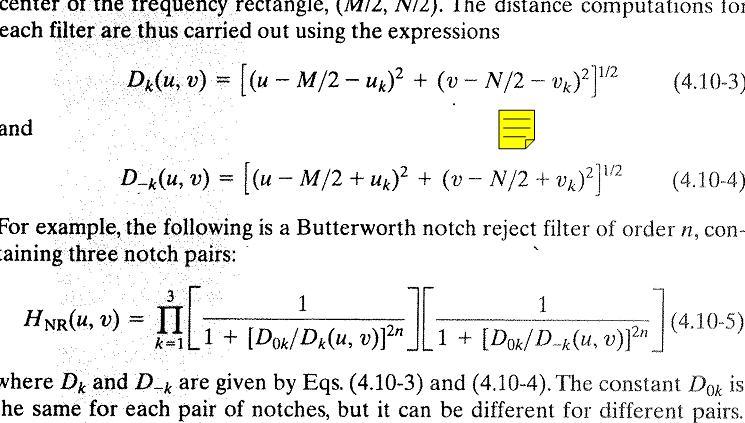


Bước 7: Bỏ phần đã mở rộng, ta thu được ảnh g(x,y) có kích thước MxN.

Biểu diễn bằng sơ đồ khối:







**Code:**

void RemoveMoire(Mat imgin, Mat imgout)

{

int M = imgin.size().height;

int N = imgin.size().width;

// Buoc 1, 2, 3

int P = getOptimalDFTSize(M);

int Q = getOptimalDFTSize(N);

Mat f = Mat(P,Q,CV\_32FC2,CV\_RGB(0,0,0));

Mat F = Mat(P,Q,CV\_32FC2,CV\_RGB(0,0,0));

int x, y, u, v;

for (x=0; x<M; x++)

for (y=0; y<N; y++)

if ((x+y)%2 == 0)

f.at<Vec2f>(x,y)[0] = 1.0\*imgin.at<uchar>(x,y);

else

f.at<Vec2f>(x,y)[0] = -1.0\*imgin.at<uchar>(x,y);

// Buoc 4

dft(f,F);

// Buoc 5

Mat H = Mat(P,Q,CV\_32FC2,CV\_RGB(0,0,0));

double D01 = 5, D02 = 5, D03 = 15, D04 = 15;

int u1 = 44, v1 = 59, u2 = 40, v2 = 119, u3 = 86, v3 = 58, u4 = 82, v4 = 119;

double D1, D1Tru, D2, D2Tru, D3, D3Tru, D4, D4Tru;

double r;

for (u=0; u<P; u++)

for (v=0; v<Q; v++) {

D1 = sqrt(1.0\*(u-u1)\*(u-u1) + 1.0\*(v-v1)\*(v-v1));

D1Tru = sqrt(1.0\*(u-P+u1)\*(u-P+u1) + 1.0\*(v-Q+v1)\*(v-Q+v1));

r = (1/(1+pow(D01/D1,2\*4)))\*(1/(1+pow(D01/D1Tru,2\*4)));

D2 = sqrt(1.0\*(u-u2)\*(u-u2) + 1.0\*(v-v2)\*(v-v2));

D2Tru = sqrt(1.0\*(u-P+u2)\*(u-P+u2) + 1.0\*(v-Q+v2)\*(v-Q+v2));

r \*= (1/(1+pow(D02/D2,2\*4)))\*(1/(1+pow(D02/D2Tru,2\*4)));

D3 = sqrt(1.0\*(u-u3)\*(u-u3) + 1.0\*(v-v3)\*(v-v3));

D3Tru = sqrt(1.0\*(u-P+u3)\*(u-P+u3) + 1.0\*(v-Q+v3)\*(v-Q+v3));

r \*= (1/(1+pow(D03/D3,2\*4)))\*(1/(1+pow(D03/D3Tru,2\*4)));

D4 = sqrt(1.0\*(u-u4)\*(u-u4) + 1.0\*(v-v4)\*(v-v4));

D4Tru = sqrt(1.0\*(u-P+u4)\*(u-P+u4) + 1.0\*(v-Q+v4)\*(v-Q+v4));

r \*= (1/(1+pow(D04/D4,2\*4)))\*(1/(1+pow(D04/D4Tru,2\*4)));

H.at<Vec2f>(u,v)[0] = (float)r;

}

Mat G = Mat(P,Q,CV\_32FC2,CV\_RGB(0,0,0));

mulSpectrums(F,H,G,DFT\_ROWS);

// Buoc 6, 7

Mat g = Mat(P,Q,CV\_32FC2,CV\_RGB(0,0,0));

idft(G,g,DFT\_SCALE);

for (x=0; x<M; x++)

for (y=0; y<N; y++) {

if ((x+y)%2 == 0)

r = g.at<Vec2f>(x,y)[0];

else

r = -g.at<Vec2f>(x,y)[0];

if (r < 0)

r = 0;

if (r > L-1)

r = L-1;

imgout.at<uchar>(x,y) = (uchar)r;

}

return;

}

**Câu 3: (2.5đ)**

Xây dựng và cài đặt thuật toán đếm hạt gạo có trong ảnh sau:

**Trả lời:**

Để đếm hạt gạo như trong hình vẽ, ta phải thực hiện các bước sau đây:

Bước 1: Dùng biến đổi top-hat để làm đậm thêm phần bóng đổ của hạt gạo.

Bước 2: Phân ngưỡng để biến ảnh xám thành ảnh nhị phân.

Bước 3: Dùng hàm FloodFill để tô màu các thành phần liên thông.

Bước 4: Đếm số lượng màu đã sử dụng. Đó chính là số lượng hạt gạo có trong ảnh.

**Code:**

void CountRice(Mat imgin,Mat imgout)

{

Mat temp = Mat(imgin.size(),CV\_8UC1);

Mat w = getStructuringElement(MORPH\_ELLIPSE,Size(81,81));

morphologyEx(imgin,temp,MORPH\_TOPHAT,w);

double max;

minMaxLoc(temp,0,&max);

threshold(temp,temp,0.4\*max,255,THRESH\_BINARY);

medianBlur(temp,temp,5);

int fromTo[] = {0,0,0,1,0,2};

mixChannels(&temp,1,&imgout,1,fromTo,3);

int x, y, M, N;

M = imgout.size().height;

N = imgout.size().width;

Vec3b p;

uchar r, g, b;

int color = 100;

for (x=0; x<M; x++)

for (y=0; y<N; y++) {

p = imgout.at<Vec3b>(x,y);

if (p[0] == L-1 && p[1] == L-1 && p[2] == L-1) {

r = color%256;

g = (color/256)%256;

b = ((color/256)/256)%256;

floodFill(imgout,Point(y,x),CV\_RGB(r,g,b));

color++;

}

}

int k, mang[1000];

for (k=0; k<1000; k++)

mang[k] = 0;

for (x=0; x<M; x++)

for (y=0; y<N; y++) {

p = imgout.at<Vec3b>(x,y);

r = p[2];

g = p[1];

b = p[0];

k = r + g\*256 + b\*256\*256;

if (k > 0)

mang[k]++;

}

FILE \*fp = fopen("ThongKe.txt","wt");

fprintf(fp,"Co %d hot gao\n",color-100);

for (k=0; k<1000; k++)

if (mang[k] > 0)

fprintf(fp,"Mau %3d co %3d pixel\n",k,mang[k]);

fclose(fp);

char buffer[256];

sprintf(buffer,"Co %d hot gao",color-100);

putText(imgout,buffer,Point(0,25),FONT\_HERSHEY\_DUPLEX,0.7,CV\_RGB(0,255,0));

return;

}

**Câu 4: (2.5đ)**

Xây dựng và cài đặt thuật toán nhận dạng khuôn mặt dùng phương pháp khuôn mặt riêng (eigenface).

**Trả lời:**

OpenCV có sẳn các hàm nhận dạng khuôn mặt dùng phương pháp khuôn mặt riêng, gồm 2 bước:

Bước 1: Huấn luyện dùng các điểm PCA.

Bước 2: Nhận dạng.

**Code:**

void TrainningAndRecognition(void)

{

vector<Mat> images;

vector<int> labels;

char filename[128];

for (int i=1; i<=10; i++) {

sprintf(filename,"faces\\ThayDuc\\%02d.bmp",i);

Mat image = imread(filename,CV\_LOAD\_IMAGE\_GRAYSCALE);

resize(image,image,Size(120,120));

images.push\_back(image);labels.push\_back(1);

}

for (int i=1; i<=10; i++) {

sprintf(filename,"faces\\Phuong\\%02d.bmp",i);

Mat image = imread(filename,CV\_LOAD\_IMAGE\_GRAYSCALE);

resize(image,image,Size(120,120));

images.push\_back(image);labels.push\_back(2);

}

for (int i=1; i<=10; i++) {

sprintf(filename,"faces\\Tuan\\%02d.bmp",i);

Mat image = imread(filename,CV\_LOAD\_IMAGE\_GRAYSCALE);

resize(image,image,Size(120,120));

images.push\_back(image);labels.push\_back(3);

}

Ptr<FaceRecognizer> model = createEigenFaceRecognizer();

model->train(images,labels);

Mat test = imread(“test.bmp”,CV\_LOAD\_IMAGE\_GRAYSCALE);

resize(test,test,Size(120,120));

int predict = model->predict(test);

char name[128];

if (predict == 1)

sprintf(name,"Duc");

else if (predict == 2)

sprintf(name,"Phuong");

else if (predict == 3)

sprintf(name,"Tuan");

else

sprintf(name,"Khong Biet");

putText(test,name,Point(0,15),FONT\_HERSHEY\_DUPLEX,1,CV\_RGB(255,255,255));

namedWindow("Result");

imshow("Result",test);

waitKey(0);

return;

}

--------------------HẾT--------------------