计算机视觉与应用实践实验报告（五）

目录

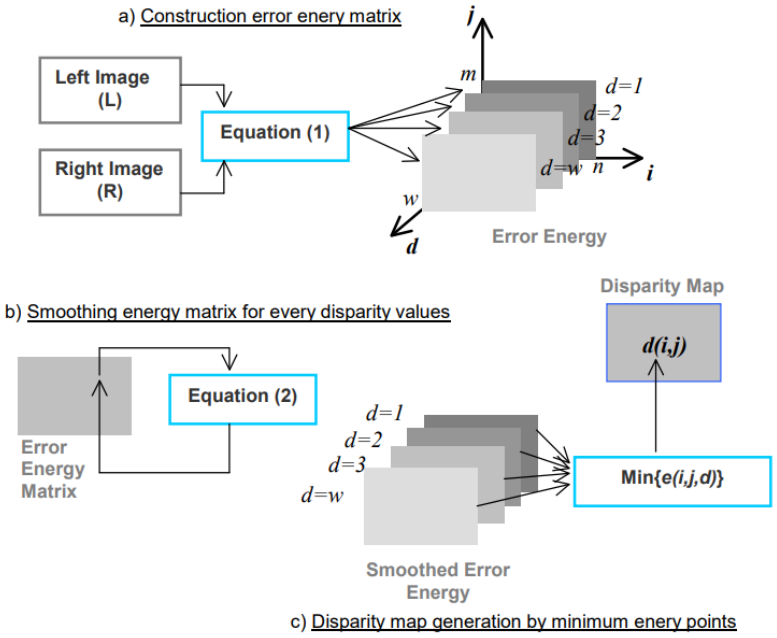
一、实验目的

- 图像视差匹配，通过立体匹配 (Stereo Matching) 得到两张图像的视差图，需要详细的实验过程和结果分析。

二、实验原理

2.1 图像视差匹配

本次实验使用的立体匹配算法思想来源于《Obtaining Depth Maps From Color Images By Region Based Stereo Matching Algorithms》。该算法属于基于灰度的匹配算法，这是一种区域相关方法， 在一幅图象中以一点为中心选定一区域（窗口），在另一幅图象中寻找与该区域相关系数最大的区域，把该找到的区域的中心认为是原来那区域中心的对应点。



（1）利用平滑函数实现全局误差能量最小化

对于视差搜索范围内的每个视差 d，计算误差能量矩阵。如上图 a 所示。

我们使用块匹配技术来为每个视差构造误差能量矩阵。对于块匹配的 n×m 窗口大小，误差能量函数可以表示为如下形式：

$$e(i,j,d)=\frac{1}{3\cdot n\cdot m}\cdot\sum_{x=i}^{i+n}\sum_{y=j}^{j+m}\sum_{k=1}^{3}\left(L(x,y+d,k)-R(x,y,k)\right)^2 \qquad (1)$$

对视差搜索范围内的视差值计算的每个误差矩阵迭代地应用平均滤波，如上图 b 所示。

$$\tilde{e}(i,j,d)=\frac{1}{n\cdot m}\sum_{x=i}^{i+n}\sum_{y=j}^{j+m}e(x,y,d) \qquad (2)$$

（2）基于最小平均误差能量的视差图

选取 error energy 最小的 d 作为视差图中（i,j）点的 d，得到视差图，如上图 c 所示

三、实验步骤

输入左视图和右视图，得到视差图和深度图。

四、关键程序代码

1、计算得到视差图，即上述的 a、b、c 过程。

```
def disparity_GEEMBSF(imgLeft, imgRight, windowSize=(3, 3), dMax=30, alpha=1):
    :param imgLeft: 左图
    :param imgRight: 右图
    :param windowSize: (n, m)，窗口的尺寸为 n x m
    :param dMax: 中值滤波迭代次数
    :param alpha: 阈值系数
    :return: 视差图，平均误差能量矩阵，函数运行时间
    """
    timeBegin = cv.getTickCount()  # 记录开始时间

    n, m = windowSize  # 窗口大小
    rows, cols, channels = imgLeft.shape  # 该实验中 imgLeft 和 imgRight 的 shape 是一样的，rows=185,cols=231,channels=3
    # 观察到论文中和实验要求中所给的左右原始图是 185(行)x231(列)像素的，而结果图中大约是 185(行)x190(列)
    cols = 190
    errorEnergyMatrixD = np.zeros((rows, cols, dMax), dtype=np.float)  # 误差能量矩阵（共 dMax 层），方便后续计算
    errorEnergyMatrixAvgD = np.zeros((rows, cols, dMax), dtype=np.float)  # 平均误差能量矩阵（共 dMax 层），方便后续计算
    imgDisparity = np.zeros((rows, cols), dtype=np.uint8)  # 具有可靠差异的视差图，将作为结果返回
    # 计算误差能量矩阵 errorEnergyMatrix，平均误差能量矩阵 errorEnergyMatrixAvg
    imgLeftPlus = cv.copyMakeBorder(imgLeft, 0, 0, n-1, m-1+dMax, borderType=cv.BORDER_REPLICATE)
    imgRightPlus = cv.copyMakeBorder(imgRight, 0, 0, n-1, m-1, borderType=cv.BORDER_REPLICATE)
    # 迭代 dMax 次
    for d in range(dMax):
        # 对整个图像进行遍历
        for i in range(rows):
            for j in range(cols):
                # 对于每个 (i, j, d) 根据公式(1)计算误差能量矩阵
```

```
                errorEnergy = (imgLeftPlus[i:i+n, j+d:j+m+d, ...] - imgRightPlus[i:i+n,
j:j+m, ...]) ** 2
                errorEnergyMatrixD[i, j, d] = np.sum(errorEnergy) / (3 * n * m)
        # 对 errorEnergyMatrix 进行遍历
        for i in range(rows):
            for j in range(cols):
                # 对于每个 (i, j, d) 根据公式(2)计算平均误差能量矩阵
                errorEnergyMatrixAvgD[i, j, d] = np.sum(errorEnergyMatrixD[i:i+n, j:j+m,
d]) / (n * m)
        # averaging filter many times. (See Figure 1.b)
        # 也就是对于每个 e(i, j, d)，进行多次平均滤波。在这里我选择执行 3 次。
        for k in range(3):
            for i in range(rows):
                for j in range(cols):
                    # 对于每个 (i, j, d) 根据公式(2)计算平均误差能量矩阵
                    errorEnergyMatrixAvgD[i, j, d] = np.sum(errorEnergyMatrixAvgD[i:i
+ n, j:j + m, d]) / (n * m)

    errorEnergyMatrixAvg = np.min(errorEnergyMatrixAvgD, axis=2)  # 平均误差能量矩阵（最
终的，只有一层）
    imgDisparity[:, :] = np.argmin(errorEnergyMatrixAvgD, axis=2)  # 视差图
    imgOrignal = imgDisparity.copy()  # 保留一份，并作为结果返回
    # 下面的部分我们实现论文中的：（包含公式 5、6、7、8、9）
    # 可靠差异的视差图
    # "Filtering Unreliable Disparity Estimation By Average Error Thresholding Mechanism"
    Ve = alpha * np.mean(imgDisparity)  # 计算 Ve
    temp = errorEnergyMatrixAvg.copy()
    temp[temp > Ve] = 0
    temp[temp != 0] = 1
    temp = temp.astype(np.int)
    imgDisparity = np.multiply(imgDisparity, temp).astype(np.uint8)  # 大于 Ve 的设置为
0
    timeEnd = cv.getTickCount()  # 记录结束时间
    time = (timeEnd - timeBegin) / cv.getTickFrequency()  # 计算总时间

    return imgOrignal, imgDisparity, errorEnergyMatrixAvg, time
```

2、 根据视差图得到深度图

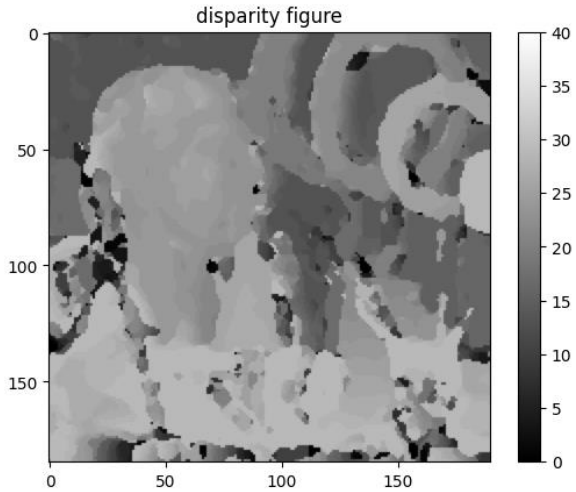
```
def depthGeneration(imgDisparity, f=30, T=20):
    """
    实现论文中的"Depth Map Generation From Disparity Map"
    根据视差图，实现深度图
    :param imgDisparity: 具有可靠差异的视差图
    :param f: 焦距
    :param T: 间距
    :return: 深度图
```

```
    """
    # 实现公式(4)
    rows, cols = imgDisparity.shape
    imgDepth = np.zeros((rows, cols), dtype=np.uint8)
    for i in range(rows):
        for j in range(cols):
            if imgDisparity[i, j] == 0:
                imgDepth[i, j] = 0
            else:
                imgDepth[i, j] = f * T // imgDisparity[i, j]
    return imgDepth
```

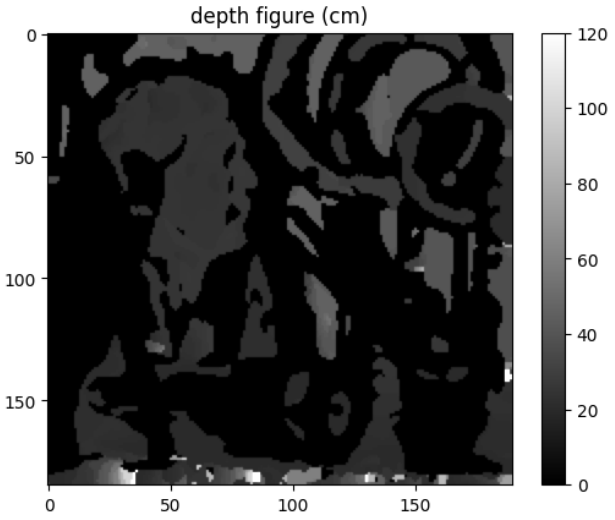
五、实验结果

（1）左右视图

（2）视差图



（3）深度图



六、实验分析与总结

本次实验相当于是对《Obtaining Depth Maps From Color Images By Region Based Stereo Matching Algorithms》论文的一个复现，因为发表时间较早，没有用到卷积神经网络等深度学习方式，只涉及机器学习中机器视觉的基本操作，无需大量数据进行训练，结果相对 groudtruth 来说差距比较大，后续可以继续在基于卷积神经网络或者 transformer 的基础上做改进。