The Open University of Israel
Department of Mathematics and Computer Science

Seminar of

Advanced Topics in Computer Science:

Neuromorphic Stereo Vision and Robot Controller

Seminar submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Department of Mathematics and Computer Science

By
Guy Tordjman

Prepared under the supervision of **Dr. Elishai Ezra Tsur**

August 2021

1 Abstract

In science fiction, as in the my imagination, artificial intelligence (AI) robots possess capabilities of full environmental sensing and flawless physical maneuverability. I keep wondering, is this fully autonomous creature can really exist? and how far is today's technology from creating such an entity?

AI is everywhere, and definitely not in a scary way. Scientists and engineers constantly building cyber physical intelligent systems targeted to improve our lives. From a smart refrigerator that makes sure there's always milk for our morning coffee[1] to a sophisticated environmental control systems, installed in buildings and factories. These systems are more energy efficient and therefore contribute to the global effort of preserving our planet[2].

As human beings, the ability to perceive the surroundings is executed effortless. In analogy to an electronic system, the surfaces and parts of a human body can be considered as a vast array of sensors, constantly transmitting data via electrical pulses to our main processing unit, AKA the brain. In the brain these electrical signals are translated to cognitive conclusions. Controlled motion of many living creatures in nature is possible through fast analysis of data perceived in the vision, touch, sound and even taste and smell designated parts of the brain.

Although currently not in production, the fully autonomous vehicle is a good example of a robotic system that faces the problem of "Driver out of the loop" where advanced vision and control theory algorithms are applied to tackle the simultaneous localization and mapping (SLAM) problem, while facing many other safety, physical constraints and environmental related challenges. The fact that autonomous vehicles, as means of transportation are intended to merge in to our existing road infrastructures and traffic, means that in order not to jeopardize the life of involved living beings, these algorithms must be dependable and run in real time. Although a solution to each of the mentioned difficult challenges does exist, when implemented in the same system, a few contradiction in the system's fundamental characteristics might emerge:

- The system must be able react in real-time, therefore, such cyber physical system hardware must support high-speed computation capabilities but still be power-efficient.

- The algorithms used to solve critical problems must be deterministic and provide a known solution to a given situation, yet they must be adaptive and have learning abilities.

- The system must be asynchronous to react in real-time when it recognizes a threat of internal or external origin. But on the other hand, the system is an integration of many subsystems that need to interact in coordination and work in synchronized cooperation to function dependably.

- The system needs to handle fast, dependable, high connectivity with numerous sensors and sub-systems. We might even require options like system update from a remote location. But on the other hand, by applying these high connectivity requirements, the system is exposed to privacy and security threats like disabling or distorting its sensors which could evolve into a fatal accident.

To confront these contradictions, scientist and engineers constantly come up with different approaches, manifested in efficient, state of the art algorithms and hardware architecture design. Since the 90's of the last century a new approach emerged. The fact that the human brain possess all the required computational capabilities and does so in real-time and with extremely low power consumption, made scientist to try and electronically model the brain as a computing unit. This brain inspired computing (BIC), also named neuromorphic engineering (NE), required deep understanding of the structure and computing processes of the brain. In the last decade big chip manufacturers like Intel and IBM reached a physical material limit when tried to create a faster and more power efficient hardware. As a result, a computation in memory (CIM) hardware architectures were introduced, Afterwards, a shift of paradigm was needed to apply spiking neural networks (SNN) algorithms that can solve problems like object detection, depth perception,and even speech recognition, which we, as humans, take for granted.

Due to the complexity of the mentioned above topics it is impractical to try and research them all at once. Therefore, in this seminar work, I'll introduce articles in the fields of neuromorphic stereo vision and neuromorphic proportional-integrate-derivative robot controller. In the first introductory part I'll briefly introduce the neuromorphic event based camera. In the first section, I'll introduce the concept of stereo vision and conclude with a in depth review of a neuromorphic cooperative algorithm for stereo vision[3][4]. The second section covers the topic of a PID controller and a review of a neuromorphic spike based PID implementation[5][6]. The third section details the project TODO and the fourth section includes conclusion and future resolutions.

Contents

2 Introduction

Neuromorphic engineering, also known as Brain Inspired Computing (BIC) is the use of very-large-scale integration (VLSI) systems containing electronic analog circuits to mimic neuro-biological architectures present in the nervous system. It is an interdisciplinary subject that takes inspiration from biology, physics, mathematics, computer science, and electronic engineering to design artificial neural systems, such as vision systems, head-eye systems, auditory processors, and autonomous robots, whose physical architecture and design principles are based on those of biological nervous systems[7].

2.1 Event Based Vision

Sight is, by far, the dominant sense in humans to perceive the world, and, together with the brain, to learn new things. Bio inspired technology of silicon retinas, or "event cameras" (EC), also commonly named dynamic vision sensors (DVS), are *asynchronous* sensors that pose a *paradigm shift* in the way visual information is acquired. This is because they sample light based on the scene dynamics, rather than on a clock that has no relation to the viewed scene. Compared to a standard camera that measures per-pixel brightness at a constant rate (frame), ECs asynchronously measures pixel-level brightness intensity changes (called *"events"*).

2.2 Principle of Operation of event cameras

Standard Cameras acquire data from all pixels at a frame rate specified by an external clock (e.g., 30 fps). ECs, such as the Dynamic Vision Sensor (DVS) [8] on the other hand, respond to brightness changes in the scene asynchronously and independently for every pixel. The output of an EC is a variable data-rate sequence of digital "events" or "spikes", with each event representing a change of brightness (log of the brightness intensity) of predefined magnitude at a pixel at a particular time. This encoding is inspired by the spiking nature of biological visual pathways. Each pixel memorizes the log intensity each time it sends an event, and continuously monitors for a change of sufficient magnitude from this memorized value (Fig. 1a). When the change exceeds a threshold, the camera sends an event, which is transmitted from the chip with the x, y location, the time t, and the 1-bit polarity p of the change (i.e., brightness increase ("ON") or decrease ("OFF")). This event output is illustrated in Figs. 1b, 1e and 1f. The events are transmitted from the pixel array to periphery and then out of the camera using a shared digital output bus readout. ECs are data-driven sensors: their

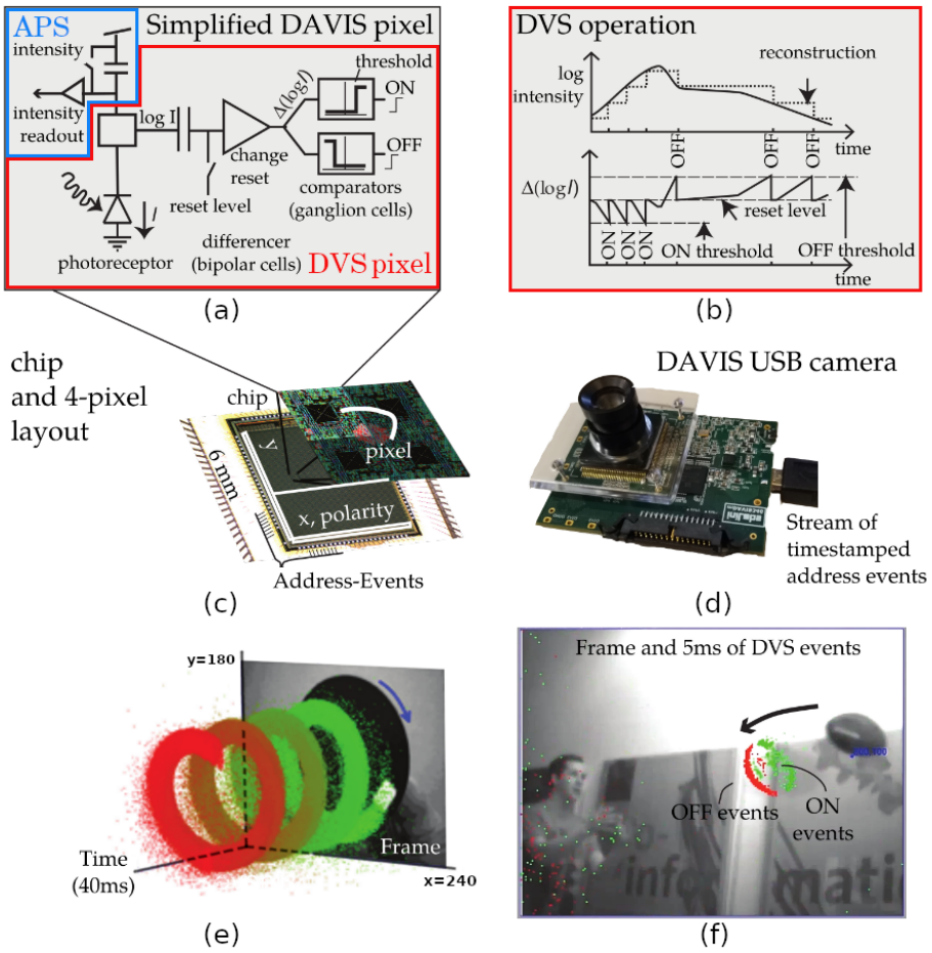


Figure 1: Summary of the DAVIS camera [4], comprising an event-based dynamic vision sensor (DVS [2]) and a frame-based active pixel sensor (APS) in the same pixel array, sharing the same photodiode in each pixel. (a) Simplified circuit diagram of the DAVIS pixel (DVS pixel in red, APS pixel in blue). (b) Schematic of the operation of a DVS pixel, converting light into events. (c)-(d) Pictures of the DAVIS chip and USB camera. (e) A white square on a rotating black disk viewed by the DAVIS produces grayscale frames and a spiral of events in space-time. Events in space-time are color-coded, from green (past) to red (present). (f) Frame and overlaid events of a natural scene; the frames lag behind the low-latency events (colored according to polarity).

output depends on the amount of motion or brightness change in the scene. The faster the motion, the more events per second are generated. Events are timestamped with microsecond resolution and are transmitted with sub-millisecond latency,which make these sensors react quickly to visual stimuli.

2.3 Advantages of event cameras

ECs offer numerous potential advantages over standard cameras::

1. High Temporal resolution: monitoring of brightness changes is fast, in analog circuitry, and the read-out of the events is digital, with a 1 MHz clock, i.e., events are detected and timestamped with microsecond resolution. Therefore, ECs can capture very fast motions, without suffering from motion blur typical of frame-based cameras. This property is illustrated in Fig. 2.

2. Low Latency: each pixel works independently and there is no need to wait for a global exposure time of the frame: as soon as the change is detected, it is transmitted. Hence, ECs have minimal latency: about $10\mu s$ on the lab bench, and sub-millisecond in the real world.

3. High dynamic range (HDR): The very high dynamic range of event cameras ($> 120dB$) notably exceeds the $60dB$ of high-quality, frame-based cameras, making them able to acquire information from moonlight to daylight. It is due to the facts that the photoreceptors of the pixels operate in logarithmic scale and each pixel works independently, not waiting for a global shutter. Like biological retinas, DVS pixels can adapt to very dark as well as very bright stimuli.

4. Low Power: Because event cameras transmit only brightness changes, redundant data is excluded and power is only used to process changing pixels. Most ECs use about $10mW$, and there are prototypes that achieve less than $10\mu W$.

2.4 Event Generation Model

An event camera has independent pixels that respond to changes in their log photo-current $L = log(I)$ ("brightness"). Specifically, in a noise free scenario, an event $e_k = (x_k, t_k, p_k)$ is triggered at pixel $x_k = (x_k, y_k)$ and at time t_k as soon as the brightness increment since the last event at the pixel, i.e. $\Delta L(x_k, t_k) = L(x_k, t_k)L(x_k, t_k\text{-}\Delta t_k)$, reaches a temporal contrast threshold $\pm C$ (Fig. 1b), i.e., $\Delta L(x_k, t_k) = p_k C$, where $C > 0$, Δt_k is the time elapsed

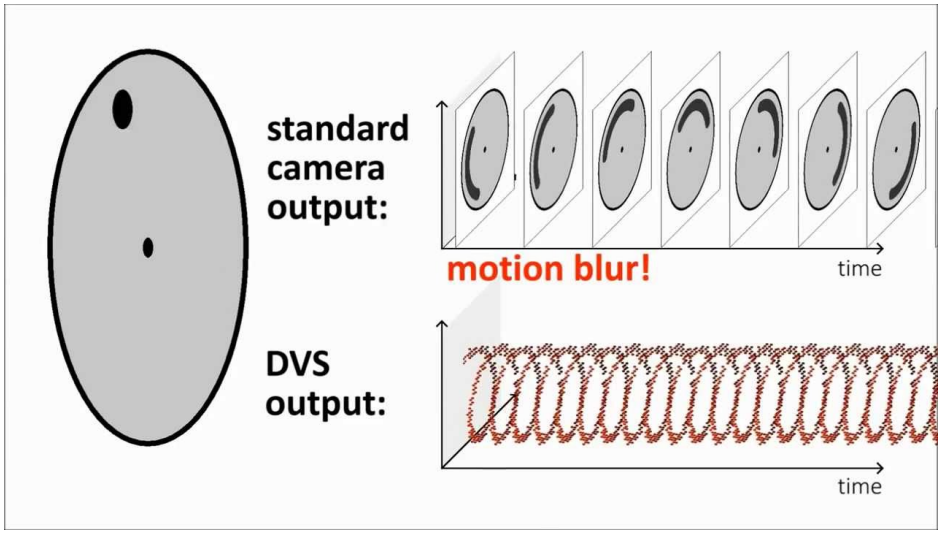


Figure 2: High temporal resolution of EC resulting in reduced motion blur.

since the last event at the same pixel, and the polarity $pk \in \{+1, -1\}$ is the sign of the brightness change. Typical DVS thresholds are set between 10% - 50% illumination change.

2.5 Event Camera Availability

Tab. 1 summarizes the most popular or recent cameras. The numbers therein are approximate since they were not measured using a common test-bed. Event camera characteristics are considerably different from other CMOS image sensor (CIS) technology, and there is a no standard specifications to be better used by researchers.

cost:Currently, a practical obstacle to adoption of event camera technology is the high cost of several thousand dollars per camera, similar to the situation with early time of flight, structured lighting and thermal cameras. The high costs are due to non-recurring engineering costs for the silicon design and fabrication (even when much of it is provided by research funding) and the limited samples available from prototype runs. It is anticipated that this price will drop precipitously once this technology enters mass production, as shown by the "Samsung SmartThings Vision" [9] consumer-grade home monitoring device which contains an event camera [10] and sells for 100 dollars.

Table 1: Estimated power consumption of classical hardware implementation versus neuromorphic hardware implementation.

	Supplier	iniVation			Prophesee				Samsung			CelePixel		Insightness
	Camera model	DVS128	DAVIS240	DAVIS346	ATIS	Gen3 CD	Gen3 ATIS	Gen 4 CD	DVS-Gen2	DVS-Gen3	DVS-Gen4	CeleX-IV	CeleX-V	Rino 3
Sensor specifications	Year, Reference	2008 [2]	2014 [4]	2017	2011 [3]	2017 [67]	2017 [67]	2020 [68]	2017 [5]	2018 [69]	2020 [39]	2017 [70]	2019 [71]	2018 [72]
	Resolution (pixels)	128 × 128	240 × 180	346 × 260	304 × 240	640 × 480	480 × 360	1280 × 720	640 × 480	640 × 480	1280 × 960	768 × 640	1280 × 800	320 × 262
	Latency (µs)	12μs @ 1klux	12μs @ 1klux	20	3	40 - 200	40 - 200	20 - 150	65 - 410	50	150	10	8	125μs @ 10lux
	Dynamic range (dB)	120	120	120	143	> 120	> 120	> 124	90	90	100	90	120	> 100
	Min. contrast sensitivity (%)	17	11	14.3 - 22.5	13	12	12	11	9	15	20	30	10	15
	Power consumption (mW)	23	5 - 14	10 - 170	50 - 175	36 - 95	25 - 87	32 - 84	27 - 50	40	130	-	400	20-70
	Chip size (mm^2)	6.3 × 6	5 × 5	8 × 6	9.9 × 8.2	9.6 × 7.2	9.6 × 7.2	6.22 × 3.5	8 × 5.8	8 × 5.8	8.4 × 7.6	15.5 × 15.8	14.3 × 11.6	5.3 × 5.3
	Pixel size (μm^2)	40 × 40	18.5 × 18.5	18.5 × 18.5	30 × 30	15 × 15	20 × 20	4.86 × 4.86	9 × 9	9 × 9	4.95 × 4.95	18 × 18	9.8 × 9.8	13 × 13
	Fill factor (%)	8.1	22	22	20	25	20	> 77	11	12	22	8.5	8	22
	Supply voltage (V)	3.3	1.8 & 3.3	1.8 & 3.3	1.8 & 3.3	1.8	1.8	1.1 & 2.5	1.2 & 2.8	1.2 & 2.8		1.8 & 3.3	1.2 & 2.5	1.8 & 3.3
	Stationary noise (ev/pix/s) at 25C	0.05	0.1	0.1	-	0.1	0.1	0.1	0.03	0.03		0.15	0.2	0.1
	CMOS technology (nm)	350 2P4M	180 1P6M MIM	180 1P6M MIM	180 1P6M	180 1P6M CIS	180 1P6M CIS	90 BI CIS	90 1P5M BSI	90	65/28	180 1P6M CIS	65 CIS	180 1P6M CIS
	Grayscale output	no	yes	yes	yes	no	yes	no	no	no	no	yes	yes	yes
	Grayscale dynamic range (dB)	NA	55	56.7	130	NA	> 100	NA	NA	NA	NA	90	120	50
	Max. frame rate (fps)	NA	35	40	NA	NA	NA	NA	NA	NA	NA	50	100	30
Camera	Max. Bandwidth (Meps)	1	12	12	-	66	66	1066	300	600	1200	200	140	20
	Interface	USB 2	USB 2	USB 3		USB 3	USB 3	USB 3	USB 2	USB 3	USB 3			USB 2
	IMU output	no	1 kHz	1 kHz	no	1 kHz	1 kHz	no	no	1 kHz	no	no	no	1 kHz

3 Neuromorphic Stereo Vision

Stereo-vision refers to the method of recovering depth information from both eyes, or in the artificial context, machine stereo vision, also referred to as stereoscopic vision, extracts the data from two visual sensors. In biology this is possible due to the laterally shifted eyes, gaining slightly different versions of a scene. The brain matches the corresponding points of both images and computes their disparity.
While biology computes disparities seemingly effortless, classical, non learning based approaches of computing stereo disparities in real-time are too computationally expensive. This is mainly caused by acquiring and processing huge amounts of redundant frame-based data. Furthermore, with increasing complex scenes and noise the computational expense of common machine vision system increases significantly. This computationally complex issue is referred to as the correspondence problem - finding the corresponding dots in a pair of images.
Neuromorphic visual sensors, similar to retinal output cells, are event-based sensors that transmit information asynchronously as a continuous stream of events. Spiking Neural Networks (SNN), due to their asynchronous operation principle, are a natural match for event-based sensors.

This section starts with defining the stereo correspondence problem and binocular disparity. Subsequently, the principles of cooperative algorithms will be introduced, followed by review of Osswald et al where a SNN was used to implement and test such algorithm on a neuromorphic hardware.

3.1 Stereo Correspondence and Binocular Disparity

Given more than a single image of the same 3D scene, taken from different points of view, the correspondence problem refers to the task of finding a set of points in one image which can be identified as the same points in another image. To do this, points or features in one image are matched with the corresponding points or features in another image. When there are exactly two cameras, observing the same 3d scene, the word *stereo* is used in "stereo matching" or "stereo correspondence problem". Binocular disparity refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes horizontal separation (parallax). The brain uses binocular disparity to extract depth information from the two-dimensional retinal images in stereopsis. In computer vision, binocular disparity refers to the difference in coordinates of similar features within two stereo images.

Given 2 images of pixel size n^2 and a pixel of one of the images, a naive

approach as depicted in *algorithm 1* takes $O(N^2)$ operation to find the corresponding pixel in the other image.

Algorithm 1 Find-Correspondence(Image Img[n,n], src p

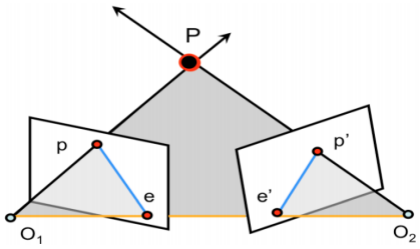
$H \leftarrow Min - Heap$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  for $j \leftarrow 1, 2, \ldots, n$ **do**
    $x \leftarrow Img[i, j]$
    $H.Insert(key = x, value = abs(p - x))$
  end for
end for
return $H.get - Min()$

Algorithm 1 is a simplified, brute force algorithm that examines all pixels in the image and picks the best match. Note that this algorithm returns the best matched pixel but this does not necessarily mean that it is the correct one. For example in the case of a single colored background or a reflecting surface. To solve such issue, a correlation based approach can be applied, where, the cross correlation score of a patch(small area of a window of pixels) from a source image to a sliding patch of the target image is calculated. The patch with the maximum correlation score is selected. Another applicable approach is based on calculating the sum of squared differences (SSD) or sum of absolute differences (SAD) between two pixel patches/windows or between features like edges, lines or corners [11].

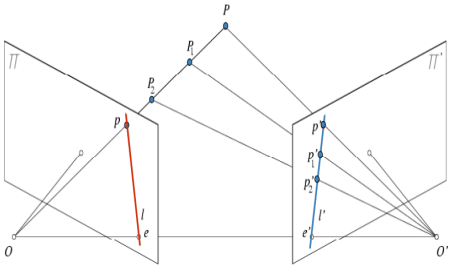
3.1.1 Epipolar Geometry and the Epipolar Constraint

Another, more efficient way is to use *epipolar geometry* to localize the search for areas where potential corresponding features/patches can be found. *Epipolar geometry* of a stereo pair is the geometry that relates the cameras, points in 3D, and the corresponding observations. As illustrated in Fig. 3a, the standard *epipolar geometry* setup involves two cameras observing the same 3D point P, whose projection in each of the image planes is located at p and p' respectively. The camera centers are located at O_1 and O_2, and the line between them is referred to as the *baseline*. The plane defined by the two camera centers and P is called the *epipolar plane*. The locations of where the *baseline* intersects the two image planes are known as the the *epipoles* e and e' . Finally, the lines defined by the intersection of the *epipolar plane* and the two image planes are known as the *epipolar lines*. The epipolar lines have the property that they intersect the baseline at the respective *epipoles* in the image plane.

Epipolar Geometry views allows us to constrain where the corresponding pixel for some image point in the first view must occur in the second view. As seen in Fig. 3b, the potential matches for a point in the first image must lie on the corresponding epipolar line of the second image. This constraint is called the *Epipolar Constraint*, which reduces the correspondence problem complexity solution to a single dimension search along conjugate epipolar lines.

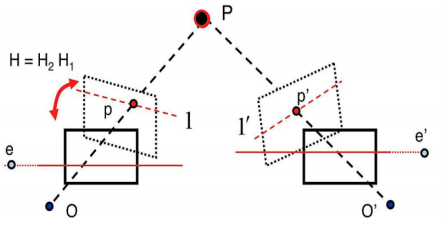


(a) The general setup of epipolar geometry.

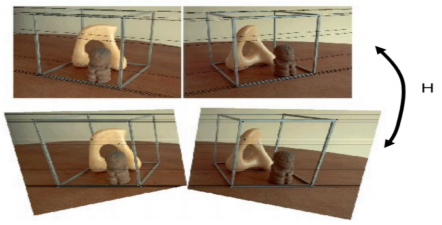


(b) Epipolar Constraint

An interesting case of epipolar geometry is shown in Fig. 4a, which occurs when the image planes are parallel to each other. When the image planes are parallel to each other, then the epipoles e and e' will be located at infinity since the baseline joining the centers O_1 and O_2 is parallel to the image planes. Another important byproduct of this case is that the epipolar lines are parallel to an axis of each image plane,then we arrive at the fact that $v = v'$, demonstrating that p and p' share the same v-coordinate. In the case that the image planes are not parallel, the same result can be achieved when applying a *rectification*[12] process to each of the images.



(a) Parallel Image Planes and The rectification problem



(b) Image Rectification

Figure 4: (a) Computing two homographs and applying them to the image planes to make the resulting planes parallel. (b) In rectified images epipolar lines are horizontal and in vertical correspondence.

3.1.2 Imposing Real World Physical Constraints

Even when applying the *Epipolar Constraint* on stereo images, there can still be ambiguous matches. For example a when finding correspondences in an image of a whiteboard or of a checkered wallpaper. In such scenarios the correspondence problem is under-constraint. Hence, to be able to solve it, heuristics of the physical world must be imposed. It is very important to understand the meaning and characteristics of the following heuristics to determine if and when to apply them when solving the matching problem.

1. **Disparity limits:** To constrain the search, a limit of the disparity value can be set. Then, when searching for a match on the epipolar line, instead of looking along the entire line, the best match is found in a limited range of disparity values.

2. **Appearance:** Features of both stereo images should have similar appearance. For example a table's corner in the left image should have similar appearance in the right one. This assumption is violated when the images taken are of a scene with reflecting surfaces, so the point of view from the stereo cameras might appear as different patterns.

3. **Uniqueness:** A point in one of the images have at most one match in the other image. In other words, a point from one image can only have one of two options: have a single matching point, or it is obscured in the other image. Note that this assumption is valid in images of opaque surfaces.

4. **Ordering:** Features should be in the same left to right order in both images. This assumption might be violated if there is a significant difference in depth between the features in the physical world.

5. **Smoothness:** Objects have mostly smooth surfaces, meaning that disparities should vary smoothly between object edges.

In section 3.2, we shall see in greater depth how these constraints are analysed to form a set of mathematical restrictions and build a SNN model for event based neuromorphic stereo vision system[3].

Now that we established a well defined and more efficient approach to solve the stereo matching problem, we can update Algorithm 1 to conduct a search in a corresponding epipolar line instead of in the whole image:

(a) (b)

Figure 5: Disparity map example. (a) The left image of a rectified pair. (b) The disparity map constructed from the left and right image.

Algorithm 2 Find-Correspondence(Image Img[n,n], src s)

$H \leftarrow MaxHeap$
$Line \leftarrow Img[s.y]$
for each $patch \in Line$ **do**
    $score \leftarrow correlation(patch, s)$
    $H.Insert(key = patch, value = score)$
return $H.getMax()$

3.1.3 Disparity Map/Matrix

Given two calibrated and parallel (or rectified) images denoted as I_l for the left image and I_r for the right image, we can use algorithm 2 to match each pixel from I_l to I_r. since they are on the same epipolar line, they share the same y coordinate. For each matching pixel pair we can extract the disparity value as the difference in number of pixels between the pixel $x_l \in I_l$ to the matching pixel in $x_r \in I_r$. The disparity value is denoted as d and is calculated as in the equation: $d = x_l - x_r$ After doing so for every pixel in I_l, we can color code each disparity value and create a disparity map M_d, as shown in Fig. 5a and Fig. 5b.

3.1.4 Calculate Depth Using Triangulation

There exist neural network based algorithms, that successfully estimate depth in an image. A few examples are [13],[14],[15]. In this section, a geometric approach is taken, where the intrinsic and extrinsic properties of the two cameras are being used to geometrically calculate the depth of a point in the scene. As shown in Fig. 6b, to successfully calculate the depth of a point in

the scene, first the values of these parameters must be available:

1. f - Focal length is the distance between the lens and the image sensor, usually measures in mm.
2. O - The Optical center of the camera sensor.
3. T - The length of the base line, which is the distance between O_l and O_r.
4. P - The point of interest.
5. Z - The depth of Z.
6. x - The x coordinate value.

Note that since p_l and p_r lie on a horizontal line they share the same y coordinate value. Let I_l, I_r be the same stereo images from 3.1.3 and their disparity map M_d is calculated and given. Using similar triangles (see Fig. 6c), we can then calculate the distance Z as follows:

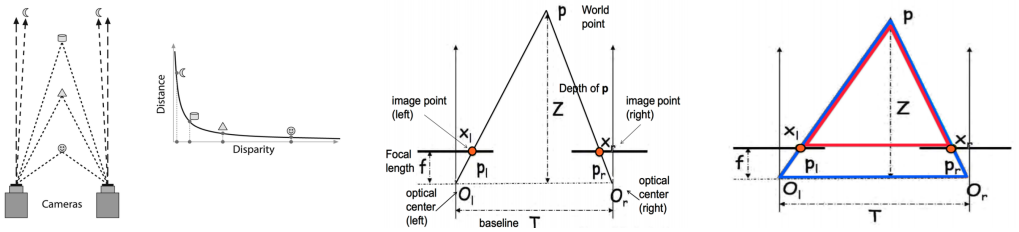
$$\frac{T}{Z} = \frac{T - (x_l - x_r)}{Z - f} = \frac{T - d}{Z - f}$$

$$\longrightarrow ZT - fT = ZT - Zd$$

$$\longrightarrow Z = \frac{fT}{d}$$

The real-world distance from one point to the camera is inversely proportional to disparity value (Fig. 6a). If the disparity value is close to zero then small variations in the disparity generate big variations in depth. On the other hand, if disparity value is big, then small variations in depth do not generate depth differences.

Now that we established the understanding of what is disparity, its properties and the relation between depth and disparity, we can look into the article [3] where a stereo event camera, together with neuromorphic hardware are used, to solve the correspondence problem.



(a) Depth disparity relation. (b) The stereo layout (c) The similar triangles

Figure 6: (a) Depth is inversely proportional to disparity. (b) The Z depth can be computed using similar triangles property. (c) The two similar triangles are marked with red and blue colors.

3.2 A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems

This section presents Osswald et al[3] novel model that solves the stereo-correspondence problem with a spiking neural network that can be directly implemented with massively parallel, compact, low-latency and low-power neuromorphic engineering devices. Rather than capturing static frames and transmitting sequences of frames discretized in time, stereo event camera transmits continuous streams of spikes, or "address-events"[16], which are produced by the individual pixels of each one of the event cameras. As mentioned in 2.2 this occurs when the brightness contrast they sense changes by an amount greater than a set threshold. All of the pixels in the sensor are independent and send their address asynchronously, in continuous time, on a digital bus, as soon as they generate a spike. Similarly, the neuromorphic processors used in the article to run the network, use address-events to receive, process, and transmit spikes.

3.2.1 The Cooperative algorithm

The spiking neural network proposed in the article is inspired by the cooperative network algorithm of Marr and Poggio[4]. Therefor, to be able to understand the operation of the stereo SNN, understanding of the cooperative algorithm itself, firstly, must be achieved. Note that this cooperative algorithm, firstly introduced in 1976, takes as input, static stereo images and not dynamic spatio-temporal visual information in the form of spike trains which are directly obtained from event-based neuromorphic vision sensors. Another difference to note, is that the network introduced by Marr and Poggio[4] itself is not a SNN.

The Marr-Poggio cooperative algorithm
To correctly analyse the depth (or disparity value) at any point in a scene, firstly, the correspondence problem must be solved with high accuracy. In other words, a correct correspondence between a point in one image to a point in the other image, must be found with minimal number of mismatches. To suppress such false correspondences cooperative algorithms can be applied. The naming, cooperative algorithms, is derived from the fact that a set of rules is established and define how the neurons of the network communicate with each other. The purpose of these communication rules is to solve the correspondence problem in a less error prone manner. Since the neurons are able to measure disparities by applying these rules, they are called disparity sensitive neurons (DSN)[17].

According to Marr and Poggio [4], there are three steps in measuring stereo disparities:

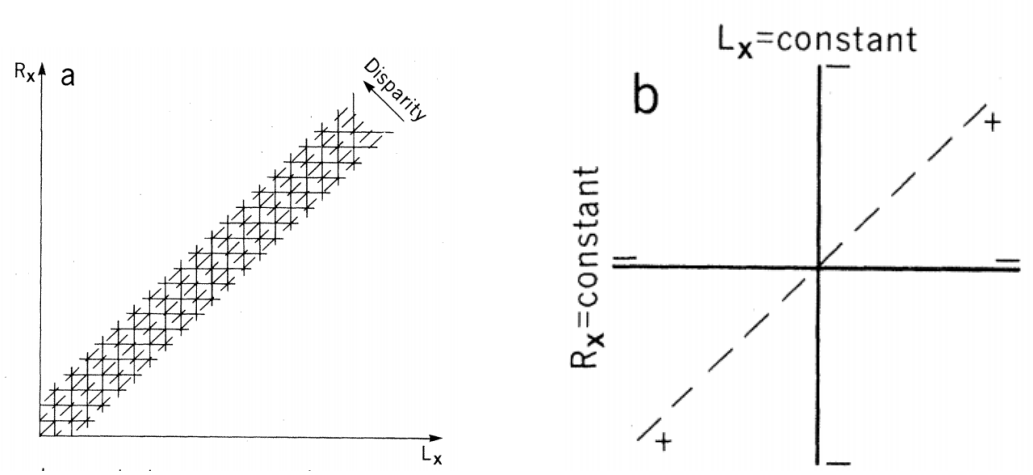
- (S1) Determination of the point of interest (POI) in the first image.
- (S2) Identification of the POI in the second image.
- (S3) Measurement of the disparity of the two pixels.

To minimize the number of wrong correspondences, the physical properties of solid bodies, two of the properties, also mentioned in 3.1.2, are considered in order to obtain additional constraints. These are the following two constraints:

- **(C1)Uniqueness:** A given point on a physical surface has a unique position in space at any one time.
- **(C2)Smoothness:** Matter is cohesive, it is separated into objects, and the surfaces of objects are generally smooth compared with their distance from the viewer[4][18].

The two physical constraints C1 and C2 are translated into conditions on a computation process. The two rules are:

- (R1) The Uniqueness Rule: Derived from C1. Each item from each image, may be assigned at most one disparity value [4][18].
- (R2) The Continuity Rule: Disparity varies smoothly almost everywhere. This condition is a consequence of the cohesiveness of matter, and it states that only a small fraction of the area of an image is composed of boundaries that are discontinuous in depth.[4][18].



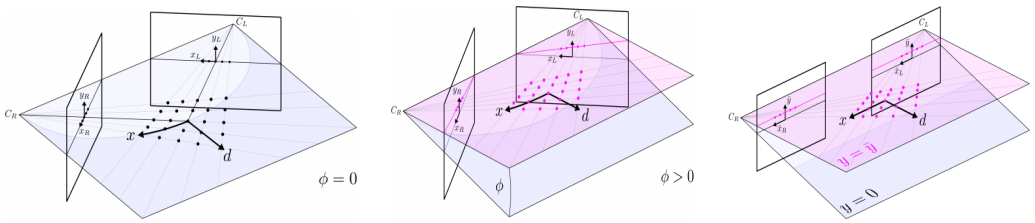
(a) The structure of the two rules (b) Lines of Inhibition and Excitation

Figure 7: (a) Structure of the two rules R1 and R2 for the case of one dimensional image. This is also the structure of the network for implementing the algorithm where solid lines represent inhibitory interactions, and dashed lines represent excitatory ones. (b) The local structure of each DSN in the network.

The algorithm is derived of these two rules, Fig. 7a exhibits their geometry in the simple case of a one dimensional image or a row in a 2D one. The L_x and L_y axis represent the positions of the elements on the left and right image. The horizontal and vertical lines represents the lines of sight from each visual sensor and their intersection point corresponds to possible disparity values. The dashed diagonal lines connect points of constant disparity. R1 states that a single disparity value may be assigned to each element. Constructing a network out of the lines in Fig. **??** by positioning a DSN in each intersection, means that only one neuron should be switched on along each horizontal and vertical line. R2 states that disparity values vary smoothly almost everywhere, which means that solution should spread along diagonal lines. Applying these two rules on the network is done by introducing competition and cooperation (Fig. 7b). Competition - at each possible match, a DNS is turned on and inhibits other DSNs on the same horizontal and vertical lines. Cooperation - The same DSN excites other DSNs, located on its disparity line. Such a network converges to a solution that obeys the two rules.

3.2.2 The spiking stereo neural network model

This section provides an explanation of the spiking stereo neural network.



(a) (b) (c)

Figure 8: The coordinate system of the network and representation of disparity space. (a) The general sensor arrangement, showing the epipolar plane that is perpendicular to the image planes. (b) The same sensor arrangement, showing an inclined epipolar plane. (c) The same sensor arrangement with rectified image planes.

The coordinate system of the network

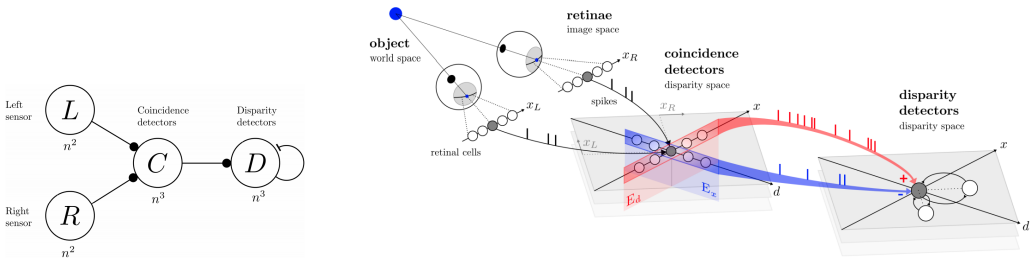
Similar to place cells in rats, each individual neuron in the network acts as a cognitive representation of a unique location in 3D space. The input from each of the event cameras (ECs) in the network is statically mapped to the the networks neurons.

In Fig. 8a a general stereo setup is shown comprising two sensors located at C_R and C_L and their epipolar planes (as seen in 3.1.1), each with its own image coordinates (x_R, y_R) and (x_L, y_L) respectively. All points lying on an epipolar plane project to exactly one line in each image, the *epipolar line*. Different epipolar planes are represented according to their inclination ϕ. The shaded blue plane indicates the horizontal epipolar plane as $\phi = 0$. The five image points from each camera and their 25 correspondences form a distorted array. Within this array, indices x and d along the diagonals are introduced. The indices are directly derived from the image coordinates:

$$x = x_R + x_L$$

$$d = x_R - x_L$$

where x is referred to as the *horizontal cyclopean coordinate* and d as the *disparity coordinate*. each epipolar plane form a layer of the network and each neuron is uniquely described as the triplet (x, ϕ, d). In Fig. 8b, the inclined epipolar plane ($\phi > 0$) adds complexity to the challenge of finding correspondences along the inclined epipolar lines. This can be solved using EC that share the same plane or by rectifying the images, as reflected in Fig. 8c. The image points are then expressed in rectified coordinates, ($\bar{x}_R$, $\bar{y}_R$) and ($\bar{x}_L$, $\bar{y}_L$) As a result of the rectification the epipolar lines are horizontal and have the same vertical coordinate : $\bar{y} = \bar{y}_L = \bar{y}_R$. A unique map $\mathbb{M}$ can then be derived, which is invariant to sensor pose, and which assigns a



(a) (b)

Figure 9: The spiking stereo network. (a) Abstract view of the network's architecture. (b) Detailed view of a horizontal layer of the network.

neuron to each epipolar pair of rectified image coordinates. This neuron can be described by the triplet (x, y, d):

$$\begin{aligned} \mathbb{M}: \quad & \mathbb{N}^3 \longrightarrow \mathbb{D}^3 \\ & (\bar{x}_L, \bar{x}_R, \bar{y}) \longrightarrow (x, y, d) = (\bar{x}_R + \bar{x}_L, \bar{y}, \bar{x}_R - \bar{x}_L) \end{aligned}$$

Each neuron is uniquely assigned a horizontal and vertical cyclopean coordinate x and y, as well as a disparity coordinate d. Together, these coordinates represent a point in disparity space $\mathbb{D}^3$, which corresponds to the neuron's cognitive representation of a location in 3D space.The absolute-world coordinates of this location are determined by the intersection of the lines of sight from the pair of rectified image points, which can be derived from the network coordinates by means of the inverse mapping function $\mathbb{M}^{-1}$.

The architecture of the network

An abstract view of the entire network architecture is given in Fig. 9a, and a detailed view of a horizontal layer of the network is illustrated in Fig.9b. The SNN extends the classical cooperative network by applying mechanisms of stereo correspondence based on temporal and spatial compliance. The retinal cells (or pixels of the EC sensors) are represented by the populations L and R and serve as the input to the network. Their size is indicated by n^2, as the sensors consist of two-dimensional pixel arrays. L and R excite a population of neurons C, referred to as the "coincidence detectors". The size of the population C scales cubically because for each pair of corresponding horizontal lines of retinal cells from L and R, a horizontal layer of neurons in C signals temporally coinciding spikes. Each C cell has a unique spatial representation in disparity space (x, y, d) (only x and d are shown in the figures) such that each spike provides evidence for a potential target at the corresponding real-world disparity position. Lastly, another population

of neurons D, termed the "disparity detectors", pool responses from C in mixed excitatory and inhibitory manner. The disparity detectors implement a binocular correlation mechanism, which is realized by integrating the responses from coincidence detectors within the planes of constant disparity E_d and cyclopean position E_x. Activity in E_d constitutes supporting evidence for true matches (excitation of disparity detector), whereas activity in E_x denotes countervailing evidence (inhibition of disparity detector). Finally, the uniqueness constraint is enforced by a winner-takes-all mechanism of mutual inhibition of disparity detectors that represent spatial locations in the same line of sight.

Simple Coincidence Detectors

The model implements a coincidence detection mechanism using neurons with leaky integrate and fire (LIF) dynamics. This following equation describes a LIF coincidence neuron membrane potential $v_c(t)$ where τ_c determines the neuron's leak and θ_c the firing threshold.

$$\tau_c \frac{dv_c(t)}{dt} = -v_c(t) + I_c(t), \qquad v_c(t) < \theta_c$$
$$v_c(t) = 0, \qquad v_c(t) \geq \theta_c$$

Each neuron receives input from a pair of epipolar retinal cells which can be described as a sum of spikes as described in the following equation where a spike is modeled with the Dirac function $\Delta(t)$ and w is the synaptic weight. i, j indicate the spike times of the retinal cells $(\bar{x}_L, \bar{y})$ and $(\bar{x}_R, \bar{y})$respectively. The subscript vector c corresponds to the neuron's unique spatial representation in disparity space using the mapping function $\mathbb{M}$.

$$I_c(t) = w \sum_i \delta_{\bar{x}_L}(t - t_i) + w \sum_j \delta_{\bar{x}_R}(t - t_i) \quad | \quad c = \mathbb{M}(\bar{x}_L, \bar{x}_R, \bar{y}) = (x_c, y_c, d_c)$$

In order to have the neuron fire when receiving two retinal spikes and not by a single one, assuming that spikes from the same retinal cell are temporally well separated, the researchers introduced a temporal sensitivity factor denoted as $S_{\Delta T}$. The sensitivity of a coincidence detector is the range of inter-ocular temporal delays between two spikes from an epipolar pair of retinal cells within which the neuron is responsive. Setting the sensitivity too high could cause long inter-ocular temporal delays, which will increase the number of coincidences associated with ambiguous disparities. On the other hand, if the sensitivity is too low, the beneficial effect of global support

from distant targets with varying disparity will be reduced. The sensitivity is settable by adjusting the parameters τ_c and $\frac{\theta_c}{w}$ as in the equation:

$$S_{\Delta T} = \tau_c \ln(\frac{1}{\frac{\theta_c}{w} - 1}) \ , \qquad 1 < \frac{\theta_c}{w} \leq 2$$

Complex disparity detectors

The disparity detectors aggregate evidence from the responses of simple coincidence detectors. A disparity neuron is also modeled using LIF neuron dynamics but with a distinct time constant τ_d and a firing threshold θ_d. The time constant of the disparity neurons determines how evidence from the past is weighted. Making an appropriate choice of τ_d is dependent on the stimuli but in general, it is significantly larger than τ_c.

$$\tau_d \frac{dv_d(t)}{dt} = -v_d(t) + I_d(t), \qquad v_d(t) < \theta_d$$
$$v_d(t) = 0, \qquad v_d(t) \geq \theta_d$$

The input of the disparity detector at $d = (x_d, y_d, d_d)$ combines the outputs from coincidence detectors within bounded planar excitatory and inhibitory regions in disparity space $C^+ \in \mathbb{D}^2$ and $C^+ \in \mathbb{D}^2$ respectively, where k represents the index of the spike times of coincidence neuron c, while w_{exc} and w_{inh} are constant excitatory and inhibitory weights, since the size of C^+ and C^- are of equal size the researchers suggested that $w_{exc} = w_{inh}$.

$$I_c(t) = w_{exc} \sum_{c \in C^+} \sum_k \delta_c(t - t_k) - w_{inh} \sum_{c \in C^-} \sum_k \delta_c(t - t_k)$$

The regions C^+ and C^- are squared windows in the plane of constant disparity E_d and the plane of constant horizontal cyclopean position E_x (Figure **??**) , which are defined relative to the disparity detector's spatial representation d.

$$C^+ = \{c \in C | (|x_c - x_d| \leq w) \wedge (|y_c - y_d| \leq w) \wedge (d_c = d_d)\}$$
$$C^- = \{c \in C | (x_c = x_d) \wedge (|y_c - y_d| \leq w) \wedge (|d_c - d_d| \leq w)\}$$

Mutual inhibition of disparity detectors

Computing the spatio-temporal cross-correlation of inter-ocular temporal images would result in the correct disparity value and should, ideally be calculated in a disparity neurons. However, cross-correlation is a complex task that involves the calculation if the co-variance followed by normalization. The disparity neurons described in the prior section do not perform a normalization operation. Therefor, calculating the disparity is based on a non-normalized, co-variance-like measure. This non normalized value creates a difficulty for setting the firing threshold of a disparity LIF neuron, θ_d. This can be easily observed by comparing the case of a single coincidence in C^+ with the case of multiple coincidences in C^+. Assuming there are no coincidences in C^- in both cases, the latter results in a stronger response while the cross-correlation coefficient would be ideal ($\rho = 1.0$) in both scenarios. If θ_d is set too low. disparity detectors are more likely to respond to false but similar targets, particularly at locations where the targets are large or rapidly moving (many coincidences), even in the presence of inhibiting evidence. In contrast, if θ_d is set too high, although the sensitivity to false disparities is reduced, the response to true disparities associated with small or slowly moving targets is also diminished (fewer coincidences). This suggests that θ_d is context-dependent and a critical parameter.

The network addresses this problem by implementing the uniqueness constraint, first described by [4] and reviewed in 3.2.1. Give a false disparity detecting neuron, the correct disparity is located somewhere along the line of sight and is simultaneously represented by another neuron. Ideally, this neuron integrates more coinciding evidence, thus evoking a faster response. This response is then recurrently fed as an inhibitory input into the neuron located at the false disparity in order to suppress its response. This winner-take-all mechanism is implemented with mutual inhibition, such that each disparity detector inhibits all the other neurons in its line of sight. Thus, the disparity LIF neuron equation is updated by subtracting an inhibitory current $I_{d-}(t)$ accordingly:

$$\tau_d \frac{dv_d(t)}{dt} = -v_d(t) + I_d(t) - I_{d-}(t), \quad v_d(t) < \theta_d$$
$$v_d(t) = 0, \quad v_d(t) \geq \theta_d \vee v_d(t) < 0$$

The inhibitory current is defined as:

$$I_{d-}(t) = w_{rec} \sum_{d \in D^-} \sum_n \delta_d(t - t_n)$$

and the inhibitory region D^- is defined by the two lines of sight (one from each retina):

$$D^- = \{d \in D | (x_d - d_d = 2\bar{x}_L) \vee (x_d + d_d = 2\bar{x}_R)\}$$

Setting the recurrent synaptic weight greater or equal to the disparity threshold $w_{rec} \geq \theta_d$ results that each spike that signals a correct disparity completely resets the membrane potential of the other neurons in the same line of sight, D^-. This action prevents neurons at false disparities from gaining a head start when competing to signal the next correct disparity after a spike.

Representation and coding of disparity

Due to the network architecture, a disparity detector neuron, placed at each spatial location in $\mathbb{D}^3$, have strongly overlapping receptive fields with its neighboring neurons and relates to the size of the receptive window in C. This causes disparity detectors to also respond to targets which are at their preferred disparity, but distant from their preferred cyclopean position.This means that such cells respond equally to any target within their receptive fields, regardless of its exact position which corresponds to a blurry dynamic disparity map. However, an accurate disparity map can be obtained by combining the output from coincidence and disparity detectors. Coincidence detectors encode all possible disparities (corresponding to true and false targets).Thus, if a disparity neuron spikes, the simultaneous responses of coincidence detectors that have an equal (or nearby) spatial representation in $\mathbb{D}^3$ reveal the correct and accurate disparities. In other words, the final output of the network is obtained from a combination of coincidence and disparity responses such that a disparity spike is only considered valid when immediately preceded by a coincidence spike representing the same location in disparity space.

The notion of events is useful to express the disparity response of the network: $e_d^+ = (\mathbf{d}, t)$ is a *unipolar disparity event* that represents a relative change in light intensity, that occurred at time t, at location $\mathbf{d}$, in $\mathbb{D}^3$. $e_c^+ = (\mathbf{c}, t)$ is, similarly, a *unipolar disparity event* that represents a relative change in light intensity, that occurred at time t, at location $\mathbf{c}$, in $\mathbb{D}^3$. $C^+ = \{e_c^+\}$ and $D^+ = \{e_d^+\}$ are the sets of all coincidence and disparity events respectively. The final output of the network is then defined as the set of *filtered unipolar disparity events*:

$$O^+ = C^+ \cap D^+$$

The network is extended by also considering the sign of intensity change, which is encoded using an additional event attribute, the polarity $s \in$ [-1, 1]. A pair of coincidence detectors, one for each polarity, is used at each location in disparity space, similar to the simple ON and OFF cells in V1. Thus, the coincidence event is redefined as: $e_c = (\mathbf{c}, s, t)$. Disparity detectors are polarity indifferent, however, the filtered disparity events inherit the polarity of the coincidence events, which results in a set O, that encodes spatially

precise and unambiguous, polarized light intensity changes in disparity space:

$$O = \{e_c \in C | [e_c]^+ \in D^+\}$$

where $[]^+$ denotes a polarity rectification that transforms the polarity event $e_c = (\mathbf{c}, s, t)$ to a unipolar event $e_c = (\mathbf{c}, t)$.

3.2.3 Results and model evaluation

The main results presented in this Section originate from numerical simulations of the stereo network performed on a standard desktop computer. The spike-based visual data used as input to the network was either obtained through real recordings made with neuromorphic silicon retinas or was synthesized in artificially, depending on the type of experiment performed.

Ground-truth evaluation method and error definition

To evaluate a complex dynamic scene, an accurate measurement of the ground truth 3D data must be acquired. The Microsoft Kinect[19] uses structured light based 3D data acquisition technique and was selected to be used in the evaluation experiments due to its reliability. The data obtained from the Kinect was recorded and translated in respect to the event stereo camera location. Matched events were triangulated and binned into $30ms$ time slices. In so doing, the reconstructed 3D point clouds are sequentially compared to the ground-truth point clouds, which have been acquired and temporally synchronized (obtained from one frame of the Kinect running at 30 Hz). The depth error ϵ_Z is the distance from a reconstructed 3D point to its nearest neighbor in the ground-truth data, along a line of sight determined by the position of the 3D point and the center of the camera. The disparity error ϵ_d can then be computed from the depth error:

$$\epsilon_d = \frac{bf}{Z^2}\epsilon_Z$$

The metric to evaluate the performance of stereo matching is based on the **percentage of correct matches (PCM)**, which is defined as:

$$PCM = \frac{1}{N_t}\sum_{i=1}^{N_t} h(e_{3D}^i) \quad with \quad h(e_{3D}) = \begin{cases} 1, & \epsilon_d \leq 1 \\ 0, & \epsilon_d > 1 \end{cases}$$

Network simulation
The input to the network simulation was the output of 180×180 pixels from each DVS. This led to a population of 2×180^3 coincidence detectors (polarities where treated with separate detectors). With another 180^3 disparity detectors, the total network initially incorporated more than 17 million neurons. However, for a natural scene within 5 meters distance from the camera, the disparity is limited to the range (0, 40) so the total number of required neurons was reduced to less than 4 million. The simulation code was entirely written in C and in a completely event-based fashion, meaning that the membrane potential of a specific neuron was only updated when it received an input leading to a very efficient implementation. For each neuron the current membrane potential and the time of the last update needed to be stored in memory ending up in an occupancy of about 30 megabytes. In the 4 seconds walking scene, roughly 1.2 million input events were processed with a total simulation run time of about 30 seconds on a single core of an i7 CPU running at 3.40 GHz.

Stereo matching performance
The network's matching performance was assessed in three different dynamic scenes involving a moving face(1), two people moving in different directions(2) and a person performing martial arts(3).The results are qualitatively shown in Fig. 11, Fig. 12 and Fig. 13 respectively. The distribution of disparity errors fits very well a half-normal distribution as shown in the histograms in Fig. 10.

Generally, it can be observed that the stereo network performs equally well for all scenes. Scene 2 outperforms the other two scenes, this is likely to do with the fronto-parallel movement of the two people. The relatively constant and low disparity error throughout all the scenes suggests that the matching performance is very robust and largely independent of how fast the objects move (illustrated by the event rate). Tab. 2 shows the PCM computation results of all scenes. A match is considered true if the associated 3D event had a disparity error of less than δ pixels and a PCM of $\delta = 1$ is the standard. $\delta = 2$ and $\delta = 3$ were calculated and found to be higher respectively, indicating that although the majority of falsely matched targets are inaccurate, they are roughly correct. Tab. 3 lists the scene statistics and the averaged results.

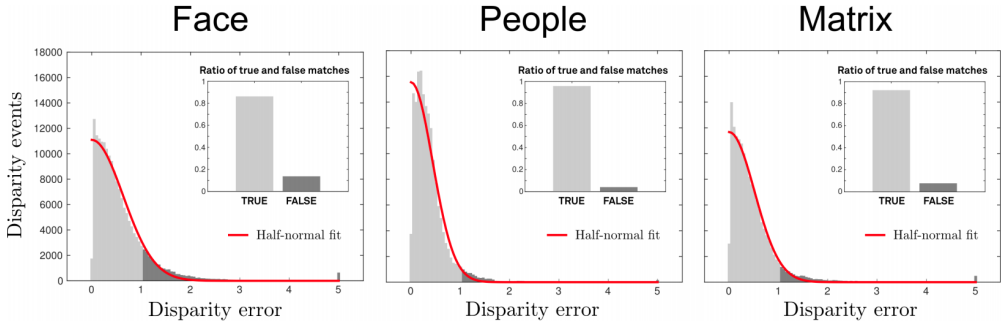


Figure 10: Disparity error histograms for all three scenes. Events corresponding to true matches (disparity error $\leq$ 1 pixel) are shaded light gray. All disparity errors that are greater than 5 pixels are contained in the last bin. A half-normal distribution was fitted to each of the histograms (red curve). The insets show the ratio of true and false matches. The "people" scene shows the best observable performance, evidenced by the ratio of true and false matches and the narrowness of the distribution of disparity errors.

Table 2: Summary of results.

Scene	PCM ($\delta = 1.0$)	PCM ($\delta = 2.0$)	PCM ($\delta = 3.0$)
Face	86.3%	97.7%	99.2%
People	95.9%	99.6%	99.9%
Matrix	92.3%	98.7%	99.5%
Average	**91.5%**	**98.7%**	**99.5%**

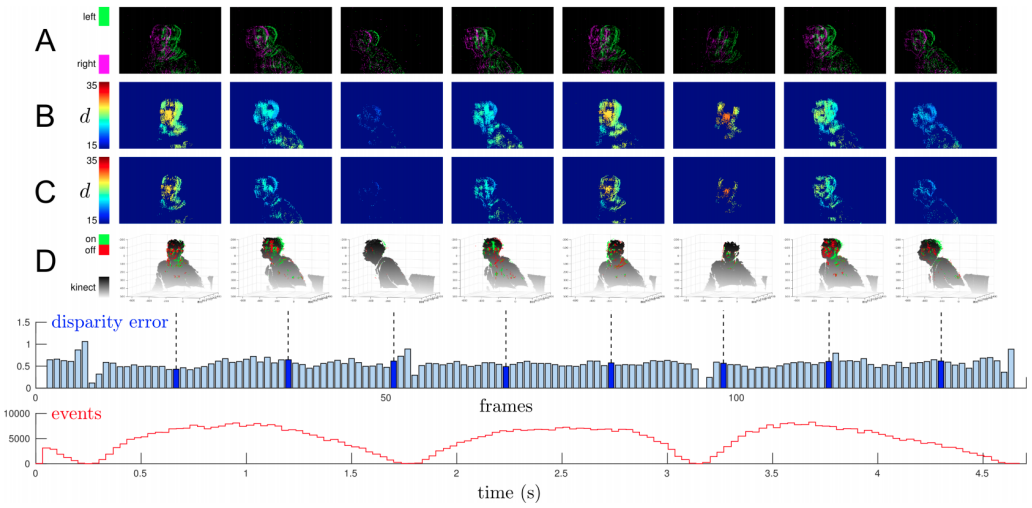


Figure 11: Qualitative and quantitative results of scene (1). This scene comprises a face that first rotates to the right (from its point of view) followed by a rotation to the left while moving towards the cameras. Finally, it rotates again rightwards and recedes. The scene lasts 5 seconds. At the bottom, the disparity error and input event rate are plotted over time. (A) Combined frames of accumulated input from the left (green) and right (purple) sensors. (B) Disparity maps generated from accumulated unfiltered disparity events. (C) Disparity maps generated from accumulated filtered disparity events. (D) 3D events generated by triangulating disparity events, whose polarity is color-coded in green (ON) and red (OFF). The ground-truth point cloud which was obtained from the Kinect is also shown (gray).

Table 3: Scene statistics and experimental results.

	Scene statistics					Averaged results				Fit	
Scene	T	n_l	n_r	n_c	n_d	$\bar{\varepsilon}_d$	s.d.	$\bar{\varepsilon}_Z$	s.d.	μ_d	σ_d
(1) Face	4.5	685k	518k	3'538k	699k	**0.64**	**2.04**	18.82	24.13	**0.50**	**0.63**
(2) People	4	666k	611k	3'250k	766k	**0.37**	**0.42**	85.77	110.08	**0.35**	**0.44**
(3) Matrix	3.0	557k	430k	3'174k	582k	**0.53**	**1.91**	32.57	32.38	**0.41**	**0.51**

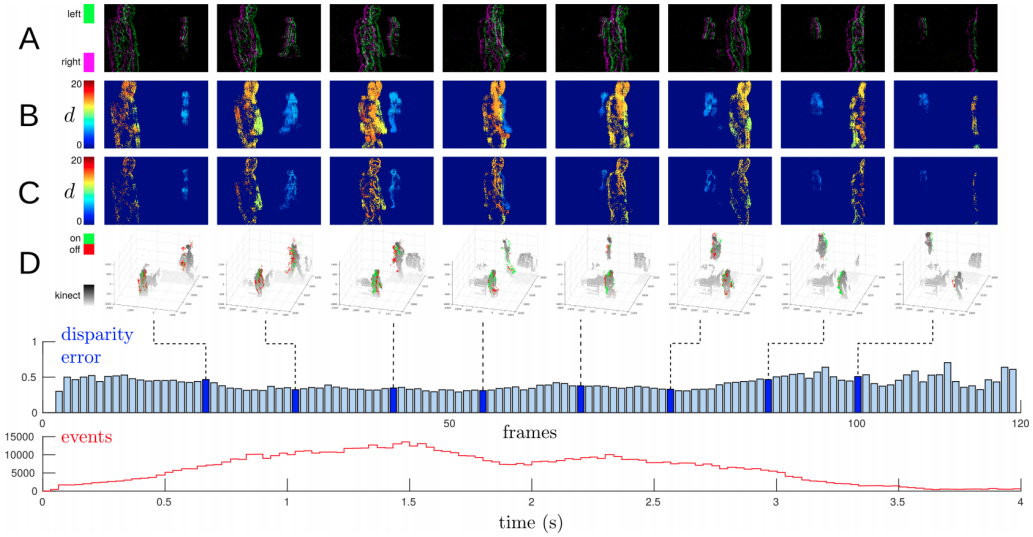


Figure 12: Qualitative and quantitative results of scene (2). This scene comprises two people that are walking in opposite directions at different depths. The scene lasts 4 seconds. At the bottom, the disparity error and input event rate are plotted over time. The scene features large disparity gradients (at the point when the people cross each other) and is therefore considered a difficult matching problem. As can be seen from the disparity maps located in the middle, the network unambiguously detects both people. This is explained by the strong motion cues that are present in this particular scenario. (A) Combined frames of accumulated input from the left (green) and right (purple) sensors. (B) Disparity maps generated from accumulated unfiltered disparity events. (C) Disparity maps generated from accumulated filtered disparity events. (D) 3D events generated by triangulating disparity events, whose polarity is color-coded in green (ON) and red (OFF). The ground-truth point cloud which was obtained from the Kinect is also shown (gray).

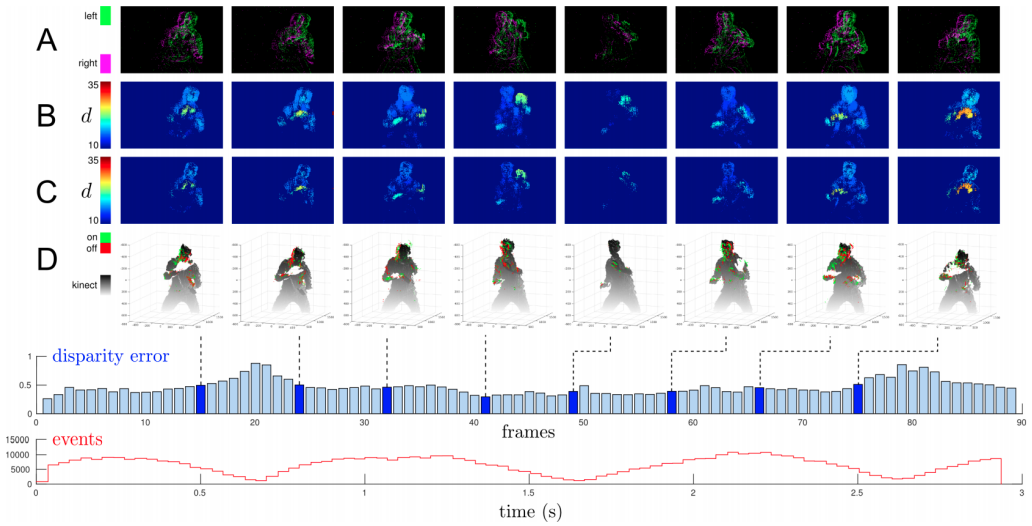
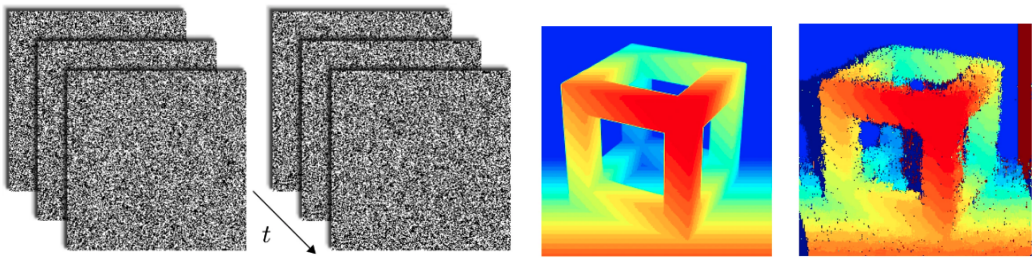


Figure 13: Qualitative and quantitative results of scene (3). This scene comprises a person performing martial arts. The scene lasts 3 seconds. At the bottom, the disparity error and input event rate are plotted over time. This scene combines large disparity gradients and various motion cues. Although the hands are partially moving forwards and backwards, the network is still capable of solving the correspondence problem. Here, it can be observed that the disparity error slightly increases when the person moves more slowly (decreased event rate). This is because there are no motion cues at these points, which are required to correctly match large disparity gradients. (A) Combined frames of accumulated input from the left (green) and right (purple) sensors. (B) Disparity maps generated from accumulated unfiltered disparity events. (C) Disparity maps generated from accumulated filtered disparity events. (D) 3D events generated by triangulating disparity events, whose polarity is color-coded in green (ON) and red (OFF). The ground-truth point cloud which was obtained from the Kinect is also shown (gray).

Response to dynamic random dot stereograms

Random-dot stereogram (RDS) is stereo pair of images of random dots which when viewed with the aid of a stereoscope, or with the eyes focused on a point in front of or behind the images, produces a sensation of depth, with objects appearing to be in front of or behind the display level [20]. A dynamic RDS was made of a sequence of RDS (examples of three subsequent RDS are shown in Fig. 14a), updated in 100Hz. The initial RDS image was computed based on a disparity image of a wireframe cube (Fig. 14b) and a random noise pattern, both of size 250×250 pixels. In order to make sure that solving the matching problem is not trivial, subsequent RDS images were generated from the previous one in a way that there was a 20% chance that each pixel would change its polarity. Assuming that even if the update rate is high, the coincidence detectors are tuned for very short temporal delays (such that they only respond to coincidence within a single RDS and not in-between



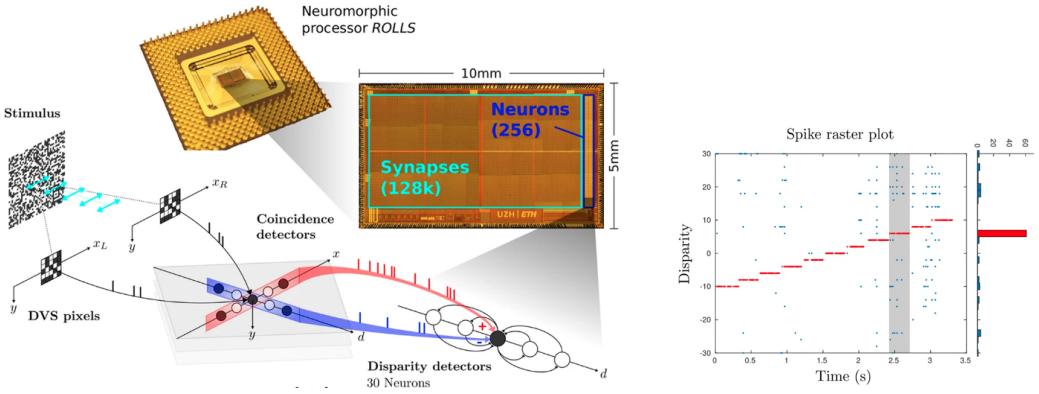
(a) (b) (c)

Figure 14: The stereo network's response to a dynamic RDS. (a) Schematic of the dynamic RDS stimulus for the left and right eye. (b) Ground-truth disparity image. Disparity is encoded by color ranging from near (red) to far (blue). (c) Disparity map generated from accumulated responses of the network while the dynamic RDS stimulus was presented for one second.

consecutive RDS images). The final response of the network is illustrated in the form of an accumulated disparity map as seen in Fig. 14c. Based on a qualitative comparison between the disparity map and the ground-truth data, the proposed network solves the stereo correspondence problem, even for a relatively complex dynamic RDS containing highly varying disparity gradients.

3.2.4 Neuromorphic hardware implementation

To validate the model with a neuromorphic hardware platform the researchers used the Reconfigurable On-Line Learning Spiking (w) neuromorphic processor device described in [21]. This device comprises an array of 256 analog Integrate-and-Fire neuron circuits, and 512×256 dynamic synapses. The time-constant of the decaying exponential can be set by a corresponding time-constant parameter and the synaptic weight is settable and parameterized as well. Programmable digital circuits embedded in the arrays allow the implementation of recurrent or multi-layer networks. The experimental setup is illustrated in Fig. 15a. An RDS stimulus was printed on a chart and moved across multiple depths in a fronto-parallel plane to the rectified stereo setup comprising the two Dynamic Vision Sensors. The pixels of the two vision sensors were grouped to form 30 binocular 21×21 (smaller subsequent RDS) receptive fields and spread along their central epipolar lines. These inputs were projected in parallel to the coincidence detector units. To implement in neuromorphic hardware the part of the model that is composed of coincidence detection units it would be necessary to use devices with larger numbers of neurons, which were not accessible to the researchers, so they



(a) (b)

Figure 15: Emulation of the stereo network on a neuromorphic processor.(a) A RDS stimulus (printed on a chart) was moved at specified depths in front of a pair of dynamic vision sensors. The depth of the stimulus was detected by 30 disparity neurons covering an equally spaced disparity range from -10 to +10. Disparity detectors were emulated by a neuromorphic processor. (b) Spike event output of the disparity neurons during stimulus presentation. The spikes from the neuron encoding the true disparity are colored red, while all the others are colored blue. The histogram shown on the right represents the distribution of disparity spikes for the time where the stimulus was at a fixed disparity d = 6 (indicated by the grey shaded region).

implemented the coincidence detector neurons using a Field Programmable Gate Array (FPGA) device.The output of these coincidence detector neurons were then sent to the silicon neurons on the ROLLS neuromorphic processor in a way to encode 30 disparities equally distributed along a line of constant cyclopean position. Mutual inhibition among disparity neurons was implemented using on-chip recurrent inhibitory connections. The stimulus was presented at equally spaced disparities ranging from -10 to +10. The response of this stereo vision systems is shown in the form of a spike raster plot in Fig. 15b. The spikes from the neuron encoding the true disparity are colored red, while all the others are colored blue. The network successfully detects the correct disparity of the stimulus in the entire range, as indicated by the red spike trains.

3.2.5 Comparing stereo correspondence performance of traditional and neuromorphic hardware

Conventional SAD based stereo correspondence algorithms typically involve many steps. However, the step that dominates the algorithm complexity is

the calculation of the sum of absolute differences at location x,y.

$$SAD(x, y, d) = \sum_{(i,j)\in W} |I_1(x + i, y + j) - I_2(x + i + d, y + j)|$$

Where I_1, I_2 are the two stereo images, d is the possible disparity shift measured and W is the window of pixels.The number of SAD operations grows with the input data rate (i.e., the camera frame-rates), irrespective of the scene contents. For comparison, a classical stereo vision system involving local stereo matching based on SAD operation assumed to be implemented on a state-of-the low power art micro-controller, such as the STM32F7 series.

In the stereo network model, the equivalent operational primitive is a coincidence detection followed by a spike integration operation (only when there are coincidences). The number of these operations depends on the response of the DVS to the contents of the scene. Considering the input to be moving RDS, similar to the scene described in Fig. 14a, the number of operations corresponds to the amount of input coincidence spikes from the DVS. This number was calculated and by also using the estimated edge density of the scene the researchers estimated the temporal resolution at which the two approaches should be compared to be 151Hz. Tab. 4 depicts the difference between the numbers of operations made with the two different implementations and provides an estimate of the power budget based on energy measurements of these primitive operations. The most significant observation is that a micro-controller implementation of a SAD stereo vision algorithm run at a temporal resolution that is comparable to the one used by the stereo SNN model (i.e., at 151Hz on ROLLS) would consume approximately an order of magnitude more power. Note that the Tab. 4 also shows how using more advance 28 nm (compared to the ROLLS that is based of 180 nm) CMOS technology[22] significantly decreases the cost of energy per operation.

Table 4: Estimated power consumption of classical hardware implementation versus neuromorphic hardware implementation.

	Update rate	Operations/s	Energy per operation	Est. power consumption
SAD	30 Hz	397 K	0.99 nJ	393 μW
SAD	151 Hz	2.00 M	0.99 nJ	1.99 mW
SSNN (180 nm)	151 Hz	147 K	1.243 nJ	182 μW
SSNN (28 nm)	151 Hz	147 K	0.197 nJ	29 μW

3.3 Summary

The reviewed article described in great details the neuromorphic stereo correspondence model. The cooperative algorithm [4], sets the foundation of the model implementation, where the use of a stereo event camera instead of conventional one, aids in solving the correspondence problem. In benefit of the EC advantages, the matching problem is solved with a low power consumption, high dynamic range, and more innovating is the introduction of the time domain as an additional matching criteria. That means that correspondence between a left and right spiking neurons or retinal cells, is defined and calculated not only by approximating the co-variance of the coincidence neurons in a spacious receptive field, but also by filtering out pairs of neuron that are not temporally correlated. Mapping the model onto a neuromorphic hardware validated the model compatibility with a neuromorphic hardware platform. To do so the ROLLS[21] and an FPGA were programmed accordingly. Power consumption wise, when implementing the network on the ROLLS vs. on a micro controller, estimated to be an order of magnitude more power efficient.

4 Neuromorphic PID Robot Controller

Today's robotics is not properly adapted or does not offer specific solutions for neuromorphic systems[23]. Available products in the market provide motor controllers such as black boxes, which receive a reference command for targeting a position of a joint or a revolution speed. These controllers communicate with each other through industrial field buses, such as Controller-Area-Network (CAN), which introduces extra latency in the control loop and forces a fixed power consumption. These systems never provide direct access to the signals that drive the motors of the robots. Typically, these motors are driven by digital circuits using pulse-width-modulation (PWM), which imposes a constant power consumption even when the joint is not moving. Bio-inspired motor control spiking systems aim to reduce this power consumption by reducing the signal activity that drives the motors through the application of a Pulse Frequency Modulation (PFM) technique[6]. In this section the work of [5] is presented, where a neuromorphic, low power consumption approach is taken to implement a spike based proportional–integral–derivative (PID) controller, denotes as sPID. In Addition, this article introduces a fully neuromorphic robotic arm (from spiking sensors to actuators), called event-driven BioRob (ED-BioRob). This work was also the first work to represent the motor control of a robotic arm in the spike-domain using PFM.

4.1 Classical PID controller

To fully grasp the spiking version of a sPID controller, firstly, understanding of the operation of a classical one must be established. A PID controller is a control loop mechanism employing feedback. A PID controller continuously calculates and error $e(t)$ as the difference between a desired setpoint (SP) (also named as a reference point) and a measured process variable (PV) and applies a correction based on proportional, integral and derivative terms[24]. Fig. 16 shows a block diagram of a PID controller, where $r(t)$ is the desired reference or setpoint $SP = r(t)$ and a measured process variable $PV = y(t) \mid e(t) = r(t) - y(t)$ and applies a correction based on the proportional integrative and derivative terms to minimize the error overtime by adjustment of the controller output $u(t)$, to a new value determined by a weighted sum of the control terms. The final form of the PID algorithm is:

$$u(t) = MV(t) = K_p e(T) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where K_p is the proportional gain, a tuning parameter, K_i is the integral gain, a tuning parameter, K_d is the derivative gain, a tuning parameter, $e(t)$

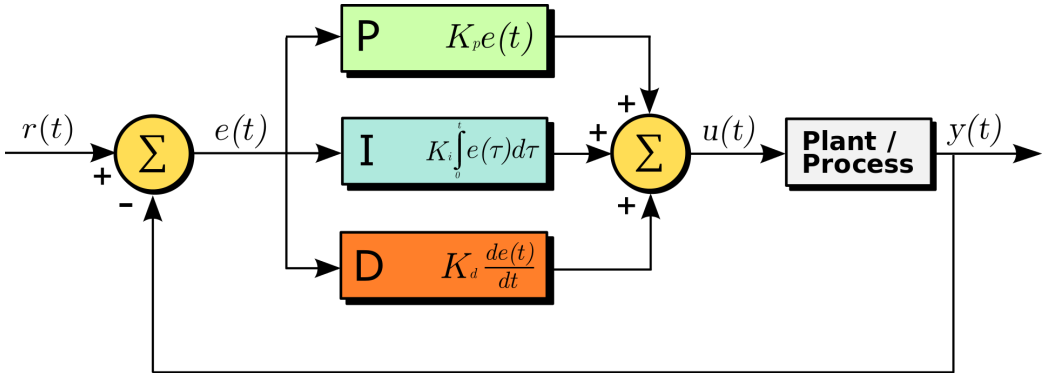


Figure 16: A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).

is the error , t is the time or instantaneous time (the present), τ is the variable of integration (takes on values from time 0 to the present t).

The transfer function of the PID controller in the laplace domain is:

$$PID(s) = \frac{X(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

where s is the complex frequency. Fig 17 shows the response of a PID controller to a step function.

4.1.1 Proportional term

The proportional term, denoted as P_{out}, produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant. $P_{out} = K_p e(t)$. A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller.

Steady-state error

Steady-state error (SSE) is defined as the difference between the the desired setpoint ($r(t)$) and the output of a system ($y(t)$) in the limit as time goes to infinity (i.e. when the response has reached steady state). A proportional controller generally operates with a SSE because a non-zero error value requires to drive it. In other words, the reference value is never achieved because as the error approaches zero, so too does the applied correction. SSE

may be mitigated by adding a compensating bias term to the setpoint and output or corrected dynamically by adding an integral term.

4.1.2 Integral term

The integral of a PID controller is the sum of the error $e(t)$ over time. The accumulated error is then multiplied by the integral constant gain K_i and added to the controller output $u(t)$. Thus, the contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral term is given by: $I_{out} = K_i \int_0^t e(\tau)d\tau$. The addition of the integral term accelerated the movement of the process towards the setpoint and eliminated the residual SSE. Nevertheless, since the contribute of the integral term is an accumulation of errors from the past, it might cause the controller output value value to overshoot the setpoint value.

4.1.3 Derivative term

The derivative of the process is the slope of the error at a current time, then, multiplying this rate of change by the derivative gain K_d. The derivative term is given by: $D_{out} = K_d \frac{de(t)}{dt}$. Derivative action predicts system behavior, therefor its effect relates to the future behaviour of the process. In order to limit the effect of noise and high frequency gain PID controllers include an additional low-pass filtering for the derivative term.

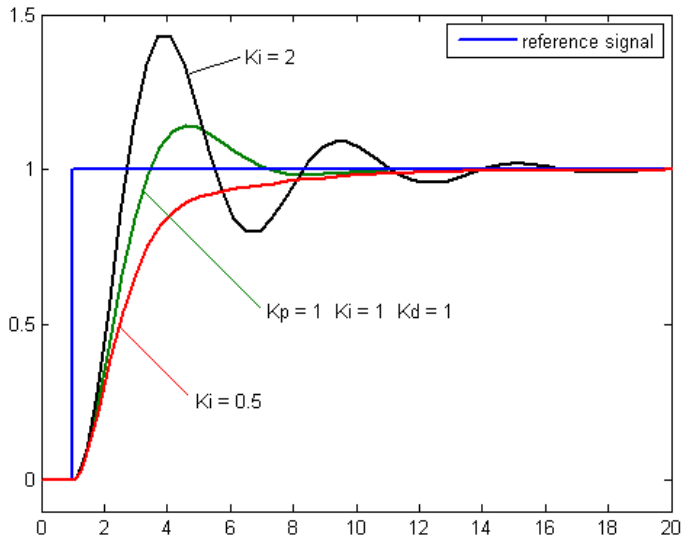


Figure 17: Response of PV to step change of SP vs time, for three values of K_i (K_p and K_d held constant)

4.2 PWM vs PFM

The system described in the article[5], is focused on driving DC motors with spikes. In robotics applications, it is common to drive DC motors using pulse-width-modulation (PWM). PWM is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. A signal is sampled at a constant period T_{PWM}, with varying high time T_h and applied to the DC motor. A duty-cycle or power-cycle is the fraction of one period in which a signal or system is high. Thus, from the motor perspective, DC power ($V_{DCmotor}$) is proportional to the PWM signal duty-cycle, as indicated by the equation:

$$V_{DCmotor} \approx \frac{T_h}{T_{PWM}} V_{PS}$$

where V_{PS} is the DC voltage that is provided by the power supply stage. In the PFM approach, the information (the desired voltage in the case of DC motor) resides in the spike's frequency. In a PFM scheme the T_h is constant and the Inter-Spike-Interval (ISI) is variable. as shown in the equation:

$$V_{DCmotor} \approx \frac{T_h}{T_{ISI}} V_{PS} = T_h \cdot Spikes_{rate} \cdot V_{PS}$$

Looking at the equation, it is apparent that if spikes are very short in duration (e.g., $T_h = 20ns$), they do not represent much energy and, thus, it will be filtered by the power stage optocouplers[25]. Hence, T_h fixed time should be longer (e.g., $T_h = 1\mu s$). This solution is provided by a spike extension (SE)block implemented as a part of the sPID controller and covered in the following section. Fig. 18 shows how in the PFM scheme, when excitation is low, spike rate is low and thus the time between spikes is high, however, when signal excitation increases, the ISI time decreases, while spike rate increases.

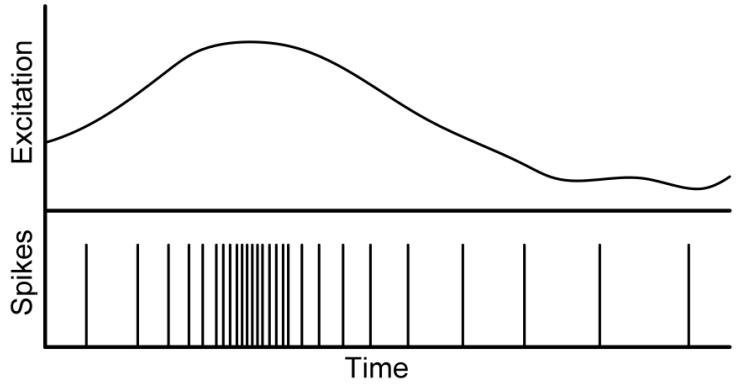


Figure 18: Spikes codification for a variable excitation level.

4.3 Spike-Based PID Position Motor Controller

The spike-based PID (sPID) controller is a Proportional-Integrative-Derivative controller completely designed considering the requirements of the spike paradigm. Both, its reference input signal and its output signal to drive the motor are spike-based signals. The PFM modulation is used along Spike-Signal-Processing (SSP) Building Blocks (based on previous work of [6]) to compute the sPID Laplace domain classic PID formulation and to drive the motors. As seen in section 4.2, the spiking reference can be used to control the speed (or DC voltage input) of a DC motor. Yet, in the article, the sPID is used to manipulate a robotic arm, hence, the spike rate value should represent a joint target position and not speed. Nevertheless, the principle of the calculation is similar and the updated equation is shown in the following sections. Fig. 19 shows the block diagram of the sPID controller.

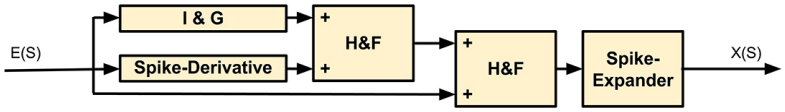


Figure 19: Spike-based PID controller created with SSP building blocks.

4.3.1 Reversed Bitwise Synthetic Spike Generator

The Reversed Bitwise Synthetic Spike Generator (RBSSG) block is responsible to convert a digital value (link position), denoted as x to a PFM signal. Fig.20 shows the circuit diagram of the SSG, as implemented in previous work [6]. The output spikes are generated by using an N-bit counter, driven by the internal clock frequency divider ,and a comparator. The counter's output is reversed bitwise and compared with the input absolute value (ABS(x)). When the input absolute value is greater than the reversed counter value, a new spike is fired. The spike rate can be controlled by adjusting the number of bits in the counter and the frequency divider applied to the internal clock. Positive and negative spikes are generated, where the input's sign, or MSB is used as an input to a DEMUX to select the right output. The Spike rate of the RBSSG output given a digital value x, is represented by the equation:

$$RBSSG(x)_{SpikeRate} = \frac{F_{clk}}{2^{N-1}(genFD+1)} \cdot x$$

4.3.2 Integrate-And-Generate Motor Neuron Model

This block is composed of a spike counter and an RBSSG, as shown in Fig. 21.The spike counter is a digital counter whose value increases when a positive

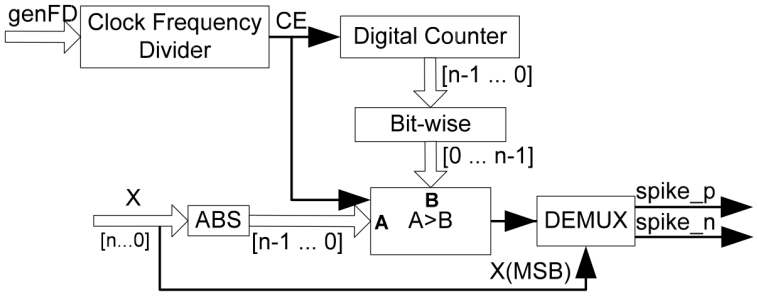


Figure 20: Reverse Bitwise Synthetic Spikes Generator internal blocks.

spike is received, and decreases when a negative spike is received. The spikes counter output will be connected to the RBSSG input, and this will generate again a new stream of spikes. These spikes will have a frequency proportional to the spike count, i.e., the spikes integration. The equation describing the output spike rate of the integrator is:

$$f_{SI\&G}(t) = k_{SI\&G} \cdot \int_0^t f_{inputSpikes} dt = \frac{F_{clk}}{2^{N-1}(genFD+1)} \cdot \int_0^t f_{inputSpikes} dt$$

And the equivalent $SI\&G$ transfer function is:

$$SI\&G(s) = \frac{F_{SI\&G}(S)}{F_{inputSpikes}(S)} = \frac{k_{SI\&G}}{s} = \frac{F_{clk}}{2^{N-1}(genFD+1) \cdot s}$$

Hence, the value of the integral gain is:

$$K_i = K_{SI\&G} = \frac{F_{clk}}{2^{N-1}(genFD+1)}$$

Fig. 22 presents an example of spikes integration that justifies the correct response of this block. The correspondence can be seen graphically.

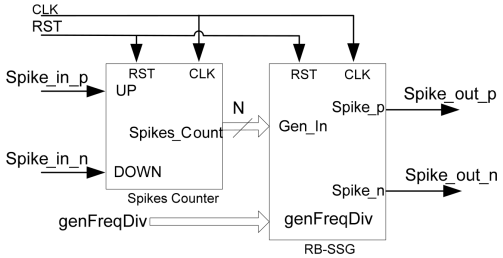


Figure 21: Spikes Integrate & Generate block diagram.

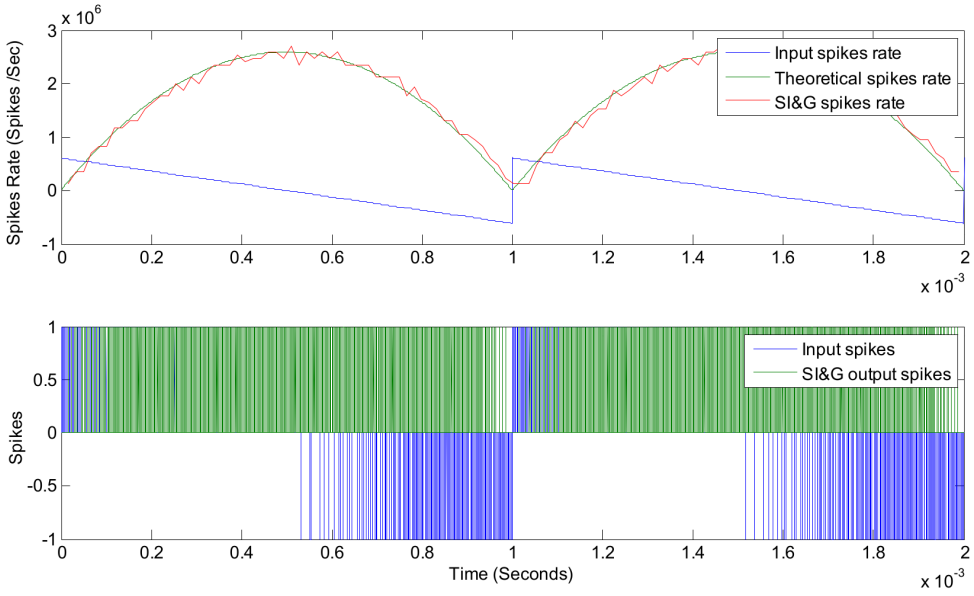


Figure 22: Both input spike rate (blue) and output spike rate (green) are represented at the bottom. The same can be seen at the top of the figure, although represented in spikes rate values. Input represents a saw tooth with positive and negative values (blue). Ideal integral is presented in green and simulated output is presented in red.

4.3.3 Spike Temporal Derivative

To implement the Spike Temporal Derivative (STD) part of the sPID, the researchers designed the block shown in Fig. 23, where the idea is to feedback an SH&F with SI&G, being SH&F the difference between input spikes, and the integration of derivative spikes.

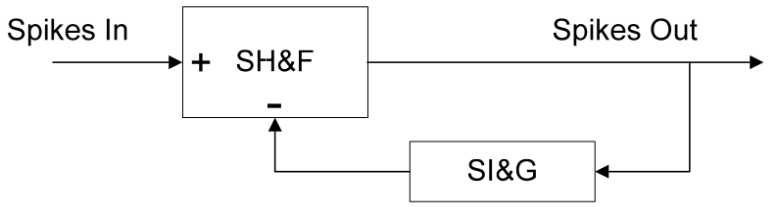


Figure 23: Spikes Temporal derivative architecture. The block consists of an Integrate and Generate block connected to a Hold and Fire to close-loop. The result is that the output rate is the derivative of the input rate.

By analyzing the topology of the STD block shown in Fig. 23, the transfer function in the laplace domain is calculated as follows:

$$SpikesOut(s) = SpikesIn(s) - SpikesOut(s) \cdot SI\&G(s)$$

$$SpikesOut(s)(1 + SI\&G(s)) = SpikesIn(s)$$

$$STD(s) = \frac{SpikesOut(s)}{SpikesIn(s)} = \frac{1}{1 + SI\&G(s)} = \frac{1}{1 + \frac{K_{STD}}{s}} = \frac{s}{s + K_{STD}}$$

and the K_{STD} constant is calculated in a similar way as $K + SI\&G$ but taking into account the circuit parameters of this derivative block:

$$K_d = K_{STD} = \frac{F_{clk}}{2^{N_d - 1}(genFD_d + 1)}$$

Fig. 24 presents an example that justifies the correct response of this block where correspondence between the theoretical and calculated STD spike rates can be seen graphically.

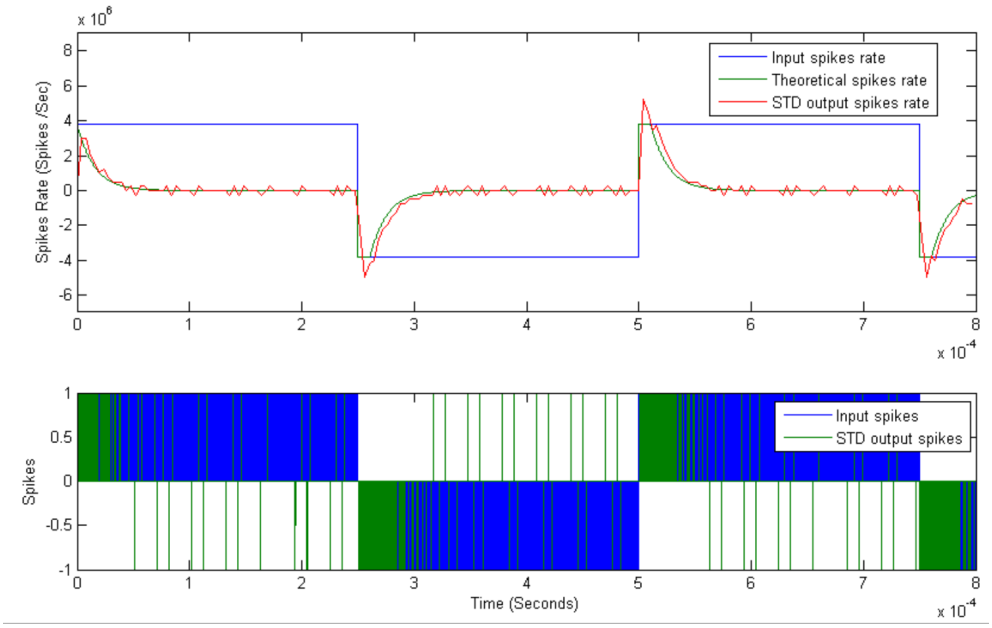
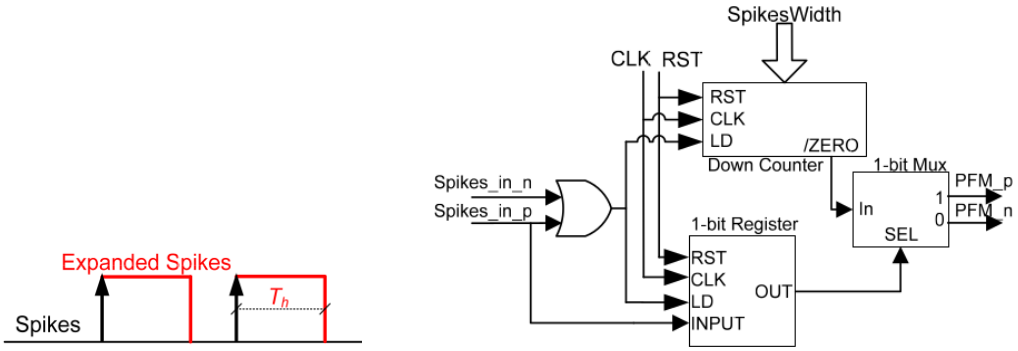


Figure 24: Spikes Temporal derivative response. Both input spike rate (blue) and output spike rate (green) are represented at the bottom. The same can be seen at the top of the figure, although represented in spikes rate values. Input represents a periodic square signal with positive and negative values (blue). Ideal integral is presented in green and simulated output is presented in red..

4.3.4 Spike Expander

Since a DC motor acts as a low pass filter[26] the narrow output spikes (1 clock cycle) from Integrate and Generate, Hold and Fire and Derivative blocks, could not be able to produce the needed force to the motor axis, or they could even be filtered. Therefor, the output spikes to the DC motor have

to be lengthened in order to allow a motor movement as seen in Fig. 25a . Fig. 25b shows the Spike Expander (SE) internal circuit, based on a digital down counter with auto-stop and an active-low $ZERO$ output signal. The idea is to set the down counter with a fixed value, denoted as $SpikesWidth$, and count down as many clock cycles as indicated by this value. While the down counter is being decremented, $ZERO$, which represents the expanded spike, will be set to '1'. The positive or negative voltage selection is demultiplexed according to the sign of the input spike.



(a) (b)

Figure 25: (a) Expanding spikes in order to drive a DC motor. (b) Spikes Expander internal components.

The extended T_h is proportional to the clock period and to the $SpikeWidth$ parameter and can be described by the equation:

$$T_h = T_{CLK} \cdot SpikeWidth$$

Since changing the time length of a single spike, proportionally changes the motor command, the SE block plays the role of the proportional term of a PID controller. The following equation defines K_p:

$$K_p = T_h \cdot V_{PS}$$

where V_{PS} is the voltage of the DC motors power supply.

4.3.5 Spike-Based PID Position Controller

The blocks presented in the prior subsections conclude the necessary sPID building blocks as previously shown in Fig. 19. The idea is to propagate, integrate and derive the error using two SH&F for spike addition, and with SI&G and STD. The error spikes, the SI&G and STD spikes are summed

and then expanded to drive the motors. The sPID transfer function is as follows:

$$sPID(s) = K_p(1 + SI\&G(s) + STD(s)) = K_p(1 + \frac{K_i}{s} + \frac{s}{s + K_d})$$

sPID controller behavior can be set up using the K blocks parameters. These parameters are defined by the expanded spikes high time T_h, and the number of bits of the digital counters of diverse RBSSG included in the blocks.

4.4 The ED-BioRob Robotic-Arm

The BioRob robot [27] is a light robotic arm based on the concepts of elastic and antagonistic actuation, which are inspired by the biological muscle-tendon elastic system. Each of the four joint of the arm has a DC-motor coupled to ropes and springs as elastic components within the rest of the arm and two sensors: a position sensor attached to the DC-motor, which consists of an opto-encoder, and an angular sensor to measure the absolute angle of the joint. Fig. 26 shows the actual position of each DC-motor in the architecture of the arm. The event driven(ED)-BioRob implementation is accomplished by integrating the spike based hardware and logic previously described. Firstly, in order to apply a position control technique to the joints of the BioRob arm with the sPID controller,[6] a new Integrate-and-Generate block was included at the output of the optical encoder sensor of each motor completing the sPID loop and therefor denotes as close-loop integrator (CL_i). This block makes compatible the output from the encoders with the spiking reference signal. Thus, the error signal E(S), which is used as the input for the sPID, can be computed. The SSP building block are implemented in two different platforms due to the limited hardware resources of the FPGAs included on the boards. Fig. 27 shows a block diagram of the implemented architecture.

The first platform is the AER-Robot[28], with a Spartan-3 FPGA is used to interface the robotic arm by accomplishing these objectives: (1) Driving the four DC motors using PFM signals. This requires the implementation of the SE block. (2) Read the motors opto-encoders and convert the data into spikes. (3) Read out the position from each motor sensor and use it as ground truth. This board is connected to a PC for parameters and reference settings using the jAER[29] (java based) software.

The second platform is the AER-Node[30], with a more resourceful Spartan-6 FPGA on board. On this board the rest of the sPID block are implemented (All but the SE block). Also, in this board a spike generator is implemented to convert the digital reference value to a spike based one and a CL_I SI&H

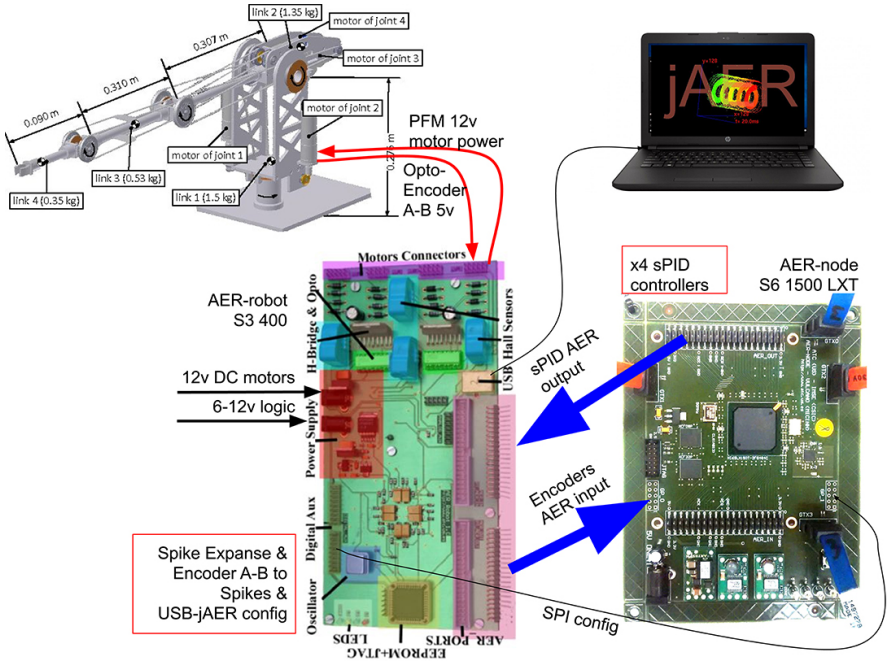


Figure 26: Spike-based PID controller for BioRob split in two FPGA platforms. The sPID without the Spikes Expander is implemented on the Spartan 6 and the Spikes Expander block is implemented on the Spartan 3 platform.

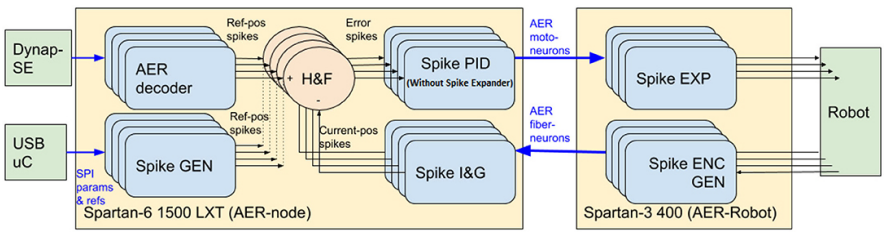


Figure 27: Spike-based PID controller created with SSP building blocks.

block is added to convert the encoders data to spikes. Lastly, a block that includes an AER decoder is implemented for interfacing a neuromorphic hardware (Dynap-SE[31] in this case) to the sPID. More details of the later block are in section 4.5.

Before moving on to the experiments, the researches have configured the K_p, K_i and K_d gains to obtain the same speed and precision for each joint. Tab.5 shows the configured parameters.

Table 5: K_p, K_i and K_d parameters of the 4 sPID controllers of the ED-BioRob joints and the close-loop integrator (CL_i).

Joints	SW	K_p	NB_i	FD_{GEN_i}	K_i	NB_d	FD_{GEN_d}	K_d	NB_{CL_i}	$FD_{GEN_{CL_i}}$
J1	1,024	2.5e-4	18	4,096	9.3e-2	22	4,096	5.8e-3	18	16
J2	1,024	2.5e-4	18	4,096	9.3e-2	22	4,096	5.8e-3	18	16
J3	512	1.2e-4	18	4,096	9.3e-2	22	4,096	5.8e-3	18	16
J4	512	1.2e-4	18	2,048	0.186	22	2,048	1.2e-2	18	16

NB is bit length, FD is frequency divider value and SW is spike width.

4.5 Experiments

4.5.1 Robot Characterization

Joint range of motion

Each of the four joints was tested by moving 180°, from −90° to 90° where 0° is the vertical home position. The stimulus spike was generated using the RBSSG block with an input ranging from -500 to 500, a clock frequency of 50 MHz and a 16-bits counter. Therefore, the generated spike rate represents a signal with negative polarity from 762.94 Kspikes/s to 0 spikes/s and positive polarity from 0 spikes/s to 762.94 Kspikes/s. The positions commands were sent separately to each joint, every second with a step value of 50 which represents a spike frequency step of 76.3 Kspikes/s. Fig. 28 shows the commands and the sensor readings (encoder) of each joint and Tab. 6 summarizes the robot joint's ground truth angles, measured using a video tool, the angle sensors readings and the spike reference signals. The robot joints successfully moved in the desired range as can be supported by the sensor reading and angles measured. The researchers claim that the differences between the angles measured with the joint sensor and the video tool can be understood due to the inertia of the iterative test performed. Also noted that joints 3 and 4 sensors were not properly aligned to the robot according to the selected home position, which caused an overflow of the 15-bit internal counter of the sensor and explains the overflows/underflows in their graphs.

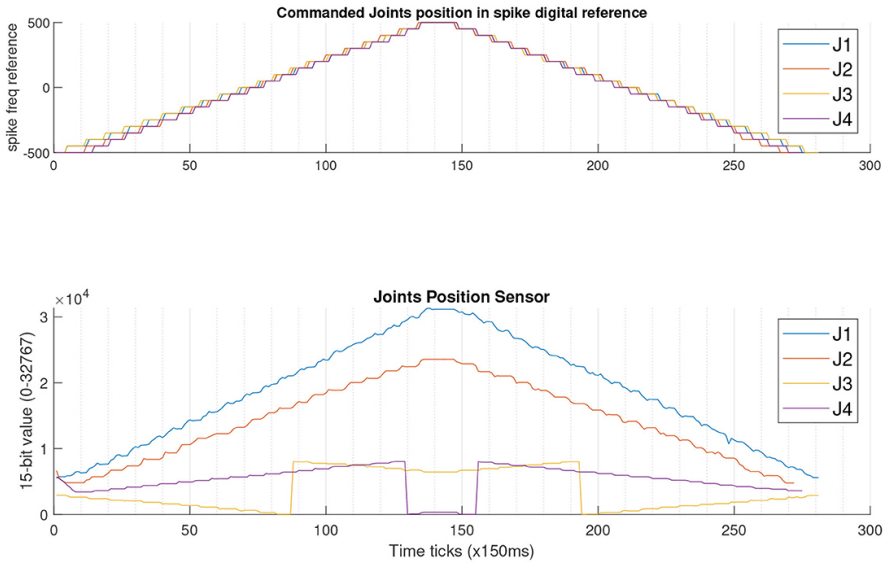


Figure 28: Characterization of joints 1–4. For each joint: (Top) The commanded position to the joint.(Bottom) The raw data read from the sensor.

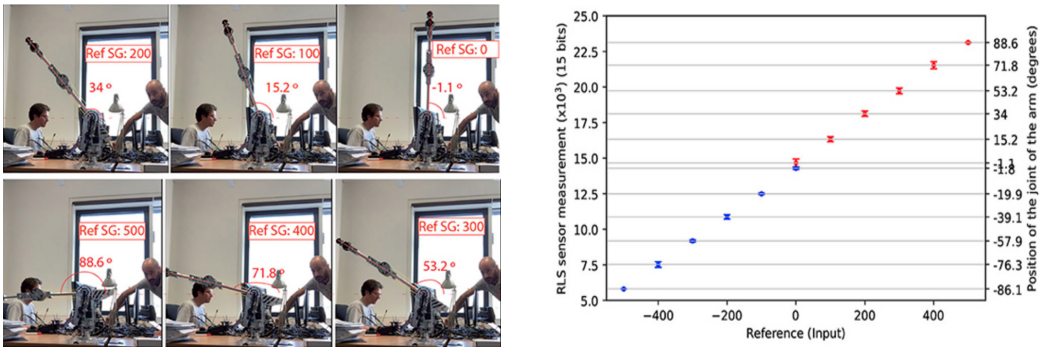
Table 6: Values for the experiments to characterize the joints (J_i) and their sensors (S_i).

Dig. Ref.	J1 (degrees)	S1 (degrees)	J2 (degrees)	S2 (degrees)	J3 (degrees)	S3 (degrees)	J4 (degrees)	S4 (degrees)
−500	−88.93	−108.63	−86.1	−83.87	−106.52	42.02	−122.12	−43.98
−400	−71.38	−116.21	−76.3	−83.52	−85.38	40.3	−104.18	−44.58
−300	−56.37	−86.80	−57.9	−63.53	−60.18	31.15	−78.45	−33.23
−200	−29.21	−58.46	−39.1	−41.9	−35.63	21.52	−50.93	−21.89
−100	−18.95	−29.75	−19.9	−22.75	−17.08	11.44	−20	−10.92
0	0	0	−1	0	1.69	0	0	0
100	18.82	28.21	15.2	20.11	20.91	−11.37	14.38	10.86
200	34.46	56.73	34	42.02	36.56	160.37	33.27	21.96
300	52.8	85.04	53.2	63.73	55.82	151.06	63.29	34.24
400	70.31	114.74	71.8	82.91	77.38	141.63	86.46	45.41
500	91.06	137.90	88.6	102.58	96.64	130.41	109.64	−124.72

Controller accuracy of motion

To test the accuracy of a the controller when commanding the joint to a specific angle, joint 2, which is the one supporting most of the weight of the arm was tested. This time a set of six reference values. from 0 to 500 in steps of 100 (an 18° step), were given to the controller to move the joint within the range from 0° to 90° repetitively as the first part of the test. Then, the same experiment was conducted four times for the range -90° to 0° without resetting the controller or re-configuring the home position. Fig 29a shows the ground truth angles, measured using a video tool. Fig. 29b shows a graph of the positions reached by joint 2 when moving according to

the given command. The RMSE of the first range (0° to 90°) is 1.61° (red marks in the figure) and the RMSE of the second range (-90° to 0°) is 3.3° (blue marks in the figure). The standard deviation is higher for the center point (0°) and for both ends of the experiment (90° and -90°).



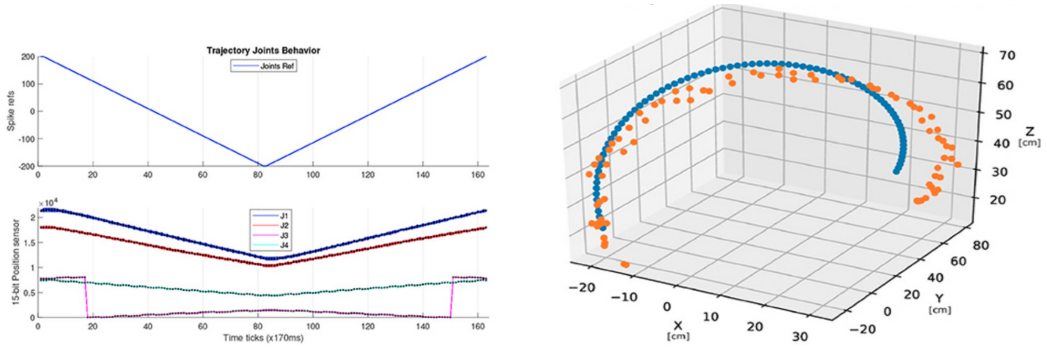
(a) (b)

Figure 29: (a) Measurement of the angles reached by joint 2. Both the input reference (labeled as Ref.SG) and the angle, are shown. The home position of joint 2 is vertical 0°. (b) Positions reached by the joint when moving from -90° to 90°. Two colors are used to distinguish between positive and negative references and to show the difference in the mid position (0°).

4.5.2 Trajectory Planning

Accuracy along a 3D trajectory

This test examined the robot performing a trajectory by moving all the joints simultaneously, with a stimuli from a digital reference of -200 to 200 with unitary steps (up to 152.6 Kspikes/s in both polarities). The experiment measurements in Fig. 30a show that there is an incremental error of 2.39% considering the measurements of the position sensors of joints 1 and 2 (blue and red traces) over 10 iterations. Joint 4 sensor demonstrates the same counter overflow as in the previous experiment. Fig. 30b shows the 3D representation of the end-effector trajectory. The blue trace shows the trajectory commanded and the orange dots represent the position of the end-effector of the robot during the 10 iterations of the experiment. The average error is 6.6°. All the details for these values are given in Tab. 7. This is a **link** to a video showing the entire 3D representation of all runs.



(a) (b)

Figure 30: (a) Ten iterations of a trajectory movement for all joints simultaneously from -200 to 200 spike refs. (b) 3D representation of the end-effector trajectory.

Table 7: Measurements summary of 10 iteration averaged for each joint. References are in digital value and in Kspikes/s, angles are in degrees and cartesian coordinates are in cm.

Dig. Ref.	Ref. (ksps)	Joint 1 (degrees)	Joint 2 (degrees)	Joint 3 (degrees)	Joint 4 (degrees)	X (cm)	Y (cm)
−200	−305.17	−49.43	−38.14	−46.45	−76.71	−14.09	−25.10
−150	−228.88	−37.42	−31.12	−36.25	−64.93	−19.53	−23.53
−100	−152.59	−23.40	−18.96	−26.93	−42.60	−20.19	−13.63
−50	−76.29	−9.33	−9.02	−11.11	−30.47	−10.56	5.16
0	0	4.62	2.03	−0.38	−6.73	5.70	28.78
50	76.29	18.70	12.77	10.60	11.11	21.10	51.79
100	152.59	35.96	23.69	21.72	22.93	31.24	70.38
150	228.88	49.42	33.83	32.40	46.23	26.24	81.38
200	305.17	55.30	37.84	33.90	59.81	23.01	82.61

Accuracy along a 2D trajectory

In this experiment the base joint (joint 1) of the arm was not used, which limited the end-effector movement to a 2D plane. Each of the remaining joint was given a different set of 4 digital reference inputs as its trajectory. The trajectory was commanded point by point in a cyclic way , from point 1 through point 4 and back to point 1. Commands were sent at 0.25 Hz and measurements were taken at 6.5 Hz(Fig. 31a and Fig. 31b). Due to the fact that the sPID controller constants (K_p, K_i, K_d) were adjusted manually to have a quick response and since the robot has elastic joints, in Fig. 31c it shows that the commanded points (blue) and the points reached by the robot (orange) did not coincide during the robotic arm movement between two consecutive points (initial and final). Tab. 8 shows the experiment average results.

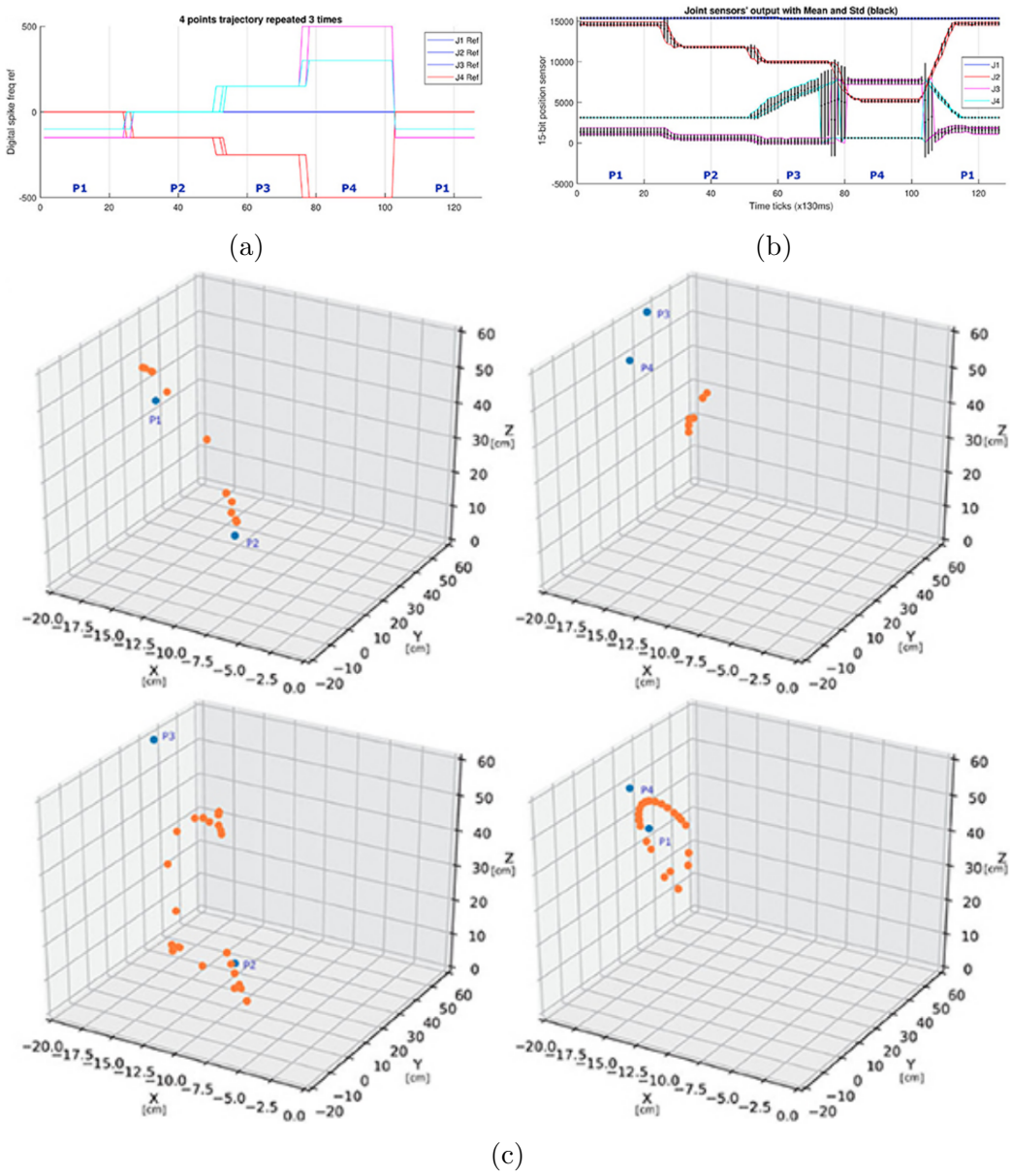


Figure 31: (a) Three repetitions of a sequence of four points commands in space (joint 1 not used) superposed. Units are ticks of 130 ms. (b) Joints mean response (colored) and Std (black) to the commands of (a) . (c) 3D representation of the trajectory segments followed by the robotic arm among four points. Blue points are the input reference points and the orange points are the trajectory sampled from one point to another.

Table 8: Measurements average of 4 iteration of trajectories for each joint. References are in digital value and in Kspikes/s, angles are in degrees and cartesian coordinates are in cm.

Ref. J2	J2	S2	Ref. J3	J3	S3	Ref. J4	J4	S4	(x, y, z)
0	−1	5667.69	−150	−36.25	1363.82	−100	−20	3470.47	(−12.01, 13.42, 49.65)
−150	−31.12	10154.28	0	1.69	558.34	0	−6.73	3129.42	(−11.26, −1.19, 46.55)
−250	−48.5	12023.23	150	32.4	39.12	150	46.23	5320.04	(−13.38, −4.37, 55.33)
−500	−86.1	14502.67	500	96.64	7808.23	300	63.29	631.61	(−4.55, −20.30, 18.84)

4.6 Dynap-SE Control of the Robot

The Dynamic Neuromorphic Asynchronous Processors (DYNAPs) are a family of low power, scalable SNN processors. The Dynap-SE spiking neuromorphic processor was used to send positions commands to the robot controller for each joint. Each Dynap-SE comprises four chips, each having four cores with configurable connectivity of neural populations. As shown in Fig. 27 the communication between the Dynap-SE and the controllers required an additional AER decoder. The reference value, represented by activity rate of the neural population in the Synap-SE is transmitted via its AER ports to the AER decoder module in the robot controller. The AER decoder is connected to the sPID module. In an experiment, the researcher tested different neuron activities to produce the combined activity required to produce a reference PFM signal, sent to the controller to change the angles of joint 4. Fig.32 shows the results with four different angles of joint 4 and the Dynap-SE software interface that produces the required PFM reference signals for the FPGA embedded sPID controller by joining several neuron-populations activity.

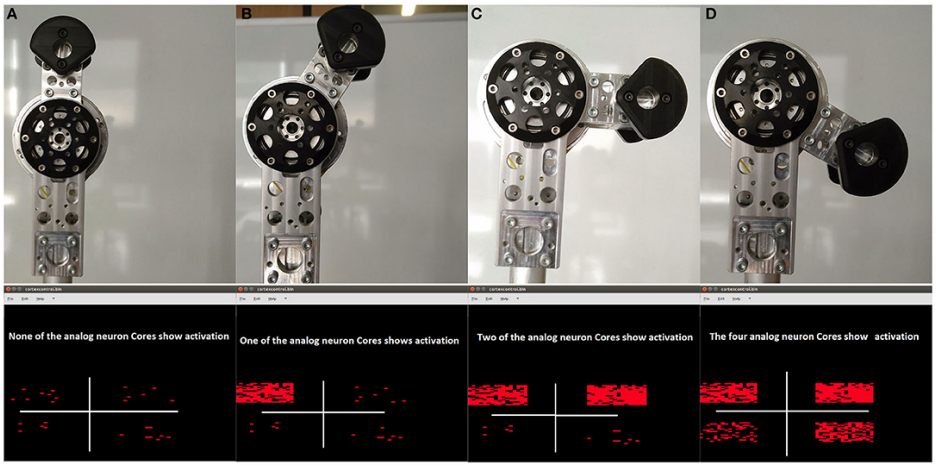


Figure 32: (Top): From left to right, robot joint positions from 0° to 130°. (Bottom): Dynap-SE software interface with neurons activity of the 4 commands. The plots show how the activity (red dots) of the populations implemented on Dynap-SE changes within the range of the joint. (A) Home, 0°, (B) 30°, (C) 90°, (D) 130°.

4.7 Summary

The article described in great details how a spike-based PID speed motor controller was adapted to control the position of the 4 joints of a light and safe physical human-robot interaction (pHRI) robotic arm, called ED-BioRob. These sPID controllers were deployed in two FPGA platforms, i.e., the AER-Robot and the AER-Node boards, which provide Address-Event-Representation interfaces for spiking systems and can drive DC motors with Pulse Frequency Modulation signals, mimicking the motor-neurons of mammals. The system allows receiving the reference signals for the joints from a computer, through USB and the open-source software jAER; or from a neuromorphic processor (DYNAP-SE) executing a spiking neural network. The experiments conducted in this work show that the sPID offers the worst RMSE of 3.3° after several iterations of joint movements from -90° to 90°. The system is totally functional for performing point-by-point trajectories. It was demonstrated that the robot can be commanded through a population of silicon neurons.

5 Stereo correspondence - Practical Results

The purpose of this section is to provide an implementation of a SAD based correspondence algorithm. It was decided to concentrate on implementing a conventional , non SNN based algorithm in order to establish deep understanding of the EC operation and to provide a starting point for a future research to be conducted in the progressive thesis. The entire code for this small project is available on a GitHub repository and can be accessed using this link: **https://github.com/Turgibot/Seminar**
A YouTube video of the projects results is also available via this link:
https://youtu.be/QrYlgrMWYEE

5.0.1 SAD based stereo matching algorithm

Algorithm 2 provides the pseudo code for finding a match of a single pixel (with a minor change of finding the minimal SAD value instead of the maximum correlation value). The code in listing 1 is part of a python file named *match_finder.py* where the position of the pixel with the minimal SAD value of a based search is returned.

```
while i <= y_end:
    j = x_start
    m = 0
    while j <= x_end:
        for k in range(3):
            src_patch[t, m, k] = left_img[i, j, k]
        m += 1
        j += 1
    t += 1
    i += 1
x_start = 0
x_end = x_start + side
while x_end < right_img.shape[1]:
    t = 0
    i = y_start
    while i <= y_end:
        j = x_start
        m = 0
        while j < x_end:
            for k in range(3):
                tgt_patch[t, m, k] = right_img[i, j, k]
            m += 1
            j += 1
        t += 1
        i += 1
```

```
26      diff_patch = abs(src_patch - tgt_patch)
27      sads.append(diff_patch.sum())
28      x_start += 1
29      x_end += 1
30  return np.argmin(sads)
```

Listing 1: calculating the SAD of a single pixel

Fig. 33 shows a run of the entire *match_finder.py* file. The application shows a stereo image. When a mouse is clicked on a pixel in the left image the matching right image pixel is found and the depth is derived from the pixels disparity and marked on the left image. The depth calculation is done trigonometrically as explained in section 3.1.4. As show in Fig. 34 and explained on section 3.1.3, calculation the disparity or depth of each of the pixel and mapping the result to a color map produced a depth map. As can be seen on the depth map the SAD cost based matching algorithm fails in smooth single colored surfaces like the the computer screen or a white wall.

Figure 33: Left and Right images of a stereo camera. Three mouse click were pressed on the left image. The corresponding matches are marked on the right image as red rectangles and the depth is calculated and shown at each click position. Actual depth is marked on the three Lego blocks

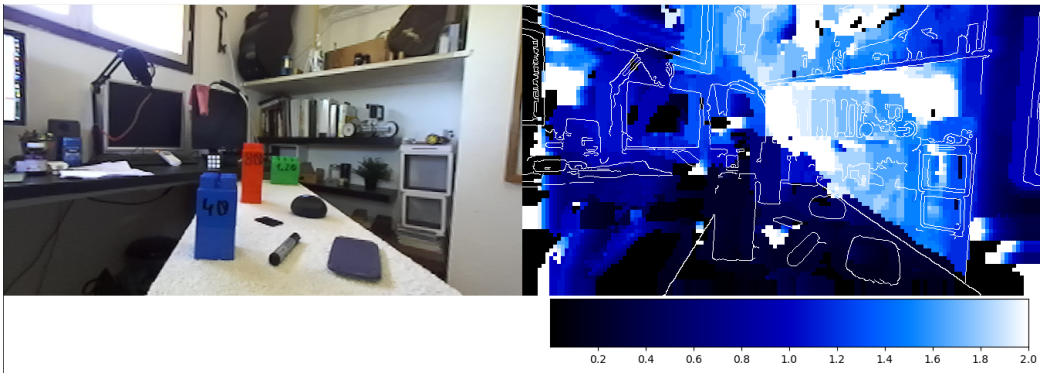


Figure 34: Left: left image of a stereo pair. Right: The corresponding depth map and a color bar. The edges of the left image are shown on the depth map to assist in locating the objects. The scale of the color bar is in meters.

5.0.2 Using the SAD based algorithm on event data

This section provides a description of the implementation of the a SAD based disparity matching algorithm that takes events from a stereo event camera as input and solves the correspondence problem using event data solely.

Due to the unavailability of event cameras, the event data was produced of a conventional pair of stereo images. To do so, the following steps were taken:

1. Use a conventional pair of rectified stereo images locally taken using a ZED stereo camera[32]. The shape of each image taken was 160X224X3.

2. Convert each of the images to a video and simulate the necessary movement to produce events. This movement was done using a saccades technique[33] and the implementation code can be found in the GitHub repository python file: *stereo_saccades.py*

3. Each of the video files (Left and Right) had to be up-sampled to simulate an increased frame rate. This was done using an open source video interpolation software called super-slo-mo[34]. The upsampling software also convert the images to a grey scale format. The original 30fps 1.5 second that had 48 frames converted to 236 frames at 157 fps.

4. To convert the the upsampled images, another open source software called Vid2E[35] was used. More than 2 million events were generated. The generated event video can be seen running the program in the *show_events_streams.py* file. A single frame of the video is shown in Fig. 35.

After generating the stereo event data, the same SAD based approach was taken to solve the correspondence problem. This step partly complies with the operation performed by the the coincidence detecting neurons described in section 3.2.2.

Fig. 36 shows a mouse click application that finds a matching event in the same manner as in section 5.0.1. Then a depth map was generated as shown in Fig. 37.

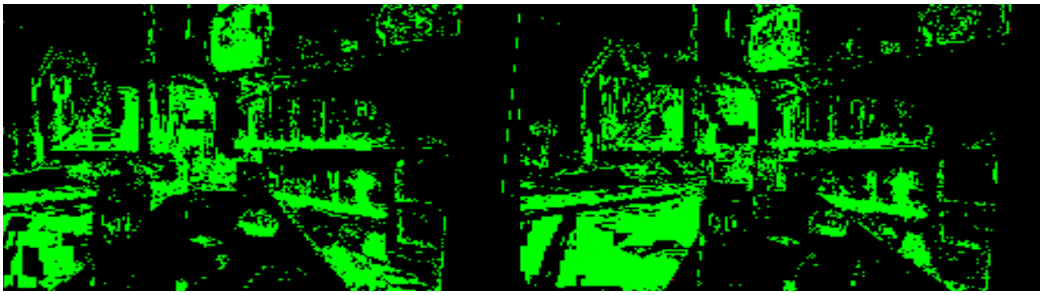


Figure 35: Stereo event camera output. Left and right images are an accumulation of approximately 9000 events in a duration of $6\mu s$. The red color is a positive polarity event and green is a negative one.

Figure 36: Left and Right frames showing $6\mu s$ of accumulated 9000 events. Mouse click were pressed on the left image. The corresponding matches are marked on the right image as white rectangles and the depth is calculated and shown at each click position.

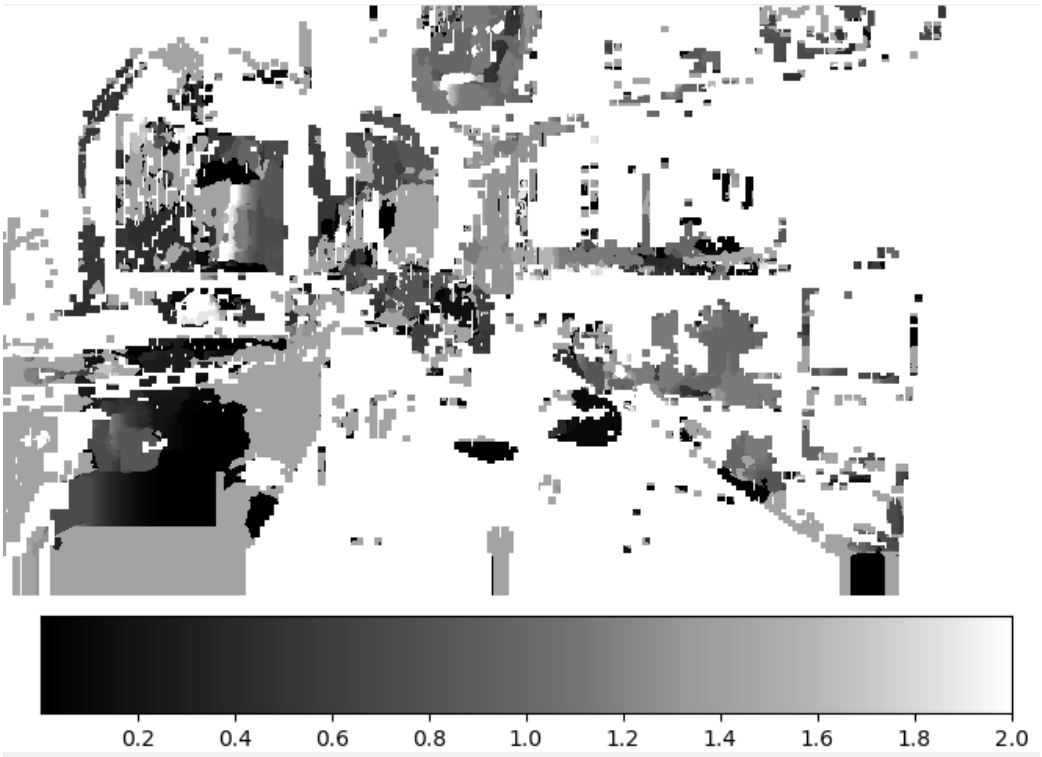


Figure 37: Depth map generated from event data. The scale of the color bar is in meters.

5.1 Summary

Analysing the event depth map qualitatively, it is possible to say that the SAD algorithm operates poorly on its given event data compared to the conventional camera data. This conclusion stands in accordance with the findings in the article. Using SAD minimization on its own as a matching criteria result in many false correspondences. The model described in the article solves these false matched by conducting eliminating steps.

1. Setting the temporal sensitivity factor $S_{\Delta T}$ shorter to eliminate a possible match between pixels on the same epipolar plane but with long spiking time difference. This was not configurable in the implemented code and was set constant as the frame period of $6\mu s$.

2. Considering the smoothness of the disparity function in space, a coincidence detector excites neighboring neurons on the same disparity space. The implemented code did not consider the smoothness of the disparity function as a matching criteria.

3. Using mutual inhibition of disparity detectors implements a winner takes all mechanism where the first to fire neuron inhibits other possible matches on the same line of sight. The implemented code does not act in an inhibiting manner of any sort and picks the matched pixel as the one with the minimal cost, even if others with an equal value exist.

4. The model uses event polarity as as additional criteria for matching where events of different polarity are disregarded. The implementation does not take in consideration the polarity of an event.

5. Another layer of inhibition is accomplished in the SNN model when a disparity neuron spike is only considered valid when immediately preceded by a coincidence spike representing the same location in disparity space. This in not implemented in the SAD algorithm as well.

References

[1] The Spoon. Lg and samsung to show off new food identifying smart fridges. `https://thespoon.tech/lg-and-samsung-to-show-off-new-food-identifying-smart-fridges-at-ces-next-week/` , 2020. [Online; accessed 25-May-2021].

[2] Ehsan Nabavi, Katherine Daniell, Elizabeth Williams, and Caitlin Bentley. *AI for sustainability: A changing landscape*, pages 157–176. Future Leaders, 01 2020.

[3] Marc Osswald, Sio-Hoi Ieng, Ryad Benosman, and Giacomo Indiveri. A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems. *Scientific Reports*, 7:40703, 01 2017.

[4] David Marr and Thomaso Poggio. Cooperative computation of stereo disparity. *Science*, 194:283 – 287, 1976.

[5] Alejandro Linares-Barranco, Fernando Perez-Peña, Angel Jimenez-Fernandez, and Elisabetta Chicca. Ed-biorob: A neuromorphic robotic arm with fpga-based infrastructure for bio-inspired spiking motor controllers. *Frontiers in Neurorobotics*, 14:96, 2020.

[6] A. Jiménez-Fernandez, G. Jiménez-Moreno, A. Linares-Barranco, M. Domínguez-Morales, R. Paz-Vicente, and A. C. Balcells. A neuro-inspired spike-based pid motor controller for multi-motor robots with low cost fpgas. *Sensors (Basel, Switzerland)*, 12:3831 – 3856, 2012.

[7] Wikipedia contributors. Neuromorphic engineering — Wikipedia, the free encyclopedia, 2021. [Online; accessed 19-August-2021].

[8] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[9] Samsung. Samsung smartthings vision - home monitoring solution. `https://www.samsung.com/au/smartthings/camera/smartthings-vision-gp-u999gteeaac/`, 2020. [Online; accessed 19-July-2021].

[10] Bo-Wook Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Chang-Woo Shin, Keunju Park, Kyoobin Lee, Jin Man Park, J. Woo, Yohan J. Roh, Hyunku Lee, Y. Wang, I. Ovsiannikov, and Hyunsurk Ryu. 4.1 a 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation. *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67, 2017.

[11] Usman Babri, Munim Tanvir, and Khurram Khurshid. Feature based correspondence: A comparative study on image matching algorithms. *International Journal of Advanced Computer Science and Applications*, 7, 03 2016.

[12] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. 12, 10 2000.

[13] Jonatan Nordh and Marcus Vikén. Self-supervised stereo depth estimation: Depth estimation in multiple environments through an adaptive cnn and ir light. 2021.

[14] Andre Rochow, Max Schwarz, Michael Weinmann, and Sven Behnke. Fadiv-syn: Fast depth-independent view synthesis. *arXiv preprint arXiv:2106.13139*, 2021.

[15] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Gaetano Pernisco, Vito Renò, and Ettore Stella. Towards real-time monocular depth estimation for mobile systems. In *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*, volume 11785, page 117850J. International Society for Optics and Photonics, 2021.

[16] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *CoRR*, abs/1906.07165, 2019.

[17] Guillermo Gallego, Tobi Delbruck, Garrick Michael Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jorg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[18] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328, 1979.

[19] Wikipedia contributors. Kinect — Wikipedia, the free encyclopedia, 2021. [Online; accessed 7-August-2021].

[20] Wikipedia contributors. Random dot stereogram — Wikipedia, the free encyclopedia, 2021. [Online; accessed 8-August-2021].

[21] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141, 2015.

[22] Ning Qiao and Giacomo Indiveri. Scaling mixed-signal neuromorphic processors to 28 nm FD-SOI technologies. *CoRR*, abs/1908.07411, 2019.

[23] Zhenshan Bing, Claus Meschede, Florian Röhrbein, Kai Huang, and Alois C. Knoll. A survey of robotics control based on learning-inspired spiking neural networks. *Frontiers in Neurorobotics*, 12:35, 2018.

[24] Kiam Heong Ang, G. Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[25] Wikipedia contributors. Opto-isolator — Wikipedia, the free encyclopedia, 2020. [Online; accessed 15-August-2021].

[26] Rubica Mikov, Ivan Virgala, and Michal Kelemen. Speed control of dc motor. *American Journal of Mechanical Engineering*, 4(7):380–384, 2016.

[27] Thomas Lens, Jürgen Kunz, Oskar Von Stryk, Christian Trommer, and Andreas Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–6. VDE, 2010.

[28] Alejandro Linares-Barranco, Rafael Paz-Vicente, Gabriel Jimenez, Juan L Pedreño-Molina, Javier Molina-Vilaplana, and Juan Lopez-Coronado. Aer neuro-inspired interface to anthropomorphic robotic hand. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1497–1504. IEEE, 2006.

[29] Tobi Delbruck. jaer - java tools for address-event representation (aer) neuromorphic processing. `https://github.com/SensorsINI/jaer`, 2007. [Online; accessed 19-May-2021].

[30] Amirreza Yousefzadeh, Mirosław Jabłoński, Taras Iakymchuk, Alejandro Linares-Barranco, Alfredo Rosado, Luis A Plana, Steve Temple, Teresa Serrano-Gotarredona, Steve B Furber, and Bernabé Linares-Barranco. On multiple aer handshaking channels over high-speed bit-serial bidirectional lvds links with flow-control and clock-correction on commercial fpgas for scalable neuromorphic systems. *IEEE transactions on biomedical circuits and systems*, 11(5):1133–1147, 2017.

[31] Saber Moradi, Qiao Ning, Fabio Stefanini, and Giacomo Indiveri. A scalable multi-core architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *CoRR*, abs/1708.04198, 2017.

[32] Stereo Labs. Zed stereo camera. `https://www.stereolabs.com/zed/`, 2020. [Online; accessed 19-July-2021].

[33] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015.

[34] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation, 2018.

[35] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.