DEVELOPMENT OF FEATURE DESCRIPTORS FOR EVENT-BASED VISION SENSORS

THESIS
Submitted in partial fulfillment of the requirements of
BITS C421T/422T Thesis

by

Varad Gunjal
ID No – 2009A3TS146G

Under the supervision of
Prof. Tobi Delbruck

Birla Institute of Technology and Science,
Pilani – K.K. Birla Goa Campus

Date - 18/12/2012

THESIS ABSTRACT

Dynamic computer vision & machine vision applications like cell-phone imaging applications and mobile robotics require low latency of input, real time processing of the same and low power consumption to ensure smooth functioning in a real-world environment. However, due to the nature of traditional frame-based imaging architectures, achieving low latency using frame-based imagers requires specialized hardware and immense computational power, which can compromise cost effectiveness and might be infeasible in certain scenarios. Hence, using customized low latency vision sensors for the purposes of machine vision presents a feasible alternative.

A suitable example of such a low latency, low power vision sensor is the event-driven 128x128 pixel CMOS vision sensor, DVS128, which responds to changes in log intensity in the observed scene. It is modeled on the working principles of the human retina and has high temporal resolution.

It is proposed to address the classical machine vision issues of real time object recognition & scene correlation using the silicon retina. For these purposes, we adopt the popular feature-based scheme, and develop an event-based detection and description scheme to explore object recognition with event-based sensors. A representation analogous to frames i.e. providing access to static scene content, is developed using information from the events detected by the retina. We present various basic schemes for address-event to frame conversion and, then, develop detectors to identify regions of 'interesting' activity, based on approaches developed in SIFT & FAST. Descriptors inspired by those developed in BRIEF & BRISK, are then built around these detected keypoints. Finally, the performance of each of the detectors & descriptors is evaluated for event-based representations with respect to accuracy. We present only qualitative results of the implemented matching as a proof of concept. All descriptors in the recorded reference scene find at least one corresponding match in another recording of the same scene. Obtaining quantitative results is out of the scope of this project.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I am deeply grateful to my thesis supervisor Prof. Dr. Tobi Delbruck, Group Leader of the Sensors Group, Institute of Neuroinformatics (INI), UZH-ETH Zurich and my immediate supervisor, Christian Brändli, PhD student in the Sensors Group, INI, UZH-ETH Zurich for giving me the opportunity to pursue my Bachelors Thesis in their group at INI and for all their help, guidance and immense patience during this period. I would also like to thank all the people at INI for accommodating me and their support during the tenure of my thesis.

CERTIFICATE FROM SUPERVISOR

This is to certify that the Thesis entitled, _Development of Feature Descriptors for Event-based vision sensors_ is submitted by _Mr. Varad Gunjal_ ID No ._2009A3TS146G in partial fulfillment of the requirements of BITS C421T/422T Thesis embodies the work done by him/ her under my supervision.

Signature of the supervisor
Name: Prof. Dr. Tobi Delbruck
Designation: Group Leader, Sensors Group, INI, UZH-ETH Zurich
Date_____

CHAPTER – I: INTRODUCTION

Vision is the main sensory perception, to gather information about the surroundings, in human beings. Due to the high dimensionality, long range and high resolution of image input, devices are also enabled with vision sensing. Recent advances in research and technology have generated the need to realize robust, low latency device-based vision perception and response capabilities. Such applications include autonomous robots for purposes like exploration, reconnaissance and as controllers on shop floors, automated decision processes in industries, interactive applications on mobile devices, surveillance & security etc. Thus, computer vision and machine vision are quite the focus of numerous research efforts.

Most machine vision scenarios demand working within a rigid framework of real-time processing, low latency in response to stimuli, low power consumption, minimal payload etc. to ensure smooth functioning. However, an overwhelming majority of the existing popular algorithms for these purposes is based on traditional frame imagers, which simply capture the scene being observed at a constant frequency, generating large volumes of redundant data. Thus, although the algorithms perform their functions in principle, due to the structure and implicit readout scheme of frames, achieving real-time performance with low latency response requires using special hardware viz. high frame-rate cameras and enabling with processors that deliver high computational performance so as to handle the high speed frames and the computations involved thereon. Despite the consequences of Moore's law and availability of such sensors and processing muscle, this approach remains a brute force solution. It has several drawbacks – namely, increased power consumption due to the large number of computations (which can be potentially detrimental to the life & performance of the system), increased costs etc. and is, clearly not the best suited solution to the problem.

Computer vision, at its core, attempts to mimic the functions of biological vision. However, in comparison, the frame-based approach adopted bears no resemblance to the scheme of biological vision. Biological vision has no concept of redundantly sampling the observed scene at a continuous frequency irrespective of change in the scene, and, in fact, performs all analysis with very low energy dissipation. Customized low latency vision sensors that mimic the functioning principles of the human retina have been developed. Hence, we propose to use such sensors and develop algorithms to perform the required vision tasks to suit their architectures. In this project, we use the DVS128 event-based vision sensor which has the salient features of low latency, low power consumption and high temporal resolution. These qualities can be harnessed to effectively address a certain sector of the complete spectrum of possible machine vision application – namely, those that focus on developing appropriate vision capabilities in a dynamic environment viz. autonomous robotics.
Computer vision tasks can be broadly classified into two categories – Recognition & Tracking. The high temporal resolution of the silicon retina gives us a very elegant solution to performing

tracking. Hence, under this domain, our objective is to work towards developing algorithms for object recognition using the event-based sensors and presenting a proof of concept for the same.

The motivation behind this research effort is to address recognition using feature-based schemes in the computer vision domain, for event-based vision sensors.
Initially, in order to develop a strong foundation for building the approach, existing popular algorithms in this domain viz. SIFT, SURF, BRISK etc. are studied in Chapter II, to understand the construction of the feature-based schemes and the mathematics behind how they perform the required functions. The event-based sensor hardware and pixel architecture is discussed in Chapter III. Examples of previous implementations are also discussed to gain an insight into the nature event-based information. Since identification of features of objects require access to static scene content, some form of persistence or continuity is required to perceive objects. However, the event data does not display these qualities. Thus, developing a novel representation that provides access to static scene content based on the event information is essential – the approaches developed in this regard are detailed in Chapter IV. Feature detection schemes for event-based sensors, built upon approaches developed in SIFT & FAST are presented in Chapter V. In this effort, we focus on implementing binary comparison-based descriptors. Since computer vision research in this domain is yet to yield an optimal pattern for feature description, we generate random patterns around keypoints and use the same for description. These descriptors are documented in Chapter VI. Finally, keypoint matching performance for observed scenes using various detectors and descriptors is presented in Chapter VII. This is presented simply as a proof of concept and documented in a qualitative sense. Quantifying the performance of our schemes is out of the scope of this effort, due to the lack of existing databases to perform standardized tests on. Finally, some footnotes regarding possible further work in this domain are presented.

CHAPTER – II: A STUDY OF EXISTING IMAGE PROCESSING PARADIGMS

Classically, research in the field of computer & machine vision has developed in an attempt to emulate the functions of biological vision by amalgamating processes and representations that comprise vision perception. The major functions of the human eye can be broadly classified into 2 categories – recognition & tracking. Any interaction with the environment is built upon input from these functions. Hence, computer vision aims at addressing these 2 problems and obtaining optimal solutions for the same before proceeding to implement learning mechanisms for a complete realization of biological vision systems.
In this research effort, we are interested in addressing the problem of recognition. Robust image correlation & object recognition has been one of these main objectives of the field of traditional computer vision & image processing. An implicit requirement for performing object recognition is temporal coherence of the light signals reflected from the object and a persistence of the scene for a critical amount of time – to ensure enough static scene information is collected for processing. A traditional frame exposure captures static scene content, satisfying both these criteria and, hence, can be used easily for performing object recognition.

For such correlation of 2 image views of the same scene, taken from different perspectives, there have, been many proposed approaches viz. direct comparison between regions, contour matching and template matching. Most of the approaches are highly sensitive to affine transformations, rotation, scale change etc. However, for robust matching, it has been observed that point correspondence identification delivers good performance. This approach, of performing matching using features, has been accepted as the most efficient and has become a generic approach for the same in frame-based vision.

This process usually involves 3 steps:

A) Detection: Points of interest are identified in the image at distinctive locations. These points should possess the property of repeatability i.e. under different viewing conditions, the same points should be identified.

B) Description: The neighborhood of every point of interest is represented by a distinctive feature vector, which is later used for comparison. For robust matching, the scheme for feature description should be resistant to noise, affine transformation etc.

C) Matching: The descriptor vectors are matched on the basis of a nearest neighbor criterion using Euclidean distance or other statistical measures and various other criteria. The speed of matching contributes significantly to the computation time involved in the entire process and it depends on the dimensionality of the feature vector.

For any feature-based image matching & recognition scheme, it is attempted to simplify the detection process without losing accuracy and reducing the size of the descriptor without losing distinctiveness. Generally, scale & in-plane rotation invariant detectors & descriptors are identified.

To gain an understanding of how these feature-based schemes are employed effectively in the ubiquitous frame-based imagers, some popular algorithms in this domain are analyzed in this chapter.

SIFT

Background

As mentioned before, the feature matching approach has emerged the most popular scheme for image correlation. To ensure the robustness of this approach however, these features must be distinctive & unaffected by the manner of image formation. High dimensional key-point descriptors generally show higher distinction and ensure correct matches. However, in the event of mismatch, correct matches can be identified by identifying sets of matches that agree upon relative scale, location & orientation. Also, extracting a greater number of features increases the probability of finding a correct corresponding match [1].

Brief Overview

The Scale Invariant Feature Transform (SIFT) algorithm works towards achieving object recognition on the basis of feature extraction and comparison. Many candidate feature types have been proposed and explored before viz. line segments [2], groupings of edges [3, 4], regions [5], color space matching [6], color histograms [7] etc. It develops and uses a new class of image features that are quite robust to scaling, affine transformations, rotation etc. The SIFT algorithm for object recognition/matching can be summarized into the following 5 steps, which are discussed in detail later –

- Features / stable points of interest are detected through a staged filtering approach in scale space.
- Image keys are created, using local gradients, which become distinctive descriptors of local regions.
- Individual features are matched to a database using a fast nearest neighbor algorithm.
- Hash table of Hough transform identifies matching clusters of features – for reliable identification, at least 3 features must be correctly matched. Clusters of ≥ 3 matches are verified by approximating the affine pose.
- The pose is verified by checking for least-squares & probability solution of other object parameters.

Algorithm Description

The SIFT approach has 4 major stages of computation involving features

a) Scale-space Extrema Detection: This involves the search for points of interest over all scales and image locations. Extrema of scale-space have been identified as the most stable points in an image and are, hence, ideal as detectors. These peaks are detected using a simple difference-of-Gaussian function.

b) Keypoint Localization: At each interest point, location & relevant scale are determined.

c) Orientation Assignment: Based on local gradient orientations, an orientation is assigned to the feature point.

d) Descriptor: At the selected scale, image gradients of small-sized cells are measured locally and transformed into a high-dimensional vector representation. The multi-dimensional histograms of local regions become the descriptors.

This reduces the image data to scale invariant vector representations of local features.

A) Scale-space extrema detection

A cascade filtering approach is used to identify image locations invariant to scale change – which yields stable features across all scales. It has been shown that, under various assumptions, the Gaussian function is the only possible scale-space kernel [8] i.e. a representation of the image across all scales can be obtained only by convolution with the Gaussian function. Detailed experimental comparisons [9] have shown that extrema of the scale-normalized Laplacian of Gaussian (LoG) function produce the most stable image features, but are very computationally intensive. Hence, SIFT generates approximate co-ordinates of this point by using a Difference of Gaussian (DoG) function – which closely follows the LoG function: $\sigma^2\Delta^2 G$
As shown in [1] this approximation can be understood from the heat diffusion equation –

$$\partial G/\partial \sigma = \sigma\Delta^2 G$$

which can be approximated to

$$(G(x, y, k\sigma) - G(x, y, \sigma))/(k\sigma - \sigma)$$

and therefore

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\,\sigma\Delta^2 G$$

This shows that when the difference-of-Gaussian function has scales differing by a constant factor. It already incorporates the σ^2 scale normalization required for the scale-invariant

Laplacian. The factor (k −1), being a constant over all scales, does not influence extrema location [1].

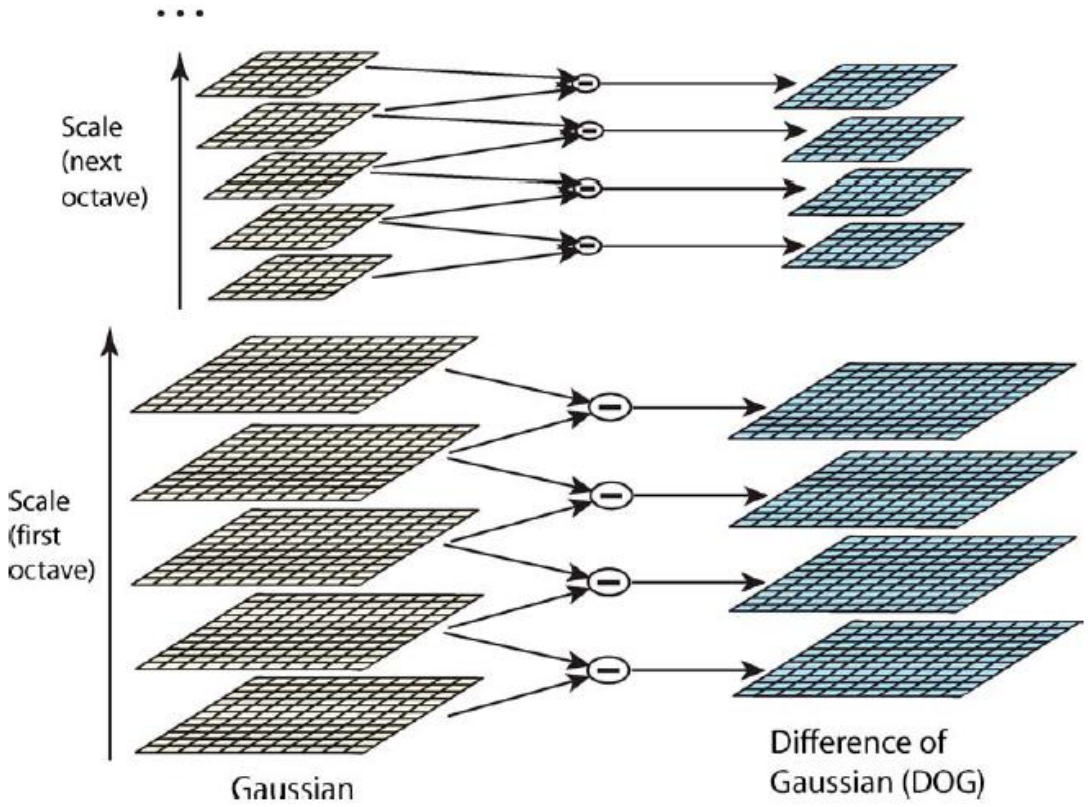


Figure 1. For each octave, repeated convolution of the original image with Gaussians generates scale space images on the left. Consecutive images in this stack are subtracted to produce the DoG images on the right [1].

Each octave is divided into s divisions: $k = 2^{1/s}$, where 'k' is the constant multiplicative factor between 2 consecutive scales. The k^{th} & $(k-1)^{th}$ images are subtracted. This process is effectively depicted in Figure 1. Each image point in such a scale space is selected and compared to 26 surrounding points (the members of surrounding 3x3 cube, centred at the event) [1]. The frequency of sampling (to detect such points in scale & space domains) cannot be decided beforehand – a tradeoff has to be reached between the level of accuracy desired & processing time:

a) Frequency of Sampling in Scale domain – As sampling frequency is increased; more extrema will be detected locally, rendering the algorithm unstable and failure in getting matches. Decreasing frequency would lead to loss of information and, hence, affect detection accuracy.

b) Frequency of Sampling in Spatial domain – Increasing sampling in this domain generates a similar problem.

B) Accurate key-point localization

After identifying a key-point candidate, a detailed fit has to be made for getting its scale, location etc. The position of the maximum is interpolated using the Taylor's expansion –

$$D(x) = D + (\partial D/\partial x)^T.x + (x^T .(\partial^2 D/\partial x^2). x)/2$$

The function, D(x), is thresholded to filter out points with low contrast. To increase processing speed, the differentials are approximated by the traditional ratio of differences of values. The value, x, thus obtained, yields the location of the extremum with reference to the identified extremum in scale space. This is added to the sample's value to get the true location.

C) Orientation Assignment

This procedure involves assigning orientation to each key-point with respect to a model co-ordinate system. The descriptor is also based on its orientation, thereby achieving rotation invariance.

Initially, a Gaussian-smoothed image closest in scale to the key-point is identified and gradient magnitudes & orientations are calculated for L(x, y) in a region around the key-point. These entries weighted by a Gaussian kernel centred at the key-point are then filled into a histogram of 36 bins using a nearest-neighbor approach.

Values of $\geq$80% of the maximum histogram value are recorded & dominant orientation is assigned to the keypoint. Multiple orientations can also be assigned.

D) Key-point Descriptor

With the earlier mathematical treatment, a repeatable 2-D co-ordinate system is available attached to each key-point, with rotation and scale invariance. A highly distinctive descriptor is now required to get perfect matches.

For high distinctiveness, a 128-dimensional vector is created for the SIFT approach. Image gradient magnitudes & orientations are calculated around the keypoint, weighted with a Gaussian centred at the key-point and rotated, as mentioned above, with respect to the keypoint. A 4x4 block, comprising of 4x4 cells is created around the key-point and 8 orientation bins are created in each. Each of the values in these 8 bins for each cell, constitutes a dimension of the descriptor vector. Feature vector is normalized to reduce contrast effects.

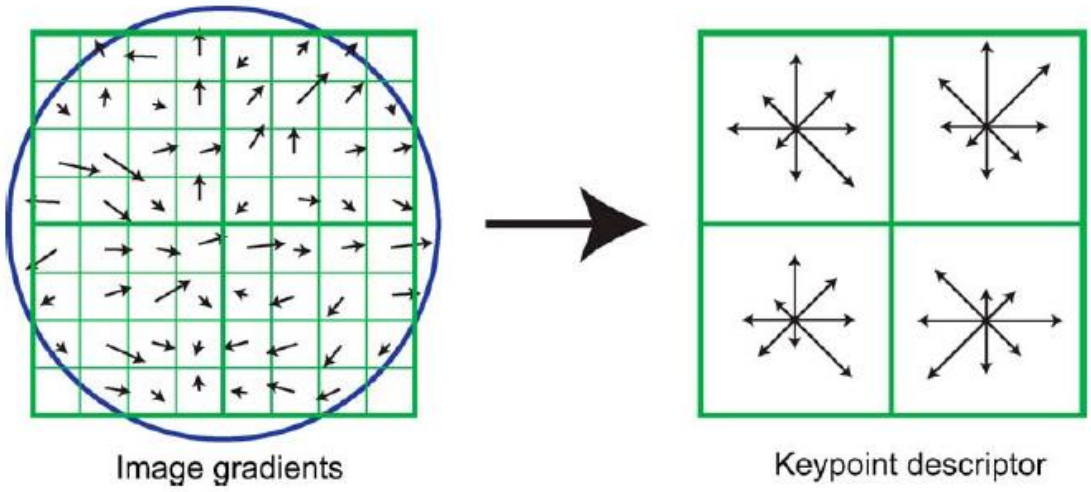


Figure 2. Keypoint descriptors are evaluated by computing the gradient magnitude and orientation at each image sample point around the keypoint location as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle [1].

E) Key-point Matching

The best match for a detected feature is found by a Euclidean distance nearest-neighbor search. Outliers are discarded using the second nearest neighbor criterion i.e. ratio of nearest neighbor distance to distance to nearest neighbor known to be on another object. A lesser ratio implies a good match.
The k-d tree best-bin-first search approach developed by Lowe [10] is used to optimize this process.

F) Clustering:

A minimum of 3 feature matches are required for affirming a positive match, Also, outlier matches must be removed.
For this purpose, each feature votes on orientation & pose of the object. The knowledge of each key-point's location and characteristics makes it possible to hypothesize their location on the object in that pose. Features that agree on the pose are collected in bins.

For bins with ≥ 3 matches, transformation parameters are estimated by computing the fundamental matrix solution. The transformation matrix is then used to measure correspondence between the features. If < 3 points remain on a model, the match is rejected.

Finally, based on the number of agreeing parameters, a probabilistic decision is made. An example of SIFT matching is depicted in Figure 3.

Figure 3. Identification of the 2 training objects on the left in a cluttered image with high occlusion, using SIFT [1]

SURF

Brief Overview

Speeded Up Robust Features (SURF) [11] approach is another scheme for identifying scale & rotation invariant detectors and descriptors, working towards achieving object recognition on the basis of feature extraction and comparison. Similar to the SIFT approach, SURF selects interest points of an image from the salient features of its linear scale-space, and then builds local features based on the image gradient distribution [1].It develops a faster and computationally cheaper method for performing image convolutions utilizing integral images & box filters. The SURF algorithm for object recognition/matching can be summarized into the following 5 steps, which are discussed in detail later

- Computation of integral images.
- Features / stable points of interest are detected by computing a discrete Hessian operator in scale space.
- Distinctive descriptors of local regions are created using gradient magnitudes and orientations in the neighborhood.
- Individual features are matched to a database using a fast nearest neighbor criterion by a priori checking the match of the sign of the Laplacian.
- As in SIFT, a hash table of Hough transform identifies matching clusters of features – for reliable identification, at least 3 features must be correctly matched. Clusters of ≥ 3 matches are verified by approximating the affine pose.
- The pose is verified by checking for least-squares & probability solution of other object parameters.

Algorithm Description

The SURF approach has 5 major stages of computation involving features –

a) Scale Space Representation: To speed up the process of evaluating feature detectors while maintaining accuracy, a different approach for representing scale space using integral images & box filters is used.

b) Keypoint Detection: This involves the search for points of interest over all scales and image locations. Maxima of the scale normalized second order derivatives i.e. the determinant of the Hessian matrix, at points of the image are detected as features [9]. These candidates are then validated if the response is above a given threshold. Both scale and location of these candidates are then refined using an iterated procedure to fit a quadratic function.

c) Orientation Assignment: To achieve rotation invariance, a dominant orientation is defined by considering the local gradient orientation distribution, estimated with Haar wavelets.

d) Descriptor: At the selected scale, making use of a spatial localization grid, a 64-dimensional descriptor is built, corresponding to a local histogram of the Haar wavelet responses. This reduces the image data to scale invariant vector representations of local features.

e) Matching: The SURF descriptors of both images are compared by a nearest neighbor criterion inspired from the SIFT algorithm, speeded-up by a priori imposing that the sign of the Laplacian is the same for corresponding descriptors. Based on geometric consistency checking (ORSA) algorithm, spurious matches are discarded.

Scale space approximation by box filters:

Defining a method which is invariant to scale change is typically achieved by simulating zoom-outs of the considered image. This corresponding scale-space representation can be obtained by convolutions of the image with a Gaussian at different scales (the Gaussian being the only scale space), as is done in SIFT. To speed-up this time-expensive procedure, SURF approximates the Gaussian kernel and its spatial derivatives by rectangular "*box*" functions, which have linear convolution complexity using the *integral image* representation developed in [12]. These concepts are discussed below –

a) Integral image and box filters: As demonstrated in [13], let 'u' be the considered digital image defined over the pixel grid J. In the following, we only consider gray valued images, to ensure robustness to color modifications (such as a white balance correction). The *integral image* of 'u' is simply defined as –

$$U(x, y) = \sum u(i, j) \mid (i \leq x, j \leq y)$$

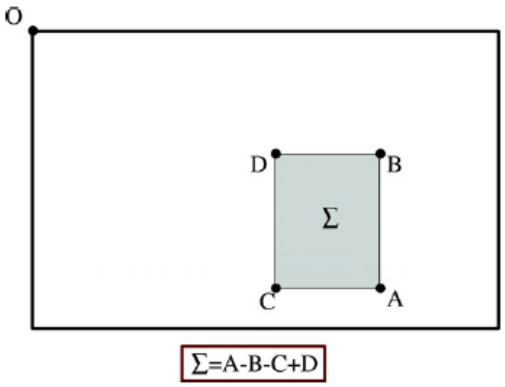


Figure 4. The sum of intensities inside a rectangular region can be calculated using only 3 addition operations and 4 memory accesses using integral images [11]

We now take interest in the convolution of 'u' with a 2-D rectangular function B_Γ with rectangular support: $\Gamma = [-W, W] \times [-H, H]$

$$B_\Gamma(x, y) = 1 \text{ if } (x, y) \in [-W, W] \times [-H, H]$$

The pre-computation of such an integral image U permits to convolve 'u' with a box B_Γ in **three operations** and four memory accesses using the formula:

$$(B_\Gamma * u) = U(x-W, y-H) + U(x+W,y+H) – U(x-W,y+H) – U(x+W, y-H)$$

b) Symmetrization: Convolution across image borders may create visual artifacts and inconsistent interest points if not handled properly. Thus, *symmetrical boundary conditions* are considered [13]. Equivalently, the original image is symmetrically extended at each side using the larger box filter size before computing the integral image

c) Comparison with Gaussian scale-space: The classical *scale-space* representation of a digital image 'u' is obtained by convolution with the Gaussian kernel –

$$u_\sigma = g_\sigma * u$$

where g_σ is the centered 2-D Gaussian kernel with parameter σ (the *scale parameter*).

Making use of the aforementioned box filters technique, such a representation can be approximated with the discrete convolution: $u_\sigma' = B_L * u$, where B_L is a box-filter with bandwidth L. In the discrete setting, the values taken by *L* are integer and odd. The corresponding value of σ is then given by –

$$\sigma^2 = (L^2 - 1)/12$$

The box filter should be normalized by a factor L^2 to obtain a unitary kernel. This is not the case for computation time consideration. This normalization is performed only once during the feature detection step. The approximated scale-scale is roughly similar to the Gaussian kernel but exhibits some strong vertical and horizontal artifacts due to the anisotropy of the box kernel.

Further analysis of the accuracy of approximation can be studied in detail in [13].

A) Keypoint Detection

Three steps are involved in the interest point detection process –

1. Feature *detection*, which is based on the scale-normalized determinant of the Hessian,
2. Feature *selection*, using non-maxima suppression and a threshold
3. Scale-space location *refinement*, using a finite difference scheme.

1. Feature Detection: As advocated by Lindeberg [14] scale-invariant features may be detected by using scale-normalized second order derivatives on the scale space representation of the considered image. These features correspond to blobs, edges or corners. Contrarily to SIFT, in which D.G. Lowe approximates the normalized Laplacian operator with a difference of Gaussians (*DoG*), SURF approximates the scale-normalized determinant of the Hessian (**DoH**) operator with a simplified formula as in [13].
According to SURF, the coefficient r(σ) permits to compensate the rough approximation of the estimated DoH operator for small scales. At each point of the scale-space, the computation of this operator thus requires 32 operations (26 additions and 6 multiplications).

2. Feature Selection: Points of interest are points of the scale-space that correspond to local maxima of the aforementioned DoH operator applied to the scale-space representation of the image. These points are selected by considering a 3x3x3 neighborhood, thus performing an exhaustive comparison of every point of the scale-space with its 26 nearest-neighbors.
In order to obtain a compact representation of the image the algorithm selects the most salient features from this set of local maxima. This is achieved by using a threshold on the response of the DoH operator for every interest point candidate (x,y) at scale σ:

$$|\mathrm{DoH}^{(\sigma)}(u)(x, y)| > t_H$$

where t_H is the threshold

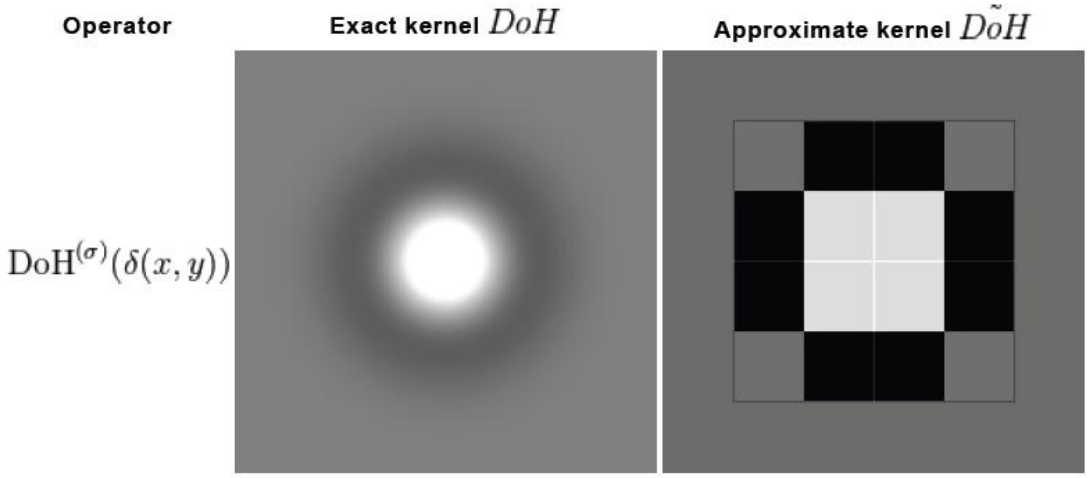


Figure 5. Haar wavelet approximation of DoH kernel used in SURF [13]

3. Scale Space Refinement: For each local maximum of the DoH operator, the localization of the corresponding interest point *M* with coordinates (x,y,σ) in the scale-space may be refined using a simple second order interpolation. More specifically, as shown in [13] we rely on a quadric fitting in scale space using the following formula:

$$M' = M+\delta \text{ where } \delta = (\delta_x\ \delta_y\ \delta_z)^T = -H^{-1}(\mathrm{DoH}^{(\sigma)}(u)(x, y)) \times \Delta(\mathrm{DoH}^{(\sigma)}(u)(x, y))$$

The computation of the 3-by-3 Hessian matrix *H* and of the gradient of the DoH operator at scale σ and point (x,y) is based on finite differences schemes on a 3x3x3 centered neighborhood. Eventually, for each interest point, in addition to its coordinates *x*,*y*, and *σ*, the sign of the Laplacian is also stored for the matching step in order to speed up the comparison of SURF.

B) Local Key-point Descriptor

The key-point descriptors, as in SIFT, are based on local gradient magnitudes & orientations. This involves 2 steps –

1. Assigning a local dominant orientation
2. Calculating gradients in cells to construct descriptors

1. Local Dominant Orientation: Gradients in the neighborhood are found by convoluting with Haar wavelet box filters & weighting with a Gaussian centered at the point of interest. A 60º window is then rotated around the disk containing all neighborhood gradients, at regular intervals of π/8 radians and a score function is evaluated (Figure 6). Orientation is that angle with the maximum score.

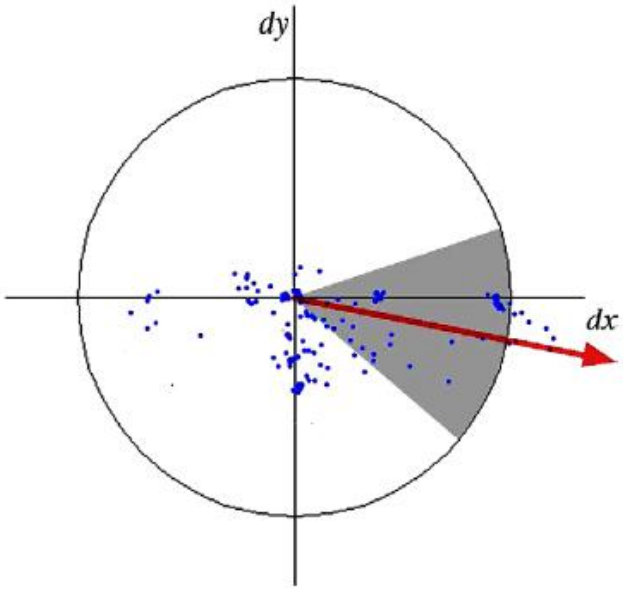


Figure 6. Orientation Assignment: A sliding orientation window, in the form of a circle sector of π/3 radians, detects the dominant orientation by evaluating Gaussian weighted Haar wavelet responses at each point [11]

2. Descriptor Construction: A square grid of side $20\sigma_i$, centered at the feature point is drawn and divided into 5x5 σ_i cells.

In each sub-region, **25** regularly spaced gradient samples are used to build a 4-dimensional vector. To do so, these gradients are first computed with scale using convolution with box filters on the regular grid, and then rotated to correspond to the aligned region.

Each cell has 4 features described i.e. local gradients' magnitudes & orientations. Since there are 16 such 5x5 cells, a 64 dimensional vector descriptor is created.

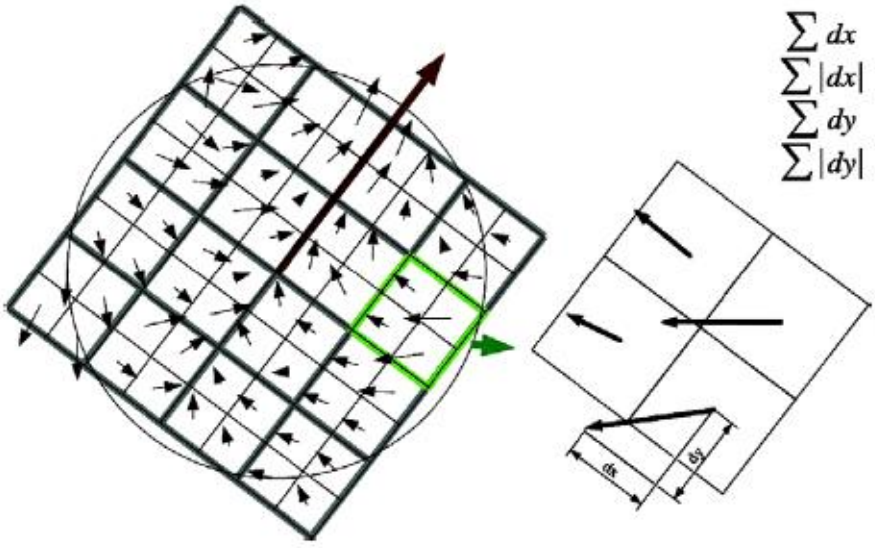


Figure 7. An oriented grid is laid over the interest point, as on the left. Orientations are evaluated using Haar wavelet responses and binned into orientation histograms as shown in the right [11]

C) Matching

The matching scheme in SURF follows almost the same routine as SIFT. Besides, to speed up the matching procedure, we compare the sign of the Laplacian between two interest points as advocated by the authors of SURF. The Euclidean distance between the query descriptor of the first image and every descriptor of the second image is computed. Only the nearest and second nearest neighbors are considered. If the ratio of these two distances is below 0.8, the match between the query and the nearest candidate is validated. Such a thresholding permits to discard numerous mismatches.

Nevertheless, it is not always sufficient. Thus, additional RANSAC (RANdom SAmple Consensus [15])-based step to check the epipolar geometrical consistency of the validated matches between the two views, are used.

FAST

Background

The first step in any scheme for object recognition/matching using features is detection of repeatable keypoints in the image, invariant to scale & rotation. Existing schemes for feature detection rely on evaluating a corner response function across the image and retaining the locally maximal response. Many approaches for a corner response function have been proposed viz. Difference of Gaussians (in SIFT), Harris corner detector [16], SUSAN [17]. However, these are too computationally intensive to use in real-time applications, as they leave very little time, if any, for processing post detection. Thus, such detection schemes are not feasible in real-time machine vision tasks like Simultaneous Location and Mapping (SLAM) in robots, despite the high-speed processing capabilities of today's hardware.

Brief Overview

The Features from Accelerated Segment Test (FAST) scheme [18] [19] for detection, builds upon previous work of the same authors to develop a high-speed corner detection algorithm. FAST belongs to a class of detectors that work by examining a small patch of an image and decide whether it 'looks' like a corner – the decision making criteria varies for each scheme. Thus, no second derivatives are required to be calculated, removing the need for a noise-reduction step. This amounts to savings in computation time, but poor performance on images with large-scale features.

Algorithm Description

A) High-speed Corner Detection

The original detector constructs a Bresenham circle of 16 pixels and radius 3 (Figure 8) around a pixel 'p', with intensity I_p. A Bresenham circle is simply the set of pixels which are included in the overlap of a real circle structure with the discrete pixels of the image. The pixel is classified as a corner "*if there exists a set of n contiguous pixels in the circle which are brighter than the intensity of the candidate pixel I_p plus a threshold t, or all darker than I_p-t"* [19]. Initially, the authors have used the value of n =12.

To make the detector reach decisions faster, the intensity of the pixels at positions 1, 5, 9, 13 (as in Figure 8) are compared with I_p. This strategy relies on the observation that if at least 3 of these 4 pixels do *not* have values above $I_p + t$ or below $I_p - t$, then p cannot be a corner. If this test is passed, then all the 16 pixels are tested to check if there are 12 contiguous pixels that satisfy this criterion. This is done for all the pixels in the image to locate corners.

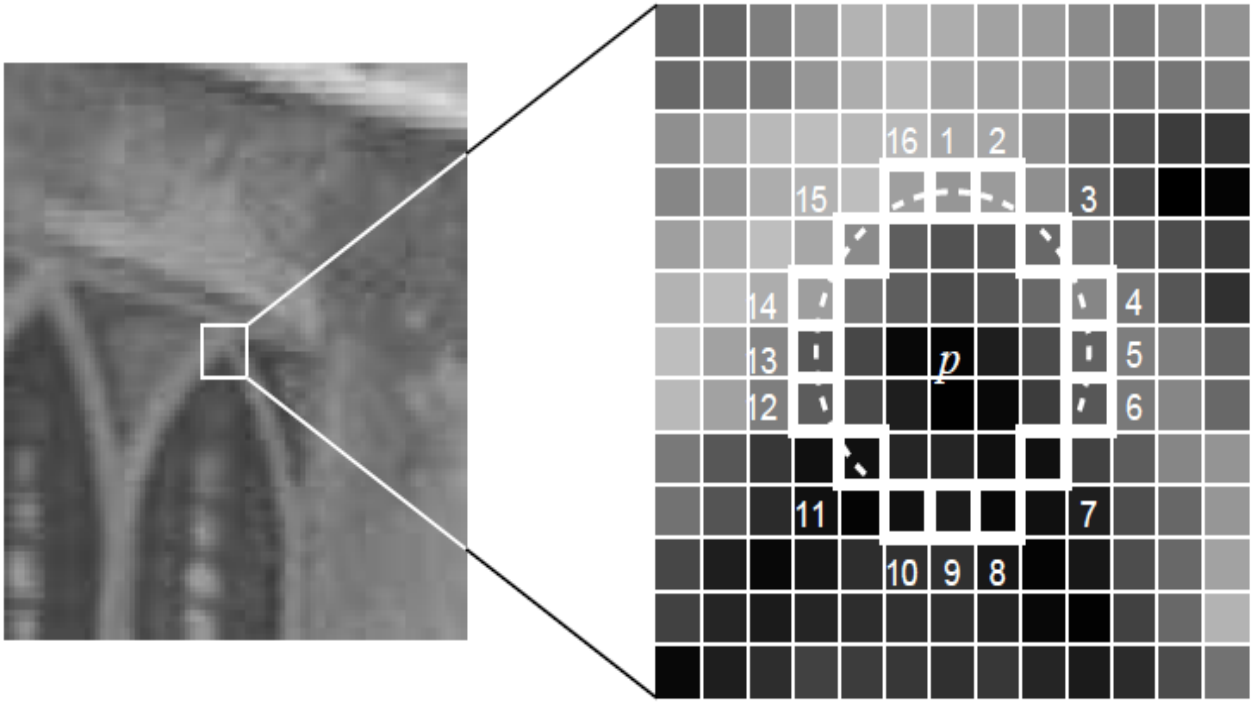


Figure 8. 12 point segment test corner detection. The highlighted squares are used in reachign a decision about the central pixel 'p' as a corner [19]

.

Despite exhibiting quite high performance, the detector has several weaknesses [19]–

- The algorithm does not work very well for n<12, generating a very large number of keypoints.
- The positions and choice of the fast test pixels makes certain implicit assumptions regarding the features.
- Many features are detected close to each other.

The corner detection in the image is further improved & speeded up using a machine learning approach to decide the set of pixels to be checked for deciding 'cornerness'.

B) Non-Maximal Suppression

To address the problem of multiple interest points being detected adjacent to each other, a non-maximal suppression is applied after this step. Hence, a heuristic score function V is calculated for each corner, which can be used to compare 2 adjacent corners and corners with lower values of V are removed. There are several intuitive definitions of V, however the function mentioned below is used in the FAST scheme –

$$V = \max \left(\sum_{(x \in S_{bright})} | I_{p->x} - I_p | - t, \sum_{(x \in S_{dark})} | I_p - I_{p->x} | - t \right)$$

With -
$S_{bright} = \{x \mid I_{p->x} \geq I_p + t \}$
$S_{dark} = \{x \mid I_{p->x} \leq I_p - t\}$

This function can, basically, be defined as the sum of the absolute difference between the pixels in the contiguous arc and the central pixel.

BRIEF

Background

Feature description, until this point, has been using a high-dimensional histogram of gradients to describe local image regions. However, despite generating highly distinctive descriptors, this approach is very computationally intensive – cutting into processing power & time both in the generation and matching steps. Various improvisations of this descriptor have been proposed for dimensionality reduction including quantizing the floating point descriptor values of the descriptor vector using very few bits per value [20, 21, 22] or, even, using hash functions to reduce them to binary strings [23]. However, many popular dimensionality reduction approaches including Principle Component Analysis (PCA) [24] and Linear Discriminant Analysis (LDA) [25] require first computing the full dimensionality of the descriptor and then process it further – increasing processing time.

Brief Overview

Binary Robust Independent Elementary Features (BRIEF) [26] presents a new approach for building descriptors which cuts short the whole computation of descriptors and subsequent conversion to binary strings by directly building the descriptor as a binary string of values from image patches. The individual bits are obtained by direct intensity comparisons of points in the vicinity. The BRIEF descriptor, thus, speeds up construction of the descriptor and also capitalizes on the speed & ease of comparison enabled in the case of binary descriptors.

Algorithm Description

A) Method

The BRIEF approach is based upon earlier developments [27, 28] wherein image patches are classified based on a small number of pairwise intensity comparisons. It is proposed to create a bit vector out of test vectors as a descriptor of an image region. This is obtained using the following scheme τ for a patch '**p**' of size SxS –

$$\tau(p; x, y) := 1 \quad \text{if } p(x)<p(y)$$
$$:=0 \quad \text{otherwise}$$

Wherein p(x) is the smoothed intensity of the pixel at a sample point $x = (u\ v)^T$. A set of n_d pairs is defined, so as to generate an n_d dimensional bit string. The BRIEF descriptor is then taken to be the n-dimensional bitstring –

$$f_{nd}(p) = \sum_{(1 \le i \le nd)} 2^{i-1}\tau(p; x_i, y_i)$$

Many BRIEF-k descriptors are tested, where $k = n_d/8$ is the number of bytes required to store the descriptor.

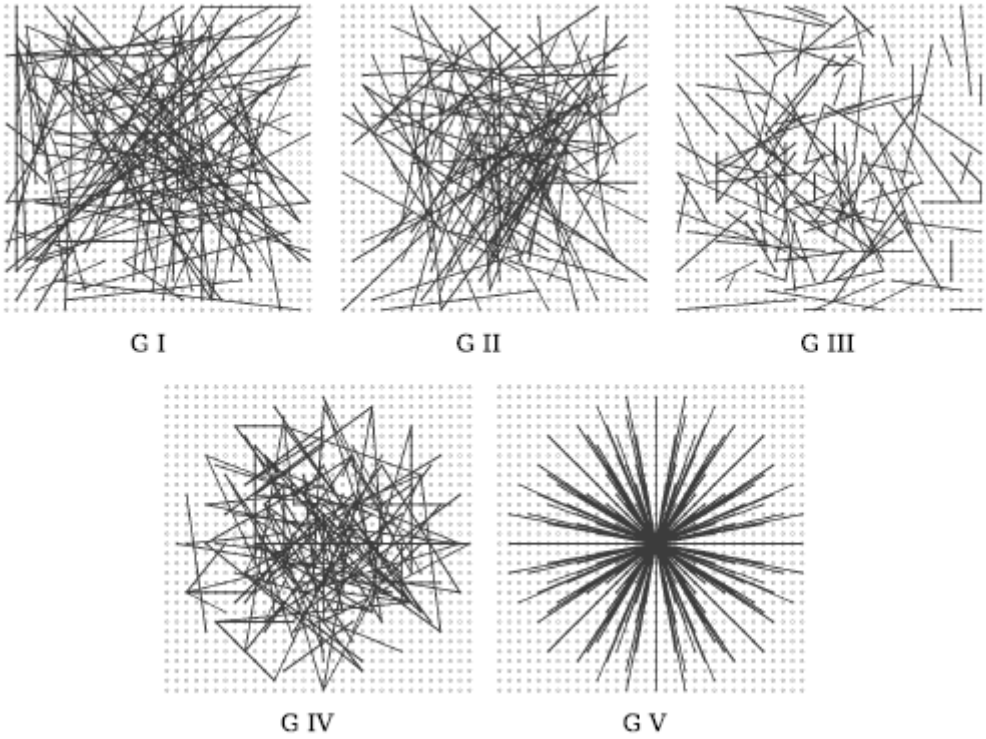


Figure 9. Different approaches to choosing test locations. All, but the last, are selected by random sampling [26]

.

B) Smoothing Kernels

Taking single pixels' information into account causes the descriptor to be very noise-sensitive. Hence, certain pre-smoothing kernels are applied to decrease this sensitivity, enforcing stability and repeatability. Gaussian blurring kernels of various window sizes are tested and a 9x9 kernel is chosen optimal [26].

BRISK

Brief Overview

The Binary Robust Invariant Scalable Keypoints (BRISK) [29] scheme presents a fresh new approach in the issue of effective development of keypoints for object recognition & matching in images. It achieves robustness & speed, at lesser computational cost, by building upon and combining the advantages of the FAST [18] scheme for detection of keypoints & the BRIEF [26] scheme for keypoint description.
BRISK can be quantified into the following steps, detailed later

- Scale-space keypoint detection
- Generation of sampling pattern
- Orientation assignment
- Keypoint description
- Descriptor Matching

Algorithm Description

A) Scale-space keypoint detection

Features of interest/keypoints, invariant to scale, are identified using a saliency criterion – employing a multi-scale implementation of Mair et al's FAST [18] keypoint detection algorithm. The maxima search is carried out in the image plane and the scale space, as well. The FAST score is used as a saliency measure.

The scale-space is divided into 'n' octaves c_i & 'n' intra-octaves d_i. As detailed in [29], the octaves are obtained by successive half-sampling of the original. The intra-octaves are obtained by successive half-sampling of the first intra-octave, which is the original image down-sampled 1.5 times.

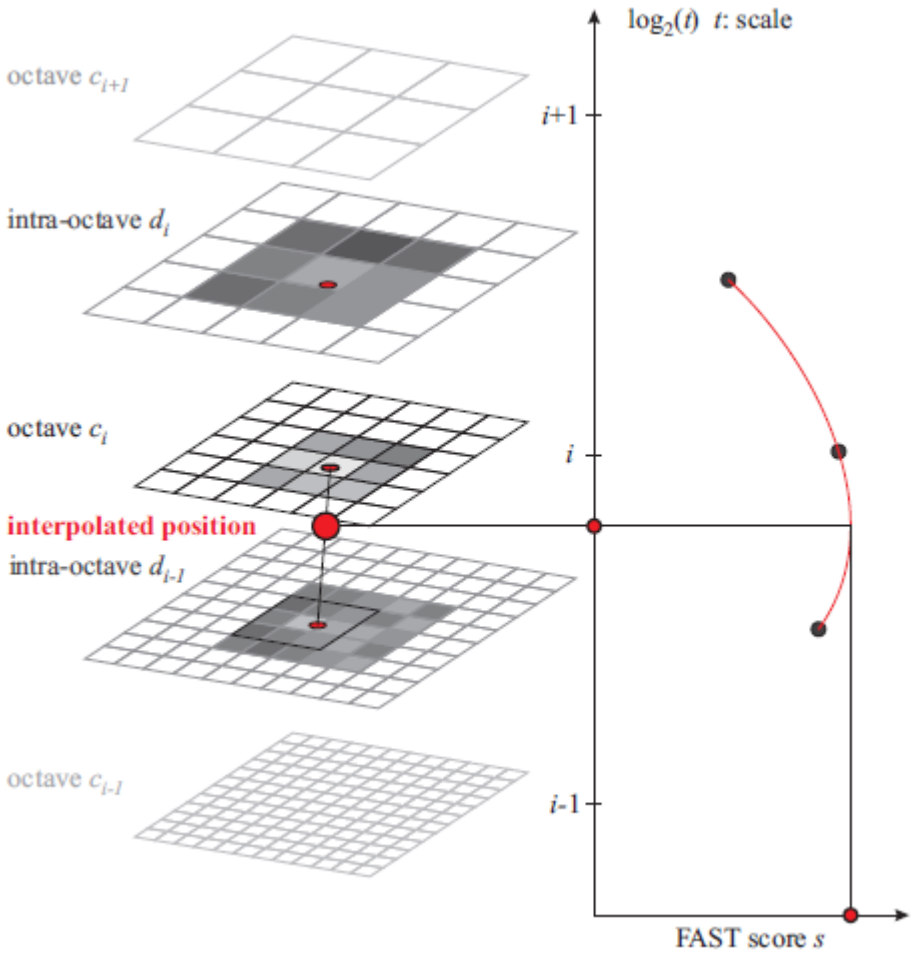


Figure 10. Scale-space interest point detection in BRISK [29]

The BRISK scheme uses the FAST 9-16 mask i.e. the detector requires that at least 9 contiguous of the surrounding 16 pixels should be sufficiently brighter / darker as compared to the central pixel. Initially, this mask is applied to each octave and intra-octave to locate regions of interest. Non-maximal suppression, in this case, requires comparison in the image as well as the scale space. The FAST score of a point is compared to its 8 neighboring scores for maximality and similarly with scores in the layers above and below. To find the optimal score and corresponding scale, a 2-D quadratic function is first fit through the 3 score patches at each scale and the interpolated maximum obtained at each scale. A 1- D parabola is fitted through these 3 values and the final score & scale estimate of the keypoint is obtained at its maximum (Figure 10).

B) Generation of sampling pattern

As opposed to the BRIEF descriptor, which it is inspired by, the BRISK descriptor employs a structured sampling pattern to sample the keypoint's neighborhood.

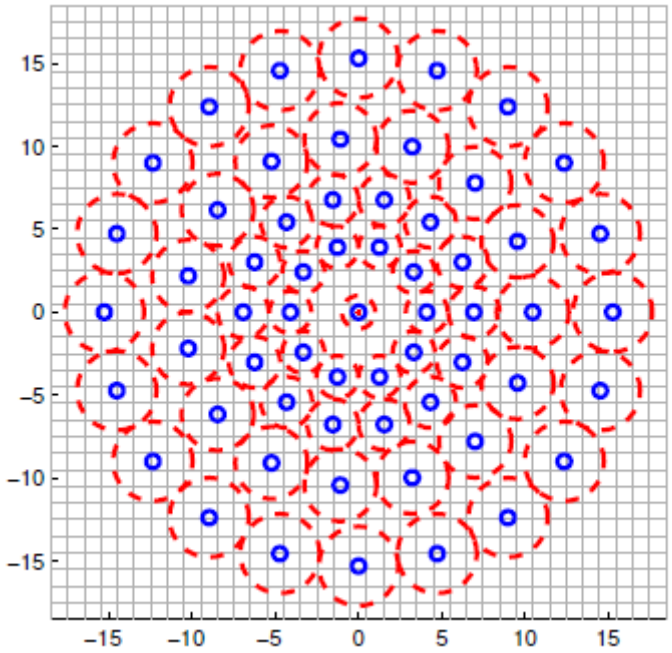


Figure 11. The BRISK sampling pattern with N=60. The blue circles denote sampling locations and the red circles denote the σ of Gaussian blurring kernels at each point [29]

The pattern used is illustrated in Figure(11) – it marks N uniformly spaced locations on circles concentric with the keypoint. This pattern, reminiscent of the DAISY [30] descriptor, forms the basis of pair formations for constructing the descriptor. To reduce sensitivity to noise, Gaussian smoothing is applied at each of the sampling locations, Moreover, to avoid aliasing, the Gaussian kernels used for each point are chosen with standard deviation proportional to the distance between the points on the respective circle. These generate N.(N-1)/2 sampling point-pairs.

Subsets of the same – S, for short distance pairings & L, for long distance pairings – are defined [29] and used independently. For a set 'A' of all sampling point pairs:

$$A = \{(p_i, p_j) \in R^2 x R^2 \mid i<N \wedge j< i \wedge I, j \in \text{Natural nos.}\}$$

The 2 aforementioned sets are defined as –

$$S = \{(p_i, p_j) \in A \mid \| p_j - p_i \| < \delta_{max}\}$$
$$L = \{(p_i, p_j) \in A \mid \| p_j - p_i \| > \delta_{min}\}$$

Namely, the long-distance pairs are used for assigning orientation and the short-distance pairs are used to construct the actual descriptor.

C) Orientation assignment

As mentioned before, long distance pairings from the pattern are used in estimating the characteristic pattern direction as it is postulated that local short-range gradients annihilate each other. The local gradient is calculated using a pair of points (p_i, p_j) as –

$$g(p_i, p_j) = (p_j - p_i).\ (\ I(p_j, \sigma_j) - I(p_i, \sigma_i)\ ) / \| p_j - p_i \|^2$$

using the smoothed intensity values at the points: $I(p_i, \sigma_i)$ & $I(p_j, \sigma_j)$. The global orientation, using only long-distance pairs, is estimated to be:

$$g = (g_x\ g_y)^T = (\ \sum_{(pi,\ pj) \in L)}\ g(p_i, p_j)\ ) / \text{length (L)}$$

The short-distance pairings are experimentally verified to not be of much consequence to calculate global dominant orientation.

D) Keypoint Description

To generate a scale & rotation invariant descriptor, BRISK uses the sampling pattern rotated by the orientation calculated above: $\alpha = \arctan2\ (\ g_y\ g_x)$ [29] around the keypoint 'k'. The descriptor is then evaluated by performing bitwise comparisons of the Gaussian-smoothed intensity values of the short-distance comparisons $(p_i^{\alpha}, p_j^{\alpha})$ around each keypoint, with values being assigned to each bit as follows –

$$b = 1, \quad \text{if } I(p_j^{\alpha}, \sigma_j) > I(p_i^{\alpha}, \sigma_i)$$
$$= 0, \quad \text{otherwise}$$

The proportional amount of Gaussian smoothing ensures that the information content at a point is not accidentally corrupted by blurring at a nearby point. Also, since the number of samples is quite low, compared to the number of pairs, generation & use of a lookup table is of reduced complexity.

E) Descriptor Matching

In comparison with schemes like SIFT & SURF, matching 2 BRISK descriptors is an extremely simple computation. The descriptors, being bit strings, the comparison is simply an evaluation of their Hamming distance – the number of dissimilar bits being a measure of their dissimilarity. In digital architectures, this simply translates to a bitwise XOR operation and a bit count – both of which can be implemented very efficiently on today's computers.

FREAK

Brief Overview

Fast Retina Keypoint (FREAK) [31] addresses the growing need to enable faster computation of descriptors and reducing their dimensionality without compromising distinctiveness while ensuring robustness to scale, rotation & noise. It presents a novel keypoint descriptor approach which attempts to emulate certain processes in the human visual system – particularly, the human retina. It builds upon the approach developed by Calonder et al in [26] and extended by Leutenegger et al in [29], of building descriptors comprising cascades of binary strings obtained using simple pair-wise intensity comparisons in a particular sampling region around the detected keypoint. FREAKs, however, exhibit faster computation speeds, more robustness compared to existing standard descriptors and with lower memory loads.
The FREAK algorithm for description has certain salient features which are summarized into the following steps

- Generation of a retinal sampling pattern around the keypoint.
- Gaussian blurring at each of the sample points.
- Coarse-to-fine ordering of the pairs of sample points compared.
- Orientation assignment.
- (The keypoint matching step) Emulating retinal saccadic search by parsing the descriptors in several steps.

Human Retina Analogy

In developing the FREAK descriptor, Alahi et al propose to emulate the topology of the retina, which is believed to be responsible for extracting details accurately from scenes. It is proposed that, since, the final output is encoded as HIGH/LOW action potential pulses transmitted to the brain, the binary descriptor obtained by simple intensity comparisons can be good enough to describe image features. Figure 12 illustrates the parallels between these 2 systems very effectively.

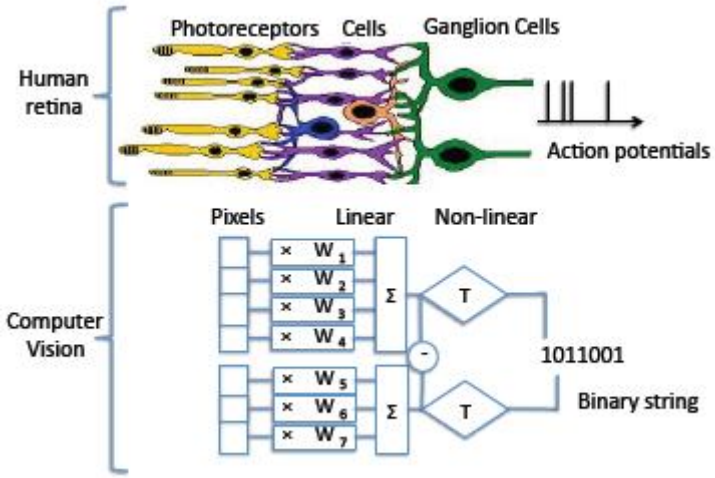


Figure 12. Translation from human retina to computer vision. Action potentials are emulated by simple binary tests over pixel regions [31]

Analogy: Human visual system to Pixel-based imaging systems

The ganglion cells in the retinal area can be segmented into 4 areas – foveal, fovea, parafoveal & perifoveal as shown in Figure 13 [32] – each of which performs a unique function in processing the scene being observed viz. the fovea captures a high resolution image and hence evaluates very minute details whereas the perifoveal observes a low resolution, low acuity image and is, possibly, used as an initial filtering stage.

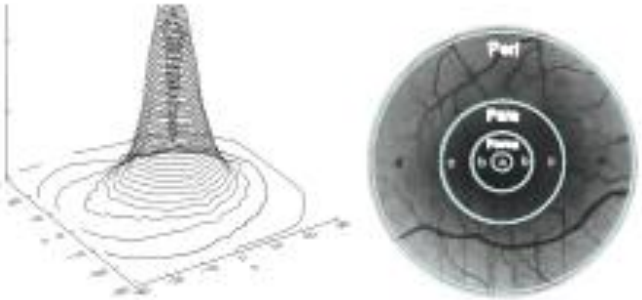


Figure 13. Spatial density distribution of ganglion cells (left) and Anatomical regions of the retina (right) [31]

Since FREAK uses a novel descriptor scheme for keypoints, building further on the BRIEF scheme, it is used on a congruent feature detector scheme used in BRISK i.e. (effectively) a multi-scale AGAST [33], implemented in BRISK. Once detected, the description scheme implemented for each keypoint is discussed further below.

Algorithm Description

A) Retinal Sampling Pattern Generation

FREAK proposes a structured approach to developing a sampling pattern around the keypoint, for the purposes of generating sets of points for pixel intensity comparison – as opposed to BRIEF & ORB [34], which use random pairs of points around the keypoint to build the descriptors. BRISK employs a circular pattern (reminiscent of [30]), with the points equally spaced on concentric circles and blurred with Gaussian kernels of varying sizes but non-overlapping fields, to make them less sensitive to noise.
The FREAK scheme suggests a sampling pattern inspired by the retina

- The sampling grid has an exponentially decreasing density of points away from the keypoint centre, where it is maximum.
- To decrease noise sensitivity, a receptive field of each sample point is defined by blurring with Gaussian kernels of varying sizes. The difference from the scheme used in BRISK is that overlap in these fields is allowed – this increases distinctiveness by capturing certain redundant information.

Figure 14. The FREAK sampling pattern with overlapping receptive fields [31]

B) Descriptor Construction

The binary descriptor, F, is constructed as a bit string – each bit value obtained by comparing the Gaussian-smoothed intensities of a pair of receptive fields as described in [31]

$$F = \sum_{(0 \le a \le N)} 2^a . T(P_a)$$

Wherein P_a is a pair of receptive fields and N is the desired size of the descriptor, and

$$T(P_a) = 1, \quad \text{if } I(P_o^{r1}) - I(P_o^{r2}) > 0$$
$$= 0, \quad \text{otherwise}$$

Where $I(P_o^{r1})$ is the smoothed intensity of the first receptive field of the pair P_a.

The number of such pairs can increase exponentially with the number of possible combinations of the receptive fields, giving rise to a high-dimensional descriptor. However, unlike SIFT & SURF, a larger number of dimensions does not necessarily imply enhanced distinctiveness. Hence, FREAK explores some strategies to select correlated & discriminant data

1> Coarse-to-fine Ordering: An algorithm to learn the best training pairs is implemented, similar to the one implemented in ORB [34].

- A matrix of the numerous extracted keypoints is constructed – with each row corresponding to a single descriptor and each column corresponding to a particular pair's comparison.
- Statistics are evaluated for each column – high variance is desired for distinct features.
- The columns are, then, ordered as per decreasing variance.

It is observed through experiments that the first 512 of the ordered pairs are most relevant. Upon grouping these into 4 clusters of 128 bits each, a coarse-to-fine ordering is automatically generated from the start to the end. Interestingly, it is also seen that the first cluster involves peripheral receptive fields and the later ones are highly-centered – reminiscent of the model of the layered model of the human retina.

C) Orientation Assignment

To ensure correct matching and correspondence between sampled pairs, the keypoint rotation must be evaluated. The angle of rotation is calculated as the sum of estimated local gradients of selected pairs. FREAK selects those pairs with mainly symmetric receptive fields with respect to the centre.
As defined in [31], the orientation is evaluated as

$$O = \left(\sum (P_o \in G) \left(I(P_o^{r1}) - I(P_o^{r2}) \right) . \left(P_o^{r1} - P_o^{r2} \right) / \| P_o^{r1} - P_o^{r2} \| \right) / M$$

Where M is the number of pairs in G and P_o^{ri} is the 2-D vector of the spatial co-ordinates of the center of the receptive field.

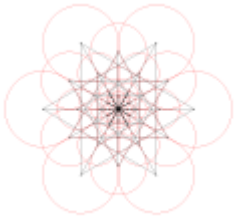


Figure 15. Pairs from the sampling pattern selected to compute orientation [31]

D) Saccadic search & Matching

This step capitalizes on the coarse-to-fine structuring of the FREAK descriptor. It is hypothesized that the retina executes a cascade filtering approach – the low-frequency data is used to estimate locations of objects while greater details are extracted from the high frequency data: both of which are detected at different locations in the retina. The FREAK search algorithm starts by parsing the descriptor in several steps and comparing sections of various sizes in each step.

Initially, the first 16 bytes of the FREAK descriptor (by analogy, representing the coarse information) are compared. This coarse comparison itself filters out an overwhelming majority of possible matching candidates. The remaining candidates are now subjected to further similar scrutiny. This cascade of comparisons speeds up the matching process and is robust to changes of illumination, viewpoint or other such affine transforms.

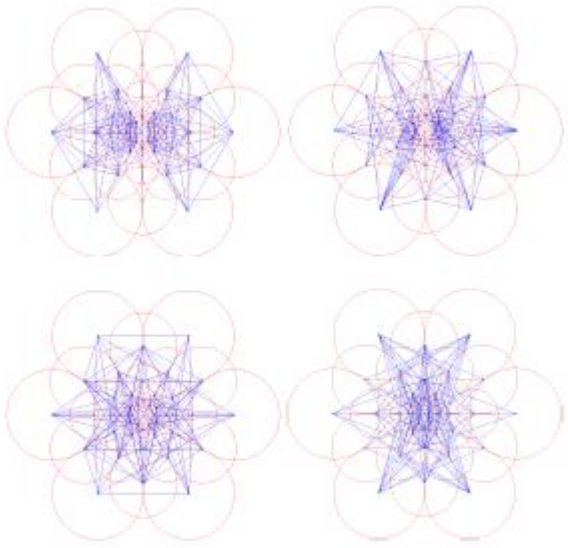


Figure 16. Coarse-to-fine analysis. First cluster involves perifoveal receptive fields and last ones, the foveal [31]

INFERENCES

The discussed algorithms provide an insight into how the classical computer vision problem of image matching has been addressed with feature-based schemes on frame-based imagers. These approaches show good performance and are perfectly suited for still images. However, if employed in videos, frame-based imagers generate enormous amounts of redundant information. Essentially, frame based video is based on a series of snapshots taken at a rapid but constant rate. Hence, pixels are sampled repetitively even if their values are unchanged. Furthermore, employing a feature-based scheme described above on the same would result in numerous redundant operations – incurring high computational costs and compromising on real-time performance. Thus, the presented algorithms present an ideal development basis for image processing applications.

However, corroborating earlier discussions, in computer & machine vision applications – where there exist restrictions on available processing power, and real time performance is of the essence in real-world scenarios – frame-based processing proves grossly inadequate. For these purposes, it is essential to cut down on the redundant operations. This can be remedied by incorporating a measure of the scene dynamic and performing computations only when changes occur in the scene, at the positions where these changes occur. Further improvements could be generated if there was a work-around for the redundancy of continuously recreating the observed scene in the image representation.

CHAPTER – III: EVENT-BASED VISION

Existing vision systems invariably follow frame-based schemes. The concept of encoding visual information in "frames" for digital representation & reproduction is ubiquitous today and is generally taken for granted. The reason for the same is evident from the chronological order of development of video technology - since the concept of frames dominated in the earlier days of still imaging, it was considered natural to extend this concept to the field of video. The output format of frames is easy to understand and has become the norm for research in machine vision. However, as discussed in the previous chapter, frame-based architectures for video and, particularly, for machine vision face a serious impediment due to the large amount of redundant information they generate. Also, to address low latency vision problems in a satisfactory manner, as are faced by embedded vision systems in uncontrolled, natural scenarios, the dynamic range offered by these sensors proves grossly inadequate [35].

As an effective alternative for frame-based imagers and to achieve the reduced processing goals mentioned in the previous chapter, a new kind of vision sensor is employed. The vision sensor used in the presented investigation efforts is such a departure from the frame concept – it is a 128x128 pixel CMOS event-driven device which responds to changes in temporal contrast. The pixels respond asynchronously to relative changes in intensity and generate an output stream of data, termed addressed events (AE) representation. This approach abandons the frame concept and models properties of biological vision to achieve higher dynamic range of processing. As detailed in [35], this device builds upon the functionality of prior frame-based temporal difference detection imagers e.g. [37], by responding to temporal contrast rather than absolute illumination, and on prior event-based imagers [38, 39].The prototype sensor has been successfully applied to accomplish stereo vision [40, 41, 42]. This chapter helps develop an understanding of the functioning of the DVS sensor.

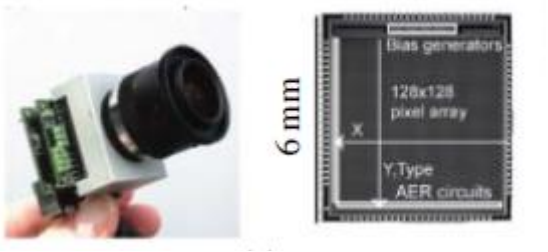


Figure 17. Prototype Silicon Retina with USB 2.0 interface [53]

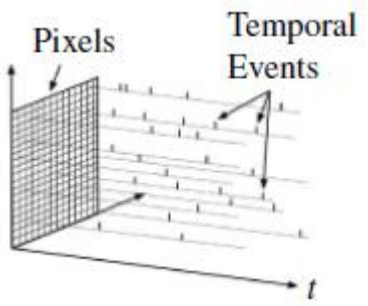


Figure 18. Temporal Asynchronous Events [53]

NEUROMORPHIC EVENT-BASED VISION

Biological vision is very different from existing imaging systems. Modern imaging systems perform poorly when dealing with complex information from real-world environments which are processed, seemingly, effortlessly by animal or human brains. The fundamental difference between natural & man-made vision systems, which accounts for the superior performance of the latter, is the style of computation & processing architecture in the nervous system – which is completely different from the synchronous architecture & computation schemes ubiquitous in the silicon-based processors today [43, 44, 45, 46, 47]. Biological systems adopt an asynchronous methodology i.e. process signals as and when they are received from sensory systems – there is no notion of continuously reproducing the full scene in view or polling the sensory system periodically. Frame-based schemes were never meant to be employed for vision sensing in real world environments.

Mead demonstrated [48] how the industrial standard VLSI technology can be used to develop circuits that mimic neural functions and to, thus, realize computational units that emulate the basic building blocks of the nervous system viz. neurons, ganglions, etc. This opened the doors to the exciting possibilities of constructing devices and realizing systems that combine the benefits of existing silicon-based fabrication & processing technology with the paradigms of biological systems. The fundamental inspiration behind such "*neuromorphic*" systems is to build circuits with computational architectures that realize asynchronous & massively parallel data-driven digital signaling at their cores.

Research in this field has perpetuated a major foray into biomimetic systems in the context of sensory abilities, most notably, vision. A variety of biomimetic vision devices have been implemented since the ground-breaking creation of a "silicon retina" by Mahowald and Mead [49][16]. It should be noted that biology has no notion of an image frame to represent visual information. Emulating biological vision, neuromorphic vision sensors feature massively parallel

preprocessing of the visual information at the pixel level, combined with highly efficient asynchronous event-based information encoding and data communication.

The sensor used in this effort is a 128x128 CMOS imaging device – referred to as the Dynamic Vision Sensor: DVS128. The underlying circuit of the DVS is an array of autonomous pixels asynchronously generating continuous-time "spike" events that encode relative changes in pixel illumination. It emulates the magnocellular transient pathway of the human retina and captures dynamic information from natural scenes at very high temporal resolutions [35]. The generated events are communicated as data packets conveying the responding pixel's *x*, *y* array address, ON/OFF polarity & a highly accurate value of the time when the event occurred. Optimization techniques are employed in data transfer operations so as to not lose the event's crucial timing information.

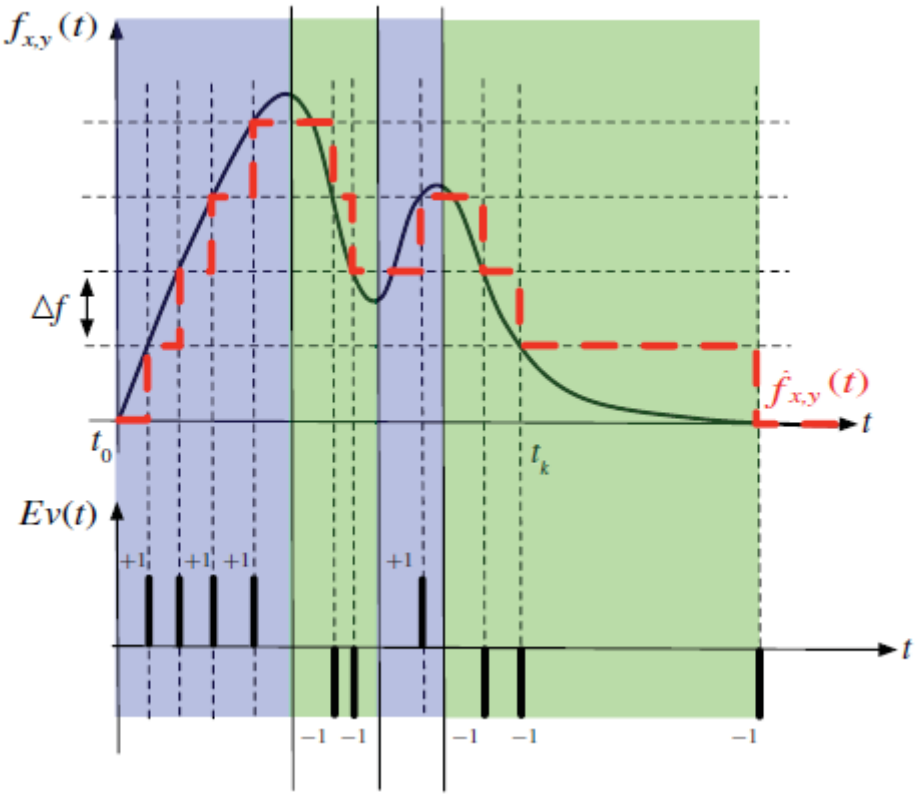


Figure 19. Codification of pixel gray variations into temporal contrast events [53]

Each DVS pixel is independent and responds to changes in log intensity. When this change in exceeds a set threshold, an ON or OFF event is generated by the pixel depending on whether the log intensity increased or decreased. The asynchronous spikes are logged as digital data packets containing the x, y co-ordinates, polarity (ON/OFF) of the event & the timestamp of the event.

Since the DVS is not clocked, no events are lost due to lack of timely polling. The timing of events can be conveyed with high temporal resolution of the order of microseconds [1]. With such temporal resolution capabilities, this vision sensor opens up new avenues for designing methods in computer vision which are impossible with standard frame-based devices. Since each pixel can be individually refreshed every 10^{-4} seconds, these sensors can capture very fast dynamic scenes at "effective frame rates" of tens of kilohertz. This can lead to drastic improvements of many popular algorithms – especially those related to dynamic changes in the observed scene viz. tracking. Thus, many limitations encountered in conventional frame-based vision are expected to be solved or reduced to some extent.

However, for all its salient features, the DVS does have certain shortcomings when juxtaposed with frame-based imagers and very different approaches need to be adopted to address classical computer vision problems with its unique architecture. It's architecture also gives rise to some roadblocks while addressing these problems. To gain an insight into these issues, it would be a good idea to study and analyze previous work in this field. Although, majority of the efforts in this field work towards achieving stereo vision & addressing related issues using the DVS, the observations in the various approaches help understand the limitations of using the retina in image processing applications. This analysis of the existing event-based algorithms helps understand the 3-D nature of the spatio-temporal events and brings to light the various causes of non-synchronicity and identifies numerous possible sources of noise while generating the AE representation. The work is detailed below.

EVENT BASED EPIPOLAR GEOMETRY

Background

Epipolar geometry is the mathematics behind stereo-vision i.e. matching between corresponding points of 2 views of a 3-D scene. Traditionally, epipolar geometry is developed for frame-based imaging and conceptualizes a fundamental matrix which contains geometric correlations.

The fundamental matrix is a basic tool in the analysis & correlation of scenes captured by two uncalibrated cameras. The fundamental matrix captures geometric information and enables evaluation of the epipolar geometry for uncalibrated perspective views [50]. It may also be used to reconstruct the scene from two uncalibrated views up to a projective transformation [51, 52].

An attempt has been made to link epipolar geometry to the temporal activation of pixels and adapt it to event-based sensors in [53]. A computational model of asynchronous event-based sensors has been developed that will be used to estimate the stereo rig's epipolar geometry

corresponding to central and more general cases.

Algorithm Description

The basic principle behind epipolar geometry with event-based vision sensors is the fact that, considering a short time interval, it is highly unlikely that 2 uncorrelated events will not have timestamps within a short period of each other. This link between stereo vision & temporal dynamics is exploited.

A) Fundamental Matrix for Still Images

A fundamental matrix relates corresponding points in 2 images with overlap, while also accounting for rotation & translation. As shown in [53], for corresponding points with coordinate vectors p & p', the fundamental matrix can be estimated using the equation

Hence, if $p = [u\ v\ 1]^T$ & $p' = [u'\ v'\ 1]^T$, we get

$$\mathbf{p}'^T F\mathbf{p} = 0 \qquad (1)$$

which works out to –

$$uu'f_{11} + uv'f_{21} + uf_{31} + vu'f_{12} + vv'f_{22} + vf_{32} + u'f_{31} + v'f_{23} + f_{33} = 0 \qquad (2)$$

This generates a set of linear equations of the form

$$Af = 0$$

with $A = [uu'\ uv'\ u\ vu'\ vv'\ v\ u'\ v'\ 1]$.
Hence, minimum 8 points are required for a unique solution. This approach is also called the 8-point algorithm, which is the most popular algorithm for estimating the essential matrix introduced by Longuet-Higgins [54].

B) Event-based Approach

It is observed that, ideally, an event in space will warrant changes in reflectance in 2 retinas observing the same scene at the same time.

Say, $Ev(m^i_k, t_1)$ & $Ev(m^j_l, t_2)$ are generated due to the event M(t) in space. These are related by the classic geometric relationship

$$(m^i_k\ 1)^T = P_i(M(t)\ 1)^T$$

where P_i is the projection matrix associated to the first retina - C_i.

Note that, despite the earlier statement, different timestamps are allotted. This deviation from ideality is caused as the DVS pixel is non-ideal in many respects – constrained by electronic architecture. As noted in [53], it's limitations include limited contrast sensitivity, variable latency from visual stimulus to event generation, random jitter in the event timing, and background event activity due to noise and leakage phenomena. At the system level, additional impacts on the event timing precision due to limited communication channel bandwidth, data collisions and asynchronous bus arbitration are observed. In a standard application environment, majority of these are negligible, and only latency and jitter are of concern.

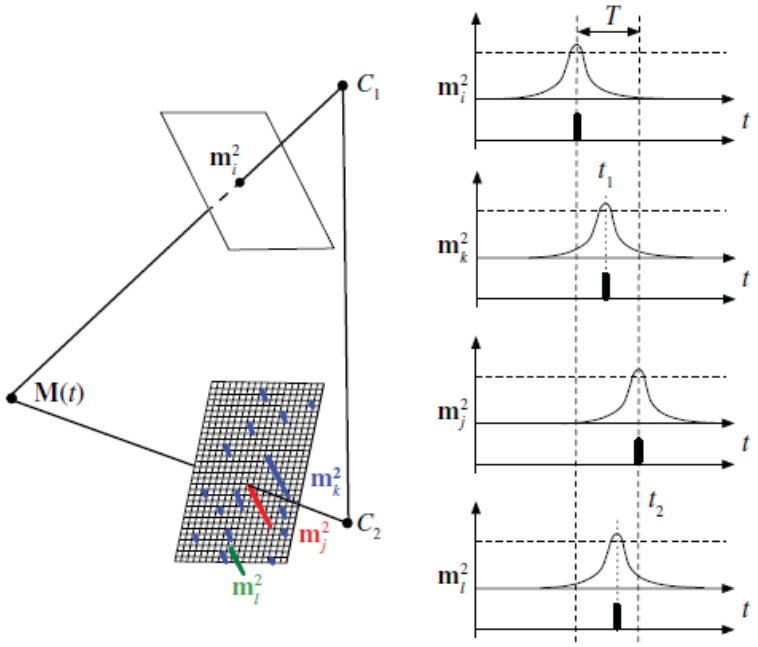


Figure 20. An event M(t), evokes a response from pixel mi1 at t1 and mj2 at t2. Due to the non-zero difference in latencies, other pixels with no a priori relationship with M(t) can generate events at the sensor output in close temporal proximity to t1 [53]

Thus, even the difference between the two timestamps cannot be quantified rigidly. Hence, t_2-t_1 is defined in probabilistic terms i.e. a stochastic time window is used and corresponding events are identified from the entries in this window. The time window is preset using knowledge of the process parameters.

In developing the parallels, a co-activation set is defined in [53]

$$A_{k,j} = \{t_k \in [t_j - T, t_j + T] \mid Ev(m^2_k, t_k) \neq 0 \text{ if } Ev(m^1_j, t_j) \neq 0\}$$

This is the set of all temporal activations of m^2_k in the other retina.

$$P(A_{k,j}) = P(\, Ev(m^2_k, t_k) \mid Ev(m^1_j, t_j) \neq 0\,)$$

The matrix of all these values $P(A_{k,j})$ is the correlation matrix B^j – which is now a coarse representation of the epipolar line as the probability of pixels' activation is directly linked to their possibility of match.

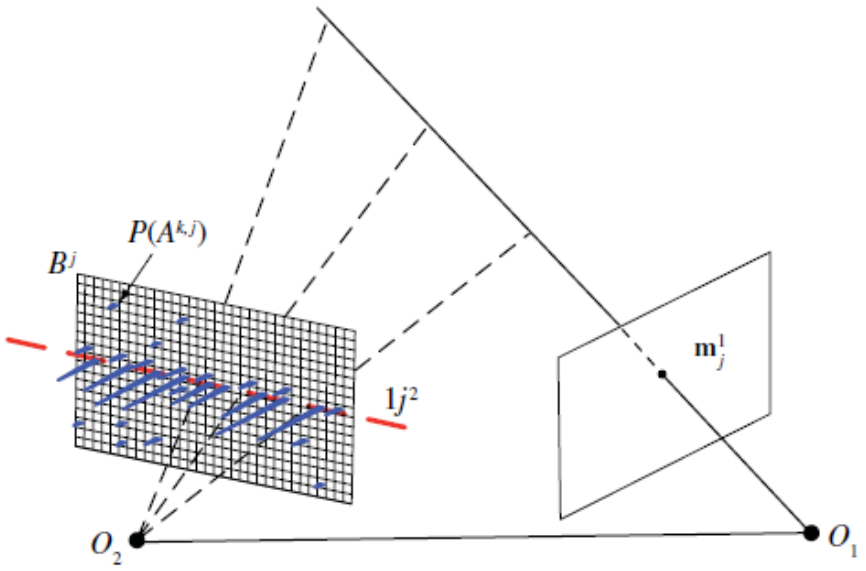


Figure 21. Epipolar Geometry from the co-activity probabilistic field [53]

These probabilities are calculated by measuring activity generated in both retinas due to 300 images from a video running on an LCD screen, taken at various depths. This setup is shown in Figure 22. The fundamental matrix can, thus, be estimated by the equation

$$I^2_j = Fm^1_j$$

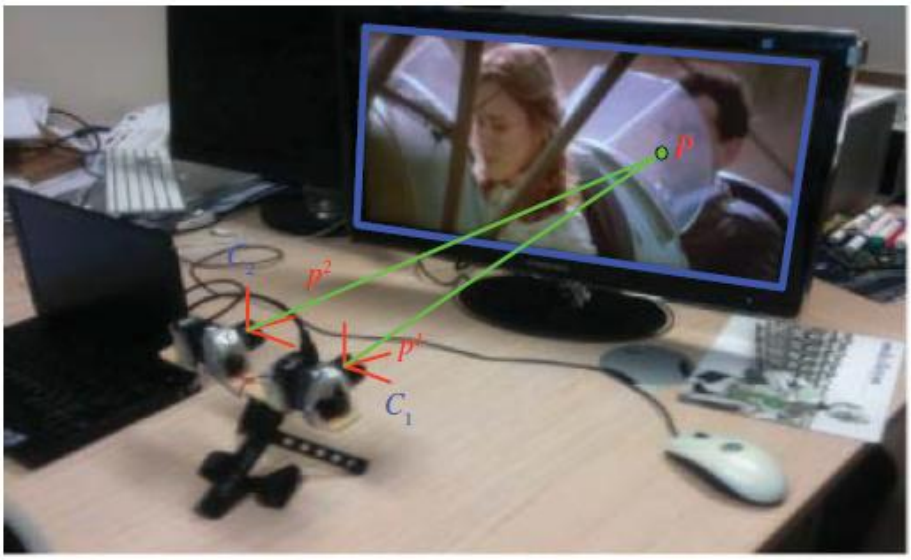


Figure 22. Experimental setup for calculation of probability coactivation field at different depths of the screen [53]

The lines, thus, obtained are compared with classically computed ones from a pair of frame-based cameras with minor deviations in the co-ordinates. The following algorithms are used in [20] for evaluating the epipolar line fit. They mainly differ in how the best pixel activity is computed and which method and data were used to compute $\mathbf{l_j}$

a) "Max-Fitting" Raw Position of the Maximally Active Pixels: In this algorithm, the epipolar line is fitted to the position of the maximally active pixels for every screen's position.

b) "Cg-Fitting": Gravity Center of B^j pos Threshold [*0, 1*]*:* This algorithm proceeds by thresholding every B^j *pos* in order to obtain the best active pixels. The position of the maximally active pixels is computed to be the the mean of the position of remaining pixels. This algorithm gives us subpixel position resolution of the max.

c) "Cov-Fitting": Covariance Fitting of l_j Using Max B^j pos > Threshold [*0, 1*]*:* This algorithm is analogous to the previous one. It differs in the fitting of $\mathbf{l}\,j$, which introduces an uncertainty measure into the location of the max position—highest coactivation location—using a covariance matrix computed from the position of the remaining thresholded points.

This approach can enable probing new geometries which prove difficult to address for frame-based ones.

STEREO VISION

If every event in one retina above can be matched to a unique event in the other retina, using the epipolar geometry calculated, stereo vision can be achieved. For this purpose, the fundamental matrix is calculated using events satisfying the time-window criterion and in addition to the timestamp constraint, a distance/co-ordinate constraint is used to add a complementary nearest neighbor criterion to the event correlation. The difference of latencies is estimated using fabrication parameters.
As in calculating epipolar geometry, the spatiotemporal correspondence set of events is defined by

$$S_{t1,t2} = \{e(\mathbf{p}, t) \mid t \in [t_1, t_2]\}$$

To discriminate matches, we can define a time window in which true matches are more likely to occur. Thus we define within a time window δt, so that for an incoming event $e(\mathbf{p}^i_1, t)$ in R_i (first retina), there exists a set $S^j(t)$ containing events in R_j (second retina) defined by (as in [55])

$$S^j(t) = \{\, e(p^j_k, t') \in R_j \mid \forall t', |t' - t| < (\delta t/2) \,\}$$

The B^j matrix generated has non-zero values at only a few places. The epipolar line is calculated according to

$$I^2_j = Fm^1_j$$

and it contains all possible matches of the event $e(\mathbf{p}^i_1, t)$ in R_j. We can then use the distances of possible matches—that lie within the temporal window δt constraint—to the epipolar lines. The set of possible matches M of an event $e(\mathbf{p}^i_1, t)$ is then decided by as shown in [55]

$$M(e(\mathbf{p}^i_1, t)) = \{ e(p^j_k, t') \in S^j(t) \mid \forall k, d(p^j_k, I_{ij}) < \Delta_p \}$$

where $d(p^j_k, I_{ij})$ is the Euclidian distance of p^j_k to I_{ij}, and Δ_p in pixels is a threshold representing the maximum distance allowed. The main idea of the stereo matching is that, considering a short time interval, it is very unlikely that two uncorrelated events happen at the same time and fall on the same epipolar line [55].

The following 4 criteria are additionally defined in [55] to identify unique stereo matches

1) The polarity of the events, taken into account so that ON events from one retina are matched only to ON events in the other retina.

2) Similarly for OFF events, the uniqueness constraint, applied so that an event cannot be matched with more than 1 event in the other retina.

3) The ordering constraint, which a classical binocular stereo constraint on ray orientation.

4) The temporal activity of pixels, where two pixels are matched if their number of total emitted events over a time period is close.

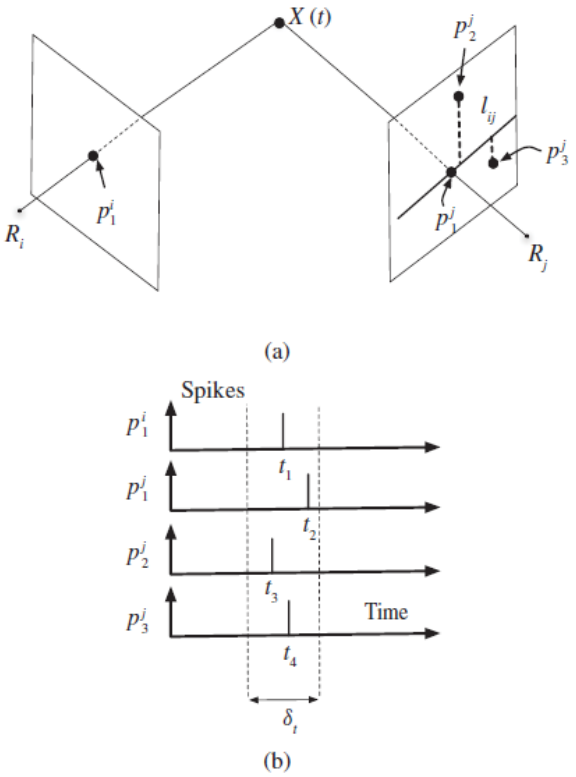


Figure 23. (a) Two events generated by a real point X at time t, are detected in the 2 retinas at the corresponding pixels at slightly different times t1 & t2 (b) With the criteria listed, there are very few possible matching candidates for the event e¬i1(pi1, t1)

INFERENCES

The study and examples presented in this chapter can help understand the working principle of the DVS sensor and deliberate on the necessity of doing computations purely on the basis of events that occur – with respect to both, time & location. The huge number of events generated also necessitates keeping the number of packet iterations at a minimum, to prevent intensive computation and performance lag issues.

Some conclusions relevant to the direction of our research effort can also be made. Namely that the DVS events from an object in the observed scene, representing information retrieved from the same, possess temporal coherence. However, the high temporal resolution results in poor persistence of the observed scene. There is no access to static scene content. Thus, due to the conditions discussed in the previous chapter, the output of the DVS sensor, cannot be used to perform object recognition in its raw form. Hence, it is essential to process the event information to incorporate the quality of persistence and, thus, take the first step towards developing object recognition algorithms for event-based sensors.

CHAPTER – IV: ADDRESS EVENT TO FRAME REPRESENTATION CONVERSION

The motivation behind this research effort is to address object recognition using low-latency, low power event-based sensors so that, coupled with the advantage of high temporal resolution for tracking, the DVS can be developed as a good alternative for machine vision.

Various approaches to recognition and image correlation exist in the computer vision domain viz. contour matching, intensity correlation, color histograms etc. However, all of these approaches are highly data-intensive, drawing on frame information like color, static pixel clusters' qualities etc. Event-based sensors do not provide the luxury of access to such a multitude of information – for example, the event information does not tell us anything about the color at that pixel location. Moreover, due to high temporal resolution and the architectural reset function in the pixel, extracting information regarding comparable contours from raw event data is quite challenging. Hence, it is proposed to look into adopting a recognition scheme that is based on sparse data correlation. The popular feature-based schemes lend themselves naturally for this purpose and, hence, we pursue implementation of feature-based schemes.

However, as mentioned in the previous chapters, a certain degree of persistence of the observed scene i.e. static scene content is required to perform feature-based recognition. Moreover, in the frame-based approach that we propose to follow, distinctive features in the natural scene, being stable, would not possess temporally varying characteristics – hence, this persistence is not only necessary but can assist our efforts.
The DVS output in its raw form cannot be directly utilized for developing a feature-based scheme. This is due to the fact that objects in the scene can be discerned, from the event information, only when there is movement. When there is no movement, the information is quite indiscernible. As shown in Figure 24 – which is a snapshot of the output observed in the AEViewer (the event data logging and observing interface for the DVS, included in jAER) due to the movement of a retina observing a particular scene – apart from the fact that there are some vertical edges in the scene, not much can be evaluated from the scene.

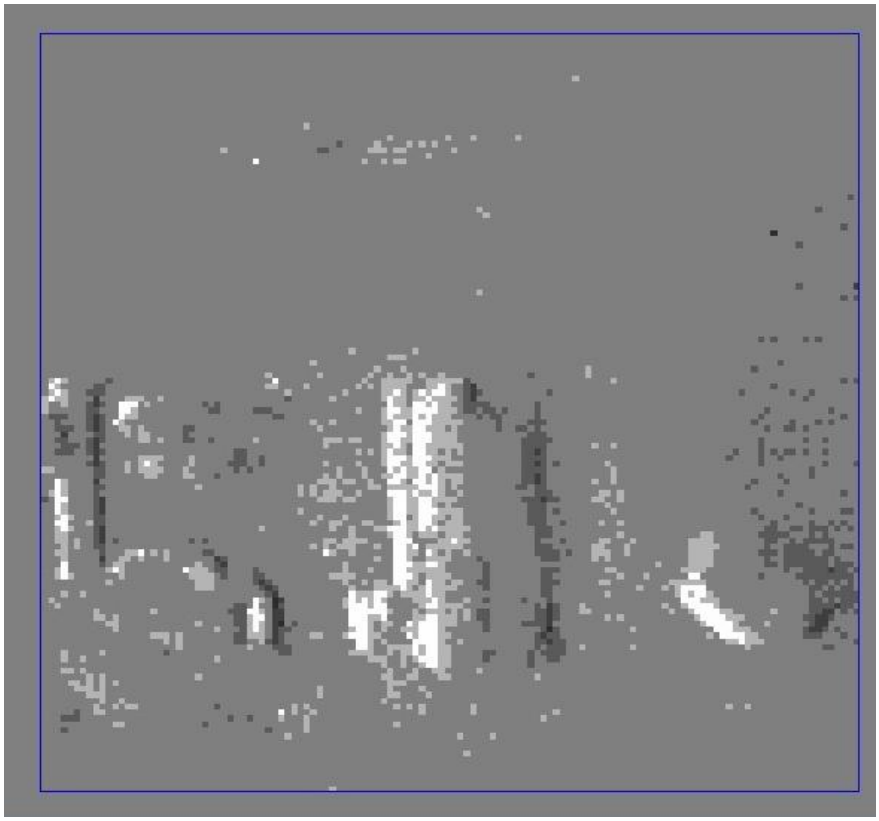


Figure 24. Events recorded from an observed scene in a split-second snapshot

ACCUMULATE FUNCTION

Posed with issue of generating an event-based representation providing access to static event by looking at events over a particular period of time, the 'Accumulate' function developed in jAER poses a possible solution. This function simply logs all the events from the instant it is called, at all locations on the pixel array. It maintains an infinite buffer at each pixel location and based on the number and polarities of the events collected, a scaled gray value is allotted to that location.

SHORTCOMINGS OF ACCUMULATE FUNCTION

The event data needs to be processed appropriately to create a representation, akin to frames, giving us access to the static content from the observed scene. However, running the 'Accumulate' function on the events generates an output as shown in Figure 25.

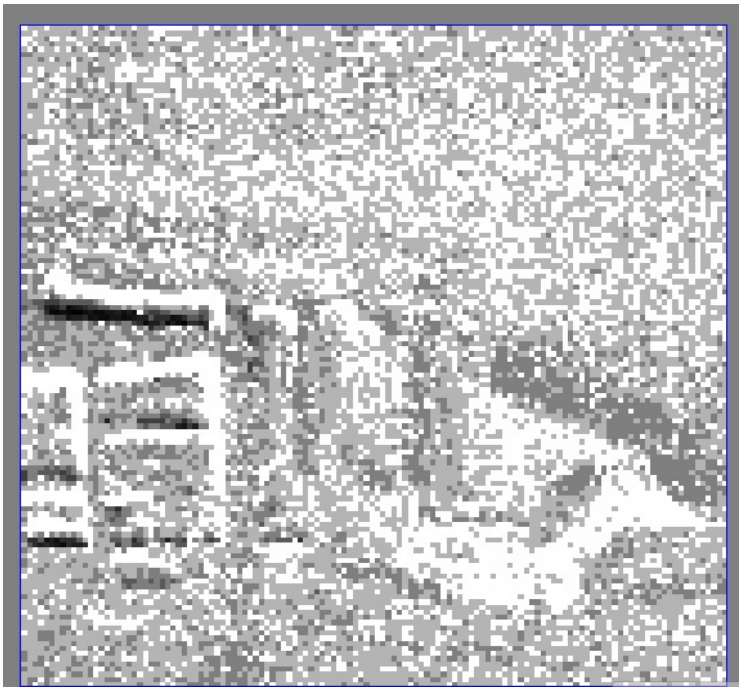


Figure 25. Output of Accumulating events from an observed scene

As is evident, this is quite indiscernible and, hence, unusable for object recognition, despite having the requisite properties of temporal coherence of the events and persistence. Its inadequacy can be explained by studying the nature of the ‘Accumulate’ function.

The ‘Accumulate’ function records events’ history from the moment it has been called. There is no mechanism via which it updates the representation being observed. Therefore, new scenes will simply be overlaid on the earlier ones making them difficult to identify. Hence, the ‘Accumulate’ function is useful for the purpose of recreating a static scene – however, if there are any changes in the scene artifacts of older objects will remain in the corresponding pixel patch and create an indiscernible milieu of events.Hence, the ‘Accumulate’ function is not the perfect option for our purposes. It should be noted that while it is necessary to obtain something analogous to frames, the dynamic information obtained from the events should be utilized so that the representation updates as new events come in and updates are evaluated at the corresponding locations only i.e. the ‘Accumulate’ function needs an ability to ‘forget’ i.e. drop out old events so that new temporally coherent events can be stored, enabling observation of the newer object contours and shapes.
One such structure that lends itself to our requirements as discussed above, is that of the Ring Buffer – a fixed size buffer in which old elements are pushed out as new events are added. This chapter explores the generation of such a 2-dimensional map representing the event space obtained from the DVS to enable addressing the development of feature identification & description for event-based sensors using the same. The various adopted methods devised to obtain such representations and implement address event to frame conversion, along with their merits, demerits and usability are discussed below.

APPROACH 1: PIXEL RING BUFFER

In this approach, a ring buffer of fixed size is maintained at every pixel position of the DVS – this size can be set to any desired value. The principle of a ring buffer is it implements a crude form of memory, whereby the stipulated processes to generate a 2-D representation of the DVS events are carried out only on the most recent events accumulated in the ring buffer. If the number of events since starting input at a particular pixel exceeds the fixed size of the ring buffer, they are pushed out to be replaced by new ones, thereby maintaining the 'most recent' representation. The gray value allotted to that pixel is decided by appropriately scaling the accumulated sum of the events' polarities at the pixel location

Figure 26. The representation of static scene content obtained by maintaining a ring buffer of length = 5 at each pixel location

As is observed in Figure 26, the representation generated in this case – although riddled with noise – shows a better representation of the observed scene than the AEViewer output presented above. 3 separate window frames in the background, a chair's back rest in the foreground to the left can be roughly identified.

APPROACH 2: PIXEL RING BUFFER UTILIZING MOST RECENT EVENTS FOR UPDATION & ANALYSIS

This approach follows a similar scheme to the one detailed above implementing an additional criterion. This criterion dictates that every time an event occurrence is detected at a particular pixel position, the timestamps of the events already in the buffer are compared to the incoming event's timestamp. If they exceed a preset window limit around the incoming timestamp, they are dropped from the buffer – thereby enabling maintaining an even more recent representation of the events. The output shows close resemblance to the pixel buffer implementation.

APPROACH 3: FRAME RING BUFFER

This approach closely follows the initial pixel buffer approach, except, in this case, only one buffer is maintained for all events rather than one at each pixel location. Events, irrespective of where they occur, are added to this buffer and dropped to make place for new events as they come in. Map values, however, are updated as before – based upon events recorded at the particular coordinate location only. This approach employs a higher rate of updation, as compared to the earlier ones.

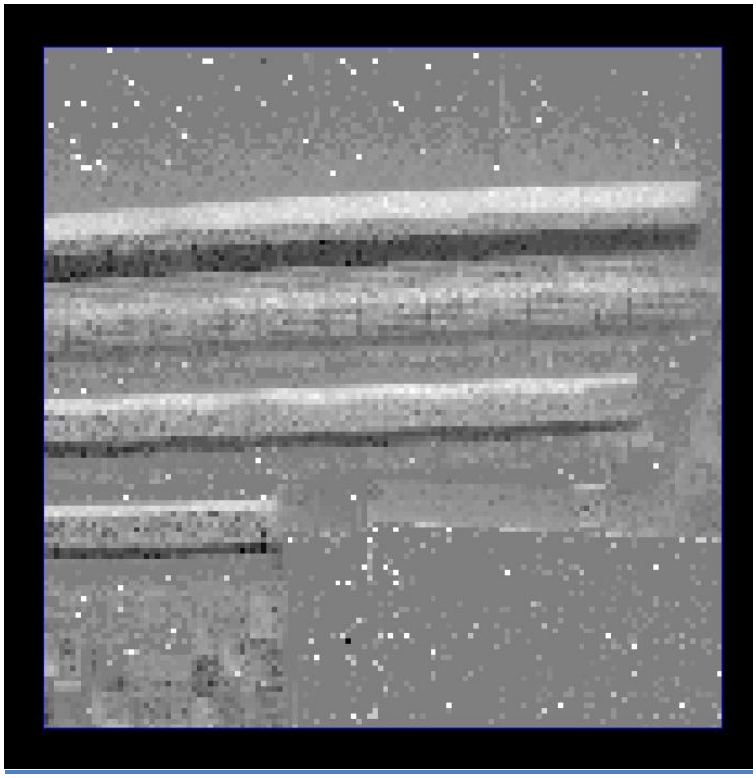


Figure 27. Frame Buffer of large size (~10^5) observing the scene

APPROACH 4: FRAME RING BUFFER UTILIZING MOST RECENT EVENTS FOR UPDATION & ANALYSIS

This approach is a similar extension of approach 3 as approach 2 is of approach 1. A timestamp criterion is included which dictates the dropping of events in the buffer that lie outside a preset time window of the incoming event.

REVERSE ENGINEERING USING SIFT/SURF-BASED FEATURES

Another proposed approach, following a different train of thought, is discussed below with analysis & critique of the same.

It is proposed to develop scale & rotation invariant cluster-based descriptors for the event-based DVS that can potentially aid in achieving object recognition using the same. However, since, there is no means of deciding which clusters are localized around distinctive features and, hence, no way of knowing which clusters can be used or how they can be used for identification with the DVS.

In order to address these issues, a back-calculating approach is proposed – Distinctive scene features would be identified using SIFT-based detectors and descriptors and be drawn onto the video input. At the same time, their locations should be recorded continuously. These locations would be linked to the input from the DVS and activity at the corresponding locations in the DVS input stream would then be recorded and the log studied to make conjectures on how to model an event-based descriptor.
This method was thought to be a structured approach towards modeling event-based descriptors. However, certain arguments identify the following demerits that led to abandoning of this approach –

- SIFT-based detectors and descriptors engineer features by performing mathematical analysis using pixel intensity values from a natural scene. These are not necessarily points of 'interest' or the most distinctive points in the natural scene – they are distinctive as per the mathematical analysis performed in the SIFT approach.
- Since the DVS output follows the natural scene, without any mathematical treatment of the same, it would be impractical to study the activity at points mathematically engineered by SIFT and model an event-based descriptor on an analysis of the same as these might not, necessarily, be 'interesting' features of the natural scene.

Thus, this approach was not pursued enthusiastically.

OBSERVATIONS

Of all the aforementioned methods, the approach of maintaining a ring buffer at each pixel location, exhibits the least amount of information loss and generates the most tangible & useful reconstruction of the observed scene.
The frame buffer approaches does not produce a discernible image. A possible reason for the same is the low amount of information represented in the image holder. Hence, majority of the pixels assume an intermediate gray value while those with very high updation rates stand out with values closer to saturation.

Incorporating a factor of time leads to loss of information and then performing detection can lead to altering of detected features. The objective behind developing such a representation is to enable feature detection from the scene at interesting regions, even when there is not enough discernible activity. Thus, incorporating a temporal criterion on the representation is counter-productive to this effort.

ANALYSIS OF PIXEL BUFFER APPROACH

Why does it work?

As is observed, the initial pixel buffer approach provides the best platform to build upon. The reasons behind its usefulness in regenerating the scene are explained below.

Figure 28 depicts the temporal variation of the observed scene and the events generated in the DVS as a result of the same. The plot in red is the integral form of generated events (depicted on the timeline below) and as can be clearly observed, the discrete event plot build upon the previous event levels and follows the natural temporal variations in the scene quite closely.

At each transition where events are shown to occur, it is observed that the discrete events' plot represents the same value of intensity in the observed scene location. However, since the DVS128 generates events based on changes in temporal contrast in the scene, not all the slight intensity variations are discerned, resulting in an approximate recreation of the scene. To ensure updation of the scene as stated earlier, the size of the pixel buffer should be optimal – small enough to ensure good recreation of the scene and large enough to account for the required bandwidth.

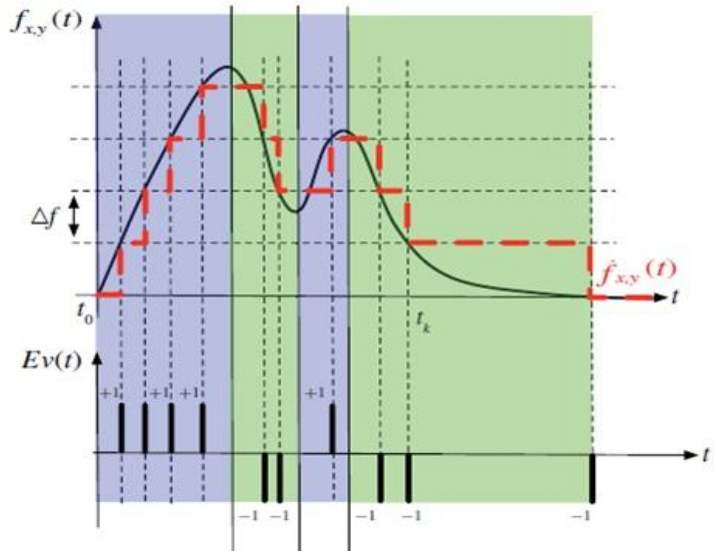


Figure 28. Generated events stream from the time-varying input of the observed scene

Shortcomings of the Pixel Buffer

In the pixel buffer approach, it is observed that the scene in view appears shadowed behind a random pattern. This pattern shows presence of artifacts of previously observed scenes and is observed to become more prominent as the size of the ring buffer is increased. The causes behind this pattern are sourced to the characteristics of the DVS pixel and investigated further below. These issues need to be addressed while working with the pixel buffer and the various methods to do so are also discussed.

1. **Background Noise:** Due to effects like thermal noise in the transistor circuits, it is possible that the current threshold for generating an ON/OFF event is exceeded, thereby generating an event despite no change in the observed scene. This effect generates aliased ON/OFF events. Since the pixel buffer approach is built upon accumulating and integrating over the events, the fixed pattern will interfere with the desired functioning of the pixel buffer, by introducing excess events (interpreted as noise) that will affect the integral at each pixel position.
   This effect cannot be controlled externally or through any hardware settings of the DVS. Hence, the Background Activity filter is implemented to run in tandem with the application.

2. **Fixed Pattern Noise:** Due to the issues of mismatch in transistors during fabrication of the underlying circuitry, a fixed pattern noise is incorporated into it. This implies that some pixels will have a higher propensity towards generating more ON events and some towards generating more OFF events. Since the pixel buffer approach is built upon accumulating and integrating over the events, the fixed pattern will interfere with the desired functioning of the pixel buffer, by introducing excess events (interpreted as noise) that will affect the integral at each pixel position.
   The effect of this fixed pattern noise is that of a constant random bias. This is a well-analyzed classical issue is mitigated by implementing a Gaussian blurring on the representation obtained. As is observed in Figure 29, this is a better representation than the one presented earlier. The outline of the chair's back in the foreground, the 3 window sections in the back ground, the outline of a sofa to the right along with an upright rectangular object can be discerned.

   Convolutions with Gaussian kernels are tested with respect to speed of convolution and the quality of the output (noise blurring). The Gaussian kernel of size 5, as observed in experiments, presents the best possible option – as sizes lesser than this, do not provide blurring as effectively while larger kernels have the peripheral values beyond the 5x5 very close to or equal to zero, thereby nullifying the effect of extra pixels included. Thus, a classical noise filter helps rid the event representation of noise as well. Thus, implementing feature detection schemes on this filtered representation would be a wise

approach. However, as will be seen in the next chapter, a trade-off is reached with the time required for computation.

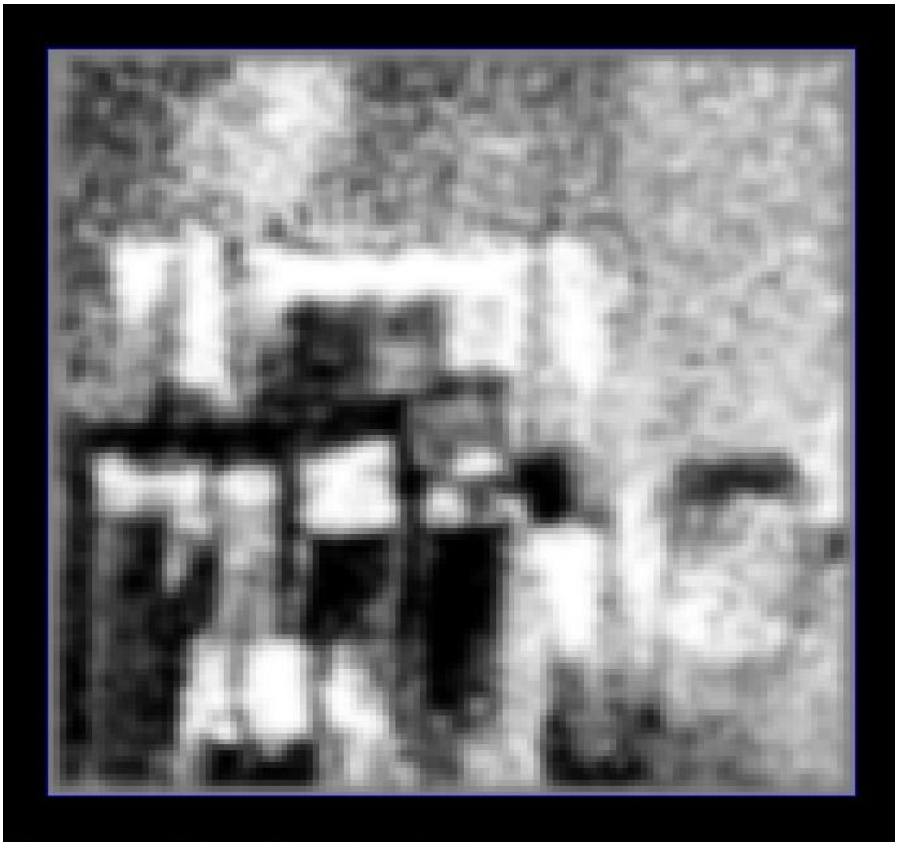


Figure 29. Output of Gaussian blur implemented on the pixel buffer representation

3. **ON/OFF Bias:** The settings of the DVS can be such that the base of the response level is closer to the ON response threshold or closer to the OFF response threshold, thereby generating a larger number of corresponding events. This threshold can also vary from unit to unit. This too can introduce excess events of a particular type creating aliases in the function of the pixel buffer. This is best understood by considering the following example.
   The retina being used generates more ON events than OFF events in the retina and it is observing a scene with an object moving in the direction of the illumination. The form of the object generates many ON events as it goes along. However, due to the imbalance between ON & OFF events, not enough OFF events are generated in the object's wake to mitigate pull the pixel integral value to the base value representing the patch being observed once the object has passed. This leaves artifacts in the form of trails of the object form as it had moved while observing the scene.

Hence, it is essential to set the ON/OFF bias to an optimal value to ensure a fair response to each event polarity.

4. **Dynamic Range:** The size of the pixel buffer can be set as desired from the AEViewer – the interface to visualize the DVS events. However, while a smaller size ensures better updation rate for a better live recreation of the scene, it might be insufficient to encompass the entire dynamic range of the pixel i.e. being able to handle events generated like very dark to very bright transitions.
   Thus, an optimal size of the pixel buffer is chosen, which extends over the entire dynamic range of the DVS.

5. **Score Initialization:** The integral value calculated at each pixel position initializes to the first event that comes into that pixel buffer based on the observed scene. Hence, there is no fair starting point for the entire pixel array. For example, on a global scale, pixels that start with observing dark portions of the scene and move towards less darker portions of the scene will generate ON events and the representation of the same will have a high gray value. However, a pixel region that moves from a bright portion to a less bright portion will generate OFF events and have a lower gray value than the aforementioned case. This, in effect, is an erroneous representation of the scene.
   Thus, to ensure a fair starting level, the entire pixel array is first exposed to a uniform surface under uniform illumination viz. a white paper, a black surface etc. before using the retina to observe any scene.

The pixel buffer approach is described in the algorithm below:

Algorithm I: Pixel Buffer

```
for all events in packet do
   if buffer at event location is not full do
      update integral map value by +/- 1 depending on ON/OFF event
   else do
      if incoming event has same polarity of that being pushed out do
         update integral map value by +/- 2 depending on ON/OFF event
      else do
         integral map value by +/- 1 depending on ON/OFF event
      end if
   end if
end for
```

The process of event-driven feature schemes based upon the pixel buffer is explained in the following pseudo code:

Algorithm II: Feature scheme

for all events in packet **do**
    update integral map value
    detect keypoints
    assign descriptors
end for

.

INFERENCES

The discussed methods develop various schemes for reconstructing the observed scenes in a representation analogous to frames. The strategy to generate 2-D representations using event information incorporates a quality of persistence in the event data and generates a perceptible recreation of the scene being observed in the output. As a usable representation has now been developed, we can focus on addressing the main issue of developing detection & description schemes.

CHAPTER – V: EVENT-BASED FEATURE DETECTION

As mentioned before, the feature-based approach to the process of recognition & image matching comprises of 3 stages – a) Feature Detection – wherein points of interest that can be robustly used for correspondence tests , b) Feature Description – generating a metric for correlation/comparison at each detected interest point and c) Matching – evaluating correspondence between sets of detected features. A similar staged approach is adopted for the event-based efforts.

As observed in the study of existing image processing paradigms, the detectors can be classified into 2 broad sections – a) detectors that identify features as extrema of convolution of the image with various kernels and b) detectors that identify features by evaluating a 'cornerness' function at each point in the image by rudimentary intensity comparisons of pairs of surrounding pixels. This chapter discusses the implementation of such existing detection algorithms adopted and applied to the 2-D representations of event information discussed in the previous section.

CONVOLUTION KERNEL – BASED DETECTORS

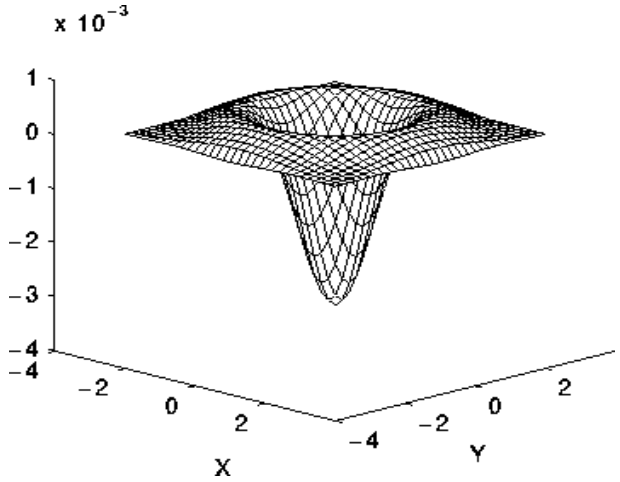
The approach adopted by vanguard algorithms like SIFT, SURF etc. follows convolution kernel-based detection algorithms. This entails convolution of the image with a kernel, which under various conditions of scale change and rotation has shown optimal isolation of interesting points in an image.

As discussed earlier, the Gaussian is the only true scale space function [8] and convolution of the image with a Laplacian of Gaussian (LoG) kernel, ultimately, gives the most stable and repeatable feature points [9]. The SIFT detector, approximates the LoG kernel by a Difference of Gaussians (DoG) and does convolutions using this kernel for improved performance. The SURF detector relies on a different kind of image representation i.e. integral images, and employs convolution with a Difference of Hessian (DoH) matrix to evaluate features. To address the issue of numerous keypoints being detected close to each other, a form of non-maximal suppression is employed which locates the absolute maximum (in the scale & spatial domain) in that neighborhood.

As an initial approach, it is proposed to develop suitable versions of these detectors for event-based representations and evaluate their feasibility in terms of computational costs of processing power and time required.

A] SIFT (Laplacian of Gaussian (LoG)) detector:

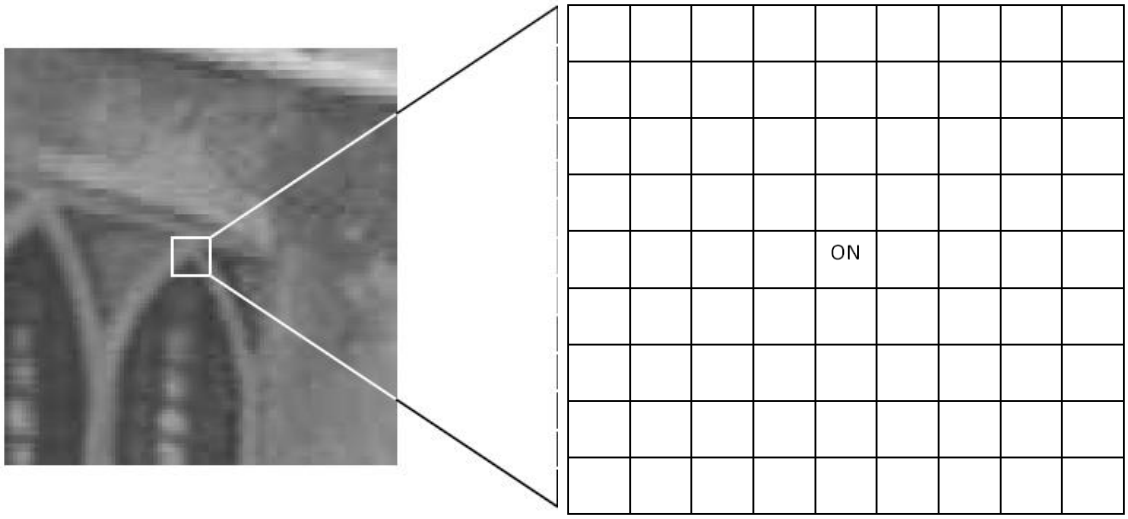
An event-based implementation of the LoG detector has been implemented. To work with discrete pixels as in the event-based scenarios, from a computational point of view, it is much more feasible to employ discretized kernels than their continuous counterparts. The continuous LoG kernel is approximated by hard-coded discrete values at each pixel location, based on the σ of the kernel.



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 30. Continous Laplacian of Gaussian kernel and discretized kernel implementing the same

This discrete form of the LoG kernel is used to perform convolution operations on the 2-D image representation obtained from the events, as discussed in the last chapter. The convolution is performed on an event-basis. This implies that as and when an event comes in, the discrete kernel being used is centered at the event location and the map values in the neighborhood and within the purview of the kernel, are updated by a value dictated by the magnitude of map update based on the nature of the event coming in and proportional to the kernel entry at that location. This can be visualized as in Figure 31.



(a)

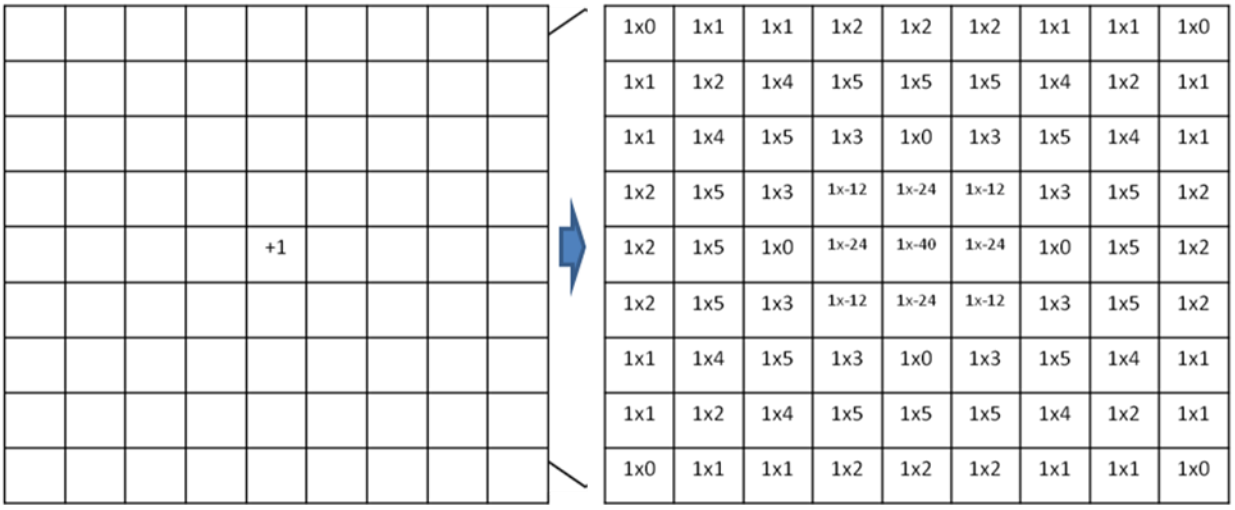


Figure 31. a) Extent of the image area that is affected when an ON event arrives at a pixel location as shown (b) The LoG kernel update due to the incoming ON event

It should be noted here, that the convolution is done on the initial response of the representation itself and not on the Gaussian-smoothed version. This is due to a trade-off observed between the observed detection performance and computation time. It is observed that when implemented on the Gaussian-smoothed representation, the computation time increases and can compromise real-time performance while ensuring accurate detection. However, implementing the LoG on the normal representation gives real time response and has proportionately less number of keypoints detected at noise responses.

Hence, the update takes place as shown in Figure 31.

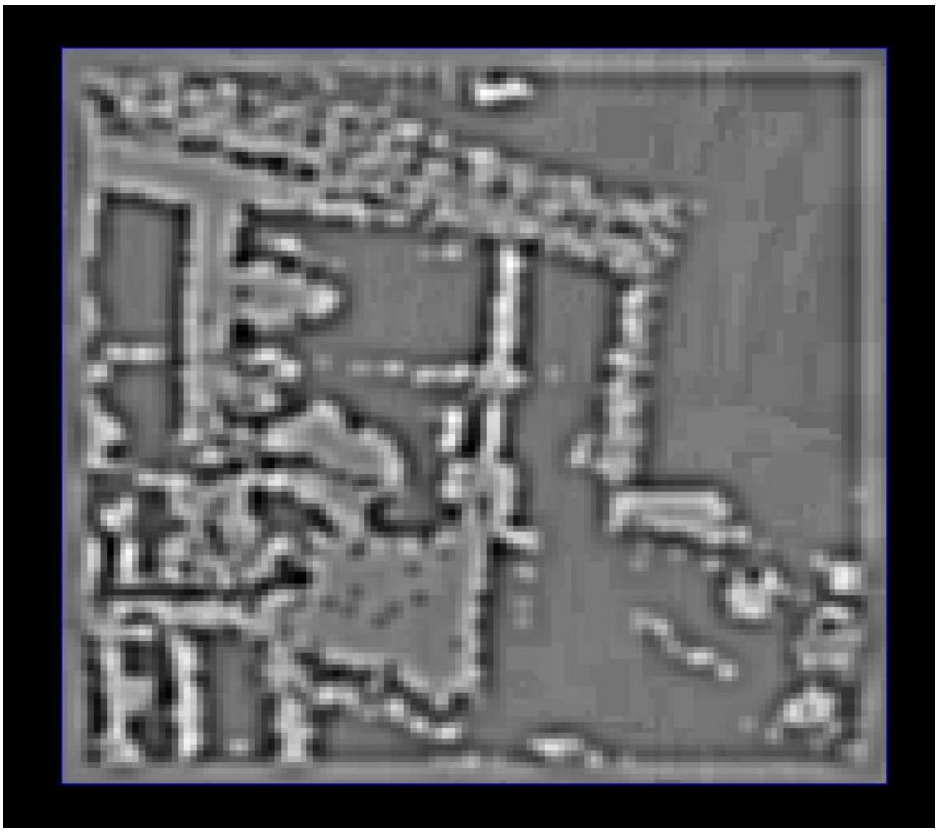


Figure 32. Convolution of the observed scene (slightly offset) with the Laplacian-of-Gaussian kernel detector

Non-Maximal Suppression

As this investigation is a first foray in this direction, the issues of scale are to be addressed in a later implementation. Hence, for the purposes of this effort, non-maximal suppression is addressed in the spatial domain only. It is implemented within the purview of the kernel itself – after performing the convolution operation using the kernel, the maximum response within the kernel's boundaries is identified and chosen as the feature of interest in that region. The selected features are then annotated and drawn onto the AEViewer as shown in Figure 35.

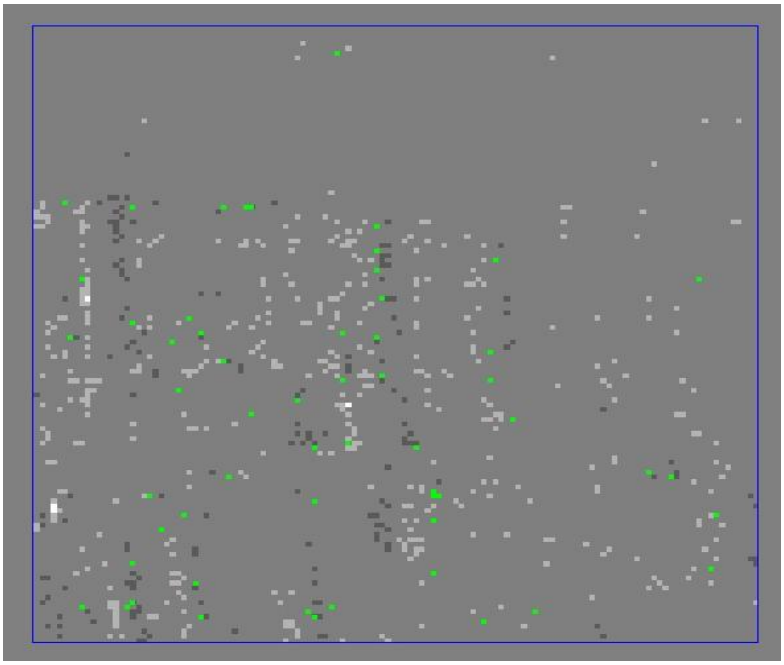


Figure 33. Green points mark the locations of the extrema of the Laplacian-of-Gaussian function. Candidates with scores within 70% of the global maxima are included.

The green dots displayed on the scene correspond to the extrema of the output generated by convolving the scene representation with the Laplacian of Gaussian kernel. The annotated keypoint candidates include those having scores (grayvalues) within 70% of the global maximum. It can be verified with the original scene that these keypoints are detected along the various edges present or highly contrasting regions in the scene. The implementation of the event-based LoG detector can be found in the '*LaplacianOfGaussianKernel*' class detailed in Appendix A.

The following pseudo code best illustrates the algorithm:

Algorithm III: Event-based LoG convolution

```
for all events in packet do
   if event location in range do
      for all pixels in LoG extent do
         if coordinate is keypoint do
            remove keypoint status
         end if
         obtain corresponding LoG kernel entry for updation
         update map value at that coordinate
         check if local extremum
         check if global extremum
         allot scaled grayvalue
      end for
   end if
   if local maximum do
      assign keypoint status
      annotate onto event viewer
   end if
end for
```

B] SURF (Difference of Hessian (DoH)) detector:

As in the above discussion, it is much more feasible to employ a discretized DoH kernel for our purposes. However, working on the lines of the SURF detection scheme requires generation of integral images, which for an image 'u' is defined as –

$$U(x, y) = \sum u(i, j) \mid (i \leq x, j \leq y)$$

which is then convoluted with the DoH kernel. However, for the event-based representation, this integral image would have to be calculated and updated for each event and the convolution operation would require performing numerous extra addition operations per event – affecting computation speed and performance immensely.

With these shortcomings in perspective, this approach is not actively pursued in our effort.

BINARY DETECTORS

The newer algorithms in this field e.g. FAST, belong to this class. This entails evaluating a 'cornerness' score at each pixel in the image. Such a function is, typically evaluated by deciding an optimal pattern around the central pixel i.e. where the event occurred, and performing simple intensity comparisons between the central & surrounding pixels. If the number of pixels in the neighborhood brighter or darker than the central pixel, exceed a certain value, then it is chosen as a feature of interest.

The fore-runner algorithm in this domain, FAST, uses a pattern in the form of a Bresenham circle of radius 3 centered at the pixel under observation. Under the condition that there is a set of 'n' contiguous pixels in the neighborhood which are brighter or darker than the central pixel, the pixel is classified as a feature/corner – based on the implementation, n = 9 or 12.
As in the previous case, it is proposed to develop suitable adaptations of this algorithm for event-based representations and evaluate computational costs.

A] FAST Detector

The event-based adaptation bears a very close resemblance to the frame-based implementation of the FAST algorithm. The very same steps employed on a frame are implemented on the 2-D representation of events that we generate using the ring buffer at each pixel. However, a plethora of noise signals come in with each packet from various sources and are incorporated into the generated representation. This representation, if used directly, will, possibly, lead to spurious detection of features. Hence, the representation is smoothed (blurred) with a Gaussian kernel, acting as a noise filter. This smoothed representation is then used as the base for detection of features.

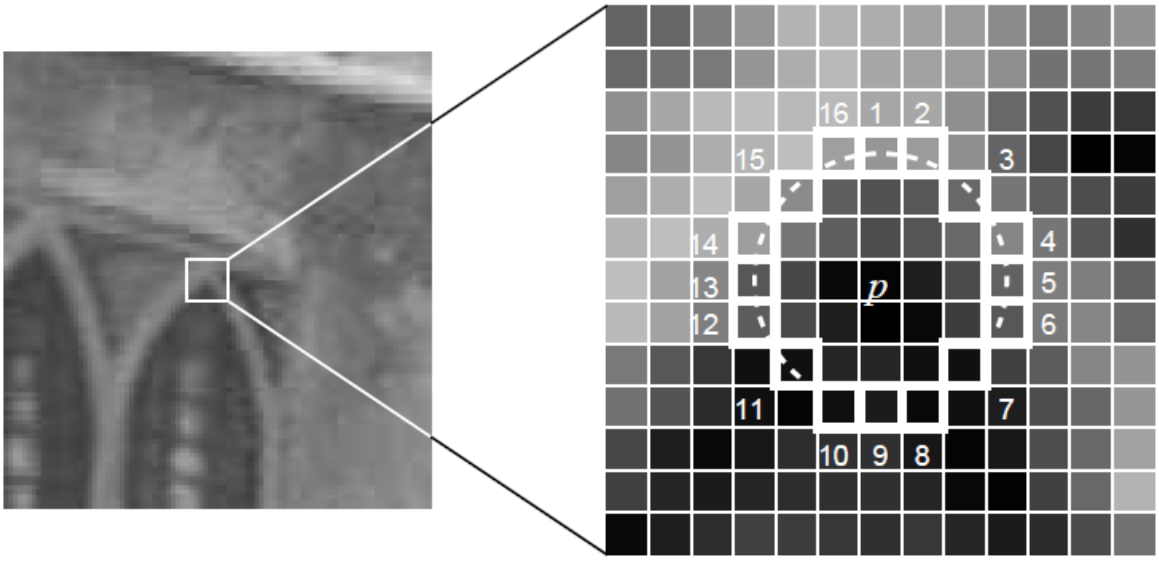


Figure 34. The FAST pattern used for event-based corner detection

As an event comes in, its value in the map is updated and the corresponding gray value of the pixel in the smoothed representation is compared to its 16 neighbors in a Bresenham circle. If a set of 12 or more contiguous pixels, that have a grayvalue greater than or less than the central pixel by a certain threshold, can be identified in the neighborhood, the point is chosen as a feature of interest. Only, if this condition is satisfied, the pixels in the neighborhood pattern are also checked for the 'cornerness' and appropriately updated.

The implementation of the event-based FAST detector can be found in the FAST class presented in the codebase of Appendix A.

INFERENCES

Thus, 2 different types of schemes (convolution kernel based and binary comparison based) are developed for feature detection. It is observed that the detected keypoints tend to lie in regions of high spatial relevance. Thus, the next step of describing the local regions for matching can now be addressed.

CHAPTER – VI: EVENT-BASED FEATURE DESCRIPTORS

Building upon the keypoints detected as per the schemes presented in the previous chapter, we can now proceed to the second step of describing local regions in feature-based image matching. In this chapter, we focus on developing descriptors for the event-based representations generated as a metric to perform matching.

As in the case of detectors, descriptors can be classified into two broad categories – a) descriptors that are built as a histogram of gradients, calculated using local neighborhood intensity values and b) descriptors that are composed as binary strings with each bit corresponding to a pair-wise intensity comparison of pixels at locations sampled from a preset pattern.

However, the former approach i.e. descriptors based on histogram of gradients, used by traditional algorithms like SIFT, SURF etc. is evaluated as quite complex as opposed to the latter and also more computationally intensive as it entails calculation of gradients in cellular blocks in the neighborhood of the central pixel - and if distinctiveness is desired, the number of such blocks should be optimum. On the other hand, the latter, used in such algorithms as BRIEF, BRISK, FREAK etc. requires only pair-wise grayvalue comparisons of pixels in a particular pattern around the central pixel.
The latter also lends itself naturally to faster matching schemes which is desirable as real-time performance with the DVS is of the essence. Matching in the latter scheme is a simple calculation of the Hamming distance between 2 vectors – an operation that can be done quite optimally with modern computer architectures – and can be speeded up a coarse-to-fine matching scheme. However, in the case of the former approach, matching needs to be done using Euclidean distance matching of high-dimensional vectors – which, irrespective of any optimization algorithm employed, would show slower performance than a Hamming distance calculation.

Thus, efforts are directed towards implementing only the binary comparison-based feature descriptors.
The next question to address is what kind of pattern to use. This is an issue being addressed in binary descriptors employed in frame-based vision as well. Hence, to obtain information about a proof of concept, various basic patterns are created and tested. This structured descriptors' effort is equivalent to developing descriptor patterns on the lines of to a structured BRIEF effort. As is the case in frame-based patterns, a rigorous qualitative or mathematical analysis of these descriptors is out of the scope of this effort.

BINARY DESCRIPTORS

The various implemented descriptor patterns are presented in Figure 37. The pixels marked with ‘X’ are the pixels at which an event occurs. The events in the surrounding on which comparisons are performed and are used to generate a descriptor, are shown around it.

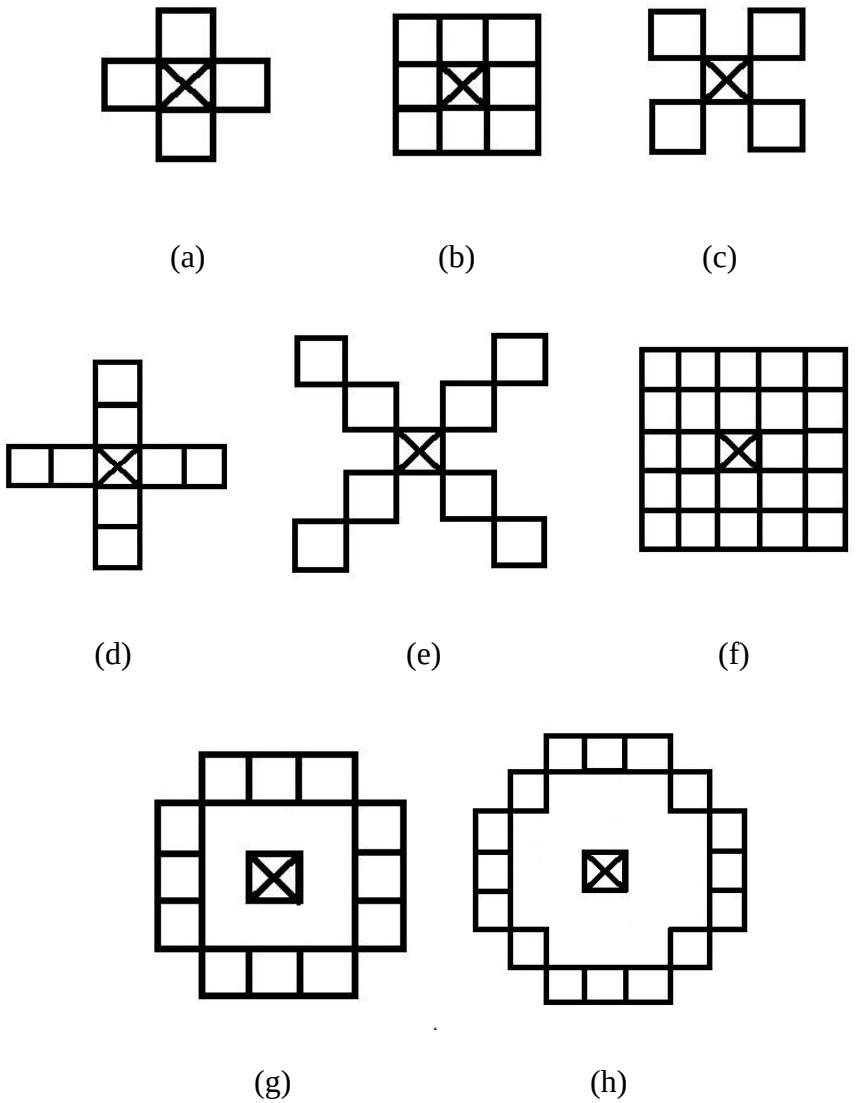


Figure 35. Pixel patterns for (a) CROSS (b) SQUARE (c) SQUARECorner (d) CROSSTwo (e) SQUARECornerTwo (f) SQUARETwo (g) OCTAGON (h) BCIRCLE descriptors

The process of forming descriptors is as follows.

Descriptors are constructed for the entire list of keypoints, which is updated with every packet. However, if the keypoint is repeated in the list, only the relevant pairs in the descriptor are updated.

If a pixel location is identified as a keypoint, the coordinates are extracted. The chosen descriptors' pattern is then centred around this point and the corresponding coordinates are noted. The gray value at these locations are compared to every other gray value in the pattern's set of coordinates. A value of 1 is recorded on the descriptor string if the preceding coordinate's gray value is greater than the next coordinate's gray value and 0 if lesser. Such comparisons of N pixels would lead to N.(N-1)/2 comparisons and, hence, a descriptor string N.(N-1)/2 bits long. Hence, for example, the CROSS descriptor generates a 10-bit string, the SQUARE descriptor generates a 36-bit string, the BCIRCLE descriptor generates a 136-bit string etc. The length of the string is proportional to its distinctiveness. The generated bit string descriptors are tehn written out ad stored in a file in the system.

The following pseudo code best illustrates the employed algorithm:

Algorithm IV: Event-based LoG convolution

```
for all points p ∈ detected keypoints do
   if p does not have descriptor do
      construct key point descriptor based on chosen pattern
      evaluate descriptor string
      write out descriptor string to a file
   else do
      for all events in packet do
         if event ∈ descriptors' map do
            update corresponding descriptor strings
         end if
      end for
   end if
end for
```

INFERENCES

Numerous descriptor patterns have been developed to describe the local region around each keypoint. The implementation of each can be studied by referring to the corresponding descriptors classes as listed in Appendix A.

CHAPTER – VII: MATCHING

As discussed in the chapter on event-based descriptors, efforts are directed towards working with binary comparison-based descriptors. One of the major reasons to pursue this line is that these schemes enable implementation of fast matching algorithms which help keep the feature-based matching operations real-time.

As described earlier, the binary descriptors, being bit strings, the comparison is simply an evaluation of their Hamming distance – the number of dissimilar bits being a measure of their dissimilarity. In digital architectures, this simply translates to a bitwise XOR operation and a bit count – a computationally cheap and speedy operation [29].

To perform matching using an event-based sensor, initially, a scene or a figure to be recognized is presented to the silicon retina. The keypoints detected along with their corresponding descriptors, are then stored in a file. To evaluate the matching performance, similar scenes are presented to the retina: the features detected in the scene in real time, are then matched to those stored in the file. As a metric for matching, the Hamming distance between all pairs of descriptors is calculated. The minimum Hamming distance for each descriptor from the reference is recorded. In case of a good match viz. the same scene being observed, this minimum should result in a small value (~0) whereas, in case of very different scenes being observed, this should be a relatively larger value. The results of the correlation tests for various descriptors on 2 passes of a particular scene are presented in Table 1 below.

Descriptor Type	Descriptors from Same Scene	Descriptors from Uncorrelated Scene
CROSS	100%	91.30%
SQUARECorner	83.33%	8.33%
OCTAGON	90.63%	81.3%

Table. 1

In this process, the retina is first shown a scene and descriptors from the scene are recorded into a file. Later, a pass is made over the same scene and descriptors are recorded from it. A 3rd recording of descriptors is done over a completely uncorrelated scene. The 3 sets of descriptors are then compared to see the extent of match or mismatch.

As presented, the keypoints are only matched qualitatively. It is observed that every keypoint in the generated set has at least one match in the other corresponding set whereas a certain degree of mismatch is identified for the uncorrelated scenes as well. In any case, the correlation with the same scene gives a better correlation match than the uncorrelated. This is, by no means, a true measure of correspondence as we overlook the cases of false positives.

Presenting quantitative results and comparisons of the schemes developed with existent algorithms like SIFT, SURF etc. is out of the scope of this effort. The reason behind this shortcoming is the fact that the performance of existing frame-based recognition algorithms like SIFT, SURF etc. is tested on a myriad of standard databases of still images taken under preset illumination conditions and satisfying various other criteria. These databases are manually labeled through tedious processes so as to enable identification of false positives, false negatives and other erroneous results in the output generated after running these algorithms. However, there exist no such standardized databases for dynamic event-based vision input. Hence, the matching algorithms cannot be used directly to identify mismatch in results. Manual labeling of certain coordinate locations is required, so that whenever a keypoint coincides with this location, it acquires this label and its descriptors are recorded. While performing matching, only those keypoints with the same labels must be compared and the response recorded. This is a long process which can be pursued as time permits.

CONCLUSIONS & RESULTS

To summarize the following goals have been achieved

- An in-depth background study has been performed on existing feature-based recognition algorithms and the DVS sensor to evaluate the principles behind recognition and how the problem should be addressed from an event-based perspective.

- An event-driven representation which gives some persistence to the event data, and generates output akin to a frame-based imager, has been developed. It is superior to simple frame-based images in the sense that it does not entail simply recreating the scene – updation and changes in the image are effected only by arrival of an event, and can still be used to effectively detect & describe features, thereby decreasing redundancy of data.

- Real-time event-based implementations of 2 classes of detection schemes – convolution-based & binary comparison-based – have been achieved. These can be used as a basis in any future applications or research efforts aimed at addressing classical computer vision problems like object recognition using feature-based schemes for asynchronous event-based sensors.

- Numerous descriptor patterns have been developed and presented, enabled with updation on an event basis. Tests documenting their performance on scenes recorded by the DVS have been recorded and presented.

- The performance of matching of the developed descriptors and the degree of agreement, on 2 individual recordings of the same scene, is presented.

Thus, a complete, albeit rudimentary, event-based feature scheme working with very low latency and consuming extremely low power has been developed. The pixel buffer representation which is adapted to access static scene content does not produce the best reconstruction of the scene and is riddled with noise and other issues. The presented qualitative results show that the descriptor schemes identify corresponding keypoints correctly, however, more than one match on the same key point is quite liable to be obtained as there is no measure for false positives.

OUTLOOK

As is observed in the output, the approach adopted to reconstruct the observed scene does not generate a very good representation and is inundated with noise and other issues arising from the characteristic structure of the DVS pixel. Hence, it is proposed to replace this processing & reconstruction step with the frame content output of the newly developed APS-DVS sensor. This output shows a much more sophisticated quality of resolution and reconstruction, which will result in better detection using the developed methods

In this effort, only a few patterns for binary comparison-based feature descriptors have been implemented and analyzed. For future efforts in this direction, testing various patterns and subjecting them to mathematical scrutiny, presents a promising possibility.

Furthermore, as one of the first efforts in this direction, the focus has been directed towards building a solid foundation for the event-based detection & description schemes and have not forayed into handling scene changes like affine transforms, rotation, scale change etc. Hence, one of the next sure stages for development in this domain (after having identified suitable descriptors), is incorporating scale & rotation invariance capabilities in the detection & description schemes. As a strong foundation for standalone detection & description schemes has been developed, the extension to scale & rotation invariance should follow development similar to the frame-based approaches i.e. orienting the local gradients with respect to a global orientation to get rotation invariance & successive down-sampling to gain scale invariance.

Another important development that can be targeted towards achieving robust algorithms using the DVS, is the development of substantial labeled databases as is required for quantifying the performance of any recognition algorithm. Efforts are already being made to obtain quantitative results for a single labeled dynamic scene.
A possible approach to generate such standardized databases is to use appropriate replications (non-reflective, properly scaled etc.) of the standard still images from the existing databases, and move them in circular patterns in front of the retina so as to prevent any directional biases in detection to label points..

Due to the inherent low-latency of the vision sensor being used, resultant low redundancy of input data, necessity based computations (enabling real-time performance in complex environments), the DVS satisfies the corresponding imposed constraints of the framework for performing dynamic computer & machine vision. Hence, implementing the aforementioned improvements in the basic algorithms and getting them at par with their frame-based counterparts, could effect a paradigm shift in autonomous vision sensing from frame-based imagers to the usage of event-based vision sensors.

BIBLIOGRAPHY/REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] E. Grimson and T. Lozano-Perez, "Localizing overlapping parts by searching the interpretation tree," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 9, pp. 469–482, 1987.

[3] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional image," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987.

[4] R. C. Nelson and A. Selinger, "Large-scale tests of a keyed, appearance-based 3-d object recognition system," *Vision Research*, vol. 38, no. 15, pp. 2469–2488, 1998.

[5] R. Basri and D. Jacobs, "Recognition using region correspondences," *International Journal of Computer Vision*, vol. 25, no. 2, pp. 141–162, 1996.

[6] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-d objects from appearance," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[7] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[8] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics*, vol. 21, no. 2, pp. 224–270, 1994.

[9] K. Mikolajczyk, *Detection of local features invariant to affine transformations*. PhD thesis, Institut National Polytechnique deGrenoble, France, 2002.

[10] J. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Conference on Computer Vision and Pattern Recognition*, (Puerto Rico), pp. 1000–1006, 1997.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[12] P. Simard, L. Bottou, P. Haffner, and Y. Lecun, "Boxlets: a fast convolution algorithm for neural networks and signal processing," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 11, MIT Press, 1999.

[13] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Lecture Notes in Computer Sciences*, vol. 3951, pp. 404–417, 2006.

[14] T. Lindeberg, "Scale-space theory in computer vision," in *Springer International Series in Engineering and Computer Science*, vol. 256, Springer, 1994.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[16] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of Alvey Vision Conference*, pp. 147–151, 1988.

[17] S. Smith and J. Brady, "Susan - a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, pp. 45–78, 1997.

[18] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1508–1511, October 2005. Oral presentation.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006. Poster presentation.

[20] C. S. T. Tuytelaars, "Vector quantizing feature space with a regular lattice," in *International Conference on Computer Vision*, 2007.
[21] M. B. S. Winder, G. Hua, "Picking the best daisy," in *Conference on Computer Vision and Pattern Recognition*, 2009.
[22] K. K. J. B. P. M. P. F. M. Calonder, V. Lepetit, "Compact signatures for high-speed interest point description and matching," in *International Conference on Computer Vision*, 2009.
[23] G. Shakhnarovich, *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.
[24] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615–1630, 2004.
[25] S. W. G. Hua, M. Brown, "Discriminant embedding for local image descriptors," in *International Conference on Computer Vision*, 2007.
[26] C. S. a. P. F. M. Calonder, V. Lepetit, "Brief: Binary robust independent features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.
[27] V. L. P. F. M. Ozusyal, M. Calonder, "Fast keypoint recognition using random ferns," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 448–461, 2010.
[28] P. F. V. Lepetit, "Keypoint recognition using randomized trees," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1465–1479, 2006.
[29] R. S. S. Leutenegger, M. Chli, "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011.
[30] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wise baseline stereo," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, pp. 815–830, 2010.
[31] P. V. Alexandre Alahi, Raphael Ortiz, "Freak: Fast retina keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
[32] M. Hogan and J. J. Wedell, *Histology of the human eye: An atlas and textbook*. 1971.
[33] D. B. M. S. E. Mair, G. D. Hagger, "Adaptive and generic corner detection based on the accelerates segment test," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.
[34] K. K. E. Rublee, V. Rabaud and G. Bradski, "Orb: An efficient alternative to sift & surf," in *Proceedings of International Conference on Computer Vision*, 2011.
[35] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 db 15 us latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 566–576, Feb 2008.
[36] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 db 30mw asynchronous vision sensor that responds to relative intensity change," in *IEEE ISSCC 2006 Dig. Tech. Papers*, pp. 508–509, 2006.
[37] E. C. G. C. U. Mallik, M. Clapp and R. Etienne-Cummings, "Temporal change threshold detection imager," in *IEEE ISSCC 2005 Dig. Tech. Papers*, pp. 362–363, 2005.
[38] E. Culurciello and R. Etienne-Cummings, "Second generation of high dynamic range, arbitrated digital imager," in *Proc. ISCAS*, vol. 4, (Vancouver, BC, Canada), pp. 828–831, May 2004.
[39] P. F. Ruedi, P. Heim, F. Kaess, E. Grenet, F. Heitger, P. Y. Burgi, S. Gyger, and P. Nussbaum, "A 128x128 pixel 120-db dynamic-range vision-sensor chip for image contrast and orientation extraction," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 2325–2333, December 2003.

[40] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *Proc. IEEE ISCAS 2007*, (New Orleans, LA), pp. 845–849, May 2007.
[41] M. Litzenberger, B. Kohn, A. N. Belbachir, N. Donath, G. Gritsch, H. Garn, C. Posch, and S. Schraml, "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *Proc. 2006 IEEE Intelligent Transportation Systems Conf. (ITSC'06)*, (Toronto, ON, Canada), pp. 653–658, September 2006.
[42] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schön, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Proc. 12th Digital Signal Processing Workshop, 4th Signal Processing Education Workshop*, (Grand Teton National Park, WY), pp. 173–178, September 2006.
[43] O. Braddick, "Neural basis of visual perception," *Elsevier, 2001*, pp. 16269–16274, 2001.
[44] E. Chicca, P. Lichtsteiner, T. Delbruck, G. Indiveri, and R. J. Douglas, "Modeling orientation selectivity using a neuromorphic multi-chip system," in *Proc. IEEE Int. Symp. Circuits Syst.*, (Island of Kos, Greece), pp. 1235–1238, September 2006.
[45] T. Y. W. Choi, P. Merolla, J. Arthur, K. Boahen, and B. E. Shi, "Neuromorphic implementation of orientation hypercolumns," *IEEE Transactions on Circuits and Systems*, vol. 52, pp. 1049–1060, June 2005.
[46] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas, T. Delbruck, S. Liu, R. Douglas, P. Hafliger, G. Moreno, and A. Civit, "Caviar: A 45 k neuron, 5 m synapse, 12 g connects/s aer hardware sensory-processing-learning-actuating system for highspeed visual object recognition and tracking," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1417–1438, September 2009.
[47] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Transactions on Circuits and Systems*, vol. 53, pp. 2548–2566, December 2006.
[48] C. Mead, *Analog VLSI Implementation of Neural Systems*, ch. "Adaptive Retina", p. 239–246. C. Mead and M. Ismail, Eds. Boston, MA: Kluwer Academic Publishers, 1989.
[49] M. A. Mahowald and C. Mead, "The silicon retina," *Scientific American*, vol. 264, pp. 76–82, May 1991.
[50] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge University Press, 2003.
[51] O. D. Faugeras, "What can be seen in three dimensions with an uncalibrated stereo rig?," in *Proc. Eur. Conf. Comput. Vis.*, pp. 563–578, 1992.
[52] R. Hartley, R. Gupta, and T. Chang, "Stereo from uncalibrated cameras," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, (Champaign, IL), pp. 761–764, June 1992.
[53] R. Benosman, S. Ieng, P. Rogister, and C. Posch, "Asynchronous event-based epipolar geometry," *IEEE Transactions On Neural Networks*, vol. 22, November 2011.
[54] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, September 1981.
[55] P. Rogister, R. Benosman, S. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Transactions On Neural Networks and Learning Systems*, vol. 23, February 2012.

APPENDIX A

Project Codebase Folder: ch.unizh.ini.projects.eventbasedfeatures

Classes:

A] Framework

This set of classes defines super sets of actual implementation of kernels, detectors and descriptors to provide a neat hierarchical structure to each implementation.

1. **Pixel Buffer:** This class contains the implementation of the pixel buffer approach discussed. It encapsulates access to the Kernel Implementor, Convolution Feature Scheme and Binary Feature Detector so as to enable access to the methods defined for the detector & descriptor of choice.

2. **Feature Method:** This class includes classes and methods that are common to the implemented detectors and descriptors e.g. the KeyPoint class which details all the required components to identify a key point.

3. **Kernel Implementor:** This class provides a platform to implement event-based convolutions with various kernels viz. Gaussian, Laplacian etc. The kernel to be implemented has to be simply listed in the initialization function of the filter.

4. **Convolution Feature Scheme:** This class forms the basis of the event-driven convolution kernel-based feature detectors. It lists all available detectors and descriptors, and provides a choice to select the desired detector and descriptor. It includes constructs to all the listed detectors and descriptors, which are called upon initialization. A filter chain included in its construct connects its function to the pixel buffer thereby ensuring that the pixel buffer can access the processes detailed therein and process events accordingly. The detectors and descriptors are listed in its initialization function.

5. **Convolution Kernel Method:** This class includes all the methods to be implemented, functions and base constructs for a convolution kernel. Any implemented kernel for event-based convolution extends this class.

6. **Binary Feature Detector:** This class forms the basis of the event-driven binary comparison-based feature detectors. It satisfies a similar functionality as the convolution feature scheme. Similarly, a filter chain included in its construct connects its function to the pixel buffer thereby ensuring that the pixel buffer can access the processes detailed therein and process events accordingly.

7. **Binary Scheme:** This class includes all the methods to be implemented, functions and base constructs for an event-based binary detector. Any implemented kernel for event-based convolution extends this class.

8. **Descriptor Scheme:** This class contains all the classes and methods that are common to or need to be implemented in every descriptor pattern defined.

B] Kernels

1. Gaussian Blur Kernel: This class implements a discretized 5x5 Gaussian kernel. Convolutions performed with the same are depicted in a separate image frame – a structure handled by the Image Display class. The values used in this kernel are verified from: http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

C] Detectors

This set of classes implements event-based feature detection schemes.

1. **Laplacian of Gaussian Kernel:** This class implements event-based convolution with a discrete Laplacian of Gaussian (LoG) kernel, inspired by the SIFT algorithm and explained in the documentation of convolution kernel-based event-driven feature detectors. The discrete LoG kernel can be verified here - http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

2. **FAST:** This class implements event-based binary comparison detection inspired by the FAST algorithm, as discussed in the documentation of binary comparison-based event-driven feature detectors.

D] Descriptors

This set of classes develops the complete set of descriptor patterns defined by similar names and as depicted in Figure 36.

1. **CROSS**
2. **CROSSTwo**
3. **SQUARE**
4. **SQUARETwo**
5. **SQUARECorner**
6. **SQUARECornerTwo**
7. **OCTAGON**
8. **BCIRCLE**