机器视觉第八次作业

1、设计一种方法实现 Stereo Correspondence，编写伪代码描述整个过程。附加分析：如何提升算法性能（动态规划？）。

Stereo Correspondence 的目标是通过两个相机视角下的图像间的关系计算对应点之间的视差。这个问题的核心是通过计算图像中的每个像素在两幅图像中对应的位置来得到视差图。伪代码如下：

输入：

左图（leftImage）和右图（rightImage）

匹配窗口大小（blocksize）

最大视差值（maxdisparity）

损失函数（通常为平方误差或者绝对误差）

输出：

视差图（disparitymap）#图中每个像素代表左右图之间的视差，若值为 0 则表明该像素点在左右两张图片中的位置几乎没有偏移。

```
function computeDisparityMap(leftImage, rightImage, blockSize, maxDisparity):
    # 获取图像的高度和宽度
    height, width = getImageSize(leftImage)
    # 创建一个与左图尺寸相同的空视差图
    disparityMap = initializeEmptyMap(height, width)
    # 对左图的每个像素进行遍历
    for y = 0 to height-1:
        for x = 0 to width-1:
            # 定义当前块的边界
            leftBlock = getBlock(leftImage, x, y, blockSize)
            # 初始化最小误差和最佳视差值
            minError = infinity
            bestDisparity = 0
            # 在右图中搜索最大视差范围内的最佳匹配块
            for d = 0 to maxDisparity:
```

```
                # 检查当前匹配的右图区域
                rightX = x - d
                if rightX < 0:
                    continue
                rightBlock = getBlock(rightImage, rightX, y, blockSize)
                # 计算误差（可以选择不同的误差度量，假设使用平方误差）
                error = computeError(leftBlock, rightBlock)
                # 如果误差更小，更新最佳视差值
                if error < minError:
                    minError = error
                    bestDisparity = d
            # 将最佳视差值存入视差图
            disparityMap[y][x] = bestDisparity
    return disparityMap
```

因为对于一个像素点，需要搜索另一个图片中所有的像素点才能得到最终的视察，时间复杂度较高，所以可以采用动态规划的方式进行算法优化，具体的思路为：为每一行创建一个矩阵，矩阵的每一列表示当前像素的视差；当前像素的最佳视差可以通过前一个像素的最佳视差和相邻像素的视差计算。伪代码如下：

```
function computeDisparityMapWithDP(leftImage, rightImage, blockSize, maxDisparity):
    height, width = getImageSize(leftImage)
    #初始化输出矩阵
    disparityMap = initializeEmptyMap(height, width)
    #对左图中的每一行进行处理
    for y = 0 to height-1:
    # 创建一个动态规划矩阵 dp，其中每个元素 dp[x][d]代表当前位置 x 的视差 d 的最小误差
    # 宽度是图像的宽度，最大视差范围是 maxDisparity
        dp = initializeEmptyArray(width, maxDisparity)
        # 初始化第一列的误差值
        # 第一列没有前一个像素可以参考，因此只能直接计算误差
```

```
            for d = 0 to maxDisparity:
            dp[0][d]=computeError(getBlock(leftImage,0,y,blockSize),getBlock(rightImage, d,
y, blockSize))
            # 遍历行中的其他像素
            for x = 1 to width-1:
                for d = 0 to maxDisparity:
                    # 保存 d 的最小误差值
                    minError = infinity
# 考虑视差 d-1, d, d+1（即考虑相邻的视差值），通过动态规划利用相邻像素之间的关联性
来减少计算量
                    for dd = max(0, d-1) to min(maxDisparity, d+1):
                        #计算误差
                        error = computeError(getBlock(leftImage, x, y, blockSize),
getBlock(rightImage, x-dd, y, blockSize))
                        dp[x][d] = min(dp[x-1][dd] + error, dp[x][d])
            # 在行结束时选择最小误差的视差值
            for x = 0 to width-1:
                disparityMap[y][x] = getBestDisparity(dp[x])
        return disparityMap
```

2、阅读论文：Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, IEEE TPAMI 2001

论文介绍了一种基于图割的快速近似能量最小化方法用于解决计算机视觉中的像素标记问题，这些问题通常需要在保持对象边界处的锐利不连续性的同时使标签在大部分区域内平滑变化。文章提出了两种算法，分别是基于扩展移动和交换移动的图切割算法，它们能够同时改变任意大的像素集的标签，与仅改变单个像素标签的标准算法（如模拟退火）相比效率更高。这些算法能够处理包括保持不连续性能量在内的广泛能量函数，并在图像恢复、立体视觉和运动估计的实验中展示了其有效性，实现了高达 98%的准确率。

文章的主要创新点在于提出了两种基于图割的算法，这些算法能够高效地处理计算机视觉中的像素标记问题，特别是在需要同时保持标签的平滑性和在对象边界处的锐利不连续性时，这两种算法分别是扩展算法和交换算法。

扩展算法基于α-β-交换移动处理任意半度量 V，从而找到全局最小值的一个已知因子内的解。这种算法通过构建图并寻找最小割来实现，其中图的节点代表像素，边代表像素间的相互联系。这种方法不要求特定的度量函数形式，具有广泛的适用性并且能够保证找到的解接近全局最优。

而交换算法则基于α-扩展移动，要求 V 必须是度量，即满足非负性、对称性和三角不等式三个基本性质的函数。这种算法通过在图上执行最小割操作来找到局部最小能量解，特别适合于需要将一个区域的标签扩展到邻近区域的问题。

这种方法在计算机视觉领域有着广泛的应用。在图像恢复方面，这些算法能够处理图像噪声和退化问题。通过最小化能量函数重建出清晰的图像，对于提高图像质量、增强图像细节以及在各种成像系统中恢复图像原貌至关重要。在立体视觉领域，它们通过估计两个图像之间的视差来帮助重建场景的三维结构，这对于自动驾驶汽车的环境感知、机器人导航以及增强现实等领域具有重要意义。在运动估计方面，这些算法能够分析视频序列中相机或物体的运动，能够应用于视频稳定、运动补偿和行为分析等。图像分割是另一个重要的应用领域，图割算法能够将图像分割成不同的区域或对象，这在医学图像分析、卫星图像处理以及工业视觉检测中有着广泛的应用。此外，这些算法还可以用于目标识别、场景理解、图像编辑等更广泛的计算机视觉任务，它们能够提供像素级的精确标签分配，从而提高这些任务的准确性和效率。

在资料查找中，从 github 上发现了相关的开源代码，它主要包含了两个文件：graph_cut_performance.py 和 minimizaition.py，这两个文件实现了论文的算法，graph_cut_performance.py 创建了一个二维图，模拟了一个图像，其中每个像素点都是图中的一个节点。节点之间的边代表像素之间的相似性或差异性，权重随机生成。minimizaition.py 读取一个灰度图像，将其转换为像素值数组，然后通过图割算法进行能量最小化。其中若要运行该代码，需要将其中的 scipy.misc.toimage 函数进行更换，因为函数在 scipy 版本 1.2.0 中被弃用，并在

1.3.0 版本中被完全删除。所以我此处使用 opencv 进行了替换且需要将输入的矩阵转为 np 数组的格式，如下图所示。

```python
def arr_to_image(a, fname):
    # 将数组转换为图像并保存
    print(a)
    a = np.array(a, dtype=np.uint8)
    cv2.imwrite(fname, a)
    return 0
```

图 1 代码替换

随后，在安装好 requirement 中的依赖后可以输入图片运行代码，同时输入的图片需要控制像素数量才能短时间内得到结果。但是在测试的过程中，为了确保运算的速度，所以输入的图片是 28x28，所以从结果来看只是单纯的变为了灰度图。

```
PS W:\Codes\VsCode\VsCode_Storage\CV\homework\FAE\FAE> python .\minimization.py .\1.jpg 30
#Energy input image: 115034.0
# nth cycle             energy
0.0020012855529785156
1                       108723.07241954212
2                       108166.26190293483
3                       108159.65387846176
4                       108159.65387846176
5                       108159.65387846176
6                       108159.65387846176
7                       108159.65387846176
8                       108159.65387846176
9                       108159.65387846176
10                      108159.65387846176
11                      108159.65387846176
12                      108159.65387846176
```

图 2 运行演示