# -\*- coding: utf-8 -\*-

"""pick-points.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1djJ1ipsZYF3WugFC\_M9tVgjI9xhMRuDO

"""

import os

import imageio

import argparse

import numpy as np

#############################################################################

# TODO: Add additional imports

#############################################################################

import matplotlib.pyplot as plt

from matplotlib.widgets import Cursor

#############################################################################

# END OF YOUR CODE #

#############################################################################

fig, axes = plt.subplots(1, 2, sharey=True)

points = [[],[]]

def get\_parser():

parser = argparse.ArgumentParser(description="Points Selection")

parser.add\_argument("image1", type=str, help="path to image 1")

parser.add\_argument("image2", type=str, help="path to image 2")

return parser

def pick\_points(img1, img2):

"""

Functionality to get manually identified corresponding points from two views.

Inputs:

- img1: The first image to select points from

- img2: The second image to select points from

Output:

- coords1: An ndarray of shape (N, 2) with points from image 1

- coords2: An ndarray of shape (N, 2) with points from image 2

"""

############################################################################

# TODO: Implement pick\_points

############################################################################

axes[0].imshow(img1)

axes[0].set\_title("IMG1")

axes[1].imshow(img2)

axes[1].set\_title("IMG2")

cursor1 = Cursor(axes[0], useblit=True, color='red', linewidth=2)

cursor2 = Cursor(axes[1], useblit=True, color='red', linewidth=2)

cid = fig.canvas.mpl\_connect('button\_press\_event', onclick)

plt.show() # stalls until plot closed

points1 = None

points2 = None

if (len(points[0])) > 0:

points1 = np.stack(points[0])

if (len(points[0])) > 0:

points2 = np.stack(points[1])

return points1, points2

def onclick(event):

'''

Event handler for button\_press\_event

@param event MouseEvent

'''

print("clicked")

if (event.inaxes is axes[0]):

points[0].append(np.array([event.xdata,event.ydata]))

if (event.inaxes is axes[1]):

points[1].append(np.array([event.xdata,event.ydata]))

# Visualization of Points

if (len(points[0])) > 0:

points1 = np.stack(points[0])

axes[0].scatter(points1[:,0], points1[:,1], color='blue')

if (len(points[1])) > 0:

points2 = np.stack(points[1])

axes[1].scatter(points2[:,0], points2[:,1], color='blue')

############################################################################

# END OF YOUR CODE

############################################################################

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

args = get\_parser().parse\_args()

img1 = np.asarray(imageio.imread(args.image1))

img2 = np.asarray(imageio.imread(args.image2))

coords1, coords2 = pick\_points(img1, img2)

assert len(coords1) == len(coords2), "The number of coordinates does not match"

filename1 = os.path.splitext(args.image1)[0] + ".npy"

filename2 = os.path.splitext(args.image2)[0] + ".npy"

assert not os.path.exists(filename1), f"Output file {filename1} already exists"

assert not os.path.exists(filename2), f"Output file {filename2} already exists"

np.save(filename1, coords1)

np.save(filename2, coords2)