Local histogram-equalization

LocalHistEqual

**input**

Image\* origImg;

int divNum; // 分割数,一般为1，2，3，4，5

**output**

Image\* qualizedImg;

/////////////////////////////////////////////////////////////

int ww = origImg -> imgWidth / divNum；

int wh = origImg -> imgHeight / divNum；

对指定图像origImg以size为ww×wh 的window为单位进行直方图均衡化处理。

但均衡化window之间存在上下及左右方向上的1/2重叠，重叠区域中的pixel的灰度值取平均。



可认为origImg被宽度为ww/2, 高为wh/2的格子分割-----见上面的示意图中的红线。

上面的示意图中的诸红线的交点，对应一个均衡化window的左上角。

但对于右方或下方不足ww，wh的红线的交点，不对应一个均衡化window。

--------------------------------------------------------------------‘

每个均衡化window对应一个均衡化kernel (stream专用).

各个kernel的参数为

**input**

Image\* origImg;

int startX, endX; // (startX, startY): 红线的交点坐标

int startY, endY; // endX = startX+ ww; endY = startY+ wh;

float weight; // 0 <= weight <=1

**output**

int\* cumuCounter; // GPGPU global memory, 与origImg同dimension size.

int\* cumuGray; // GPGPU global memory, 与origImg同dimension size.

各个window的均衡化结果累加于cumuGray，同时 ++cumuCounter;

-----------------------------------------------------------------:

对上述各个专用stream同期后，

执行一个计算最后结果的kernel ------ thread dimension 与origImg 同.

**input**

int\* cumuCounter; // GPGPU global memory, 与origImg同dimension size.

int\* cumuGray; // GPGPU global memory, 与origImg同dimension size.

unsigned char t0;

float c1 = 255/ln(250/(t0+1));

float c2 = ln(t0+1);

**output**

unsigned char\* equalizedImgData; // GPGPU global memory, 与origImg同dimension size.

各个thread计算,

unsigned char g = origImg[y][x];

if g**≦**t0

else

if g**≧**250

else

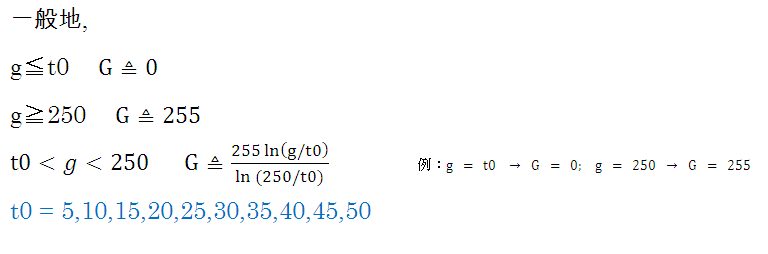
c1\*(ln(g) –c2);

equalizedImgData[y][x] = (unsigned char) ( weight\*(cumuGray[y][x] / cumuCounter[y][x] - G) + G);

qualizedImg -> imgData = equalizedImgData;

将origImg中的其他member值copy到equalizedImg中相应的field上。

图像值log化备考



||||||||||||||||||||||| END ||||||||||||||||||||||||||||||||