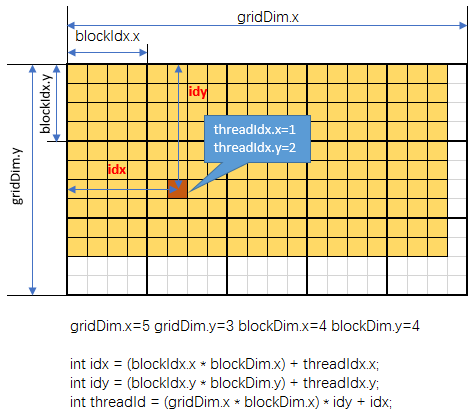
# 使用CUDA实现图像RGB到GRAY的转换

## 说明

这里使用CUDA来对图像数据进行处理，线程块上分布二维数据，这意味着需要两个线程索引，这样我们才可以用二维的方式访问图像数据：



以下是对一些概念的解释：

gridDim.x 线程网络X维度上线程块的数量

gridDim.y 线程网络Y维度上线程块的数量

blockDim.x 一个线程块X维度上的线程数量

blockDim.y 一个线程块Y维度上线程的数量

threadIdx.x 线程块X维度上的线程索引

threadIdx.y 线程块Y维度上的线程索引

使用网格的数量及划分是是在调用核函数时设置的，以下面的例子是所用到的图像为例，图像大小为557×313，如果每个block中的线程数量为16×16（即blockSize的值）则需要的block数量为35×20（即为gridSize的值）。

Dim3是CUDA中一个比较特殊的数据结构，我们可以用这个数据结构创建一个二维的线程块与线程网络。

## 2.代码

|  |
| --- |
| #include "cuda\_runtime.h"  #include "device\_launch\_parameters.h"  #include <opencv2\opencv.hpp>  #include <stdio.h>  using namespace std;  using namespace cv;  \_\_global\_\_ void rgba2gray(uchar3\* rgba\_data, uchar\* gray\_data, int rows, int cols)  {  int idx = (blockIdx.x \* blockDim.x) + threadIdx.x;  int idy = (blockIdx.y \* blockDim.y) + threadIdx.y;  int threadId = (gridDim.x \* blockDim.x) \* idy + idx;  if (threadId < rows \* cols)  {  uchar R = rgba\_data[threadId].x;  uchar G = rgba\_data[threadId].y;  uchar B = rgba\_data[threadId].z;  gray\_data[threadId] = .229f \* R + .587f \* G + .114f \* B;  }  }  int main()  {  //载入图像  Mat image\_rgb = imread("cinque\_terre\_small.jpg");  //定义灰度图像  Mat image\_gray = Mat(image\_rgb.size(), CV\_8UC1);  //像素的数量  const size\_t pix\_num = image\_rgb.rows \* image\_rgb.cols;  //获取图像数据指针  uchar3\* h\_rgb\_data = (uchar3\*)image\_rgb.ptr<uchar>(0);  uchar\* h\_gray\_data = image\_gray.ptr<uchar>(0);  //准备设备数据  uchar3\* d\_rgba\_data;  uchar\* d\_gray\_data;  cudaMalloc(&d\_rgba\_data, pix\_num \* sizeof(uchar3));  cudaMalloc(&d\_gray\_data, pix\_num \* sizeof(uchar));  cudaMemcpy(d\_rgba\_data, h\_rgb\_data, pix\_num \* sizeof(uchar3), cudaMemcpyHostToDevice);  //计算  int thread = 16;  const dim3 blockSize(thread, thread);  const dim3 gridSize((image\_rgb.cols + thread - 1) / thread, (image\_rgb.rows + thread - 1) / thread);  rgba2gray <<<gridSize, blockSize >>>(d\_rgba\_data, d\_gray\_data, image\_rgb.rows, image\_rgb.cols);  cudaDeviceSynchronize();  cudaMemcpy(h\_gray\_data, d\_gray\_data, sizeof(unsigned char) \* pix\_num, cudaMemcpyDeviceToHost);  cudaDeviceReset();  imshow("color", image\_rgb);  if (image\_gray.data)  {  imshow("gray", image\_gray);  }  waitKey(0);  return 0;  } |