南京郵電大學
Nanjing University of Posts and Telecommunications

（2023-2024 学年 第 2 学期）

计算机视觉实验报告

题　　目　相机标定

所在学院　自动化学院、人工智能学院

专　　业　人工智能

年级班级　B210416

学　　号　B21080526

姓　　名　单家俊

授课教师　范保杰

2024　年 5 月 16 日

实验二 Matlab 相机标定

一、实验目的

学会使用相机标定工具箱

二、实验内容

1. 标定相机内参
2. 标定相机外参

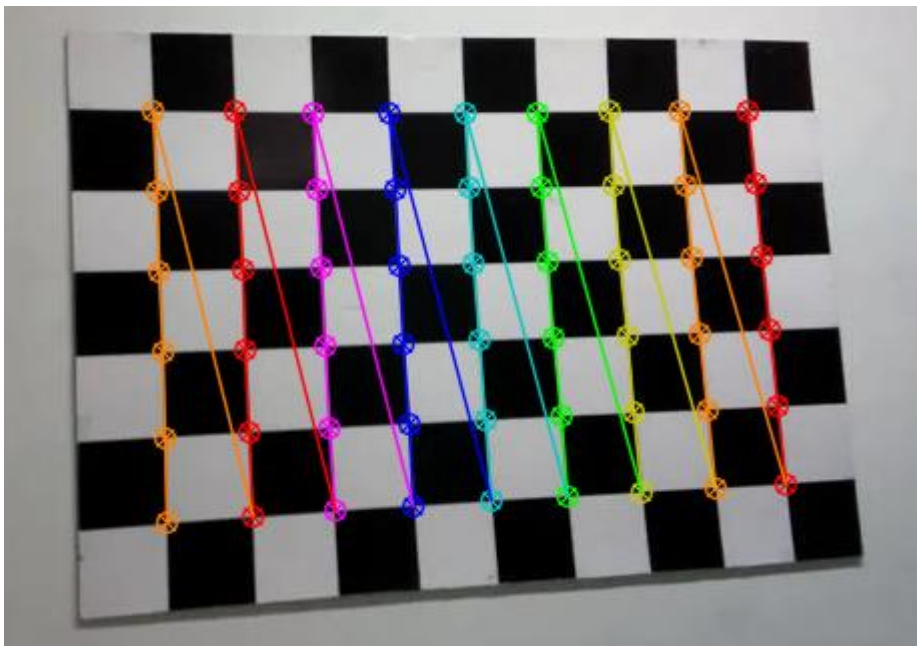
三、实验说明

实验代码：

```
import cv2  # 导入 OpenCV 库。
import numpy as np  # 导入 NumPy 库，用于高效的矩阵和数组操作。
import os  # 导入 OS 库，用于与操作系统交互。
import glob  # 导入 glob 库，用于文件路径名的模式匹配。

# 定义棋盘格的尺寸，这里是 6x9。
CHECKERBOARD = (6, 9)
# 设置寻找角点的终止准则。
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
# 创建列表以存储棋盘格的 3D 点。
objpoints = []
# 创建列表以存储棋盘格角点在图像中的 2D 点。
imgpoints = []
# 初始化 3D 点的坐标，这里仅初始化 x 和 y，z 设置为 0。
objp = np.zeros((1, CHECKERBOARD[0] * CHECKERBOARD[1], 3), np.float32)
objp[0, :, :2] = np.mgrid[0:CHECKERBOARD[0],
0:CHECKERBOARD[1]].T.reshape(-1, 2)
prev_img_shape = None  # 初始化之前图像的尺寸。
# 使用 glob 模式匹配来获取所有棋盘格图片的路径。
# images = glob.glob('./images/IMG_20240402_160207.jpg')
# for fname in images:
img = cv2.imread('picture1.png')  # 读取每一张图片。
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # 将图片转换为灰度图，因为寻找角点在灰度图上进行。

# 寻找棋盘格角点。
```

```
ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK +
cv2.CALIB_CB_NORMALIZE_IMAGE)
# 如果找到足够数量的角点，则细化它们的位置并将它们添加到列表中。
if ret == True:
    objpoints.append(objp)
    corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
    imgpoints.append(corners2)

    # 将找到的角点绘制到图片上，以便可视化。
    img = cv2.drawChessboardCorners(img, CHECKERBOARD, corners2, ret)

cv2.imshow('img', img)  # 显示图片。
cv2.waitKey(0)  # 等待用户按键。

cv2.destroyAllWindows()  # 关闭所有 OpenCV 窗口。

h, w = img.shape[:2]  # 获取最后处理的图片的尺寸。

# 使用 3D 点和对应的 2D 图像点进行相机标定。
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1], None,
                                                   None)

# 打印相机标定的结果。
print("Camera matrix : \n")
print(mtx)  # 相机矩阵
print("dist : \n")
print(dist)  # 畸变系数
print("rvecs : \n")
print(rvecs)  # 旋转向量
print("tvecs : \n")
print(tvecs)  # 平移向量
```

实验结果：



```
Camera matrix :

[[464.49405703   0.         256.69513204]
 [  0.         450.33390296 211.82106216]
 [  0.           0.           1.        ]]
dist :

[[ 0.14528703 -0.02491773 -0.00448671 -0.0158665  -0.11402032]]
rvecs :

(array([[-0.21669059],
       [-0.04920857],
       [ 1.53448393]]),)
tvecs :

(array([[ 7.13158331],
       [-2.03725579],
       [12.66491579]]),)

进程已结束，退出代码为 0
```

四、实验心得

在这次相机标定实验中，我使用了 OpenCV 库和 NumPy 库，对相机进行了标定。通过处理多张棋盘格图像，成功计算出了相机的内参矩阵、畸变系数、旋转向量和平移向量。

首先，我编写了代码以读取棋盘格图像，并将其转换为灰度图。然后，使用 cv2.findChessboardCorners 函数检测棋盘格的角点，并通过 cv2.cornerSubPix

函数进一步细化角点的位置。在成功检测到角点后，我将这些角点存储起来，用于相机标定。在处理完所有图像后，我使用 cv2.calibrateCamera 函数进行相机标定，得到了相机矩阵（内参矩阵）、畸变系数、旋转向量和平移向量。

标定结果显示了相机的内外参数。相机矩阵（mtx）表示了相机的焦距和光心位置，畸变系数（dist）显示了相机镜头的径向和切向畸变情况。旋转向量（rvecs）和平移向量（tvecs）则描述了相机在三维空间中的姿态和位置。

通过实验，我体会到准确检测和细化棋盘格角点是确保标定结果准确的关键。使用 cv2.cornerSubPix 函数来细化角点位置，显著提高了检测精度，进而提高了相机标定的准确性。此外，为了获得准确和稳定的标定结果，使用多张不同角度和位置的棋盘格图像进行标定是必要的。多样化的图像能够提供更多的特征点，提高标定的鲁棒性和精度。

实验中得到的畸变系数显示了相机镜头的径向和切向畸变情况，了解和校正这些畸变对于图像处理和计算机视觉应用至关重要。得到的相机矩阵和畸变系数可以用于图像校正和 3D 重建。通过校正图像中的畸变，可以显著提高图像的质量和精度，为后续的计算机视觉任务打下基础。

这次实验让我掌握了相机标定的基本方法，理解了相机内外参数对图像处理的影响。这为我今后在计算机视觉领域的研究和应用打下了坚实的基础。为了进一步提高标定结果的准确性，可以使用更多的棋盘格图像进行标定，尤其是增加不同视角和距离的图像。此外，在角点检测前，对图像进行一些预处理，如去噪和平滑处理，可以提高角点检测的成功率和精度。总的来说，这次实验让我对相机标定有了更深入的了解，并为今后的图像处理工作提供了宝贵的经验。

南京郵電大學
Nanjing University of Posts and Telecommunications

（2023-2024 学年 第 2 学期）

计算机视觉实验报告

题　　目　特征点检测与匹配

所在学院　自动化学院、人工智能学院

专　　业　人工智能

年级班级　B210416

学　　号　B21080526

姓　　名　单家俊

授课教师　范保杰

2024　年 5 月 16 日

实验三 特征点检测与匹配

一、实验目的

检测图像中的特征点（如 harris 等），并完成两幅或多幅图像中特征点的匹配。

二、实验内容

1. 检测图像中特征点
2. 匹配图像中的特征点

三、实验说明

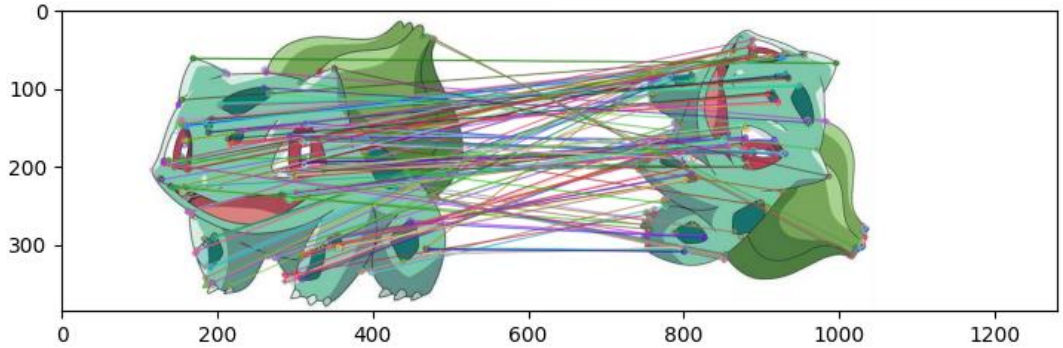
根据上课内容，深入理解 harris 角点检测的原理，编程实现经典的 harris 角点检测程序，利用欧氏距离作为度量准则，来判断特征点间的相似程度，完成特征点的匹配。

实验代码：

```
import numpy as np
import matplotlib
import matplotlib.pylab as plt
import cv2

matplotlib.use('TkAgg')
# Harris 角点检测
img = plt.imread("wangba2.jpeg").copy()
img_ = img.copy()
gray = np.float32(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY))
harris = cv2.cornerHarris(gray, 2, 3, 0.04)
harris = cv2.dilate(harris, None)  # 膨胀，方便显示
img[harris > 0.01 * harris.max()] = [255, 0, 0]

plt.subplot(1, 2, 1)
plt.axis("off")
plt.imshow(img_)
plt.subplot(1, 2, 2)
plt.axis("off")
plt.imshow(img)
plt.show()
# ORB 特征点提取
img = plt.imread("wangba.jpeg").copy()

orb = cv2.ORB_create()  # 可以自定义很多参数
kp = orb.detect(img)  # 特征点
```

```
kp_img = cv2.drawKeypoints(img, keypoints=kp, outImage=None, color=300)

# 绘制
plt.subplot(1, 2, 1)
plt.axis("off")
plt.imshow(img)
plt.subplot(1, 2, 2)
plt.axis("off")
plt.imshow(kp_img)
plt.show()

# ORB 特征点匹配
img1 = plt.imread("wangba.jpeg").copy()
img2 = plt.imread("wangba2.jpeg").copy()
orb = cv2.ORB_create()
# 特征点、描述子
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
# 匹配
match = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True).match(des1, des2)
match = sorted(match, key=lambda x: x.distance)
# 取最近的 minN 个绘制
minN = len(match)
# minN = 40
img3 = cv2.drawMatches(img1, kp1, img2, kp2, match[:minN], None)
plt.imshow(img3)
plt.show()
```

实验结果：



四、实验心得

在这次实验中，我使用了 Harris 角点检测器来检测图像中的特征点，并完成了两幅或多幅图像中特征点的匹配。通过实验，我对特征点检测和匹配的过程有了更深入的理解。

特征点检测是图像处理中的关键步骤。Harris 角点检测器能够有效地检测图像中的角点，这些角点在不同视角和光照条件下具有较高的稳定性和可辨识性。在实验中，Harris 角点检测器准确地识别出图像中的显著特征点，为后续的特征描述和匹配提供了良好的基础。

通过实验结果的可视化，我观察到大部分特征点能够正确匹配，但也存在少量误匹配，主要出现在图像边缘区域或纹理较为均匀的区域。为提高匹配精度，未来可以尝试更多的特征点检测算法，结合多种特征描述子，并优化特征匹配算法，使用如 RANSAC 等鲁棒性算法过滤掉错误匹配。

这次实验让我掌握了 Harris 角点检测器的使用方法，也深入理解了特征点匹配的原理和挑战，为我今后在图像处理和计算机视觉领域的研究打下了基础。

五、思考题

Harris 特征点的不足与改进对策？

不足：

1. 对尺度变化不敏感：

Harris 角点检测器对图像的尺度变化不敏感，这意味着在不同尺度下检测到的特征点可能会有所不同。对于图像中存在不同大小物体的情况，Harris 角点检测器可能无法稳定检测出相同的特征点。

2. 对旋转不敏感：

Harris 角点检测器对图像的旋转不敏感，导致在旋转后图像中的特征点检测结果可能会发生变化，这限制了其在旋转不变性要求较高的应用场景中的有效性。

3. 噪声敏感性：

Harris 角点检测器对图像噪声较为敏感，特别是在高噪声环境下，可能会产生大量的误检特征点，影响后续的特征匹配和分析。

4. 特征点的分布不均匀：

Harris 角点检测器在检测特征点时，可能会在高频区域检测到大量特征点，而在低频区域检测到的特征点较少，导致特征点分布不均匀，影响匹配效果。

改进对策：

1. 使用多尺度特征检测器：

为解决尺度不敏感的问题，可以使用多尺度特征检测器，如 SIFT（尺度不变特征变换）和 SURF（加速稳健特征），它们能够在不同尺度下检测稳定的特征点，增强对尺度变化的鲁棒性。

2. 引入旋转不变性：

为解决旋转不敏感的问题，可以采用 ORB（定向快速和旋转不变的特征）等旋转不变特征检测器，ORB 在特征检测和描述过程中加入了旋转不变性，使得特征点在旋转图像中也能保持稳定。

3. 噪声预处理和后处理技术：

为减小噪声的影响，可以在图像预处理阶段使用去噪算法，如高斯滤波、中值滤波等，减少图像噪声。另外，可以结合 RANSAC（随机抽样一致性）等算法，在特征匹配阶段过滤掉噪声导致的误匹配特征点，提高匹配的准确性。

4. 结合其他特征点检测器：

为解决特征点分布不均的问题，可以结合 FAST（特征加速段检测器）等快速特征检测器，均衡特征点在图像中的分布。通过多种检测器的结合，可以获取更多的有效特征点，提高匹配的效果。

南京郵電大學
Nanjing University of Posts and Telecommunications

（2023-2024 学年 第 2 学期）

计算机视觉实验报告

题　　目　掌握图像处理工具箱

所在学院　自动化学院、人工智能学院

专　　业　人工智能

年级班级　B210416

学　　号　B21080526

姓　　名　单家俊

授课教师　范保杰

2024　年 5 月 16 日

实验一 掌握图像处理工具箱

一、实验目的

学习 OPENCV 和 Matlab 的图像处理工具箱。

二、实验内容

1. 在 VS/python 环境下，安装配置及使用 OPENCV 图像处理工具箱。
2. 安装配置及使用 Matlab 图像处理工具箱。

三、实验说明

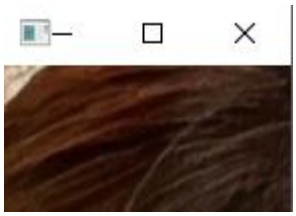
实验代码：

```
import cv2
import numpy as np

kernel = np.ones((5, 5), np.uint8)

img = cv2.imread('test.jpg')
cv2.namedWindow('test', cv2.WINDOW_KEEPRATIO)
cv2.imshow('test', img)
cv2.waitKey(0)
cv2.imwrite("test1.jpg", img=img)

imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray, (7, 7), 0)
imgCanny = cv2.Canny(img, 150, 200)
imgDilation = cv2.dilate(imgCanny, kernel, iterations=1)
imgErode = cv2.erode(imgDilation, kernel, iterations=1)

cv2.imshow("Gray Img", imgGray)
cv2.imshow("Blur Img", imgBlur)
cv2.imshow("Canny Img", imgCanny)
cv2.imshow("Dilation Img", imgDilation)
cv2.imshow("Erode Img", imgErode)

cv2.waitKey(0)

print(img.shape)
imgResize = cv2.resize(img, (1000, 500))
print(imgResize.shape)
imgCropped = img[46:119, 352:495]
cv2.imshow('Resize Img', imgResize)
```

```
cv2.imshow('Cropped Img', imgCropped)
cv2.waitKey(0)

width, heigth = 250, 300
pts1 = np.float32([[111,219],[287,188],[151,482],[352,440]])
pts2 = np.float32([[0,0],[width,0],[0,heigth],[width,heigth]])
matrix = cv2.getPerspectiveTransform(pts1,pts2)
imgOutput = cv2.warpPerspective(img, matrix, (width, heigth))

cv2.imshow('Img', img)
cv2.imshow('Output', imgOutput)
cv2.waitKey(0)
```

实验结果：
灰度图、模糊图片、提取边缘、边缘膨胀、边缘细化：

缩放、裁剪：



图片扭曲：

四、实验心得

通过这次实验，我学会了如何使用 opencv 库进行基本的图像处理操作。

五、思考题

Matlab 与 OPENCV 中图像数据的存取过程中，初始的行列坐标分别为多少？

Matlab 中，图像的左上角像素的坐标是 (1, 1)。

OpenCV 中，图像的左上角像素的坐标是 (0, 0)。