CS512 Assignment 6: Optical Flow

Adithya Sreenath Chandrashekarapuram

Department of Computer Science

Illinois Institute of Technology

November 26, 2017

Abstract:

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second.

Implementation:

1. First we take the images and apply Gaussian filter of size 3x3 to eliminate noise:

```
I1 = np.array(Image1)
I2 = np.array(Image2)
S = np.shape(I1)
I1_smooth = cv2.GaussianBlur(I1, (3,3), 0)
I2_smooth = cv2.GaussianBlur(I2, (3,3), 0)
```

2. We then find the derivative filters:

```
Ix = signal.convolve2d(I1_smooth,[[-0.25,0.25],[-0.25,0.25]],'same') +
signal.convolve2d(I2_smooth,[[-0.25,0.25],[-0.25,0.25]],'same')

Iy = signal.convolve2d(I1_smooth,[[-0.25,-0.25],[0.25,0.25]],'same') +
signal.convolve2d(I2_smooth,[[-0.25,-0.25],[0.25,0.25]],'same')

It = signal.convolve2d(I1_smooth,[[0.25,0.25],[0.25,0.25]],'same') +
signal.convolve2d(I2_smooth,[[-0.25,-0.25],[-0.25,-0.25]],'same')
```

3. We find the feature vectors:

```
features = cv2.goodFeaturesToTrack(I1_smooth,10000,0.01,10)
  feature = np.int0(features)
  for i in feature:
          x,y = i.ravel()
          cv2.circle(I1_smooth,(x,y),3,0,-1)
```

4. We then create the u and v vector for good features obtained and find the derivates for the neighbouring pixels:

```
u = v = np.nan*np.ones(S)

for l in feature:
          j,i = l.ravel()
          IX = ([Ix[i-1,j-1],Ix[i,j-1],Ix[i-1,j-1],Ix[i-1,j],Ix[i,j],Ix[i+1,j],Ix[i-
1,j+1],Ix[i,j+1],Ix[i+1,j-1]])
```

```
IY = ([Iy[i-1,j-1],Iy[i,j-1],Iy[i-1,j-1],Iy[i-1,j],Iy[i,j],Iy[i+1,j],Iy[i-
1,j+1],Iy[i,j+1],Iy[i+1,j-1]])
IT = ([It[i-1,j-1],It[i,j-1],It[i-1,j-1],It[i-1,j],It[i,j],It[i+1,j],It[i-1,j+1],It[i,j+1],It[i+1,j-1]])
```

Here we use the minimum least squares solution approach:

```
LK = (IX, IY)
LK = np.matrix(LK)
LK_T = np.array(np.matrix(LK))
LK = np.array(np.matrix.transpose(LK))
A1 = np.dot(LK_T,LK)
A2 = np.linalg.pinv(A1)
A3 = np.dot(A2,LK_T)
(u[i,j],v[i,j]) = np.dot(A3,IT)
```

5. Finally we plot on the image and display it to the user:

```
plt.subplot(1,1,1)
  plt.title('Optical Flow')
  plt.imshow(I1,cmap = cm.gray)
  for i in range(S[0]):
        for j in range(S[1]):
              if abs(u[i,j])>t or abs(v[i,j])>t:
                    plt.arrow(j,i,v[i,j],u[i,j],head_width = 5, head_length = 5,
color = "b")
  plt.show()
```

Manual:

There are 2 files provided. The MyLucaskanade.py file and the VideoLK.py file.

MyLucasKanade.py:

- Run the program to get the plot which has the LK algorithm implemented on a set of 2 images.
- The algorithm takes too long and cannot run on a video as the frames change too fast before the code can compute the output and therefore has been written to take 2 images as inputs and display the output.
- The images can be changed as desired to find the optical flow for different images.

VideoLK.py:

- This program consists of the required tasks but uses the opencv function to implement.
- Pressing ‘p’ will pause the video.
- Pressing ‘ESC’ will close the program.

RESULT

INPUT:

Image 1:

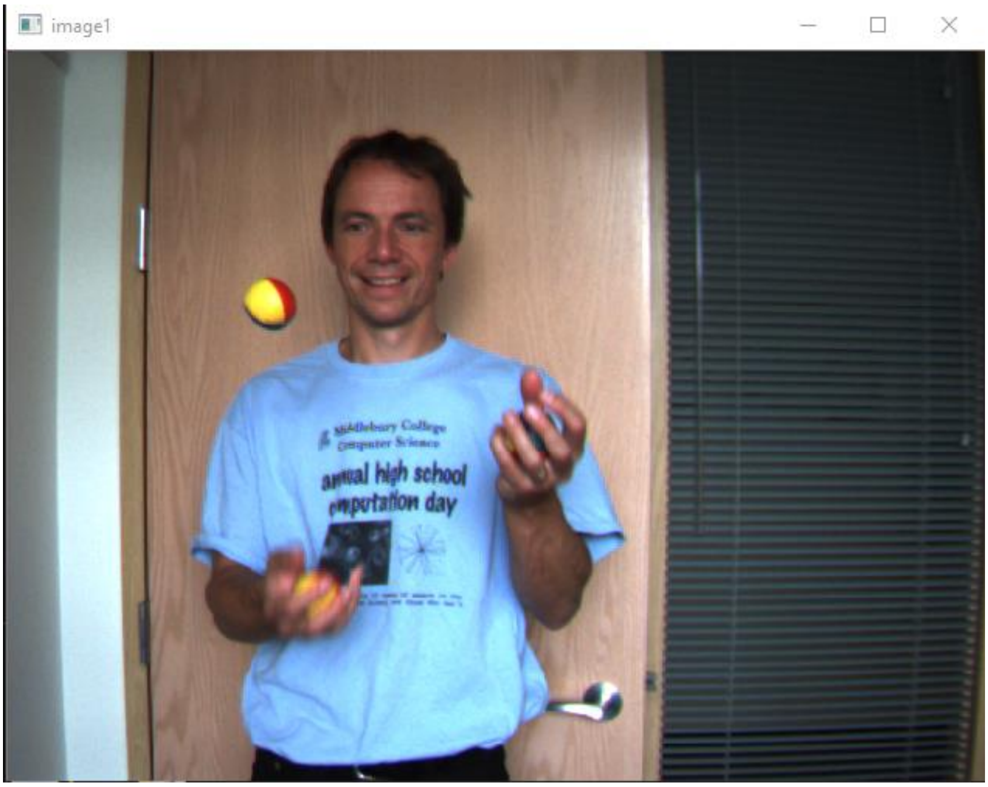
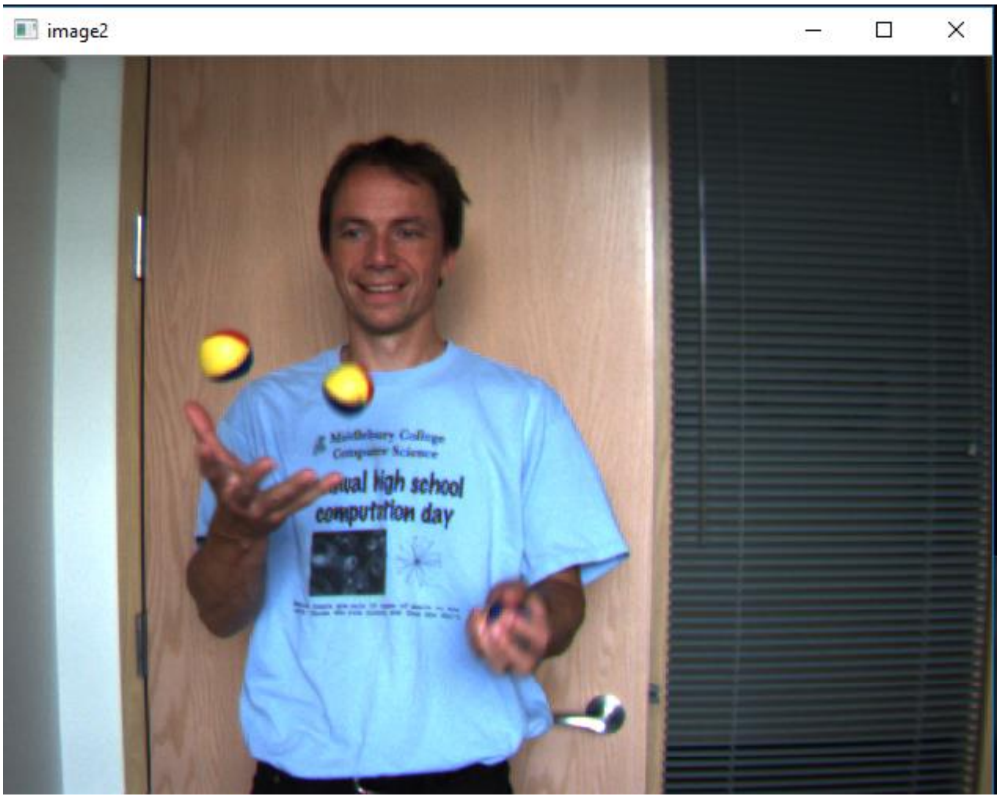
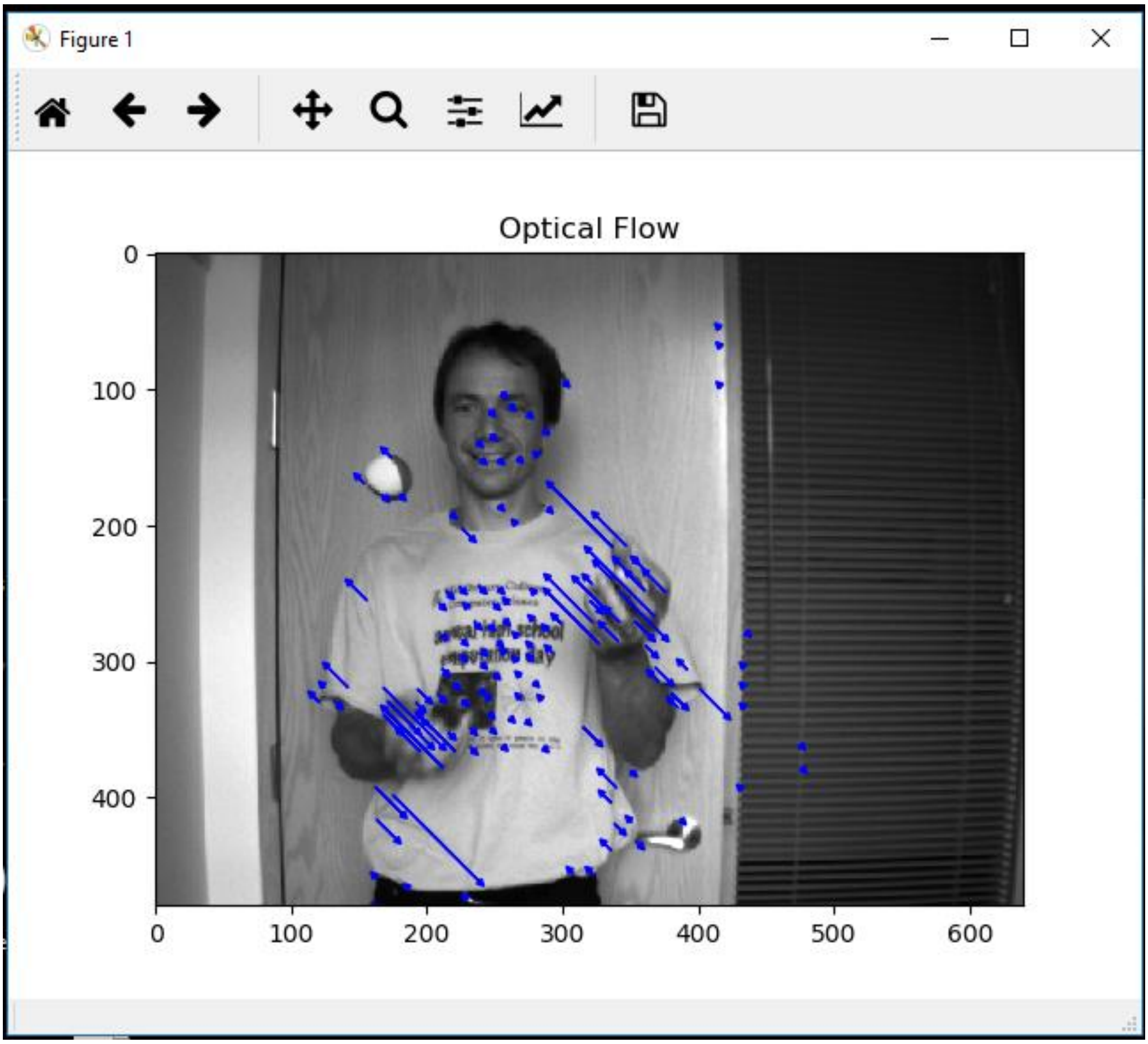


Image 2:



Output:



References:

https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html

https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method

https://en.wikipedia.org/wiki/Optical_flow

http://vision.middlebury.edu/flow/