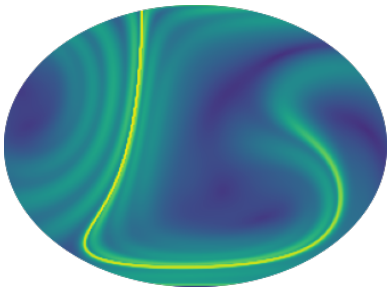
Exploration of LCS Theory on Crowd Video Analysis

Bryan Cohen



Department of Mathematics
Durham University
United Kingdom
April 26th 2024

Plagiarism Declaration

This piece of work is a result of my own and I have complied with the Department's guidance on multiple submission and on use of AI tools. Material from the work of others not involved in the project has been acknowledged, quotations and paraphrases suitably indicated, and all uses of AI tools have been declared.

Acknowledgements

I would like to thank my Academic Supervisor Dr Hossein Amini-Kafiabad for his invaluable support on this project and the many times that he was called upon to have impromptu meetings with me. I would like to thank my two dear friends for supporting me through this journey. I would also like to thank my mother for supporting me with food and my father for continually reminding me of the importance of this project. Finally, I thank my closest friend who without, this project would not be concluded.

Contents

Chapter 1

Introduction

1.1 Lagrangian Coherent Structures

Dynamical Systems are one of the most ubiquitous systems that appear in the real world. We see their existence in the flows of ocean currents, pendulums in clocks or orbital paths of the solar system. They describe some of the most fundamental motion patterns in this world. Dynamical Systems Theory is then the mathematics that describes the underlying mechanics of these systems. One of the central problems of Dynamical Systems theory is to find out how stable a particular dynamical system is.

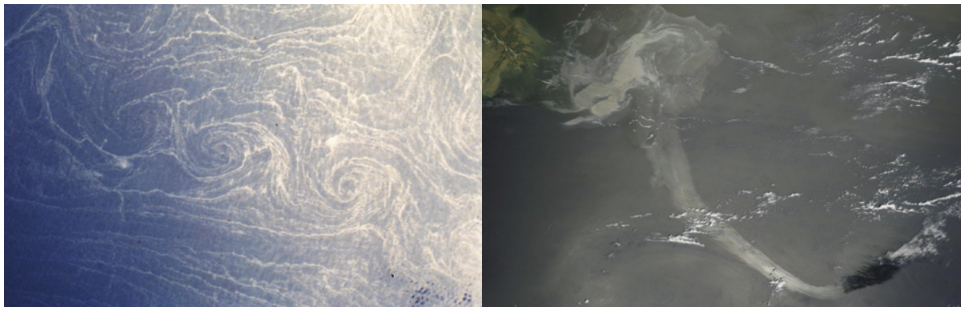


Figure 1.1: Left: Eddies in the Mediterranean Sea. Right: The Deepwater Horizon oil spill seas a long tendril of oil suddenly appear. Both images taken from [13]

One way in which this has been done is by finding the fixed points of a dynamical system and then characterising the motion around these fixed points. The problem with using fixed points as a theory of stability is that they struggle to capture the underlying mechanics that underpins

the movement of a dynamical system. In fact, dynamical systems are also characterised by the larger scale flow structures we see in them. For example, we may spot eddy currents in the ocean on smaller scales or on larger scales we may spot strong currents that advect particles in the dynamical system for large distances.

These flow structures are often invariant with the underlying motion of the dynamical system, thus making them lagrangian by nature. For example, the eddies seen above 1.1 are often moving with the underlying currents of the ocean. We may thus call these sorts of structures **Lagrangian Coherent Structures** (LCS). There are three types of LCS: Hyperbolic LCS are the locally most attracting or repelling material surfaces in the phase space, Elliptic LCS are closed material surfaces that make up the lagrangian equivalents of vorticies and Parabolic LCS are characterised by minimal shearing and stretching, travelling through a medium as jet cores [13].

We will see that, Hyperbolic LCS are ridges of a scalar field called the Finite Time Lyapunov Exponent (FTLE) Field. This scalar field measures the growth of displacements one sees within a particular dynamical system as time goes on. The growth of these displacements can describe shear flow in some regions of the dynamical system or flow that exhibits exponential growth in other regions of the dynamical system. It is, in fact, possible to single out a ridge of the FTLE field that represents only one of these regimes of growth. We will see that this is a powerful tool for characterising the different types of flow in a region.

1.2 Crowd Video Analysis

The continued increase in world population and migration to large cities means that crowded spaces are becoming more and more prevalent. With this, the management of these spaces and their safety is becoming more important. As an example, crowd crush disasters in the Hajj pilgrimage have occurred multiple times in recent years. The first happened in 1990, killing 1462 people [1]. While measures were put in place to increase safety for the next pilgrimages, a further 2411 people were killed [10] in a stampede that occured in 2015, outside the bridge that connects to the great mosque of Mecca. Thus, there is a real need to understand when these crowded situations can occur and how we can stop these events from happening in the first place. Crowded Video Analysis (CVA) is an answer that meets this demand. Using computational techniques, it extracts as much information

from a video of a crowded scene with the aims of informing governments, city planners or other interested parties.

Broadly, there are four reasons for which we will be interested in using CVA. They are: crowd motion analysis, which aims to extract information about how a crowd moves within videos; crowd behavior recognition, which aims to recreate or detect certain movements seen in crowds; crowd anomaly detection, which aims to detect when unusual or dangerous events occur in a video; crowd surveillance, which aims to track, identify or count the people or discrete constituents (cars, bikes etc) seen in a video for the safety of the general public [26], [29]. While this is not an exhaustive list, it touches upon the main motivations that we are interested in.

There are many methods to conduct CVA. One of the main methods that is currently trending, perhaps unsurprisingly, is the use of machine learning [17]. However, before this rise in popularity, physically based methods were more popular. The theory of Statistical Physics was used to describe a crowd by its microstates and macrostates so that any change in the macrostate could be linked to an anomalous event [30]. Social Force Models, models that simulate a pedestrians movement through a mix of Newtonian forces and sociological forces (for more read [7]) were used to find abnormal movements within crowds.

More pertinent to us, however, are the methods based on Dynamical Systems Theory. In [25], they identified sources and sinks within a crowd video to find the dominant crowd flows. Similarly, Solmaz et al. [4] identifies many other types of crowd flows from the eigenvalues of the dynamical system seen in a crowd video. They use the change in eigenvalues of the dynamical system as a possible sign of changes in behaviour within the crowd.

Within these methods are those that are based in Lagrangian Coherent Structures. Ali [2] uses Lagrangian Coherent Structures to segment the crowd flow into distinct types of flow. In [27], LCS Theory was used to detect when a person picked up an object in a video. Although the application of LCS theory is varied in CVA, we don't see any applications that use results in the LCS theory which are past identifying the Hyperbolic LCS using FTLE Fields. Thus, this paper will attempt to explore this unexplored territory.

1.2.1 Structure of the Paper

The paper will be divided into three sections excluding the Introduction and Conclusion. The first section will cover the derivation of the Finite Time Lyapunov exponent. The second section will then move onto the application

of the FTLE to a crowd video following [2]. In the third section, we introduce a new application of LCS theory given by the variational theory [12] which we apply to the crowd video. We see promising results.

1.2.2 Datasets

We leave a remark on the main datasets we used in the making of this paper. It is important to note that not all of the datasets listed here have been used to write the paper, instead, some were used purely for experimentation purposes.

- **UCF Crowd Segmentation Dataset** [2] - This dataset holds many of the most useful videos. As the dataset is for crowd segmentation, the videos are primarily of crowds moving as a collective, from an aerial point of view and with a fixed camera. Many of the videos depict typical benign crowd behaviour such as marathons, daily traffic or crowded city centres. These videos will give a good benchmark for the methods that we develop.

- **Abnormal/Normal Crowds Dataset** [21] - This dataset contains 8 videos of abnormal scenes such as a police shootout or violence breaking out in a crowd and 12 videos of normal behaviour videos such as in the **UCF Crowd Segmentation Dataset** [2]. This dataset was used to explore whether the methods developed in this paper were useful for detecting abnormal behaviour in a video. We note that it came with a further separate 7 abnormal videos and 10 benign videos.

- **BEHAVE Dataset** [24] - This dataset is comprised of a set of four long videos of people acting out various different behavioural actions. The aim of the dataset is to be able to detect the different behaviours seen in the video.

- **Car Crash Dataset** [28] - This dataset consists of dashcam footage of car crash incidents. Each video is 50 frames long and there are 1500 videos.

We note that some YouTube videos were also used for experimentation, although all results in the paper come from the datasets listed above.

1.2.3 Computer Programming

For the numerical application of the methods used in this paper, we have used the programming language Python. We list the various Python packages we have used in the making of this paper:

- Scipy [20] - An open source package for interpolation, integration and differentiation algorithms.
- Numpy [5] - An open source package for manipulating arrays of many dimensions. A powerful tool for vector calculations.
- OpenCV [3] - An open source package for computer vision, useful for reading and writing video files and general manipulation of images.
- Matplotlib [15] - An open source package for data visualisation. All of the data plots in this paper were completed using Matplotlib.

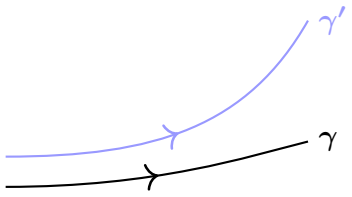
Finally, we note that all of the source code in this paper will be made available in a github repository. The link is given below:
https://github.com/Senryushi/Exploration-of-LCS-on-CVA

Chapter 2

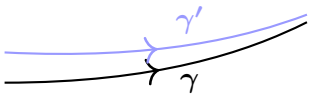
Finite Time Lyapunov Exponents

We start our exploration of applying LCS theory to crowd flows by looking at the fundamental building block of LCS theory. That is, the Finite Time Lyapunov Exponent (FTLE).

Lyapunov Exponents appear in Dynamical Systems theory when determining how stable a particular trajectory is [19]. Specifically, Lyapunov Exponents tell us if for some principle curve $\boldsymbol{\gamma}$ that is parameterised by $t \in \mathbb{R}$, other curves or trajectories $\boldsymbol{\gamma'}$ that start close to this principle trajectory remain close to that trajectory as $t \to \infty$. It follows then that **Finite Time** Lyapunov Exponents are the finite time equivalents of this. Where instead, we measure how far away a trajectory $\boldsymbol{\gamma'}$ that starts close to some main trajectory $\boldsymbol{\gamma}$ at time t_0 deviates away from this trajectory at later time t. In fluid dynamical applications of FTLE, it is common to think of a particles initial position $\mathbf{x_0}$ and later time positions $\mathbf{x}$ in some flow instead of the trajectories themselves. We will understand FTLEs in this fluid dynamical way.



(a) High Lyapunov Exponent trajectories



(b) Low Lyapunov Exponent trajectories

We may use the FTLE to find flows that are significantly different from each other. Intuitively, a particle $\mathbf{y_0}$ that starts close to some particle $\mathbf{x_0}$ will not remain close to that particle if they lied in regions of flow that are distinctly different to each other. We will see that the FTLE defines the boundaries between these distinct regions of flow.

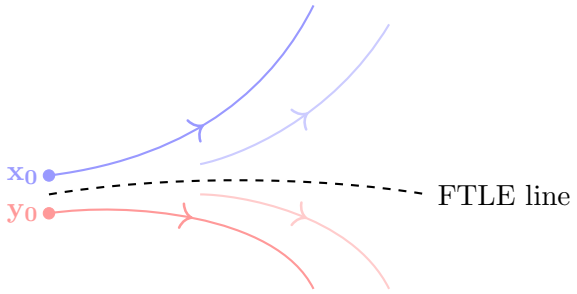


Figure 2.2: Particles $\mathbf{x_0}$ and $\mathbf{y_0}$ lie in distinct regions of flow and the boundaries of these flows is defined by the line of the FTLE.

2.1 The Flowmap

To start, let us have some velocity field $\mathbf{v}(\mathbf{x}, t) \in \mathbb{R}^2$, where we choose $\mathbb{R}^2$ because the methods we develop will be applied to video data. The velocity field depends on $\mathbf{x} \in U \subset \mathbb{R}^2$, a 2-dimensional position vector, and time $t \in [a, b] \subset \mathbb{R}$. Where U is a open subset of $\mathbb{R}^2$ representing the frame of the video and the interval $[a, b]$ represents the time interval of the video. Let us have some particle which lies at an initial position $\mathbf{x_0} \in U$ at initial time t_0 that is advected by this velocity field. To find it's position at later times t we solve the differential equation,

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t). \tag{2.1}$$

This has a general solution of the form $\mathbf{x}(t; t_0, \mathbf{x_0})$. It will be useful for us to define a mapping that describes the movement of particles from the initial time positions $\mathbf{x_0}$ to later time positions $\mathbf{x}(t; t_0, \mathbf{x_0})$. This is described in the function below:

Definition 2.1.1 (Flowmap)**.** Let us have some particle at initial position $\mathbf{x_0} \in U$ and initial time t_0. It is advected by the velocity field $\mathbf{v}(\mathbf{x}, t) \in \mathbb{R}^2$. The position of the particle at later time t is given by the **Flowmap** $F_{t_0}^t$: $U \to \mathbb{R}^2$,

$$F_{t_0}^t(\mathbf{x_0}) := \mathbf{x}(t; t_0, \mathbf{x_0}). \tag{2.2}$$

Let us have a look at how this Flowmap acts on the saddle-flow velocity field:

Example 2.1.2 (Saddle Flow)**.** The velocity field for a Saddle-flow with $\mathbf{x} = (x, y)$ is given by,

$$\begin{aligned} \dot{x} &= x, \\ \dot{y} &= -y. \end{aligned}$$

It has solutions,

$$\begin{aligned} x(t; t_0, x_0) &= x_0 e^t, \\ y(t; t_0, y_0) &= y_0 e^{-t}, \end{aligned}$$

for $\mathbf{x_0} = (x_0, y_0)$. The flowmap is thus given by this solution,

$$F_{t_0}^t(\mathbf{x_0}) = (x_0 e^t, y_0 e^{-t}).$$

Now, let us fix the initial conditions and calculate where the particle is advected to by the flow. Let $\mathbf{x_0} = (1, 1)$ at $t_0 = 0$. Then, at $t = ln(2)$ the flowmap tells us the particle's final time position,

$$\begin{aligned} F_0^{ln(2)}(\mathbf{x_0}) &= (1 \cdot e^{ln(2)}, 1 \cdot e^{-ln(2)}) \\ &= (2, \frac{1}{2}). \end{aligned}$$

We see that the Flowmap mapped the particle from $(1, 1) \to (2, \frac{1}{2})$. We can see this movement when we view this velocity field on a graph as shown below 2.3.

Note that in the example above 2.1.2, we could have chosen any particle as $\mathbf{x_0}$, not just $(1, 1)$. In fact, we may apply the Flowmap to all particles in U so that we have a Flowmap field which gives the later time positions of all particles initially in U. In this way, a lot of the machinery that we derive will be applicable to the whole space it is defined on.

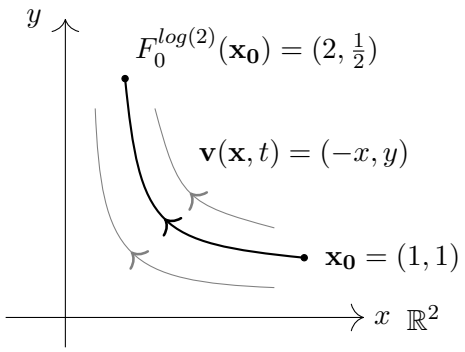


Figure 2.3: A particle's initial time position is mapped to its final time position by the Flowmap

2.2 Derivation of the FTLE

Recall that we are trying to find how the proximity of trajectories of particles that lie in a velocity field $\mathbf{v}(\mathbf{x}, t)$ changes from some initial time t_0 to some later time t. In particular, we would like to find particles that initially start close to each other, but at later times end up further away from each other. To this end, let us have some initial particle $\mathbf{x_0}$ and a particle close by given by $\mathbf{x_0} + \delta\mathbf{x}$, where $\delta\mathbf{x}$ is some infinitesimal perturbation vector. We can then find the later time positions of the particles with the Flowmap.

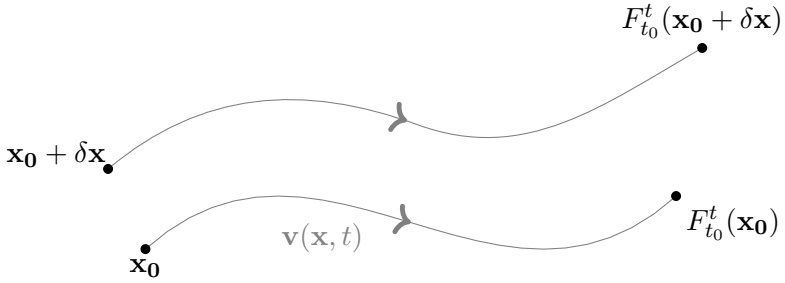


Figure 2.4: Initial time particles are taken to their later time positions by the Flowmap

Note that we didn't choose a specific $\delta\mathbf{x}$, thus $\mathbf{x_0} + \delta\mathbf{x}$ can lie anywhere in some neighbourhood of $\mathbf{x_0}$. We may define perturbation vectors between the pair of particles at initial and later times.

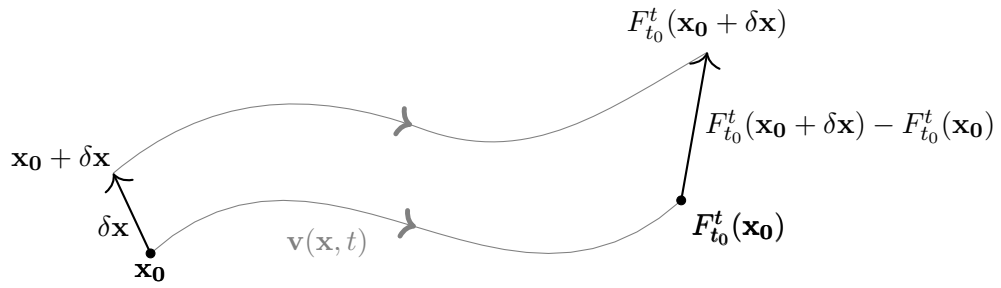


Figure 2.5: Vectors between the pair of particles are advected with the particles

To see the affect of the velocity field on the two particles, we would like to be able to find a relation between the two perturbation vectors $\delta\mathbf{x}$ and $F^t_{t_0}(\mathbf{x_0}+\delta\mathbf{x}) - F^t_{t_0}(\mathbf{x_0})$. As in [23], we start by taking the Taylor expansion of $F^t_{t_0}(\mathbf{x_0}+\delta\mathbf{x})$ around $\delta\mathbf{x}$.

$$F^t_{t_0}(\mathbf{x_0}+\delta\mathbf{x}) = F^t_{t_0}(\mathbf{x_0}) + \frac{dF^t_{t_0}(\mathbf{x_0})}{d\mathbf{x_0}}\delta\mathbf{x} + O((\delta\mathbf{x})^2). \tag{2.3}$$

Now getting rid of higher order terms in $\delta\mathbf{x}$, rewriting the derivative with respect to $\mathbf{x_0}$ as the gradient ∇_0 and rearranging the equation slightly,

$$F^t_{t_0}(\mathbf{x_0}+\delta\mathbf{x}) - F^t_{t_0}(\mathbf{x_0}) = \nabla_0 F^t_{t_0}(\mathbf{x_0})\delta\mathbf{x}. \tag{2.4}$$

We have arrived at a relation between the two perturbation vectors that we defined in 2.5. We can rename these vectors as follows, $\boldsymbol{\xi_0} = \delta\mathbf{x}$ and $\boldsymbol{\xi} = F^t_{t_0}(\mathbf{x_0}+\delta\mathbf{x}) - F^t_{t_0}(\mathbf{x_0})$.

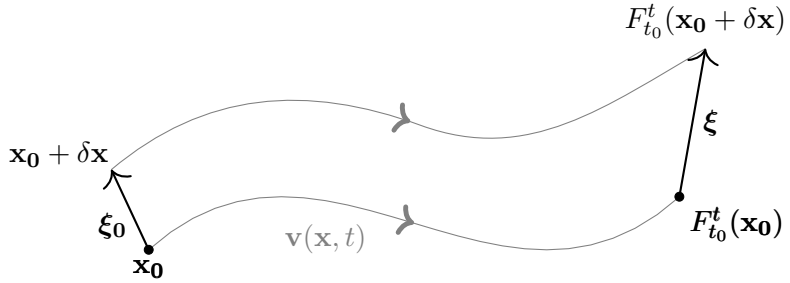


Figure 2.6: The initial perturbation vector $\boldsymbol{\xi_0}$ and final perturbation vector $\boldsymbol{\xi}$.

Then the final relation between the two perturbation vectors simplifies to:

$$\boldsymbol{\xi} = \nabla_0 F_{t_0}^t(\mathbf{x_0})\boldsymbol{\xi_0} \tag{2.5}$$

We note that $\nabla_0 F_{t_0}^t(\mathbf{x_0})$ is just the 2-dimensional Jacobian matrix with respect to the initial particle positions. We note again that our choice of $\boldsymbol{\xi_0}$ is arbitrary, $\boldsymbol{\xi_0}$ may point in any direction.

For trajectories that grow apart, one expects the size of the initial perturbation vector to increase significantly as time goes on. Thus, let us look at the size of the final perturbation vector.

$$\begin{aligned}
\|\boldsymbol{\xi}(t)\|^2 &= \|\nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{\xi}(t_0)\| \\
&= \langle \nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{\xi}(t_0), \nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{\xi}(t_0)\rangle \\
&= \langle \boldsymbol{\xi}(t_0), [\nabla F_{t_0}^t(\mathbf{x_0})]^T \nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{\xi}(t_0)\rangle \\
&= \langle \boldsymbol{\xi}(t_0), C_R(\mathbf{x_0})\boldsymbol{\xi}(t_0)\rangle,
\end{aligned}$$

where the inner product $\langle \cdot, \cdot \rangle$ is the usual euclidean inner product. We used transpose rules in the second line to move the Jacobian to the other side of the inner product. We see that the length of the perturbation vector is given by an inner product of the initial perturbation vectors with a tensor $C_R(x_0)$:

Definition 2.2.1 (Cauchy-Green Strain Tensor)**.** The following 2-dimensional tensor is called the Right Cauchy-Green Strain Tensor:

$$C_R(\mathbf{x_0}) = [\nabla F_{t_0}^t(\mathbf{x_0})]^T \nabla F_{t_0}^t(\mathbf{x_0}). \tag{2.6}$$

Similarly, the Left Cauchy-Green Tensor is given by:

$$C_L(\mathbf{x_0}) = \nabla F_{t_0}^t(\mathbf{x_0})[\nabla F_{t_0}^t(\mathbf{x_0})]^T. \tag{2.7}$$

The Right Cauchy-Green Strain Tensor tells us how much the initial perturbation vector $\boldsymbol{\xi_0}$ is stretched in the flow as time goes on. As $\nabla F_{t_0}^t(\mathbf{x_0})$ is a 2-dimensional invertible matrix, $C_R(\mathbf{x_0})$ is a symmetric tensor. Thus it must have corresponding orthonormal eigenvectors $\boldsymbol{v}_i$ and eigenvalues λ_i [14, Haller 2023]. Then it satisfies the eigenvalue equation:

$$C(x_0)\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{2.8}$$

with eigenvalues ordered so that $0 < \lambda_1 \leq \lambda_2$ without loss of generality. Then looking at the effect of the flow on the eigenvectors $\boldsymbol{v}_i$:

$$\begin{aligned}\|\nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{v}_i\| &= \sqrt{\langle \nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{v}_i, \nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{v}_i\rangle} \\ &= \sqrt{\langle \boldsymbol{v}_i, C_R(\mathbf{x_0})\boldsymbol{v}_i\rangle} \\ &= \sqrt{\langle \boldsymbol{v}_i, \lambda_i \boldsymbol{v}_i\rangle} = \sqrt{\lambda_i \langle \boldsymbol{v}_i, \boldsymbol{v}_i\rangle} \\ &= \sqrt{\lambda_i},\end{aligned}$$

where we used the eigenvalue equation 2.8 to get from the second line to the third line and we used the fact that our eigenvectors are unit length in the final step. Thus the eigenvectors grow by a factor $\sqrt{\lambda_i}$ when they are advected by the flow.

As our choice of $\boldsymbol{\xi_0}$ was arbitrary, we could choose $\boldsymbol{\xi_0} = \boldsymbol{v}_2$. We do in fact choose λ_2 as it is the largest eigenvalue, and so $\boldsymbol{v}_2$ is the direction in which the largest stretching occurs. In other words, $\boldsymbol{v}_2$ represents the direction of furthest growth in displacement relative to the particle of interest $\mathbf{x_0}$. Particles that lie in this direction move furthest away from the particle of interest $\mathbf{x}_0$.

It is generally accepted in the literature that flows in this direction grow at an exponential rate *(see [14] page 148)*. Thus, we have:

$$\|\nabla F_{t_0}^t(\mathbf{x_0})\boldsymbol{v}_2\| \sim e^{\Lambda t} = \sqrt{\lambda_2}, \tag{2.9}$$

where Λ represents the growth rate. Then, rearranging this equation and normalising this growth rate for the time interval t_0 and t that we calculate it over, we arrive at the definition of the **Finite-time Lyapunov Exponent**.

Definition 2.2.2 (Finite-time Lyapunov Exponent (FTLE))**.** Let us have a velocity field $\mathbf{v}(\mathbf{x}, t) : U \times [a, b] \to \mathbb{R}^2$. For any initial position $\mathbf{x_0} \in U$ at initial time $t_0 \in [a, b]$, the largest amount of perturbation around the particle when it is advected to a final time position $\mathbf{x}$ at time t is given by λ_2. Then, the **Forward Finite-time Lyapunov Exponent** (**Forward FTLE**) is given by

$$\Lambda_{t_0}^t(\mathbf{x_0}) = \frac{1}{2|t - t_0|} ln(\lambda_2(\mathbf{x_0})). \tag{2.10}$$

We obtain the **Forward Finite-time Lyapunov Exponent Field** by calculating the Forward FTLE for all $\mathbf{x_0} \in U$. Similarly, the **Backward Finite-time Lyapunov Exponent** (**Backward FTLE**) is given by

$$\Lambda_t^{t_0}(\mathbf{x_0}) = \frac{1}{2|t - t_0|} ln(\lambda_2(\mathbf{x_0})). \tag{2.11}$$

We obtain the **Backward Finite-time Lyapunov Exponent Field** by calculating the Backward FTLE for all $\mathbf{x_0} \in U$.

The Forward FTLE field measures the amount of perturbation that occurs around some initial particle $\mathbf{x_0}$ when it is advected by the flow. In particular, high values of the Forward FTLE will correspond to flows with trajectories that diverge away from the initial particle $\mathbf{x_0}$ as they are advected by the flow.

In contrast, the Backward FTLE field measures the amount of perturbation that occurs around some final particle $\mathbf{x}$ when it is advected backwards in time by the flow. Thus, high values of the Backward FTLE will correspond to flows with trajectories that diverge from the final particle position when advected in backwards time. It follows then that in forward time, high Backward FTLE values correspond to trajectories that converge to the particle $\mathbf{x}(t; t_0, \mathbf{x_0})$ as they are advected by the flow.

2.3 Conclusion

We now have some scalar field that measures the maximal growth of perturbation vectors around some initial particle $\mathbf{x_0}$ in the dynamical system. It turns out that when we calculate the FTLE field for certain dynamical systems, we find a curve made up of particularly high values of the FTLE field. These curves are the most repellent curves in the dynamical system and closely resemble the LCS that we are after. In the following chapter, we calculate the FTLE fields for a particular dynamical system and we also calculate it for an example crowd video. To do this, there are many hurdles that we must cross.

Chapter 3

Application of the FTLE to Crowd Video Analysis

FTLEs give a quantitative metric on the relative growth of particle displacements for each particle in the flow. For regions of high FTLE values, the relative growth of displacements between the particles is high, however, for low values, those displacements shrink over time.

This idea may be translated to crowds. A dense crowd of people may act very much like a fluid, with each person representing a particle. With this in mind, we may locate the shrinking of displacements between people by the high values of the backward FTLE field. Then, high values of backwards FTLE may indicate a dangerous crowd crush scenario or a high density of crowds. Our aim is to apply the FTLE field to crowd videos to see if we can identify regions of high density in the crowd video.

3.1 Discretising the FTLE field

To apply the FTLE to video data, we must first discretise the problem. First, we take a grid of $N \times M$ initial particle positions that lies in a rectangular domain U. Then, we advect the particles to later time t using the Forward Euler method. That is, for a particle $\mathbf{x}_{i,j}(t_k) = (x_{i,j}(t_k), y_{i,j}(t_k)), i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\}$ at time $t_k \in [t_0, t]$, we approximate the later time $t_k + \delta t$ position of this particle by

$$\mathbf{x}_{i,j}(t_k + \delta t) = \mathbf{x}_{i,j} + \delta t\, \mathbf{u}(x_{i,j}, y_{i,j}, t_k), \tag{3.1}$$

where $\mathbf{u}(x_{i,j}, y_{i,j}, t_k)$ is the particle's velocity at time t_k. This method works well for small values of δt as the error increases with the size of it. We calculate the later time positions of these particles for all time in the interval of interest t_0 to t. The period of this interval is $T = t - t_0$, which we can use to calculate the total number of time-steps $\mathcal{N} = \frac{T}{\delta t}$ by fixing δt so, $k \in \{1, \ldots, \mathcal{N}\}$.

Using this, we can iteratively calculate the non-initial positions of the particles from t_0 to t to obtain the full trajectories of the initial grid of particles. This is given by a set of $N \times M$ matrices $\boldsymbol{\mathcal{X}} = \{\boldsymbol{\mathcal{X}}_{t_0}, \ldots, \boldsymbol{\mathcal{X}}_{t_\mathcal{N}}\}$ where:

$$\boldsymbol{\mathcal{X}}_{t_k} = \begin{pmatrix} \mathbf{x}_{0,0}(t_k) & \ldots & \mathbf{x}_{N,0}(t_k) \\ . & & . \\ . & & . \\ . & & . \\ \mathbf{x}_{0,M}(t_k) & \ldots & \mathbf{x}_{N,M}(t_k) \end{pmatrix} \tag{3.2}$$

We can split this matrix into it's x and y components:

$$\mathbf{X}_{t_k} = \begin{pmatrix} x_{0,0}(t_k) & \ldots & x_{N,0}(t_k) \\ . & & . \\ . & & . \\ . & & . \\ x_{0,M}(t_k) & \ldots & x_{N,M}(t_k) \end{pmatrix}, \mathbf{Y}_{t_k} = \begin{pmatrix} y_{0,0}(t_k) & \ldots & y_{N,0}(t_k) \\ . & & . \\ . & & . \\ . & & . \\ y_{0,M}(t_k) & \ldots & y_{N,M}(t_k) \end{pmatrix}$$

Note that $\boldsymbol{\mathcal{X}}$ which defines the set of all particle positions for all time and all particles, holds the approximate equivalence:

$$\boldsymbol{\mathcal{X}} \approx (F_{t_0}^{t_0}(\boldsymbol{\mathcal{X}}_{t_0}), \ldots, F_{t_0}^{t_\mathcal{N}}(\boldsymbol{\mathcal{X}}_{t_0})). \tag{3.3}$$

As in our derivation of the FTLE in the last chapter, we need the gradient of the Flowmap to calculate the Right Cauchy-Green Strain tensor. Recall the gradient of the Flowmap is just the Jacobian of the Flowmap with respect to the initial particle positions. More precisely:

$$\nabla_0 F_{t_0}^t(\mathbf{x_0}) = \begin{pmatrix} \frac{\partial x}{\partial x_0} & \frac{\partial x}{\partial y_0} \\ \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial y_0} \end{pmatrix}. \tag{3.4}$$

To numerically calculate this, we use finite central differences. This approximates the partial derivatives so that, for example:

$$\frac{\partial x}{\partial x_0} \approx \frac{F_{t_0}^t(x_0 + h, y_0) - F_{t_0}^t(x_0 - h, y_0)}{2h}, \tag{3.5}$$

for a small perturbation constant h. However, in our discretisation it is unwieldy to calculate the Flowmap for a small perturbation because it is computationally expensive to calculate the later time position of a particle that is displaced by h for the calculation for every partial derivative. It would mean having to calculate the later time positions of five different sets of $N \times M$ particles. Thus, instead we calculate:

$$\frac{\partial x}{\partial x_0} \approx \frac{\mathbf{X}_{i+1,j}(t_k) - \mathbf{X}_{i-1,j}(t_k)}{\mathbf{X}_{i+1,j}(t_0) - \mathbf{X}_{i-1,j}(t_0)}. \tag{3.6}$$

We rely on the individual particles being close to each other so that h, which is given by the distance between each adjacent particle, is small. The full approximation of the Jacobian is then:

$$\nabla_0 F_{t_0}^t(\mathbf{x_0}) \approx \begin{pmatrix} \frac{x_{i+1,j}(t_k) - x_{i-1,j}(t_k)}{x_{i+1,j}(t_0) - x_{i-1,j}(t_0)} & \frac{x_{i,j+1}(t_k) - x_{i,j-1}(t_k)}{y_{i,j+1}(t_0) - y_{i,j-1}(t_0)} \\ \frac{y_{i+1,j}(t_k) - y_{i-1,j}(t_k)}{x_{i+1,j}(t_0) - x_{i-1,j}(t_0)} & \frac{y_{i,j+1}(t_k) - y_{i,j-1}(t_k)}{y_{i,j+1}(t_0) - y_{i,j-1}(t_0)} \end{pmatrix}. \tag{3.7}$$

We note that for the edge cases $i, j = 0$ or $i = N, j = M$ we use forward differences and backward differences respectively.

Using this approximation for the Jacobian, we are able to calculate the Right Cauchy-Green Strain Tensor using our definition 2.6. Then, using `numpy.linalg` python functions we can extract the eigenvalues for each particle trajectory and calculate the FTLE value for each initial particle position $\mathbf{x_0}$. Once we calculate this for the whole initial grid of particles $\boldsymbol{\mathcal{X}}_{\boldsymbol{t_0}}$, we have the FTLE field.

3.2 Example: The Double Gyre-flow

To illustrate the machinery that we have just developed, we apply it to a classic periodic dynamical system that has been extensively studied [8], [22].

Definition 3.2.1 (The Double Gyre-flow)**.** The Double Gyre-flow velocity field $\mathbf{g} : \mathbb{R}^2 \to \mathbb{R}^2$ is given by:

$$\begin{aligned} a(t) &= \varepsilon sin(\omega t) \\ b(t) &= 1 - 2\varepsilon sin(\omega t) \\ f(x,t) &= a(t)x^2 + b(t)x \\ \mathbf{g}(x,y,t) &= \begin{pmatrix} -\pi A sin(\pi f(x,t)) cos(\pi y) \\ \pi A cos(\pi f(x,t)) sin(\pi y) f_x(x,t) \end{pmatrix} \end{aligned}$$

where f_x denotes a partial derivative with respect to x, and ε and ω are given constants.

This dynamical system consists of two vortices that oscillate in the x-axis periodically. Importantly, it has boundaries on the rectangular box $U = [0, 2] \times [0, 1]$ such that the normal velocities at these boundaries vanish. To see this, substitute in $x = 0, 2$ and $y = 0, 1$ and the x-component and y-component of the velocities will vanish respectively. Thus, the Double Gyre-flow is a closed system in the domain $[0, 2] \times [0, 1]$ (i.e. no particle that lies in the box at initial time ever leaves the box at later times).

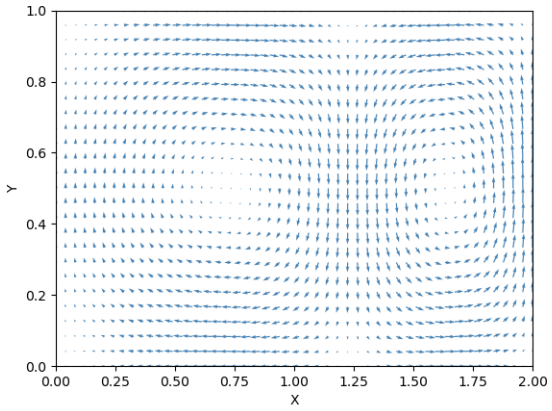
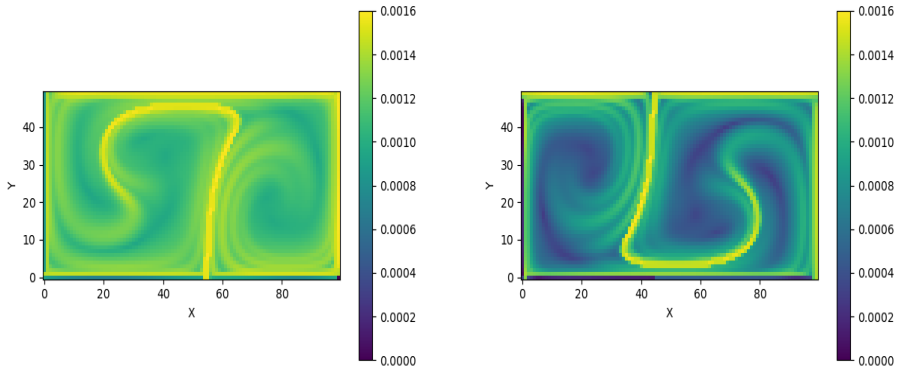


Figure 3.1: The velocity field of the Double Gyre-flow at time $t = 2.5$

To calculate the FTLE field on the domain $[0, 2] \times [0, 1]$ we arbitrarily initialise a grid of particles that are equally spaced from each other (e.g. 200×100 particles). Then, we calculate the full trajectories of each of the particles from an initial time $t_0 = 0$ to final time $t_1 = 10$ using the iterative Forward Euler method we outlined above. From here, we calculate the Forward FTLE field (from $t_0 = 0$ to $t_1 = 10$) and the Backward FTLE field (from $t_0 = 10$ to $t_1 = 0$) on the whole time period.

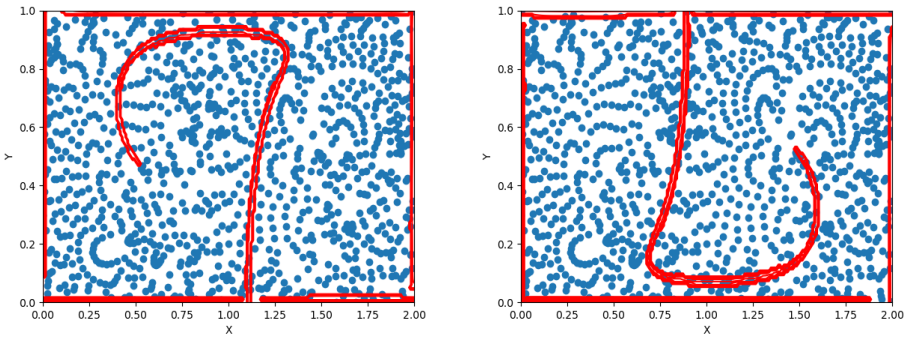
Looking at the FTLE plots below, we see a clear line in both the forward and backward FTLE fields. The line is known as a ridge of the FTLE field. A ridge corresponds to a curve in the FTLE field whose values are locally maximal. In the forward FTLE field, this is the curve that the particles in the flow diverge from. In the backwards FTLE field, the ridge represents the curve that particles are most attracted to in forward time. So, we expect to see dense regions of particles following the curve in the backwards FTLE



(a) (b)

Figure 3.2: (a) Forward FTLE Field for Double Gyre-flow (b) Backward FTLE Field for Double Gyre-flow

field and sparse regions of flow following the curve in the forward FTLE field.



(a) Forward FTLE ridge (b) Backward FTLE ridge

Figure 3.3: The final positions of particles initially in a grid at $t_0 = 0$ and advected to $t = 10$ with the ridges of forward and backwards FTLE overlayed over it

We clearly see the density of particles being higher around the backward FTLE ridge and lower around the forward FTLE ridge, as expected.

We could think of these ridges as material curves that evolve in time with the velocity field. In fact, in some cases, these curves define repellant Lagrangian Coherent Structures [11], [12]. One way we can see this is because particles that lie on these curves stay on them for indefinite time. In particular, the forward FTLE ridge is an unstable manifold and the backward FTLE ridge is a stable manifold of the corresponding dynamical system in the infinite time limit (if such a limit were to exist).

It is also important to note that ridges of the FTLE are not always strong attractors or repellers of particles. This is because the FTLE only depends on the growth of displacements between the particles and not the direction of growth between the particles [11]. Thus, strongly shearing flows will also show up as strong ridges in the FTLE field [12].

3.3 Optical Flow

Our goal now is to apply the techniques we have just introduced to videos of crowds and vehicles. Given this, we first require a method to obtain our velocity field $\mathbf{v}(\mathbf{x}, t)$. For this reason, we have a short excursion to Optical flow.

The problem we want to solve is as such: Given two frames that are consecutive to each other, we would like to find the velocity vector field between the two frames. Methods which produce velocity fields for the whole frame are called **dense** while methods which only calculate velocity vectors for specified points in the frame are called **sparse**. We are looking for dense velocity fields. These velocities are two-dimensional projections of real life velocities. Thus, optical flow methods are merely approximations of real world velocities. One may cover the third dimension by using multiple cameras that capture the same scene. However, in most cases, a single camera is enough. Moreover, in most real world cases it isn't practical to use more than one camera to study a scene. Indeed the datasets we use in this project only use a single camera. The task is then to maximise the amount of information we get from this 2-dimensional video frame data.

To do this, we need an understanding of the limitations of the Optical Flow techniques we use. In this vein, we will need to understand optical flow. To start, let us have an intensity function $I(x, y, t) : U \to \mathbb{R}$ which gives the intensity value (brightness) of a pixel at a given time and position on a two-dimensional frame. We would like to find the displacement of a

pixel from one point to another in the time of one frame. Let us have a pixel at point (x_0, y_0) at time t_0. It's intensity value is $I(x_0, y_0, t_0)$. Given the pixel is moving, we assume the pixel moves a small distance δx, δy over a small time δt. Here, we make a key assumption which is central to most optical flow techniques:

Assumption 3.3.1. The intensity I of an image point remains constant over time.

Thus, given a pixel on a car (for example), that same pixel on the car in the next frame must have the same intensity value. If this were not the case, it would make finding the corresponding pixel in the next frame impossible as their intensities would not match. It means that we prefer videos that are highly textured. It also means that it is nearly impossible to find velocity vectors of videos with lots of noise. Thus, the videos that we would like to use are ones that are highly textured and with minimal noise. We can represent the one-to-one correspondence 3.3.1 between pixels in the equation below:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t).$$

Then, taking the Taylor expansion of this,

$$I(x, y, t) = I(x, y, t) + \delta x \frac{dI}{dx} + \delta y \frac{dI}{dy} + \delta t \frac{dI}{dt} + O(\delta x^2, \delta y^2, \delta t^2),$$

and getting rid of higher order terms and rearranging, we reach the following equation [6],

Definition 3.3.2 (Optical Flow Equation)**.** Given a video and assuming that the displacements between pixels in successive frames $(\delta x, \delta y)$ and the time between successive frames δt are small, an intensity function $I(x, y, t) : U \to \mathbb{R}$ must satisfy the **Optical Flow Equation**:

$$\frac{\delta x}{\delta t}\frac{dI}{dx} + \frac{\delta y}{\delta t}\frac{dI}{dy} + \frac{dI}{dt} = 0. \tag{3.8}$$

where $\left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}\right)$ is the optical flow vector.

The optical flow vector represents the velocity for each pixel in-between each frame. Once we have all optical flow vectors for all pixels in the frame, we have the completed velocity field $\mathbf{v}(\mathbf{x}, t)$. A method using this theory has been developed by Bruce D. Lucas and Takeo Kanade [16].

One issue with this method is that it assumes that the displacements are small. This isn't always the case. For fast moving objects, the displacements $\delta x, \delta y$ can be big enough that the Taylor Expansion approximation becomes inappropriate. To get around this problem Farneback [9] used a pyramid method so that the Taylor Expansion approximation becomes valid again. We use this optical flow method for which there exists a function in the OpenCV Python library [3].

3.3.1 Criteria for Datasets

Now knowing our assumptions from the optical flow methods, we may characterise the datasets that we would like to use. The videos must be:

- with little noise.
- taken from an aerial point of view as the velocities will naturally translate to the 2-dimensional projection.
- be highly textured.

Importantly, the videos need not have stationary video cameras as the FTLE is a lagrangian tool, it will tell us movement relative to the moving constituents. However, stationary cameras will be preferred as most moving cameras can shake and produce noise.

3.4 Methodology

Figure 3.4: A frame of the motorway video which we will do our analysis on

As we stated earlier, the use of LCS theory in the context of CVA was pioneered by Saad Ali and Mubarak Shah [2]. Their application used the FTLE field to segment regions of the dynamical system into distinct regions of flow. Until the end of the chapter, we will closely follow their method and improve upon it.

To aid in our endeavours, we will focus on a particular example video taken from the **UCF Crowd Segmentation Dataset** [2]. The video is of a motorway with two lanes going in opposite directions to each other and a junction joining it. This video is particularly useful as it has a shear boundary at the barrier that separates the two lanes moving in opposite directions. Thus, it will be a good test for producing shear FTLE ridges. To analyse this video of length $\boldsymbol{N}$ frames, we start by using the Farneback Optical Flow and obtain a velocity field $\mathbf{v}(\mathbf{x}, t)$ of length $\boldsymbol{N} - 1$ frames.

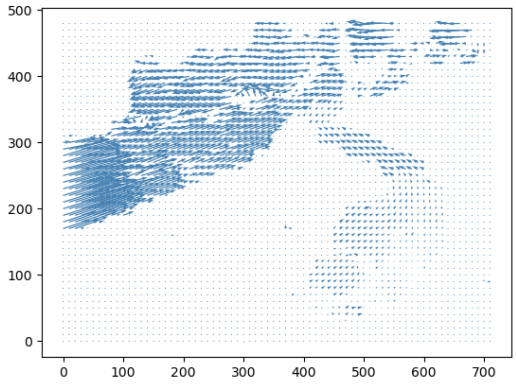


Figure 3.5: The velocity field $\mathbf{v}(\mathbf{x}, t)$ of a frame of the motorway video calculated by Farneback optical flow

With this velocity field, we will use the same procedure as we did for the Double Gyre-flow 3.2.1. Earlier, we arbitrarily initialised a grid of particles. This time, we choose to place a particle every p (typically $p = 3, 4$) pixels to lessen the computational load. Then, before advecting the particles, we average the velocity field over a number of frames n_V to obtain the mean velocity field $\tilde{V}$. Note that we choose $\tilde{V}$ to have $\boldsymbol{M} = \left\lfloor \frac{\boldsymbol{N}-1}{n_V} \right\rfloor$ mean frames so that $\boldsymbol{M} \in \mathbb{Z}_+$.

If we calculate the FTLE field over some sliding time window, it generally does not smoothly lead onto the next time window. This is because the

FTLE's that we calculate may not belong to the same dynamical system [13]. In the case of real world videos, individual dynamical systems are characterised by global motion patterns that are consistent with each other over multiple frames. This motivates us to calculate FTLE's within a time duration T that encompasses distinct dynamical systems. Of course, the durations for which the FTLE is calculated will be chosen by hand, although the automatic detection when a scene is dynamically different from another is an interesting topic for research. Each of these shorter time durations represented by a number of mean frames $\tilde{n}_i$ is called a block. Where each i represents the time window that we calculate the FTLE in. We now have two options for calculating the FTLE:

1. Calculate the full trajectories of all particles then calculate the FTLE field for each $\tilde{n}_i$ mean frame block, re-initializing the grid of particles at the start of every block.

2. Calculate the full trajectories of all particles then calculate the FTLE field for all $\boldsymbol{M}$ mean frames.

For example, if we have a video with $\boldsymbol{N} = 113$ frames, then the velocity field will have 112 frames. Taking the mean velocity field every $n_V = 5$ frames, we then have $\boldsymbol{M} = \left\lfloor \frac{112}{5} \right\rfloor = 22$ mean frames. Then:

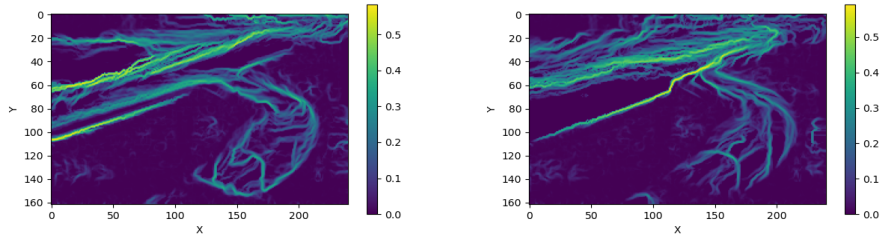
1. If we spot that the dynamical system shifts after 10 mean frames into the video, we may calculate the FTLE field every $\tilde{n}_i = 10$ frames, giving us a total of 2 blocks to calculate the FTLE fields.

2. If the dynamical system doesn't change within the video we calculate the FTLE field for a single time for all 22 mean frames.

In the case of our video, the motion patterns do not change significantly enough to motivate using more than one time window. Thus, we calculate the Forward FTLE field and Backward FTLE field for the full number of $\boldsymbol{M}$ mean frames.

Now we have the time durations, we can advect the particles using the mean velocity fields that we calculated. However, the velocity field is only defined for the grid points that we defined it on. Thus, we must use an interpolation scheme when the particle that is being advected lands on a point in between the grid points. For this, we use `scipy.interpolate.RegularGrid Interpolator` [20] which interpolates the velocity vectors inbetween the grid

points where the velocity field is defined. Once the particles have been advected, we calculate the FTLE field using the equation 2.9. With this, we obtain the following FTLE fields:



(a) Forward FTLE of full video

(b) Backward FTLE of full video

Focusing on the forward FTLE field, we can see a prominent ridge that runs across the FTLE field. As predicted, this ridge corresponds to the barrier between the two lanes. This makes sense as FTLE ridges are curves that particles move most away from. The particles in the left lane are moving towards the camera while the particles in the right lane are moving away from the camera. Thus, a perturbation vector that joins a particle in one lane to a particle in the other lane will stretch by a significant factor, resulting in the FTLE ridge we see above.

(a) $t = t_0$

(b) $t = t_0 + T$

Figure 3.7: A perturbation vector crossing the lane barrier at $t = t_0$ is stretched a lot when the particles are advected to $t = t_0 + T$.

Thus, FTLE ridges of shear flows (like the ones in this example) define boundaries between flows of significantly different dynamics. A similar ridge can be found in the backward FTLE. This one defines the boundary between flows in the right lane and the space to the right of it that has no moving vehicles. Thus, this defines a boundary between stationary and non-stationary flows. We may expect, as we have a ridge in the backwards FTLE, that it corresponds to an attractive region. However, this is not true

as the flow around this boundary is characterised by shear flow which shows no characteristic difference in its flow going forwards in time or backwards in time.

3.4.1 Shear and Normal Flow

Let us now explore this idea of a boundary further. We construct a video of a stream of particles moving in opposite directions to each other in the x-axis. Above the halfway line, the stream of particles move in the positive x direction, while below the halfway line, the stream of particles move in the negative x-direction. The video of the particles is surrounded by a box that defines a periodic boundary. A frame of this video is shown below.

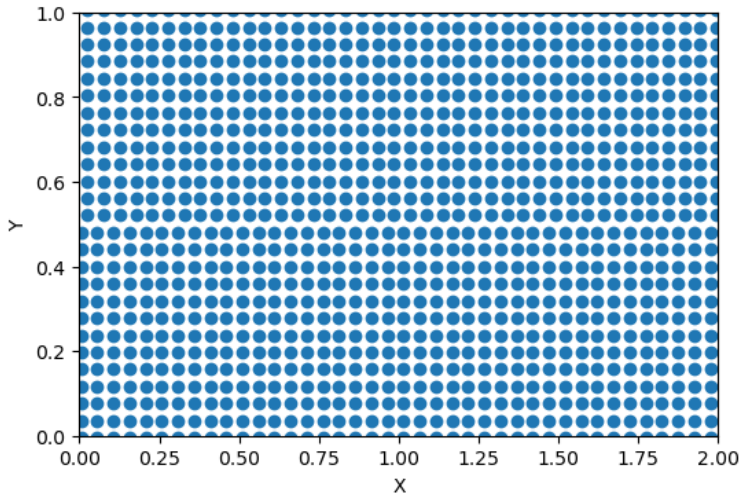
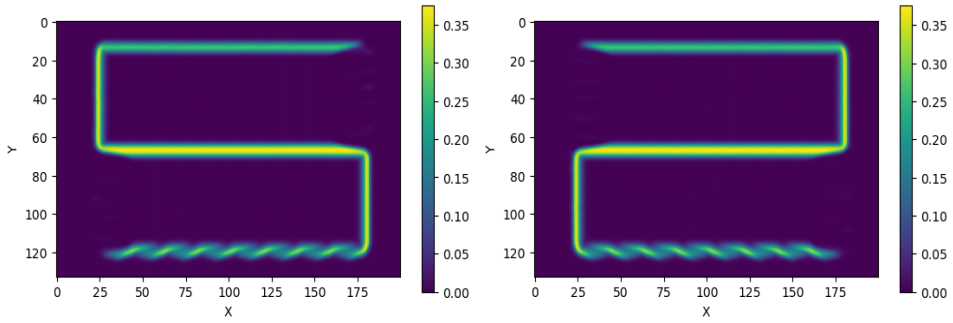


Figure 3.8: A frame of a video of a stream of particles travelling in the positive x direction above $y = 0.5$ and travelling in the negative x direction below $y = 0.5$.

We can then apply the FTLE field analysis to this video by initiating particles over the whole video and using the methods we outlined in Section 3.1. It is important to note that we also initialise particles over the boundary box and in the white space around the boundary box too. As there are no moving particles in the white space, the velocity field will be zero in this space. The output of the FTLE field analysis is:



(a) Forward FTLE of full video

(b) Backward FTLE of full video

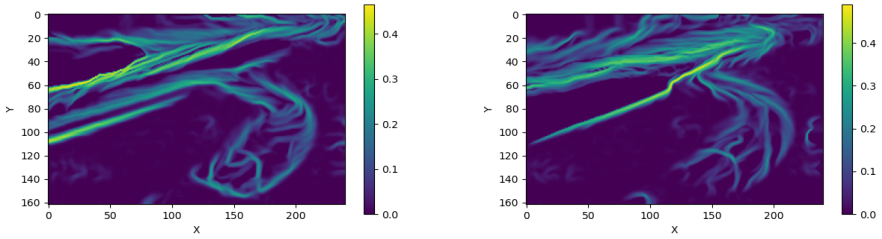
We see a very strong ridge running through the centre of the frame of the video. We expect this because at the $y = 0.5$ line there is very strong shear flow. We also see ridges at the top and the bottom of the FTLE field plot. This represents the top and bottom sides of the boundary box in the video. To explain what is happening, let us focus on the top side of boundary. Above the boundary, the initialised particles are stationary, however below the boundary, these particles are moving in the positive x direction. The perturbation vectors joining the particles in these two different regions will then stretch a very large amount (similarly to 3.7), leading to a strong FTLE ridge. A similar argument can be made for the bottom ridge of the FTLE field (the waves are a product of the Optical Flow method).

The only difference in the forwards and backwards FTLE frames are the positions of the vertical ridges that join the three horizontal ridges together. Again, to help explain the situation, we focus on the left side of the boundary box in the Forward FTLE. The initialised particles outside the boundary box will be immobile while the particles above the $y = 0.5$ line will move away from the left side boundary. Thus, a perturbation vector joining these two regions together will be significantly stretched in the direction normal to the boundary. This produces a strong FTLE ridge. Below $y = 0.5$ however, the particles are being moved towards the left side boundary, thus we see a strong ridge in the backwards FTLE in this place. Similar arguments can be made for the right side boundary.

We see that we have two qualitatively different types of FTLE ridge. One that is produced by strong shear flows and one that is produced by particles moving away from a boundary normal to it. In fact, we will discover that this is what characterises different types of FTLE ridge.

3.4.2 Minor Improvements

The FTLE field that we produced for the motorway video is noisy and there appears to be many candidates for a prominent FTLE field ridge. This makes determining the most prominent motions in the video more difficult. Thus, we apply two improvements. Firstly, we apply a simple gaussian filter to the velocity fields which smoothens out any noise.



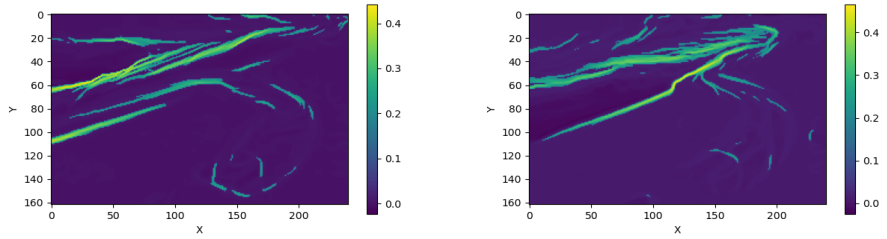
(a) Forwards (b) Backwards

Figure 3.10: FTLE of the full video with Gaussian filter

Secondly, we apply a threshold regime taken from [18]. This is a simple way to single out the most prominent FTLE ridges seen in the FTLE field. We do this by applying some threshold number α and a magnification factor $\beta \in [0, 1]$. Then, given some value $\Lambda_{t_0}^{t_0+T}(\mathbf{x_0})$ of the FTLE for some particle $\mathbf{x_0}$:

$$\hat{\Lambda}_{t_0}^{t_0+T}(\mathbf{x_0}) = \begin{cases} \beta\Lambda_{t_0}^{t_0+T}(\mathbf{x_0}), & \text{if } \Lambda_{t_0}^{t_0+T}(\mathbf{x_0}) \geq \alpha \\ (1-\beta)\Lambda_{t_0}^{t_0+T}(\mathbf{x_0}), & \text{otherwise} \end{cases} \tag{3.9}$$

Thus, values of FTLE that exceed the threshold α are magnified by β. While those values under the threshold are damped by a factor of $1 - \beta$. Applying this to our motorway video we see that we only leave the most prominent FTLE ridges.



(a) Forwards

(b) Backwards

Figure 3.11: FTLE with Gaussian Filter and Threshold at $\alpha = 0.35$, $\beta = 0.95$

3.5 Conclusion

In this chapter, we have seen the applicability of FTLE's to crowd videos or any videos for that matter, so long as they adhere to the criteria given by Optical Flow 3.3.1. We first discretised the method of finding the FTLE field using various numerical techniques. We found that the ridges of the FTLE fields correspond to regions of high density regions in forwards time and low density regions in backwards time. Following in the footsteps of [2], we found strong repelling and attracting ridge curves. Together, these ridges give the boundaries of regions of flow that are distinctly different from each other. These ridges were the locally maximal values of the length of a perturbation vector as particles are moved by the flow of a field.

While the method is robust for videos that have clear and dominant flows in them, the method falls apart in several cases. For example, if the constituents in the video are too sparsely spread out, the FTLE field will not capture the global motion patterns of the video but instead capture the motion patterns of the constituents themselves. This is counterproductive as the whole point of the FTLE is to capture the global motion patterns. On the other hand, if the constituents of the videos are too dense and we see a lot of occlusion (one object blocking the view of another object as they pass each other), the optical flow velocity field will look chaotic and thus the paths of the particles will look random. We will not be able to capture the movements of the constituents accurately. Thus, the density of the constituents plays a key part in the validity of the final output.

Further to this, as we saw in 3.4.1 the FTLE field does not tell us any-

thing about the direction of the perturbations around a particle. Perturbations can be shear to the ridge curves but also normal to them too. We are particularly interested in the normal perturbations as shear perturbations do not define dominant flow patterns, while normal perturbations do. If it is possible to separate the normal and shear perturbations in the flow, we may be able to identify the dangerous regions of the dynamical system in the video. Following in this line, we will look precisely into this idea and create a new method for understanding the tangent and normal perturbations of flow around a particle.

Chapter 4

Normal and Shear perturbation fields

In crowded scene situations, the most dangerous events arise when the constituents experience a compressive force amongst themselves, creating dense regions. Examples of this can be seen everywhere such as in car accidents or crowd crush scenarios. In the interest of Crowd Analysis then, we may study when these flows that create dense regions occur.

Recall that dense regions of flow correspond to trenches in the Forwards FTLE field or ridges in the Backwards FTLE field. Naively, we may find these flows that lead to dense regions by finding these ridges. However, the ridges pf FTLE fields do not always coincide with dense regions. This is because they also coincide with strong shear flows too as we saw in the previous chapter. Shear flows are also of interest to us because of the the global nature in which they outline the flow. We saw before that they are boundaries to distinctly different types of flow. Thus, we focus our attention on identifying where shear and non-shear flows occur with respect to ridges of the FTLE field.

Given a ridge curve, non-shear flows are flows that travel normal to the ridge curve. In contrast to this, shear flows are flows that travel tangent to ridge curves. In reality, most flows to the ridge curves are not perfectly normal or tangent. Thus, the problem reduces to finding the ridge curves and measuring the amount of flow normal or tangent to the ridge curve. To do this, we follow in the footsteps of George Haller [12] and give a mathematical definition for where these flows occur in the mathematical setting. We then move to the Crowd Video setting where the mathematical structure does not apply smoothly, leading us to a first step into finding these normal and

tangent flows in crowd videos.

4.1 Derivation of Normal and Tangent perturbation fields

In [12], Haller defines a material surface as follows:

Definition 4.1.1 (Material Surface)**.** A **material Surface** is the $t = const.$ slice of an invariant manifold $\boldsymbol{\mathcal{M}}$ of system 2.1 in the extended phase space of $U \times [a, b]$ generated by an $n-1$ dimensional surface of initial conditions $\boldsymbol{\mathcal{M}}(t_0)$ by the flowmap $F_{t_0}^t$:

$$\boldsymbol{\mathcal{M}}(t) = F_{t_0}^t(\boldsymbol{\mathcal{M}}(t_0)) \tag{4.1}$$

where $dim(\boldsymbol{\mathcal{M}}(t)) = n-1$, $t, t_0 \in [a, b]$ and $\boldsymbol{\mathcal{M}}(t_0) \subset U$.

In the case of 2-dimensions, the material surface is a material curve. We then would like to quantify how much the particles around this material curve move in the normal and tangent directions with respect to the curve itself.

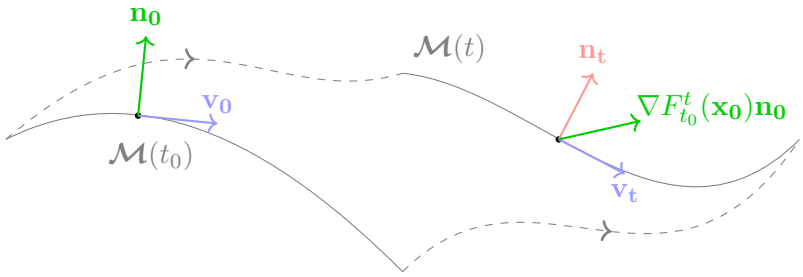


Figure 4.1: The initial material curve $\boldsymbol{\mathcal{M}}(t_0)$ is advected to its later time position $\boldsymbol{\mathcal{M}}(t)$. The unit normal and unit tangent vectors are also both advected to later times

Firstly, consider a point $\mathbf{x_0} \in \boldsymbol{\mathcal{M}}(t_0)$, then a unit normal $\mathbf{n_0}$ to the material curve at this point lies in the normal space $N_{\mathbf{x_0}}\boldsymbol{\mathcal{M}}(t_0)$. A unit tangent vector $\mathbf{v_0}$ to the point lies in the tangent space $T_{\mathbf{x_0}}\boldsymbol{\mathcal{M}}(t_0)$. As the point is advected in time to a new position by the Jacobian $\mathbf{x_t} = \nabla F_{t_0}^t(\mathbf{x_0})\mathbf{x_0} \in \boldsymbol{\mathcal{M}}(t)$, the tangent vector is also similarly advected $\mathbf{v_t} = \nabla F_{t_0}^t(\mathbf{x_0})\mathbf{v_0} \in T_{\mathbf{x_0}}\boldsymbol{\mathcal{M}}(t)$.

However, when the normal is advected in the same way there is no guarantee that it is advected to the normal space $N_{\mathbf{x_0}}\boldsymbol{\mathcal{M}}(t)$. Thus, the normal vector will be some vector pointing in some arbitrary direction $\nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0}$. The unit normal to the invariant manifold at later times then lives in the later time normal space $\mathbf{n_t} \in N_{\mathbf{x_t}}\boldsymbol{\mathcal{M}}(t)$.

With this, Haller then quantifies the growth of perturbation vectors that are normal to the material curve given by the normal repulsion rate:

Definition 4.1.2 (Normal Repulsion Rate)**.** The **Normal Repulsion Rate** is given by:

$$\rho_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) = \langle \mathbf{n_t}, \nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0} \rangle. \tag{4.2}$$

Let us convince ourselves that this is true. Substituting in the scalar product equation:

$$\begin{aligned} \langle \mathbf{n_t}, \nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0} \rangle &= \|\mathbf{n_t}\| \|\nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0}\| \cos\theta \\ &= \|\nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0}\| \cos\theta \end{aligned}$$

where we used the fact that $\mathbf{n_t}$ is unit length in the second line. From the picture below, it is clear that $\|\nabla F_{t_0}^t(\mathbf{x_0})\mathbf{n_0}\| \cos\theta$ is simply the projection of the advected normal vector onto the axis facing in the direction normal to the material curve. Thus, giving the normal growth rate of the advected normal.

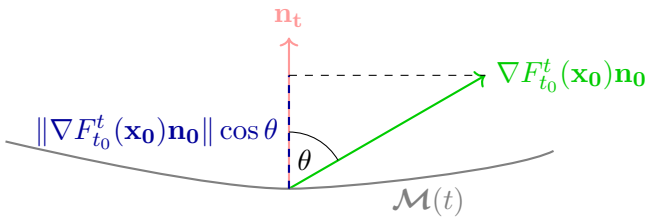


Figure 4.2: The advected normal vector is projected onto the normal to the material curve. This gives the growth rate of the perturbation vector in the normal direction.

Haller also gives the tangential growth rates:

Definition 4.1.3 (Tangent Repulsion Rate)**.** The **Tangent Repulsion Rate** is given by:

$$\sigma_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) = \sqrt{\|\nabla F_{t_0}^t(\mathbf{x_0})\mathbf{v_0}\|}. \tag{4.3}$$

These definitions give a concrete method of measuring how much the perturbation vectors are growing with respect to the tangent and normal of the material curve. We may also define the ratio of these two quantities:

Definition 4.1.4 (Repulsion Ratio)**.** The **Repulsion Ratio** is given by the ratio of the Normal Repulsion Rate to the Tangent Repulsion Rate. It is given by:

$$\nu_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) = \frac{\rho_{t_0}^t(\mathbf{x_0}, \mathbf{n_0})}{\sigma_{t_0}^t(\mathbf{x_0}, \mathbf{n_0})}. \tag{4.4}$$

If $\nu_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) > 1$, then the normal perturbations dominate over the shear perturbations. We are interested in these cases as they are indicative of areas of the flow that are more normal repelling than shear repelling which are precisely the sorts of flows that are dangerous to crowds.

Haller also gives a simple equation for the calculation of the Normal and Tangent Repulsion Rates.

$$\rho_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) = \frac{1}{\sqrt{\langle \mathbf{n_0}, C_R(\mathbf{x_0})^{-1}\mathbf{n_0}\rangle}} \tag{4.5}$$

$$\sigma_{t_0}^t(\mathbf{x_0}, \mathbf{n_0}) = \sqrt{\langle \mathbf{v_0}, C_R(\mathbf{x_0})\mathbf{v_0}\rangle} \tag{4.6}$$

Using this, we may expect that we can characterise the tangential and normal growth in the FTLE fields produced in our CVA analysis. However, it is not as easy as applying this theory straight into our CVA analysis. This is because to use this theory, the ridge curves of the FTLE must be invariant manifolds. In general, it is not the case that ridge curves in the FTLE of crowd videos are invariant manifolds. This is because they are invariant sets in the extended phase space, thus they move with the flow of the particles. It follows then, that in the limit of infinite time, FTLE ridges do converge to invariant manifolds. Of course, in our real world applications an infinite time limit does not exist. Thus, we make the assumption that the FTLE ridge curves approximately equate to invariant manifolds of the dynamical system. We may do this because we take the time interval that we calculate the FTLE over to be the whole duration of the video. This means we are evaluating the FTLE for the largest length of time possible. Although this length of time is not equivalent to infinity, it is as close as possible to it. It follows then, that under this approximation, longer videos will be more accurately approximated to invariant manifolds of the dynamical system. Even with this approximation however, we are able to make out the normal flow structures and tangent flow structures that we couldn't make out in the FTLE field.

4.2 Methodology

We start where we left off in chapter 3 where we were able to find the backwards and forwards FTLE fields for any given video. From this, our first step is to identify the FTLE ridge curves and create arrays of points that define each of the ridge curves. These arrays will look like an $n_p \times 2$ array, where n_p represents the number of points in the curve. Given an $M \times N$ matrix array of FTLE values, we may apply a threshold 3.9 and single out the most prominent ridge curves. On our motorway video, this looks as follows:

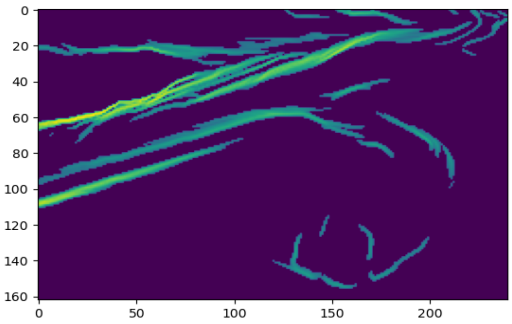


Figure 4.3: The FTLE field with a threshold of $\alpha = 0.3$ and $\beta = 1$

Although we have the most prominent ridge curves, the curves are not distinct enough as they are many pixels wide. This will make it harder to define curves as an array of single points strung together. Thus, we create an "outline" of the curves only leaving the most maximal points in a local neighbourhood. More precisely, we take a square with side length $L \in \mathbb{Z}_+$ (such that $L \ll M, N$) and place it on one of the non-zero pixel values. Then we find the maximal value inside this box and store the result inside an array. After this, we move the box to another non-zero pixel and repeat the process there. After going through all the pixels, we are left with an array storing all of the locally maximal points in the FTLE field.

These locally maximal ridge points give the "outline" of the ridges that we are looking to discretise. Now, we may sort these points into distinct curves. Given any number of curves $\boldsymbol{\gamma_1}, \boldsymbol{\gamma_2}, ..., \boldsymbol{\gamma_c}$, each curve is populated by a number of sorted points $\boldsymbol{\gamma_i} = \{\mathbf{p_1^i}, ..., \mathbf{p_e^i}\}$ ordered so that they outline a ridge curve. We define the endpoints of this curve:

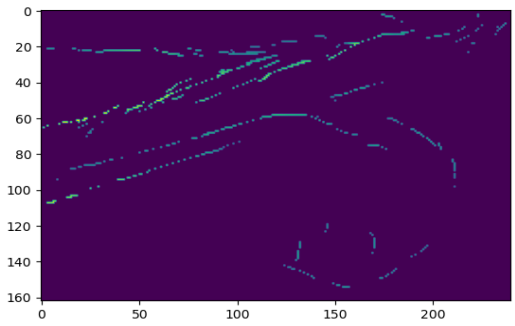


Figure 4.4: The locally maximal ridge points with $L = 11$

Definition 4.2.1 (Endpoint)**.** The **endpoints** of a curve $\boldsymbol{\gamma_i}$ are given by the points $\mathbf{p_1^i}$ and $\mathbf{p_e^i}$ in the ordered set of points $\boldsymbol{\gamma_i}$ where plotting the points in order traces out a ridge curve of the FTLE field $\Lambda_{t_0}^t(\mathbf{x_0})$.

Given this, the idea is to start with any point $\mathbf{P}$ that is yet to be sorted and to compute its distance to all endpoints $\mathbf{p_1^i}, \mathbf{p_e^i}$ for $i \in 1, .., c$. We then find the minimum such distance denoted by $\mathbf{\Delta P_{min}}$. Let the endpoint whose distance is closest to $\mathbf{P}$ be denoted by $\mathbf{p^i_{closest}}$, then if the minimal distance $\mathbf{\Delta P_{min}}$ is smaller than some radius r, we add the point $\mathbf{P}$ to the curve $\boldsymbol{\gamma_i}$ such that it lies next to $\mathbf{p^i_{closest}}$ and becomes an endpoint itself. If $\mathbf{\Delta P_{min}}$ is greater than the radius, then we create a new curve $\boldsymbol{\gamma_{c+1}}$ with $P \in \boldsymbol{\gamma_{c+1}}$.

Example 4.2.2. Let us have the simple case of a single curve $\gamma_1 = \{(0,0),(1,1),(2,2)\}$ with radius $r = 2$. In this case, the there are only 2 endpoints $\mathbf{p_1^1} = (0,0)$ and $\mathbf{p_e^1} = (2,2)$. Consider the unsorted point $\mathbf{P} = (3,3)$. The distance from this point to all end points is given by:

$$\mathbf{\Delta P_1} = \|\mathbf{P} - \mathbf{p_1^1}\| = 3$$
$$\mathbf{\Delta P_e} = \|\mathbf{P} - \mathbf{p_e^1}\| = 1,$$

and so,

$$\mathbf{P_{min}} = \mathbf{p_e^1} = (2,2).$$

Clearly $\mathbf{\Delta P_{min}} < r$ so we add it to the curve next to the edge point $\mathbf{p_e^1}$. Thus, the new curve is given by $\gamma_1 = \{(0,0),(1,1),(2,2),(3,3)\}$ with new endpoints $\mathbf{p_1^1} = (0,0)$ and $\mathbf{p_e^1} = (3,3)$.

In the case that $r = \frac{1}{2}$, $\Delta\mathbf{P_{min}} > r$ so we would initialise a new curve $\gamma_2 = \{(3,3)\}$ where the endpoints would be $\mathbf{p_i^2} = (3,3)$ and $\mathbf{p_e^2} = (3,3)$

Repeating this process for all points in the array of locally maximal points, we arrive at a full set of ridge curves $\boldsymbol{\gamma_1}, ..., \boldsymbol{\gamma_c}$.

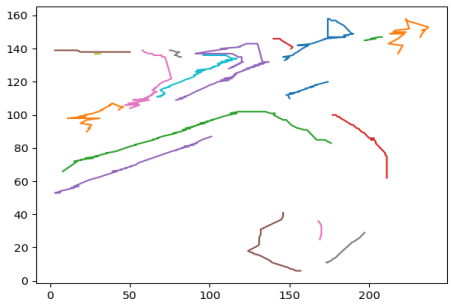


Figure 4.5: All ridge curves $\boldsymbol{\gamma_1}, ..., \boldsymbol{\gamma_{18}}$ for the motorway video

The ridge curves we see above are not smooth and not indicative of the shape of the FTLE ridge curves. To remedy this, we apply a spline approximation via the `scipy.interpolate.splprep` and `scipy.interpolate.splev` functions from the `scipy` [20] library in Python. These functions approximate a given curve $\boldsymbol{\gamma_i}$ to a smooth cubic spline curve and then evaluates the curve for some given parameter values $s \in \boldsymbol{I}$ where $\boldsymbol{I}$ is some closed interval. Then, the points along the curve are now given by $\mathbf{p_j^i} = \mathbf{p_i}(s_0)$ for some $s_0 \in \boldsymbol{I}$. The result of this, is the following:

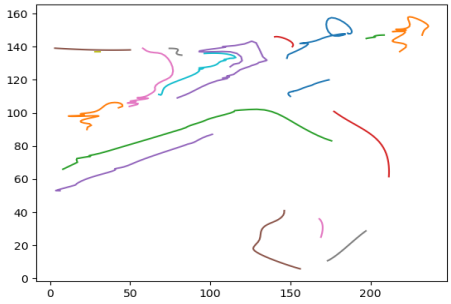
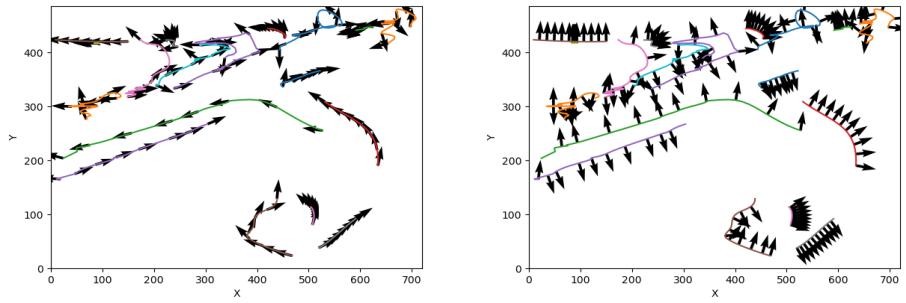


Figure 4.6: Cubic spline approximation applied to the ridge curves $\boldsymbol{\gamma_1}, ..., \boldsymbol{\gamma_{18}}$ for the motorway video

As can be seen, the curves are much smoother now. Using these curves, we now find the tangent and normal vectors for each point along the curve. We dp this by using the `scipy.interpolate.splev` function again. It is able to calculate it's first derivatives at any specified point along the curve. Thus, for some ridge curve $\boldsymbol{\gamma_i} = (x_i(s), y_i(s))$ and a point $\mathbf{p_i}(s_0)$ for some $s, s_0 \in \boldsymbol{I}$, we calculate the tangent and normal vectors to the ridge curve at these points:

$$\mathbf{v_i}(s_0) = \left(\left.\frac{dx_i(s)}{ds}\right|_{s=s_0}, \left.\frac{dy_i(s)}{ds}\right|_{s=s_0} \right),$$
$$\mathbf{n_i}(s_0) = \left(-\left.\frac{dy_i(s)}{ds}\right|_{s=s_0}, \left.\frac{dx_i(s)}{ds}\right|_{s=s_0} \right),$$

where we defined the normal as the unit tangent vector rotated anticlockwise by $\frac{\pi}{2}$. Thus, we obtain tangent and normal vectors for each point $\mathbf{p_i}(s_0)$ along each curve $\boldsymbol{\gamma_i}$. The pair of vectors $(\mathbf{v_i}, \mathbf{n_i})$ define a moving frame along the curve $\boldsymbol{\gamma_i}$.



(a) Tangent Vectors (b) Normal Vectors

Figure 4.7: The unit tangent and unit normal vectors for a select few points along the ridge curves.

Recall that we are trying to describe the growth of perturbation vectors in precisely the directions given by $(\mathbf{v_i}, \mathbf{n_i})$. We now initialise particles on each point $\mathbf{p_i}(s_0)$ along the ridge curves $\gamma_\mathbf{i}$ with a grid of 3×3 particles, the centre particle being the point $\mathbf{p_i}(s_0)$. These particles are aligned to the corresponding moving frame for each point on the curve. We see this below:

At this point, we have reached essentially the same point as in the application of the FTLE to the CVA, where we initialised a grid of particles

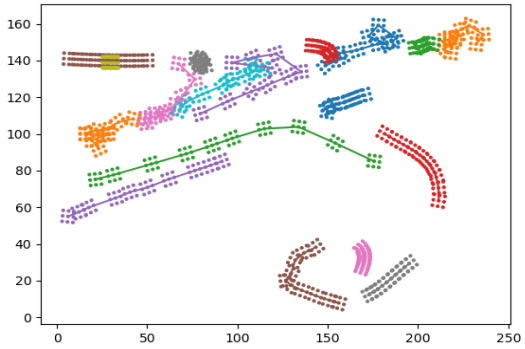


Figure 4.8: Particles are initialised onto the grid oriented with the moving frame. Note that the distances between particles and the number of particles have been altered for clarity.

on the whole frame. We now advect these particles with the flow. This is a simple case of iteratively applying the Forward Euler method. Once we have advected the particles for the full duration of the video, we obtain the following map of the particles:

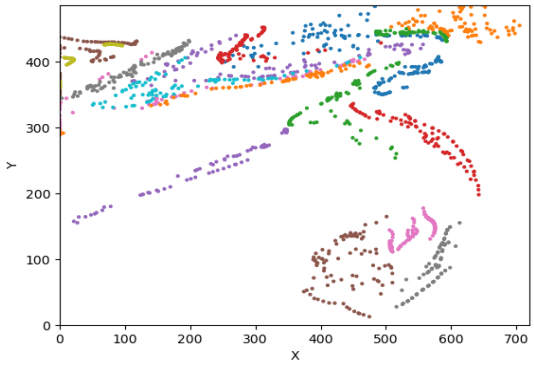


Figure 4.9: Advected particles where each colour represents a corresponding ridge line that each particle originated from.

Observing this figure, we do see some clear examples of shear flow and normal flow. The red particles that started on the middle-right of the frame clearly end up sheared with respect to the flow. Thus, for this curve we

expect to see high shear growth values. The collection of pink particles at the bottom of the frame show strong normal perturbations. Thus, we expect the growth of normal vectors there to be strong.

We now attempt to calculate the Normal Repulsion Rates 4.1.2 and Repulsion Ratio 4.1.3 using equations 4.5. To calculate this, we need the initial tangent vector $\mathbf{v_i} \in \mathcal{M}(t_0)$, the initial normal vector $\mathbf{n_i} \in \mathcal{M}(t_0)$ and the Jacobian $\nabla F_{t_0}^t(\mathbf{x_0})$. We already have the initial tangent and normal vectors. Then, all that is left to calculate is the Jacobian. We may simply do this by using central finite differences to calculate the partial derivatives on the centre particles in the 3×3 grid. Putting all of this together, we obtain the Tangent Repulsion Rate and the Repulsion Ratio:

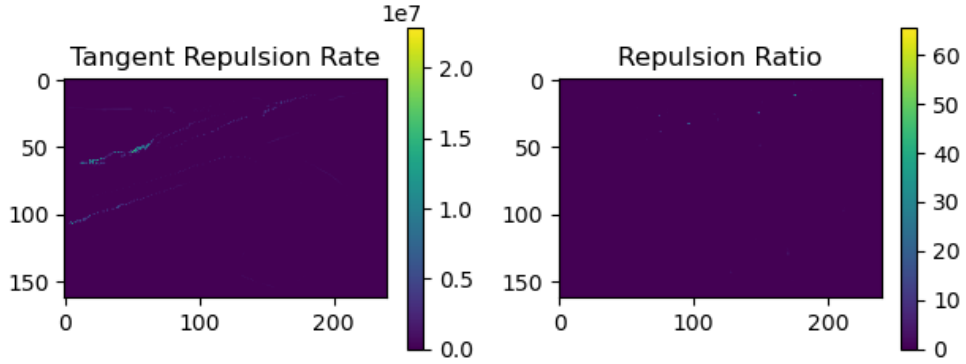


Figure 4.10: The Tangent Repulsion Rate and Repulsion Ratio for the motorway video

This perturbation field is not particularly revealing. Although there is a small cluster of high values in the left of the Tangent Repulsion Rate which is as expected, as this video was mainly comprised of shear flows. However, the Repulsion Ratio is void of any high values. We plot the Normal Repulsion Rate Field and the base 10 logarithm of the tangent repulsion rate below 4.11:

Looking at this figure, the order of values in the Normal Repulsion Rate field is $\sim 10^4$ while the Tangent Repulsion rate is in the order of $\sim 10^5$ to $\sim 10^7$. Thus, we expect the order of the Repulsion Ratio to be of order 10^{-1} to 10^{-3}. This explains why Repulsion Ratio field was empty. It suggests that the dominant flows in the video are shear flows for this dynamical system, which we know to be true.

As a proof of concept, let us return to the video with a stream of particles going in opposite directions 3.8. We saw before that the FTLE field

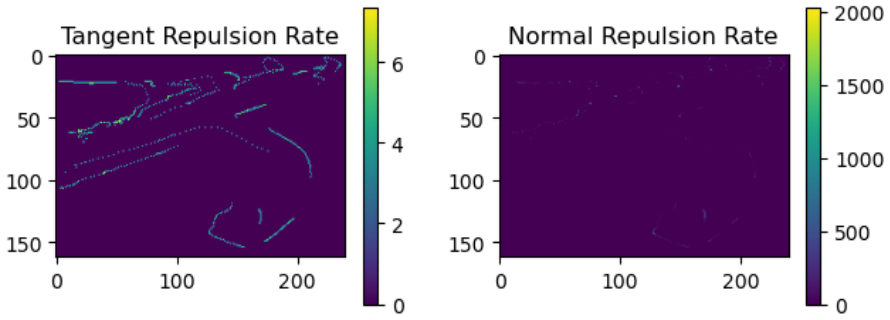


Figure 4.11: The base 10 logarithm of the Tangent Repulsion Rate and usual Normal Repulsion Rate field for the motorway video

described strong shear flow on the $y = 0.5$ centre line and strong normal flow to the ridge curve on the top left and bottom right side of the boundaries parrallel to the y-axis. Let us confirm our suspicions.

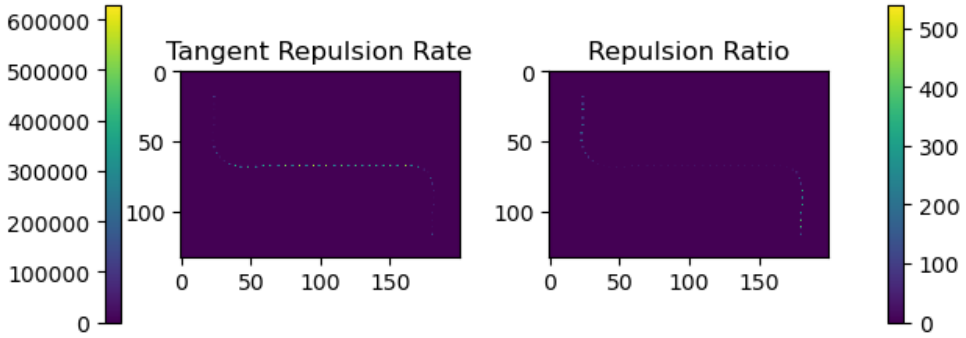


Figure 4.12: The Tangent Repulsion Rate and the Repulsion Ratio

We see clearly that the Tangent Repulsion rate only has ridges on the central line, however, the opposite can be seen in the Repulsion Ratio. The high values of the FTLE lies on the sides which corresponds to the sides of the boundary box. This is exactly as we predicted. The flows that are tangent to the ridge curve are identified by the central ridge in the tangent repulsion rate and the dominant flows that are more normal to the ridge curve are identified by the ridges parallel to the y-axis in the Repulsion Ratio. Thus we have a successful method of identifying the normal and tangent perturbation ridges.

4.3 Conclusion

Although we are able to successfully split the normal and tangent directions of growth, this is by no means a perfect method. For one, the way in which we map out the ridge curves is not efficient in the slightest. Often times the curves that we end up identifying are not smooth and don't accurately represent the ridge curves. There are more efficient methods that pick out ridge curves from scalar fields for better results, thus, a next step would be to implement these methods.

Another drawback of the current method is that the tangent and normal ridge curves that we end up with are not smooth curves. They are instead a discrete set of points that determines an outline of the ridge curve. We lose a significant amount of detail with this. It would be ideal to have a method that preserves the continuity of the ridge curves that we find from the FTLE plot.

Chapter 5

Concluding Remarks

We set off on this journey to explore the possibilities of applying LCS theory to CVA. We started by deriving the fundamental building blocks of LCS theory, the FTLE field. From there, we applied this method to a crowd video. We successfully found the ridge lines which led us to the different types of ridges in the FTLE field. We then found the normal and tangent perturbation ridge curves which allowed us to see what types of flows were more dominant in the dynamical system.

However, the methods developed fall short of what would be required to be a benefit in the real world. Firstly, the methods that we developed were highly reliant on the optical flow methods. While simple to implement and easy to understand, using Farneback Optical Flow has it's drawbacks. The criteria 3.3.1 stopped us from using a lot of crowd video datasets that were available online. Thus, finding better optical flow methods that are more robust to noise and lack of texture would be a good next step.

Another issue is that a lot of the videos would simply not possess the strong dominant flow structures that are required for the use of these methods. Lots of videos would depict people walking in streets or cars on a highway which is very useful for the detection of shear crowd flow, but for the dominant normal flow structures we were seeking, they were lacking. Thus, there is a need for a dataset that shows these normal flow structures. It is also important to focus on a method that is more sensitive to minute movements of the constituents in the video, thus allowing us to catch more detail in the videos with little dominant flow structures. This would be helpful as these are the videos that are most prevalent in the real world.

In Chapter 4, we go on to explore the different types of ridges in the FTLE field inspired by [12]. We stopped at the definitions of the repulsion

ratio and the tangent repulsion rates, but Haller goes on to give a full definition of a Hyperbolic LCS in a FTLE field. He and Farazmand follows this up with a paper [8] that explains the precise numerical method that we may use to find these Hyperbolic LCS in the FTLE field. The method that they give is more complete than ours and is applicable to the CVA case, although there may be some numerical complications. We strongly urge those interested to read this paper and to attempt applying it to CVA.

Although there is a lot more in this topic to be pursued, we have certainly shown the potential that LCS theory has in CVA. Thus, as an exploration of the possibility of extending the influence of Lagrangian Coherent Structure Theory to Crowd Video Analysis, we believe it was successful. There are many more unexplored avenues that one may take in this direction and we hope that we were able to garner even the smallest interest in this topic.

Bibliography

[1] N Al-Mughrabi. More than 700 pilgrims die in crush in worst haj disaster for 25 years, 2015.

[2] S Ali and M Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] B.E.Moore B.Solmaz and M.Shah. Identifying behaviors in crowd scenes using stability analysis for dynamical systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2064–2070, 2012.

[5] C.R.Harris, K.J.Millman, and S.J.Walt et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[6] K Dawson-Howe. *A Practical Introduction to Computer Vision with OpenCV*. Wiley, 2014.

[7] D.Helbing and P.Molnár. Social force model for pedestrian dynamics. *Physics Review E*, 51:4282–4286, May 1995.

[8] M Farazmand and G Haller. Computing lagrangian coherent structures from their variational theory. *Chaos, An Interdisciplinery Journal of Non-linear Science*, 22, 2012.

[9] G Farneback. Two-frame motion estimation based on polynomial expansion. *Scandinavian Conference on Image Analysis*, 2003.

[10] J Gambrell. Ap count: Over 2,400 killed in saudi hajj stampede, crush, 2015.

[11] G Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 2002.

[12] G Haller. A variational theory of hyperbolic lagrangian coherent structures. *Physica D*, 2010.

[13] G Haller. Lagrangian coherent structures. *Annual Reviews*, 47, 2015.

[14] G Haller. *Transport Barriers and Coherent Structures in Flow Data: Advective, Diffusive, Stochastic and Active Methods.* Cambridge University Press, 2023.

[15] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[16] B D Lucas and T Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

[17] et al M Bendali-Braham, J Weber. Recent trends in crowd analysis: A review. *Machine Learning with Applications*, 2021.

[18] et al M K Lim, C S Chan. Detection of salient regions in crowded scenes. *Electronics Letters*, 2014.

[19] A Pikovsky and A Politi. *Lyapunov Exponents: A Tool to Explore Complex Dynamics.* Cambridge University Press, 2016.

[20] P.Virtanen and R.Gommers et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[21] A.Oyama R.Mehran and M.Shah. Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–942, 2009.

[22] N T Ouellette S Balasuriya and I I Rypina. Generalized lagrangian coherent structures. *Physica D: Nonlinear Phenomena*, 372, 2018.

[23] F Lekien S C Shadden and J E Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212, 2005.

[24] S.Blunsden and B.Fisher. The behave video dataset: ground truthed video for multi-person behavior classification. *Annals of the BMVA*, 2010(4):1–11, 2010.

[25] G.Vizzari S.D.Khan and S.Bandini. Identifying sources and sinks and detecting dominant motion patterns in crowds. *Transportation Research Procedia*, 2:195–200, 2014.

[26] et al. T Li, H Chang. Crowded scene analysis: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25, 2015.

[27] H.Theisel T.Senst, A.Kuhn and T.Sikora. Detecting people carrying objects utilizing lagrangian dynamics. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, pages 398–403, 2012.

[28] Q.Yu W.Bao and Y.Kong. Uncertainty-based traffic accident anticipation with spatio-temporal relational learning. In *ACM Multimedia Conference*, May 2020.

[29] Q Yu X Zhang and H Yu. Physics inspired methods for crowd video surveillance and analysis: A survey. *IEEE Access*, 2018.

[30] et al Y. Zhao, M. Yuan. Crowd macro state detection using entropy model. *Physica A: Statistical Mechanics and its Applications*, 2015.