计算机视觉实践实验报告（五）

目录

[一． 实验目的 2](#_Toc134133027)

[二． 实验原理 2](#_Toc134133028)

[2.1图像的视差 2](#_Toc134133029)

[2.2立体匹配算法 3](#_Toc134133030)

[2.3 SAD算法 3](#_Toc134133031)

[2.4 SSD算法 4](#_Toc134133032)

[三． 实验步骤 6](#_Toc134133033)

[四． 数据集 6](#_Toc134133034)

[五． 程序代码 6](#_Toc134133035)

[六． 实验结果 12](#_Toc134133036)

[6.1原图 12](#_Toc134133037)

[6.2 使用SAD 13](#_Toc134133038)

[6.3SAD与SSD 13](#_Toc134133039)

[6.4不同窗口比较 15](#_Toc134133040)

[七．实验分析与总结 17](#_Toc134133041)

# 一． 实验目的

* 通过立体匹配算法实现对两张图像的视差计算
* 得到图像的视差图
* 获得物体间的距离信息

# 二． 实验原理

主要包括图像的视差，立体匹配算法，SAD算法，SSD算法 4个部分。

# 2.1图像的视差

图像的视差是指两幅图像中对应像素之间的水平位移量。当我们从不同的位置或角度观察同一个物体时，由于物体在空间中的位置不同，它在两个视点上的投影也会有所不同，这种差异导致了两个视点上的图像像素之间的视差。

视差可以用来计算物体间的距离。通过计算两幅图像中对应像素之间的视差，可以确定左右相机之间的距离，进而计算出物体到相机的距离。这是计算机视觉中很重要的一项技术，可以用于三维重建、虚拟现实、机器人视觉等领域。

在立体视觉中，视差可以通过左右相机之间的基线长度和两个像素之间的像素位置差来计算。视差越大，表示物体离相机越近；视差越小，表示物体离相机越远。通常情况下，视差的取值范围是从0到最大视差，最大视差的取值取决于相机的分辨率和基线长度

# 2.2立体匹配算法

立体匹配算法是图像视差匹配的一种常见方法，用于计算两幅图像中对应像素之间的视差。立体匹配算法可以分为局部匹配和全局匹配两种类型。

局部匹配算法是指以一个像素为中心，计算该像素周围一定范围内的像素与右图像素的相似度，并选择相似度最高的像素作为匹配点。常用的局部匹配算法有SAD（Sum of Absolute Differences）、SSD（Sum of Squared Differences）、NCC（Normalized Cross Correlation）等。

全局匹配算法则是指对整张图像进行匹配，将匹配问题转化为一个能量最小化问题，并通过优化能量函数来得到最终的匹配结果。常用的全局匹配算法有Belief Propagation、Graph Cut、Semi-Global Matching等。

# 2.3 SAD算法

局部匹配算法中的 SAD (Sum of Absolute Differences) 算法是一种常用的立体匹配算法，用于计算两幅图像中对应像素之间的视差。该算法假设对应物体在两幅图像中的像素位置相邻，因此只需要在一个窗口范围内比较对应像素之间的灰度差异，即可得到视差。

下面是 SAD 算法的详细过程：

对于图像中每个像素点，设置一个固定大小的窗口（例如 5×55×5），在另一幅图像中搜索相应的像素点。

将两个像素点之间的灰度值做差，取绝对值，然后求和。这个求和值称为 SAD 值。

将窗口在另一幅图像中向右平移一个像素，重复上述操作，得到相邻像素点的 SAD 值。对于每个像素，选择 SAD 值最小的像素对应的视差作为该像素的视差值。

对整个图像重复上述操作，得到每个像素点的视差图。

SAD 算法的缺点是计算量大，可能会造成误匹配。因此，实际应用中常采用一些优化方法来提高算法效率和准确度，例如基于像素点相似性的区域匹配算法、多分辨率匹配算法、自适应窗口大小算法等

# 2.4 SSD算法

局部匹配算法中的 SSD (Sum of Squared Differences) 算法也是一种常用的立体匹配算法，用于计算两幅图像中对应像素之间的视差。和 SAD 算法类似，SSD 算法也是在一个窗口内比较对应像素之间的灰度差异，但不同的是它计算的是差值的平方和。

下面是 SSD 算法的详细过程：

5×55×5），在另一幅图像中搜索相应的像素点。

对于图像中每个像素点，设置一个固定大小的窗口（例如 5×55×5），在另一幅图像中搜索相应的像素点。

将两个像素点之间的灰度值做差，取平方，然后求和。这个求和值称为 SSD 值。

将窗口在另一幅图像中向右平移一个像素，重复上述操作，得到相邻像素点的 SSD 值。对于每个像素，选择 SSD 值最小的像素对应的视差作为该像素的视差值。

对整个图像重复上述操作，得到每个像素点的视差图。

SSD 算法相较于 SAD 算法，计算的是差值的平方和，因此对于像素值差异较大的区域，可能更容易产生较大的误差。但是，由于 SSD 算法计算的是平方和，因此对于像素值差异较小的区域，可能会比 SAD 算法更精确。

SSD 算法同样也存在一些优化方法，如基于像素点相似性的区域匹配算法、多分辨率匹配算法、自适应窗口大小算法等。

# 三． 实验步骤

图示

描述已自动生成

# 四． 数据集

自己收集的对同一个物体不同视角的图片



# 五． 程序代码

1.导入需要的包

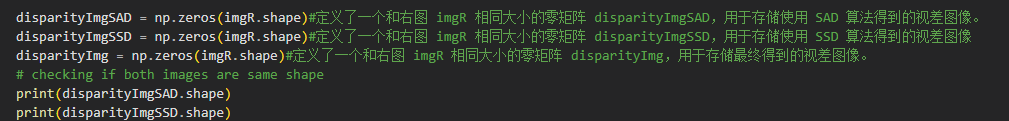
图形用户界面, 文本

描述已自动生成

2. 加载数据集，并且定义存储结果图像

文本, 聊天或短信

描述已自动生成



3.只使用SAD方式计算视差文本

描述已自动生成

4.使用SAD和SSD

文本

描述已自动生成

5.不同种窗口进行比较

文本

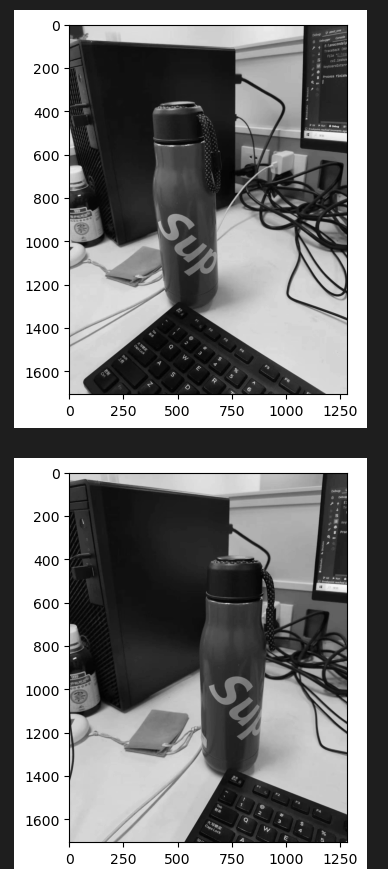
描述已自动生成

文本

描述已自动生成

# 六． 实验结果

# 6.1原图



# 6.2 使用SAD

图片包含 图示

描述已自动生成

# 6.3SAD与SSD

图示

低可信度描述已自动生成

图片包含 图示

描述已自动生成

# 6.4不同窗口比较

图片包含 QR 代码

描述已自动生成

图示

低可信度描述已自动生成

# 七．实验分析与总结

* **SAD方法：**

SAD (Sum of Absolute Differences) 是一种计算图像视差的方法，其基本思想是计算左右两幅图像对应区域内像素值的差值的绝对值之和，来评估两幅图像中像素的匹配程度。

SAD方法的实现步骤如下：

选择一个窗口在左图像上进行滑动，在滑动的过程中，记录下每个窗口在左图像上的位置。

在滑动的同时，在右图像上对应位置的窗口内搜索与左图像窗口相似的区域，记录下这个区域在右图像上的位置。

计算左右两个窗口内对应像素值之间的绝对差值，并将这些差值累加起来，得到该左图像窗口和右图像对应区域的SAD值。

在所有右图像区域中选择SAD值最小的一个，将这个最小值作为该左图像窗口的匹配值，用该匹配值计算该窗口的视差值。

对左图像中所有窗口进行上述操作，得到整个图像的视差图。

SAD方法的优点是计算简单，实现容易，但其缺点也很明显：它不能很好地应对纹理复杂度高的情况，比如森林、草地等场景，这是因为SAD只考虑了像素之间的差异，而未考虑像素之间的相关性。

* **SSD方法：**

SSD（Sum of Squared Differences）方法是一种常见的视差匹配方法，它与 SAD 方法类似，但是在计算匹配代价时使用的是两个窗口内像素值的平方差，而不是绝对值的和。

具体地，给定左图像中的一个像素点，我们需要在右图像中找到与之对应的像素点。首先，在右图像中选取与该像素点对应的一个窗口，然后在左图像中选取一个大小相同的窗口。接下来，分别计算这两个窗口内像素值的平方差，将其累加起来得到匹配代价。将所有可能的窗口匹配代价计算出来后，选择匹配代价最小的窗口作为匹配窗口，计算该窗口在左图像中与对应窗口在右图像中的中心点之间的距离，即为该像素点的视差值。

SSD 方法的优点是，它对像素值的差异较大的区域比 SAD 方法更为敏感，因为平方差的值较大。但是，与 SAD 方法相比，SSD 方法计算代价的复杂度更高，因为要进行平方操作。

在实际应用中，我们可以通过调整窗口的大小来平衡匹配代价和计算复杂度之间的权衡。较小的窗口通常会导致更准确的匹配，但也会增加计算复杂度，而较大的窗口会减少计算复杂度，但也会降低匹配的准确性。

* **实验结果分析：**

对于 SAD 与 SSD 方法的比较，从图像效果来看，两者效果都很不错，都能够得到较为清晰的视差图。但是从计算速度与噪声敏感度来看，SAD 方法比 SSD 方法更快，同时更加噪声不敏感，因此在实际应用中更加常用