Motion estimation report

Name:-Sai Venkat

Motion estimation models are used in computer vision and video processing to estimate the motion between consecutive frames in a video sequence. The goal is to identify how each pixel has moved between two frames, which is useful for tasks such as video compression, object tracking, and motion-based segmentation.

There are various types of Motion estimation models, but today we are going to see three types of models which is used for motion estimation.

1. Lucas Kanade method
2. Horn Schunck method
3. Optical flow using RAFT method(CNN based model)

So let's see what **lucas kanada method** works

Lucas Kanade Method is based on something known as Brightness constancy assumption. The key idea here is that pixel level brightness won't change a lot in just one frame. It assumes that the colour of an object does not change significantly and significantly in the previous two frames.

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

Brightness constancy constraint

$$I_t + I_x u + I_y v = 0$$

Simplified Brightness constancy constraint

Optical flow is only valid in regions where

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{pmatrix}$$

Implementation of lucas kanada method:-

First we need to grayscale the image to get even brightness so that model can observe the change in brightness.

<matplotlib.image.AxesImage at 0x7fae2d4d6280>

Next step would be Robot & Laplacian of Gaussian Mask for Image Derivatives

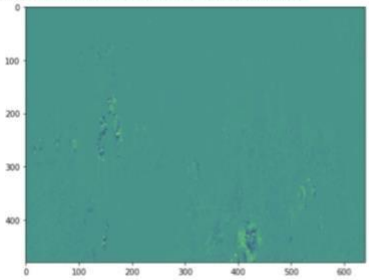
```
[[0.08333333 0.16666667 0.08333333]
 [0.16666667 0.         0.16666667]
 [0.08333333 0.16666667 0.08333333]]
```

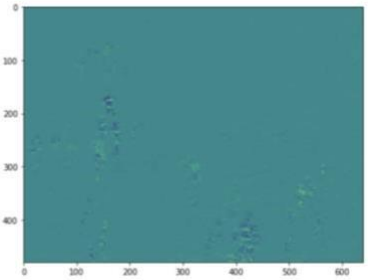
Remove the noise using **Gaussian Blur**

<matplotlib.image.AxesImage at 0x7fae2a40b1f0>

Calculate U , V using Lucas Kanade Algorithm

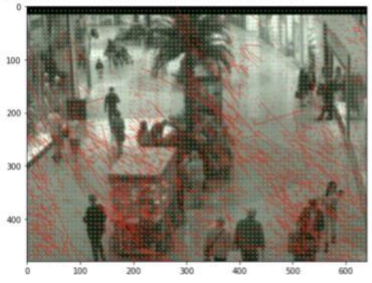
<matplotlib.image.AxesImage at 0x7fae2d2001f0>



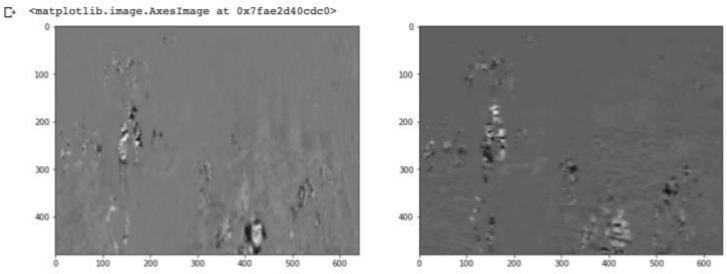


After total process output of **Lucas Kanade Algorithm**

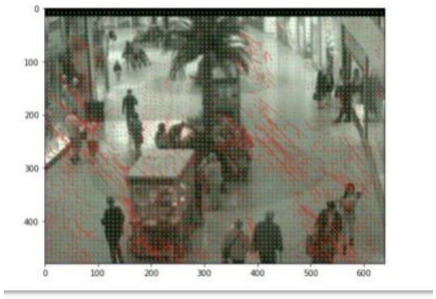
<matplotlib.image.AxesImage at 0x7fae29686370>



As same we are going to calculate the u , v using **horn schunck method**



After total process output of **horn schunck method**



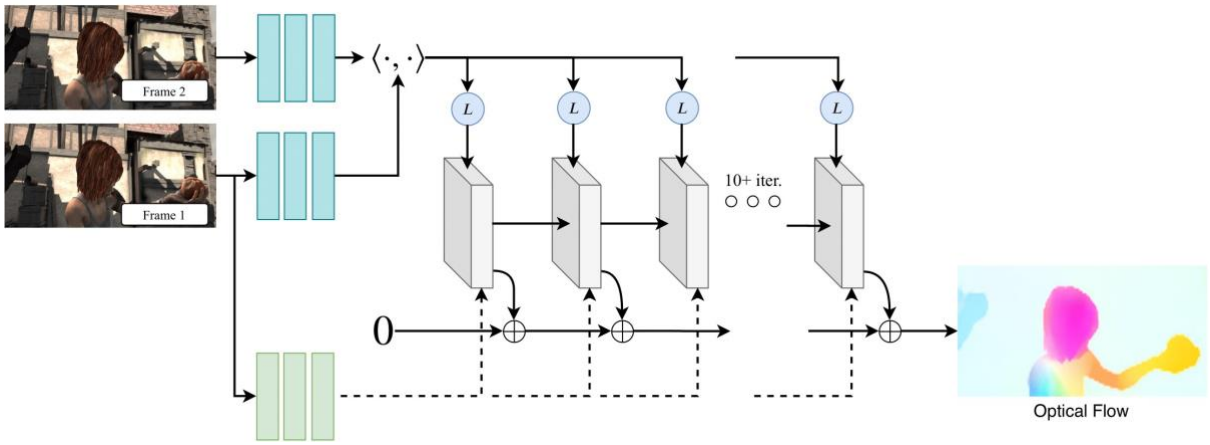
Now special method and my favourite method **RAFT**

RAFT: Recurrent All-Pairs Field Transforms for Optical Flow which is originally introduced in ECCV2020 by Teed et. al. in Princeton University and prized Best Paper Award.

Briefly, RAFT has below features

- Recurrent optical flow estimation
- Compute pixel-wise correlation between pair-wise input images and reuse it in the following recurrent step
- Lightweight, rapid inference, and high accuracy

Methodology of RAFT:-



This is a CNN-based method, as we all know how CNN is used for image classification now a days; if that's the main idea of this method, why can't we use it for motion estimation?

So this method is introduced for motion estimation.

This is the CNN of the RAFT

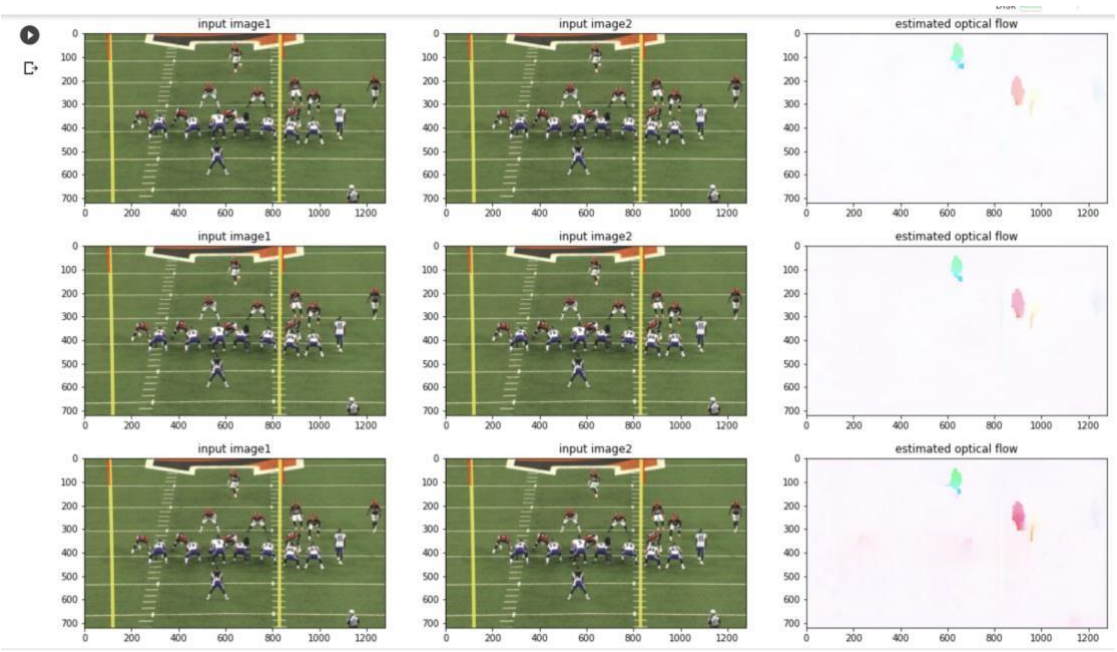
```
RAFT(
  (fnet): BasicEncoder(
    (norm1): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3))
    (relu1): ReLU(inplace=True)
    (layer1): Sequential(
      (0): ResidualBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (relu): ReLU(inplace=True)
        (norm1): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm2): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      )
      (1): ResidualBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (relu): ReLU(inplace=True)
        (norm1): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm2): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      )
    )
    (layer2): Sequential(
      (0): ResidualBlock(
        (conv1): Conv2d(64, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (conv2): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (relu): ReLU(inplace=True)
        (norm1): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm2): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm3): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (downsample): Sequential(
          (0): Conv2d(64, 96, kernel_size=(1, 1), stride=(2, 2))
          (1): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        )
      )
      (1): ResidualBlock(
        (conv1): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (relu): ReLU(inplace=True)
        (norm1): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm2): InstanceNorm2d(96, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
      )
    )
    (layer3): Sequential(
      (0): ResidualBlock(
        (conv1): Conv2d(96, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (relu): ReLU(inplace=True)
        (norm1): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm2): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (norm3): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        (downsample): Sequential(
          (0): Conv2d(96, 128, kernel_size=(1, 1), stride=(2, 2))
          (1): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
        )
      )
      (1): ResidualBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

As it is CNN I used Cuda for fast computation

Now let's see the **output of RAFT**

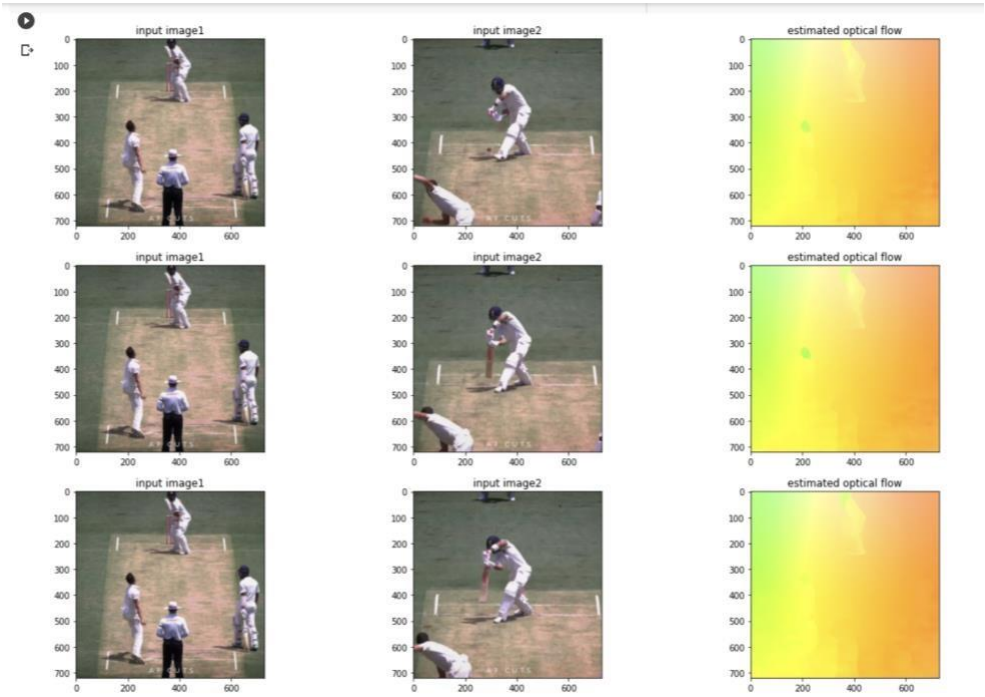
We see the densely coloured objects in the estimated optical flow that the motion detected by RAFT
As we can directly upload video and can check the motion estimation using RAFT

Output for the video input:-

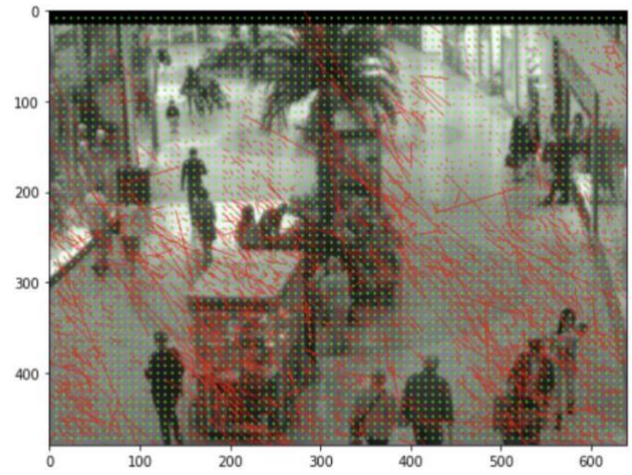


Let's see another video input:-

This video input contains Virat Kohli playing his significant shot, which is a cover drive, so let's see the motion estimation for that.

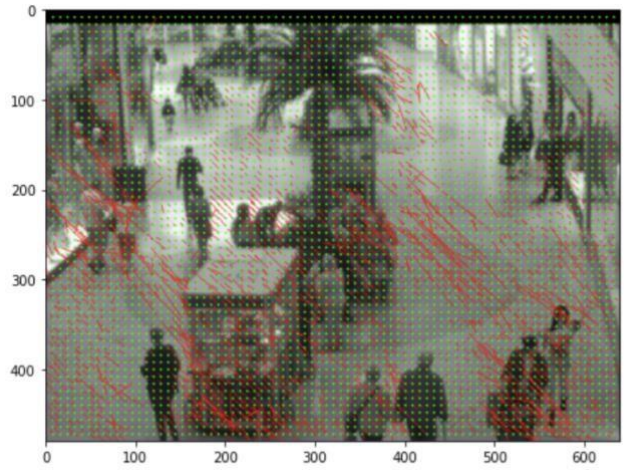


First let’s see the outputs of 3 models

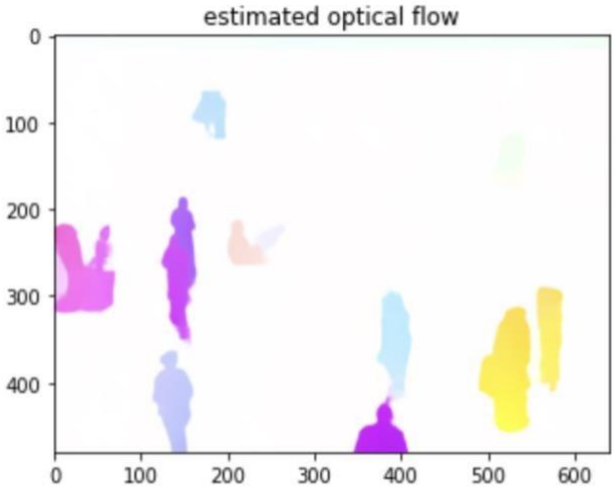


Lucas Kanade method

Horn Schunck method



RAFT method



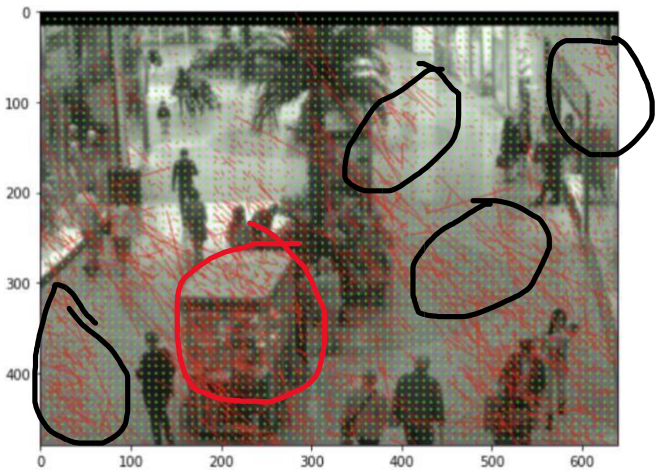
First let's see the input frames:-

Previous frame:-

Current frame:-

Compare and contrast the effectiveness of models:-

Lucas-Kanade is a feature-based method that tracks the movement of specific image features, such as corners or edges.

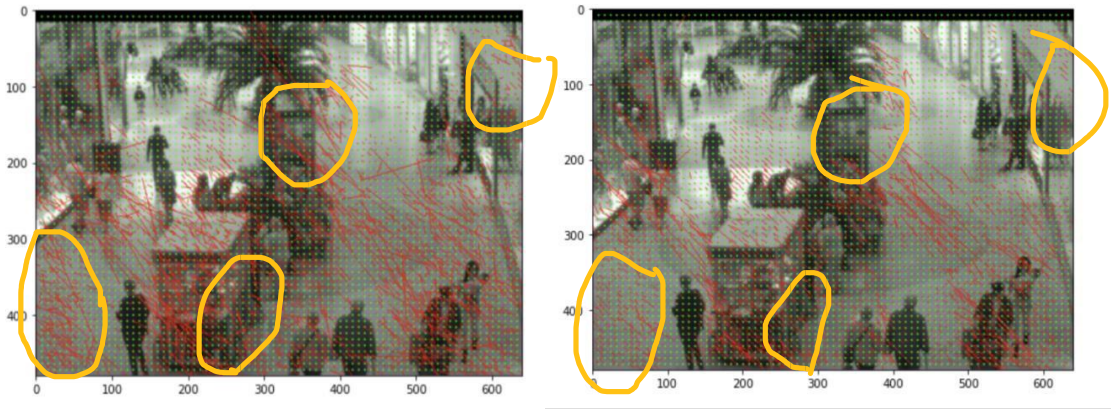


here we can see it is detecting the small movement in the game box available in the place which is a vending machine used to play which contain a movable picker.

Disadvantage of Lucas-Kanade is it is sensitive to noise as we see there is no movements which is circled by black round but it show the optical flow there.

As we see that Lucas-Kanade is sensitive to noise but Horn-Suchunk is best solution for it.

Horn-Schunck is an optical flow-based method that estimates the motion of every pixel in an



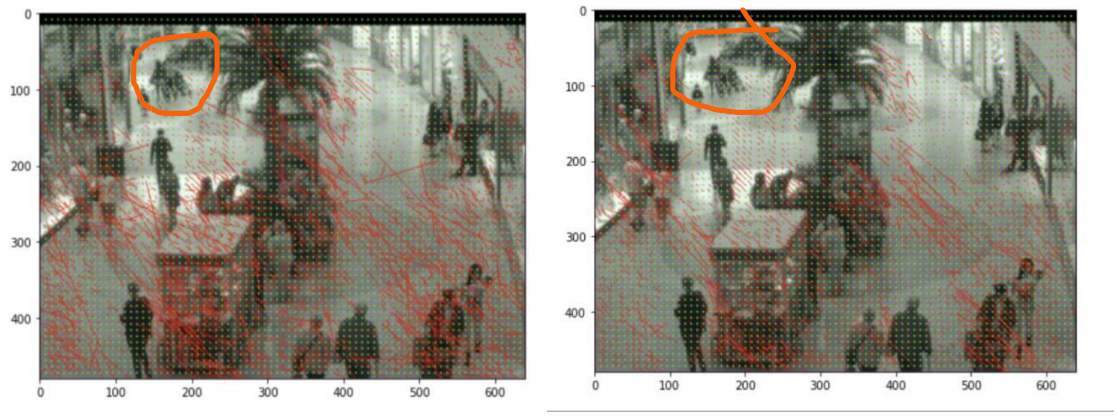
Here main advantage of Horn Schunck is it is not sensitive to noise.

This yellow rounded show you how Horn Schunck avoid noise in the data.

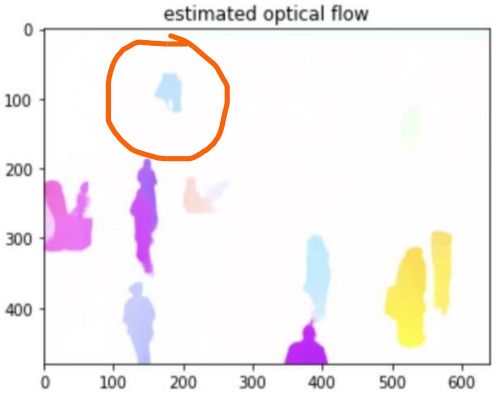
Now it’s time to compare and contrast the RAFT.

As we see it is not same like other methods , it detection the motion and it is in the form of density based.

As we see there is a movement in the frames where a couple passes the lady how is using trolly to move his baby but in results of lucas and horn there is no sign of it.

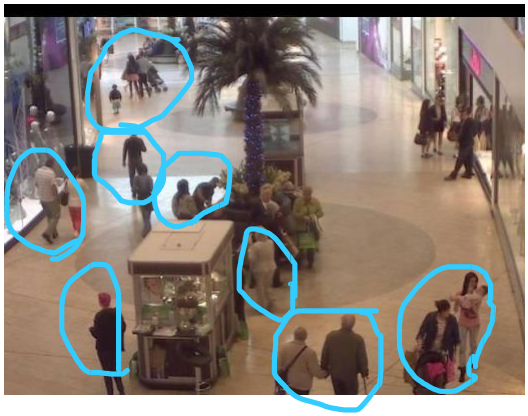


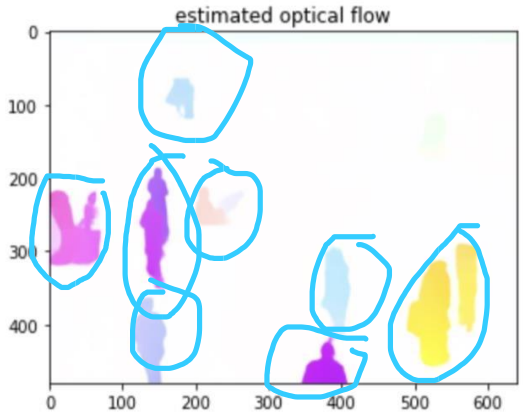
But is it detected by RAFT.



so by this we can say that RAFT is the best motion estimation model as we see that lucas is noise sensitive and horn is not able to detected the motion of complex movement which it cannot detect the movement of the couple.

Movements in the current frame:-





RAFT give output exactly same which we can say from the above inference that RAFT is best motion estimation model compare to other estimation models , but we cannot say lucas and horn are worst as motion estimation depends on the specific requirements of our application, such as the type of motion, computational resources, and accuracy requirements.

Implementation of code is available submitted in Moodle as IPYNB file.