### Intel® Threading Building Blocks (Intel® TBB)功能介绍与实用

同济大学 2151136 朱开来

Intel® Threading Building Blocks (Intel® TBB) 是一个用于并行编程的开源 C++ 库，它提供了一系列功能和组件，用于简化并行任务的创建和执行。首先，我们来看一下Intel® TBB里面有哪些重要的组成部分：Tasks（任务）：并行计算的基本单元，代表要执行的并行操作；

Parallel Algorithms（并行算法）：用于高效处理数据的一系列并行算法，如排序、循环、归约和扫描等；Synchronization Primitives（同步原语）：用于管理线程间同步和互斥访问的工具，如互斥锁、条件变量和原子操作；Memory Allocator（内存分配器）：提供并行化的内存分配和释放操作，提高多线程环境下的性能；Task Scheduler Extensions（任务调度器扩展）：允许开发人员自定义和扩展任务调度器的行为，优化调度策略和资源分配。这些组件共同构成了 Intel® TBB 的功能，帮助开发人员简化并行任务的创建和执行，提高多核处理器的并行计算性能。

Intel® TBB的主要工作流程如下：创建任务、任务调度、并行执行、同步和互斥、结果汇总。具体来讲，首先开发人员使用 Intel® TBB 提供的接口创建任务，并将任务提交给任务调度器。接着任务调度器接收到提交的任务后，根据任务的依赖关系和可用的处理资源，动态地将任务分配给可用的线程池中的线程，线程池中的线程并行执行各自被分配的任务，利用多核处理器的并行计算能力。同时在任务执行过程中，使用同步原语（如互斥锁和条件变量）来管理线程之间的同步和互斥访问，确保数据的一致性和正确性。最后当所有任务完成执行后，可以将结果进行汇总或合并，以获得最终的计算结果。  
 图像特征提取在计算机视觉和图像处理领域中具有重要性，它是实现高级图像分析和理解的关键步骤，平常的算法对于图像特征提取常常具有速度较慢，效率较低的缺点，我们可以使用Intel® TBB实现并行操作，从而加速图像特征提取，能够取得更高的效率。以下是使用Intel® TBB来简单实现图像特征提取的代码：

#include <iostream>

#include <vector>

#include <tbb/tbb.h>

// 图像处理函数：边缘检测

void imageProcessing(const std::vector<int>& image, std::vector<int>& processedImage)

{

// 获取图像尺寸

int width = 512; // 假设图像宽度为512

int height = image.size() / width;

// 边缘检测算法

for (int y = 1; y < height - 1; ++y)

{

for (int x = 1; x < width - 1; ++x)

{

// 计算梯度

int gx = image[(y \* width + x + 1)] - image[(y \* width + x - 1)];

int gy = image[((y + 1) \* width + x)] - image[((y - 1) \* width + x)];

// 计算边缘强度

int gradient = std::sqrt(gx \* gx + gy \* gy);

// 将结果存储到处理后的图像中

processedImage[(y \* width + x)] = gradient;

}

}

}

// 图像特征提取函数

void imageFeatureExtraction(const std::vector<int>& processedImage, std::vector<int>& features)

{

// 定义图像特征提取任务类

class ImageFeatureExtractionTask

{

private:

const std::vector<int>& processedImage;

std::vector<int>& features;

public:

void operator()(const tbb::blocked\_range<int>& range) const

{

for (int i = range.begin(); i != range.end(); ++i)

{

// 执行图像特征提取操作

// 示例中只是将处理后的图像复制到特征向量中

features[i] = processedImage[i];

}

}

ImageFeatureExtractionTask(const std::vector<int>& procImg, std::vector<int>& feat)

: processedImage(procImg), features(feat) {}

};

// 创建任务调度器

tbb::task\_scheduler\_init init;

// 定义图像和特征向量的大小

int imageSize = processedImage.size();

features.resize(imageSize);

// 创建图像特征提取任务

ImageFeatureExtractionTask task(processedImage, features);

// 并行执行图像特征提取任务

tbb::parallel\_for(tbb::blocked\_range<int>(0, imageSize), task);

}

int main() {

// 读取图像数据

std::vector<int> image; // 假设已经读取了图像数据

// 创建处理后的图像

std::vector<int> processedImage;

processedImage.resize(image.size());

// 执行图像处理

imageProcessing(image, processedImage);

// 创建特征向量

std::vector<int> features;

features.resize(processedImage.size());

// 执行图像特征提取

imageFeatureExtraction(processedImage, features);

// 处理提取后的特征向量

return 0;

}