MEng Individual Project

Imperial College London

Department of Electrical and Electronic Engineering

SLAM Using A Neuromorphic Camera to Estimate the Trajectory of a Randomly Moving Robot and to Localise Objects of Interest in the Surrounding Environment.

Author:
Tejas Dandawate

Supervisor:
Prof. Pier Luigi Dragotti

Second Marker:
Prof. Patrick A. Naylor

January 27, 2022

Abstract

Neuromorphic sensing is a novel way of encoding analogue signals, inspired by the biological processing of information in our brains. Neuromorphic sensing is based on time encoding, where rather than recording the amplitude of the input signal at predefined times, one records the time instants where the amplitude surpasses a certain trigger mark. This leads to a very efficient way of acquiring and processing signals.

Simultaneous localisation and mapping (SLAM) is a popular application of robotics and computer vision, in which a robot moves autonomously along an unknown trajectory. A useful application of this would be in cases such as finding the location of natural fires, in which it is important to know both the mapping of the surrounding environment as well as the locations of both the robot and the objects of interest.

The main aim of the project is to utilise a neuromorphic camera to carry out SLAM (moving in an unknown trajectory), mapping out the environment around the robot and finding the location it is in. This could possibly leverage spiking neural networks to further emulate biological responses to the stimuli the camera would receive. Then, if time permits, the next step would be to carry out some object recognition to localise objects or points of interest in the local environment.

Acknowledgements

Thanks mum!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Neuromorphic data is a novel representation of data that has overarching benefits that are yet to be fully explored in a plethora of applications. The many purported benefits of event-based cameras have the potential to revolutionise efficient, low-power computer vision due to their efficient encoding of information. Since the cameras are asynchronous in nature they allow for low-latency feeds with very little motion blur and other similar visual artefacts. As well as this their pixel structure allows for a very high dynamic range making the uses of these cameras even more apparent.

With the emergence of the IoT, sensors have become, or at the very least will become, ubiquitous in everyday life. These devices operate using power at a premium since they need to make full use of their limited batteries. This scarcity is further emphasised when attempting to do more power-intensive tasks. One such task is computer vision, which is becoming evermore prevalent as a means of human-computer interaction. Classical frame-based cameras are very power intensive, and do not allow for a low active duty cycle (i.e., allowing for device sleep/idle time). The usual way this issue is alleviated it by the system reacting to an event trigger. Such events may include things such as; motion, timing, acceleration or temperature. When compared to the function of event-based cameras, where event detection in built into the system, this can be seen as a stop-gap measure.

The high temporal resolution and dynamic range of event-based cameras also presents further benefits when compared to frame-based cameras. These features allow for videos to be represented in a very high fidelity format that preserves more data in fast-paced and brightly lit environments. This may lead to more reliable use of the data for computer vision and even other purposes.

1.2 Objectives

The objectives of this project were roughly divided into three two main sections; main objectives and extensions.

1.2.1 Main Objectives

- Evaluate different Artificial Neural Network (ANN) models on neuromorphic data from external datasets.

  This allows for determining the performance of both traditional Neural Networks (NNs) as well the performance Spiking Neural Networks (SNNs) for comparison. These networks can take two main forms:

  - Two networks sequentially processing data. The first would be converting reconstructing spiking data into an intensity video, and the second would be applying pre-existing computer vision networks to analyse the frame-based output.
  - Having one network that takes the spiking data as input to directly carry out the intended function.

  The system will initially be used to solve a classification task.

- Create practical set-up to experimentally record data.

  Using a neuromorphic camera, data can be experimentally captured to assess the performance of any built networks on more realistic unseen input data.

- Carry out Simultaneous Localisation and Mapping (SLAM) for a robot moving in an unknown trajectory.

  Use most efficient NN model to solve the more complex task of SLAM. The neuromorphic camera can be used to capture data from a room in the absence of large sets of labelled datasets.

1.2.2 Challenges & Fall-backs

- Only using existing datasets rather than practical set-up.

  Since there is sufficient amounts of existing data-sets, they can be split into training, validation and testing sets themselves. This eliminates the need to generate more data for the testing process.

- Rather than focusing on SLAM the emphasis may be on object detection/recognition

  The project has be segmented into individual milestones, and so even if the final one of a SLAM algorithm isn't achieved, it is easy to change the scope of the project to be simply a classification or gesture recognition network.

1.3 Report Structure

A brief outline of the report is as outlined:

1. Background

   This section gives a background to the project subject. There are outlines of previous works in the field in order to highlight gaps in knowledge where more work can be done. It provides a good illustration of what issues there are and what value there would be in solving them.

2. Implementation Plan

   This section has a list of planned objectives and their estimated completion date.

3. Evaluation Plan

   This section presents a variety of ways in which the final project could be evaluated at each step. This allows for a comparison between existing solutions as well every iteration of solution given during the project.

4. Ethical, Legal and Safety Plan

   Any ethical, legal and safety considerations taken into account during the project are given in this section.

Chapter 2

Background

This chapter outlines background information required for understanding the basis for the project. The theory and literature serves to outline the main concepts used for neuromorphic SLAM, as well as to reveal gaps in existing research that require solidifying.

2.1 Event Cameras

Event based cameras can be described as 'bio-inspired sensors that differ from conventional frame cameras: Instead of capturing images at a fixed rate, they asynchronously measure per-pixel brightness changes, and output a stream of events that encode the time, location and sign of the brightness changes' [1].

2.1.1 Benefits

Event-based cameras are purported to provide a number of benefits including, but not limited to;

- **Very high temporal resolution**

  The reason for this is that whereas frame-based cameras have a certain frame-rate, event-based cameras do not have this limitation, meaning the "blind time" between frames is eliminated. The reason for this is that the function of a frame-based camera is dependent on the global shutter to capture the light at a particular instant, whereas event-based cameras can be thought of as having individual shutters for each pixel that are shut whenever an even occurs.

- **High dynamic range**

  The reason for this is again the fact that each pixel has its own individual shutter, but as well as this they all use a logarithmic scale, meaning they function well from very bright to very dim environments as well as fast shifts between the two.

- **Low power consumption**

- **High pixel bandwidth**

  Each pixel can capture events at the rate of kHz. This has the effect of reducing blur since there is a very high temporal resolution to begin with. This makes the system very responsive and therefore ideal for real-time systems.

- **Efficient Encoding**

  Since events are asynchronous and spatially sparse (i.e there are mainly 0 values in the matrix), the encoding is very efficient, as opposed to frame-based cameras that produce data that is very spatially dense.

The above benefits are very persuasive reasons to adopt neuromorphic cameras in many different applications. It is conceivable that if algorithms can make use of these benefits (since most classical algorithms play to the strengths of the data generated by frame-based cameras), real-time systems could be completely revolutionised.

2.1.2 Function

Event-based cameras differ from frame based cameras fundamentally, in that they do not rely on a global shutter closing at regular intervals to record information of a scene. Instead each pixel closes whenever it detects an 'event' occurring. The way to detects such events are dictated by the 'event generation model'[1].

Each pixel responds to changes in its log photo-current ($L = log(I)$, where I is the perceived brightness), giving the system a very high dynamic range. A recorded event 'k' has the format $e_k = (\mathbf{x}_k, t_l, p_k)$. THis is known as the Address Event Representation (AER). The first value is the spacial location of the event ($\mathbf{x}_k = (x_k, y_k)^\top$), the second value t_k is the temporal location, and the final value $p_k \in 1, -1$ indicates the polarity of the event (i.e in which direction the brightness gradient was changing). The brightness increment between two events at the same pixel is given by the equation $\Delta L(\mathbf{x}_k, t_k) = L(\mathbf{x}_k, t_k) - L(\mathbf{x}_k, t_k - \Delta t_k)$. In a perfect (noise free) environment an event is fired whenever the brightness increment reaches a temporal contrast threshold given by the equation $\Delta L(\mathbf{x}_k, t_k) = p_k C$ $(C > 0)$. It should be noted that the value of C could be variable and therefore different for $p_k = \pm 1$.

Additionally, we can approximate the temporal derivative of a pixels brightness by utilising the following Taylor expansion:

$$\Delta L(\mathbf{x}_k, t_k) \approx \frac{\delta L}{\delta t}(\mathbf{x}_k, t_k)\Delta t_k$$

$$\frac{\delta L}{\delta t}(\mathbf{x}_k, t_k) \approx \frac{\Delta L(\mathbf{x}_k, t_k)}{\Delta t_k} = \frac{p_k C}{\Delta t_k}$$

The above approximation, however, is only true under the assumption that Δt_k is exceedingly small. Since unlike frame-based cameras we do not measure absolute brightness, this is an indirect way of measuring and keeping track of the brightness within the frame.

Figure 2.1 shows the basic functionality of an event based camera. **(a)** is the simplified circuit diagram of the DAVIS pixel, which in **(b)** is used to convert light into events (shown in real life in images **(c)** and **(d)**). **(e)** shows how this setup would view a white square rotating on a black disk. It is a stream of events going from the past in green to the present in red. These events can then be seen overlaid on a natural scene in **(f)**.

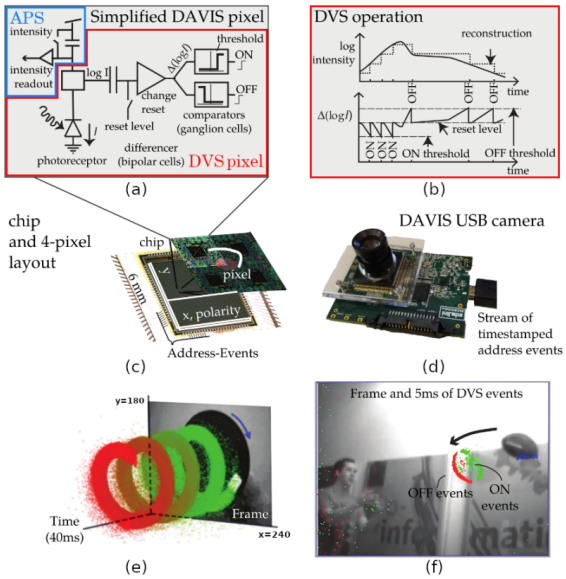


Figure 2.1: Summary of DAVIS event based camera[1]

2.2 Spiking Neural Networks and Neural Heterogeneity

Above are very persuasive reasons for utilising neuromorphic systems, but there still many challenges posed when attempting to do so. For example, each pixel only responds to brightness

change, but the problem is that such a change could be a result of not only scene changes, but also the position of the camera within the scene. For this reason most neuromorphic systems have currently been limited to stationary cameras. As well as this the system is especially prone to stochastic noise, due to inherent shot noise in photons and from transistor circuit noise [1]. In order to tackle such issues, it is useful to look at existing examples of spiking neural systems, such as biological brain with neurons working based on a spiking function. It is known that the brain is heterogeneous on every scale, in the past this was though to be simply a by-product of noisy processes, but more recently it can be shown that by evolving our largely homogeneous Spiking Neural Networks (SNNs), we can create more stable and robust systems [8].

The foundations of SNNs are in computational neuroscience. The mechanisms of neurons in the brain are the inspiration behind creating artificial neural networks with neurons that spike in the same way. Neurons in an a typical Artificial Neural Network (ANN) have a weight, bias and activation function. this means the output of the neuron can be specified as:

$$y = \theta(\sum_{j=1}^{n} w_j x_j - u_j)$$

This model was inspired by the biological neuron model shown in Figure 2.2. The function of the neuron is similar in that it produces an output based on a function of the input stream, but he form of this output is in the form of a 'spike' rather than a continuous function. The Σ shown in the diagram is actually the integration of all excitory and inhibitory input signals to the dendrites within the soma of the neuron. We can see that the electric potential of the neuron needs to exceed a certain threshold in order for the output of the neuron to be a spike. Spiking neural networks use a similar model of neurons in order to leverage the aforementioned benefits of neural heterogeneity.

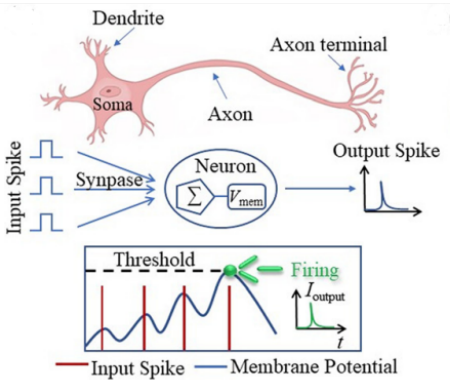


Figure 2.2: Diagram of structure and function of a biological neuron [2]

NEED TO READ EVENT PAPER TO SEE THEIR APPROACH TO STOCHASTIC NOISE [1]. ALSO READ PAPER[8] TO INCLUDE IMAGE OF AND EXPLANATION OF BRAIN NEURONS

2.3 Existing Algorithms for Event Analysis

For SLAM and pose estimation the problem again is that classical systems heavily rely on the structure of conventional cameras, and so there needs to be a radical paradigm shift in order to take events as inputs instead. The reason for this is that their function depends on iterative changes to the location probabilities using inputs, for which spiking inputs are ideal.

2.3.1 Optic-flow Methods

More classic computer vision techniques using optic-flow constraints can now be utilised to characterise the events detected by pixels. In frame-based systems, optic flow methods create a flow-field that describe the displacement vector (signifying direction and magnitude of movement) for each

pixel in the frame. A core constraint in this derivation is that the intensity of a local time-varying image region is constant under motion (for at least a short amount of time)[9].

NEED TO READ AND UNDERSTAND WHY THE FOLLOWING IS TRUE [9] [1]

$$\Delta L \approx -\nabla L \cdot v \Delta t_k$$

2.3.2 Localisation using Probabilistic Filters

Bayesian Inference

Unlike most other previous systems, probabilistic filters such as Bayesian filters inherently work for the new scenario with event-based cameras. Bayes's theorem can be derived from simple probabilistic rules[10]. We know $P(X|Y) = \frac{P(X,Y}{P(Y)}$ and similarly $P(Y|X) = \frac{P(Y,X)}{P(X)}$. Therefore we can re-arrange both to give $P(X|Y)P(Y) = P(Y|X)P(X)$, since $P(X,Y) = P(Y,X)$. Then from there formula for Bayesian inference can be trivially obtained:

$$P(XZ) = P(Z|X)P(X) = P(X|Z)P(Z)$$

$$P(X|Z) = \frac{P(Z|X)P(X)}{P(Z)}$$

In the above equations X is known as the prior (which is the assumed location of the camera) and Z is known as the posterior (which is the measurement taken by the sensor or camera). $P(Z|X)$ is known as the likelihood function, which indicates how likely it is to have received the particular reading given the assumed position.

Monte-Carlo Localisation

Now that we have the concept of Bayesian inference we can adapt it to create an efficient localisation algorithm. It includes initialising a number of particles that act as predictors of where in the map the camera is. We can now give the probability of a posterior camera position given a sensor reading.

The probability distribution $P(X|X)$ is a continuous function, and so updating each of the posteriors for every value of X is a computationally difficult problem. We can instead break it up into smaller bins to alleviate this issue. When generating these particles we can represent the probability distribution, albeit at a lower granularity. The benefit of this is that even though we cannot see the full distribution the peaks (i.e. the locations the camera is most likely to be in) are very well defined.

Now we can use the above simplification to carry out the following steps:

1. Randomly assign particle distribution across map
2. Apply Bayes' law to measurement to update particle distribution

   Bayes' rule can be simplified to be:

   $$w_{i_{x+1}} = P(z|x_{i_x}) \times x_{i_x}$$

   This can be done since the ignored multiplier on the right hand side will be normalised in the next step.
3. Normalise particle weights

   Now the particles will have weights that no longer sum to 1, and so we need to normalise them again to follow the usual rules of probability:

   $$w_{i_{x+1}} = \frac{w_i}{\sum_{i=1}^{N} w_i}$$
4. Re-sample particle distribution

   We now need to create a new set of particles that all have the same weight ($\frac{1}{N}$, but whose spacial distribution now reflects the probability density.

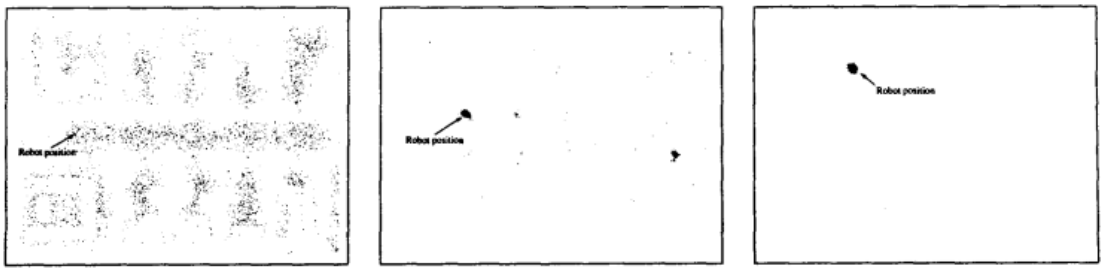


Figure 2.3: Example of Monte-Carlo localisation[3]

Figure 2.3 shows a typical example of the algorithm. The leftmost panel shows the random initialisation (or previous particle distribution), which then becomes the centrally shown distribution after one iteration. Since only one measurement is taken and the room is symmetrical it is possible that it could be in one of two location (hence the two dense clusters). After one more reading in the next iteration the algorithm is quickly able to narrow down the location of the robot. We can also see that any movement of the robot causes some noise to be added to the known robot location as we are estimating the location based on simple odometry using hardware such as wheel encoders to estimate the robots motion.

This algorithm, however, is classical and has therefore been mostly superseded by deep neural networks in most modern day applications. As well as this it is only applicable to localising the robot, whereas we want to be able to simultaneously map it. SLAM can be achieved using similar methods by identifying and moving around located points of interest and estimating their positions as well as the robots. These tasks have been efficiently solved by neural networks for classical frame based data, but there is still much ongoing research on how to do the same with spiking data.

2.3.3 SLAM Algorithm

The Monte-Carlo localisation technique is useful for estimating a vehicles position (i.e, its location and orientation) given a model (or map) of the environment surrounding it. Now we must do the pose estimation of a robot while simultaneously generating a map of its surroundings (as shown in Figure 2.4). It is clear from the diagram that the equipment is not perfect and in an idealised environment. There us uncertainty in both the robot position (since there is uncertainty in the robots odometry), and there is also uncertainty in the measurements the robot takes.

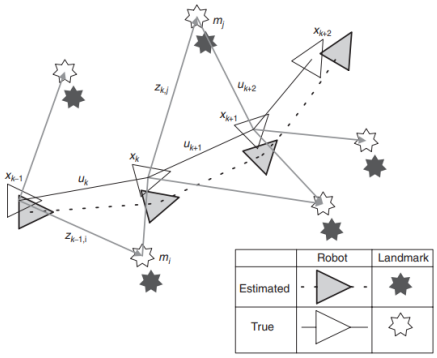


Figure 2.4: Diagram showing the fundamentals of the SLAM problem[4]

There are two main branches of SLAM algorithms; filtering methods and smoothing methods. The former includes methods such as the Extended Kalman Filter (EKF) and particle filtering (similar to the Monte-Carlo filtering explained earlier). With these methods the state is estimated iteratively on the job as latest measurements are input into the system. The latter uses the set of complete measurements to estimate the full trajectory of a robot. Pose (or factor) graph optimisation is one such smoothing method that has become exceedingly popular in modern-day SLAM solutions.

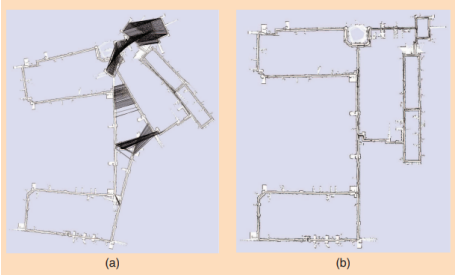


Figure 2.5: Diagram showing loop closer and optimisation steps of SLAM pose graph optimisation technique[5]

The pose graph optimisation technique relies on memorising the robots relative position and the readings it took in that position. For example we can see a possible robot trajectory in 2.4. The robot and its new relative positions are all nodes of the graph and are connected by edges. Whenever a new node is created in the graph, a reading is also taken and stored. Then whenever a new node is visited it is compared with previously stored readings, and if there is a reading that is similar to a very high degree, 'loop closure' can take place. In essence this means that the robot is now visiting a location it has already visited, and so the nodes can be joined together. Then once each the loop has been established we need to optimise the graph so that each edge (and therefore node) is at its most likely position relative to the other edges. The way this is done is that each edge of the graph has a relative certainty associated with it, and this certainty dictates how flexible that particular edge is in the optimisation process. Once the loop has been closed the edges are moved such that the overall certainty of the graph is maximised. Once this is complete the readings can be stitched together to form a map of the environment, and the location of the robot within it is very well defined. This process can be seen clearly in Figure 2.5, where in **(a)** loop closure takes places, and **(b)** shows the map created after the subsequent optimisation phase. The maps are created using 'binary occupancy grids' or 'probabilistic grids'. The former simply stores binary 1's and 0's on whether a particular section of the grid is occupied or not. The latter adapts this by having probabilities of occupancy for each section rather than just binary values.

2.4 Image Reconstruction Algorithms

Image reconstruction has been implemented for event data using on the direct optimised versions of Convolutional Neural Networks (CNNs). An example of this is the network named 'U-net'[11] which managed to reconstruct a video using 10M parameters to analyse events from an AER camera protocol. Recent work by Rebecq *et al.* illustrates a novel network architecture that reconstructs a video from a stream of events [6]. These methods are purported to allow the introduction of mainstream computer vision research to event cameras. Figure 2.6 shows an example of how converting spiking data to a video stream allows for use of classical computer vision algorithms.

A naive approach would be take take each would be to take each event $e_k = (\mathbf{x}_k, t_l, p_k)$ and assuming that the firing was due to the brightness change was due to a brightness change above a threshold $\pm C$ which is a constant that could be set by the user. If this was the case events could be directly integrated to recover the intensity map of images. however, the value C in reality does not remain constant and is heavily dependent on other factors such as event rate, temperature, and sign of brightness change. The implementation outlined instead makes use of a Recurrent Neural Network (RNN), that takes as input sets of events within a spatio-temporal window. For example, a stream of events will be broken down into sequences given by $\epsilon_i \; \forall i \in [0, N-1]$. Since each sequence is of fixed length N the framerate of the output video from the RNN is proportional to the event rate. Figure 2.7 shows demonstrates the functionality of such a network. Each event window ϵ_k is converted to a 3D event tensor and passed into the network along with the last K constructed images to generate the latest iteration of the image. It is clear from this that each new image is constructed by fusing the previous K images with the new stream of events.

NEED TO EXPLAIN ARCHITECTURE PROPERLY[6]

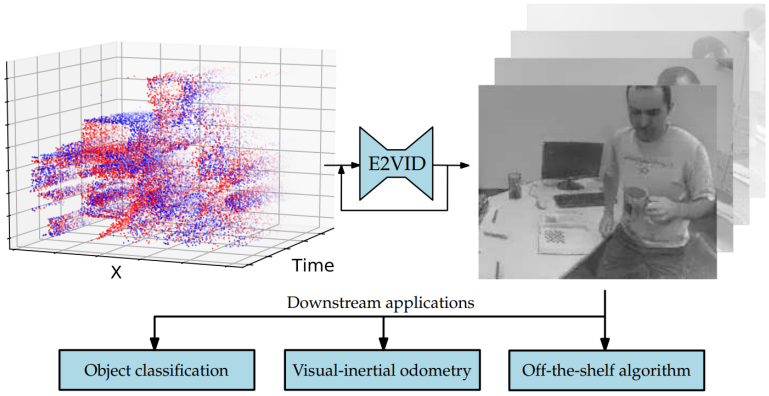


Figure 2.6: Illustration of mapping of spiking data to video stream to apply off-the-shelf algorithms to[6]

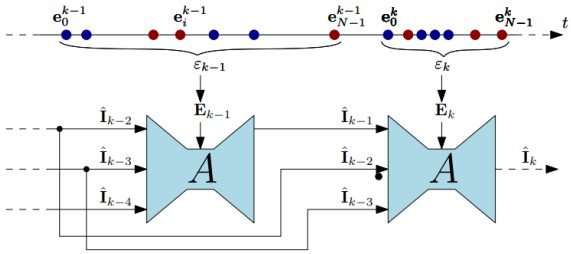


Figure 2.7: Overview of RNN used to generate video from sets of events[6]

2.5 Existing Datasets

There already exists many repositories of recorded neuromorphic data to get familiar with spiking data. In this section there are some examples of such datasets and a quick overview of their contents.

2.5.1 Neuromorphic-MNIST and Other Neuromorphic Datasets

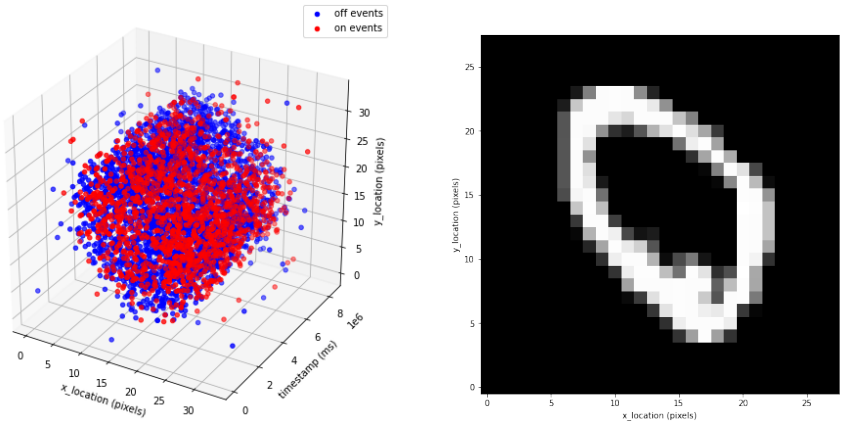
This data-set is a spiking version of the original frame-based MNIST dataset [12][7]. The dataset is identical to the original MNIST dataset, which is a set of handwritten digits, in all ways (including scale, size and sample split) bar one - it was captured using an ATIS sensor mounted on a motorised pan-tilt unit. This sensor moved while viewing the MNIST examples on an external monitor.

For each item in the dataset there is a binary file which has a list of events. Each event is characterised by a 40 bit unsigned integer. The integer gives the following information of a particular event:

- bit 39 - 32: X location (in pixels)
- bit 31 - 24: Y location (in pixels)
- bit 23: Polarity (0 for OFF, 1 for ON)
- bit 22 - 0: Timestamp (in microseconds)

An example of a visualisation of this data is shown in Figure 2.8, where the image used from the MNIST dataset is on the right in part **(b)** and the resulting spikes from the event-based camera are shown on the left in **(a)**. Here 'on events' are events where the intensity of a the particular pixel increased by an increment greater than a threshold and the 'off events' are when the intensities decrease by a increment less than a threshold. The representation is clearly very different to the one given by a classical camera, and therefore the use of such data has to have a different approach to classical techniques.

More examples of neuromorphic datasets include:



(a) (b)

Figure 2.8: Visualisation of events from a single training sample from the NMNIST dataset

- **DVS128**

  This dataset is a set of 11 hand gestures from 29 subjects under 3 illumination conditions. It was created to help create a real-time gesture recognition system that utilises the low power capabilities of frame-based cameras[13].

- **Heridelberg Spiking Datasets**

  The Spiking Heidelberg datasets for spiking neural networks[14] are useful for realising that spiking data is useful for so many more applications than computer vision. This dataset is split into two; The Spiking Heidelberg Digits (SHD) dataset and the Spiking Speech Command (SSC) dataset. Both of these datasets are audio-based classification datasets for which input spikes and output labels are provided.

2.5.2 Non-neuromorphic Datasets

Other data-sets such as fashion MNIST could also be converted to spiking times by treating image intensities as input currents to model neurons, so that higher intensity pixels would lead to earlier spikes, and lower intensity to later spikes.

Chapter 3

Implementation Plan

3.1 Timeline

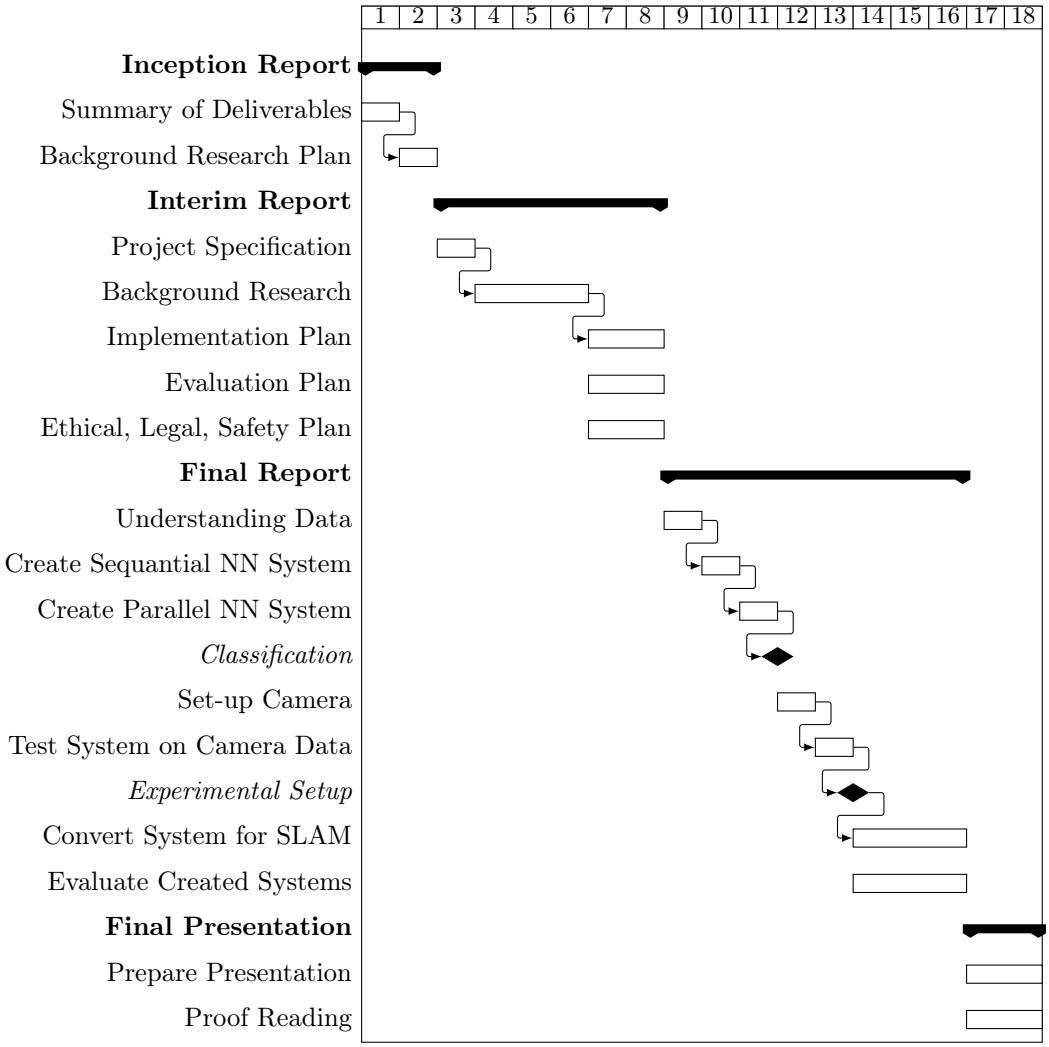


Figure 3.1: Gantt chart showing fortnightly progress

3.2 Objectives

Name	Description	Timeline
Summary of Deliverables	Creating initial plan of deliverables to be present in final project. As well as this it is important to have some initial plans for fall-backs to ensure that a complete project can be achieved.	Timeline
Background Research Plan	Create an initial list of relevant papers to kickstart project.	Timeline
Project Specification	Create a specification that precisely defines the goals of project as well as fall-backs for each case.	Timeline
Background Research	Use initial list of papers to conduct research behind the topic of the project. This involves looking at previous works and their respective gaps in knowledge. This is in order to give a basis on which to begin implementation, as it should provide all necessary tools and knowledge to begin initial preparations. It is also useful so give an insight on the need for the project and what problems it aims to overcome.	Timeline
Implementation Plan	Using analysis from background reading create a structured implementation plan outlining objectives and milestones, including a deadline for each.	Timeline
Evaluation Plan	Create a structure for the evaluation of any implemented material, including detailed explanations fo any formula and significance of any values when compared to baselines.	Timeline
Ethical, Legal, Safety Plan	Critically evalutate any ethical, legal or safety concerns that many arise as a result of this project and outline various ways in which to mitigate any possible issues.	Timeline
Understanding Data	Load and evaluate the utility of various data-sources and pre-process them ready for use in future NNs.	Timeline
Create Sequential NN System	Create model that first reconstructs frame based video from neuromorphic data to then pass into adaptations of existing networks that carry out classification on frame-based videos.	Timeline
Create Parallel NN System	Create a single NN that takes neuromorhpic data and directly creates the required output (classification or SLAM).	Timeline
Set-up Camera	Create set-up to obtain real-world data from event-based camera. This also required to create a pipeline to process data to be used in created NNs.	Timeline
Test System on Camera Data	Since real world data may be less idealised than pre-existing datasets, it is important to test the functionality of the system with data obtained from camera. As well as this tests can be done under various more extreme circumstances to test robustness of system.	Timeline

Name	Description	Timeline
Convert System for SLAM	Change function of ststem and train for SLAM rather than classification.	Timeline
Evaluate Created Systems	Use metrics listed in the evaltuation plan to check the performance of the system in order to compare each iteration in the project to each other as well as other existing solutions from other researchers.	Timeline
Prepare Presentation	Create slides and content to present final project results.	Timeline
Proof Reading	Check through report and presentation to remove errors and make any last-minute changes,	Timeline

Table 3.1: Explanations of objectives given in Figure 3.1

Chapter 4

Evaluation Plan

The evaluation plan is as given by the typical machine learning pipeline[15].

4.1 Dataset Preparations

4.1.1 Dataset Splitting

It is commonplace to split the shuffled dataset into three segments; training, validation and testing. The training data is what is used during each iteration of the back-propagation process. The validation data is what is unseen during this process and is instead used during the process to give an estimation of a models performance while training hyper-parameters. Finally, the testing data is withheld until it is needed to compare different final implementations with each other. It is vital the the training and validation data are withheld while training since otherwise they would not serve as a simulation of unknown data to measure the performance of the system in an unbiased manner. COmmon splits for training, validation and testing datasets are 60%/20%/20% and 80%/10%/10% respectively.

4.1.2 Dataset Cross-validation

With smaller datasets the splitting of data may mean that there is too little left to train with. This problem can be alleviated by using cross-validation. This method entails dividing hte dataset into a certain number of segments. Then in the first iteration of the learning process the first segment is used as testing data while the rest is used as training and validation. Then in the next iteration the process can be repeated by using the second segment as the testing, and so on until each segment has been used as testing data. Finally we can use the average of the errors for each testing dataset as the 'global error estimate'. It can be noted that the same segmentation and iteration process can be used for the training and validation datasets.

4.2 Evaluation Metrics

For a classification task, when we obtain the results from the test dataset (as shown in Table 4.1) we can calculate a variety of evaluation metrics that give various insights on our final model.

Labels	Predictions
1	1
1	2
3	8
9	9
6	9
⋮	⋮

Table 4.1: Example of results when inputting test data from NMNIST dataset[7] into the final model

4.2.1 Confusion matrix

Confusion matrices act as a visualisation of a systems performance. It shows possible true labels as well as possible predicted labels on either side, and filled in are the number of results that fit in each segment. In Table 4.2 the confusion matrix for the NMNIST dataset is shown as an example. It should be noted that a similar confusion matrix should be created taking each class as positive, then each metric can be calculated by taking the averages (as shown in Section 4.2.6). For each of the cells the number of matching records are stored to calculate each of the evaluation metrics. The table includes True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN).

		Predicted Class			
		1	2	3	...
Actual Class	1	TP	FN	FN	...
	2	FP	TN	TN	...
	3	FP	TN	TN	...
	4	FP	TN	TN	...
	5	FP	TN	TN	...
	⋮	⋮	⋮	⋮	⋱

Table 4.2: One particular confusion matrix for NMNIST dataset[7] for class 1 as the positive class

4.2.2 Accuracy

The accuracy of the system is a proportion of samples correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Note: classification error can also be used and is defined as $1 - accuracy$.

4.2.3 Precision

Precision is the proportion of positively predicted samples identified correctly.

$$Precision = \frac{TP}{TP + FP}$$

It should be noted that a high precision may mean that there are many false positives.

4.2.4 Recall

Recall is the proportion of actual positives correctly classified.

$$Recall = \frac{TP}{TP + FN}$$

It should be noted that a high recall may mean a lot of positive samples may be missed.

4.2.5 F-measure/F-score

This is defined as the harmonic mean of precision and recall in order to get one number as an average measure of performance.

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

4.2.6 Micro and Macro Averaging

Macro-averaging involves taking an average on the class level. Metrics are calculated for each class and then averaged at the end. Micro-averaging involves taking an average on the item level (i.e., taking the average of each of TP, FP, TN and FN to get the averages metrics).

4.3 Baselines for Comparison

In order to measure the performance of the system against current solutions it is useful to have a list of baseline performances. This way it can be inferred if there is an improvement being made by any newly created systems. Examples of systems that can be used include the reconstruction algorithm posed by H. Rebecq *et al.*[6] and existing gesture recognition algorithms for the DVS128 dataset[13].

4.4 Additional Testing with Camera

If the model were to train using only the datasets available, there is a risk that the data would be unbalanced and the network is training on a very specific set of idealised readings. Testing with extraneous data generated by a camera under various different conditions would allow for evaluating the performance of the networks on data that is completely unseen and dissimilar.

Chapter 5

Ethical, Legal and Safety Plan

In general the project in unproblematic, and posed minimal safety risks other than the ones presented when working with computers for extended periods of time (e.g., RSI, eye-strain, back aches etc.). In terms of ethical and legal considerations, however, it is important to make sure to use the practical camera setup in a sound manner. When collecting data from others it is important to be mindful of privacy issues. Consent must be taken from any participants if ever the need arises to take videos of others, but for the vast majority of project only videos of myself will be taken. As well as this it is important to consider where the system may be used if it is every put into production. Especially in IoT applications, privacy is of utmost concern, luckily with spiking data, if an intermediate video reconstruction cannot be obtained from the system, very little information can be discerned from spiking data. Therefore it would be very difficult to identify any individual within neuromorphic data itself.

Bibliography

[1] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[2] X. Liang, X. Zhang, J. Xia, M. Ezawa, Y. Zhao, G. Zhao, and Y. Zhou, "A spiking neuron constructed by the skyrmion-based spin torque nano-oscillator," *Applied Physics Letters*, vol. 116, no. 12, p. 122402, 2020.

[3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1322–1328.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[6] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3857–3866.

[7] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[8] N. Perez-Nieves, V. C. Leung, P. L. Dragotti, and D. F. Goodman, "Neural heterogeneity promotes robust learning," *bioRxiv*, pp. 2020–12, 2021.

[9] G. Gallego, C. Forster, E. Mueggler, and D. Scaramuzza, "Event-based camera pose tracking using a generative event model," *arXiv preprint arXiv:1510.01972*, 2015.

[10] T. C. Wallstrom, "The marginalization paradox and the formal bayes' law," in *AIP Conference Proceedings*, vol. 954, no. 1. American Institute of Physics, 2007, pp. 93–100.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[14] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[15] M. R. Antoine Cully and J. Wang, "Introduction to machine learning (co395), lecture 3," University Lecture, 2020.