**数据结构与算法课程实验报告**

|  |  |
| --- | --- |
| **实验名称：** | **四叉树自适应模糊** |

|  |  |
| --- | --- |
| **学 员 姓 名：** |  |
| **学 号：** |  |
| **年 级：** | **大二** |
| **学 员 队：** | **计算机五队** |

摘　要

本实验旨在通过建立四叉树对图像进行分割，实现图像的模糊处理。首先，通过将原始图像分割成小块，每个小块代表四叉树的一个节点，构建了整个图像的四叉树结构。然后，在四叉树中对每个节点进行模糊处理，可以采用均值模糊等方法，处理可以在节点的像素级别或节点整体级别进行。通过递归处理整个树，从根节点开始逐层对每个子节点进行模糊处理，直到达到预定的处理层次或条件。最后，通过将处理过的四叉树重新合并为图像，得到模糊处理后的结果。

在建立四叉树的过程中，对于图像建立四叉树的条件，引入了成员函数判断四叉树是否需要继续建立新节点细化，即通过判断结点内像素的颜色方差是否满足一定条件。如果满足条件，则不再细化，并标记该结点的像素可进行模糊处理；否则，继续分割成四个象限，生成新节点，继续判断。

对颜色相近的区域进行模糊处理时，采用了简单的均值模糊方法，遍历结点的像素的颜色值取平均值，并进行多次高斯模糊处理，以使得像素点更加平滑。实验的整体思路涉及图像分割、模糊处理、递归处理和图像重建等步骤，通过这些步骤完成了对图像的模糊处理。

第1章　实验原理

1.1　实验内容

1.对图像建立四叉树

2.对于输入的图像，四叉树能够输出模糊的结果

3.对颜色相近的区域进行模糊

1.2　算法思路

1.图像分割： 将原始图像分割成小块，每个小块代表四叉树的一个节点。这样，整个图像可以表示为一个四叉树结构，其中每个节点对应于图像中的一个区域。

2.模糊处理： 对四叉树中的每个节点进行模糊处理。可以采用均值模糊等。模糊处理可以在节点的像素级别或节点整体级别进行，具体取决于所选的算法。

3.递归处理： 由于四叉树的层次结构，可以采用递归的方式处理整个树。从树的根节点开始，递归地对每个子节点进行模糊处理，直到达到预定的处理层次或条件。

4.重建图像： 处理完成后，可以通过将处理过的四叉树重新合并为图像，从而得到模糊处理后的图像。

1.2.1　针对问题1 对图像建立四叉树

四叉树是一种树状数据结构，它将空间分割成四个象限，每个象限又可以继续分割成四个象限，以此类推，形成一个树状结构。四叉树通常用于对二维空间进行分割和索引。

对于图像建立四叉树，是把原始像素分割成四个象限，其中要保存结点(其实是像素点的集合)的深度(depth)，位置信息(Pos)，和对四个象限的指针(ru, lu, ld, rd)，结点的长宽信息（height, width），还有此像素的颜色信息(rgb)。

建立四叉树，对于结点应该给出两种初始化的方法，一种是通过颜色信息和长宽信息来建立，以方便从原始图像到根节点的建立，另一种是通过结点位置和长宽信息来建立，以方便四叉树的深入细化。

1.2.2　针对问题2 对于输入的图像，四叉树能够输出模糊的结果

对于输入的图像输出模糊结果，就要在四叉树的建立新结点的条件上加以限制，要用成员函数判断四叉树需不需要继续建立新节点细化(IsMakeNewNode)，当结点内像素的rgb值的方差满足一定条件(tolerance)时则不再细化，并给出一个信号(valid)，来确定这个结点的像素可以进行模糊处理了。如果不满足条件则继续细化，也就是继续分成四个象限，生成四个新节点，继续判断。

1.2.3　针对问题3 对颜色相近的区域进行模糊

对于像素的模糊处理，这里使用简单的均值模糊，也就是遍历结点的像素的rgb值取其平均赋给整个结点的每个像素。

为了使像素点更平滑，在像素被四叉树处理后，整体使用核为3\*3的高斯模糊处理了二十遍。

第2章　实验结果

2.1　实验数据及实验设计

实验测试了tolerance为10，20，30，40，50，60的情况，可以看到图像因tolerance取值不同模糊程度也不一样。

2.2　实验结果

四叉树处理后的图。

tolerance=10



tolerance=20



tolerance=30



tolerance=40



tolerance=50



Tolerance=60

形状, 正方形

描述已自动生成

tolerance=30时用高斯模糊处理后对比，右图为高斯处理后的，可见像素点变得平滑了

图片包含 动物, 螃蟹, 食物, 桌子

描述已自动生成

2.2　实验结果分析

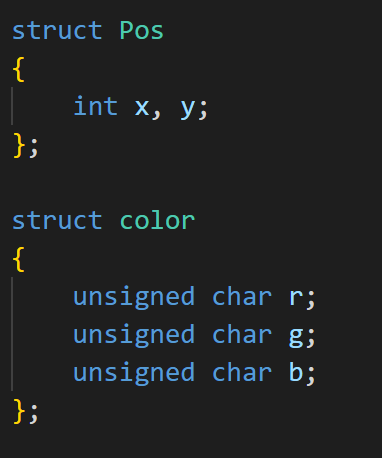
由图像可以看见，在像素信息更多，也就是像素点rgb值方差更大的地方，如螃蟹腿，螃蟹壳的边缘和地板和墙的交界处，四叉树分的比较细，在像素信息较少的地方，比如地板和墙壁和猫咪的背部，四叉树分的比较粗略，马赛克点更大。可见四叉树能够根据像素点信息的多少来进行模糊。

而tolerance取值不同，即对于像素点间方差大小的容忍程度不同，四叉树的细致程度也不同。容忍程度大，分的就更加粗略，模糊程度也更大，容忍程度小，就分的细致，模糊程度较小。

第3章　源代码解析

主要代码在头文件QuadTreeNode.h里面，设计并实现了一个图像处理的类 QuadTreeNode，该类用于对图像进行分块处理和模糊算法操作，还有将四叉树转换成图片，四叉树扩展，对四叉树处理后的图像高斯模糊的三个函数。

首先用结构体，定义了结点的位置和颜色数据。



然后是四叉树结点的类：

文本

描述已自动生成

后面图像处理还包括这三个函数，分别进行了从四叉树到图像的转换和四叉树的扩展，和最终的高斯模糊。

图形用户界面, 文本

描述已自动生成

3.1

QuadTreeNode类中包含以下成员：

文本

描述已自动生成

成员变量：

QuadTreeNode \*ru, \*lu, \*ld, \*rd;：分别表示四个象限的子节点。

Pos position;：当前节点在图像中的位置。

int height, width;：当前节点的高度和宽度。

bool valid;：标志当前节点是否有效。

color \*\*rgbs;：保存图像信息的二维数组。

int depth;：当前节点的深度。

成员函数：

QuadTreeNode(color \*\*r, int wd, int ht);：构造函数，用于根据给定的图像信息、宽度和高度创建节点。

屏幕上有字

描述已自动生成

QuadTreeNode(int posx, int posy, int wd, int ht);：构造函数，用于根据给定的位置、宽度和高度创建节点。

文本

描述已自动生成

bool IsMakeNewNode(int Tolerance);：判断是否需要继续细化节点，根据给定的容差值判断。遍历节点算出均值，然后对方差进行计算。

再将rgb值的方差与tolerance进行比较。进行判断。

如果方差大于tolerance的平方就继续细化，如果方差小于tolerance的平方就不再细化建立新结点。

文本

描述已自动生成

void Fuzzify();：对节点进行模糊化操作。

就是直接遍历节点内的所有像素点算出均值再遍历给像素点赋予均值。并且给valid赋true，方便后面从四叉树转回图像时判断是否停止往四叉树深处去。

屏幕上有字

描述已自动生成

3.2

在此头文件里的其他三个函数功能如下：

void TreeToImage(QuadTreeNode \*rt, color \*\*img);

看是否已经到了深度最大的结点。是则把四叉树里的像素值信息(rt->rgbs)赋给图片的像素(img)，否则递归调用遍历四叉树的四个子结点。

文本

描述已自动生成

void TreeExtend(QuadTreeNode \*rt, int Tolerance);

这个是对四叉树进行结点扩展的操作函数。

判断条件是四叉树当前结点满足IsMakeNewNode，也就是当前结点的像素的方差值不太大了，是可以容忍的（tolerance）,或者当前结点的长宽小于10个像素(可以减少遍历时间)。

如果判断为true则对该结点进行模糊操作。

否则继续细分结点。这里以第一象限（upperright）为例：

文本

描述已自动生成

void Gauss(QuadTreeNode \*rt);

这里用一个核为3\*3的一个已经归一化了的高斯矩阵（直接用的结论）来对整个图像进行处理。使得模糊后的图像的过度自然一些。

图片包含 文本

描述已自动生成

Quadimg\_todo.cpp文件里就加了几句函数调用语句，然后为img随color一起分配了空间。

文本

描述已自动生成