2016-3-20

Cecil Wang

Microsoft

HOMEWORK3

目录

[一、作业描述 2](#_Toc446281314)

[二、完成过程 2](#_Toc446281315)

[2.1 Nearest最邻近插值法 2](#_Toc446281316)

[2.2 Bilinear双线性插值法 2](#_Toc446281317)

[2.3 多线程编程 3](#_Toc446281318)

[2.3.1 parfor 3](#_Toc446281319)

[2.3.2 GPU 3](#_Toc446281320)

[三、实验结果 4](#_Toc446281321)

[四、灯谜 4](#_Toc446281322)

# 一、作业描述

Write your own imresize() function code to simulate the matab function imresize(). You should implement at least the ‘nearest’ and the ‘bilinear’ methods. Compare you result with the matlab function imresize().

# 二、完成过程

## 2.1 Nearest最邻近插值法

在做resize操作的时候，我才用的方式是使用生成图像的坐标反推原图像对应的坐标，这样做的好处是，当resize系数小于1的时候操作量会相对小一点。这样假设的基础是： resize系数大于1的时候，通常会认为图像失真严重，故更优先考虑resize系数小于1的情况。我们通过以下公式：

相应程序如下：

1 for i = 1 : height

2 x = max(fix(i \* fscale), 1);

3 for j = 1 : width

4 y = max(fix(j \* fscale), 1);

5 output(i, j, :) = input(x, y, :);

6 end

7 end

## 2.2 Bilinear双线性插值法

与nearest做法相似，我们同样适用生成图像的坐标反推原图像对应的坐标，坐标映射公式同nearest的公式。为了方便，我们保证srcX 与srcY的值为double类型，由此可得到如下公式

相应的周围四个点坐标为

生成图像的像素值计算公式如下

## 2.3 多线程编程

MATLAB中提供了多种多线程编程方式，在此我尝试了其中两张方式，一是parfor，二是对gpuArray使用arrayfun。

### 2.3.1 parfor

parfor与for用法别无二致，语法如下

**parfor loopvar = initval:endval; statements; end**

唯一需要注意的是，parfor并不支持循环嵌套。此外，当循环次数很小时并不提倡使用parfor，因为其本身存在操作的损耗，并不能起到加速的作用，反而会拖慢程序。

### 2.3.2 GPU

首先需要介绍的时候workspace与GPU间的数据交互

1 Agpu = gpuArray(a);

2 output = gather(outputgpu);

其中gpuArray是将a传入GPU的显存中，gather将GPU的显存中的数据加载到workspace中。

MATLAB提供了多种只用GPU的方式，尤其对于NVIDIA的显卡，MATLAB提供了其自身代码、C/C++、CUDA的混合编程模式，可以最大化资源利用率。同时MATLAB也提供了一些简单的GPU调用方式：

* 内建函数(build-in)，例如fft等函数，可以直接对GPU上的数据进行操作
* arrayfun。arrayfun函数会自动检测数据是在workspace还是GPU上，由此决定是使用CPU还是GPU

在此我们主要关注于arrayfun。

workspace和GPU上的数据都可以使用arrayfun函数。其操作是对每一个输入变量执行相同的操作，这是MATLAB所提倡的向量化操作方式的精髓所在。其语法如下：

**[B1,...,Bm] = arrayfun(func,A1,...,An,Name,Value)**

其中func是函数句柄，我们可以将其看作为c++中的函数指针。A1-An为传入参数，B1-Bm为输出参数。需要特殊注意的是，当arrayfun对workspace中的数据进行操作的时候，是可以访问函数外的数据（即非传入参数），但是对GPU中的数据进行操作的时候只能使用传入参数。

对于bilinear函数有如下代码

1 Agpu = gpuArray(a);

2 Bgpu = gpuArray(b);

3 Ugpu = gpuArray(u);

4 Dgpu = gpuArray(d);

5 Lgpu = gpuArray(l);

6 Rgpu = gpuArray(r);

7

8 f = @(a,b,u,d,l,r)(1-a) \* (1-b) \* u + ...

9 (1-a) \* b \* d + ...

10 a \* (1-b) \* l + ...

11 a \* b \* r;

12 outputgpu = arrayfun(f,Agpu, Bgpu, Ugpu, Dgpu, Lgpu, Rgpu);

13

14 output = gather(outputgpu);

其中Agpu、Bgpu、Ugpu、Dgpu、Lgpu、Rgpu是提前预处理的数据，正是由于arrayfun的局限性才导致需要这些冗余操作。

# 三、实验结果



从左到右依次是原图、nearest、bilinear。

# 四、灯谜

像素