Statistical approach to time-to-impact estimation suitable for real-time near-sensor implementation

Anders Åström[a,*] **and Robert Forchheimer**[b]
[a]Swedish National Forensic Center, Linköping, Sweden
[b]Linköping University, Department of Electrical Engineering, Linköping, Sweden

Abstract. We present a method to estimate the time-to-impact (TTI) from a sequence of images. The method is based on detecting and tracking local extremal points. Their endurance within and between pixels is measured, accumulated, and used to achieve the TTI. This method, which improves on an earlier proposal, is entirely different from the ordinary optical flow technique and allows for fast and low-complex processing. The method is inspired by insects, which have some TTI capability without the possibility to compute high-complex optical flow. The method is further suitable for near-sensor image processing architectures. © *The Authors. Published by SPIE under a Creative Commons Attribution 4.0 International License. Distribution or reproduction of this work in whole or in part requires full attribution of the original publication, including its DOI.* [DOI: 10.1117/1.JEI.31.6.063023]

Keywords: high-dynamic range image processing; near-sensor image processing; real-time image processing; time-to-impact.

Paper 220495G received May 15, 2022; accepted for publication Oct. 31, 2022; published online Nov. 18, 2022.

1 Background

There are numerous applications for a system that can estimate time-to-impact (TTI) from a sequence of images generated by a video camera. The applications range from vehicle collision warning sensors to robotics and safety systems in industry. TTI estimation is a special case of general motion estimation. The aim is to estimate when a possible collision may occur between the camera and an object seen by the camera.

The image processing needed to perform real-time TTI estimation usually requires a fair amount of hardware and processing resources. Spatial motion within the image is typically based on estimating the optical flow.[1] To do this in real time requires fast computing hardware and data storage that can hold one or more frames. The camera itself needs to produce low-noise images of good quality.[2] If used in outdoor applications, the dynamic range of the camera needs to be high.[3] Inspired by the vision system of insects our ambition has been to find a simpler and faster method for TTI estimation. One characteristic of insect vision is the use of extremely localized spatial and temporal contrasts.[4]

In two previous papers, we presented a method based on estimating the "inverse" of the motion (how long an image feature stays at the same pixel position).[5,6] This approach drastically reduces the computational load. We also showed that the method lends itself naturally to a smart sensor architecture denoted near-sensor image processing (NSIP).[7]

In this paper, we extend our search for a method which, like our previous proposal, lends itself to a low-complex hardware implementation. The two methods detect LEPs in the same way. However, they use completely different methods to compute TTI. The presented method is much more stable in the presence of noise, particularly in the 2D case. The main reason for this is that the previous method required runs of 10 to 20 consecutive uninterrupted LEPs, which made it highly noise sensitive. In contrast, the current method requires no more than two frames, which, besides being less sensitive to noise, also allows for a faster decision.

Like in our previous papers, we investigate the case where the relative motion is "head-on" meaning that the motion of the camera is co-linear with the line-of-sight. Furthermore, the object

*Address all correspondence to Anders Åström, anders.astrom@polisen.se is assumed to fill up the entire field-of-view and has a depth that is essentially constant over the image.

In Sec. 3, we give a short background to TTI estimation and present our alternate way to compute optical flow and to estimate TTI for the 1D case. Sections 4 and 5 extend the method to 2D and give simulation and experimental results. In Sec. 6, we discuss implementation issues based on dedicated architectures and performance estimates. Section 7 finally summarizes the results and discusses conclusions that can be drawn from them.

2 TTI Estimation, 1D

It is well-known that TTI estimation from a camera input does not require absolute measurements of velocity or distance. Instead, motion in the image plane, such as the optical flow,[8] is sufficient data for the purpose. An object that contains contrast variations will generate an image sequence where such image motions (displacements) expand from a central point, denoted focus-of-expansion (FOE). The rate of expansion is small close to this point and increases further out in the image. Assuming a pinhole camera model, the displacement at an image point will be proportional to the distance between this point and FOE. The proportionality factor k will change as the camera moves closer to the object. However, it is sufficient to get just one measurement of k in order to estimate TTI provided that the relative velocity between the camera and the object stays constant. As shown earlier, TTI is inversely proportional to k.[5] Naturally, with several estimates at different points in time, the accuracy can be improved and will also allow to track nonconstant camera and/or object velocity.

We next describe our proposed method to estimate k.

2.1 *Statistical Approach to Estimate k*

Our earlier approach for the TTI application utilized two main principles, namely (i) that motion vectors are only computed at specific feature points which we denote local extremal points (LEPs) and (ii) that time, rather than displacement, is measured. Based on how long of a time period, a feature point stays within one pixel position we obtained the optical flow data needed for our purpose by taking the inverse of this time interval.

The algorithm is based on the same principle as the previous algorithm, to use the stability of LEPs in the image to compute TTI. However, the previous algorithm required a large number of frames before producing a result. The algorithm, based on statistics, can achieve an answer from just two consecutive frames similar to the case for a traditional optical-flow method.

We will describe the method by starting with the 1D case. We assume that there are N pixels altogether along a line, that the frame rate is high so that consecutive frames are available at approximately the same TTI measurement instance (same k) and that the FOE is located at the center of the line. We further index the pixels with $i = 0,1,2,\ldots,N/2$ in both directions starting from the FOE. Displacements in the image plane due to the motion is denoted $D(i)$ and is expressed in pixels with reference to the previous frame.

By $f(i)$, we denote the existence of an LEP at location i at the time of measurement, i.e., $f(i) = 1$ if there is a LEP, otherwise $f(i) = 0$. Likewise, $f'(i)$ denotes the existence of a new LEP [meaning that $f(i) = 0$ in the previous frame]. The frame rate is assumed to be high enough that all new LEPs are captured. This requires that the maximum displacement D appearing at $i = N/2$ is less than the interpixel distance $D_{N/2} < 1$.

However, as we will see later, the method fails gracefully for larger displacements.

Given that an LEP at position i moves at a speed of D_i pixels/frame, this LEP will, on the average, stay at the same pixel location for a duration of $1/D_i$ frames which we denote as the LEP run length. It will then have moved to a neighboring pixel. In our previous method, we measured the run lengths of the LEPs that required to collect many frames, particularly to get correct values for LEPs close to the FOE.

However, as there is a close relationship between the appearance of a new LEP and the ending of a run length, it is possible to estimate k from this relationship in a statistical sense.

Based on the (optical flow) relation $D_i = k \cdot i$ and the fact that a new LEP appears whenever the total displacement reaches the distance of one pixel (after some number of frames), a proportion equal to D_i of the LEPs will generate new LEPs at the next frame. In our 1D setup, there can be up to two LEPs located at the distance i from FOE. However, in 2D, there can be several of them. Alternatively, we can view the proportion as a probability that a new LEP will appear at a neighbor location given a LEP at location i. It should be observed that, due to the definition of LEPs, two of them cannot exist at neighboring locations.

The expected number of new LEPs at the next frame can thus be written as

$$\sum_i f'(i) \approx \sum_{\mathrm{i}} f(i) \cdot D_i = \sum_i f(i) \cdot k \cdot i, \tag{1}$$

where the summation is taken over the whole image line (both instances of i).

From this, we see that

$$k \approx \frac{\sum_i f'(i)}{\sum_i \mathrm{if}(i)}. \tag{2}$$

If the LEPs are sufficiently spread out across the image, which is the case for typical natural images, we can approximate (see Appendix):

$$\sum_i \mathrm{if}(i) \approx \frac{\sum_i f(i)}{N} \sum_i i. \tag{3}$$

This will lead to the following estimate of k:

$$k \approx \frac{N \sum_i f'(i)}{\sum_i i * \sum_i f(i)} = \text{constant} \cdot \frac{\sum_i f'(i)}{\sum_i f(i)}. \tag{4}$$

With FOE located at the midpoint of the sensor, it can be shown that constant $\approx 4/N$.

2.2 *Finding Local Extremal Points*

LEPs are defined as local maximum or minimum points in the image. In 1D, we compare each pixel with its two neighbors. If the pixel has a higher/lower value than its neighbors, it will be considered a local maximum/minimum. Although susceptible to noise, the LEP normally needs to be visible in at least two successive frames to allow for the computation of its displacement. Our proposed algorithm is statistical and a few erroneous LEPs will not distort the TTI estimation severely. Since the LEPs are based on the relative intensities between neighboring pixels, they can be extracted nicely by cameras that are able to form such information dynamically, for instance, NSIP cameras.[9,10]

2.3 *Estimating* k *and TTI*

The basis of the algorithm is thus the following:

1. find all LEPs in the image, $f(i) = \begin{cases} 1, & \text{if LEP}, \\ 0, & \text{if not LEP}, \end{cases}$
2. mark all new LEPs, $f'(i) = \begin{cases} 1, & \text{if LEP and same pixel in previous image not a LEP} \\ 0, & \text{otherwise}, \end{cases}$
3. count all LEPs, $\sum f(i)$,
4. count all new LEPs, $\sum f'(i)$.

The TTI can now be computed as

$$\mathrm{TTI} = \frac{T_d}{k}, \tag{5}$$

where T_d is the time between two frames and k is given by Eq. (4).

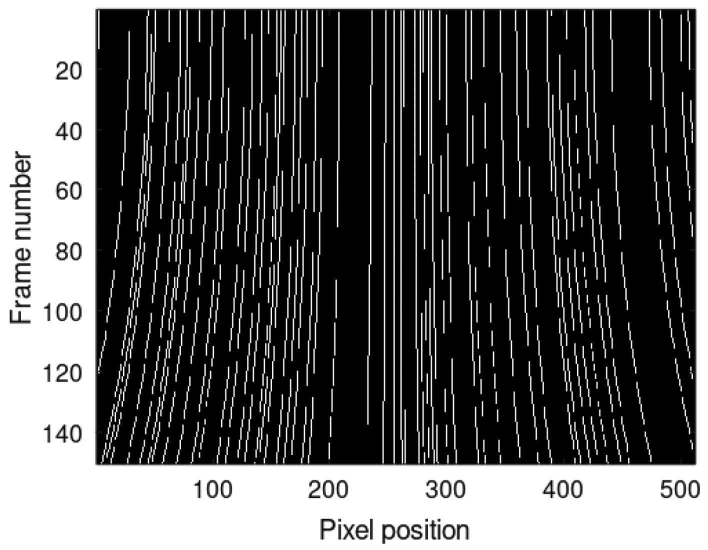

Fig. 1 Collected LEPs (white pixels) from a sequence of 150 frames. The FOE is located at the center position.

2.4 *Simulation*

We simulate the algorithm using a 1D image of size $N = 512$. We zoomed into the image in 150 steps. For the first 15 steps, the zoom factor was 0.1 pixels per frame (for the outermost pixel). For the next 15 steps, the zoom factor was 0.2 and so on. For each step, we compute all LEPs. Plotting the LEP-images for all 150 1D frames results in the image shown in Fig. 1. It is seen that the LEP positions are quite stable in the beginning (top of the image) while they change their positions at the end (bottom of the image).

Computing the sums of $f(i)$ and $f'(i)$ as a time sequence gives the data shown in Fig. 2. The blue curve corresponds to $f(i)$ and the red curve to $f'(i)$.

Defining, for convenience, $R = k \cdot N$ which we denote the "zoom factor," it is seen that R corresponds to twice the maximum displacement. Applying Eq. (4) and multiplying by N, we thus get an estimate of the applied zoom factor. Figure 3 shows, for each frame, the computed value of R (in green) and the corresponding true value in blue. The black dots represent the average of the 15 steps with the same step length. Here we can see that the more data we have

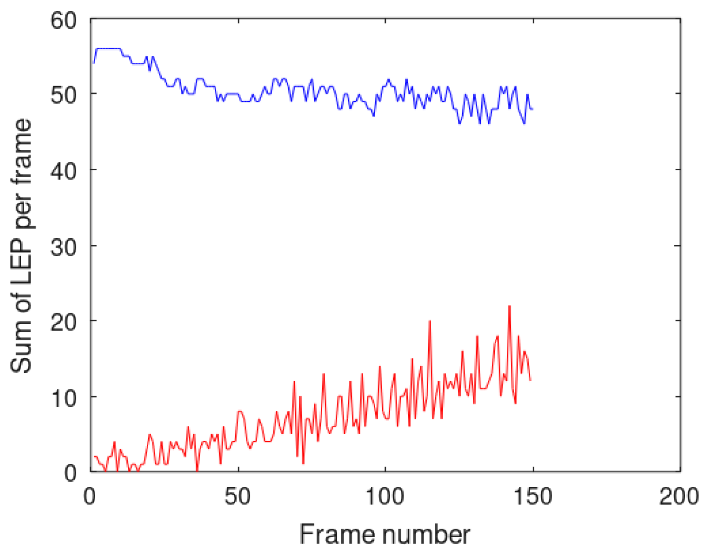

Fig. 2 Plots of the number of LEPs (blue) and new LEPs (red) as function of frame index.

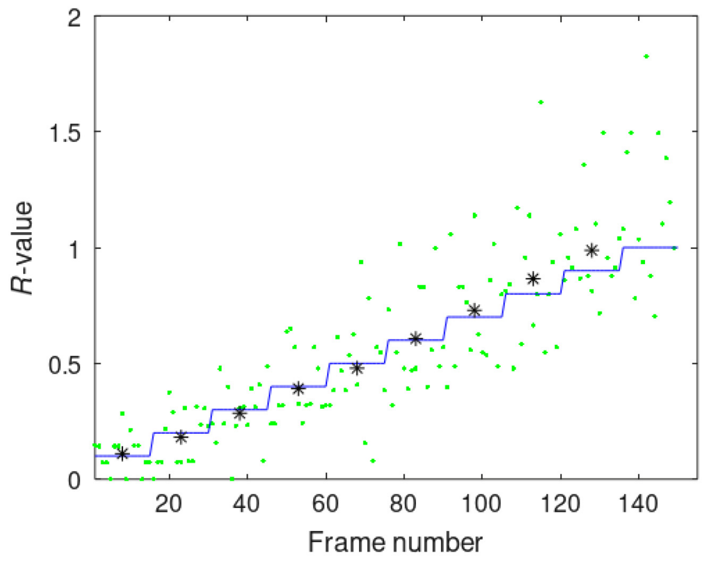

Fig. 3 Plots of estimated zoom factor (green dots) and ground truth (blue line). Black dots correspond to the average of the estimates within each zoom value.

(green points) the better approximation of $R(k)$ we get. In Eq. (4), this corresponds to forming an average value of $f'(i)$, which gives an average value of k since $f(i)$ is fairly stable, as shown in Fig. 2.

2.5 *Graceful Degradation*

From Fig. 3, it is seen that as the zoom factor increases, the algorithm will start to deviate from the correct result. The reason is that the method requires that all displacements are below 1 pixel. However, the problem is not that severe. As is seen in Fig. 4, for zoom factors up to 3 (maximum displacements up to 1.5 pixels), the error will be relatively small. For larger zoom factors, the error will increase sharply. However, it is possible to detect when this is the case. In Fig. 4, the green dots represent sample R values, the black dots represent the average within each group of 15 frames, and the blue line represents the actual zoom factor at each frame.

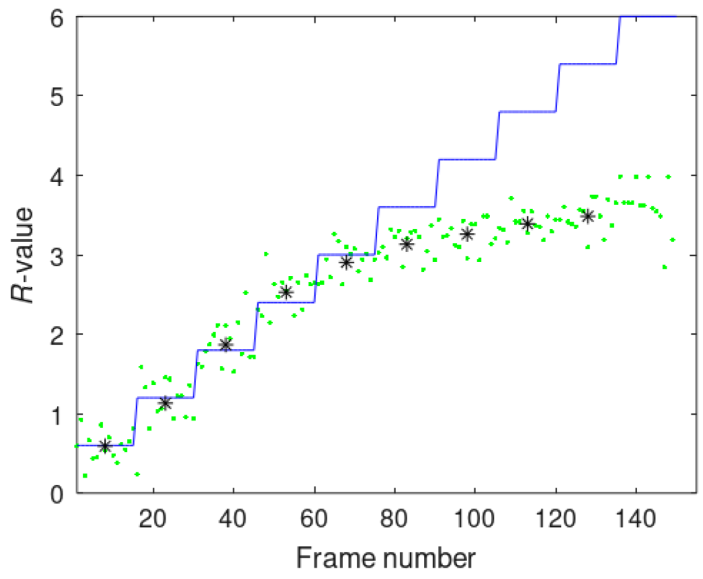

Fig. 4 Similar to Fig. 3 but for large zoom factors.

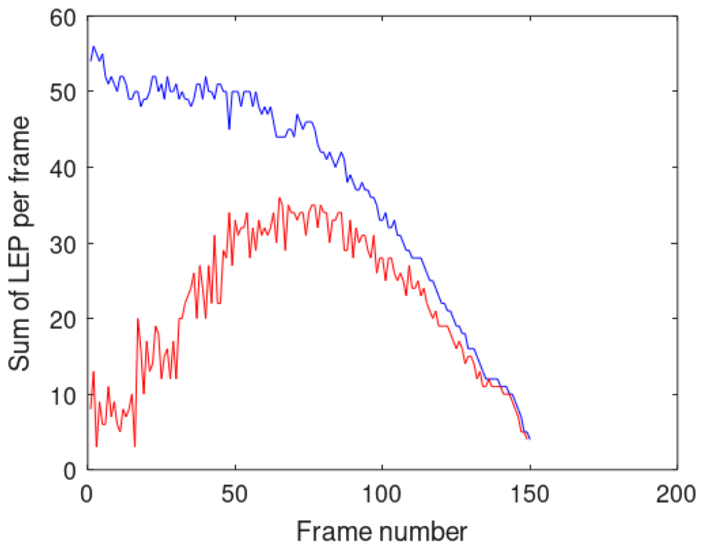

Fig. 5 Similar to Fig. 2 but for large zoom factors.

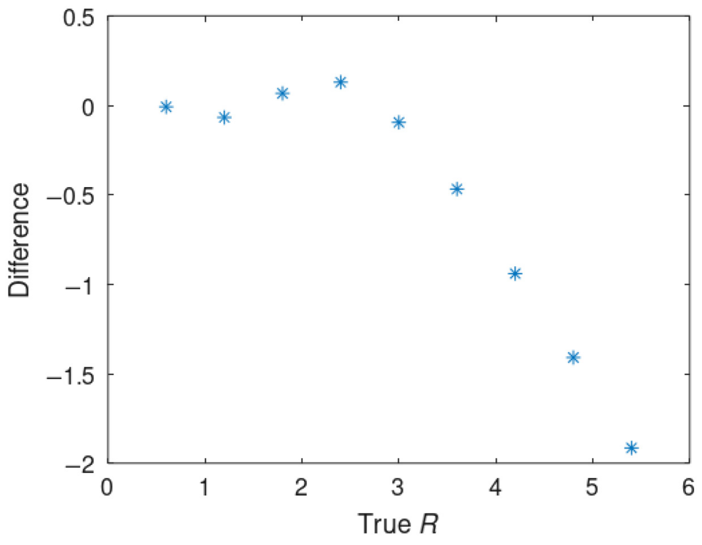

Fig. 6 Difference between true and estimated zoom factor.

It is seen that that the R values are slightly below the correct values when between 2 and 3. Above 3, the R values will level out.

The reason that R levels out is that $f'(i)$ will approach $f(i)$, as shown in Fig. 5, limiting the R value. This will be the case when essentially all LEPs are new LEPs.

The difference between the true zoom factor and the estimated value are plotted against the true zoom factor in Fig. 6. When >3, the estimate will be lower than the true value. Since we can estimate R, we know when it is out of range. The corresponding impact time is then shorter than the predicted time.

2.6 *Dependence on the Number of LEPs*

The algorithm is relatively insensitive to the number of LEPs. If we take the same 1D image and perform low-pass filtering before zooming, we get the LEPS shown in Fig. 7.

The corresponding graphs of the sums of $f(i)$ and $f'(i)$ are shown in Fig. 8. When compared with Fig. 2, it is seen that the number of LEPs has been reduced by a factor of 2.5.

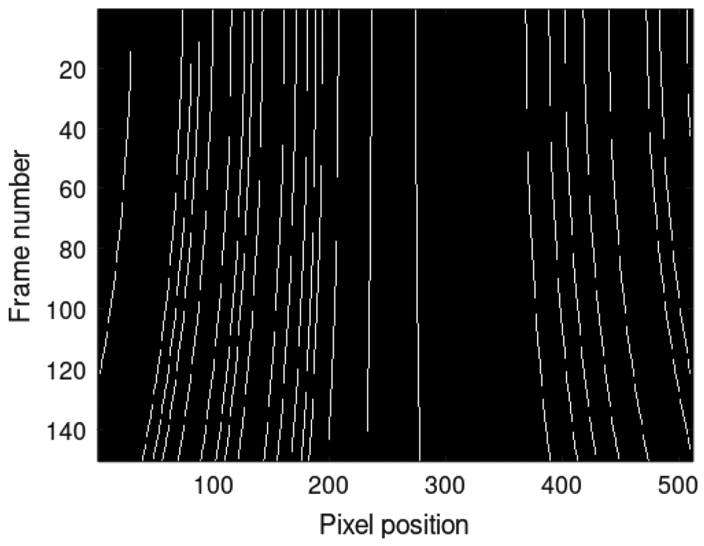

Fig. 7 LEPs generated from the low-pass filtered image.

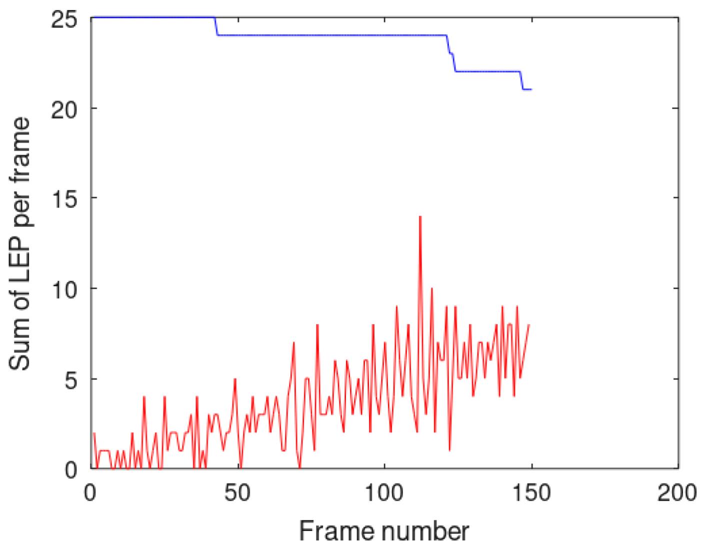

Fig. 8 Sums of LEPs (blue) and new LEPs (read) for low-pass filtered image showing reduced number of LEPs.

The R values are noisier. However, the average value is stable as is seen in Fig. 9.

2.7 *Limitations*

If the distance to the object is d and the field of view is N, the zoom factor between two frames is given by

$$\frac{R}{N} = \frac{\Delta d}{d}. \tag{6}$$

The movement toward the camera between two frames differing in time by $\underline{\Delta} t$ given the velocity v is

$$\Delta d = v\Delta t. \tag{7}$$

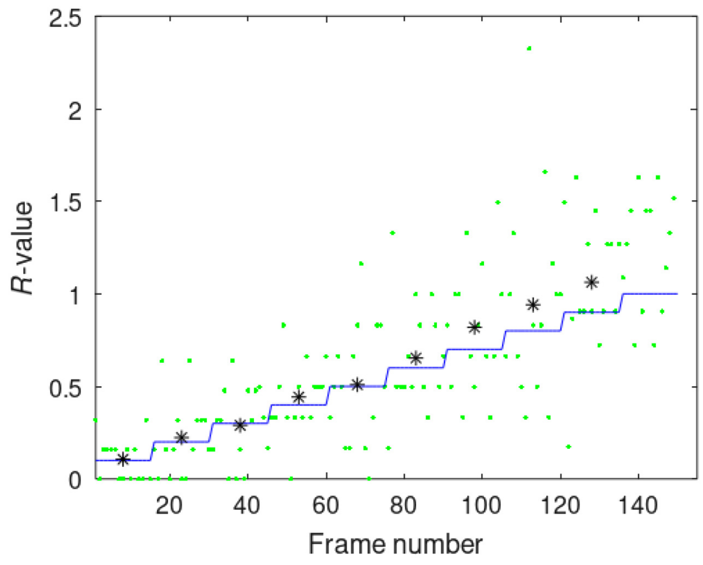

Fig. 9 Estimated values of zoom factor *R* for low-pass filtered frames.

The zoom factor must be smaller than some constant C. We have shown previously that C can be around 3:

$$R = \frac{\Delta d}{d}\frac{N}{2} = \frac{v\ \Delta t\ N}{2\ d} < C. \tag{8}$$

This means that the velocity v must be smaller than

$$v < \frac{2\ C\ d}{N\Delta\ t}. \tag{9}$$

3 Extension to 2D

3.1 *Estimating* k

The estimation of k in the 2D case follows the same principle as in the 1D case. The proportion of LEPs that will generate new LEPs in the next frame depends on the distance from the FOE. The displacements in off-angled directions (not along a horizontal or vertical line through the FOE) are larger than their horizontal or vertical components, but so are also the distances that the LEPs need to move before reaching a new pixel and thus generate new LEPs. For this reason, it is necessary to compensate for those larger distances between the pixels. Thus a LEP located at position (i, j) with respect to the FOE is subjected to a displacement of

$$D_{ij} = k\sqrt{i^2 + j^2}, \tag{10}$$

but the step required to give rise to a new LEP at the next frame is simultaneously increased due to the longer distance between off-angled neighboring pixels. For example, this extended distance is a factor of $\sqrt{2}$ of the interpixel distance for diagonally located pixels and in general depends on the angle given by $\tan(j/i)$.

This function is not trivial as it depends on the pixel shape and fill factor of the specific sensor in use. For the moment, we will simply assume the existence of such a function $s(i, j)$. In the following, the 2D image is assumed to have a size of $N \times N$ pixels.

The expected number of new LEPs is then given by

$$\sum_{i,j} f'(i,j) \approx \sum_{\mathrm{i,j}} f(i,j)\frac{D_{ij}}{s(i,j)} = \sum_{i,j} f(i,j) \cdot k\sqrt{i^2+j^2}/s(i,j). \tag{11}$$

Thus

$$k = \frac{\sum_{i,j} f'(i,j)}{\sum_{i,j} f(i,j)\sqrt{i^2+j^2}/s(i,j)}. \tag{12}$$

Similar to the 1D case, we will use the approximation

$$\sum_{i,j} f(i,j)\sqrt{i^2+j^2}/s(i,j) \approx \frac{1}{N^2}\sum_{i,j}\frac{\sqrt{i^2+j^2}}{s(i,j)}\sum_{i,j} f(i,j). \tag{13}$$

Thus for the 2D case, we have

$$k \approx \mathrm{const} \cdot \frac{\sum_{i,j} f'(i,j)}{\sum_{i,j} f(i,j)}, \tag{14}$$

where const $= \frac{N^2}{\sum_{j,j}\frac{\sqrt{i^2+j^2}}{s(i,j)}}$.

Using some further approximations (see Appendix), the following result is achieved:

$$k \approx \frac{3(1+\sqrt{2})\sqrt{\pi}}{4\ N}\frac{\sum f'(x,y)}{\sum f(x,y)}. \tag{15}$$

3.2 *Finding Local Extremal Points in 2D*

In the 2D case, there are several different ways to define an LEP. The two most common ways are based on the chessboard distance and the city-block distance. In the chessboard case, the pixel is compared to its eight neighboring pixels. In the city-block case, there are four neighboring pixels.

The image shown in Fig. 10 was used as the object. Camera motion toward this object is simulated by zooming in on the image.

Figure 11 shows the LEPs in a 2D image based on the chessboard distance. As in the 1D case, an LEP is defined as a pixel which has a higher value than its neighbors.

In the same way as in the 1D case, we define $f'(i,j)$ as the new LEPs appearing in a consecutive frame. The image in Fig. 12 shows $f'(i,j)$ when zooming into the image in Fig. 10.

3.3 *Estimating* k

For the 2D simulation, we generated an image sequence of 15 frames. The first six frames were generated using a zoom factor of 0.2 pixels/row per frame, where each row consists of 512 pixels. The following five frames were generated using a zoom factor of 0.4 pixels/row per frame, and the last four frames were generated using a zoom factor of 0.6 pixels/row per frame. As in the 1D case, the zoom factor corresponds to twice the largest horizontal/vertical displacement in the frames appearing at the four midpoints of the frame borders since the FOE is located at the center of the frames.

Summing up all $f(i,j)$ and $f'(i,j))$ as a function of time, we get the graphs shown in Fig. 13.

If we divide them with each other and apply the constant factor given in Eq. (15), we get the points shown in Fig. 14.

It is seen that we get three clusters with a scale factor of 1, 2, and 3 between them, which matches well with the true zoom factors.

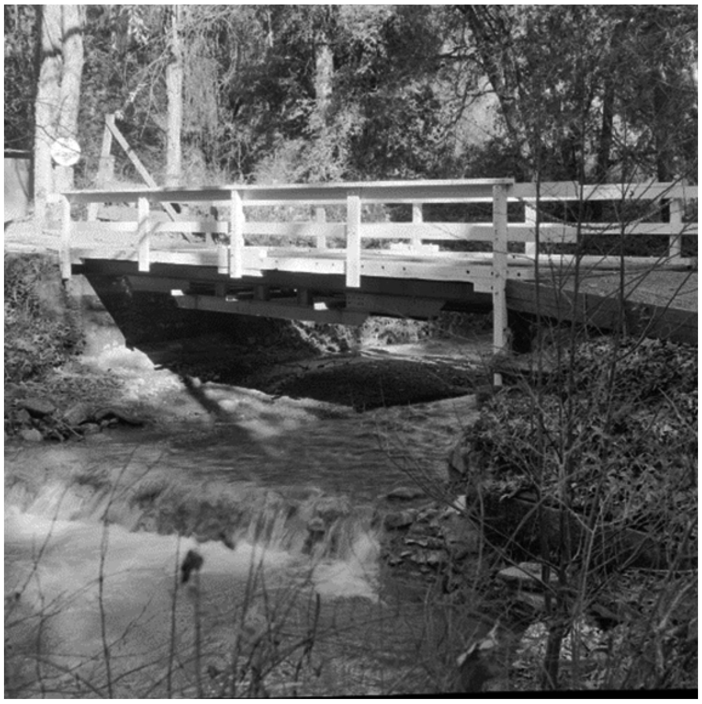

Fig. 10 Test image used for 2D TTI simulation.

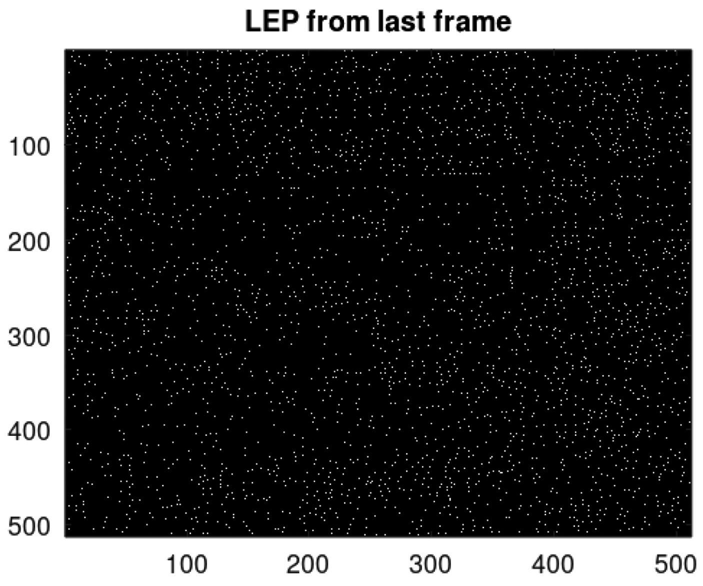

Fig. 11 2D LEPs computed from the image in Fig. 10.

4 Experiments

The experimental setup consists of a moving platform that moves toward the camera in a controlled fashion (see Fig. 15). The platform carries a picture which is the input to the algorithm. The distance between the camera and the image is initially 180 cm to become 171 cm after 50 frames. The camera has a frame rate of 1 frame/s. The speed of the platform is 0.12 cm/s, for the first 25 frames, and then 0.24 cm/s. It should be noted that the setup of this system is many orders of magnitude slower than the expected performance of a dedicated sensor. The reason for this was to simplify the setup using a standard consumer camera. From a principal point of view, this

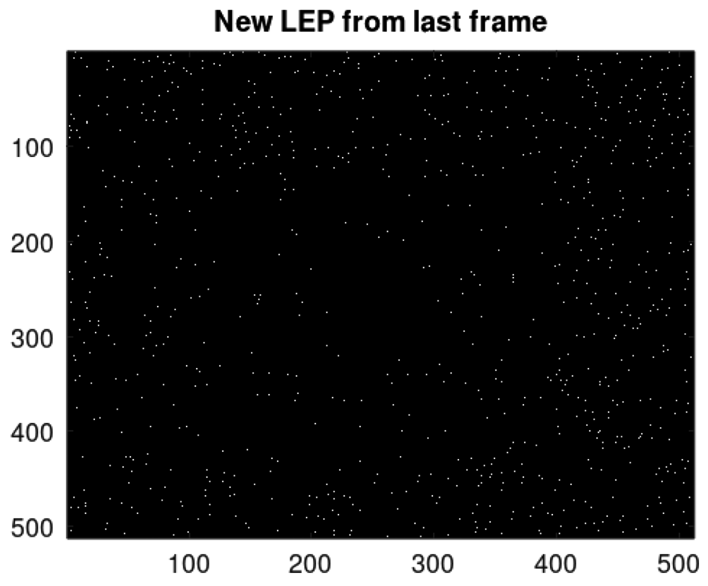

Fig. 12 New LEPs appearing between two consecutive frames when zooming into the test image.

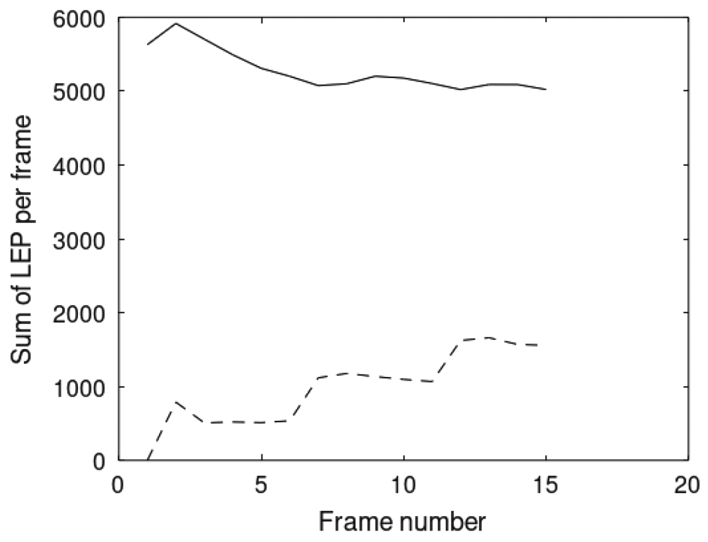

Fig. 13 Sums of 2D LEPs (line) and new LEPs (dashed line).

amounts to merely a temporal scale factor with no other expected influence on the measurements.

From the high-resolution color camera image, we extracted a 512×512 grayscale image, which was at the center of the original image (see Fig. 16).

The TTI for the first 25 samples should be between 1550 and 1475 s depending on if the first frames or last frames are used for the estimation. Likewise, for the last 25 samples, the TTI should be between 738 and 712 s. Figure 17 shows the results from the experiment. The blue line is the actual value for each sample and the red line is the 3×1 average of the blue line. We can see that we have good agreement between the theoretical values and the experimental results.

In Eq. (9), it is seen that v is bounded by the inverse of the resolution, i.e., if we downsample by a factor of 4, we will increase the maximum value of v by a factor of 4.

In Fig. 18, the frame was further downsampled to 128×128 before the algorithm was applied. Here we can see that for the higher speed (frames 26 to 50), we get almost the same results as for the original image. For the first 25 samples, the difference varies from 0% to 100%.

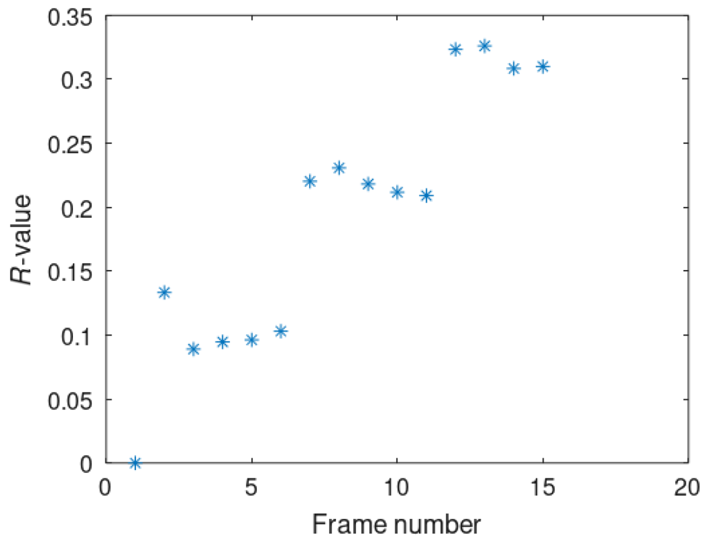

Fig. 14 Estimated zoom factors from the data in Fig. 13.

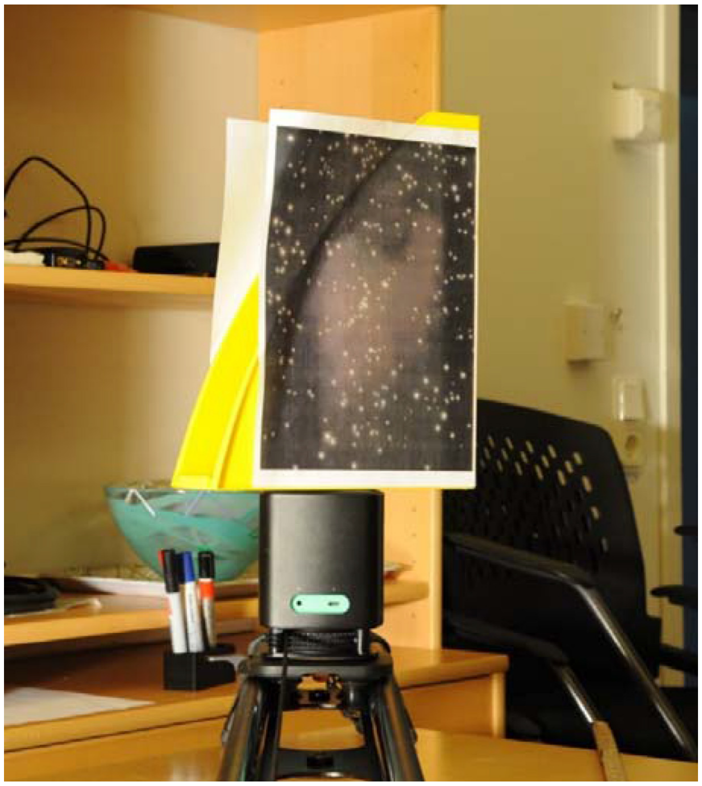

Fig. 15 Moving platform used in the experimental setup.

The reason for the difference between the two resolutions can be found in Fig. 19 where we have plotted the number of LEPs and the number of new LEPs for each frame. In the first part, there are very few new LEPs per frame which makes it sensitive to noise.

5 Implementation Issues

Our aim has been to develop a method that lends itself to a low-complex implementation, ultimately a single chip solution. In this section, we discuss the basic ingredients of such an

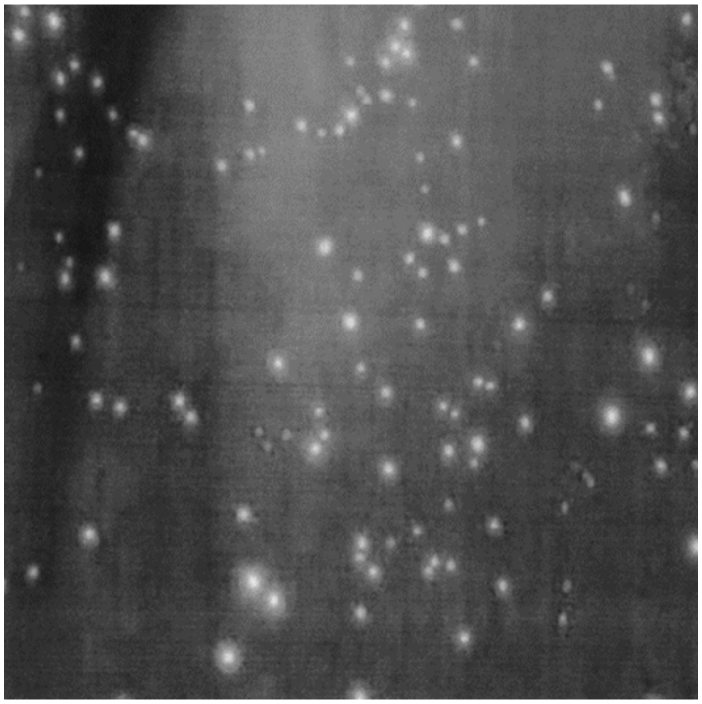

Fig. 16 Grayscale image extracted from the color camera.

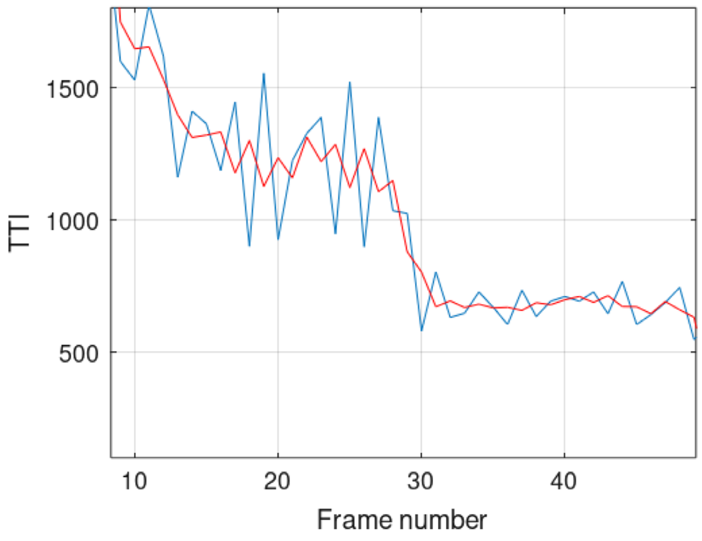

Fig. 17 Estimated TTI from the experimental setup.

implementation. The philosophy is to use the NSIP paradigm. According to this principle, some of the early feature extractions are performed in the analog domain utilizing the behavior of the photosensing circuitry itself.[9]

5.1 *Sensor-Pixel Processor and the NSIP Paradigm*

NSIP is based on a tight integration between the photosensitive element and a low-complex digital pixel processor. A/D conversion is not supported in hardware and is typically not used as part of the required signal processing. Instead, the fact that a photodiode discharges linearly in

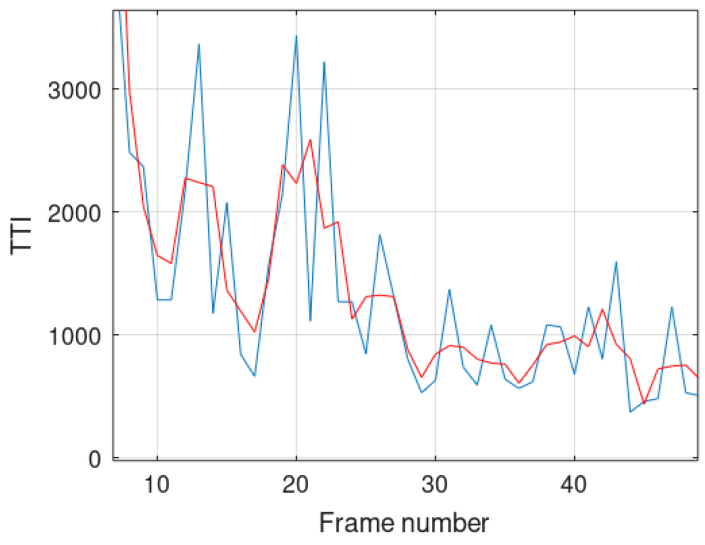

Fig. 18 Estimated TTI for the down-sampled frames (blue) versus full resolution frames (read).

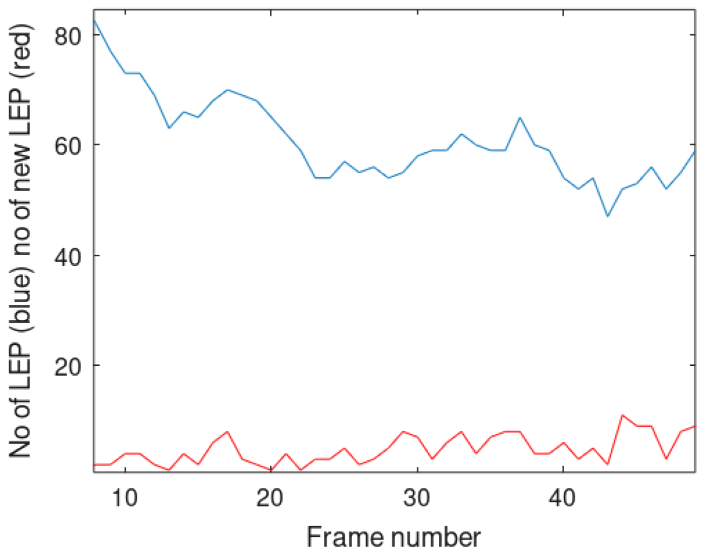

Fig. 19 Number of LEPs (blue) and new LEPs (red) for the down-sampled frames.

time when exposed to light is combined with a comparator/threshold circuit to turn the light intensity into a corresponding timing event as shown in Fig. 20.

The output from the comparator is fed into a digital processor element. To implement the TTI algorithm, this processor consists of a combinatorial circuit, which connects to the eight neighboring comparator outputs and two one-bit memories. The combinatorial circuit detects asynchronously when its own comparator returns a binary zero, whereas the neighboring comparators are still returning binary ones, thus indicating the occurrence of an LEP. One of the bit memories stores this information. The second bit memory indicates if the LEP is new or existed in the previous exposure.

5.2 *Single-Chip System*

Figure 21 outlines the basic layout of a single-chip solution. All photodiodes and PEs on the same row are controlled in parallel thus forming a single instruction multiple data (SIMD) architecture. The comparator outputs as well as the bit memories are forwarded to column buses in a

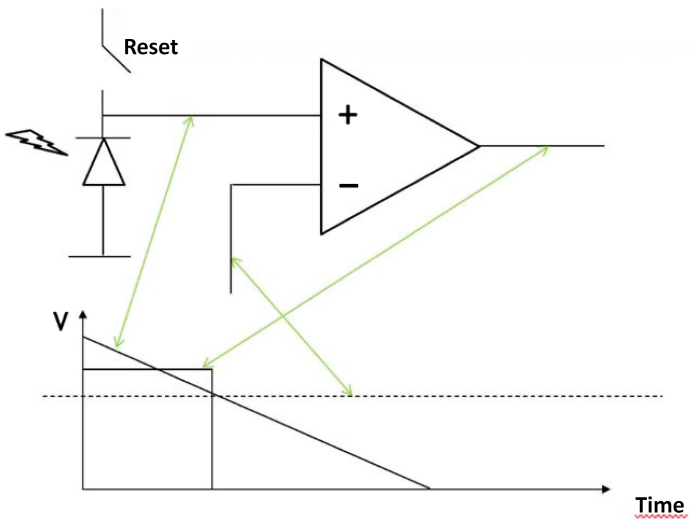

Fig. 20 The photodiode and a comparator turn the incoming light into a binary pulse, the length of which is inversely proportional to the light intensity.

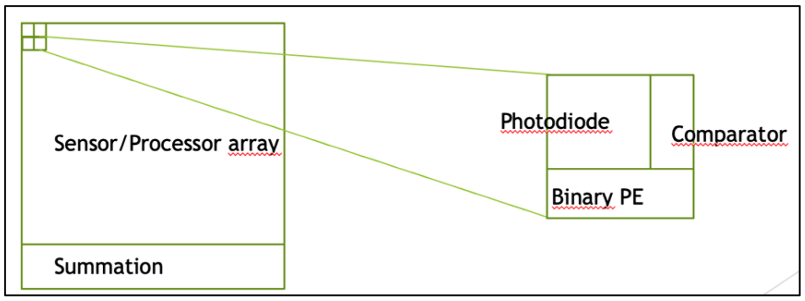

Fig. 21 The layout of a one-chip system showing the photodiode and a processing element being repeated at each pixel position.

row-by-row fashion. A summation circuit at the bottom part of the chip produces the two sums of LEPs and new LEPS by scanning through the rows while adding and accumulating their results.

The size of the chip and the number of transistors is estimated as follows.

Assuming the use of a 100-nm CMOS technology and a sensor size of 256×256, the pixel pitch is estimated to be $14 \times 14\ \mu\text{m}^2$ out of which 50% consists of the photodiode. The summation network is purely combinatorial and consists of cascaded one-bit adders. To make it faster, it can be implemented as a two-layer structure (e.g., eight groups of 32 one-bit adders in the first layer and seven 8-bit adders in the second layer). Based on previous implementations of such networks made by one of the authors, its size is estimated to occupy a depth of 50 μm. Depending on the desired output (separate LEP sums, k value, or TTI), a fix-point multiplier/divider unit may be included as well. Since it only needs to compute one value per frame, it can be sequential in nature, e.g., computing a reciprocal iteratively to perform division through the multiplier unit. Altogether, including pads and some decoding circuits, the chip size is estimated to be $<20\ \text{mm}^2$. Naturally, using a more advanced VLSI technology, the size of the chip would shrink accordingly.

The summation network as well as the available light intensity will define the limiting speed in which the circuit can operate. It is estimated that a two-layer summation network requires $<0.5\ \mu$s. Since available LEPs as well as new LEPs are summed, each row computation will take 1 μs leading to a frame time Δt of 0.25 ms. According to Eq. (9), this corresponds to the following minimum value of the TTI:

$$\mathrm{TTI}_{\min} = \frac{d}{v} > \frac{N\Delta t}{6} \approx 5 \text{ (ms)}.$$

6 Conclusions

A new LEP-based method for estimating TTI has been presented. Like the original method,[5] it captures newly appearing LEPs as well as already available LEPs in a series of images. However, based on a statistical model, estimation of TTI can be done already after very few (down to two) images. LEPs are simply estimated using nearest neighbor comparison. Experiments with larger regions for defining LEPs turned out not to improve the result in the presence of noise. The method is illustrated for 1D and 2D images using simulated and real data. Issues related to a possible single-chip implementation were discussed. Based on the NSIP concept such a chip will utilize analog and digital processing. Due to a low clock frequency and mostly asynchronous circuits, the power consumption is expected to be very low. More detailed knowledge about the performance of such a chip would require simulations to be performed on a specific design. Using several of such sensors together or dividing up the sensor area into several separate areas opens for additional measurements, such as estimating stereo and shape from the TTI values.[6] Possible applications of such TTI-sensors are in the fields of automotive, robotics, automation, and safety, as collision warning sensors, docking sensors, and position sensors.

7 Appendix

7.1 *Justification of Eq. (3)*

Equation (3) reads

$$\sum_i \mathrm{if}(i) \approx \frac{\sum_i f(i)}{N} \sum_i i.$$

Let us assume that $f(i)$ is a binary random vector with probability q that $f(i) = 1$ and probability $(1 - q)$ that $f(i) = 0$.

Taking the expectation of the left-hand side gives:

$$E\left\{\sum_{i=1}^{N} \mathrm{if}(i)\right\} = \sum_{i=1}^{N} E\{\mathrm{if}(i)\} = \sum_{i=1}^{N} iE\{f(i)\} = q\sum_{i=1}^{N} i = \frac{qN}{2}(1+N).$$

Similarly for the right-hand side:

$$E\left\{\frac{\sum_i f(i)}{N}\sum_{i=1}^{N} i\right\} = \frac{1}{2}(1+N)\sum_{i=1}^{N} E\{f(i)\} = \frac{qN}{2}(1+N).$$

The approximation is thus exact in the statistical mean sense.

Naturally, this does not guarantee that the approximation is "good" in specific cases. In our case, we use 1D images of size 512 pixels. Table 1 shows the result of 100 outcomes at three different values of the probability q for this vector size. It shows the correct value (rounded over the 100 trials) and four statistical measures for the approximation error.

Table 1 shows the correct value (rounded over the 100 trials) and four statistical measures for the approximation error. It is seen that the error increases when there are few LEPs and can reach more than 10% in such cases.

Table 1 Approximation errors for different q-values.

	$q = 0.2$	$q = 0.5$	$q = 0.8$
$\sum_i i * f(i)$	17,000	35,000	53,000
Standard deviation	744 (4.4%)	878 (2.5%)	745 (1.4%)
Maximum difference	1787	1999	1632
Minimum difference	−1666	−2298	−2352
Average difference	34	−4.5	−3.5

7.2 *Justification of Eq. (14)*

Let us consider the denominator in Eq. (14):

$$\sum_{i,j} f(i,j)\sqrt{i^2+j^2}/s(i,j).$$

We will break this down into a substantially simpler computation through a series of approximations.

First, as noted in Eq. (13), the two factors within the summation sign are separated:

$$\sum_{i,j} f(i,j)\sqrt{i^2+j^2}/s(i,j) \approx \frac{1}{N^2}\sum_{i,j}\frac{\sqrt{i^2+j^2}}{s(i,j)}\sum_{i,j} f(i,j).$$

Second, we will choose $s(i,j)$ to be a constant function, leaving $\sum_{i,j}\sqrt{i^2+j^2}$ to be computed.

For this purpose, we will approximate the discrete sum by a continuous sum:

$$\sum_{i,j}\sqrt{i^2+j^2} \approx \iint_{-\frac{N}{2},-N/2}^{N/2,N/2}\sqrt{x^2+y^2}\,\mathrm{d}x\mathrm{d}y \approx \iint_{0,0}^{R,2\pi} r^2\mathrm{d}r\mathrm{d}\varphi = \frac{2\pi}{3}R^3,$$

where yet one more approximation is introduced by taking the second integral over a circular area of radius R instead of the quadratic area of size $N * N$.

Setting $\pi R^2 = N^2$ gives $R = \frac{N}{\sqrt{\pi}}$.

So that

$$\sum_{i,j}\sqrt{i^2+j^2} \approx \frac{2}{3\sqrt{\pi}}N^3.$$

For example, when $N = 512$, the above approximation gives 50,482,829, whereas the discrete sum is 51,521,231. Thus the approximation error is <2%.

Next, we set $s(i,j) = \frac{1+\sqrt{2}}{2}$ as a reasonable midpoint value. Thus

$$\text{const} \approx \frac{3(1+\sqrt{2})\sqrt{\pi}}{4N}.$$

Inserting in Eq. (14) gives, $k = \frac{3(1+\sqrt{2})\sqrt{\pi}}{4N}\frac{\sum_{i,j} f'(i,j)}{\sum_{i,j} f(i,j)}$

A more detailed analysis shows that $s(i,j)$ is not a constant but grows approximately linearly from 1 to $\sqrt{2}$ for directions between 0 deg and 45 deg. Taking this into consideration when evaluating Eq. (14) would increase the above value of const by about 1.5%.

References

1. S. Shah and X. Xuezhi, "Traditional and modern strategies for optical flow: an investigation," *SN Appl. Sci.* **3**, 289 (2021).
2. A. El Gamal, "Trends in CMOS image sensor technology and design," in *Int. Electron. Devices Meet. Digest of Technical Papers*, pp. 805–808 (2002).
3. A. Guilvard et al., "A digital high dynamic range CMOS image sensor with multi-integration and pixel readout request," *Proc. SPIE* **6501**, 65010L (2007).
4. D. Clark et al., "Defining the computational structure of the motion detector in drosophila," *Neuron* **70**, 1165–1177 (2011).
5. A. Åström and R. Forchheimer "Low-complexity, high-speed, and high-dynamic range time-to-impact algorithm," *J. Electron. Imaging* **21**(4), 043025 (2012).
6. A. Åström and R. Forchheimer, "Time-to-impact sensors in robot vision applications based on the near sensor image processing concept," *Proc. SPIE* **8298**, 829808 (2012).
7. A. Åström and R. Forchheimer, "A high speed 2D time-to-impact algorithm targeted for smart image sensors," *Proc. SPIE* **9022**, 90220Q (2014).
8. R. C. Nelson and J. Aloimonos, "Obstacle avoidance using flow field divergence," *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-11**(10), 1102–1106 (1989).
9. R. Forchheimer and A. Åström, "Near-sensor image processing. A new paradigm," *IEEE Trans. Image Process.* **3**(6), 735–746 (1994).
10. G. Gallego et al., "Event-based vision: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **44**(1), 154–180 (2020).

Anders Åström received his MS degree in computer engineering in 1988 and his PhD in 1993 both from Linkoping University, Sweden. His research areas include architecture and algorithms for smart image sensors. He was an associated professor at Linkoping University until 1999. He has been a vice president of Combitech AB (a subsidiary of Saab AB) and a head of the Industry and Image Processing. He is currently working at the Swedish Police. He holds several patents.

Robert Forchheimer graduated in electrical engineering from KTH, Stockholm, in 1972 and received his PhD from Linköping University in 1979. He founded the Division of Information Coding at LiU which contributed under his leadership to the current techniques for digital television (MPEG). He has further published in the fields of smart vision sensors, packet radio, computer networks, organic electronics, and bioinformatics. He has also cofounded several spin-off companies from the university.