**Python OpenCV：利用鼠标移动缩放图片**

**一、实现目标**

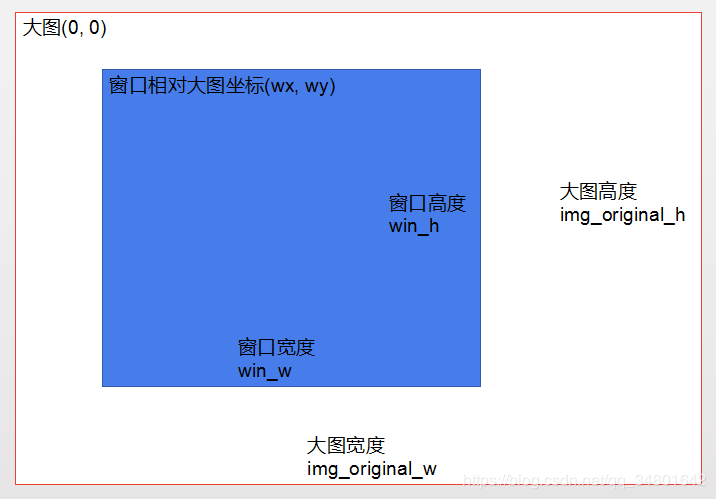
  在[OpenCV](https://so.csdn.net/so/search?q=OpenCV&spm=1001.2101.3001.7020)中通过鼠标左键拖拽移动图片，通过鼠标滚轮前后旋转放大缩小图片。

**二、实现背景**

  在前一篇博客[在OpenCV中使用滚动条显示大图](https://www.jianshu.com/p/e0ab51b217bb)中发现自己画滚动条移动大图略显繁琐，且缩放图片存在一定不足，故尝试利用鼠标拖拽来实现图片移动，同时完善使用鼠标滚轮缩放图片的程序。

**三、实现方法**

  利用OpenCV中的鼠标事件实现。当鼠标左键点击时保存初始坐标，按住鼠标左键并拖拽后记下拖拽坐标，二者相减得出相对移动坐标，最后以此计算窗口相对于大图的坐标(wx, wy)，并根据窗口宽高显示截取的图片。



**四、运行环境**

  1. 在windows下已安装Python（必须的，我使用的是python3.7）;  
  2. 安装PyCharm（方便编程调试）  
  3. 安装opencv-python（在python下安装配置OpenCV）

**五、运行代码**

import cv2

def mouse(event, x, y, flags, param):

global flag, x1, y1, x2, y2, wx, wy, move\_w, move\_h, dst

global zoom, zoom\_w, zoom\_h, img\_zoom, flag\_har, flag\_var

if event == cv2.EVENT\_LBUTTONDOWN: # 左键点击

if flag == 0:

flag = 1

x1, y1, x2, y2 = x, y, wx, wy # 使鼠标移动距离都是相对于初始点击位置，而不是相对于上一位置

elif event == cv2.EVENT\_MOUSEMOVE and (flags & cv2.EVENT\_FLAG\_LBUTTON): # 按住左键拖曳

if flag == 1:

move\_w, move\_h = x1 - x, y1 - y # 鼠标拖拽移动的宽高

if flag\_har and flag\_var: # 当窗口宽高大于图片宽高

wx = x2 + move\_w # 窗口在大图的横坐标

if wx < 0: # 矫正位置

wx = 0

elif wx + win\_w > zoom\_w:

wx = zoom\_w - win\_w

wy = y2 + move\_h # 窗口在大图的总坐标

if wy < 0:

wy = 0

elif wy + win\_h > zoom\_h:

wy = zoom\_h - win\_h

dst = img\_zoom[wy:wy + win\_h, wx:wx + win\_w] # 截取窗口显示区域

elif flag\_har and flag\_var == 0: # 当窗口宽度大于图片宽度

wx = x2 + move\_w

if wx < 0:

wx = 0

elif wx + win\_w > zoom\_w:

wx = zoom\_w - win\_w

dst = img\_zoom[0:zoom\_h, wx:wx + win\_w]

elif flag\_har == 0 and flag\_var: # 当窗口高度大于图片高度

wy = y2 + move\_h

if wy < 0:

wy = 0

elif wy + win\_h > zoom\_h:

wy = zoom\_h - win\_h

dst = img\_zoom[wy:wy + win\_h, 0:zoom\_w]

elif event == cv2.EVENT\_LBUTTONUP: # 左键释放

flag = 0

x1, y1, x2, y2 = 0, 0, 0, 0

elif event == cv2.EVENT\_MOUSEWHEEL: # 滚轮

z = zoom

if flags > 0: # 滚轮上移

zoom += wheel\_step

if zoom > 1 + wheel\_step \* 20: # 缩放倍数调整

zoom = 1 + wheel\_step \* 20

else: # 滚轮下移

zoom -= wheel\_step

if zoom < wheel\_step: # 缩放倍数调整

zoom = wheel\_step

zoom = round(zoom, 2) # 取2位有效数字

zoom\_w, zoom\_h = int(img\_original\_w \* zoom), int(img\_original\_h \* zoom)

# print(wx, wy)

wx, wy = int((wx + x) \* zoom / z - x), int((wy + y) \* zoom / z - y) # 缩放后窗口在图片中的坐标

# print(z, zoom, x, y, wx, wy)

if wx < 0:

wx = 0

elif wx + win\_w > zoom\_w:

wx = zoom\_w - win\_w

if wy < 0:

wy = 0

elif wy + win\_h > zoom\_h:

wy = zoom\_h - win\_h

img\_zoom = cv2.resize(img\_original, (zoom\_w, zoom\_h), interpolation=cv2.INTER\_AREA) # 图片缩放

if zoom\_w <= win\_w and zoom\_h <= win\_h: # 缩放后图片宽高小于窗口宽高

flag\_har, flag\_var = 0, 0

dst = img\_zoom

cv2.resizeWindow('img', zoom\_w, zoom\_h)

elif zoom\_w <= win\_w and zoom\_h > win\_h: # 缩放后图片宽度小于窗口宽度

flag\_har, flag\_var = 0, 1

dst = img\_zoom[wy:wy + win\_h, 0:zoom\_w]

cv2.resizeWindow('img', zoom\_w, win\_h)

elif zoom\_w > win\_w and zoom\_h <= win\_h: # 缩放后图片高度小于窗口高度

flag\_har, flag\_var = 1, 0

dst = img\_zoom[0:zoom\_h, wx:wx + win\_w]

cv2.resizeWindow('img', win\_w, zoom\_h)

else: # 缩放后图片宽高大于于窗口宽高

flag\_har, flag\_var = 1, 1

dst = img\_zoom[wy:wy + win\_h, wx:wx + win\_w]

cv2.resizeWindow('img', win\_w, win\_h)

cv2.imshow("img", dst)

cv2.waitKey(1)

win\_h, win\_w = 600, 800 # 窗口宽高

wx, wy = 0, 0 # 窗口相对于原图的坐标

wheel\_step, zoom = 0.05, 1 # 缩放系数， 缩放值

img\_original = cv2.imread("E:/VSCode/git/my\_program/ImageMoveZoom/1.jpg") # 建议图片大于win\_w\*win\_h(800\*600)

img\_original\_h, img\_original\_w = img\_original.shape[0:2] # 原图宽高

cv2.namedWindow('img', cv2.WINDOW\_NORMAL)

cv2.moveWindow("img", 300, 100)

zoom\_w, zoom\_h = img\_original\_w, img\_original\_h # 缩放图宽高

img\_zoom = img\_original.copy() # 缩放图片

flag, flag\_har, flag\_var = 0, 0, 0 # 鼠标操作类型

move\_w, move\_h = 0, 0 # 鼠标移动坐标

x1, y1, x2, y2 = 0, 0, 0, 0 # 中间变量

cv2.resizeWindow("img", win\_w, win\_h)

dst = img\_original[wy:wy + win\_h, wx:wx + win\_w]

cv2.setMouseCallback('img', mouse)

if img\_original\_w > win\_w:

flag\_har = 1

if img\_original\_h > win\_h:

flag\_var = 1

cv2.waitKey()

cv2.destroyAllWindows()

2020.06.02版本：缩减全局变量个数，将关联较少的功能（如矫正坐标，计算缩放倍数）单独定义成函数，精炼鼠标事件函数。

import cv2

# 全局变量

g\_window\_name = "img" # 窗口名

g\_window\_wh = [800, 600] # 窗口宽高

g\_location\_win = [0, 0] # 相对于大图，窗口在图片中的位置

location\_win = [0, 0] # 鼠标左键点击时，暂存g\_location\_win

g\_location\_click, g\_location\_release = [0, 0], [0, 0] # 相对于窗口，鼠标左键点击和释放的位置

g\_zoom, g\_step = 1, 0.1 # 图片缩放比例和缩放系数

g\_image\_original = cv2.imread("E:/VSCode/git/my\_program/ImageMoveZoom/1.jpg") # 原始图片，建议大于窗口宽高（800\*600）

g\_image\_zoom = g\_image\_original.copy() # 缩放后的图片

g\_image\_show = g\_image\_original[g\_location\_win[1]:g\_location\_win[1] + g\_window\_wh[1], g\_location\_win[0]:g\_location\_win[0] + g\_window\_wh[0]] # 实际显示的图片

# 矫正窗口在图片中的位置

# img\_wh:图片的宽高, win\_wh:窗口的宽高, win\_xy:窗口在图片的位置

def check\_location(img\_wh, win\_wh, win\_xy):

for i in range(2):

if win\_xy[i] < 0:

win\_xy[i] = 0

elif win\_xy[i] + win\_wh[i] > img\_wh[i] and img\_wh[i] > win\_wh[i]:

win\_xy[i] = img\_wh[i] - win\_wh[i]

elif win\_xy[i] + win\_wh[i] > img\_wh[i] and img\_wh[i] < win\_wh[i]:

win\_xy[i] = 0

# print(img\_wh, win\_wh, win\_xy)

# 计算缩放倍数

# flag：鼠标滚轮上移或下移的标识, step：缩放系数，滚轮每步缩放0.1, zoom：缩放倍数

def count\_zoom(flag, step, zoom):

if flag > 0: # 滚轮上移

zoom += step

if zoom > 1 + step \* 20: # 最多只能放大到3倍

zoom = 1 + step \* 20

else: # 滚轮下移

zoom -= step

if zoom < step: # 最多只能缩小到0.1倍

zoom = step

zoom = round(zoom, 2) # 取2位有效数字

return zoom

# OpenCV鼠标事件

def mouse(event, x, y, flags, param):

global g\_location\_click, g\_location\_release, g\_image\_show, g\_image\_zoom, g\_location\_win, location\_win, g\_zoom

if event == cv2.EVENT\_LBUTTONDOWN: # 左键点击

g\_location\_click = [x, y] # 左键点击时，鼠标相对于窗口的坐标

location\_win = [g\_location\_win[0], g\_location\_win[1]] # 窗口相对于图片的坐标，不能写成location\_win = g\_location\_win

elif event == cv2.EVENT\_MOUSEMOVE and (flags & cv2.EVENT\_FLAG\_LBUTTON): # 按住左键拖曳

g\_location\_release = [x, y] # 左键拖曳时，鼠标相对于窗口的坐标

h1, w1 = g\_image\_zoom.shape[0:2] # 缩放图片的宽高

w2, h2 = g\_window\_wh # 窗口的宽高

show\_wh = [0, 0] # 实际显示图片的宽高

if w1 < w2 and h1 < h2: # 图片的宽高小于窗口宽高，无法移动

show\_wh = [w1, h1]

g\_location\_win = [0, 0]

elif w1 >= w2 and h1 < h2: # 图片的宽度大于窗口的宽度，可左右移动

show\_wh = [w2, h1]

g\_location\_win[0] = location\_win[0] + g\_location\_click[0] - g\_location\_release[0]

elif w1 < w2 and h1 >= h2: # 图片的高度大于窗口的高度，可上下移动

show\_wh = [w1, h2]

g\_location\_win[1] = location\_win[1] + g\_location\_click[1] - g\_location\_release[1]

else: # 图片的宽高大于窗口宽高，可左右上下移动

show\_wh = [w2, h2]

g\_location\_win[0] = location\_win[0] + g\_location\_click[0] - g\_location\_release[0]

g\_location\_win[1] = location\_win[1] + g\_location\_click[1] - g\_location\_release[1]

check\_location([w1, h1], [w2, h2], g\_location\_win) # 矫正窗口在图片中的位置

g\_image\_show = g\_image\_zoom[g\_location\_win[1]:g\_location\_win[1] + show\_wh[1], g\_location\_win[0]:g\_location\_win[0] + show\_wh[0]] # 实际显示的图片

elif event == cv2.EVENT\_MOUSEWHEEL: # 滚轮

z = g\_zoom # 缩放前的缩放倍数，用于计算缩放后窗口在图片中的位置

g\_zoom = count\_zoom(flags, g\_step, g\_zoom) # 计算缩放倍数

w1, h1 = [int(g\_image\_original.shape[1] \* g\_zoom), int(g\_image\_original.shape[0] \* g\_zoom)] # 缩放图片的宽高

w2, h2 = g\_window\_wh # 窗口的宽高

g\_image\_zoom = cv2.resize(g\_image\_original, (w1, h1), interpolation=cv2.INTER\_AREA) # 图片缩放

show\_wh = [0, 0] # 实际显示图片的宽高

if w1 < w2 and h1 < h2: # 缩放后，图片宽高小于窗口宽高

show\_wh = [w1, h1]

cv2.resizeWindow(g\_window\_name, w1, h1)

elif w1 >= w2 and h1 < h2: # 缩放后，图片高度小于窗口高度

show\_wh = [w2, h1]

cv2.resizeWindow(g\_window\_name, w2, h1)

elif w1 < w2 and h1 >= h2: # 缩放后，图片宽度小于窗口宽度

show\_wh = [w1, h2]

cv2.resizeWindow(g\_window\_name, w1, h2)

else: # 缩放后，图片宽高大于窗口宽高

show\_wh = [w2, h2]

cv2.resizeWindow(g\_window\_name, w2, h2)

g\_location\_win = [int((g\_location\_win[0] + x) \* g\_zoom / z - x), int((g\_location\_win[1] + y) \* g\_zoom / z - y)] # 缩放后，窗口在图片的位置

check\_location([w1, h1], [w2, h2], g\_location\_win) # 矫正窗口在图片中的位置

# print(g\_location\_win, show\_wh)

g\_image\_show = g\_image\_zoom[g\_location\_win[1]:g\_location\_win[1] + show\_wh[1], g\_location\_win[0]:g\_location\_win[0] + show\_wh[0]] # 实际的显示图片

cv2.imshow(g\_window\_name, g\_image\_show)

# 主函数

if \_\_name\_\_ == "\_\_main\_\_":

# 设置窗口

cv2.namedWindow(g\_window\_name, cv2.WINDOW\_NORMAL)

# 设置窗口大小，只有当图片大于窗口时才能移动图片

cv2.resizeWindow(g\_window\_name, g\_window\_wh[0], g\_window\_wh[1])

cv2.moveWindow(g\_window\_name, 700, 100) # 设置窗口在电脑屏幕中的位置

# 鼠标事件的回调函数

cv2.setMouseCallback(g\_window\_name, mouse)

cv2.waitKey() # 不可缺少，用于刷新图片，等待鼠标操作

cv2.destroyAllWindows()

**六、运行结果**

原图：



水平移动图片（鼠标左键点击并向左拖拽）：



垂直移动图片（鼠标左键点击，向上拖拽）：



鼠标滚轮后转缩小图片：



鼠标滚轮前转放大图片：



**七、参考资料**

上一篇：[在OpenCV中使用滚动条显示大图](https://www.jianshu.com/p/e0ab51b217bb)