2016-3-20

HOMEWORK3

Cecil Wang
MICROSOFT

目录

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：

$$\mathrm{srcY} = \mathrm{dstY} * (width_{src} / width_{dst})$$

$$\mathrm{srcX} = \mathrm{dstX} * (height_{src} / height_{dst})$$

相应程序如下：

```
for i = 1 : height
    x = max(fix(i * fscale), 1);
    for j = 1 : width
        y = max(fix(j * fscale), 1);
        output(i, j, :) = input(x, y, :);
    end
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

$$\mathrm{minX} = \max(\mathrm{floor}(\mathrm{srcX}), 1)$$

$$\mathrm{maxX} = \min(\mathrm{ceil}(\mathrm{srcX}), height_{src})$$

$$\mathrm{minY} = \max(\mathrm{floor}(\mathrm{srcY}), 1)$$

$$\mathrm{maxY} = \min(\mathrm{ceil}(\mathrm{srcY}), width_{src})$$

$$\mathrm{a} = abs(\ srcY - minY\ )$$

$$\mathrm{b} = abs(\ srcX - minX\ )$$

相应的周围四个点坐标为

$$(\mathrm{minX}, \mathrm{minY})\ (minX, maxY)\ (maxX, minY)\ (maxX, maxY)$$

生成图像的像素值计算公式如下

$$(1-\mathrm{a})(1-\mathrm{b})(\mathrm{minX}, \mathrm{minY}) +$$
$$(1-\mathrm{a})\mathrm{b}(\mathrm{maxX}, \mathrm{minY}) +$$
$$\mathrm{a}(1-\mathrm{b})(\mathrm{minX}, \mathrm{maxY}) +$$
$$\mathrm{ab}(\mathrm{maxX}, \mathrm{maxY})$$

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

```
Agpu = gpuArray(a);
Bgpu = gpuArray(b);
Ugpu = gpuArray(u);
Dgpu = gpuArray(d);
Lgpu = gpuArray(l);
Rgpu = gpuArray(r);

f = @(a,b,u,d,l,r)(1-a) * (1-b) * u + ...
                  (1-a) * b * d + ...
                  a * (1-b) * l + ...
                  a * b * r;
outputgpu = arrayfun(f,Agpu, Bgpu, Ugpu, Dgpu, Lgpu, Rgpu);

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