**武 汉 工 程 大 学**

**计算机科学与工程学院**

**《图像处理与机器视觉》实验报告[1]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 专业班级 | | 智能科学与技术1班 | 实验时间 | | **2022年10月26日** | | |
| 学生学号 | | 2005120203 | 实验地点 | | 机房 | | |
| 学生姓名 | | 方成杰 | 指导教师 | | 闵锋 | | |
|  | | | | | | | |
| 实验项目 | | 数字图像基本操作及灰度调整 | | | | | |
| 实验类别 | | 基础性 | 实验学时 | | **4** | | |
| 实验目的及要求 | | 1. 掌握图像读取与显示的基本方法；  2. 掌握vc6.0中OpenCV环境的配置方法；  3. 实现图像的灰度化,反色,翻转,缩放,叠加等功能;  4. .掌握直方图均衡化的方法，编程实现。 | | | | | |
| 成 绩 评 定 表 | | | | | | | |
| 类 别 | | 评 分 标 准 | | 分值 | | 得分 | 合 计 |
| 上机表现 | | 按时出勤、遵守纪律  认真完成各项实验内容 | | **30**分 | |  |  |
| 报告质量 | 程序代码规范、功能正确  填写内容完整、体现收获 | | | **70**分 | |  |
| **说明：**  **评阅教师：**  **日 期： 2022 年 月 日** | | | | | | | |
| 实 验 内 容 | | | | | | | |
| 1. 用vc6.0建立一个单文档工程，打开一个图像并显示出来，将此窗口抓图后，与核心代码一起粘贴到此：  def main():  img = cv2.imread("wit2.jpg")  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  show\_img\_plt([img])  return 0   def show\_img\_plt(img\_arry):  size = len(img\_arry)  plt.figure()  for i in range(1, size + 1):  plt.subplot(1, size, i)  plt.imshow(img\_arry[i - 1])  plt.show()   1. 实现图像的R，G，B通道分解，将每个通道的图像保存下来，与核心代码一起粘贴到此：   def main():  img = cv2.imread("wit2.jpg")  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  R = RGB\_single(img.copy(), 0)  G = RGB\_single(img.copy(), 1)  B = RGB\_single(img.copy(), 2)  show\_img\_plt([R, G, B])  return 0   def RGB\_single(img, channel):  for i in range(0, 3):  if i == channel:  continue  img[:, :, i] = 0  return img   1. 实现图像的灰度化，将灰度图像窗口抓图后，与核心代码一起粘贴到此：   def main():  img = cv2.imread("wit2.jpg")  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  R = img.copy()[:, :, 0]  G = img.copy()[:, :, 1]  B = img.copy()[:, :, 2]  gray = 0.2\*R + 0.7\*G + 0.1\*B  cv2.imshow("G", gray.astype(np.uint8))  cv2.waitKey()  return 0  4. 实现图像的反色，将反色图像窗口抓图后，与核心代码一起粘贴到此：  def main():  img = cv2.imread("wit2.jpg")  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  reverse = img\_reverse(img.copy())  show\_img\_plt([img, reverse])  return 0   def img\_reverse(img):  for i in range(len(img)):  for j in range(len(img[0])):  for c in range(0, 3):  img[i][j][c] = 255 - img[i][j][c]  return img  5. 实现图像的放大缩小功能，将放大缩小图像窗口抓图后，与核心代码一起粘贴到此：  def scale(img, target\_size):  H, W, C = img.shape  th, tw = target\_size  result = np.zeros((th, tw, C), np.uint8)  for i in range(th):  for j in range(tw):  for k in range(C):  # 找到在原图中对应的点的(X, Y)坐标 坐标变换如下  # x' = x \* c1  # y' = y \* c2  corr\_x = int(i \* (H / th))  corr\_y = int(j \* (W / tw))  # 防止越界  if corr\_x + 1 >= H:  corr\_x = H - 2  if corr\_y + 1 >= W:  corr\_y = W - 2  # 左上角的点  point1 = [corr\_x, corr\_y]  point2 = (point1[0], point1[1] + 1)  point3 = (point1[0] + 1, point1[1])  point4 = (point1[0] + 1, point1[1] + 1)  # 双线性插值  fr1 = (point2[1] - corr\_y) \* img[point1[0], point1[1], k] + (corr\_y - point1[1]) \* img[  point2[0], point2[1], k]  fr2 = (point2[1] - corr\_y) \* img[point3[0], point3[1], k] + (corr\_y - point1[1]) \* img[  point4[0], point4[1], k]  result[i, j, k] = (point3[0] - corr\_x) \* fr1 + (corr\_x - point1[0]) \* fr2   return result  6. 实现图像的水平和垂直翻转功能，将水平和垂直翻转图像窗口抓图后，与核心代码一起粘贴到此：  def main():  img = cv2.imread("wit2.jpg")  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  # 两种实现方式  reverse2 = img[::-1, ::, :]  reverse = img[::, ::-1, :]  reverse = img\_horizontal\_reverse(img.copy())  show\_img\_plt([img, reverse, reverse2])  return 0   def img\_horizontal\_reverse(img):  temp = img.copy()  wide = len(img[0])  high = len(img)  for i in range(len(img)):  for j in range(wide):  temp[i][wide - 1 - j] = img[i][j]  return temp   1. 实现两幅图像的叠加功能，将r=0.8,0.5,0.2叠加图像窗口抓图后，与核心代码一起粘贴到此：   def img\_overlay(img1, img2, r):  W = max(img1.shape[0], img2.shape[0])  H = max(img1.shape[1], img2.shape[1])  img1 = cv2.resize(img1, (H, W))  img1 = np.array(img1)  img2 = np.array(img2)  return img1\*r + img2\*(1 - r)   1. 实现图像的直方图均衡化功能，将直方图均衡化图像窗口抓图后，与核心代码一起粘贴到此：   def histogram\_equalization(img):  H, W, C = img.shape  pix\_count = H \* W  BGR = np.zeros((3, 256))  for i in range(H):  for j in range(W):  for c in range(C):  BGR[c, img[i, j, c]] += 1  BGR = BGR / pix\_count  for i in range(3):  for j in range(1, 256):  BGR[i, j] += BGR[i, j - 1]  BGR = np.around(BGR \* 255)  for i in range(H):  for j in range(W):  for c in range(C):  img[i, j, c] = BGR[c, img[i, j, c]]  return img | | | | | | | |