南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

“计算机视觉与应用实践”
课程作业

姓 名：余 涛　　学 号：122106222833

专 业：计算机科学与工程

项目名称：通过立体匹配(Stereo Matching)得到两张图像的视差图

时间：2023 年 4 月 29 日

1、实验目标

（1）、图像视差匹配；

（2）、通过立体匹配(Stereo Matching)得到两张图像的视差图。

2、具体要求

（1）、图像视差匹配；

（2）、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。

3、总体思路

立体匹配算法通常由四个部分组成，包括：匹配代价计算，代价聚合，视差计算和视差优化。

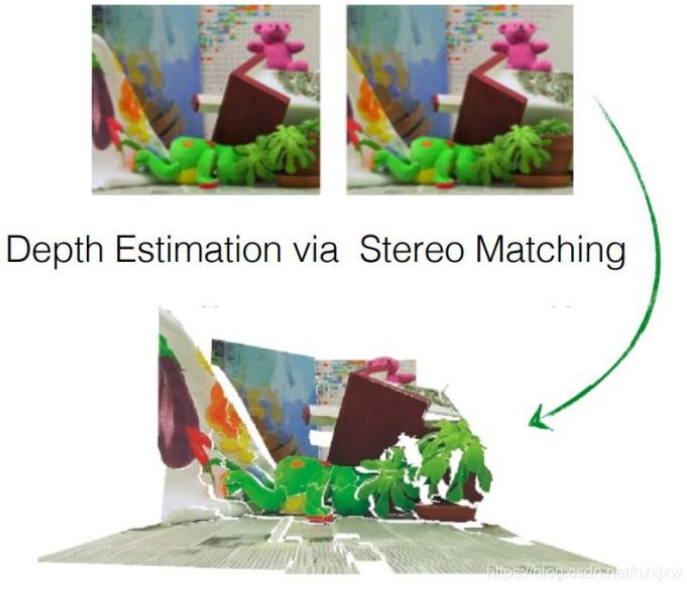
双目立体匹配一直是双目视觉的研究热点，双目相机拍摄同一场景的左、右两幅视点图像，运用立体匹配匹配算法获取视差图，进而获取深度图。而深度图的应用范围非常广泛，由于其能够记录场景中物体距离摄像机的距离，可以用以测量、三维重建、以及虚拟视点的合成等。

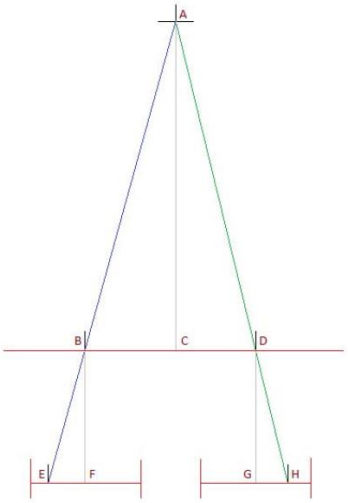
4、项目实现

一、原理与实现

1、立体匹配原理与理解

立体匹配技术就是通过匹配两幅或者多幅图像来获得视差(disparity)图。通过立体匹配可以获得深度，进行深度估计。





Derivation:

$$\frac{d}{f} = \frac{B}{Z}$$

$$Z = \frac{Bf}{d}$$

我们人就是一个双目视觉系统

距离越远，左眼和右眼看到的区别越小
距离越近，左眼和右眼看到的区别越大

立体匹配算法通常由四个部分组成，包括：匹配代价计算，代价聚合，视差计算和视差优化。

双目立体匹配一直是双目视觉的研究热点，双目相机拍摄同一场景的左、右两幅视点图像，运用立体匹配匹配算法获取视差图，进而获取深度图。而深度图的应用范围非常广泛，由于其能够记录场景中物体距离摄像机的距离，可以用以测量、三维重建、以及虚拟视点的合成等。

2、视差图

双目立体视觉融合两只眼睛获得的图像并观察它们之间的差别，使我们可以获得明显的深度感，建立特征间的对应关系，将同一空间物理点在不同图像中的映像点对应起来，这个差别，我们称作视差(Disparity)图像。

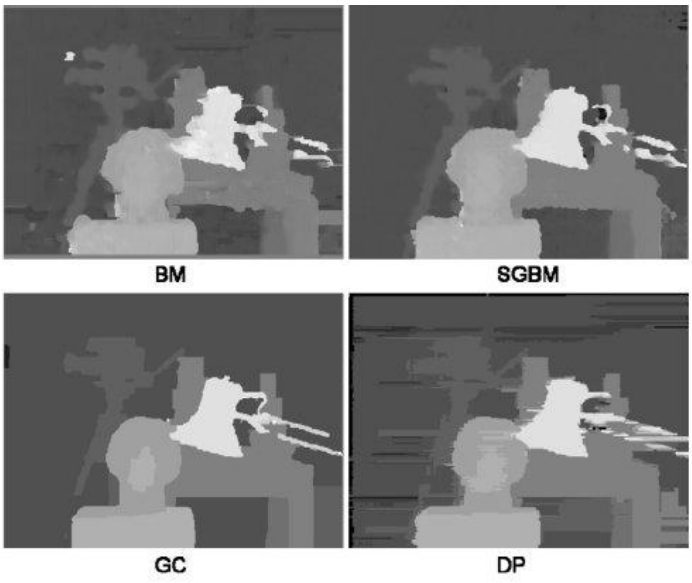
3、双目图片进行立体匹配获取深度图

①摄像机标定（包括内参和外参）

②双目图像的校正（包括畸变校正和立体校正）

③立体匹配算法获取视差图，以及深度图

④利用视差图，或者深度图进行虚拟视点的合成



4、记录下 Binocular Stereo Matching 的点

（1）、双目立体视觉系统可以根据左右相机拍摄得到的场景图像，恢复图像中物体的位置、外部特征等信息，还可以恢复出场景里障碍物的位置，本质是从二维图像中恢复出三维信息的问题；

（2）、双目立体视觉系统包含：离线相机标定、双目图像采集和矫正、立体匹配、计算深度信息；

（3）、立体匹配：在左右图像上寻找对应的匹配点，利用三角测量原理计算视差；

（4）、传统立体匹配有四个步骤：匹配代价计算，代价聚合、代价最优化计算、视差调整；

（5）、匹配代价计算：针对双目摄像机拍到的一对图像，针对图像上的某点，在经过极线约束过后的右幅图像上的同水平方向上找到同源点，计算两个点的相似性。匹配代价计算分为两类，一个是根据原始图像的像素的相似性去比较，比如比较左右视图内对应的一个区域内的全部像素求差的绝对值再求和(SAD)，另一个就是根据特征描述的角度上比较，比如 SIFT、BRIEF,构建成功的匹配代价函数是立体匹配算法的关键，设计出抗干扰、抗噪声并且对光照保持不敏感的代价函数可以有效提升立体匹配的精度，所以匹配代价函数在基于全局优化的立体匹配算法和基于局部优化的立体匹配算法；

（6）、匹配代价聚合：将待匹配的像素点周围一个匹配窗口，将这个窗口范围内的所有像素点计算匹配代价，，再对全部的匹配代价进行求和或者平均值，把这个值作为该待匹配下像素点的匹配代价。根据窗口内像素点到中心点像素的距离作为不同权重来对匹配代价进行加和的方法称为高斯滤波匹配法；

（7）、代价最优化计算：选择匹配代价函数最小的那个像素点是真实对应的匹配点。但是对于弱纹理和重复纹理的区域，经常出现很多区域与待匹配区域相似，所以基于局部优化算法针对这种图像匹配效果较差。代价最优化计算的目的是让匹配代价函数能够更加真实反映出待匹配区域和真实区域的相似性，全局优化算

法没有代价聚合步骤，全局优化的立体匹配算法选择构造全局能量函数的方法，在代价匹配中增加平滑项，目标像素的能量函数值与其领域范围的近点的像素的能量函数值差距较小，与领域内函数距离较远点的能量函数值差距大；

（8）、视差图后处理：为了提升最后的匹配精度。大部分立体算法得到的视差图是离散的，经过例如采用中值滤波算法连续化最终的视差图，得到更加精确的预测视差图。

二、具体步骤及过程】

1. 预处理：

```
    # 彩色图->灰度图
        if(img1.ndim == 3):#判断为三维数组
            img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)  # 通过
    OpenCV 加载的图像通道顺序是 BGR
        if(img2.ndim == 3):
            img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
        # 直方图均衡
        img1 = cv2.equalizeHist(img1)
        img2 = cv2.equalizeHist(img2)

        return img1, img2
```

2. 消除畸变：

```
    undistortion_image = cv2.undistort(image, camera_matrix, dist_coeff)
    return undistortion_image
```

3. 获取畸变校正和立体校正的映射变换矩阵、重投影矩阵：

```
# 读取内参和外参
    left_K = config.cam_matrix_left
    right_K = config.cam_matrix_right
    left_distortion = config.distortion_l
    right_distortion = config.distortion_r
    R = config.R
    T = config.T
    # 计算校正变换
    R1, R2, P1, P2, Q, roi1, roi2 = cv2.stereoRectify(left_K, left_distortion, right_K,
right_distortion,
                                                      (width, height), R, T,
alpha=0)

    map1x, map1y = cv2.initUndistortRectifyMap(left_K, left_distortion, R1, P1,
(width, height), cv2.CV_32FC1)
    map2x, map2y = cv2.initUndistortRectifyMap(right_K, right_distortion, R2, P2,
(width, height), cv2.CV_32FC1)

    return map1x, map1y, map2x, map2y, Q
```

4. 立体校正检验：

```
    # 建立输出图像
    height = max(image1.shape[0], image2.shape[0])
    width = image1.shape[1] + image2.shape[1]

    output = np.zeros((height, width, 3), dtype=np.uint8)
    output[0:image1.shape[0], 0:image1.shape[1]] = image1
    output[0:image2.shape[0], image1.shape[1]:] = image2

    # 绘制等间距平行线
    line_interval = 50  # 直线间隔：50
    for k in range(height // line_interval):
        cv2.line(output, (0, line_interval * (k + 1)), (2 * width, line_interval * (k + 1)),
(0, 255, 0), thickness=2, lineType=cv2.LINE_AA)

    return output
```

5. 视差计算：

```
# SGBM 匹配参数设置
    if left_image.ndim == 2:
        img_channels = 1
    else:
        img_channels = 3
    blockSize = 3
    paraml = {'minDisparity': 0,
              'numDisparities': 128,
              'blockSize': blockSize,
              'P1': 8 * img_channels * blockSize ** 2,
              'P2': 32 * img_channels * blockSize ** 2,
              'disp12MaxDiff': 1,
              'preFilterCap': 63,
              'uniquenessRatio': 15,
              'speckleWindowSize': 100,
              'speckleRange': 1,
              'mode': cv2.STEREO_SGBM_MODE_SGBM_3WAY
              }

    # 构建 SGBM 对象
    left_matcher = cv2.StereoSGBM_create(**paraml)
    paramr = paraml
    paramr['minDisparity'] = -paraml['numDisparities']
    right_matcher = cv2.StereoSGBM_create(**paramr)

    # 计算视差图
    size = (left_image.shape[1], left_image.shape[0])
```

```
        if down_scale == False:
            disparity_left = left_matcher.compute(left_image, right_image)
            disparity_right = right_matcher.compute(right_image, left_image)
```

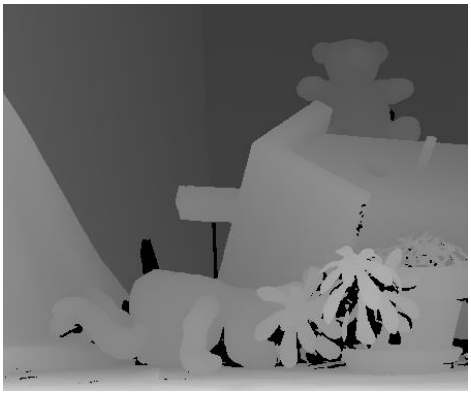
6. 将 h×w×3 数组转换为 N×3 的数组：

```
    height, width = points.shape[0:2]

    points_1 = points[:, :, 0].reshape(height * width, 1)
    points_2 = points[:, :, 1].reshape(height * width, 1)
    points_3 = points[:, :, 2].reshape(height * width, 1)

    points_ = np.hstack((points_1, points_2, points_3))

    return points_
```

三、结果展示

示例一：

以下为左图片，右图片，视差图



示例二：

以下为左图片，右图片，视差图（灰度图）

三、代码运行说明

（1）、代码使用 python 编写，运行时请使用 pycharm 等 python 编译器。

（2）、 Stereo_Matching 总共以下文件： stereo_matching.py, stereoCamera.py 以及 image_file 文件。

（3）、使用时直接运行 stereo_matching.py 文件，此外 image_file 包含左图片，右图片。