README.md

Project Report

Team Members (Group 3)

- Ravi Rahar (19/11/EC/014)
- Asad Nizami (19/11/EC/013)
- Umesh (19/11/EC/015)
- Harsh (19/11/EC/016)
- Nitish (19/11/EC/012)

Procedure

- First we find frame by frame optical flow of the video is computed. We used `cv2.calcOpticalFlowFarneb` algorith to calculate dense optical flow.
- Then we convert it into HSV image to see the optical flow obtained. We can calculate average optical flow average of this flow over time.
- We can plot them using quiver plot.
- Using the optical flow, a set of particles are then moved over the frame to construct the streaklines.
- These are then used to compute similarity in a 8-connectivity neighborhood.
- And then a watershed is taken to separate motion from rest of the image

Structures Obtained

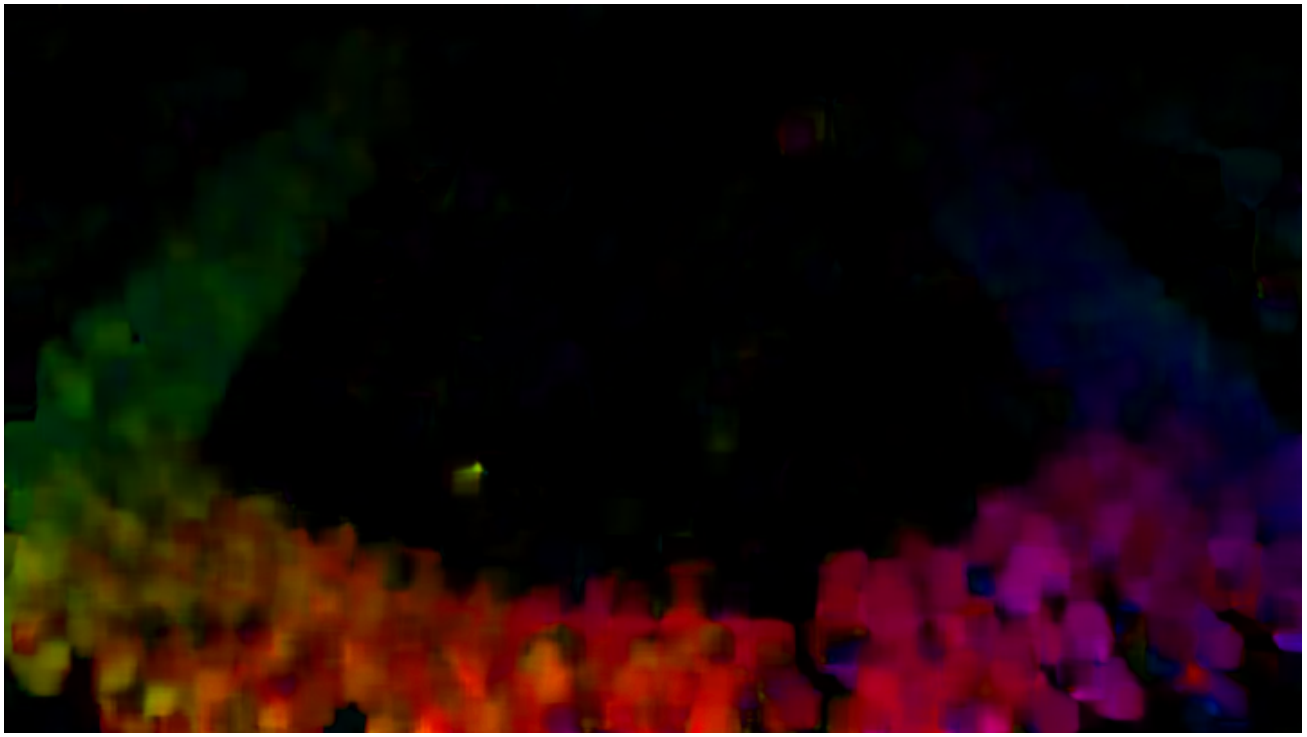
- After Optical flow
  After calculating optical flow we obtain a 3D matric of shape `(video_height, video_width, 2)` . It stores obtained and which is then used as a look up table for where each particle would have moved.
- After StreakLines We store streaklines in a 4D matrix of shape `(total_no_of_frames, video_height, v` x and y co-ordinates will then give you a 2D array which stores stores `x` and `y` co-ordinate values of all streakline. It looks like:

X	**x1**	**x2**	**x3**	**.....total_no_of_frames....**	**xn**
Y	y1	y2	y3	.....total_no_of_frames....	xn

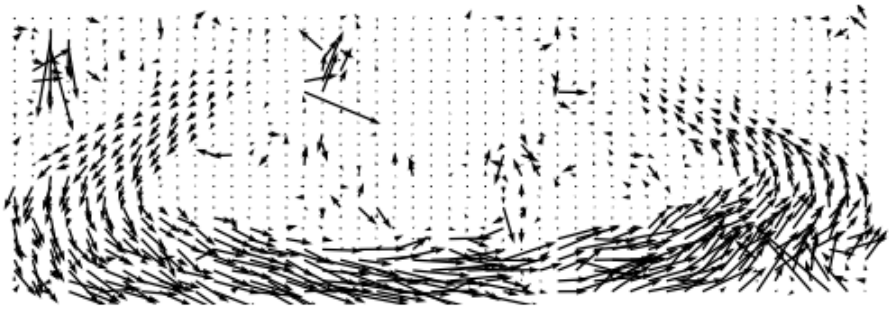
Results

- Original Image

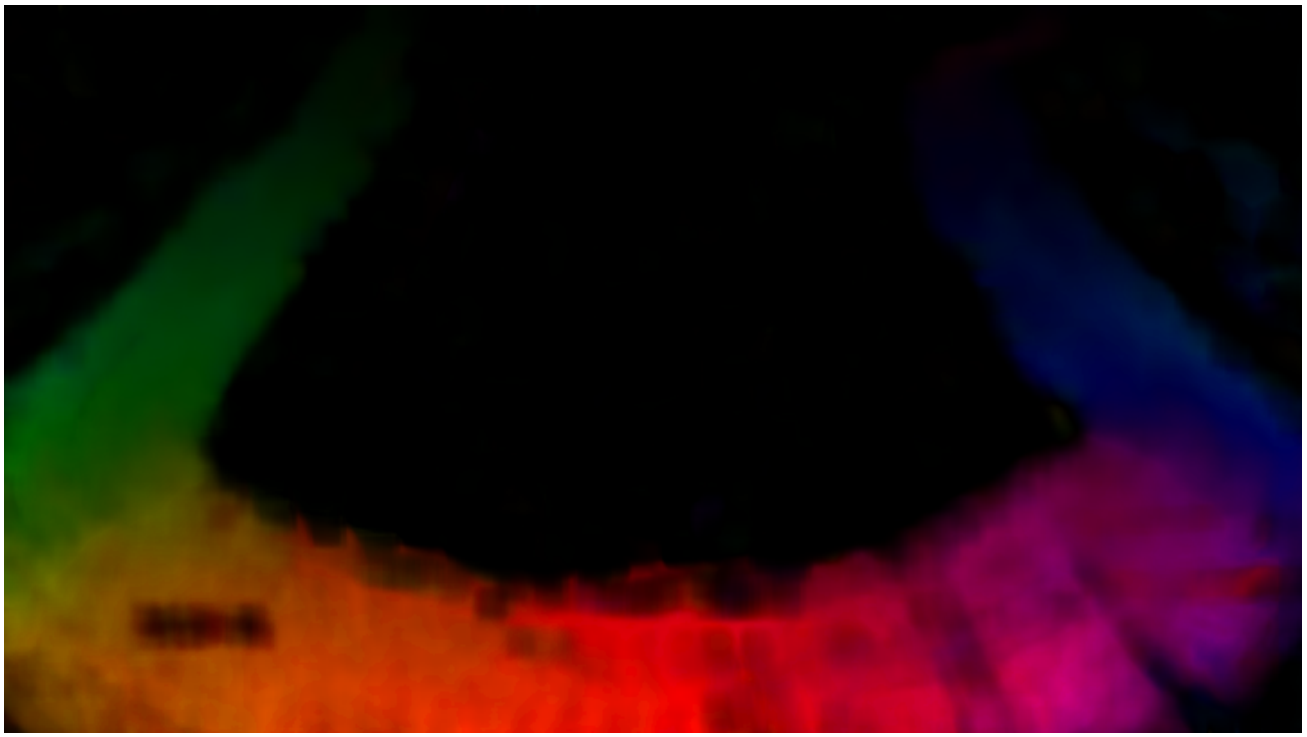
- Dense Optical Flow



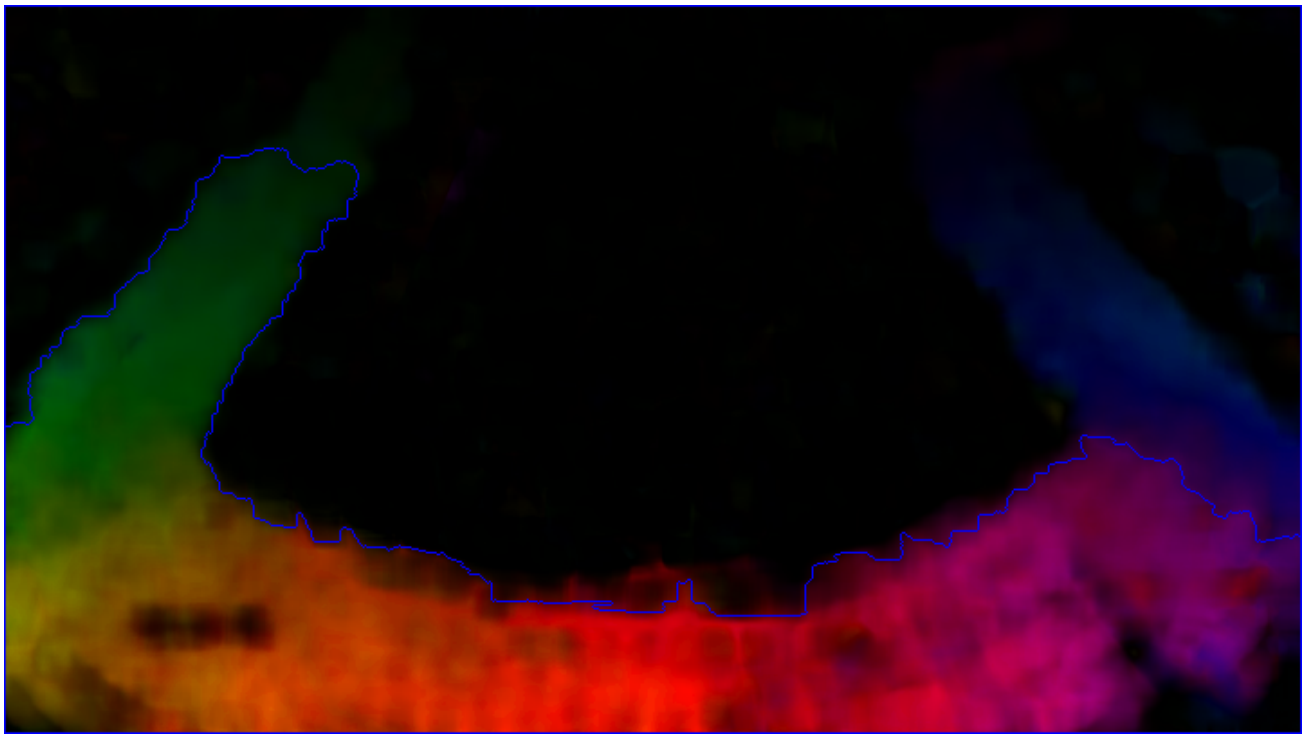
- Streaklines



- Similarity



- Watershed



Running for first time

```
cd CrowdFlowDetection
python main.py
```

NOTE:

- Do not foget to change video file path in `main.py` to video you want to run it on.
- Also change different function to obtain results at different steps.

References

Implementing crowd flow detection using steaklines. Based on:

```
A Streakline Representation of Flow in Crowded Scenes by Ramin Mehran†, Brian E. Moore‡, Mubara
```