Report

April 14, 2019

1 Optical Flow using Lucas Kanade Method

1.0.1 Assignment 4

Shubh Maheshwari 20161170 In this assignment our goal is to apply optical flow for tracking objects in a moving scene

The method works by finding movement of pixels in the 2 frames. Here we find the least square solution on a vector using the gradient in the x&y direaction to predicts the movement of the pixels

We are given the a dataset containg 2 & 8 frames.

```
In [2]: # Imports
        import os
        import cv2
        import numpy as np
        from scipy import signal
        import matplotlib.pyplot as plt

        %matplotlib inline
```

```
In [48]: # Helper functions
         def display_animation(frame_list):
             pass

         def display_images(img_list, shape,fig_size=(8,8),tile=None,is_gray=None):
             """
                 Display multple images using matplotlib
                 @param img_list:=> mxn matrix of images to be displayed
                 @param shape:=> mxn shape
                 @param is_gray:=> mxn matrix, is the i,j th the image grayscaled

                 return None
             """

             if is_gray is None:
                 is_gray = np.zeros(shape)
```

```
    m,n = shape
    fig = plt.figure(figsize=fig_size)

    for i in range(m):
        for j in range(n):
            ax = fig.add_subplot(m,n,i*n + j+1)
            if is_gray[i,j] == 1:
                ax.imshow(img_list[i][j],cmap='gray')
            else:
                img_list[i][j] = cv2.resize(img_list[i][j],(200,200))
                ax.imshow(img_list[i][j])
            ax.axis('off')
    plt.show()
    return

def display_opticalflow_results(im1,im2,u,v,fig_size=(16,16),arrow_thres=0.02):
    """
        Display results of optical flow
        @param img_list:=> mxn matrix of images to be displayed
        @param shape:=> mxn shape
        @param is_gray:=> mxn matrix, is the i,j th the image grayscaled

        return None
    """

    fig = plt.figure(figsize=fig_size)

    #     Images
    ax = fig.add_subplot(3,2,1)
    ax.imshow(im1,cmap='gray')
    ax.set_title("Image T:1")
    ax.axis('off')

    ax = fig.add_subplot(3,2,2)
    ax.imshow(im2,cmap='gray')
    ax.set_title("Image T:2")
    ax.axis('off')

    # Vectors
    ax = fig.add_subplot(3,2,3)
    ax.imshow(u,cmap='gray')
    ax.set_title("U")
    ax.axis('off')

    ax = fig.add_subplot(3,2,4)
    ax.imshow(v,cmap='gray')
    ax.set_title("V")
    ax.axis('off')
```

```
    # Magnitude
    ax = fig.add_subplot(3,2,5)
    ax.imshow(u*u + v*v,cmap='gray')
    ax.set_title("U^2 + V^2")
    ax.axis('off')

    # Angle
    ax = fig.add_subplot(3,2,6)
    ax.imshow(np.arctan2(v,u),cmap='gray')
    ax.set_title("arc(v/u)")
    ax.axis('off')

    fig = plt.figure(figsize=fig_size)
    ax = fig.add_subplot(1,2,1)
    ax.imshow(im1,cmap='gray')
    ax.set_title("Optical flow Arrows")

#     arrow_ind_y,arrow_ind_x = np.where(u*u + v*v > arrow_thres)
    kp = cv2.goodFeaturesToTrack(im1, 100, 0.01, 10, 3)
    for arrow_ind in kp:
        x,y = arrow_ind[0]
        y = int(y)
        x = int(x)
        ax.arrow(x,y,u[y,x],v[y,x],head_width = 1, head_length = 5, color = (0,1,0))

    ax = fig.add_subplot(1,2,2)
    ax.imshow( (u*u + v*v>arrow_thres),cmap='gray')
    ax.set_title("Optical flow Mask")

    ax.axis('off')


    plt.show()

    return None
# 2d Convolution
def conv2d(a, f,pad=True):
    """
        Run 2d Convolution on a Matrix using the filter f
        @param a:=> 2d Matrix
        @param f:=> filter (list or numpy array)

        return convolved matrix
    """
    f = np.array(f)
    s = f.shape + tuple(np.subtract(a.shape, f.shape) + 1)
    strd = np.lib.stride_tricks.as_strided
```