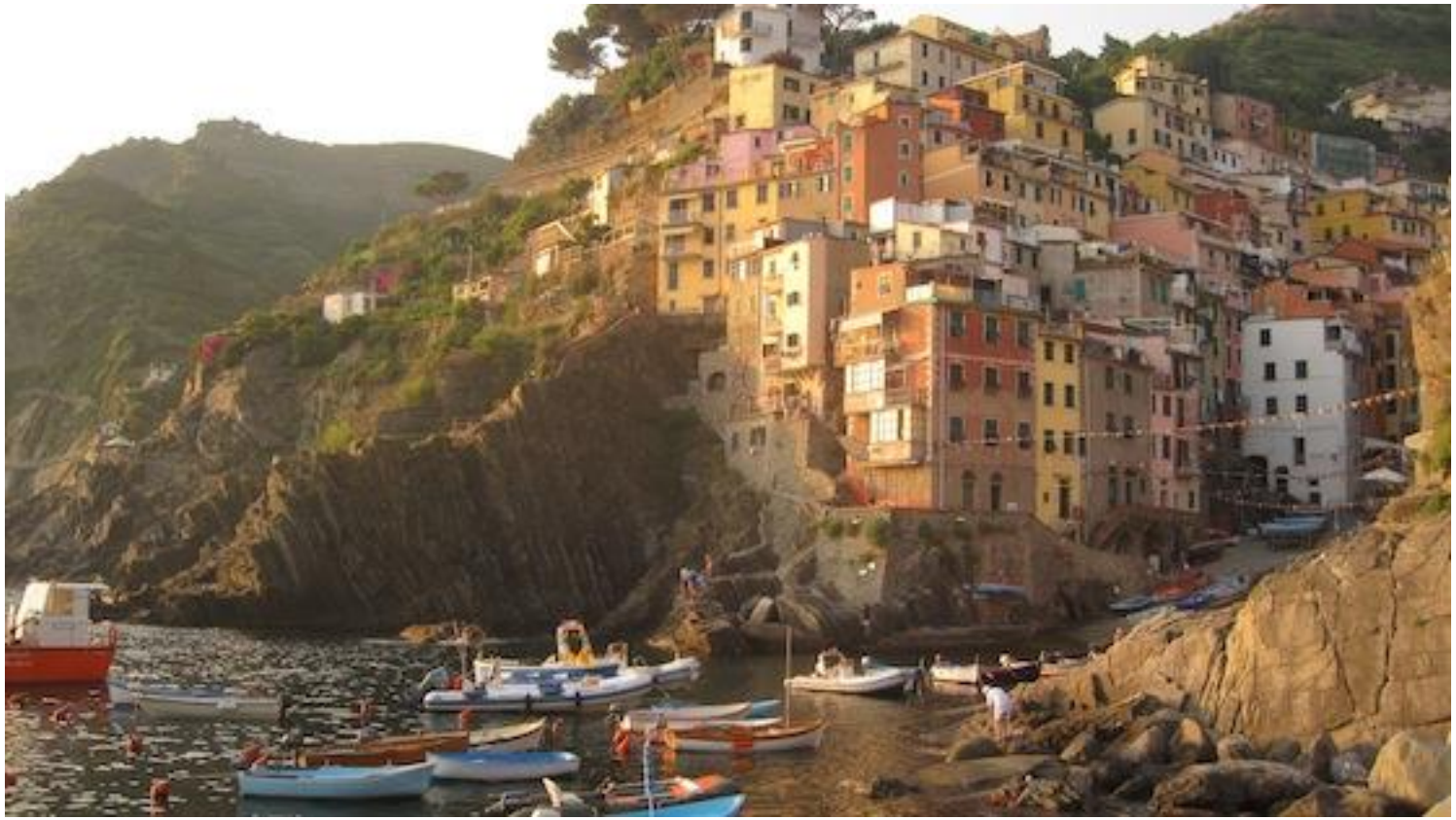


GPU并行计算与CUDA编程 第5课

第一版 讲师 罗韵（WeChat：LaurenLuoYun）

本周介绍内容

- 结合之前的知识，讲解两个图像处理的案例：
- 1. 并行化实现图像的RGB转灰度图
- 2. 并行化实现图像的均值模糊处理

实践案例1：并行化实现图像的RGB转灰度图

代码讲解

- 灰度图的求解公式：

```
const unsigned char R = rgbaImage[threadId].x;
const unsigned char G = rgbaImage[threadId].y;
const unsigned char B = rgbaImage[threadId].z;
greyImage[threadId] = .299f * R + .587f * G + .114f * B;
```

图像预处理函数

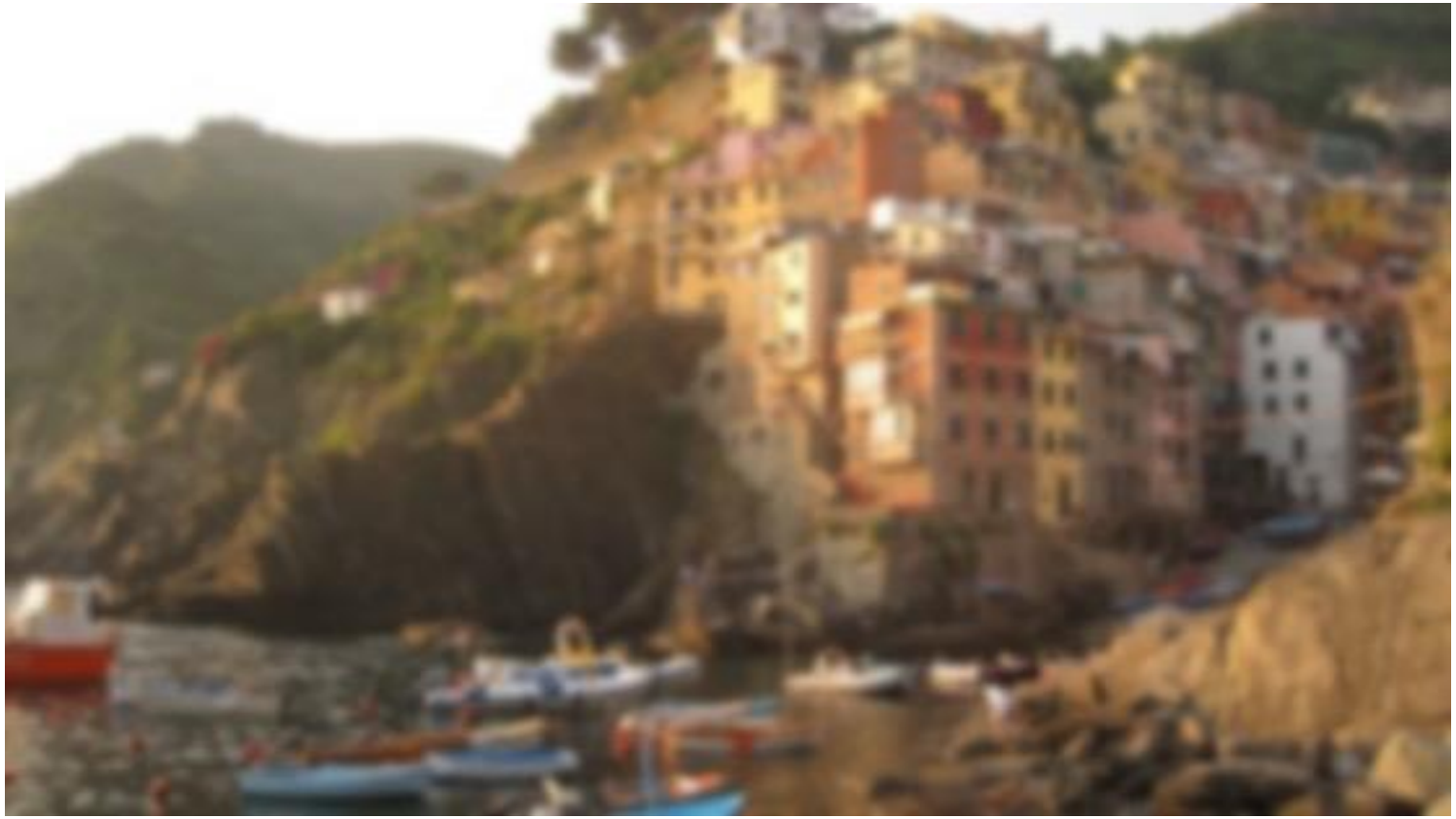

```
void preProcess(uchar4 **inputImage, unsigned char **greyImage,
                uchar4 **d_rgbaImage, unsigned char **d_greyImage,
                const std::string &filename) {
  //make sure the context initializes ok
  checkCudaErrors(cudaFree(0));

  cv::Mat image;
  image = cv::imread(filename.c_str(), CV_LOAD_IMAGE_COLOR);
  if (image.empty()) {
    std::cerr << "Couldn't open file: " << filename << std::endl;
    exit(1);
  }

  cv::cvtColor(image, imageRGBA, CV_BGR2RGBA);

  //allocate memory for the output
  imageGrey.create(image.rows, image.cols, CV_8UC1);

  //This shouldn't ever happen given the way the images are created
  //at least based upon my limited understanding of OpenCV, but better to check
  if (!imageRGBA.isContinuous() || !imageGrey.isContinuous()) {
    std::cerr << "Images aren't continuous!! Exiting." << std::endl;
    exit(1);
  }

  *inputImage = (uchar4 *)imageRGBA.ptr<unsigned char>(0);
  *greyImage  = imageGrey.ptr<unsigned char>(0);

  const size_t numPixels = numRows() * numCols();
  //allocate memory on the device for both input and output
  checkCudaErrors(cudaMalloc(d_rgbaImage, sizeof(uchar4) * numPixels));
  checkCudaErrors(cudaMalloc(d_greyImage, sizeof(unsigned char) * numPixels));
  checkCudaErrors(cudaMemset(*d_greyImage, 0, numPixels * sizeof(unsigned char))); //make sure no memory is left
      laying around

  //copy input array to the GPU
  checkCudaErrors(cudaMemcpy(*d_rgbaImage, *inputImage, sizeof(uchar4) * numPixels, cudaMemcpyHostToDevice));

  d_rgbaImage__ = *d_rgbaImage;
  d_greyImage__ = *d_greyImage;
}
```

并行化处理Kernel

```
int thread = 16;
int grid = (numRows()*numCols() + thread - 1)/ (thread * thread);
const dim3 blockSize(thread, thread);
const dim3 gridSize(grid);
rgba_to_greyscale<<<gridSize, blockSize>>>(d_rgbaImage, d_greyImage, numRows(), numCols());
```

```
__global__
void rgba_to_greyscale(const uchar4* const rgbaImage,unsigned char* const greyImage,int numRows, int numCols){
    int threadId = blockIdx.x * blockDim.x * blockDim.y + threadIdx.y * blockDim.x + threadIdx.x;
    if (threadId <  numRows * numCols){
            const unsigned char R = rgbaImage[threadId].x;
            const unsigned char G = rgbaImage[threadId].y;
            const unsigned char B = rgbaImage[threadId].z;
            greyImage[threadId] = .299f * R + .587f * G + .114f * B;
    }
}
```

实践案例2：并行化实现图像的均值模糊处理



代码讲解

- 均值模糊的求解公式：

```
Array of weights:

 0.0  0.2  0.0
 0.2  0.2  0.2
 0.0  0.2  0.0

Image (note that we align the array of weights to the center of the box):

   1  2  5  2  0  3
      -------
   3 |2  5  1| 6  0       0.0*2 + 0.2*5 + 0.0*1 +
     |       |
   4 |3  6  2| 1  4   ->  0.2*3 + 0.2*6 + 0.2*2 +   ->  3.2
     |       |
   0 |4  0  3| 4  2       0.0*4 + 0.2*0 + 0.0*3
      -------
   9  6  5  0  3  9

        (1)                        (2)                (3)
```

- 步骤： 拆分三个通道—>每个通道分别做模糊处理—> 重新合并起来
- 代码讲解

本周作业

- 使用CUDA并行化方法实现图像的水平翻转。（图片素材使用课程提供的图片素材即可）
- （如果对图像处理毫无入门的同学，可以自行模拟三组数据做并行化映射处理。）

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间