Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Qualitative Analysis of Optical Flow for Motion Anomaly Detection

Pranav Megarajan

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr.Paul G. Ploeger
Santosh Thoduka M.Sc.

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Pranav Megarajan

Abstract

Motion anomaly detection is the process of detecting anomalies by analysing motion information. While detecting unusual and abnormal movements from visual information and past experiences is easy for humans, it is not the same for robots. Motion anomaly detection plays a crucial role in developing more safe and robust robots. It can aid in the prediction of any adversarial situations beforehand and thus help avoid or overcome the situation. Optical flow gives the information of apparent motion of objects in a scene captured by a visual perception sensor. Most motion anomaly detection algorithms use the optical flow information in order to model the normal motion patterns and detect anomalies as any motion patterns that do not conform to the learned model. This approach is particularly problematic when considering the varied contexts in which motion anomaly detection is necessary.

This work focuses on developing a context-independent motion anomaly detection algorithm which overcomes the problem with learning based approaches. The developed method uses techniques devised to analyse oriented textures and vector fields. In addition to making use of these techniques, this work suggests improvements to the existing techniques as well. A few criteria on which the motion anomalies can be detected have also been devised. The motion anomaly detection algorithm has been tested on a publicly available dataset and demonstrations have been made to show the generalizing capability of the algorithm.

Acknowledgements

I thank my supervisors Prof. Dr. Paul G. Plöger and M. Sc. Santosh Thoduka for providing me the opportunity to work on this Research and Development project. I am much obliged to thank M. Sc. Santosh Thoduka for his continuous guidance and valuable support throughout the entire duration of this project. I would also like to thank Pooja Bhat for her valuable suggestions in order to improve this report. Finally, I extend my gratitude to my family and friends for their enduring support, undying inspiration and endless encouragement.

Contents

List of Figures

1

Introduction

Detecting unexpected movements or motion anomalies in the environment can be easy for humans and in some cases comes naturally by reflex but its not the same for robots. The increasing demand for robots to be more safe and robust in its operating environment has driven the growth in research in several domains of robotics. Anomaly detection is one of them[12]. Anomalies can be detected in many different ways. This work focuses on solving the problem of anomaly detection by analysing information about motion. Autonomous robotic systems are designed to function as expected in a dynamically changing real-world environment without any kind of human intervention in carrying out the assigned tasks. In order to maintain the autonomy, robots have to deal with unknown environments and several issues of uncertainties in simple and complex environments. A first step towards dealing with it will be to design the robot with the intelligence to detect any sort of abnormalities. This leads us to building and implementing robotic systems which are capable of detecting anomalies at a early stage before any faults occur and help us overcome the anomalous situation.

Anomalies are carefully defined with respect to the robotic task and environment at hand. Since anomaly is a context-specific term, it also becomes quite challenging to design a robust and flexible anomaly detection algorithm. The anomalies that we want to detect regard to unexpected movements or motion anomalies in the environment and the robot itself. The motion information can be collected in several ways but this work focuses on collecting and analysing motion information through

visual perceptions. This typically involves collecting raw information by a visual spectrum camera mounted on the robot or placed in the working environment of the robot. It also requires image processing algorithms to work on the collected information. One such image processing tool that is available is known as Optical Flow.

Optical Flow is the pattern of apparent motion of objects through an image sequence caused due to the relative motion between the object and the camera [62]. Technically, it can be considered as a two dimensional vector field which gives us the displacement and directional information showing the motion of objects from one frame to another in the image sequence. Calculating optical flow gives us directional information about motion in the regions of interest within the images. This gives us the opportunity to capture anomalies that can be identified with respect to motion. Once the nominal states are defined with respect to motion, the anomalies are identified whenever the the regions of interest or the objects of interest do not exhibit nominal motion patterns.

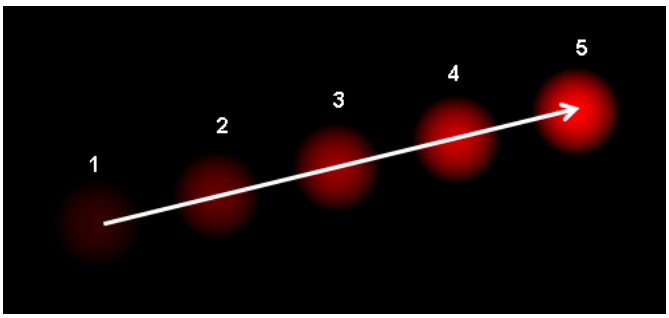


Figure 1.1: Optical flow vector of an object in motion [39]

1.1 Motivation

The primary motivation behind motion anomaly detection is to be able to predict forthcoming adversarial scenarios and aid in precaution and detect symptoms of possible faults. Optical flow gives us rich motion information from videos and sequences of images which can aid in this process of anomaly detection. A qualitative analysis on optical flow can be useful in order to generalize and condense the information provided by optical flow and can help us make decisions and actions in an intuitive manner. For real-time systems which are required to be robust in the most

dynamic of environments, anomaly detection plays a major role. One such system would be an automated surveillance system, where there is a hard real-time constraint for it to analyse and predict anomalies in the most dynamic of environments. There are a plethora of use cases, a few of which will be discussed below, where motion anomaly detection can be of much help in achieving the robustness.

1.1.1 Potential Applications

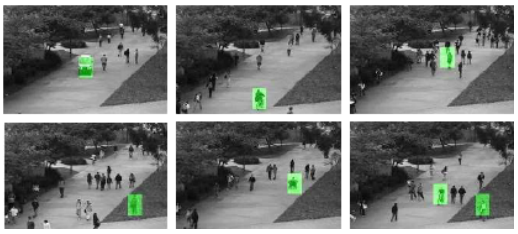


Figure 1.2: Automated Crowd Surveillance [13]

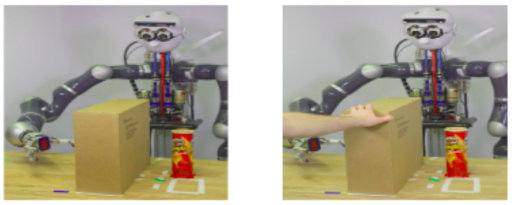


Figure 1.3: Motion anomalies in Manipulation[24]

- **Automated Surveillance:** From traffic signals[32] to patrolling robots [11], motion anomaly detection plays a key role in and stands as the backbone of such systems. In traffic signals and public spaces, surveillance systems detect any unwanted or unexpected movement by vehicles and pedestrians and help in curbing crimes and accidents.
- **Robot Manipulation:** Manipulation tasks aided and guided by vision are becoming increasingly prevelant. Tasks like grasping [25] to tasks like palletizing use visual systems to close the feedback control loop. Detecting motion anomalies pertaining to the robotic task at hand can increase the safety and reliability of such systems.
- **Human-Robot Collaboration:** In order to achieve a fully fledged collaborative environment between humans and robots, robots must not only grow in their reliability but also should be fully aware of its environment and should be precautious in its planning and actions [23]. Detecting anomalous movements in the environment can increase the environmental awareness of the

robotic system even without being given direct information to the robot by a co -working human. This helps in building trust between between humans and robots, hence achieving a fully fledged collaborative workspace.

- **Autonomous cars and Mobile Robots:** Speaking of real-time systems, the constraints can't get much more harder than for autonomous cars, where human lives are at stake. Scenarios such as detecting lane crossing, sudden overtaking , turning at junction points,etc., can be done using refined motion anomaly detection algorithms. In general, mobile robots [30] have to be fully aware of sudden changes that occur in its environment, be it a sudden closing of the door or a person passing by suddenly, etc.

1.2 Challenges and Difficulties

The usual set of challenges in motion anomaly detection comprise of the detector supporting the hard real-time constraints because most use cases involve the system to react quickly in order to avoid faults or dangerous situations. Also as discussed before, anomalies are context-specific and usually once a system is in place, it is hard to extend the system to be able to detect new anomalies.

1.2.1 Context-Specific Nature of Anomalies

Designing algorithms to detect motion anomalies needs a turn-key approach since anomalies are context-specific. One of the most challenging tasks is to come up with a detector that is able to generalize well over all the contexts, with minimum modifications as possible. There needs to be approaches, even if it involves learning techniques [37], that is intuitive so that one can be able to understand and apply the same over different contexts.

1.2.2 Real-time Approach

Motion anomaly detection is implemented in situation which need an immediate response and hence the need for real-time capabilities. Hence, in order to cope up with the requirements, different learning techniques are used to learn all the possible anomalies for a given context and thereby provide the real-time abilities for the system. Learning or detecting motion anomalies is not a straight forward approach and usually the algorithms involve learning complicated patterns from the image sequences.

1.2.3 Robust Detection Methods

Motion information that is collected through visual spectrum cameras are usually very noisy and detection algorithms need to be robust to such noisy information. There are cases where anomalies and nominal situations are difficult to distinguish and noise makes it even more improbable, the only way out is to make our algorithm as robust as possible. Using a non-robust algorithm in such cases would lead us to generate a lot of false positives for anomalies.

1.3 Problem Statement

Most motion anomaly detection algorithms are context specific and make use of a number of different machine learning techniques. In addition to requiring large amounts of data and labeling cost to train these learning models, these approaches only work for situations where the context of the system remains constant. Most research pertaining to motion anomaly detection has been done in the domain of automated surveillance and robot manipulation. The situations under scrutiny in such applications usually consist of a camera that is fixed in its position and motion anomalies are detected from the dynamic environment. This project aims at exploring into methods which can provide alternatives to such previously established approaches.

There exists no method or algorithm that handles motion anomaly detection in an intuitive and robust manner as humans do. Moreover, they are not generalizable

over different applications and especially in the context where the camera frame is dynamic as in the case of mobile robots and autonomous cars. Therefore, instead of using the existing approaches, the challenge is to build an approach that generalizes motion anomaly detection and to make it as context independent from the states and motion patterns of the robotic system. The problem becomes even more complex when one considers the uncertainties that are involved in collecting the data that is used to detect anomalies. These uncertainties usually originate from the vision system and the robots motion estimation processes.

2

State of the Art

This section discusses the state of the art algorithms available for motion anomaly detection and the state of the art for the techniques used in the algorithm built during this project. Motion anomaly detection has been researched only in a limited number of domains and anomaly detection using computer vision is still considered to be an ill-posed problem [4].

The efforts put in to research in motion anomaly detection are mostly for automated crowd surveillance and for robot manipulation and much less comparatively for mobile robots. This is due to the fact that the former two cases deal with motion anomalies with static cameras but mobile robots have to deal with a dynamic camera feed and infer motion anomalies. This complexity of the dynamic camera feed makes for a challenging problem in motion anomaly detection. The underlying sections discuss the varied approaches taken towards detecting motion anomalies.

2.1 Learning Based Approaches

Anomaly detection in general is the problem of detecting the abnormal from the normal. In any particular scenario, it is hard to define the entire set of anomalous situations and hence most algorithms that have been developed make use of the most amount of data with normal scenarios and detect anomalies as outliers. The learning based approaches can be divided into supervised and unsupervised learning.

2.1.1 Supervised Learning

Supervised learning typically require labeled datasets to develop the motion anomaly detector. They use different types of classifiers which are trained on the labeled data to provide a turn-key solution to the problem. The drawback with this approach is that they are problem specific and it's usually infeasible to generalize to new contexts and scenarios. This section consists of approaches which are completely supervised and semi-supervised.

Model-Based Approaches:

This approach involves in building models from the given collection of data belonging to the normal class. The data which does not comply to the built model are characterized as anomalies. Such an approach is especially useful when there are a varied set of normal motions which are not easily distinguishable, and which causes the clustering algorithms to perform badly.

There are number of different models that have been employed till date. Hidden Markov Models(HMM) [27] [66], Markov Random Field(MRF) [26] [8] models are a couple of models that have been used frequently for many use cases [7]. In [19], they discuss a new approach where they extend the one-class support vector machine(SVM).

In [67], they present a novel anomaly detection framework which deals with anomalies by sectioning them into motion anomalies and appearance anomalies. The motion anomalies are dealt with a technique using cut-bin histograms. The appearance anomalies are detected by an approach known as SVDD(Support Vector Data Description), which excludes anomalous objects by obtaining a spherical shaped boundary around the normal objects. The two approaches are then combined to yield a more comprehensive approach towards detecting anomalies in crowd surveillance scenarios. The overall model can be described as in the figure 2.1. This falls under a semi-supervised approach.

In [7], the motion patterns of the objects in the scene are learnt by pixel level probability density functions(pdfs). Gaussian mixture models(GMM) are used to model the motion and the size parameters of the objects at a particular location.

This approach not only helps in detecting motion anomalies but also helps in better detection of objects using the size parameter obtained from the model.

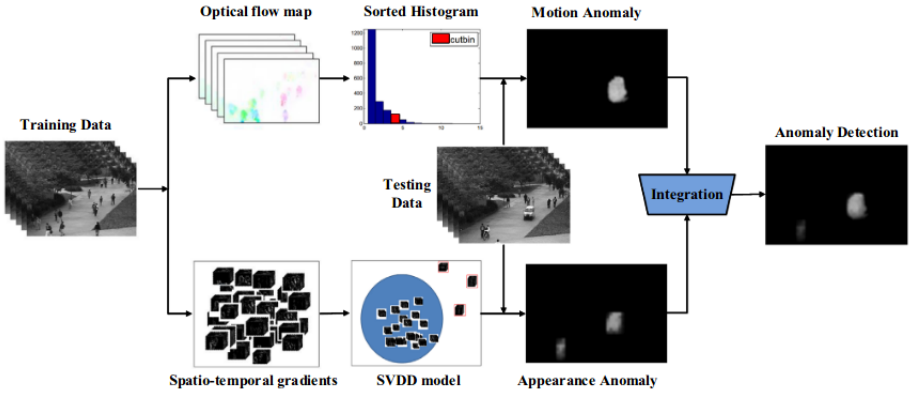


Figure 2.1: **Motion and Appearance cues:** This figure shows how optical flow is aggregated into the model to learn the motion parameters along with the spatio-temporal gradients that help in building the SVDD model. Spatio-temporal analysis plays a major role in motion anomaly detection algorithms. [67]

In [28], they propose a methodology for motion anomaly detection in crowded scenes as well as localization. As in other algorithms, a joint detector of spatial and temporal anomalies is proposed. It also makes of a set of dynamic texture models which are used to produce spatial saliency scores and to model behavior from data to produce temporal saliency scores. By examining the scores of these detectors at progressively larger regions of support to produce globally consistent anomalies. This is especially useful when there is a denser crowd than usual. Thus, we see that model based approaches have been tested to their fullest in the process of motion anomaly detection.

Convolution Neural Networks(CNNs:)

In [46] and [51], the authors propose an interesting way of detecting anomalies. In the latter work, the authors are also able to localize the anomalies. In [46], they use the representational capabilities of CNNs to detect anomalies. The authors keep track of the changes in the CNN features across time in order to distinguish between

normal motions and anomalous motions. They make use of the TCP(Temporal CNN pattern) as described in their work to enhance the predictions along with the optical flow motion features. Instead of examining problem specific features in order to detect anomalies as in the other model based approaches, this method is a much more generalized motion anomaly detector.

In [51],they propose an approach to detect and localize anomalies using fully convolutional neural networks(FCNs). A pre-trained FCN is used to detect anomalies by transferring into an unsupervised FCN. This method also ensures the detection of global anomalies.

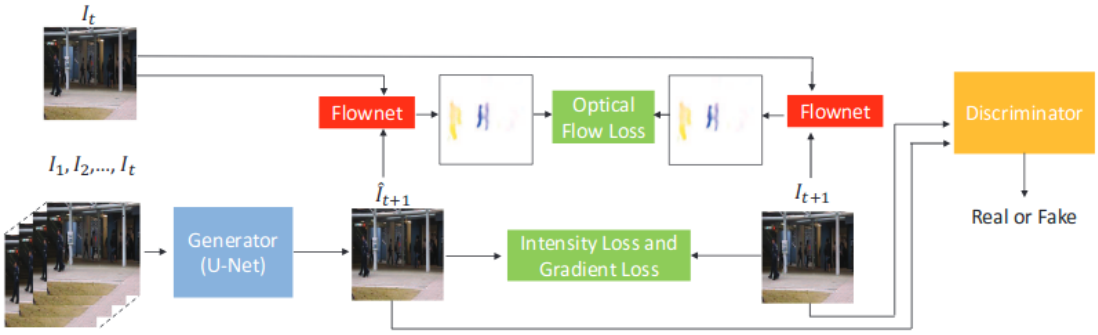


Figure 2.2: **Future Frame Prediction:** The figure shows the application of optical flow constraints and appearance constraints to better the predictions. Flownet is used to calculate optical flow and the discriminator is used to determine whether the prediction is real or fake. [31]

Stepping towards a more intuitive approach, motion anomalies are also detected by making future motion predictions via **future frame predictions** in image sequences. In [31], they propose a method, as shown in figure 2.2, that is able to overcome the drawback of other motion anomaly detection algorithms based on learning which detect anomalies by the minimization of reconstruction error. They detect anomalies by enforcing the optical flow flow between the predicted frame and the ground truth frame to be consistent. In addition to the spatial constraints, enforced by intensity and gradients, the addition of the optical flow constraint helps in better prediction of the normal future frames and thereby segregating the anomalies.

In [29], they propose a dual GAN based future frame prediction, which yields

in reduced blurring in the resulting frame. The other methods focus on directly hallucinating pixel values from the previous frame but this method enforces the future predicted frames to be consistent with the pixel-wise values for flow. This is all done in a dual learning mechanism with a generator and discriminator to enhance the video prediction plays a crucial role in detecting motion anomalies as proposed in [31].

2.1.2 Unsupervised Learning

In this section, we discuss unsupervised learning methodologies that have been implemented for specific cases and work without labeled datasets. In most cases, to identify anomalies from the normal cases, different clustering algorithms are used. There are also other methods like RPCA(Robust Principal Component Analysis) [9].

Clustering

In unsupervised approaches, clustering algorithms play a major role. K-means[33] and its variations are used in a lot of cases. In [52], x-means is used to detect motion anomalies where they are trying to detect collisions via visual data. It is used in approaches to improve grasping prediction. A slight disturbance stands out as anomaly which is detected. In [50], k-means++ is used as hard clustering to seed the EM(Expectation Maximization) algorithm used to detect motion anomalies.

RPCA

In [20], they propose a method to apply RPCA for efficient background subtraction and henceforth detect anomalies. This helps in cases where the background also might contain moving components. By utilizing this method, The background components in motion which form a part of the normal scene are absorbed into the background and can help better detect motion anomalies by subtracting unwanted motion information.

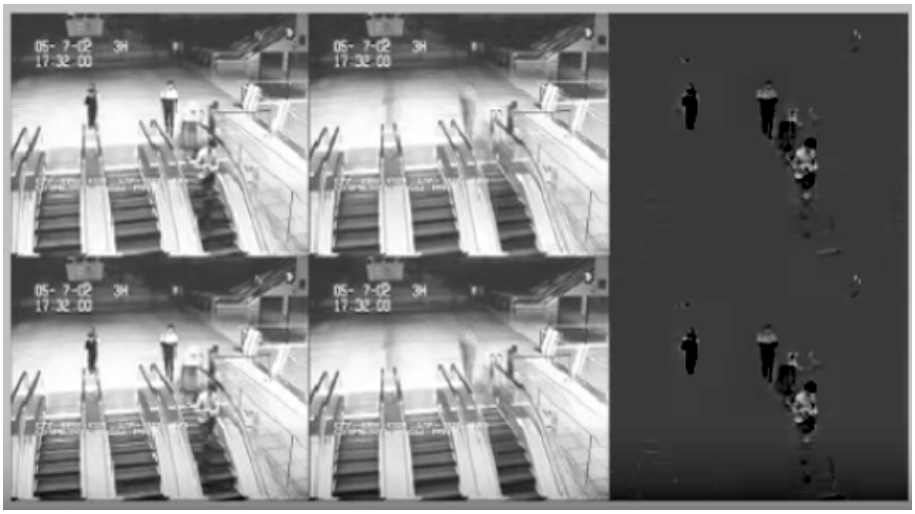


Figure 2.3: **RPCA Background Subtraction:** The escalator movement is absorbed in to the background by using the RPCA method. Motion anomalies can be better detected without having to deal with motion components which are not anomalous. [14]

Limitations: Learning Based Approaches:

The techniques discussed above have drawbacks due to the fact that they are dealing with anomalies, which in itself is context specific. Overall, the techniques perform well, but only for specific problems.

- The models and classifiers require a large amount of data to learn from or to be trained.
- The suggested methods are problem-specific and do not generalize well in other scenarios of motion anomaly detection.

2.2 Non-Learning Based Approaches

This section discusses a couple of algorithms which have not used any of the learning approaches as above. In [36], a motion anomalies are detected by calculating **social force**, a measure of the interaction forces of the people in a crowd. By determining the regions of high and low social forces, anomalies are classified. "The velocity of an individual in the crowd is described as a result of a personal desire force and interaction forces."[1]

In [10], the mobile robot is used for patrolling campuses. The algorithm helps in finding **sparse feature correspondences** between sequence of images. The mobile robot is fed with reference images and the mobile robot is controlled to align the robot along with the reference images using a special algorithm. The anomalies are detected when the correspondence matches goes below a certain threshold. In most of the cases, motion anomalies are detected by using problem-specific approaches. Non-learning approaches work in cases with minimal noise and use conventional techniques as suggested in [12].

In [42] , they suggest an improvised motion anomaly detector based on interaction forces. This is used to detect anomalies in crowded environments. It mainly uses an objective function to minimize the interaction forces and to drift the population of particles to the areas of high motion.

Limitations: Non-Learning Based Approaches

- The non-learning based techniques used are mostly derived from conventional anomaly detection techniques and fail in cases where there is too much noise or when a high degree of control over the system is not possible as in [10].
- The techniques only provide a solution for specific problems and fail to generalize to other typical motional anomaly detection problems.

3

Methodology

In this chapter, all the different techniques and algorithms that were used in the final motion anomaly detection implementation will be discussed in detail. Each of the underlying sections provide understanding and intuition as to why one method might behave the way it behaves during the implementation process. The chapter has been broken up into sections and subsections considering the flow of the final algorithm in mind. The following sections discuss the motion data capturing techniques, data analysis techniques and other key components of the motion anomaly detection algorithm.

3.1 Motion Data

The required motion information for the analysis is collected by measuring the optical flow from the image sequences fed to the detector. There are various approaches to obtain optical flow data. The different techniques in optical flow calculation as given in [62] are:

- Phase Correlation
- Block Based Method
- Differential Method and
- Discrete Optimization Method

Based on the performance factors as suggested in [6], namely the accuracy and density of measurements,the most popular approach has been the differential approach. The state-of-art implementations of the differential approaches compromise of the Lucas-Kanade method and the Gunnar Farneback method. The primary difference between these two is discussed below.

Differential Methods:

1. Lucas-Kanade Method - A Sparse optical flow method, which yields the flow data only along a sparse set of features, typically corners of the objects observed in the sequence of images. The outputs of this method can be visualized as show below.

2. Farneback Method - A Dense optical Flow method, which yields pixel-wise optical flow estimation, which gives us more information to deal with and also enables us to apply our algorithm on the obtained data.

Figure 3.1: Lucas-Kanade Optical Flow [39]

Figure 3.2: Farneback Optical fLow[35]

This project utilizes the output from the latter method. The output of the farneback method is equivalent to obtaining a vector per pixel which takes a lot of computation time which is challenging for real-time applications but is sometimes unavoidable for certain use cases like ours.

3.1.1 Farneback Optical Flow

As proposed in [16] and discussed in [2], Farneback optical flow is obtained based on the spatial and temporal variations in intensity or brightness in the entire set of pixels in an image. The following equations are taken and adapted from [16].

- The algorithm is based on polynomial expansion of a neighborhood set of pixels.
- The polynomial used for approximation is given by :

$$f_1(x) = x^T A_1 x + b_1^T x + c_1$$

- After undergoing a displacement d, the polynomial can be given by :

$$f_2(x) = f_1(x - d) = (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1$$
$$\implies f_2(x) = x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1$$
$$\implies f_2(x) = x^T A_2 x + b_2^T x + c_2$$

- Equating the co-efficients in the above polynomials gives us,

$$A_2 = A_1$$
$$b_2 = b_1 - 2A_1 d$$
$$c_2 = d^T A_1 d - b^T d + c_1$$

- The important observation is that,

$$b_2 = b_1 - 2A_1 d$$

- The required optical flow value can be obtained as follows, as long as A is non-singular,

$$2A_1 d = -(b_2 - b_1)$$
$$d = -\frac{1}{2}A_1^{-1}(b_2 - b_1)$$

3.1.2 Motion Interpretation with Optical Flow:

There is a disadvantage when it comes to interpreting the motion field with the optical flow values obtained.

Motion Field: The motion field can be defined as the projection of the real world 3D motion vectors on the image or camera plane.

Optical Flow: With comparison to motion field, optical flow is just the 2D displacement of pixels in the image plane.

Usually, optical flow corresponds to the motion field but there are special cases where the correspondence fails. One such situation is a rotating barber's pole as shown in figure below.

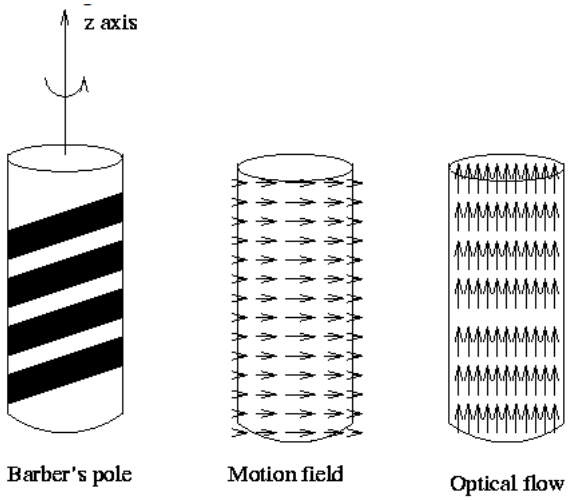


Figure 3.3: Motion Field vs. Optical Flow [47]

Thus, we can see that in cases like the barber's pole, optical flow can only yield the apparent motion of the pixels, the stripes which appear to move upwards,and not the actual real world 3D motion. This is also known as the **Aperture problem**.

In such cases, the motion anomaly detector might fail when trying to judge whether the pole has moved upwards or is still rotating about its axis.

3.2 Dynamical Systems

Dynamical systems can be defined as systems that evolve with time as given by a rule [55]. This rule is usually a differential equation. The rule stands on a state space. The **state space** or usually called the **phase space** is the set of all possible states a dynamical system can take. The number of degrees of freedom of the system is the dimension of the phase space which is also the dimension of the differential equation governing the system. The differential equation that rules a one dimensional system is a one-dimensional ODE(Ordinary Differential Equation) of the form:

$$\dot{x} = \phi(x) \tag{3.1}$$

where ϕ(x) is a **vector field** which gives the evolution of each point in the phase space of the system. The trajectory of the system can be obtained by iterating over the rule and estimating the future states. This one dimensional system's behavior can be visualized using a phase line [1].

3.2.1 Flows as Dynamical Systems

Flows can be regarded as dynamical systems as they represent an evolution with time and can described by a rule over the phase space of the systems[61]. Flows can be described as **linear dynamical systems**, where the vector field is a function of the position in the phase space.
Flows are defined by two-dimensional autonomous ODEs, represented as:

$$\dot{x} = \phi(x) = Ax + b \tag{3.2}$$

where,
A is the characteristic matrix
x is the position vector

[1] A phase line shows the qualitative behaviour and the nature of the solution of an autonomous ODE of single dimension, $\frac{dx}{dy} = f(x)$ [64]

b is a vector where when b = 0,the equilibrium point, the point where $\dot{x} = 0$, is at the origin.
Such systems can be visualized on a **phase plane**.

3.2.2 Vector Fields

The trajectory of a dynamical system can be visualized by a geometric representation of the trajectories of the system over the phase plane. This representation of trajectories is essentially a **vector field** or in terms of dynamical systems, a **phase portrait** [60]. The phase plane yields us a visual display of the characteristics of the system under consideration. It has a two-dimensional phase space.

The vector field can be formed by plotting the set of differential equations governing the systems over the phase space. In the vector field, each point has a vector ascribed to it which describes the derivative of the point with respect to time. These vector fields arising from the linear dynamical systems of two dimensional ODEs usually fall into visually distinctive patterns. Such vector fields can be visualized as shown below:

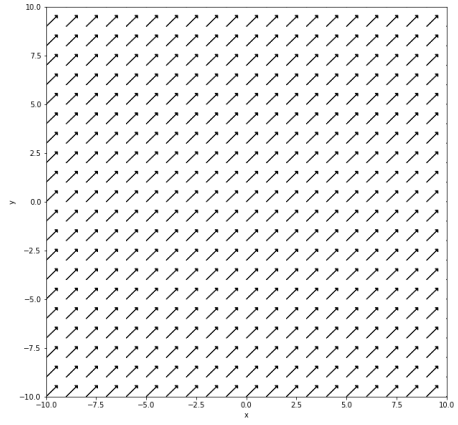


Figure 3.4: Constant Vector Field

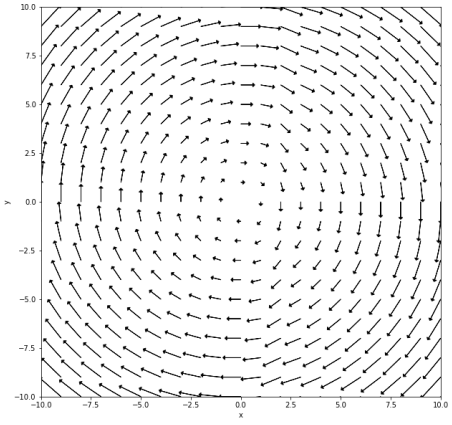


Figure 3.5: Circular Vector Field

Coming to the optical flow information, we can see that the output of the above discussed optical flow calculation methods yield us an optical flow field, which is essentially a vector field. Thus, we can intuitively see that, these fields can be represented in the form of linear dynamical systems and analysed further.

Phase Portraits

In the above section, we saw what typical vector fields look like. The vector fields generated by the differential equations of linear dynamical systems fall into six different types and are termed as phase portraits. These six types have specific names attributed to them. These phase portraits aid in giving visual information about the stability of the system under consideration. By locating the vectors of positions in the phase space, the possibility of the point in space reaching the equilibrium position can be known.

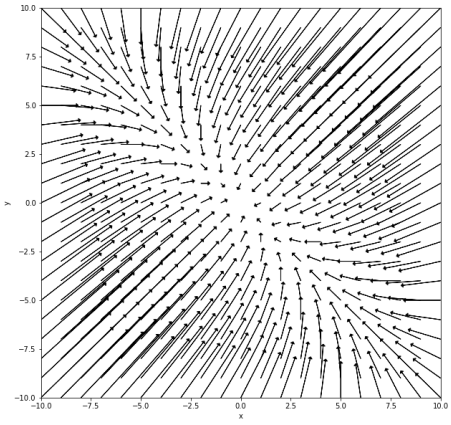


Figure 3.6: Node

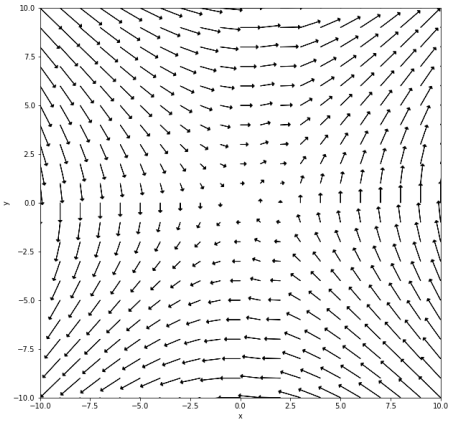


Figure 3.7: Saddle

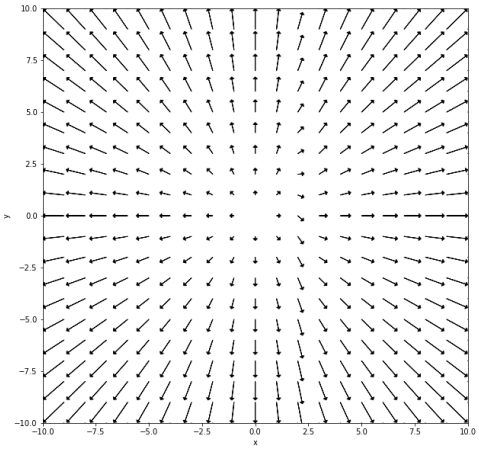


Figure 3.8: Star-node

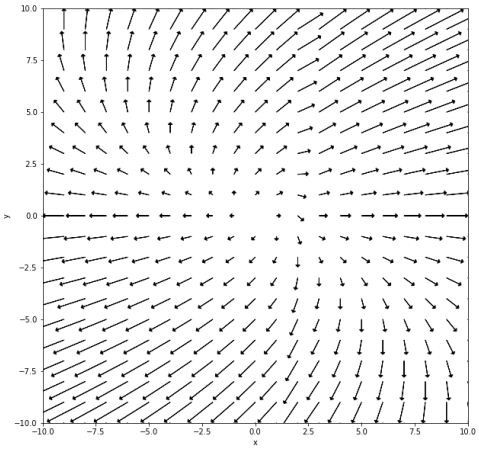


Figure 3.9: Improper-node

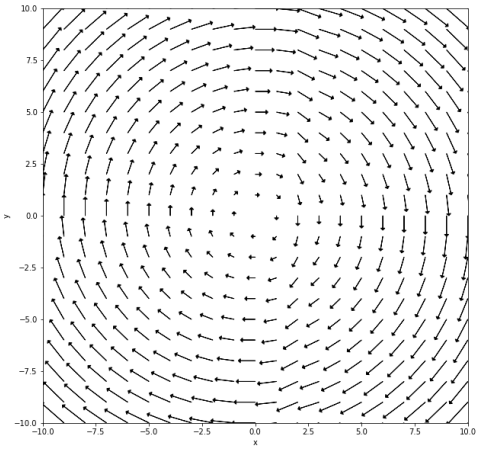


Figure 3.10: Center

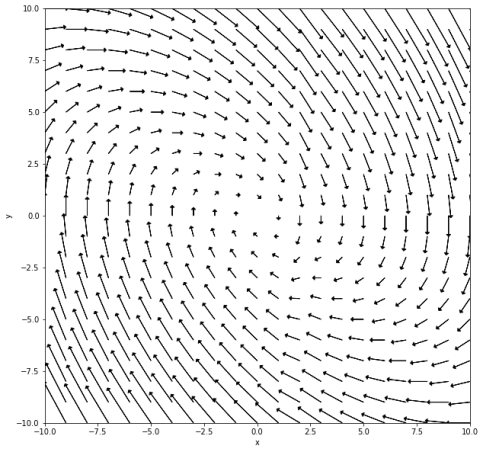


Figure 3.11: Spiral

The idea of our approach includes the classification of the optical flow fields into one of the phase portraits. As seen above, each of the phase portrait is distinctive. These phase portraits can be known by analysing the characteristic matrix of the system. By analysing the eigen values of the system, one can determine its phase portrait. By representing the optical flow field as a linear dynamical system and by analysing the characteristic matrix, the classification of the optical flow field can be obtained.

Optical Flow as Linear Dynamical System:

As discussed above, each of the vectors in the vector field have two components, which are the derivatives of its position (x,y) with respect to time. For a given time instance t:

The system equation is given by :

$$\dot{X} = A\bar{x} + B \tag{3.3}$$

where,

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{3.4}$$

$$\dot{x} = \frac{dx}{dt} = f(x, y) \tag{3.5}$$

$$\dot{y} = \frac{dy}{dt} = g(x, y) \tag{3.6}$$

The differential equations governing the system can be written of the form:

$$\dot{x} = ax + by + \epsilon \tag{3.7}$$

$$\dot{y} = cx + dy + \epsilon \tag{3.8}$$

where ϵ is the noise factor. When put in matrix notation, the system can be modeled as,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{3.9}$$

Now, from the system above, we can notice that the **characteristic matrix or companion matrix, A** of the system is given by,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{3.10}$$

and also B is given by,

$$B = \begin{bmatrix} e \\ f \end{bmatrix} \tag{3.11}$$

Analysis on the Characteristic matrix:

The phase portrait of the system can be identified by analysing the characteristic matrix A of the system. The solution to the system of equations can be considered to be,

$$X = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix} \tag{3.12}$$

The **Jacobian** of the systems of equations is,

$$J_X(x,y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} \tag{3.13}$$

In terms of jacobian, the system can also be defined by,

$$\dot{X} = J_X(x,y)\bar{x} + B \tag{3.14}$$

Thus, by comparing with the previous system equation we get that,

$$A = J_X(x,y) \tag{3.15}$$

From a dynamical system point of view, by analysing the **eigen values** of the jacobian of the system, the dynamical system can be classified based on its stability and the type of phase portrait it can be related to. Thus, we can see that by analysing the characteristic matrix of the optical flow field represented as a linear dynamical system, the classification of the type of flow field can be obtained.

3.3 Data Analysis Tools

In this section, we will discuss about certain data analysis methods which have been useful in carrying out the proposed approach. The motion data that is acquired by optical flow calculation gives yields us a three dimensional array with the flow values for each pixel. This data has to be analysed for various information from checking for pixels with similar flow values to fitting lines and equations to describe the flow and to analyse the fitted equations, etc. This section provides a brief discussion of all the techniques used for analyses. The requirements are as follows:

Identifying regions of similar flow: The motion anomaly detection algorithm needs to identify regions of similar flow orientations or directions and flow magnitudes, in order to derive patterns and its features out of the raw flow data that is given.

For this purpose, we use the k-means approach, which is discussed in the following section. This clustered data is also used for the line fitting process used to detect the critical point, the point of zero flow of the vector field. The line fitting process is also discussed in this section.

Characteristic matrix analysis: The motion anomaly detection algorithm also needs other tools to analyse the system matrices in order to provide a classification. In our case eigen values are used.

Data Clustering - K-Means

The k-means is a centroid based clustering technique for partitioning n-dimensional data into k sets which are especially useful for multivariate data [33]. The k-means clustering algorithm involves a couple of steps, the assignment step and an update step run over several iterations until non change is observed in either of the steps, indicating the best possible clustering achieved by using a centroid based approach. The algorithm begins by intializing k random centroids within the data. The algorithm as proposed in [33] and discussed in [3] is described below. The following equations are taken from [3]:

- Assignment step :
  In this step, each centroid is ascribed to each of the clusters. The clusters are formed by assigning the data points to its nearest centroid based on the euclidean distance(L_2).
  $$\underset{c_i \epsilon C}{\operatorname{argmin}}\, dist(c_i, x)^2 \tag{3.16}$$
  where c_i represents a centroid from set C and the data point x is assigned to the centroid with minimum euclidean distance.

- Centroid update:
  Let S_i represent the set of data points in the i^{th} cluster. The centroids are computed once again by taking the mean of all data points in each of the

clusters.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \epsilon S_i} x_i \tag{3.17}$$

The above two steps are iterated over until the distance between the points to the centroids in all the clusters are minimized or when the number of iterations have to be limited. For multivariate data, this is one of the best available technique to cluster data.

Drawback in K-means:

The algorithm described above will surely converge to give us the result but there exists a problem.

- The algorithm yields more accurate clustering for an optimal k value. The problem is to determine the inflection point at which the rate of change in the average distance within each of the clusters changes suddenly. By analysing this, the optimal k- value might be chosen. There are several algorithms to choose the optimal k [41].

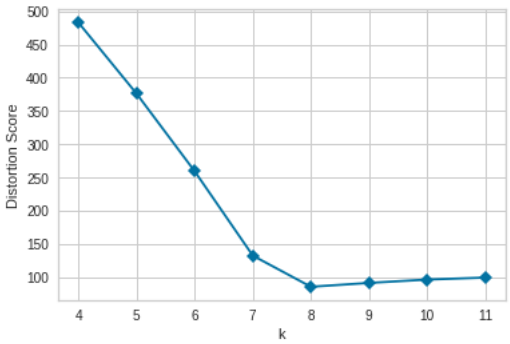


Figure 3.12: Visualizing the elbow for K-Means. The inflection point in the curve gives an optimal k value.

The above figure, shows the variation of the distortion score [65], the sum of euclidean distances of the points to centroids within each cluster. The inflection point shows the optimal k -value which can be chosen to yield the accurate clustering possible.

Line Fitting - Least Squares Regression

Least squares regression is a technique to find a best fit line for a given set of data [54]. Generally, the given set of data points contain noise which hinders the observer to find perfect linear relationship between the variables under consideration, say x and y. In order to find the linear relationship between the variables, we try find the best approximate line that can be fitted to the data. The following equations are taken and adapted from [54] and [18]:

- Let the given set of points be (x_i,y_i), where $i \epsilon 1...N$.
- The data is to desired to be fitted to a line of the form

$$y = mx + c \tag{3.18}$$

  or we can generalize it to any function as,

$$\bar{y} = A\bar{x} \tag{3.19}$$

  where $\bar{x}$ is the desired vector containing the solutions, m and c for example, A is the corresponding matrix with the x_i data or $f(x_i)$ and y contains all the y_i data.

- To find $\bar{x}$, we try to minimize the difference between the actual $\bar{y}$ values and the predicted values with $\bar{x}$.

$$\min \|\bar{y} - A\bar{x}\|^2 \tag{3.20}$$

  This is a typical problem in the case of **quadratic programming(QP)**[2]

[2] A quadratic programming problem is a mathematical optimization problem whose function to be minimized is a quadratic function. It is a type of non-linear programming. [38]

- Solving for $\bar{x}$:

$$\begin{aligned} f(\bar{x}) = \|\bar{y} - A\bar{x}\|^2 &= (\bar{y} - A\bar{x})^T.(\bar{y} - A\bar{x}) \\ &= (\bar{y}^T - \bar{x}^T A^T)^T.(\bar{y} - A\bar{x}) \\ f(\bar{x}) &= \bar{y}^T\bar{y} - 2\bar{x}^T A^T\bar{y} + \bar{x}^T A^T A\bar{x} \\ \implies \nabla_x f(x) &= -2A^T\bar{y} + 2A^T A\bar{x} \end{aligned} \tag{3.21}$$

To find the minimum, set its derivative to zero. that is,

$$\begin{aligned} \nabla_x f(x) &= 0 \\ \textbf{Solution :} \implies \quad \bar{x} &= (A^T A)^{-1} A^T \bar{y} \end{aligned} \tag{3.22}$$

Thus, the **closed-form** solution to line fitting is obtained as above.

Drawback in Least Squares Method: The solution can be obtained by solving the linear system of equations only as long as $A^T A$ is non-singular. The singularity occurs when all the x_i data points are equal [54].

Eigen Values and Eigen Vectors:

Eigen Values are special values associated with a linear system of equations, usually represented in its matrix form, and are also known as characteristic values. It is considered equivalent to matrix diagonalization and plays a crucial roles in characterizing a system. In dynamical systems point of view, eigen values of the characteristic matrix aids in stability analysis and classification of phase portraits. Each eigen value can be coupled with an eigen vector. This decomposition of a matrix into eigen values and eigen vectors is called eigen decomposition. This decomposition always exists as long as the matrix is a square matrix. This is the eigen decomposition theorem [58].

- Let A be a square matrix which is a linear transformation. The eigen value and eigen vector can be obtained if there is some λ and $X \epsilon \mathbb{R}^n \neq 0$ such that:

$$AX = \lambda X \tag{3.23}$$

- The eigen values are obtained by setting:

$$|A - \lambda I| = 0 \tag{3.24}$$

The obtained eigen values can be real numbers or complex numbers depending on the kind of transformation performed by the characteristic matrix.

Eigen values in Dynamical Systems: In case of dynamical systems, the signs of the eigen values indicate the behavior of the phase plane of the system [63]. Considering λ_1 and λ_2 as eigen values of the system:

- If sign(λ_1) $\neq$ sign(λ_2), the eigen vectors meet at a **saddle point**.
- If sign(λ_1) & sign(λ_2) = +ve, the eigen vectors meet at an unstable node and the system **diverges** away from that point.
- If sign(λ_1) & sign(λ_2) = -ve, th eigen vectors meet at a stable node and the system **converges** to the point.

Jordan Canonical Form

The jordan canonical form maps matrices representing linear transformation to a unique square matrix. It is useful in finding the similarity between matrices. The matrices with similar jordan form represent similar kind of transformations [49]. This is usually good for computations and description.

Relation with eigen values: As discussed in [44]. Let A be matrix representing any linear transformation:

- If A has real, distinct eigen values, the jordan matrix typically takes the form:

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{3.25}$$

- If A has complex eigen values, the jordan matrix typically takes the form:

$$\begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} \tag{3.26}$$

where, α and β are the real and imaginary parts of the complex eigen values.

3.4 Optimization Techniques

This section provides information about the optimization techniques used for solving the non-linear least squares problem. The two methods discussed below are the LM(Levenberg-Marquardt) algorithm and the CMA-ES(Covariance Matrix Adaptation - Evolutionary Strategy). The former is a widely used optimization algorithm which has been around for than five decades and the latter is based on evolutionary algorithms which was introduced in the past most recent decade. The requirement behind using these approaches is for:

Estimating the characteristic matrix: The characteristic matrix is estimated through a non-linear optimization process. This is the part where the raw optical flow data is tried to be approximated as a linear dynamical system. The two different techniques used are explained below in detail.

Non-Linear Least Squares:

The typical non-linear least squares is a least squares problem where the parameters enter the model in a non-linear manner. Usually it involves the minimization of a residual function as:

$$\min f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x) \tag{3.27}$$

where, $x = (x_1, ...x_n)$ is a vector and r_j is a non-linear function. In such cases, there is no **closed form** solution as in 3.22.

3.4.1 Levenberg-Marquardt(LM) Algorithm

The LM algorithm is one of the most widely used non-linear least squares optimization algorithm. It can be used for linear as well for non-linear functions. The main idea behind it is to approximate a given non-linear function as a quadratic around a region of evaluation and use **Lagrange multipliers** along with the gauss-newton approach, which is equivalent to combining the vanilla **gradient descent** and the **Gauss-Newton** approach to find the optimal values. The algorithm can be explained as below. The equations and discussions are taken and adapted from [43]:

- Let f from 3.27 be represented in terms of a residual vector r:$\mathbb{R}^n \rightarrow \mathbb{R}^m$, where m $\geq$ n and $r(x) = (r_1(), r_2()...r_m())$. Now we can rewrite f(x):

$$f(x) = \frac{1}{2}\|r(x)\|^2 \tag{3.28}$$

- In linear case, the derivatives of f can be written using the jacobian matrix:

$$J(x) = \frac{\partial r_j}{\partial x_i}, i \leq j \leq m, 1 \leq i \leq n. \tag{3.29}$$

$$f(x) = \frac{1}{2}\|Jx + r(0)\|^2 \tag{3.30}$$

- By setting ∇ f(x) =0, the linear closed form solution can be obtained by ,

$$x_{\min} = -(J^T J)^{-1} J^T r \tag{3.31}$$

  where, r(0)=r.

- But in the non-linear case, f(x) can not be written in terms of the Jacobian with r as a hyperplane in space.Thus, we have:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^T r(x) \tag{3.32}$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) \tag{3.33}$$

- With the least squares problem, it can be seen that $\nabla^2 f(x)$, the Hessian can be gotten for free with if the jacobian is known and if the $\nabla^2 r_j(x)$ or $r_j(x)$, the residual are small, which usually happens around the solution, and this can also be used if the function is assumed to be a quadratic around any region:

$$\nabla^2 f(x) = J(x)^T J(x) \tag{3.34}$$

- **Gradient Descent:** The simple vanilla gradient descent parameter updation is given by:

$$x_{i+1} = x_i - \lambda \nabla f \tag{3.35}$$

- **Gauss-Newton**The Gauss-Newton approach uses a Taylor series approximation of $\nabla f(x)$ around a current x_i, and neglects the higher order terms by assuming f as a quadratic function

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0) + higher \quad order \quad terms \quad of \quad (x - x_0) \tag{3.36}$$

- We can solve for the parameter update by solving for $\nabla f(x) = 0$

$$\implies x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{3.37}$$

- The above described gradient descent and Gauss Newton algorithms can now be blend to give us an hybrid update rule:

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i) \tag{3.38}$$

  where H is the Hessian, and λ is the lagrange multiplier which can be adapted. The update rule for λ:
  If function value reduces: Reduce λ by a factor,

If function value increases : Increase λ by a factor
Thus helping us move more in the direction where the slope is less and less in the direction when the slope is high.

- **LM update rule:** But the problem with the above update is, the Hessian H is ignored when λ is too high. This can be overcome by including the Hessian in the update as,

$$x_{i+1} = x_i - (H + \lambda.diag[H])^{-1} \nabla f(x_i) \tag{3.39}$$

This is the complete LM update rule, which overcomes the problem of optimizing non-linear functions by using a quadratic function approximation technique.

- **LM Implementation as a Trust Region Algorithm:** The above implementation was a typical line-search method [3]. The second approach to optimization is the trust region approach. In this approach, there's a model $m^{(k)}$ which approximate a function around the region $x^{(k)}$, which is denoted by Δ where the assumed approximation yields expected results as the actual function. This region of confidence, Δ is called the **trust-region**. This approach updates Δ at each iteration similar to the line-search method which updates λ

- The trust region model function is given as the quadratic approximation of f(x) as:

$$m^{(k)} = f(x^{(k)}) + \nabla f(x^{(k)}).p + \frac{1}{2} p^T H p \tag{3.40}$$

where x_k is the current x, H is the approximated Hessian, and p is the step-size which yields the parameter update:

$$x^{(k+1)} = x^{(k)} + p \tag{3.41}$$

- The parameter update p can be solved by minimizing the trust region function as below. There are also certain conditions in which we can judge whether p is

[3] It is an iterative optimization strategy where the parameter update is done by deciding on the direction of descent and then fixing a step-size [43]

global minimum.

$$\min_{\|p\| \leq \Delta} f(x^{(k)}) + \nabla f(x^{(k)}).p + \frac{1}{2} p^T H p \tag{3.42}$$

- The trust region Δ is updated depending on whether the approximated function closely matches with the actual function, which can be given by the ratio:

$$\rho_k = \frac{f(w^{(k)}) - f(w^{(k)} + p^*)}{f(w^{(k)}) - m^{(k)}(p^*)} \tag{3.43}$$

  Δ Update rule:
  If $\rho_k \approx 1$ and above a certain threshold x: increase Δ,
  If $\rho_k \leq x$: decrease Δ

- Thus, the LM algorithm can be implemented as a trust region approach, where the main difference is that the adaptation is carried out on the parameter Δ and not on the parameter λ as in the line-search approach.

Drawback in LM:

Quadratic Approximation: As discussed before, the LM algorithm works by using a quadratic approximation of the residual function around a region and hence assumes that the $\nabla^2 r_j(x)$ is very less. This assumption does not help in cases when the residual $r_j(x)$ is large, and the Hessian cannot be approximated as done in this method in 3.34 [43]. Thus, the performance degrades accordingly.

Initial Point Sensitivity : The drawback above is also one of the main reason why an **initial point** x_0, used for initlaizing the optimization process might fail if it falls within such a region as described above. Overall, the LM algorithm is sensitive to the starting location of the optimization process and might get stuck in a local minima. This might also result in **slow convergence**.

3.4.2 CMA-ES Algorithm

In this section, we discuss about a state-of-art algorithm for non-linear and non-convex optimization based on evolutionary strategies.
Evolutionary algorithms, as discussed in [5] can be seen implementing two major approaches:

- Genetic Algorithms
- Evolutionary Strategies

The CMA-ES, Covariance Matrix Adaptation- Evolutionary Strategy, as the name suggests is an algorithm based on evolutionary strategy and falls into the second category of evolutionary algorithms.

Evolutionary Algorithms - Methodology

In terms of optimization problems, the general steps involved in evolutionary algorithms include [5][59]:

1. **Population Initialization**: To generate set of random solutions(solution vectors represented as genomes).
2. **Fitness Test:** The population set is tested for fitness on the objective function.
3. **Selection:** To select the best individuals(solutions) with higher fitness values for reproduction.
4. **Genetic operations:** Typical reproduction operations such as **Crossover** and **Mutation** are performed on the selected individuals to yield offspring.
5. **Generate New population:** Evaluate the offspring and replace the lesser-fit individuals in the old population with offspring and iterate over from third step until the best solution is found .

Evolutionary Strategy(ES)

Given an objective function to minimize, evolutionary strategies operate by generating a set of solutions from a normal distribution with mean μ and standard deviation σ [40] .
If the solution vector is given by $(x_1, x_2...x_n)$, the mean and standard deviation vectors would be $(\mu_1, \mu_2...\mu_n)$ and $(\sigma_1, \sigma_2...\sigma_n)$ respectively.

- Let's assume a 2D optimization problem, the respective vectors would be,

$$\mu = (\mu_x, \mu_y)$$

$$\sigma = (\sigma_x, \sigma_y)$$

- Initial μ is set at the origin and a new population is generated.
- Evaluate the fitness and choose the best solution.
- The chosen best solution is selected to be the new μ and a new population is generated around the best solution by adding noise to the best solution based on σ.
- The above two steps are repeated until best solution is achieved.
- In some approaches, the σ values can be adapted based on the fitness value of the generated best solution.

CMA-ES

As discussed above, while the simple case of evolutionary strategy only deals with the μ and σ parameters of each dimension separately, the CMA-ES approach extends the parameter space to the covariance matrix [40]. It adapts the mean μ_x, sigma σ_x as well as the covariance components σ_{xy}, basically the mean and the covariance matrix which is used generate new populations during the process. The procedure for adapting is explained below. The equations and discussions are taken and adapted from [40], [22] and [21].
Motivation: The basic idea with generating populations with individual solutions

using normal distribution is equivalent to setting a search space and exploring it. When the predicted solution is far away from the desired solution, the search space is increased and when the desired solution is close by, the search space is decreased. This is achieved by adapting the covariance matrix used to generate solution populations.

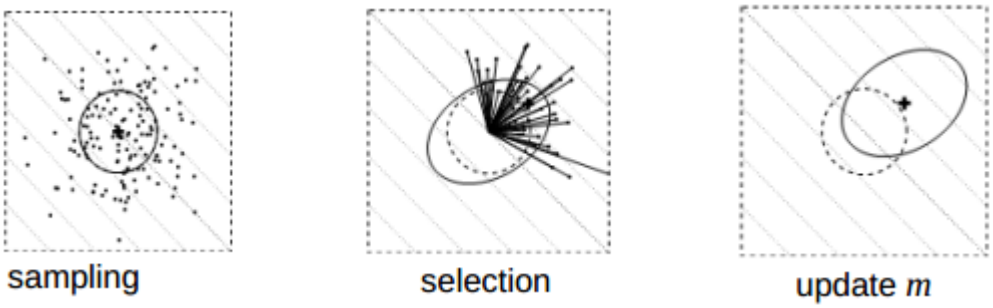


Figure 3.13: CMA-ES μ updation [21]. The mean, m(μ) updation by picking the best individuals closer to the solution, where the solution is achieved by moving towards the upper right corner.

- **Population generation:** The population is initialized and respective mean values are calculated with N members as follows:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.44}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.45}$$

- **Initial Covariance Matrix:** The initial covariance matrix components are calculated as :

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2 \tag{3.46}$$

$$\sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_y)^2 \tag{3.47}$$

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{3.48}$$

and the matrix is given by,

$$\Sigma_{xy} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \tag{3.49}$$

- **Selection and Mean(μ) updation:** A fraction of individuals, N_{best} which are deemed among the fittest are selected and the new mean of the population is calculated among the N_{best} members of the selected population . This is also shown in 3.13.

$$\mu_x^{(t+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} x_i \tag{3.50}$$

$$\mu_y^{(t+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} y_i \tag{3.51}$$

- **Covariance Matrix Adaptation:** As seen in the previous step, the mean was updated as $\mu^{(t+1)}$, at the (t+1)th step with the selected (t+1)th population ,but in this step, the algorithm updates the covariance matrix based on th new population at (t+1) but uses the mean, μ^t from the previous population at (t). This brings in the information of how close or far away the predicted best solution is from the desired solution.

$$\sigma^{2,(t+1)}{}_x = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^t)^2 \tag{3.52}$$

$$\sigma^{2,(t+1)}{}_y = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (y_i - \mu_y^t)^2 \tag{3.53}$$

$$\sigma^{(t+1)}{}_{xy} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^t)(y_i - \mu_y^t) \tag{3.54}$$

Thus CMA-ES implements an evolution strategy, where the mean, variances as well as the co-variances are updated. By this way, the population iteratively converges towards the best possible solutions.

Drawback in CMA-ES:

Memory Efficiency: As with other evolutionary algorithms, the generation of populations and other operations take a toll on the system memory. Thus, compared to other numerical optimization methods, it is not memory efficient [5].

Similarity to LM:

Trust Region(LM) and Search Space(CMA-ES) : In LM, λ, the Lagrange multiplier or Δ, the trust region is increased or decreased based on how close or further the solution is. In the same manner, the covariance matrix Σ_{xy} is altered in order to increase or decrease the search space in the CMA-ES approach.

Advantage over LM:

- **Initial Point Sensitivity:** The CMA-ES is not sensitive to the initial point for the optimization process as LM, even though CMA-ES does not always provide the global minima. The algorithm converges almost at the same rate irrespective of the initial point. But, the time taken almost increases exponentially for LM based on the dimension of the problem and the initial point. This makes CMA-ES a better option when **faster convergence** is required when a good inital point assumption is not possible.
- **Gradient Calculation:** CMA-ES is a **gradient-free optimization** method.
- **Assumptions: No assumptions**,(works for non-convex, non-smooth functions, etc.[21]) or function approximations involved as in LM, which makes assumptions that fail under certain conditions.

4

Motion Anomaly Detection Algorithm

In this chapter, the motion anomaly detection algorithm is elaborated in detail in a step-by-step manner explained under different sections. The algorithm behind both the linear and non-linear implementations are discussed and are compared between each other. The motion anomaly detection algorithm steps includes the representation of data, making a qualitative analysis with the linear and non-linear approaches, estimating critical points and detecting motion anomalies with the help of certain parameters involved in the qualitative analysis process. These steps are explained individually in the underlying sections.

Firstly, we shall discuss the steps involved in the classification of images based on the optical flow vector fields in the upcoming section. This classification, the qualitative analysis part is the core and base for the upcoming anomaly detection part where the classification is made making use of also the other parameters involved in this process.

4.1 Phase Portrait Classification- Qualitative Analysis

In this section, we shall discuss about the steps involved in the qualitative analysis of the optical flow field. The aim of this section is to map a given optical flow vector field to a particular type of phase portrait discussed in the previous chapter. The first step involved is to obtain raw data and process it i a manner that the classification is possible.

Raw Optical Flow Data:

Owing to the fact that this algorithm is to be tested on autonomous systems working in real-time dynamic environments, handling noise is a must. The noise is mainly due to:

- **Vision sensors** used to capture the movements are prone to inducing noise in the data.
- The optical flow estimation algorithm that is used is based on a **two-frame motion estimation** method. It fails to take advantage of the spatio-temporal consistency, in simple terms the average image over an extended time, to yield highly accurate results.[16]

Dominant Orientation Field

The raw optical flow field is highly noisy and needs some smoothening operation. This can be done by calculating the dominant flow in each region and building a new image with only the dominant orientation in each location. The two steps involved are:

- Scan a window of fixed size over the image and obtain a dominant orientation at each location.
- Build a new image from these dominant orientations, thus reducing the size of the image which aids in computation as well as representation.

As done in [16], the final orientation field is obtained by calculating the dominant orientation and the magnitude at the centre (m,n) of the window of size (NxN). The following equations are taken from [44]:

$$\hat{\theta} = \tan^{-1}\left(\frac{\sum_{i=m-\frac{N}{2}}^{i=m+\frac{N}{2}}\sum_{j=n-\frac{N}{2}}^{j=n+\frac{N}{2}} G_{ij}^2 \sin 2\theta_{ij}}{\sum_{i=m-\frac{N}{2}}^{i=m+\frac{N}{2}}\sum_{j=n-\frac{N}{2}}^{j=n+\frac{N}{2}} G_{ij}^2 \cos 2\theta_{ij}}\right) \tag{4.1}$$

where G_{ij} is from the polar representation of the flow field gradient vector as $G_{ij}e^{i\theta_{ij}}$, where G is the vector magnitude and θ is the direction. The effect of this averaging

operation can be seen in fig.4.2 and fig.4.3.
Similarly, the dominant flow magnitude across the image is obtained as:

$$\rho = G_{mn}\left[\frac{\sum_{(i,j)\epsilon W} \|G_{ij}\cos(\theta_{ij} - \hat{\theta}_{mn})\|}{\sum_{(i,j)\epsilon W} G_{ij}}\right] \tag{4.2}$$

where, W stands for the window of size NxN that is panned across the image and ρ is the dominant magnitude at the centre (m,n) of the window.

Figure 4.1: A typical noisy optical flow field superimposed on the image sequence [17]. Usually, unevenly shaped objects exhibit noisy flow, as seen in the image, the trees, poles and their shadows add more noise than others.

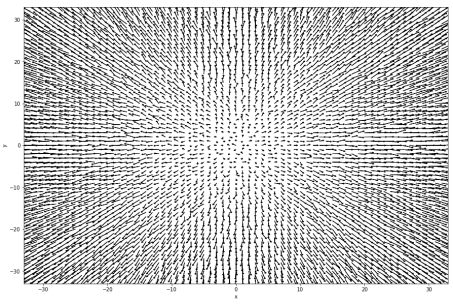


Figure 4.2: Typical noisy optical flow field

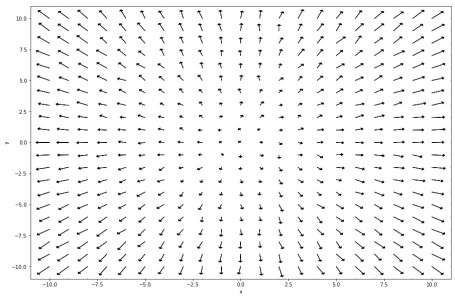


Figure 4.3: Smoothened optical flow field with computed dominant orientations and magnitudes

The figures, fig.4.2 and fig.4.3 show that the noisy data has been smoothened out and the size of the image is reduced as well, noted by number of gradient vectors, one per each pixel present in the image.

Flow classification by region

The smoothened image is then subject to segmenting based on flow. This segmentation is done based on the flow type belonging to one of the phase portraits. For this purpose, we use a window that we pan across the image and classify each section of the image based on flow type. A window of size NxN is scanned across the image with varying step size as per the application. This finally results in a segmented image which.

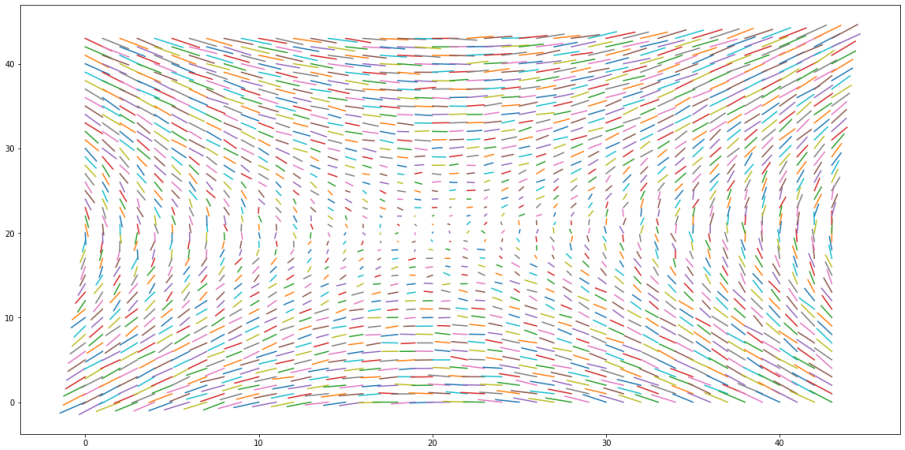


Figure 4.4: Optical flow orientation field to be segmented based on flow type.

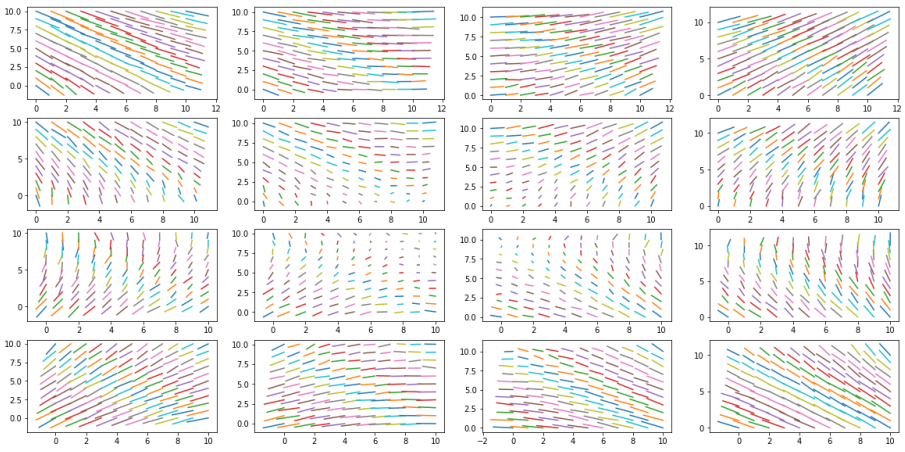
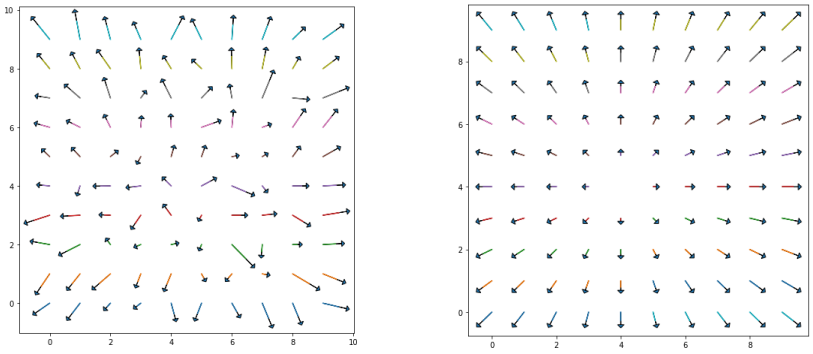
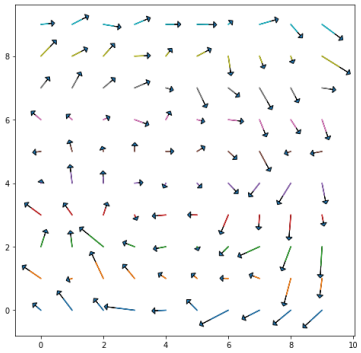


Figure 4.5: Breaking the orientation field window-by-window for classification and segmentation.

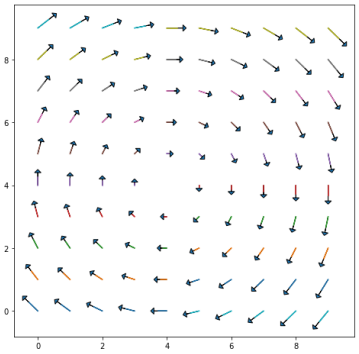


(a) Dominant flow field.

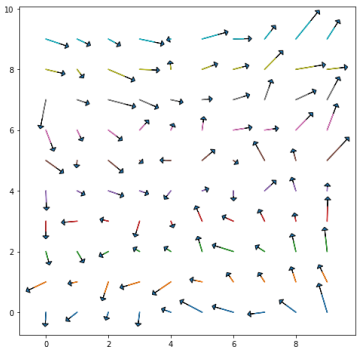
(b) Star-Node phase portrait



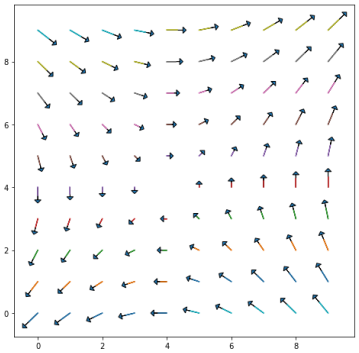
(c) Dominant flow field



(d) Centre phase portrait



(e) Dominant flow field



(f) Saddle phase portrait

Figure 4.6: Mapping flow fields in each window to phase portraits. The dominant field images on the left are mapped to their respective phase portraits in the right column of this figure.

The above sectioned windows in fig.4.5 are then run through a phase portrait classifier algorithm, and are mapped to a particular type of phase portrait as shown in fig.4.6. Basically, the optical flow field is approximated to a linear dynamical system whose vector field best matches the optical flow field. Depending on the number of windows per flow field, which varies depending on the window size and image size, an equal number of classifications are done. Following the phase portrait classification, the critical point of each phase portrait is estimated and further other parameters are analysed and utilized in the motion anomaly detection process.

This mapping is achieved by using the algorithms described in the underlying sections. There are two algorithms that are described below. The non-linear algorithm was proposed in [44] and the linear algorithm is an improvised version from [53].

4.2 Non-Linear Algorithm

As proposed in [44] , the vector fields can be assumed as oriented textures and there is a method to analyse these oriented textures by mapping them to phase portraits. The algorithm makes use of the fact that only a finite number of qualitatively different phase portraits arise for linear dynamical systems [45].
The first challenge in mapping the fields is to model the flow field as a linear dynamical systems. To model the flow field, the parameters of the linear dynamical systems have to be estimated. The parameters of the system are as discussed in the previous chapter, from equations 3.3 to 3.9 are given by:

- The linear dynamical system equation:

$$\dot{X} = A\bar{x} + B \tag{4.3}$$

- The parameters to be estimated are all the unknowns in the below equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{4.4}$$

- There are in total six parameters, those that form the A and B matrices, to be estimated.

- The number of parameters is crucial because of the non-linear approach to optimization. The more the number of parameters, the more the complexity and time taken for estimation.

Approximating Flow Field to a Linear System - Methodology:

The main objective is to device an approach to approximate the given field to a linear system. As proposed in [44], this is done by using a measure of difference in the orientation between two vector fields, one which is the obtained flow field and the other is the predicted or generated flow field. The approach is as follows:

- Consider two oriented textures as shown in the image below:

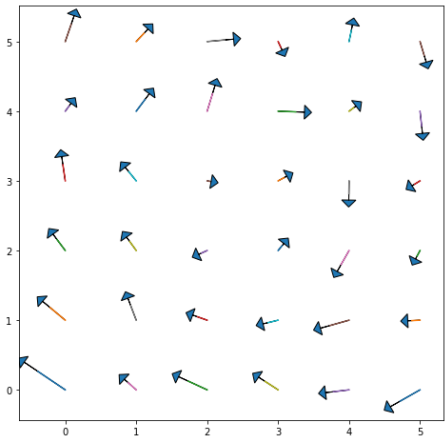


Figure 4.7: Flow field to be mapped

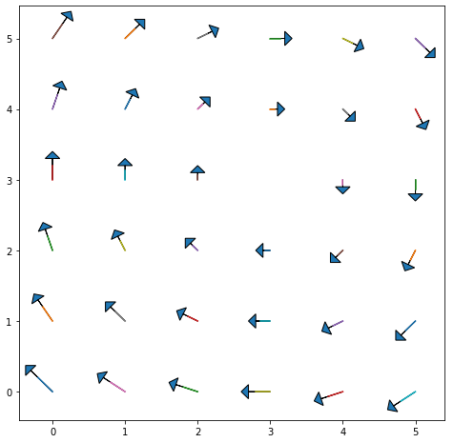


Figure 4.8: Predicted flow field by approximation

- The closeness or similarity between the two fields is measured by calculating the area of the triangle formed by two vectors at identical positions (i,j) in the two fields. the two vectors from the same location are chosen as shown in fig.4.9 and 4.10. The following equations inn this section are taken and adapted from [44].

- The difference in the orientation between the vectors is obtained as the area of the triangle formed by both these vector. This area will serve as a measure of difference and thus optimizing for this would yield us an optimal estimate

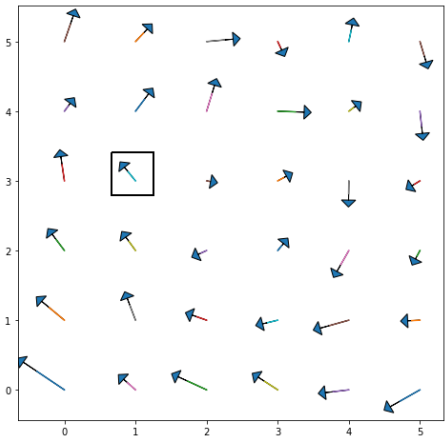


Figure 4.9: Vector chosen from flow field to be mapped

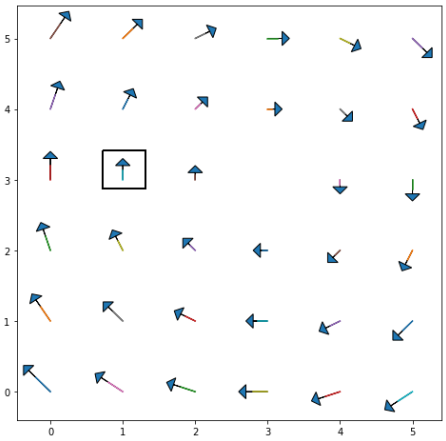


Figure 4.10: Vector chosen from same location in Predicted flow field

of the matrix values.The area between the vectors as area of triangle is given by:

$$A_{ij} = \frac{1}{2} R_{1_{ij}} R_{2_{ij}} |sin(\theta_{1_{ij}} - \theta_{2_{ij}})| \tag{4.5}$$

where, (i,j) is the location of the vector in the vector field,
$R_{1_{ij}}, R_{2_{ij}}$ are the length of the vectors in the vector field 1 and 2,
$\theta_{1_{ij}}, \theta_{2_{ij}}$ are the orientation angles of the vectors.
It can be noted that this measure of difference is independent of the sign of the directions on the vectors. Thus, instead of vector fields, one can just view them as orientation fields without the arrow indicating the direction as in fig 4.4 and fig.4.5.

- **Objective Function:** The obtained areas are summed up over the entire image as:

$$S = \sum_{i,j \epsilon F} A_{ij} \tag{4.6}$$

where, F is the area over which the field exists. The sum of areas over the entire field is the objective function to be optimized in this algorithm. As one can see, reducing the sum of areas would result in the reduced difference between the individual vectors in the vector field.

- **Optimization Problem:** The typical optimization problem can be described

as given in [56]:

$$\bar{x}^* = \arg\min_{\bar{x}\epsilon V} f(\bar{x}) \tag{4.7}$$

where V is the solution set from which we find the optimal solution $\bar{x}^*$.

- The optimization problem in this case will require us to generate the predicted flow field, which in turn requires the variables a,b,c,d,e and f from matrices A and B. The function to be minimized is the sum of all the areas function.

$$f(\bar{x}) = S(\bar{x}) \tag{4.8}$$

$$\bar{x} = [a, b, c, d, e, f] \tag{4.9}$$

$$\bar{x}^* = \arg\min_{x\epsilon V} S(\bar{x}) \tag{4.10}$$

- **Optimization technique:** As the parameters above enter the objective function in a non-linear manner, the above optimization problem can be solved using two methods as a **non-linear least squares fitting** problem:

  - Levenberg-Marquardt(LM) Algorithm
  - CMA-ES (Covariance Matrix Adaptation- Evolution Strategy)

  The former approach, LM was suggested for this problem and tested out in [44].

Critical-Point Estimation:

The critical point or fixed point for the flow field and phase portrait is described as the position of zero flow. This position of zero flow is the position to which the flow converges or diverges from, in the phase portrait. This is calculated by setting the flow values to:

$$A\bar{x} + B = 0 \tag{4.11}$$

$$\bar{x_0} = -A^{-1}B \tag{4.12}$$

where, $\bar{x_0}$ is the **critical point**.

LM vs. CMA-ES:

- The LM approach performs equally well or sometimes better than CMA-ES when the critical point is within the flow field.

- As the critical point is shifted away from the flow field, as in typical application scenarios, the time taken taken to predict for LM and the misclassification rate stands poor when compared to CMA-ES.

- An example failure case -The prediction results for a **Saddle** node with critical point at (-30,-30) generated with :

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.13}$$

LM Algorithm		
Window Size	Classification	Time Taken(s)
4 x 4	Node	4.7
8 x 8	Node	53
16 x 16	Star	228

Table 4.1: Prediction - LM

CMA-ES		
Window Size	Classification	Time Taken(s)
4 x 4	Saddle	4.2
8 x 8	Saddle	9.7
16 x 16	Saddle	34

Table 4.2: Prediction - CMA-ES

The LM Algorithm fails to classify correctly every single time irrespective of the window size and in spite of taking much longer than the CMA-ES algorithm. This is the case for other phase portraits as well.

- For the implementation of the non-linear algorithm, CMA-ES is clearly a better choice.

4.3 Linear Algorithm

The previous section explained the non-linear approach to the phase portrait classification problem. The biggest problem with the non-linear algorithm presented above is the **computational time** efficiency. The linear algorithms are preferred in cases where the computational capability of the systems is low, as in most embedded devices and also when there is a need for real-time capability.

In the linear algorithm, the problem is solved with a different approach. This approach separates the critical point estimation problem completely from the parameter estimation and classification problem. After the critical points have been estimated, the critical point informations is used to estimate the parameters in a more robust and intuitive manner than compared to the previous approach, which is more of a brute force search methodology.

4.3.1 Critical Point Estimation

The critical point estimation problem is tackled by making use of the orientation data. The obtained orientation data is made use of directly without any additional transformations or assumptions. The problem is mainly solved by a clustering algorithm. This approach was introduced in [53]. The algorithm can be explained as follows:

- Cluster the orientation angles in the flow field.
- Fit lines to the points clustered by the orientation angle data.
- Identify critical points the closest point to all the lines, in an ideal flow field without noise, it is the intersection of all the fitted lines.

Clustering - Orientation Angles

The pattern of arrangement of the the orientation angles among the phase space positions is taken advantage of in this method. The clustering process can be done by using the k-means algorithm as described in the previous section. The number

of clusters is set to an optimal value depending on the size of the window that is scanned across the flow image for classification.
The clustered orientation angles are shown in the figure 4.11. There is a clear pattern that is visible from the clustering that can be noticed. All the clusters seem to converge or diverge towards or outwards from a point in the phase plane. This point when compared against the actual phase portrait seems to be the critical point of the system.

Fitting Lines:

As discussed in the above section and as seen in the figure 4.11, points with similar orientation angles seem to lie along a space that is converging towards a certain point with its area growing outwards from it. This can be taken advantage of and the critical point of the system may be found by fitting a line along each of the clusters. In turn, the critical point of the system can be found by solving for the lines (i.e point of intersection of the lines). In an ideal case where the system is free of noise, this would be possible.
The line fitting process has to be more robust to noise, especially when considering optical flow fields. The lines therefore should be fit by robust regression techniques. This can done by:

- Fit line by linear **least squares regression** as discussed in the previous chapter.
- Reject lines which have high average residual error(standard deviation of the fitting process) in the fitting process, the lines for which a confident fit is not obtainable due to noise.
- Also, as suggested in [53], we can use **least median squares regression** [48], and then reject lines with high standard deviation.

The lines fit by the least squares regression technique can be viewed as in figure 4.12. We can see that the lines extend beyond the phase space, thus indicating we can identify critical points with zero flow which are within and beyond the flow field. Also, all the lines do not perfectly intersect at he same point.

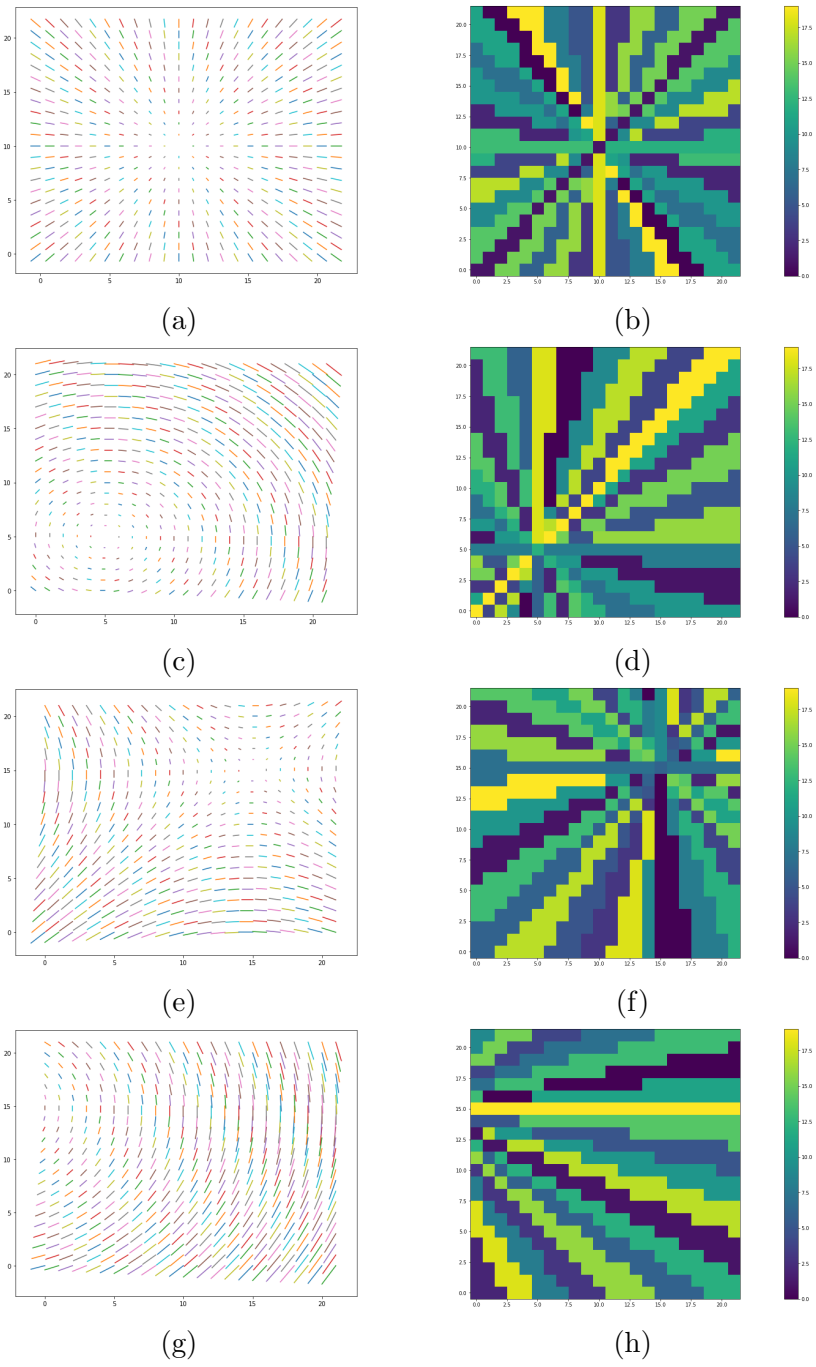


Figure 4.11: **Clustering:** Visualizing the flow field by clustering on orientation angle data. Positions with similar orientation angles are clustered and shown with similar color codes

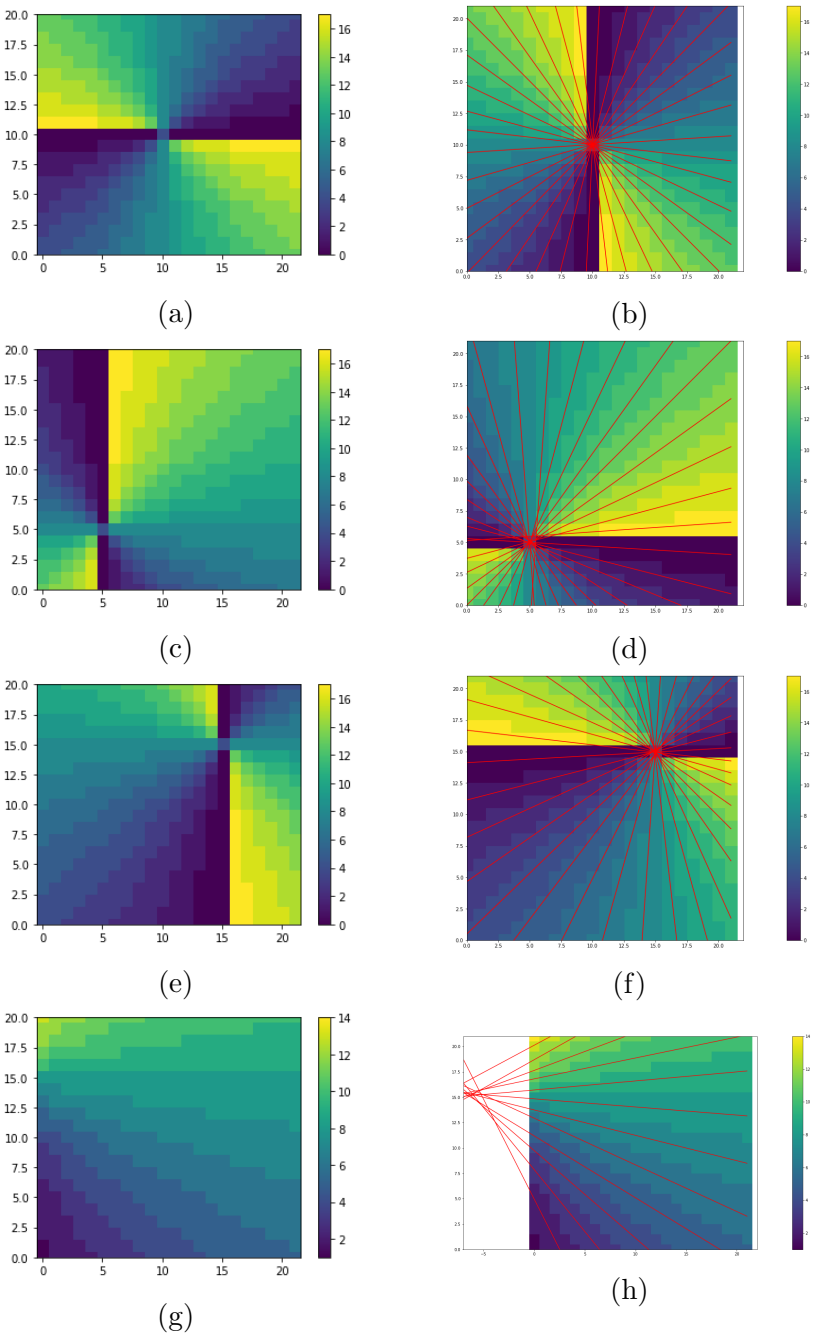


Figure 4.12: **Line fitting:** Visualizing the lines fit on the flow field clustered on orientation angle data. It can be noticed that all the lines do not intersect at the same point in space and also the point of intersection is beyond the observed flow field as in (h).

Identifying Critical Points:

As discussed above, the critical points can be found by solving for the intersection of all the lines fitted on the clustered data. The main hindrance in this straight forward approach is usually the noise involved. The noise in the orientation angles cause slight deviations in the fitted lines, thus preventing all the lines from meeting at the same point.
To solve this problem, we can solve by finding the critical point as the point that is closest to all the lines. This would be the best possible estimate in such a situation show in figure 4.13.

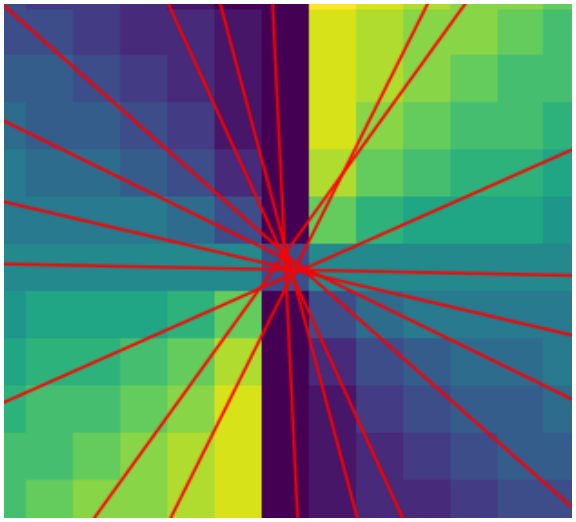


Figure 4.13: Visualization of the intersection of all the fitted lines. It can be noticed that all the lines do not intersect at the same point.

As discussed earlier, the point that is closest to all the lines can be found by optimizing a function that calculates the distance from a point to the line. Thus, we can the model the systems as follows. The equations are taken from [53]:

- Let the equation of a fitted line be as follows:

$$\alpha_i x + \beta_i y + \gamma = 0 \tag{4.14}$$

- The shortest distance from a point to a line L_i is given by the below equation.

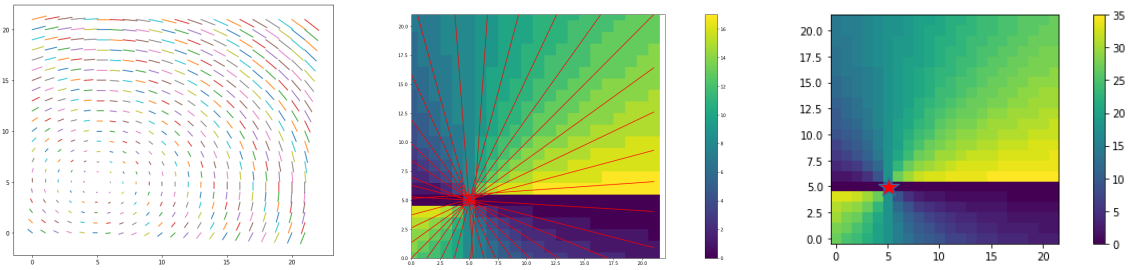
Let the point be (x,y):

$$\left(\frac{\alpha_i x + \beta_i y + \gamma_i}{\sqrt{\alpha_i^2 + \beta_i^2}}\right)^2 \tag{4.15}$$

where α_i, β_i and γ_i are the parameters describing the line L_i.

Thus, by optimizing the median of the above function, the critical point can be estimated robustly.

The overall process can be visualized as given if figure 4.14. This shows the critical point estimation for a center flow field.



(a) Orientation Field (b) Fitted Lines (c) Critical point

Figure 4.14: The flow of the critical point estimation process can be seen above. given the orientation field, the data is clustered, lines fitted and the estimated critical point.

4.3.2 Characteristic Matrix Estimation

The key difference between the earlier proposed approaches in [53] and [44] is that the phase portrait detection makes use of the critical point detection in order to make the algorithm linear and robust. In the above section, the critical point was detected in a linear approach. The critical point can lie within the visible section of the flow field or beyond it. The critical point is made use of by analysing the linear system as an affine transformation.
Considering the linear system to be an affine transformation [34], the system can be described as follows:

- **Affine Transformation:** The model for affine transformations in $\mathbb{R}^2$ can be described as:

$$\dot{X} = A\bar{x} + B \tag{4.16}$$

  To make it more clear, we shall see what each component does to the system.

- **Linear Transformation:** It can be defined as:

$$\dot{X} = A\bar{x} \tag{4.17}$$

  which brings about rotation,scaling, reflection, etc. in the $\mathbb{R}^2$ space.

- **Translation:** A translation can be gotten by just summing up a vector such as:

$$\dot{X} = \bar{x} + B \tag{4.18}$$

  which translates every point including the zero point(origin) in the $\mathbb{R}^2$ space.

- **Difference between linear transformation and translation:** The main difference between the transformation that is key to this approach is the **preservation of origin**, the zero point in the linear dynamical system.

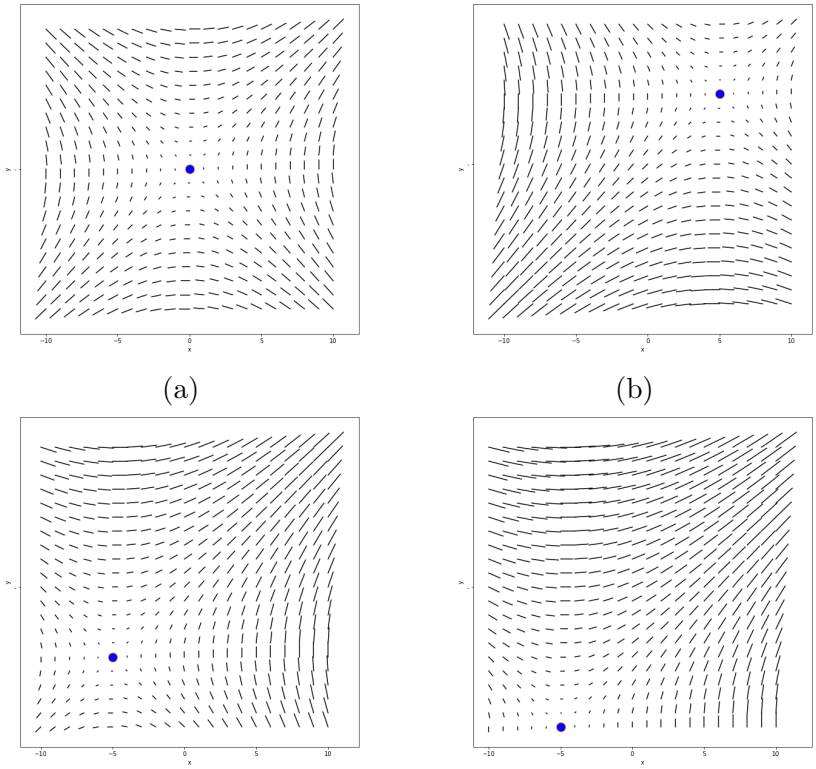
Thus, we can see that the affine transformation is a combination of :

$$\dot{X} = Linear \quad transformation(\bar{x}) \quad + \quad Translation(\bar{x}) \tag{4.19}$$

On comparison with eq. 4.16, we see that B is a purely translational component.

Approximating to Linear Transformation:

By identifying the origin or critical point,the affine transformation can be approximated to a linear transformation. This approximation will thus help in solving the problem in a linear approach. The problem once approximated to a linear transformation can be solved by the minimizing the residual error, as in the normal **linear least squares** regression problem.
The purely translational effects of the B matrix can be seen as shown in fig. 4.15



(a) (b)

(c) (d)

Figure 4.15: **Effect of Translation Matrix:** The flow fields above have been created with the same linear transformation part(A) but varying B, the translation part. It can be seen that the critical point has been translated along the field in proportion to the B matrix translation values.

The flow fields above have been created using the following values for the A and B matrices.
The same A matrix was used for all the saddle flow fields:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The B matrix was varied as:
B =[0,0], B= [-5,-5], B= [5,5] and B = [10,5] for the sub figures (a),(b),(c) and (d) respectively in figure 4.15.

Interpretation of Moving Critical Point: The moving critical point suggests that the fixed point of the affine transformation is also moving. The movement of the fixed point is due to the translation effect on all the (x,y) values in $\mathbb{R}^2$. The **origin** of the field has been shifted. The key insights can be listed as follows:

- As seen from 4.15(a), the corresponding **B** (translation) matrix is **[0,0]**. This is true in all the cases, all different types of phase portraits.
- When the origin, (x,y) = (0,0) in the phase space, the $\mathbb{R}^2$ matches with $(\dot{x}, \dot{y})$ = (0,0), the critical point of the flow field, then the B matrix can be ignored.
- From the above observation, we can approximate the affine transformation to a linear transformation when the **co-ordinate system** has been **shifted** to match with the critical point of the flow field.

$$\dot{X} = A\bar{x} + B \tag{4.20}$$

When critical point and origin coincide, we can ignore the translation component:

$$\implies B = 0 \tag{4.21}$$

Which makes it a purely **linear transformation**:

$$\implies \dot{X} = A\bar{x} \tag{4.22}$$

Solving for Characteristic matrix

The approximated problem can now be solved by using the below technique:

- Thus, the problem can be solved linearly by using the information at hand by:

$$Optical \quad Flow \quad Values \implies \dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{4.23}$$

$$Shifted \quad co-ordinate \quad system \implies \bar{x} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.24}$$

- The system can be solved for A by solving by **linear least squares regression** method:

$$\|\dot{X} - A\bar{x}\|^2 = 0 \tag{4.25}$$

4.3.3 Phase Portrait Classification

As discussed in the previous chapter, eigenvalues and Jordan canonical forms are used to classify the flow fields as phase portraits. As discussed in [44], the classification that is followed here is as shown in figure 4.17. Additionally, in the case of only distinct eigen values, the classification can also be visualized as shown in figure 4.16. This is based on the quadratic formula that arises when solving for the characteristic polynomial in calculating eigen values as in [63]:

$$\lambda = \frac{1}{2}(p \pm \sqrt{\Delta}) \tag{4.26}$$

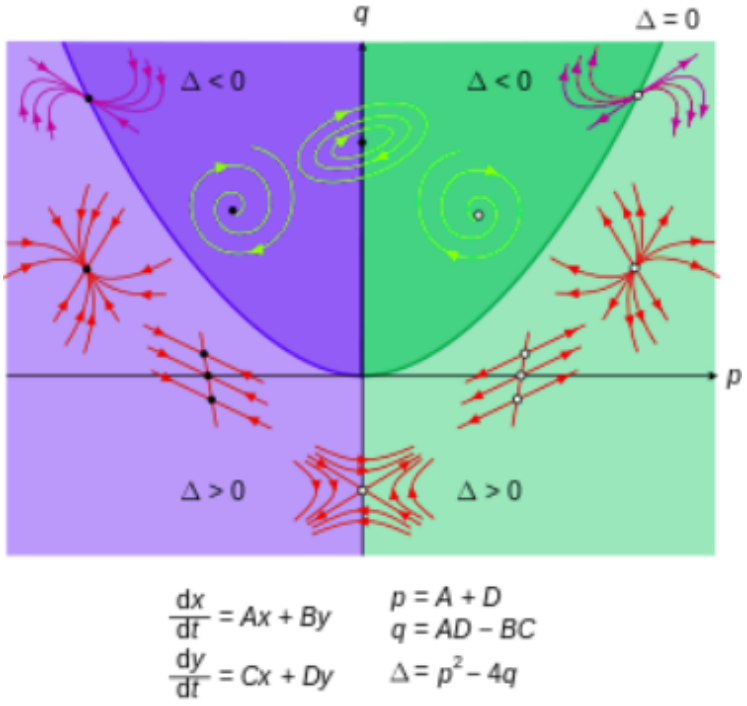


Figure 4.16: Classification of phase portraits in the case of distinct eigenvalues. [63]

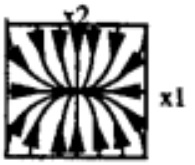
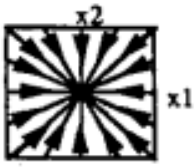
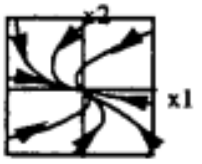
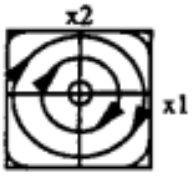
CASE	JORDAN FORM	TYPE OF PHASE PORTRAIT	APPEARANCE OF PHASE PORTRAIT
1) Real distinct eigenvalues $\lambda 1, \lambda 2$ with $\lambda 1 > \lambda 2$	(a) $\begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda 2 \end{bmatrix}$ $\lambda 1$ and $\lambda 2$ have the same sign	NODE	
	(b) $\begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda 2 \end{bmatrix}$ $\lambda 1$ and $\lambda 2$ have opposite sign	SADDLE	
2) Equal eigenvalues $\lambda 1 = \lambda 2 = \lambda 0$	(a) $\begin{bmatrix} \lambda 0 & 0 \\ 0 & \lambda 0 \end{bmatrix}$	STAR-NODE	
	(b) $\begin{bmatrix} \lambda 0 & 1 \\ 0 & \lambda 0 \end{bmatrix}$	IMPROPER NODE	
3) Complex eigenvalues $\lambda 1 = \alpha + \iota\beta$ $\lambda 2 = \alpha - \iota\beta$ $\alpha = \frac{1}{2}\,\mathrm{tr}(A)$ $\beta = \frac{1}{2}\sqrt{-\Delta}$	$\begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}$ (a) $\alpha = 0$	CENTER	
	(b) $\alpha \neq 0$	SPIRAL	

Figure 4.17: Overall classification of phase portraits based on eigenvalues and Jordan canonical forms. [44]

4.4 Motion Anomaly Detection

This section describes the methodology on which the motion anomalies are detected based on the information obtained with the phase portrait classification process. The motion anomalies are detected in three distinct ways which shall be discussed below in this section. These three factors are derived from the overall characteristics that can be attributed to a given orientation field.

Given two orientation fields, which are usually obtained over different periods of time from optical flow fields, and given their approximated linear dynamical systems, we can detect the motion anomalies by detecting the differences in:

- Phase Portrait
- Critical Point
- Fit : The closeness or similarity between two orientation fields.

4.4.1 Change in Phase Portrait

Given two orientation fields obtained over time, one way of determining motion anomalies is by detecting the change in the overall flow pattern, which is given by the change in the phase portrait of the field under observation. In real-world cases, the flow field image exhibits different types of flows in different parts of the image. As seen in the figure 4.18

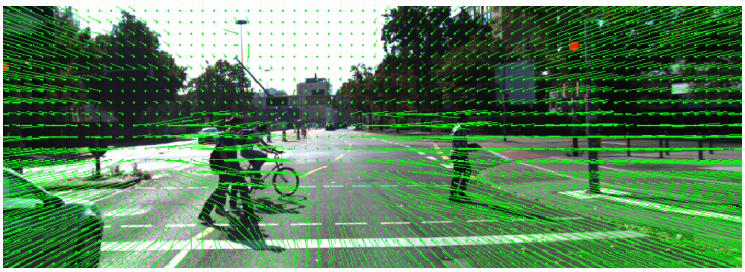


Figure 4.18: Different parts of the image exhibit different flow patterns. To detect motion anomalies, each part of the image is observed by breaking the image into windows and checked for a change in flow pattern or phase portrait.[17]

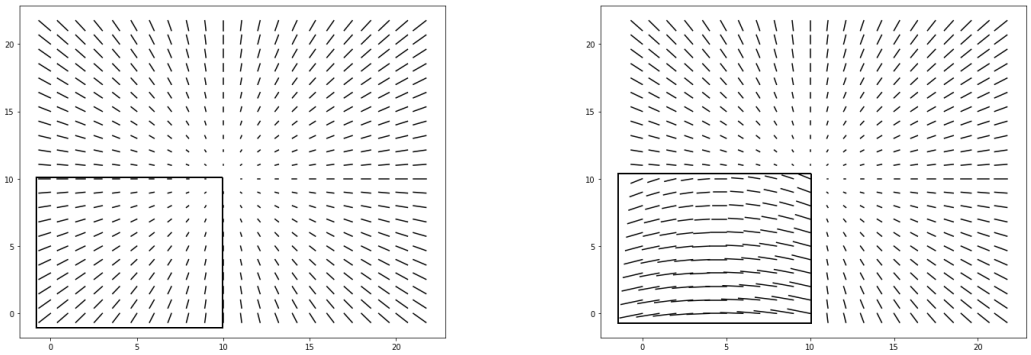


Figure 4.19: **Change in Phase Portrait:** The change in the flow pattern is reflected in the type of phase portrait. The selected window exhibits a phase portrait change from star-node to a saddle, which can be caused from movements such as the one shown above, people moving across the road in front of a car at a junction.

A motion anomaly is detected whenever the phase portrait changes from one frame, recorded at the previous instant of time to the current frame. As shown in the figure 4.19

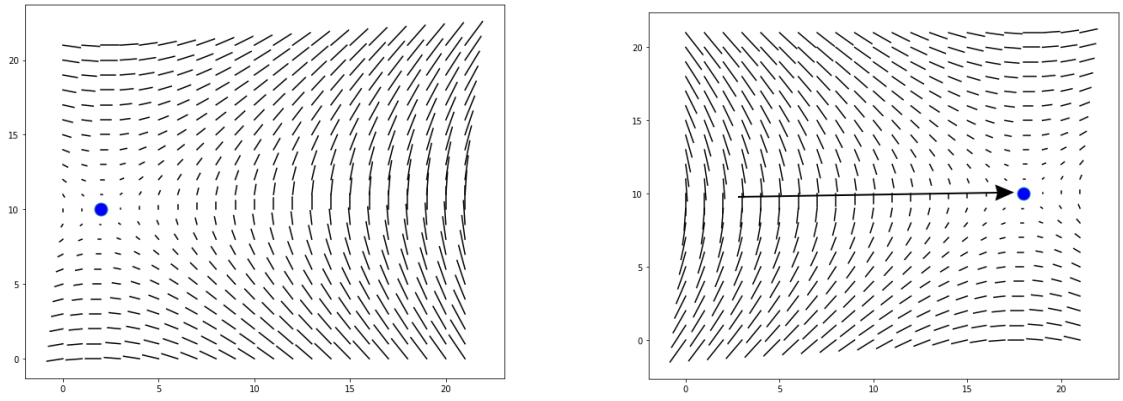
4.4.2 Shift in Critical Point

Another important feature of the optical flow field is the critical points of the phase portraits that have been detected. There are instances where the phase portrait might not change but the critical point shift gives indication of a significant change in the optical flow field.

Whenever the critical point of a phase portrait exhibits a change in its position above a certain threshold limit, a motion anomaly can be detected on the basis of this change. This can be seen as shown in the figure 4.20.

4.4.3 Variance by Fit

When the phase portrait of the field and the critical point position do not aid in detecting anomalies, fit is the additional variable by which we can gather useful information in order to detect anomalies. This fit is a measure of similarity or closeness between two orientation fields. Given the two orientation fields, the fit



(a) (b)

Figure 4.20: **Shift in Critical Point:** Visualization of sudden and significant movement of critical point from one part of the flow field to another. This significant change in position of the critical point suggests change in the movement patterns in the field under observation which can be detected as an anomaly.

between two fields or the fit between the flow field and the phase portrait is given by as in [44]:

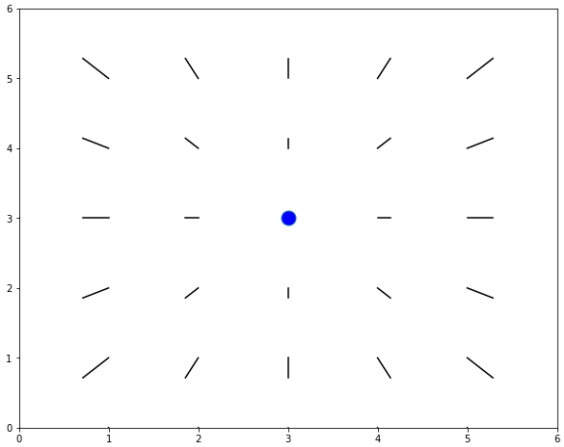
$$C = 1 - \frac{S}{N^2/2} \tag{4.27}$$

where, C is the fit value
S = Sum of areas of the triangle formed by the field vectors as in equation 4.6.
N = Size of the window while detecting phase portraits which is usually NxN.
Thus, we can also note that the measure of fit is a normalized one irrespective of the size of the window used for scanning through the field.

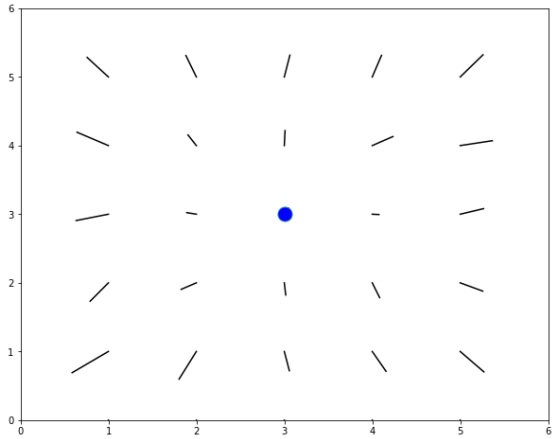
One such situation where variance by fit aids in the detection of an anomaly when the phase portrait change and critical point shift detector do not help is exhibited in figure 5.3. Whenever the fit varies between two orientation fields beyond a certain threshold limit, a motion anomaly can be detected.

Overview:

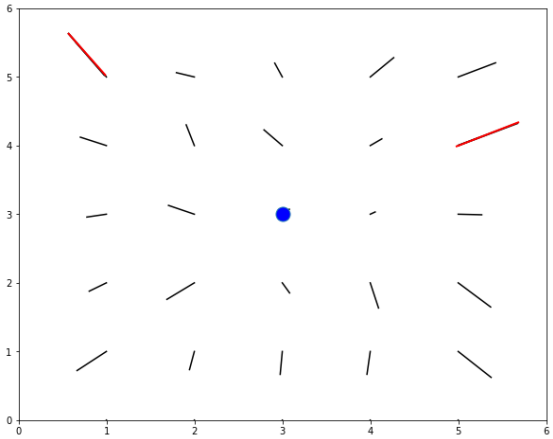
From the above sections we can see an almost **hierarchical** motion anomaly detection algorithm.



(a) Mapped phase portrait



(b) Orientation Field at (t-1)



(c) Orientation Field at (t)

Figure 4.21: **Variance by Fit:** This is a case where the two orientation fields observed at time instants t-1 and t share the same phase portrait and critical point but the fit varies. The orientations marked in red in (c) show drastic difference.

The overview can be given as follows:

- Firstly, the major changes in motion patterns and movements are detected as changes in phase portraits.
- Secondly, a shift in the entire flow field is detected as deviations in the critical point.
- Thirdly, the minor disturbances act as noise and are detected as variance in fit. The overall algorithm can be summarized as in the flowchart in the figure 4.22.

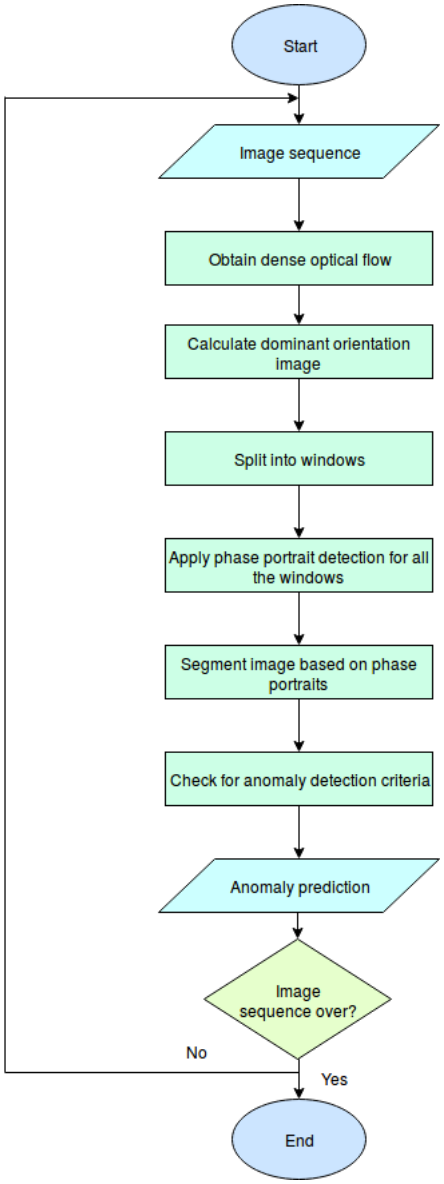


Figure 4.22: The flowchart describes the sequence of steps to be followed given an input sequence of images for motion anomaly detection. Each of the processing part in the flowchart has been explained in detail in the previous sections.

5

Experimental Evaluation

In this chapter, we shall discuss the about the choice of datasets and use them to test the proposed algorithm. As proposed earlier, this algorithm was developed to be able to generalize to scenarios irrespective of the motion of the camera. Thus, we shall be testing the algorithm with datasets obtained with static cameras and also with dynamic cameras. The static camera scenario was tested for a crowd activity analysis dataset and results are discussed. The dynamic camera scenario takes examples from datasets which were not intended for anomaly detection but were used to demonstrate the capability of the algorithm. This is due to the lack of labeled datasets for motion anomaly detection in dynamic cameras.

5.1 Static Camera

In order to demonstrate the generalizable capability of the algorithm, it was first tested on use cases which involve static cameras. Most state of the art motion anomaly detection algorithms have been devised for automated crowd surveillance applications. This usually involves a static camera placed in public spaces like gardens or institutional campuses. The cameras are usually positioned at some higher altitude to capture the large public spaces. In order to eliminate the tedious and monotonous job of a human to monitor the crowd, automated surveillance algorithms have been developed. We shall demonstrate the proposed algorithm for the same purpose.

5.1.1 Dataset Description

The dataset that was chosen for the experiment was the **UMN crowd activity**[57] dataset by the University of Minnesota. The dataset consists of scenes captured from various public spaces like gardens, halls, etc. A few examples of from the dataset are discussed below:

Normal scenes:

The normal scenes usually consists of people strolling the public spaces without any mishap as shown in figure 5.1. People move in different directions and do not stick to a constant direction all the time and people enter the scene through different directions as well, all of which adds to the complexity of the anomaly detection process.

(a) Gardens (b) Halls (c) Public pathways

Figure 5.1: **UMN- Normal Crowd Movement:** Examples from the UMN dataset showing normal scenarios. Normal cases involve people dispersed along the frame moving in random directions and entering and exiting the frame at different locations.

Anomalous Motion Scenes:

The anomalous motion consist of people exhibiting sudden disruptive motions as in emergency situations as shown in figure 5.2. In some cases, the sudden movement of people is along the direction of motion and in other cases, people exit the frame in random directions. This makes it challenging for the algorithm to distinguish

anomalous situations when the sudden movements are along the direction of usual movement of the crowd.

(a) Gardens (b) Halls (c) Public pathways

Figure 5.2: **UMN-Anomalous Crowd Movement:** Examples from the dataset showing anomalous scenarios. Anomalous cases involve sudden clamorous movement by the crowd, sometimes in the direction of usual movement or in random directions.

5.1.2 Experimentation

The main experiment was done on the UMN crowd activity dataset. The algorithm was implemented by varying one of the parameters to test for the most accurate one. The critical parameters that were involved are shown below. The UMN dataset contains:

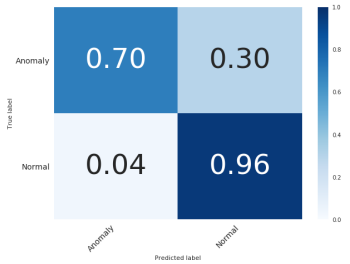
- Total Frames: 7719
- Anomalous Frames: 1136

The experiment was done for different prediction rates. Each of the tests used a different number of frames to average and then predict the result. The algorithm was tested by averaging over 10, 20 and 30 frames before prediction. The results showed that the prediction rate of 1 prediction / 30 frames performs better than making predictions over a lesser number of frames. the resulting confusion matrices are as follows:

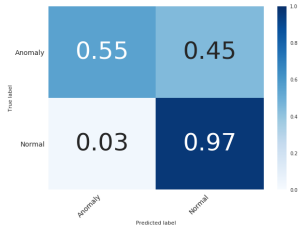
A motion anomaly was detected whenever a large number of windows showed a change in their respective phase portraits and a change in the magnitude of flow was

Parameter	Value
Actual Frame Size	320 x 240
Dominant Orientation Image Size	40 x 30
Frame Rate	30 fps
No. of Frames / Prediction	30,20,10
Anomaly Detector Window Size	10 x 10
Noise Threshold	0.05
Min. no. of Phase portrait changes	7

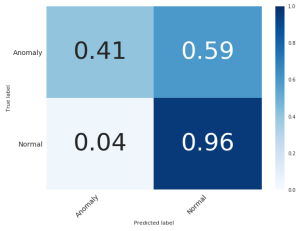
Table 5.1: Parameter Table



(a) Prediction per 30 frames



(b) Prediction per 20 frames



(c) Prediction per 10 frames

Figure 5.3: **Confusion Matrices:** The normalized confusion matrices for different prediction rates. It can be seen that a prediction per 30 frames works better than the other two.

examined as well. The limit for the number of changes is given as in the table 5.1. A couple of examples from the experiment are discussed below:

Failure Examples:

Failure Case 1:

One of the most frequent failure case was when the entrance of people from different directions into the frame lead to a change in the phase portrait of the associated window. Even though the motion of the people entering the frame is stable, a new addition of flow vectors causes a disturbance to be detected as a phase portrait change. This is shown in figure 5.4.

Failure Case 2:

Another case of failure was due to random movement in all the frames caused by people on the pathways. The random movement was not a clamorous one and hence was not classified as an anomaly in the ground truth data. This caused an increase in the number of phase portrait changes leading to a motion anomaly. This is shown in figure 5.5.

5.1.3 Additional Examples - QMUL Dataset

In order to prove the use case of the algorithm in other static camera cases, one more dataset was used. Instead of experimenting using the whole dataset, only a few scenes were picked in order to demonstrate the capability. The QMUL dataset [15] was used for this case. This dataset consists of footages from a traffic junction. The traffic junction is usually monitored to detect illegal crossings or U-turns and other such anomalies. A few examples of illegal U-turns were selected and tested.

Normal Scenes:

The normal scenes, when viewed from the static camera, just consists of vehicles moving across the junction in a straight path. The vehicles move vertically, from the top of the frame to the bottom or vice versa and horizontally, from the right of the frame to the left or vice versa as shown in figure 5.6.

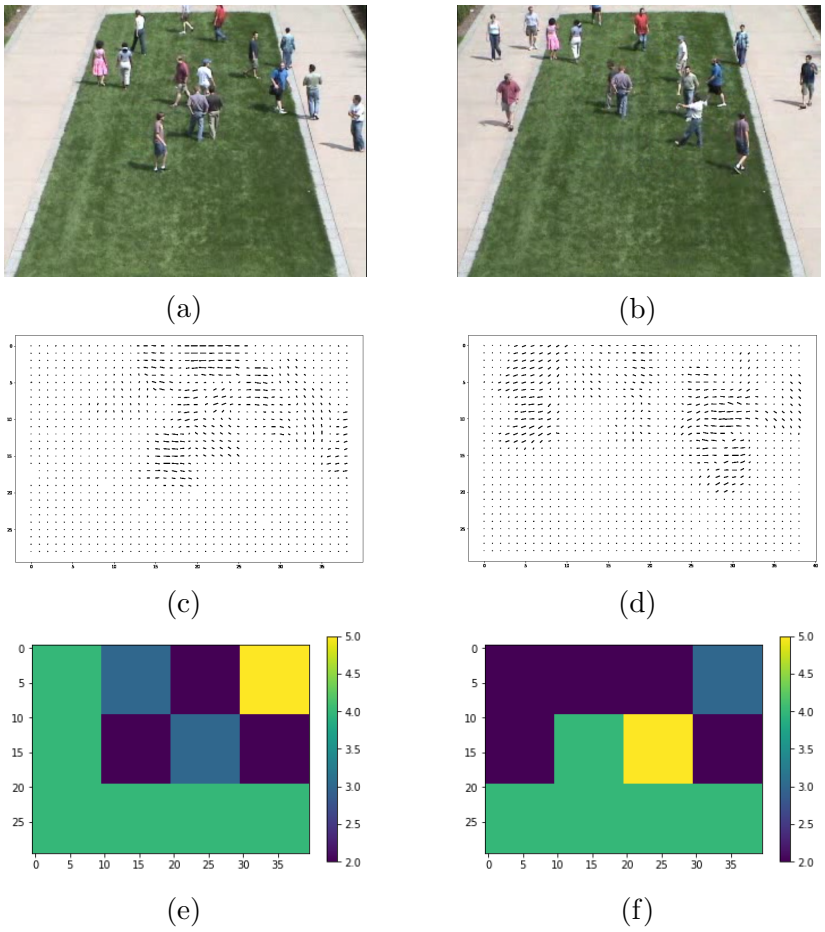


Figure 5.4: **Failure Case 1:** The Flows and the varied phase portraits assigned to the section of windows is shown. The change caused by the entry of people from the top corner of the frame as shown in (b) causes an increase in number of phase portrait changes and effects a false positive in our case. Areas exhibiting similar pattern of flow are assigned similar colors as shown in (e) and (f)

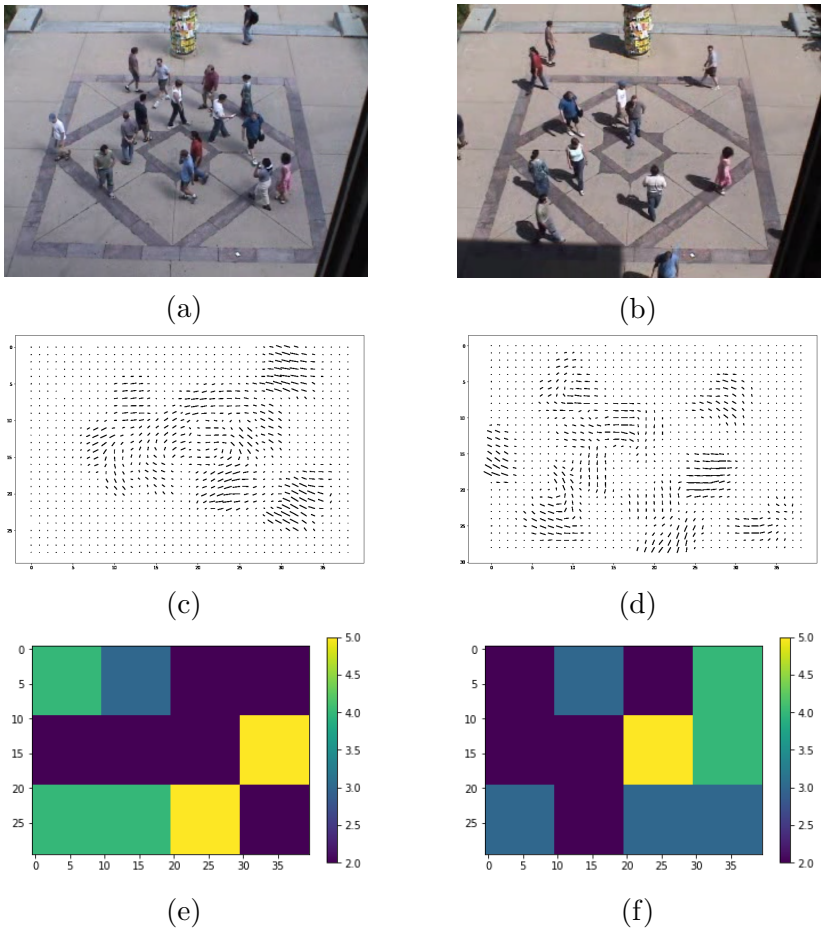


Figure 5.5: **Failure Case 2:** The Flows and the varied phase portraits assigned to the section of windows is shown. The change caused by random movement of people in all the windows as shown in (b) causes an increase in number of phase portrait changes and effects a false positive in our case. Areas exhibiting similar pattern of flow are assigned similar colors as shown in (e) and (f)

(a) Vertical Flow of traffic

(b) Horizontal Flow of Traffic

(c) Cross flow along the junction

Figure 5.6: **QMUL-Normal Traffic:** Examples from the dataset showing normal flow scenarios. Normal cases involve the usual flow of traffic across the junction essentially resulting in vertical, horizontal and cross flow as shown in the figure. [15]

Anomalous scenes:

The abnormal cases of traffic flow which have been examined consists of various instance where there is an illegal U-turn by the drive. This results in a flow pattern that normally is not a part of the traffic flow as shown in figure 5.7.

(a)

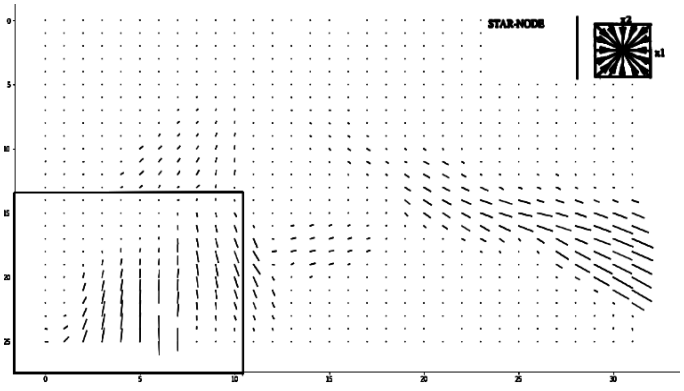
(b)

Figure 5.7: **QMUL- Illegal U-turns:** Examples from the dataset showing abnormal flow scenarios. Abnormal scenarios involve cases such as illegal u-turns as seen in the figure.

Motion Anomaly Detection - Illegal U-turns:

The motion anomalies are detected by detecting the change in the phase portraits associated with each of the windows. The normal motion pattern, as one can see

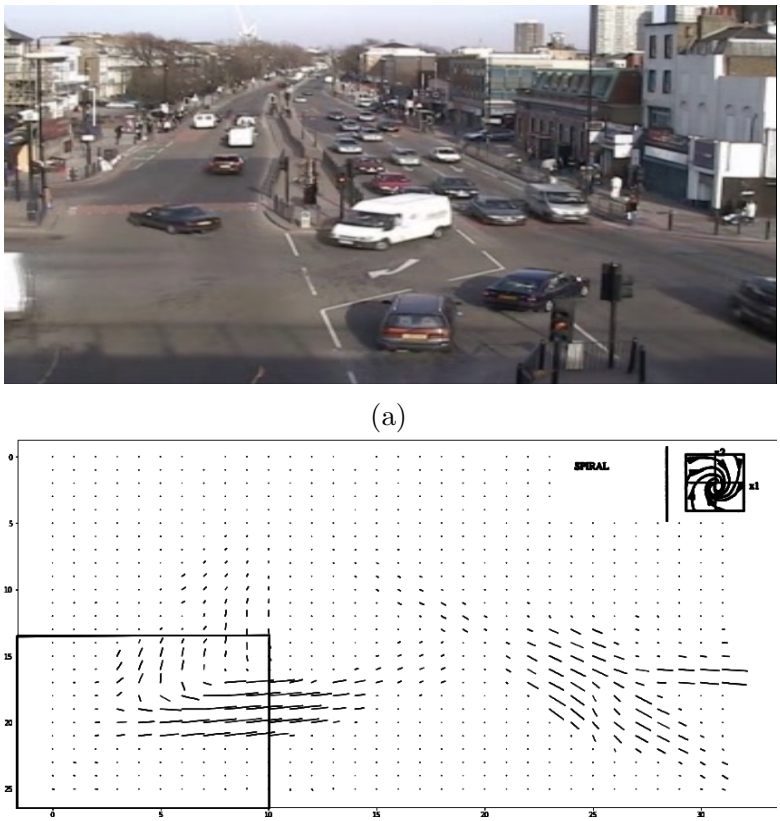
(a)



(b)

Figure 5.8: **Detecting Normal Flow:** The normal traffic flow patterns obtained from the traffic flow. The detected phase portrait is of type star-node.

are mostly linear and converge to the end of the road does. The abnormal motion pattern, in this case, a U-turn results in somewhat a distorted optical flow field which is close to being a circular since the car exhibits a semi-circular motion to complete a U-turn. The resulting changes in the phase portrait are as shown in figure 5.9.



(a)

(b)

Figure 5.9: **Detecting Anomalous Flow:** The anomalous traffic flow pattern obtained from the illegal U-turn in the traffic. The detected phase portrait is of type spiral.

5.2 Dynamic Camera

In this section, the algorithm is tested on examples from dynamic camera scenarios. Due to lack of labeled datasets for motion anomaly detection in the dynamic camera cases, a few examples from an already available dataset which is suitable for demonstration has been chosen. The KITTI dataset [17] was examined to find scenarios where motion anomaly detection would be useful. A few challenging scenarios have been chosen and the algorithm was put to test in those cases.

5.2.1 Data Description

The KITTI dataset has been used to demonstrate the capability of the algorithm in dynamic camera scenarios. The dataset has set benchmarks for optical flow and various tasks which use optical flow. It consists of a collection of videos from a standard station wagon mounted with two high-resolution color and grayscale video cameras which was driven along highways and rural areas.

5.2.2 Motion Anomaly Detection

Scenario 1: Detecting Sharp Turns

A few examples were picked from the dataset to demonstrate the algorithm's capability to detect sharp turns exhibited by the car along its run. This is detected by tracking the critical point of the flow pattern over a period of time. The critical point in terms of the overall flow pattern by the car gives an indication of where the car is heading towards. When the car is moving straight, the critical point is usually at the center. When there is a sharp turn exhibited by the car, the critical point exhibits a large shift, from the center or from one side of the frame to another. Detecting this large shift helps in detection of any sharp turns and similar movements. This is shown in figure 5.10.

Scenario 2: Vehicle Overtake

The algorithm can also be used to detect some rapid overtakes as the car moves along. Usually, the uneven shape of the car that overtakes causes a distortion in

the flow and thus effecting a phase portrait change or we can directly calculate the change in fit since the phase portrait and the critical point are similar. This is shown in figure 5.11.

5.3 Discussions:

Thus, the algorithm was tested on the complete UMN crowd activity dataset and a few examples were shown from the QMUL dataset and the KITTI dataset. The algorithm performed well when the number of frames averaged per prediction were higher and within the frame rate of the video sequence. A few things that caused hindrance to the performance of the algorithm are as follows:

- The false positive values are lower because the algorithm detects the starting frame of all the motion anomalies but misses out on the frames that are in between the start and the end of the motion anomaly. Many of the frames that lie in between are taken as normal, since the motion anomaly becomes the new normal for the algorithm.
- The algorithm fails in many cases to detect anomalies when there are rapid changes in the magnitude of the flow.
- When the magnitude of flow increases with the same motion pattern, the algorithm assumes the anomaly to be a normal motion pattern.
- An integration of a method is always needed to check the change in magnitude is required to detect anomalies with rapid change in magnitudes with same pattern. A threshold for the change was given during the tests.
- The tests were also performed with a noise threshold parameter since the algorithm classifies even the smallest amounts of magnitude to the closely matching phase portrait.

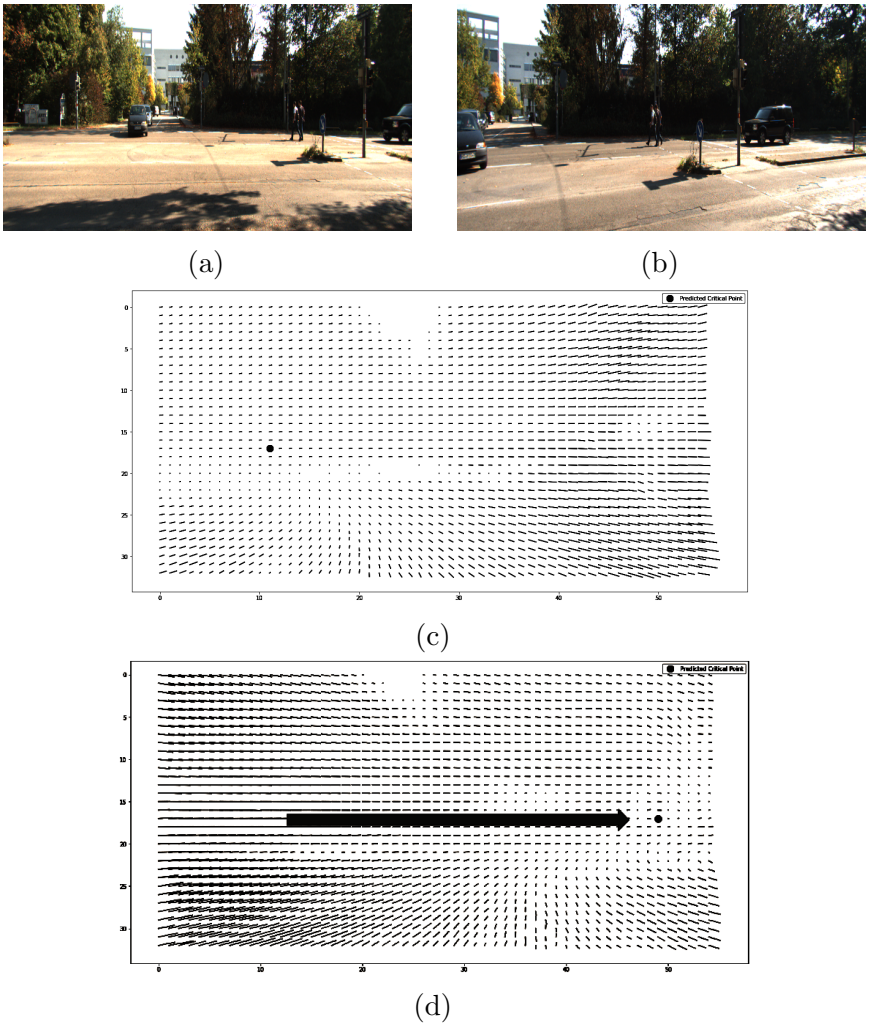
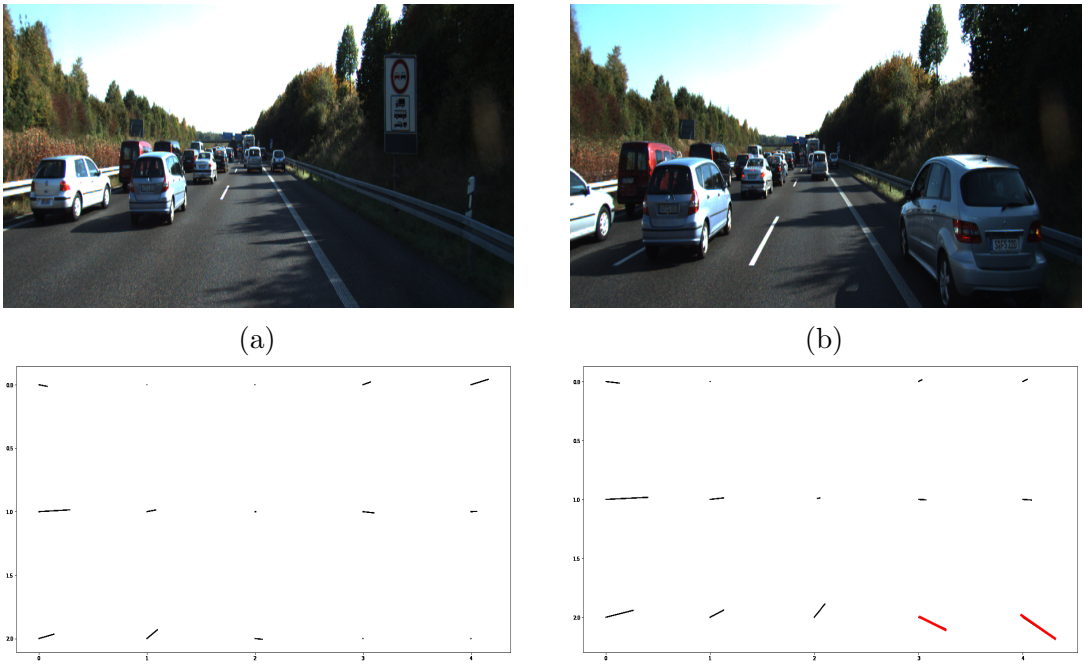


Figure 5.10: **Detecting Turns:** Visualizing the shift in critical point when the car does a sharp turn from (a) to (b). The sudden significant shift of the critical point serves as a good indicator for sharp turns as shown in (c) to (d).



(a) (b)

(c) (d)

Figure 5.11: **Vehicle Overtake:** In cases of similar flow and critical points, a difference in fit between the detected phase portraits can be examined to detect the motion anomalies. Rather than a bigger flow image, a reduce down dominant orientation image is shown where the vectors in the bottom left in (d) indicate a major difference in the orientation image due to the car entering the scene in (b)

6

Conclusions

In this work, a context-independent motion anomaly detection algorithm was developed and tested. The motion anomalies were detected over image sequences by obtaining and analysing the optical flow information generated using the Farneback optical flow method. This dense optical flow information was analysed to detect different kinds of flow patterns which result due to different motion patterns in the system under observation. By approximating the optical flow fields to linear dynamical systems, the different flow patterns were mapped to different phase portraits. This required the use of sophisticated non-linear optimization techniques in order to make the algorithm suitable for real-time motion anomaly detection. During this process, a novel linear algorithm for phase portrait classification was developed and tested. Further, by segmenting the flow field based on the different phase portraits and by analysing the information obtained through classifying flow patterns, a motion anomaly prediction technique was developed.

The developed algorithm evaluated on the UMN crowd activity dataset and the results were discussed. A few samples from other datasets like the QMUL traffic junction dataset and KITTI dataset were also tested in order to prove the capability and the generalizability of the detection algorithm. The experiment was intended to be done on static camera cases and dynamic camera cases but due to lack of labeled datasets only a few samples were tested for the dynamic camera case. Rigorous tests on the dynamic camera case would help in improving the algorithms capability.

6.1 Contributions

The major contributions of this work comprises of:

- A context-independent motion anomaly detection algorithm which detects motion anomalies by mapping optical flow field patterns to a finite set of phase portraits.
- A faster non-linear phase portrait classification algorithm by using evolution strategies for non-linear optimization.
- A novel linear algorithm for phase portrait classification.
- Evaluation of the performance of the detection algorithm on a real-time static camera scenario.
- Demonstration of the generalizing capability of the detection algorithm with samples from dynamic camera scenarios.

6.2 Lessons learned

- Detecting motion anomalies is complicated not just because of the nature of the problem, that anomalies are context-specific, but also because of the kind of data on which the anomaly prediction has to be made.
- The accuracy of optical flow did not match the expectations which lead to complications in the detection process.
- Numerical optimization techniques are key to real-time implementations of such algorithms.

6.3 Future work

The algorithm that has been developed has to be tested rigorously with dynamic camera scenarios. At first sight, the detector seem to be able to generalize well but more evaluations are needed. The algorithm only detects motion anomalies with varied flow patterns but does not capture well the change of magnitude of flow levels over time. The algorithm serves as a good framework but needs additional methods

and improvisations to be able to work for different and complicated scenarios.

The integration of this approach with other model based methods would be another direction to take the idea forward. Due to it's intuitive nature in which it detects anomalies, it can be used along with other state of the art model based algorithms that have been proven useful previously.

The optical flow information used can be obtained by other methods which give values at a higher accuracy. The implementation of the algorithm along with accurate optical flow estimators would be of great help in making better flow pattern predictions which in turn lead to accurate motion anomaly predictions.

References

[1] Md. Atiqur Rahman Ahad. Computer vision and action recognition - a guide for image processing and computer vision community for action understanding. In *Atlantis Ambient and Pervasive Intelligence*, 2011.

[2] Ajmal Mian. Optical flow and tracking, 2018. URL `http://teaching.csse.uwa.edu.au/units/CITS4402/lectures/Lecture11-Optical%20Flow%20and%20Tracking.pdf`. [Online; accessed 31-Dec-2018].

[3] Andrea Trevino. Introduction to k-means clustering, 2019. URL `https://www.datascience.com/blog/k-means-clustering`. [Online; accessed 3-January-2019].

[4] Borislav Antic and Björn Ommer. Spatio-temporal video parsing for abnormality detection. *CoRR*, abs/1502.06235, 2015. URL `http://arxiv.org/abs/1502.06235`.

[5] Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997. ISBN 0750303921.

[6] D. J.and Beauchemin S. S. Barron, J. L.and Fleet. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, Feb 1994. ISSN 1573-1405. doi: 10.1007/BF01420984. URL `https://doi.org/10.1007/BF01420984`.

[7] A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[8] Y. Benezeth, P. . Jodoin, V. Saligrama, and C. Rosenberger. Abnormal events detection based on spatio-temporal co-occurences. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2458–2465, June 2009. doi: 10.1109/CVPR.2009.5206686.

[9] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011. ISSN 0004-5411. doi: 10.1145/1970392.1970395. URL `http://doi.acm.org/10.1145/1970392.1970395`.

[10] Punarjay Chakravarty, Alan M Zhang, Ray Jarvis, and Lindsay Kleeman. Anomaly detection and tracking for a patrolling robot. 01 2007.

[11] Punarjay Chakravarty, Alan M Zhang, Ray Jarvis, and Lindsay Kleeman. Anomaly detection and tracking for a patrolling robot. 01 2007.

[12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL `http://doi.acm.org/10.1145/1541880.1541882`.

[13] Yie-Tarng Chen, Wen-Hsien Fang, Chih-Yuan Lee, and Kai-Wen Cheng. Abnormal detection in crowded scenes via kernel based direct density ratio estimation. 07 2015. doi: 10.1109/ChinaSIP.2015.7230397.

[14] cvx research. Demo: Rpca using tfocs, 2011. URL `http://cvxr.com/tfocs/demos/rpca/`. [Online; accessed 5-January-2019].

[15] David Russell. Qmul junction dataset, 2008. URL `http://www.eecs.qmul.ac.uk/~sgg/QMUL_Junction_Datasets/Junction/Junction.html`. [Online; accessed 5-January-2019].

[16] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45103-7.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] Gene H. Golub, michael A. Saunders. Linear least squares and qp, 2019. URL `http://infolab.stanford.edu/pub/cstr/reports/cs/tr/69/134/CS-TR-69-134.pdf`. [Online; accessed 5-January-2019].

[19] Nico Görnitz, Mikio Braun, and Marius Kloft. Hidden markov anomaly detection. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1833–1842. JMLR.org, 2015. URL `http://dl.acm.org/citation.cfm?id=3045118.3045313`.

[20] Charles Guyon, Thierry Bouwmans, and El hadi Zahzah. Robust principal component analysis for background subtraction: Systematic evaluation and comparative analysis. In Parinya Sanguansat, editor, *Principal Component Analysis*, chapter 12. IntechOpen, Rijeka, 2012. doi: 10.5772/38267. URL `https://doi.org/10.5772/38267`.

[21] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL `http://arxiv.org/abs/1604.00772`.

[22] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398. URL `https://doi.org/10.1162/106365601750190398`.

[23] B. Hayes and J. A. Shah. Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6586–6593, May 2017. doi: 10.1109/ICRA.2017.7989778.

[24] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan D. Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *CoRR*, abs/1703.03512, 2017. URL `http://arxiv.org/abs/1703.03512`.

[25] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan D. Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *CoRR*, abs/1703.03512, 2017. URL `http://arxiv.org/abs/1703.03512`.

[26] J. Kim and K. Grauman. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2928, June 2009. doi: 10.1109/CVPR.2009.5206569.

[27] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1446–1453, June 2009. doi: 10.1109/CVPR.2009.5206771.

[28] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):18–32, Jan 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.111.

[29] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. Dual motion GAN for future-flow embedded video prediction. *CoRR*, abs/1708.00284, 2017. URL `http://arxiv.org/abs/1708.00284`.

[30] R. Lin, E. Khalastchi, and G. A. Kaminka. Detecting anomalies in unmanned vehicles using the mahalanobis distance. In *2010 IEEE International Conference on Robotics and Automation*, pages 3038–3044, May 2010. doi: 10.1109/ROBOT.2010.5509781.

[31] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection - A new baseline. *CoRR*, abs/1712.09867, 2017. URL `http://arxiv.org/abs/1712.09867`.

[32] Chen Change Loy, Timothy M. Hospedales, Tao Xiang, and Shaogang Gong. Stream-based joint exploration-exploitation active learning. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1560–1567, 2012.

[33] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[34] George E. Martin. *Affine Transformations*. Springer New York, New York, NY, 1982. ISBN 978-1-4612-5680-9. doi: 10.1007/978-1-4612-5680-9_15. URL https://doi.org/10.1007/978-1-4612-5680-9_15.

[35] Mathworks. Farneback optical flow class, 2018. URL https://www.mathworks.com/help/vision/ref/opticalflowfarneback-class.html. [Online; accessed 31-Dec-2018].

[36] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–942, June 2009. doi: 10.1109/CVPR.2009.5206641.

[37] H. Nallaivarothayan, D. Ryan, S. Denman, S. Sridharan, and C. Fookes. An evaluation of different features and learning models for anomalous event detection. In *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Nov 2013. doi: 10.1109/DICTA.2013.6691480.

[38] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 01 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5.

[39] OpenCV. Opencv optical flow, 2018. URL https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html. [Online; accessed 31-Dec-2018].

[40] otoro. A visual guide to evolution strategies, 2017. URL http://blog.otoro.net/2017/10/29/visual-evolution-strategies/. [Online; accessed 5-January-2019].

[41] D T Pham, S S Dimov, and C D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005. doi: 10.1243/095440605X8298. URL https://doi.org/10.1243/095440605X8298.

[42] R. Raghavendra, A. Del Bue, M. Cristani, and V. Murino. Optimizing interaction force for global anomaly detection in crowded scenes. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 136–143, Nov 2011.

[43] Ananth Ranganathan. The levenberg-marquardt algorithm. *Tutorial on LM algorithm*, 11:101–110, 2004. URL `http://users-phys.au.dk/jensjh/numeric/project/10.1.1.135.865.pdf`.

[44] A. R. Rao and R. Jain. Analyzing oriented textures through phase portraits. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, pages 336–340 vol.1, June 1990. doi: 10.1109/ICPR.1990.118126.

[45] A.Ravishankar Rao and Brian G Schunck. Computing oriented texture fields. *CVGIP: Graphical Models and Image Processing*, 53(2):157 – 185, 1991. ISSN 1049-9652. doi: https://doi.org/10.1016/1049-9652(91)90059-S. URL `http://www.sciencedirect.com/science/article/pii/104996529190059S`.

[46] Mahdyar Ravanbakhsh, Moin Nabi, Hossein Mousavi, Enver Sangineto, and Nicu Sebe. Plug-and-play CNN for crowd motion analysis: An application in abnormal event detection. *CoRR*, abs/1610.00307, 2016. URL `http://arxiv.org/abs/1610.00307`.

[47] Robyn Owens. Optical flow, 2018. URL `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html`. [Online; accessed 31-Dec-2018].

[48] Peter J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984. doi: 10.1080/01621459.1984.10477105. URL `https://www.tandfonline.com/doi/abs/10.1080/01621459.1984.10477105`.

[49] Rowland, Todd and Weisstein, Eric W. Jordan canonical form — MathWorld, a wolfram web resource, 2019. URL `http://mathworld.wolfram.com/Eigenvalue.html`. [Online; accessed 5-January-2019].

[50] D. Ryan, S. Denman, C. Fookes, and S. Sridharan. Textures of optical flow for real-time anomaly detection in crowds. In *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 230–235, Aug 2011. doi: 10.1109/AVSS.2011.6027327.

[51] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, Zahra Moayed, and Reinhard Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 02 2018. doi: 10.1016/j.cviu.2018.02.006.

[52] D. Schiebener, N. Vahrenkamp, and T. Asfour. Visual collision detection for corrective movements during grasping on a humanoid robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 105–111, Nov 2014.

[53] C. . Shu, R. Jain, and F. Quek. A linear algorithm for computing the phase portraits of oriented textures. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 352–357, June 1991. doi: 10.1109/CVPR.1991.139715.

[54] Steven J. Miller. Method of least squares, 2019. URL `https://web.williams.edu/Mathematics/sjmiller/public_html/BrownClasses/54/handouts/MethodLeastSquares.pdf`. [Online; accessed 5-January-2019].

[55] D. H. Terman and E. M. Izhikevich. State space. *Scholarpedia*, 3(3):1924, 2008. doi: 10.4249/scholarpedia.1924. revision #137545.

[56] Marco Alexander Treiber. *Optimization for Computer Vision: An Introduction to Core Concepts and Methods*. Springer Publishing Company, Incorporated, 2013. ISBN 1447152824, 9781447152828.

[57] UMN. Unusual crowd activity dataset of university of minnesota, 2006. URL `http://mha.cs.umn.edu/proj_events.shtml`. [Online; accessed 5-January-2019].

[58] Weisstein, Eric W. Eigen value— MathWorld, a wolfram web resource, 2019. URL `http://mathworld.wolfram.com/Eigenvalue.html`. [Online; accessed 5-January-2019].

[59] Wikipedia contributors. Evolutionary algorithm — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Evolutionary_algorithm&oldid=865781438. [Online; accessed 7-January-2019].

[60] Wikipedia contributors. Phase space — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Phase_space&oldid=871980248. [Online; accessed 3-January-2019].

[61] Wikipedia contributors. Dynamical system — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Dynamical_system&oldid=862630052. [Online; accessed 3-January-2019].

[62] Wikipedia contributors. Optical flow — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Optical_flow&oldid=842522656. [Online; accessed 28-Dec-2018].

[63] Wikipedia contributors. Phase plane — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Phase_plane&oldid=873346729. [Online; accessed 5-January-2019].

[64] Wikipedia contributors. Phase line (mathematics) — Wikipedia, the free encyclopedia, 2019. URL https://en.wikipedia.org/w/index.php?title=Phase_line_(mathematics)&oldid=860830139. [Online; accessed 3-January-2019].

[65] Yellowbrick. Elbow method, 2019. URL http://www.scikit-yb.org/en/latest/api/cluster/elbow.html. [Online; accessed 3-January-2019].

[66] Dong Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 611–618 vol. 1, June 2005. doi: 10.1109/CVPR.2005.316.

[67] Ying Zhang, Huchuan Lu, Lihe Zhang, and Xiang Ruan. Combining motion and appearance cues for anomaly detection. *Pattern Recognition*, 51:443 – 452, 2016.

ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2015.09.005. URL http://www.sciencedirect.com/science/article/pii/S0031320315003362.