第一次编程作业

胡炳宇 22307140007
2024.3.14

实验目的：

采用C++编程，读取一个高清图像文件“input1.bmp”，针对给定目标模板“input2.bmp”在原图中进行检索查找，并以最快的速度定位到原图中的最佳匹配坐标（按照匹配准确度高低排序），在本实验报告最后列出可能的坐标位置并求解其平均精度。

运行环境：

Windows 11系统，使用Visual Studio 2022，release编译运行。

实现思路

由于测试集以及input文件全部为.bmp(位图)文件，程序首先需要做的就是读取并保存BMP文件中的色彩数据。由于测试图片均为rgb图片，故而只需要能够处理24位的bmp文件。幸运的是.bmp文件内部存储的色彩信息（灰度图，RGB或ARGB）直接以二进制的形式暴露在外，十分方便借助计算机软件进行简单或深入的分析。

根据数字图像处理的知识，BMP格式分为位图文件头域，位图参数头域，调色板域和位图数据域。进一步讲我们日常生活中遇到的 .bmp 格式图片的文件头长度绝大多数都是 54 字节，其中包括 14 字节的 Bitmap 文件头以及 40 字节的 DIB (Device Independent Bitmap) 数据头，或称位图信息数据头（BItmap Information Header），其中包含位图参数头域和调色板域，之后的数据就是实验需要的以数组形式排列的所有像素的色彩信息。值得注意的是，整个 BMP 文件中，只有此处使用大端（big endian），色彩信息是bgr(具体的BMP细节见参考链接（1))。

得到BMP图片后可以开始进行匹配，观察得到目标图片是在原图模糊和缩小得到的，故而需要将输入的原图进行一定的模糊并按照一定的宽高比例压缩。这样之后才能进行像素点级别的匹配并最终在原图中输出目标图片的坐标。对图像的模糊采用高斯模糊进行。所谓模糊，可以理解成每一个像素都取周边像素的平均值，在取每个点的权重时，正态分布显然是一种可取的权重分配模式。这就是

高斯模糊（详见参考链接（2））。而对图像的缩小则是采用双线性插值法进行。即利用待求像素四个相邻像素的灰度，按照其距内插点的距离赋予不同的权重，进行线性内插（详见参考链接（3））。

在进行像素即匹配时采用的方法则是考虑两幅图的相关系数，也即 NCC 归一化互相关.利用公式 $\rho_{f,t}=\frac{Cov(f,t)}{\sqrt{Var(f)Var(t)}}=\frac{1}{n}\sum_{x,y}\frac{1}{\sigma_f\sigma_t}(f(x,y)-\mu_f)(t(x,y)-\mu_t)$ 计算目标图片与当前区域的相关性，取值范围[-1，1],其中-1 表示负相关，0 表示不相关，1 表示相关。只要遍历缩小后的图片，比较相关度，选取相关度最高的位置则大概率是截取图片的位置。

代码实现

首先是读取.bmp 文件：定义结构体 typedef struct tagBITMAPFILEHEADER，typedef struct tagBITMAPINFOHEADER，分别为位图文件头域和位图数据信息域。由于它们的大小固定，所以利用 C++中不同大小的数据类型表示它们如 unsigned short 表示两个字节，unsigned long 表示四个字节。这样可以完全保存 BMP 54 个字节的文件头。之后，定义* pColorData 用来保存 bmp 中的全部像素色彩信息。在 imread 函数中，在读取过前 54 位的信息后，fread 将剩下的满足条件的信息全部存入了* pColorData 中。在这之后，为了方便计算设置一个二维向量 rgb2gray 来同时将 rgb 数据转存为灰度图像。同时为了计算 NCC 的方便，如果位图大小较小，直接对其计算均值和方差（实际为计算目标 16*16 图片的均值和方差）。

接下来实现高斯模糊也即计算高斯系数，利用二维平面的标准正态分布公式 $\mathrm{f}(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$ 并进行归一化即可得出。这里我采用$\sigma=3$ 以及高斯半径定义为 2；

之后实现双线性插值法放缩图像。同样，对每一个像素点，有 $f(x,y)=f(Q_{11})(x_2-x)(y_2-y)+f(Q_{21})(x-x_1)(y_2-y)+f(Q_{12})(x_2-x)(y-y_1)+f(Q_{22})(x-x_1)(y-y_1)$ 由此即可得到所有点缩小后的数据然后存入新的 BMP 数据中。

对于图像的真实值，使用 struct true_data 来保存，其中有真实值的起始 x,y 坐标（以图片左下角为坐标原点），还有真实值的对应高度和宽度。这些是用来计算之后的准确率匹配度和交并比（IoU）。

准备工作结束，进入 match 函数。首先需要确定出截图大概的缩小比例，而者将根据截图真实大小与当前大小的比例关系确定，由此我们可以得到一个大概的图片缩放范围。之后我们只需要在这个范围内遍历原图的宽高缩放比以期望获得最大的准确率匹配度。同时由于程序效率的限制，最终选定的缩放比遍历步长为 0.02 每次。此外，我们认为当准确率匹配度达到 90%时已经达到目的，所以此时遍历会直接结束。获得压缩过的图片后直接对缩小后的图片 change_bmp 进行 16*16 范围的遍历。为此都得出一个相关系数。只需要记录最接近 1 的相关系数出现时对应在 change_bmp 中的位置坐标（x , y）再根据宽高缩放比例就可以得到我们找到的目标图在原图中的起始位置（start_x , start_y）。这将支撑我们计算准确率匹配度和交并比。其中准确率匹配度即我们匹配到的目标图与它的真实值相交的面积除以真实值的面积，而交并比则是相交的面积除以两图的并的面积。二者取值范围都是[0，1]，而且越接近 1 匹配度越好。当然，在图形匹配中，失配时不可避免的，对于这些点，直接定义准确率匹配度为 0，同时交并比也自然为 0，这些点不纳入后续的考虑。

整体而言，本次实验过程为：读取.bmp 文件，获得色彩数据，对原图进行高斯模糊，按照一定宽高比缩放原图得到新图，计算新图各部分与缩小后截图的归一化相关系数，取相关性最好的点的坐标并转换到原图，最后求出匹配率准确度和交并比。

实验结果

在全部 101 张图片中，匹配成功（匹配率准确度>0)的数量为 82，成功率 81.2%，所得平均精度为 68.6%，最高的精度为 97.9%较为良好。在 release 编译后匹配全部 101 组图像总共用时 4 分钟,处理一组图像平均速度为 2.4 秒，最快速度为 0.123 秒。值得注意的是：test053 测试时间 68 秒，超过总时长的 1/4。原因为其缩放比例太大（0.3）导致像素级匹配计算数据极大，故而时间倍增。

最后以“input1.bmp”,“input2.bmp”为例，这时的原图压缩比例设定为：高 0.1±0.02，宽 0.15±0.02。但事实上，只有在压缩比例为高 0.1，宽 0.15 时，

此样例才能出现匹配率准确度才大于 0 且同时它达到了非常高的 94.4%。故而在 input1 的匹配中，平均匹配率准确度为 94.4%，且认为压缩比例（0.1，0.15）为实际压缩比例。可以得出，input2 在 input1 中的起始坐标是（531，715)。（以图像左下角为坐标原点）。

参考链接

参考链接（1)：浅谈图像格式 .bmp - 知乎 (zhihu.com)

参考链接（2)：高斯模糊(高斯滤波)原理以及计算过程 高斯滤波公式-CSDN博客

参考链接（3)：一篇文章为你讲透双线性插值 - 知乎 (zhihu.com)