COMPUTER VISION

Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception

A. Mitrokhin*, P. Sutor*†, C. Fermüller, Y. Aloimonos

The hallmark of modern robotics is the ability to directly fuse the platform's perception with its motoric ability—the concept often referred to as "active perception." Nevertheless, we find that action and perception are often kept in separated spaces, which is a consequence of traditional vision being frame based and only existing in the moment and motion being a continuous entity. This bridge is crossed by the dynamic vision sensor (DVS), a neuromorphic camera that can see the motion. We propose a method of encoding actions and perceptions together into a single space that is meaningful, semantically informed, and consistent by using hyperdimensional binary vectors (HBVs). We used DVS for visual perception and showed that the visual component can be bound with the system velocity to enable dynamic world perception, which creates an opportunity for real-time navigation and obstacle avoidance. Actions performed by an agent are directly bound to the perceptions experienced to form its own "memory." Furthermore, because HBVs can encode entire histories of actions and perceptions—from atomic to arbitrary sequences—as constant-sized vectors, autoassociative memory was combined with deep learning paradigms for controls. We demonstrate these properties on a quadcopter drone ego-motion inference task and the MVSEC (multivehicle stereo event camera) dataset.

INTRODUCTION

In recent years, the limits of modern artificial intelligence (AI) and learning have been thoroughly explored in a plethora of datasets, learning architectures, general tasks to solve, and modalities of data. In addition, focus is shifting to what AI can do for modern robotics and what new forms of it are necessary. What is the science of putting together the different cognitive modalities of an intelligent system, such as vision, motoric actions, audio, and other various sensors? Currently, there are no real theories on how to feasibly achieve this, mainly because the representations and processing or learning techniques involved in the different disciplines also differ greatly from each other. Each time a new task is explored, scientists tend to start from the individual components.

One of the more important facets of modern robotics is the notion of integrating the sensory perceptions experienced by an agent with its motoric capabilities—the actions it can perform—to facilitating active perception (*1*, *2*), which is considered vital to the existence of autonomous, learning agents that experience the world and are required to interact with it to the best of their abilities (*3*, *4*, *5*). The issue of separation of modalities is present in this discipline; the perceptual space and the action space are mainly kept separate, with a central learning mechanism to infer the action given the perception or vice versa. Likewise, the various forms of perception themselves are largely learned or processed separately, embedded into vector representations, and essentially concatenated together as input to the aforementioned central learning mechanism. It is clear that we must do better than this. Ideally, the perceptions and actions experienced in the moment and in the past should influence the future actions of the agent and perhaps even bias its expectations for future perception. The main concern is how to integrate the two vastly different modalities of action and perception together in an effective way.

We propose starting with an integration constraint from the raw data themselves, that is, from the beginning, we require that perception and action need to eventually "bind" together. For this to happen, there should be some currency through which the perceptual module will interact with the control module or the planning module and so on. This currency is the hyperdimensional vector, a vector that exists in an extremely high-dimensional space. Here, we mainly concern ourselves with hyperdimensional binary vectors (HBVs) and the notion of facilitating hyperdimensional active perception (HAP). Whether you are an image, video, motion sequence, control sequence, concept, word, or sound, you are represented by an associated HBV. Furthermore, if you are a sequence of any of these modalities, then you are also an HBV of equal dimension constructed by constituent elements of the sequence, thereby existing in the same space as an encoding. Last, when considered together, such as a mix of modalities or sequences of them, this is also represented by an HBV constructed by binding each modality together, with, yet again, equal dimension. The integration problem now acquires new meaning—all information is represented as long binary vectors that are meaningfully constructed.

Although the philosophy of how to handle different modalities of data is important to facilitate active perception, we must also consider the "hardware" aspects. Some sensors are more suited to perceiving information that we consider suitable for the problem. Classical vision tends to focus on cameras that are red-green-blue (RGB) light intensity based, although we find that biological organisms tend to interpret signals in very particular ways. One example of this is motion. Motion is not well represented in classical cameras and vision techniques—a consequence of traditional vision being frame based and only existing in the moment, whereas motion is a continuous entity. So-called neuromorphic cameras attempt to capture this notion of seeing motion. The more recent development of the dynamic vision sensor (DVS) follows these lines in an interesting way, seeing sparse events in time, as opposed to pixels and intensities. With the introduction of such neuromorphic hardware, we are ready to cross the bridge of frame-based vision and develop a concept: motion-based vision. The event-based sensor provides dense temporal information about scene changes, allowing for accurate, fast, and sparse perception of the dynamic aspect of the world. When it comes to the marriage of action and perception, the fast, asynchronous events that the DVS sees are desirable for facilitating action. As of now, there are no real standards for how to effectively learn from DVS in the manner done for regular RGB cameras; it is all experimental.

Department of Computer Science, University of Maryland, College Park, MD 20742, USA.
*These authors contributed equally to this work.
†Corresponding author. Email: psutor@umd.edu

After recent accounts in the field (*6*), an "active perceiver knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception." The "what" question has to do with scene selection and fixation. The "when" question has to do with temporal selection, an instant in time and an extent (a history or episodic memory). The "how" question has to do with mechanical alignment, sensor alignment, and priming, and the "where" question has to do with viewpoint selection and the relationships between agent pose, sensor pose, and object pose.

All these questions can be addressed using our hyperdimensional framework. The what question amounts to developing a scene model from a series of fixations. Each fixation is characterized by "location" and "content" (the image in a window around the fixation point). If X_i is the hyper vector denoting the fixation location and Y_i is the hyper vector denoting the image around X_i, then by binding X_i to Y_i and summing up the hyper vectors for all fixations, we obtain $\sum X_i * Y_i$, a scene hyper vector based on fixations. Similarly, the when question can be addressed with hyper vectors. If $V(t)$ and $M(t)$ are the visual and motor signals during an action over time t, then for any t, the binding of the hyper vector $V(t)$ with $M(t)$ gives a new hypervector that is an "instant." Turning a sequence of such instants into a new hypervector leads to a history vector. The same can be said about the rest of the questions: For example, alignment problems amount to servoing of different kinds, where servoing is viewed as a prediction of an action with a goal. All of the above constitute future research goals.

Here, we focus on a simple problem that will allow us to evaluate our approach in depth. Thus, we focus on the problem of three-dimensional (3D) motion. First, we describe a framework for integrating multiple modalities such as perception and action together into a single space to produce features for learning in a manner that is desirable for active perception. In addition, we show how the DVS can be used in such a framework. We begin with some background information about HBVs and what they are capable of. Next, we describe how recent work on HBVs can be used to effectively generate meaningful, semantically informed HBVs and integrate them together to form representations for sequences or sets of data. After that, a method for encoding more complex signals, such as images, is discussed. We then describe recent work on DVS that was used to generate motion-based perception and how HBV representations of this can be made. In addition, we describe a method for integrating action and perception into a single space and show how the resulting HBVs could be used as "memories" of previous actions and perceptions. We show results on the use of HBVs with DVS-based information and how it compares with a more traditional vision approach to DVS signals to predict velocity and ego-motion. Last, we discuss our results and how they can be used in future applications to further facilitate the use of HAP and form a more coherent path to better AI in robotics.

RESULTS

In this section, we discuss the formalization of our HAP framework for the integration of action and perception, and any background information needed to understand it. We then provide experiments demonstrating the properties of this framework and how it can be used to facilitate active perception and create memories using a neuromorphic camera.

Properties of HBVs

Much of the inspiration behind the ideas discussed in this paper comes from Kanerva's work on hyperdimensional computing (*7*). The basic idea is to think about what sort of operations and information are computable and encodable with vectors of extreme dimensions. In the case of binary inputs, we find that there are several desirable properties inherent to the space of HBVs. We describe these properties in this section to provide some necessary background.

Consider the case of a binary vector space of 10,000 dimensions, containing a staggering number of unique vectors (exactly $2^{10,000}$). It is unlikely that any system will require so many unique vectors because it would not even have enough time to enumerate all possibilities. The space forms a very high-dimensional hypercube, with as many vertices as unique points, and it is apparent that the distribution of points (vertices) from the perspective of any one point (vertex) is always the same. This is akin to how a cube looks the same from the perspective of any corner. Thus, binary vectors are a way to create "arbitrary" representational vectors for computers.

Consider the Hamming distance (H) between two binary vectors $|a*b|$, where * is the bitwise XOR operation and magnitude is the number of 1's in the result. The number of components of disagreement between a and b is thus the distance between them. We can normalize this value to be agnostic to the dimensions of the vectors. If n is the dimension of the vectors, then H can be normalized to the range of values between 0 and 1 by dividing the result by n, referred to as normalized Hamming distance (H_n). By assuming that each binary vector is equally likely, it is clear that the average H_n between vectors is 0.5 (5000 bits), with an SD of 0.005 (50 bits), by a binomial distribution. The same is true for the distribution of distances from any one point to any other point. However, straying from the average distance of 0.5 is very difficult. Because each increment 0.005 is an SD, it is highly unusual for two random vectors to have a distance of 0.475 or 0.525, which is a whole 5 SDs. Even a distance of 0.333 is far too unlikely to be random. As a result, HBVs are quite tolerant to noise in measurement. A distance deviating a few SDs from 0 may as well be the same vector in most applications. Likewise, randomly drawn vectors are nearly guaranteed to be close to a distance of 0.5. As a consequence, HBVs are very good candidates for encoding various forms of information into constant-sized vectors.

Kanerva describes three particular operations that are well suited for encoding information:

1) The XOR operation: Because XOR is an involution when one operand is fixed, associative, and commutative, we have that $c * a = (a * b) * a = b$. We can recover a or b exactly if we have one or the other or approximately when noise is present.

2) The permutation Π: This permutes a vector x's components into a new order by computing the product Πx. If the permutation is randomly generated for a long binary vector, then the new binary vector is very likely to have a $H_n \approx 0.5$. We can represent Π as a permutation of index locations 1 to n. The product simply swaps components of x to the order in Π.

3) The consensus sum, $c_+(A)$, over the set of vectors A, or simply $x +_c y +_c z$: This sum counts 1s and 0s component wise across each element of A and sets the component to the corresponding value with the bigger count. Ties, only possible in a sum of an even number of elements, can be broken by randomly choosing 0 or 1.

Other options exist for encoding, but these three are of particular concern to this paper. Note that mapping by XOR or permuting preserves distances. For a given mapping a

$$H(a*x, a*y) = |a*x*a*y| = |a*a*x*y| = |x*y|$$

$$H(\Pi x, \Pi y) = |\Pi x * \Pi y| = |\Pi(x*y)| = |x*y|$$

The permutation example, in particular, is due to permutations distributing across XOR and the fact that permutations do not change the number of 1s or 0s in the result of an XOR.

With the three operations above, we can now represent more complex data structures entirely with HBVs:

1) A set can be represented as follows: Given $\{C_1, C_2,\ldots, C_n\}$ and a mapping of each element to HBVs $z_1, z_2,\ldots, z_n$, we encode the set as the XOR of each HBV or as

$$z = z_1 * z_2 * \ldots * z_n$$

No matter the order of each C_i, the representation will always be the same, thus imitating a set. There are some limits to this representation. For example, although testing equivalency is simple (check whether XOR is the 0 vector), testing membership and enumeration cannot be done without having another binary vector representation in place specifically for this. Furthermore, adding the same element has the effect of removing it, and care must be taken to prevent that from occurring.

2) Similarly, for an ordered pair (A, B), we can choose a random permutation Π and encode the pair as $c = \Pi a * b$, where a and b are the HBVs associated with A and B, respectively. The permutation Π thus encodes the data type of the ordered pair as well. Given c and knowing Π and one of the pairs, the other can be found.

3) Sequences can be interpreted as a succession of ordered pairs. Thus, a sequence $C_1, C_2,\ldots C_n$ is equivalent to

$$c = \Pi^{n-1} c_1 * \Pi^{n-2} c_2 * \ldots * c_n$$

where c_i is the corresponding HBV. The Π_i refers to a permutation that has applied itself to itself i times, before being applied to the HBV's components. Note that XORing two separate sequences will remove any patterns they both share. Furthermore, sequences can be shifted by permuting the vector again with Π to move further backward (to add a new part of the sequence), or these can be shifted forward in time by applying Π^{-1} or the depermutation.

4) Data records are referred to by Kanerva as a method for storing object-like properties. For example, a data record could consist of name, age, and sex. Each of these is given a random identifier r_i, and the data record format can be represented as a matrix $R = [r_1 r_2 \ldots r_n]$. To bind a value to each identifier, we can XOR the value with the corresponding identifier. Thus, given i values, v_i, a particular data record is represented as

$$R * V = [r_1 r_2 \ldots r_n][v_1 v_2 \ldots v_n]^T = r_1 * v_1 +_c r_2 * v_2 +_c \ldots +_c r_n * v_n$$

where the bracket notation defines a "matrix" of vectors, with multiplication of matrices after the typical notation, but with XOR and consensus sum replacing multiplication and addition of elements. To isolate the value of a field, we simply compute $R * r_i$ or $R * v_i$ if we wish to find the data type that a value is associated with. This is due to the fact that each term in the consensus sum will produce random noise (as distributing the XOR across the data record will yield randomness), but the term containing the value will be significant. You will get an approximately similar result, that is, the nearest neighbor will be, with high likelihood, the corresponding value or identifier.

With these simple structures, data records, sets, and sequences can be combined to encode and represent all manner of data in a consistent way.

Numerical values versus categorical values

One of the simplest and most necessary forms of data is that of raw numerical values. How should the distance H between vectors representing numbers correlate to their objective value? One simple answer is to directly embed the linear relationship between numbers in a higher-dimensional space as a line or a curve in that space. This is possible with binary vectors, although some care should be taken to ensure that these values make sense in their distances. For example, for a given value X, the values $X - 1$ and $X + 1$ should be nearest neighbors to X but on roughly opposite locations in space from X.

With categorical values, we are unsure of what the appropriate distance from them should be. For example, what is the distance between "*a*" and "*b*"? For this, we must rely on distributional semantics, where the distributional properties of how these values occur and co-occur in data resolve their semantic meaning toward each other. Values that appear in similar distributions of data are taken to be closer in meaning. It is clear that both numerical and categorical data are important in AI. The real difficulty in representing them comes into play when we consider that they have to exist in the same space.

Representing both numerical and categorical data in the same space

In previous work (*8*), a method for encoding multiple modalities was developed to both create HBVs for observations of data, using distributional semantics, and give a framework for forming more complex sequences and sets of differing modalities, with potential for their own distributed semantics. Although this is not the particular subject of this paper, it is worth remarking on this process because we used the numerical representations of it in our experiments.

Consider that you have multiple modalities, such as vision, audio, text, etc. Each of these has a finite number of unique values to represent an atomic piece of information. Consider the special case where the atomic piece of information does not actually have any distance associated with it. This is true for characters because any meaning ascribed to them is purely distributional (how these values are distributed in relation to others). Pixel intensity and related measures all have inherent meaning in their values, directly tied to the distance between each value (e.g., red channel intensity of 55 versus 123). Thus, we desire the ability to encode both distributional and nondistributional values in a framework for HBV representation of multiple modalities.

Taking characters as an example for convenience, we could find distributionally meaningful HBVs for the space of characters by the method described in (*8*). First, assign every character a randomly selected HBV. Then, obtain distributional counts of how often two characters may co-occur. The distributional counts could be naturally represented as a graph, where the vertices are characters and the directed edges are co-occurrences. This is similar to systems such as word2vec (*9*) or GloVe (*10*). Each character has a "mass" of how often it is seen and an edge weight determined by how often that occurrence occurs. A good choice of HBVs for each character would make sure that characters that co-occur often are nearer than those that do not. This is referred to as a geometric interpretation of semantics. To find such vectors, we can simulate a model that characterizes energy in the current system of HBVs. Let $X^{(k)}$ be a matrix of current HBVs, with m being rows per vertices and n being columns per bit length. The problem is formulated as

$$\arg\min_X \left(T\left(X^{(k)} + X \right) \right)$$

Here, T is a function that measures the total energy in the system described by a given matrix. Our goal is to find a perturbation of 1s and 0s, referred to as X, that minimizes the energy measure by T. Then, $X^{(k+1)} = X^{(x)} + X$ becomes our new set of vectors. We repeat this process until the energy is minimized. The energy is computed as

$$T(A) = \sum_{i}^{m}\sum_{j}^{n}\max\Big(F_{\text{conn}}(A,i,j) + F_{\text{prox}}(A,i,j), 0\Big)$$

This formula expresses the sum of all unresolved forces in the matrix A, given a graph of co-occurrences. The functions F_{conn} and F_{prox} represent the connective and proximal forces of the system. Together, they enumerate the force being applied on the bit at row i and column j to switch its value. The connective force for a vertex is the resultant force across all co-occurrence pairs of that vertex. An edge's weight (co-occurrence likelihood) dictates its force. The connective force is computed by the following equation, where M_i is the relative mass of vertix i and $W_{i,k}$ is the edge weight between vertices i and k (note that the weight is 0 if a connection does not exist)

$$F_{\text{conn}}(A,i,j) = \sum_{k\neq i}^{m} M_i W_{i,k} C_{\text{conn}}(A_{i,j}, A_{k,j})$$

The proximal force is caused by the proximity of a vertex to a vertex that it shares a co-occurrence edge with. The force that each vertex generates scales with its mass (occurrence likelihood). The proximal force is computed by the following equation, where $H_n(a, b)$ is the normalized Hamming distance between vertices a and b

$$F_{\text{prox}}(A,i.j) = \sum_{k\neq i}^{m} \frac{M_i M_k}{H_{n(A_i,A_k)}} C_{\text{prox}}(A_{i,j}, A_{k,j})$$

The functions C_{conn} and C_{prox} are simply functions that determine whether the bit in question wants to change or not, given its current state and the state of its incident vertex

$$C_{\text{conn}}(a,b) = \begin{cases} -1, \text{if } a = b \\ 1, \text{if } a \neq b \end{cases} \quad C_{\text{prox}}(a,b) = \begin{cases} 1, \text{if } a = b \\ -1, \text{if } a \neq b \end{cases}$$

Techniques exist for minimization of T and are stochastic in nature, involving gradual flipping of bits to minimize energy, often using simulated annealing. As shown in (*8*) and here again in Fig. 1, for convenience, the energy of the same system generated by random graphs of equal number of vertices with and without proximal force only reaches exact 0 when the proximal force is off.

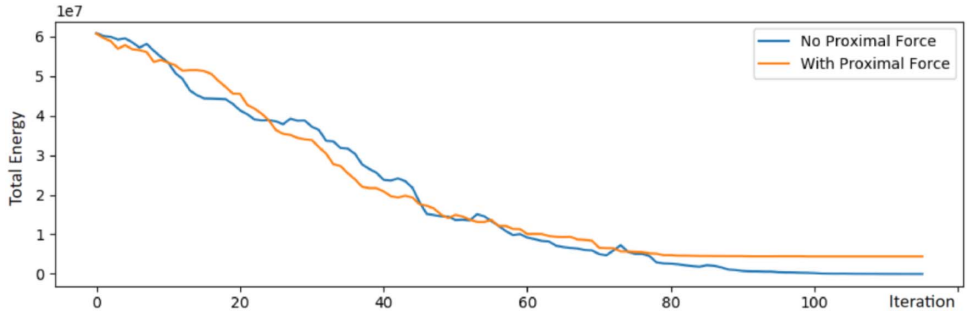


Fig. 1. Tension minimization. An example of how tension is minimized in a graph represented the co-occurrences of vertices. As the total energy in the system decreases with each iterations, the positions of the HBVs associated with each vertex are in geometrically "better" locations, relative to each other. When proximal force is turned off, the system is enabled to reach a singularity state, and thus, the tension will naturally reach 0. However, when the proximal force is present, it is unlikely that a state that fully minimizes the oscillations will be found, and the minimization will converge at a nonzero solution.

Encoding images as HBVs

To be able to encode more complex structures as HBVs, such as images, care must be taken to preserve the meaningfulness of both the values of pixel intensity and location. How can this be done with the basic structures we can encode? First, we describe how pixel intensity is represented with HBVs. Consider the case of a single-channel, gray-scale image. We have 256 unique values across the space of intensities, all evenly spaced from one another. To reconstruct this with vectors, we first require 256 vectors and then require them to be spaced such that the nearest neighbors of a particular intensity vertex are the closest intensities in reality. Likewise, intensities that are further away in value should have a further H or H_n distance proportionally. A proportionally similar distance should be found for pixels in lesser or greater intensities. Essentially, the intensities, which are naturally visualized as a 1D line, are embedded into a higher-dimensional space as a curve or a line of vectors. We can visualize this by taking a particular vertex and connecting it to each other vertex, requiring that the edge weights decrease proportionally to the difference in intensity between the vertices, evenly spaced out. Applied to every vertex, we arrive at the realization that a fully connected graph (apart from vertices connected to themselves) is required. Thus, every vertex should have a vector that satisfies the properties of the intensity scale. Figure 2 shows how that this is visualized on a small scale. When this graph is minimized with the technique described in the previous section, the result forms a distance matrix similar to the one shown in Fig. 3. As we can see, the distances increase away from the diagonal entry, similar to what intensities do in a single channel, from the perspective of a particular intensity value.

With representations for the intensities, we can now focus on representing location. Permutations hold the positional semantics for sequences of HBVs; a particular permutation is attached to a sequence of a single data type. This is movement through sequences in a one directional sense. For 2D data such as images, we require two permutation operations, one for the row and one for the column location. For each location, we permute the intensity representations appropriately. Consider a single such pixel; we can place it in the proper location in an image by permutations before XORing with other pixels—the concept of which is shown in Fig. 4. As a small example, consider a 3 by 3 image I. Let I_{ij} be the pixel intensity representation at row i and column j. Then, the HBV representation of the image is constructed as follows

$$\begin{cases} (X_{00} * CX_{01} * C^2X_{02}) * \\ R(X_{10} * CX_{11} * C^2X_{12}) * \\ R^2(X_{20} * CX_{21} * C^2X_{22}) \end{cases}$$

This simple but powerful formulation allows us to encode an image of arbitrary dimensions in meaningful ways. Similar to their role in sequences, permutations of the resulting HBV will shift the image in space accordingly, and entire sections of the image can be removed or concatenated with others by XORing with the sections to be removed from and to be added to the full image as shown in Fig. 5.

Creating memories with HBVs

With the ability of HBVs to combine with other HBVs to encode more and more information but in the same vector length, we have a natural method to create semantically significant and informed memories. As an example, a series of images can be "remembered" by forming a data record with HBVs of each image, where a particular vector is taken to signify the location in time of the image. When presented with a particular image, its existence inside of a memory can be checked by simply XORing the memory with the image. The nearest neighbor of the result will be the location in time of the image. Likewise, when interested in the image at a particular location, the same process will reveal what the nearest matching image will be.

Using neuromorphic visual information

The DVS (*11*) is one of the few currently available bioinspired (also often referred to as "neuromorphic") vision sensors. This camera is an asynchronous differential sensor: Each pixel acts as a completely independent circuit that tracks the light intensity changes. As soon as the light intensity changes by a certain predefined percentage (called sensitivity), the pixel sends the current 32-bit timestamp (with the clock synchronized across all independent pixels) to the common asynchronous bus. The sent data record—a triple (x, y, t), where x and y are pixel coordinates and t is the timestamp—is called an event.

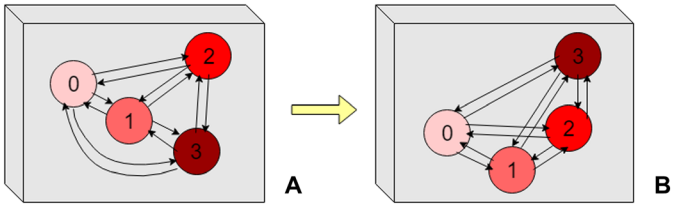


Fig. 2. Intensity minimization. A visualization of intensity minimization. We start with random vectors for four intensities ranging from 0 to 3 (**A**). A complete graph is formed, connecting each vertex to each other, with decreasing weight in proportion to the intensity difference. When minimization is performed, the result forms a line or semicircle of appropriate distances (**B**).

Given this, one can see the challenges in processing event-based data: The video "stream" contains no classical frames, but instead it is a stream of 3D events. Moreover, the events can only be triggered by light intensity changes, which means that they only happen on the object edges or textured regions. Nevertheless, the event-based sensors are not devoid of advantages—the extremely high temporal resolution of DVS allows capturing an equivalent of roughly 10,000 frames per second and literally seeing the motion of the objects, with their trajectories being smooth and continuous, not interrupted by the frame rate. DVS and similar sensors are becoming increasingly valued in the robotics and vision community, and some of the recent works (*12*, *13*, *14*) have described methods of processing event data. We argue that event-based neuromorphic cameras are indispensable for motion-related tasks (such as navigation, obstacle avoidance, and tracking), where traditional appearance-based vision requires costly frame-to-frame matching, and the lack of intensity information does not prevent motion pattern analysis.

We extracted the motion information from the event timestamps by building a "time image" (*13*): In a given time slice of the (x, y, t) space (that is, for a small interval of time), the events are projected on the image plane and the timestamps of the events that fall on the same pixel are averaged out, so the pixel intensity is a function of time t. This representation has several benefits: It is easy to compute, is robust toward noise, allows for asynchronous data analysis (because the time image can be computed at any moment in time), and preserves the motion information stored within the event stream. An example of the time image, together with the corresponding "classical" frame, can be seen in Fig. 6.

Learning from event stream classically versus with HBVs

To test how well visual data can be encoded by HBVs when compared to classical techniques, we set up an experiment where a classical convolutional neural network (CNN) architecture, inspired by NVIDIA PilotNet (*15*), was trained on the DVS event stream to predict velocities. The input layer of the CNN accepted time images as described in the previous section, essentially interpreted as a single-channel image. Both the event stream and velocities were collected via an autonomous robotic platform that was tracked by the VICON Motion Capture System. The magnitudes of the velocities were in the order of 1 m/s, and the DVS camera experienced 6D motion. With this setup, the CNN was able to achieve an accuracy of up to 7 cm/s, amounting to 7% of the velocity magnitude, despite the sparseness of the event data.

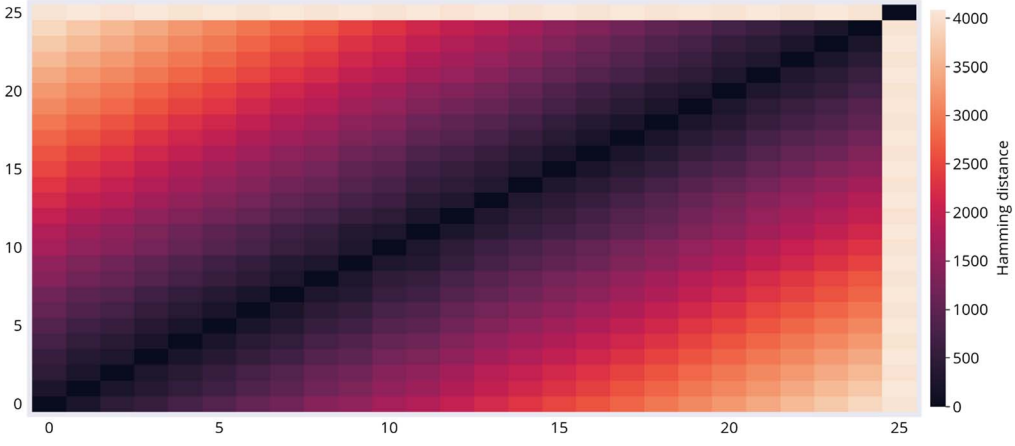


Fig. 3. Hamming distance visualization of intensities. Visualization of the Hamming distance between intensity values in the range of 0 to 25. Note the increasing distances away from the diagonal, as one would expect. For DVS or images, a full 256-intensity range can be easily developed with the same properties.

Likewise, the time image data were also converted into HBV encodings in a fashion similar to how image codes are constructed. This was achieved by first constructing an intensity space consisting of 255 HBVs of length 8000, each corresponding to one of the 255

intensity values of the time image and minimizing it to generate appropriate HBV representations. When visualized, this space is similar to what is shown in Fig. 3 with 25 intensity values. We only used 255 HBVs because the intensity value of 0 indicates no event and thus does not need to be encoded into the HBV. These representations are then used to generate sequences encoding the pixel intensities and their locations in each time image into a single HBV. Because of the fact that bits in HBVs do not necessarily correlate to neighboring bits, it does not make sense to use a similar CNN-based network for HBVs as we did in our first experiment. As a result, a fully connected, six-layered neural network was used with a similar number of parameters. When trained, HBV-based learning achieved an error of about 9 cm/s. Although slightly worse than the CNN, this result is impressive because the size of the input was naturally compressed as HBVs, from the DVS's resolution of 346 × 260 (about 90,000 pixels) to 8000 bits. Inspired by these results, we then investigated the HBV's ability to encode sequences of frames as data records and binding the velocity vectors to the frame vectors.

Perception to action binding with HBVs

The HBV vectors allow any amount of data of any type to be represented in a fixed length of bits, up to the informational capacity provided by the binary space. We would expect HBVs to be able to create data records from different modalities to bind them together in a single representational space—ideally, binding perception to action as a single data record. We constructed an experiment where, given the visual input, the system was able to remember the action it needed to take in the form of a velocity vector.

First, we describe how to represent a 3D velocity vector as an HBV. We construct HBV representations of each component individually, that is, we run vector space minimization three times with different starting seeds for random HBVs. We then create a data record from the three spaces for a single velocity. To do this, we must select three random binary vectors as identifiers for the *X*, *Y*, and *Z* components of velocity to construct the data record with. We will refer to these as *X*, *Y*, and *Z*. Because these are random, they are nearly unrelated, with distances close to 4000 bits from each other (for consistency, we use 8000-bit vectors in all our experiments). When component values are bound to the identifiers, they each create three separate uncorrelated HBVs. This velocity data record is extended with an entry for the associated time image HBV encoding. Likewise, this encoding is bound to a randomly selected identifier for time images, referred to as *T*. The resultant data record thus exists in an action perception space and can be considered a memory.

To create many such memories, we take a DVS recording with corresponding motion ground truth and create a memory of each time image and velocity. Our experiment: Given a time image, find the correctly associated velocity using only memories. To isolate the time image in the data record of the memory, we XOR *T* with the memory *m* and find the nearest neighbor to the input time image encoding. Afterward, we XOR the candidate memory with *X*, *Y*, and *Z* to retrieve individual velocity components. In all our experiments, we achieved 100% accuracy. This means that the information capacity of the data record with 8000 bits far exceeds the requirements to store the time image and velocities encoded within.

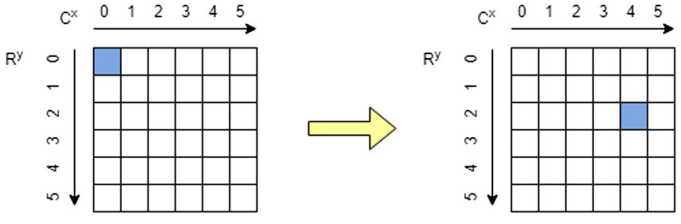


Fig. 4. Moving pixels spatially. A pixel *p* is relocated from the origin position to row 2 and column 4 by performing a permutation R^2C^4p.

Information capacity of HBVs

A natural next experiment would be to see how much information can fit into 8000-bit vectors. Because each time image contains quite a bit of information, we chose to encode entire sequences of time images in a single data record, with randomly generated HBV identifiers T_i tied to each frame position in the sequence. We then tested the ability of data records to localize the given image in the memories containing only time image sequences.

Figure 7 shows our findings. We conducted the experiments for sequences of up to 700 frames in length and measured the Hamming distance of the input frame HBV to the closest match in memory of sequences. Naturally, the longer the sequence, the more information is contained within every data record. Thus, we expect that the Hamming distances will grow even closer to 4000, which corresponds to completely uncorrelated vectors. Our results confirm this trend and show that we can safely code up to around 200 frames without worry of false positives. However, the likelihood of errors is small even after this point. In conclusion, a single 8000-bit HBV can provide for huge information density, without needing any learning.

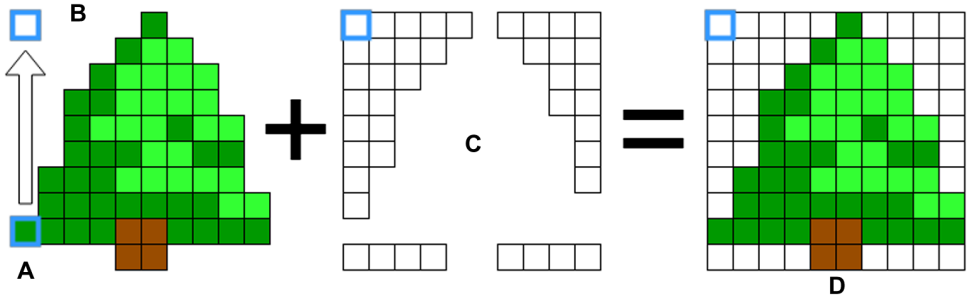


Fig. 5. Composing image encodings. A simple example of how an image encoding can be manipulated. The reference pixel can be arbitrarily moved through permutations as shown in (**A**) by permutating the current encoding appropriately [in the case of (**B**), this is a permutation of R^{-8}]. A new image (**C**), aligned on a similar reference pixel (the origin here); the two images can be composed to construct an arbitrary image (**D**).

Theoretical limits on capacity of HBVs

Although we have empirical results on how well information can be encoded into HBVs, we can arrive at more practical bounds for this in relation to the accuracy of recall. Regardless of the length of the HBVs, the odds of a particular bit being 0 or 1 is 0.5, assuming that the vectors are randomly generated. Suppose we have performed a consensus sum on a number of data points using random vectors as a basis for identifying each data point (i.e., a data record). The result can be thought of as a memory consisting of

these data points. Let m be the memory, then (with the right-most side as shorthand notation)

$$m = x_1 * a +_c x_2 * b +_c x_3 * c \ldots = x_1 a + x_2 b + x_3 c + \ldots$$

When we test for the presence of a particular value, say, d, we first compute $m * d$. The result should be close to a particular x_i, if d is present. Naturally, to determine this, we compute the distance H. This involves another XOR with each x_i individually. Consider the following example

$$\begin{aligned} |\, m * d * x_1 \,| &= |\, x_1 a d x_1 + x_2 b d x_1 + x_3 c d x_1 + \ldots \,| \\ &= |\, a d + x_1 x_2 b d + x_1 x_3 c d + \ldots \,| \end{aligned}$$

When computing the H_n distance, the above becomes the probability of each bit being 1 in this vector. Because we are computing a consensus sum, the above is equivalent to tossing a coin the same number of times as you have terms in the sum and setting the consensus to whichever side landed more often. This is true because we know that, if the vectors are random, then the XOR of two random vectors yields a random vector. However, if $a = d$, then the term $ad = 0$ or the zero vector. This would create a bias; one of our coin tosses will now always be the same. More generally, let the probability of 1 for the term ad be p. This bias is what causes our consensus sum to give a H_n that is different in a statistically significant way than the expected value of 0.5. Given n vectors in a consensus sum and p, the probability for a bit in $|\, m * d * x_1 \,|$ being 1 is

$$(1-p)\sum_{k=n/2}^{n-1} \frac{(n-1)!}{2^{n-1}k!(n-k-1)!} + p\sum_{k=n/2-1}^{n-1} \frac{(n-1)!}{2^{n-1}k!(n-k-1)!}$$

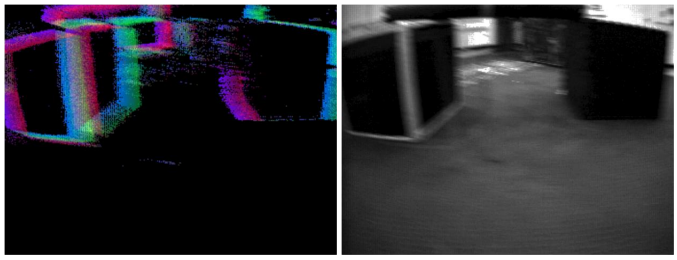


Fig. 6. An example of DVS data visualization. On the left, a time image is shown, in which green is the average timestamp, and the positive/negative per pixel event counts are shown in red/blue. On the right, the corresponding gray-scale image of the scene is shown. Note the motion blur on the classical frame.

Sensorimotor representation through binding visual and motor information

To demonstrate an example application of our framework, we applied it to the task of ego-motion estimation in the autonomous vehicle setting. We used the multivehicle stereo event camera (MVSEC) (*16*) dataset for our experiments. The MVSEC dataset features DAVIS240 event recordings together with ground truth velocity, taken from a car driving during the day and night. The MVSEC dataset has recently gained popularity for evaluation on event-based self-driving car data. We formed a separate memory for each degree of freedom (rotation and x/y translation) using the first 15 s of the "outdoor day 1" sequence and then demonstrated the performance on the rest of the sequences (two during the day and three during the night). We compute the memory as follows, for each degree of freedom

$$m = v_1(a_1 + a_2 + \ldots) + v_2(b_1 + b_2 + \ldots) + \ldots + v_n(z_1 + z_2 + \ldots)$$

Here, the vectors v_i are basis vectors for our velocity space, that is, every vector represents a particular range of velocities with a certain step. In our experiments, 0.001 was used as the step. The distinct letters in $a_i, \ldots, z_i$ represent groups of images by their ground truth velocity class. Conceptually, the memory m is just a consensus sum of every image vector bound to every corresponding 1D velocity represented as an HBV. See text S1 for more details on how the image vectors and velocity vectors are formed in this experiment.

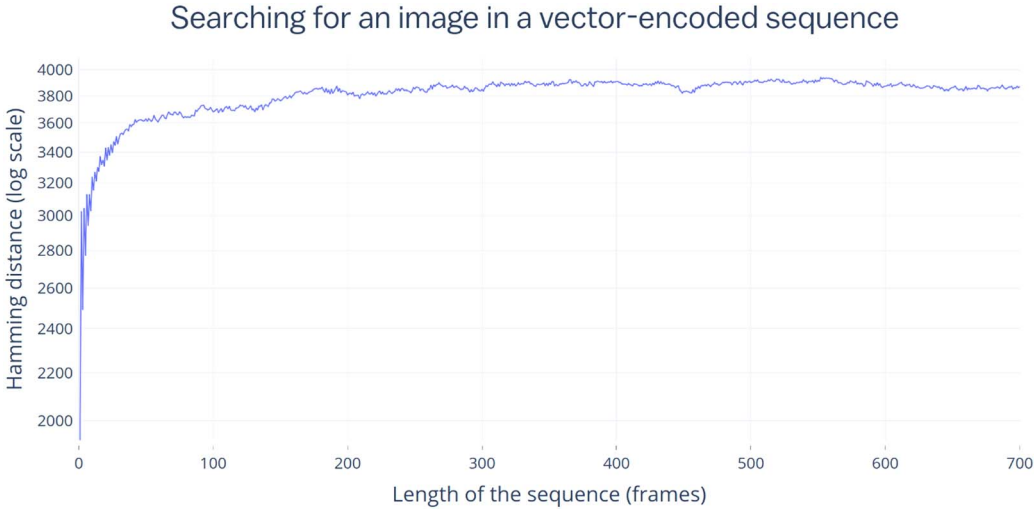
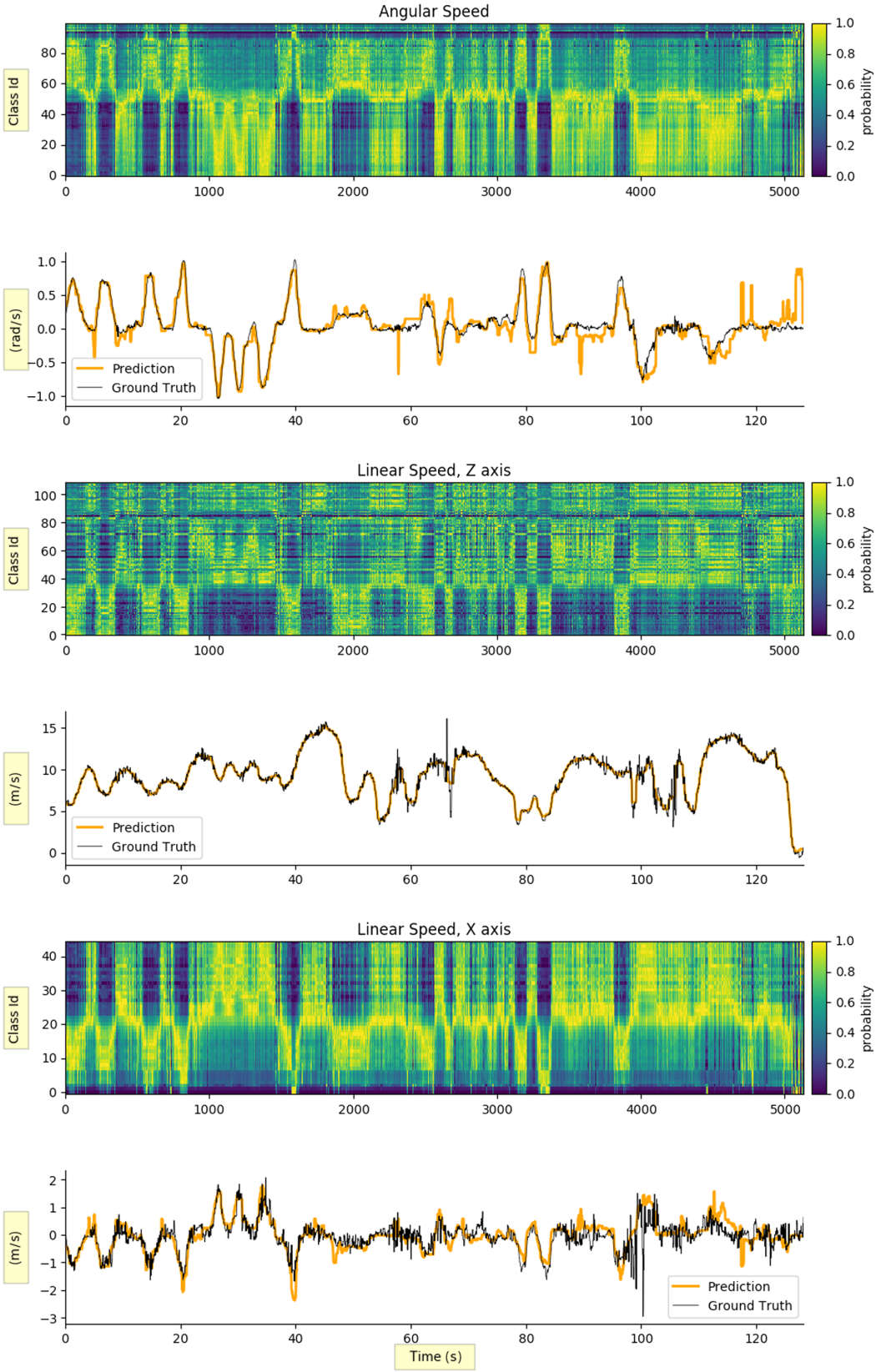


Fig. 7. Hamming distance decay in data records. The decay in nearest neighbor matches of Hamming distance as the number of time images in a memory increases. As more frames of motion are put into the encoding, more noise is introduced, making the deviation caused by a match less and less statistically significant. At 700 frames, we are still about 3 to 4 SDs of likelihood away from random vectors.

Similar to the "Theoretical limits on capacity of HBVs" section, given an unseen image d, we compute the probability of it being associated with each of the basis velocity vectors v_i. We then choose the v_i yielding the highest probability, equivalent to the smallest H between mv_i and d.

The MVSEC dataset has been used previously in a number of neural network-based estimation algorithms. Typically, learning approaches (*14*, *16*, *17*) use the "outdoor day 2" sequence (12,196 ground truth samples) for training and test on outdoor day 1 (5134 ground truth samples), trained for 30 to 50 epochs. We want to emphasize the ability of our pipeline to train in a single pass in our



AQ77–AQ79 **Fig. 8. Results for HBVs on outdoor day 1 in the MVSEC dataset.** Qualitative results on the outdoor day 1 subset of the MVSEC dataset. Results are split into the angular and linear speed inference. The probability plots show the likelihood of each velocity class being correct across all inferences, with a light-green color indicating the higher probabilities. One can see that the light green forms the path of predictions across the probability field.

Table 1. Quantitative results for the MVSEC dataset across the five subsets. The table gives the number of frames in each subset, length in seconds, the size of the angular and linear bins used for HBVs, number of vectors in the rotation, *X* and *Z*, and the average endpoint error (AEE), average relative pose error (ARPE), and the average relative rotational error (ARRE).

MVSEC subset	Outdoor day 1	Outdoor day 2	Outdoor night 1	Outdoor night 2	Outdoor night 3
Frames	5134	12196	5133	5497	5429
Length (s)	128.325	304.875	128.3	137.4	135.7
Ang. bin (rad/s)	0.02	0.02	0.02	0.02	0.02
Lin. bin (m/s)	0.08	0.08	0.08	0.08	0.08
Rotation (clusters)	104	101	40	74	87
X (left-right, clusters)	47	44	24	40	33
Z (front-back, clusters)	119	311	244	251	228
AEE	0.810	1.03	0.933	1.16	0.940
ARPE	0.122	0.225	0.243	0.0948	0.0831
ARRE	0.0994	0.108	0.0632	0.116	0.121

experiments, with only 500 samples, taken from outdoor day 1, while producing comparable results on this and other sequences. Neural approaches tend to use a much higher ratio of training to test set. We present the qualitative results of our ego-motion estimation in Fig. 8 on outdoor day 1, as well as the other subsets, with detailed analysis and explanation in text S1. We also provide quantitative results in Table 1. The binning of velocity sizes is directly proportional to our inference time. For 500 velocity classes, we perform at an average of 572 inferences per second (time slices per second). If we halve the number of velocity classes (250), then the inference rate approximately doubles (1084). Training time is extremely fast and proportional to the number of training frames, requiring only 0.5 and 2 s for 500 and 1500 frames, respectively. Retraining on the fly is possible, and training can be done in an online fashion. We received these results on a regular laptop with central processing unit, running at 3 GHz, on Pythonic code (apart from the pyhdc library).

DISCUSSION

Our results show promise for the use of HBVs in machine learning and AI. It gives an alternative method to normalize what the input of a learning system should look like and is capable of being agnostic to the data modality. We have shown that a learning system can still learn from HBVs, although they inherently hide the raw data in their encodings. We surmise that the reason this is possible is that the distributions of 1s and 0s in HBVs have very stark, statistical properties, from which a learning system can still learn. Furthermore, HBVs can be used to store memories of all kinds, including those both action and perception, as shown in our results. Moreover, the tolerance of this system to how much data can be encoded while still having statistically significant Hamming distances is impressive. Furthermore, combining this with the neuromorphic DVS camera shows that hyperdimensional perception can be achieved with even nonstandard data representations, such as event clouds.

We believe that further investigation must be done into the realm of motion-based vision with HBVs and the potential to be used in regular vision as well. Further research must be done on the power to integrate multiple forms of perception with both neuromorphic sensors and classical vision. One of the main weaknesses of the use of HBVs lies in the density of the data to represent. For example, traditional vision is far better suited to the traditional vision techniques than HBVs currently are, because of the density of information provided by traditional, frame-based RGB cameras. The prevalence of CNNs and their power to learn *N*-dimensional data is also another weakness of HBVs, which as of yet have no such formulation for processing data in the method that window-based convolutions provide. It is theoretically possible to imitate CNN window-based processing of HBVs; however, it seems that appropriate neural networks/alternate learning structures need to be discovered to make full use of the power of HBVs. These areas will be subjects of future work in the still early field of HBVs.

MATERIALS AND METHODS

To perform our experiments, we have developed an open-source library for Python for accelerated manipulations with long binary vectors (pyhdc). The library supports vector lengths of up to ~8000 bits and is capable of performing 1.4×10^5 permutations per second and 3.0×10^{-7} XORs per second on a modern laptop in a single thread. The code is available online: https://github.com/ncos/pyhdc.

In our first experiment, we used the CNN with an architecture similar to (*15*) but replacing the output layer with a three-node fully connected layer instead of one to account for three dimensions in the velocity vector. The fully connected neural network used with HBVs has 8000, 4000, 1000, 200, 50, and 10 nodes in its dense layers. For both networks, the L2 norm was used as a loss function, with the Adam optimizer set to a learning rate of 0.001.

In all experiments, we used a dataset collected using DAVIS 346b and DAVIS 240b sensors (with event resolutions of 346 by 260 and 240 by 180, respectively). Only the DVS parts of the sensors were used, whereas the corresponding gray-scale images were ignored. To record a reliable and realistic dataset, we used a customized Qualcomm Flight quadrotor platform fitted with one of the cameras (fig. S6). The platform was tracked indoors via the VICON Motion Capture System, capable of providing a positioning resolution of up to 0.3 mm at an

update rate of 200 Hz. About 17,000 frames of the dataset were used in our experiments, with 10% used for validation.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/4/30/eaaw6736/DC1
Text S1. MVSEC experimental details.
Fig. S1. Theoretical likelihood of a consensus term to be 1.
Fig. S2. Results for HBVs on outdoor day 2 in the MVSEC dataset.
Fig. S3. Results for HBVs on outdoor night 1 in the MVSEC dataset.
Fig. S4. Results for HBVs on outdoor night 2 in the MVSEC dataset.
Fig. S5. Results for HBVs on outdoor night 3 in the MVSEC dataset.
Fig. S6. Drone used in dataset collection.
Fig. S7. Memorization pipeline.
Fig. S8. Memory retrieval pipeline.
Movie S1. Experimental drone and dataset.
Movie S2. HBV representations for intensities.
Movie S3. Encoding and memory binding.
Movie S4. Tension relaxation.
Movie S5. Outdoor day 1.
Movie S6. Outdoor day 2.
Movie S7. Outdoor night 1.
Movie S8. Outdoor night 2.
Movie S9. Outdoor night 3.

REFERENCES AND NOTES

1. R. Bajcsy, Active perception. *Proc. IEEE* **76**, 966–1005 (1988).
2. Y. Aloimonos, *Active Perception* (Psychology Press, 2013).
3. J. Aloimonos, I. Weiss, A. Bandyopadhyay, Active vision. *Int. J. Comput. Vis.* **1**, 333–356 (1988).
4. D. H. Ballard, C. M. Brown, Principles of animate vision. *Comput. Vis. Image Underst.* **56**, 3–21 (1992).
5. D. Ballard, Animate vision. Artificial Intelligence, 48, 5786. *Guest Editorial: Qualitative Vision* **117**, (1991).
6. R. Bajcsy, Y. Aloimonos, J. K. Tsotsos, Revisiting active perception. *Auton. Robots.* **42**, 177–196 (2018).
7. P. Kanerva, Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognit. Comput.* **1**, 139–159 (2009).
8. P. Sutor, D. Summers-Stay, Y. Aloimonos, A computational theory for life-long learning of semantics, in *International Conference on Artificial General Intelligence* (Springer, 2018), pp. 217–226.
9. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space. arXiv:1301.3781 (7 September 2013).
10. P. Jeffrey, R. Socher, C. Manning, Glove: Global vectors for word representation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 25 to 29 October 2014 (Association for Computational Linguistics, 2014), pp. 1532–1543.
11. C. Brandli, R. Berner, M. Yang, S.-C. Liu, T. Delbruck, A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid State Circuits* **49**, 2333–2341 (2014).
12. A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, D. Scaramuzza, Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. *IEEE Robot. Autom. Lett.* **3**, 994–1001 (2018).
13. A. Mitrokhin, C. Fermüller, C. Parameshwara, Y. Aloimonos, Event-based moving object detection and tracking, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 1–9.
14. C. Ye, A. Mitrokhin, C. Fermüller, J. A. Yorke, Y. Aloimonos, Unsupervised learning of dense optical flow, depth and egomotion from sparse event data. arXiv:1809.08625 (25 February 2019).
15. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, End to end learning for self-driving cars. arXiv:1604.07316 (25 April 2016).
16. A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, K. Daniilidis, The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robot. Autom. Lett.* **3**, 2032–2039 (2018).
17. A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, T. Delbruck, EV-IMO: Motion segmentation dataset and learning pipeline for event cameras. arXiv:1903.07520 (18 March 2019).

Funding: We acknowledge the support of the National Science Foundation under grant BCS 1824198, ONR under grant N00014-17-1-2622, and the Northrop Grumman Mission Systems University Research Program. **Author contributions:** All authors contributed to the study's conceptualization, investigation, methodology, and visualization and wrote and edited the manuscript. P.S. and A.M. analyzed the data and wrote the software. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials.

Submitted 15 January 2019
Accepted 18 April 2019
Published 15 May 2019
10.1126/scirobotics.aaw6736

Citation: A. Mitrokhin, P. Sutor, C. Fermüller, Y. Aloimonos, Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Sci. Robot.* **4**, eaaw6736 (2019).

Science Robotics

Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception

A. Mitrokhin, P. Sutor, C. Fermüller and Y. Aloimonos

Sci. Robotics **4**, eaaw6736.
DOI: 10.1126/scirobotics.aaw6736