P60 1

单向一阶锐化函数文件singleDirSharpen.m:

function g = singleDirSharpen( f, h )

[moduleLen, moduleCol] = size(h);

[m,n] = size(f);

g=f;

if m>=moduleLen && n>=moduleLen

t = floor(moduleLen/2);

for i=1:m-moduleLen+1

for j=1:n-moduleLen+1

sum = double(0);

for k=1:moduleLen

for l=1:moduleLen

sum = sum + double(f(i+k-1, j+l-1))\*double(h(k,l));

end

end

g(i+t, j+t) = uint8(sum);

end

end

end

end

全方向一阶Priwitt分子锐化函数文件priwittSharpen.m:

function g = priwittSharpen( f, dx, dy )

[moduleLen, moduleCol] = size(dx);

[m,n] = size(f);

g=f;

if m>=moduleLen && n>=moduleLen

t = floor(moduleLen/2);

for i=1:m-moduleLen+1

for j=1:n-moduleLen+1

sum1 = double(0);

sum2 = double(0);

for k=1:moduleLen

for l=1:moduleLen

sum1 = sum1 + double(f(i+k-1, j+l-1))\*double(dx(k,l));

sum2 = sum2 + double(f(i+k-1, j+l-1))\*double(dy(k,l));

end

end

g(i+t, j+t) = sqrt(sum1\*sum1+sum2\*sum2);

end

end

end

end

测试文件sharpenDemo.m:

rows = 2;

cols = 2;

f = imread('img/Fig1006(a).tif');

h1 = [1,2,1;0,0,0;-1,-2,-1];

g1 = singleDirSharpen(f, h1);

h2 = [1,0,-1;2,0,-2;1,0,-1];

g2 = singleDirSharpen(f, h2);

dx = [-1,-1,-1;0,0,0;1,1,1];

dy = [-1,0,1;-1,0,1;-1,0,1];

g3 = priwittSharpen(f, dx, dy);

figure();

subplot(rows, cols, 1);

imshow(f);

title('原图');

subplot(rows, cols, 2);

imshow(g3);

title('任意方向一阶priwitt锐化');

subplot(rows, cols, 3);

imshow(g1);

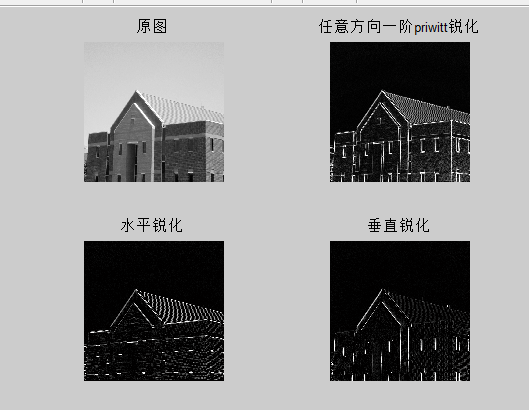
title('水平锐化');

subplot(rows, cols, 4);

imshow(g2);

title('垂直锐化');

测试结果：

水平锐化勾勒出水平边界，垂直锐化勾勒出垂直边界，任意方向勾勒出两者。

P77

1

水平镜像函数文件horizontalMirror.m:

function g = horizontalMirror( f )

[m, n] = size(f);

g = zeros(m, n, 'uint8');

for i=1:m

for j=1:n

g(i,j)=f(i,n-j+1);

end

end

end

垂直镜像函数文件verticalMirror.m:

function g = verticalMirror( f )

[m, n] = size(f);

g = zeros(m, n, 'uint8');

for i=1:m

for j=1:n

g(i,j)=f(m-i+1,j);

end

end

end

旋转函数文件myRotate.m:

function g = myRotate( f, a )

[m,n] = size(f);

%背景灰度值

bgValue = 255;

%变换后的坐标

I = zeros(m,n,'int16');

minI=1;

maxI=1;

J = zeros(m,n,'int16');

minJ=1;

maxJ=1;

for i=1:m

for j=1:n

I(i,j)=int16( double(i\*cos(a))-double(j\*sin(a)) );

if I(i,j)<minI

minI = I(i,j);

end

if I(i,j)>maxI

maxI = I(i,j);

end

J(i,j)=int16( double(i\*sin(a))+double(j\*cos(a)) );

if J(i,j)<minJ

minJ = J(i,j);

end

if J(i,j)>maxJ

maxJ = J(i,j);

end

end

end

%负数则平移

if minI<=0

maxI = maxI-minI+1;

I = I-minI+1;

end

if minJ<=0

maxJ = maxJ-minJ+1;

J = J-minJ+1;

end

g = zeros(maxI, maxJ, 'uint8');

%初始化为背景颜色

g = g+bgValue;

%填充

for i=1:m

for j=1:n

g(I(i,j), J(i,j))=f(i,j);

end

end

%填坑

for i=1:maxI

front = 1;

rear = maxJ;

%寻找左右非背景边界

while front<rear && g(i,front)==bgValue

front=front+1;

end

while rear>front && g(i,rear)==bgValue

rear=rear-1;

end

%填坑

for j=front:rear

if g(i,j)==bgValue

g(i,j)=g(i,j-1);

end

end

end

end

垂直错切文件verticalShear.m:

function g = verticalShear( f, b )

[m,n] = size(f);

%背景颜色

bgValue = 255;

%错切后的坐标

I = zeros(m,n,'int16');

minI=1;

maxI=1;

J = zeros(m,n,'int16');

for i=1:m

for j=1:n

I(i,j)=int16( double(i)+double(b\*j) );

if I(i,j)<minI

minI = I(i,j);

end

if I(i,j)>maxI

maxI = I(i,j);

end

J(i,j)=j;

end

end

%负数则平移

if minI<=0

maxI = maxI-minI+1;

I = I-minI+1;

end

g = zeros(maxI, n, 'uint8');

%初始化为背景颜色

g = g+bgValue;

%填充灰度值

for i=1:m

for j=1:n

g(I(i,j), J(i,j))=f(i,j);

end

end

%填坑

for i=1:maxI

front = 1;

rear = n;

%寻找左右非背景边界

while front<rear && g(i,front)==bgValue

front=front+1;

end

while rear>front && g(i,rear)==bgValue

rear=rear-1;

end

%填坑

for j=front:rear

if g(i,j)==bgValue

g(i,j)=g(i,j-1);

end

end

end

end

水平错切文件horizontalShear.m:

function g = horizontalShear( f, d )

[m,n] = size(f);

%背景灰度值

bgValue = 255;

%变换后的坐标

I = zeros(m,n,'int16');

J = zeros(m,n,'int16');

minJ=1;

maxJ=1;

for i=1:m

for j=1:n

I(i,j)=i;

J(i,j)=int16( double(i)+double(d\*j) );

if J(i,j)<minJ

minJ = J(i,j);

end

if J(i,j)>maxJ

maxJ = J(i,j);

end

end

end

%负数则平移

if minJ<=0

maxJ = maxJ-minJ+1;

J = J-minJ+1;

end

g = zeros(m, maxJ, 'uint8');

%初始化为背景颜色

g = g+bgValue;

%填充

for i=1:m

for j=1:n

g(I(i,j), J(i,j))=f(i,j);

end

end

%填坑

for i=1:m

front = 1;

rear = maxJ;

%寻找左右非背景边界

while front<rear && g(i,front)==bgValue

front=front+1;

end

while rear>front && g(i,rear)==bgValue

rear=rear-1;

end

%填坑

for j=front:rear

if g(i,j)==bgValue

g(i,j)=g(i,j-1);

end

end

end

end

测试文件transformDemo.m:

rows = 2;

cols = 5;

f = imread('img/Fig1006(a).tif');

g1 = horizontalMirror(f);

g2 = verticalMirror(f);

g3 = myRotate(f, 0.5);

g4 = myRotate(f, 2);

g5 = verticalShear(f, 0.5);

g6 = verticalShear(f, 2);

g7 = horizontalShear(f, 0.5);

g8 = horizontalShear(f, 2);

figure();

subplot(rows, cols, 1);

imshow(f);

title('原图');

subplot(rows, cols, 3);

imshow(g1);

title('水平镜像图');

subplot(rows, cols, 4);

imshow(g2);

title('垂直镜像图');

subplot(rows, cols, 5);

imshow(g3);

title('弧度为0.5的旋转图');

subplot(rows, cols, 6);

imshow(g4);

title('弧度为2的旋转图');

subplot(rows, cols, 7);

imshow(g5);

title('系数为0.5的水平错切图');

subplot(rows, cols, 8);

imshow(g6);

title('系数为2的水平错切图');

subplot(rows, cols, 9);

imshow(g7);

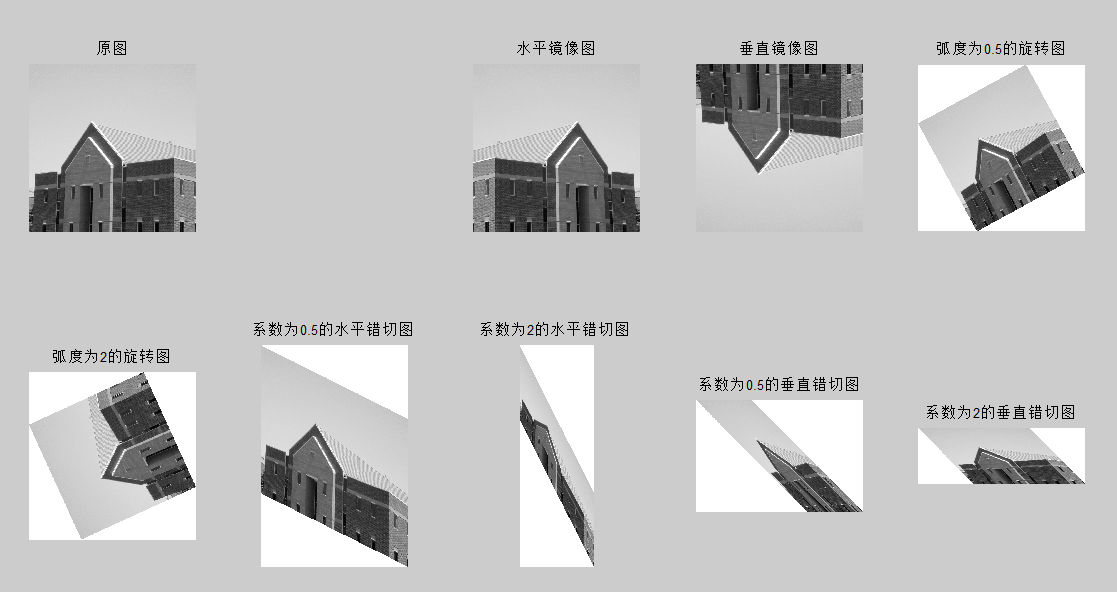
title('系数为0.5的垂直错切图');

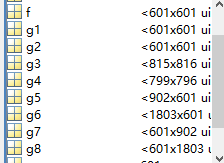
subplot(rows, cols, 10);

imshow(g8);

title('系数为2的垂直错切图');

测试结果：

图片大小被自动处理了，看起来宽高和预料的不一致，但观察workspace中的变量可知行列是正确的：



g1,g2是镜像，行列不变；g3,g4旋转，行列不规则变化；g5,g6垂直错切列不变；g7,g8水平错切行不变。