https://doi.org/10.2352/ISSN.2470-1173.2017.11.IMSE-194

Fast, Low-Complex, Non-Contact Motion Encoder based on the NSIP Concept

Åström Anders[1] and Forchheimer Robert[2]
[1]Combitech AB, Universitetsvägen 14, 583 31 Linköping, Sweden
[2]Div. of Information Coding, Linköping University, 581 83 Linköping, Sweden

Abstract

We describe the implementation of a non-contact motion encoder based on the Near-Sensor Image Processing (NSIP) concept. Rather than computing image displacements between frames we search for "LEP stability" as used successfully in a previously published Time-to-Impact detector. A LEP is a single pixel feature that is tracked during its motion. It is found that this results in a non-complex and fast implementation. As with other NSIP-based solutions, high dynamic range is obtained as the sensor adapts itself to the lighting conditions.

Background

We have previously [1][2][3][5] presented a sensor/processor for Time-to-impact (TTI) estimation based on the Near-Sensor Image Processing (NSIP) concept. From a sequence of images, TTI aims at estimating the time when a possible collision may occur between a forward-moving camera and an object seen by the camera.
It is a well-known fact that the image processing required to perform real-time TTI estimation requires a fair amount of hardware resources. The dynamic range of the camera needs to be high, particularly for outdoor applications. To compute spatial motion within the images, optical flow is typically estimated. To do this in real time requires fast computing hardware and data storage that can hold one or more frames.
The solution used in [1] was based on estimating the "inverse" of the motion (how long an image feature stays at the same pixel position). It was shown that this approach drastically reduces the computational load and also lends itself naturally to the NSIP smart sensor architecture [4][7].
In this paper we describe how the same approach can be used to obtain a low-complex and fast non-contact motion encoder. Such an encoder can be used to find for example the speed of a conveyor belt.
Normally a mechanical encoder connected to the conveyer motor would be used. However, if for some reason the encoder should not be in physical contact, optical techniques can be used. One solution is to use a Doppler laser, which is accurate but expensive. Another solution is to use a camera, which is relatively less expensive but requires more processing.

The NSIP architecture

The NSIP architecture can be viewed as a parallel (SIMD) processor integrated with the optical sensor. The number of processors equals the number of pixels in the sensor. In the 1-dimensional case, each such "pixel processor" has a bit-slice architecture containing a one-bit accumulator, a register file of R bits and a simple ALU. Neighboring pixel processors are able to communicate with each other so that a bit can easily be transferred between them. Due to the SIMD type of programming control all pixel processors will perform the same operation. Thus, if a bit is moved from one processor to a left neighbor, the corresponding bit in all processors will be moved. The pixel processor view is particularly useful to understand the interaction between the light sensor and the readout circuit. The CMOS sensor element and its corresponding readout circuit have the principle design shown in Figure 1.

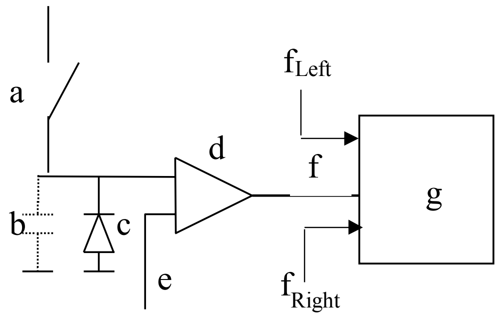

Figure 1, Sensor Processing Element (SPE)

Each pixel consists of a photo diode (c) which is precharged (through switch a) in its reverse direction (thus acting as a capacitor, b) to a fixed level U_0 and then allowed to discharge when illuminated by light. For each photon hitting the diode, there is a possibility of an electron-hole pair formation. When this happens the charged particles are separated and a current pulse is generated that partially discharge the photo diode. The total discharge after a certain exposure time is thus proportional to the amount of light that has fallen on the sensor during this time. To a first approximation the voltage over the photo diode is proportional to the remaining charge (assuming that the diode capacitance is constant). Thus, the relationship between the light intensity I and the voltage U over the photo diode during exposure can be approximated by the linear function

$$U(t) = U_0 - kIt$$

where t is the exposure time and k is a proportionality constant, see Figure 2.

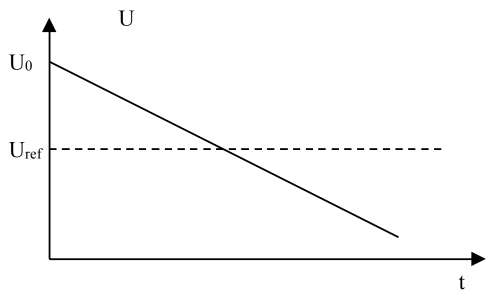

Figure 2, Discharge of photodiode

In a conventional CMOS camera the exposure time, t, is the same for all the pixels and the resulting voltage, *U(t)*, reflects the light intensity at each pixel. This requires that analog levels can be read out from the sensor chip and A/D-converted to allow for further digital processing. However, for an NSIP device the A/D converter will need to be incorporated on the sensor chip otherwise only very rudimentary analog pre-processing can be done on the picture.
Although it is commonplace nowadays to integrate an A/D-converter with CMOS sensors, such a device will take up silicon space and also consume substantial power, at least if high performance is required. The NSIP concept solves this problem in software instead of hardware. Typically, a fix threshold voltage, U_{ref}, is set (on pin e in figure 1), against which *U(t)* is compared (using comparator d) and the time for each photo diode to reach the threshold value is measured. This will lead to an inversely proportional relation between the measurement and the light intensity. Various ways to linearize this relationship exist, see[6]. In many cases the non-linear relationship does not cause any problems and in some cases, such as with scenes containing high-dynamic range of intensities, the non-linearity will even become an advantage.

The NSIP algorithm

Typical traditional steps to estimate conveyor belt speed using a camera is given by the following algorithm:
1. Take two pictures separated slightly in time. Exposure settings need to be chosen such that some image contrast is obtained, either from the belt itself or from objects residing on it
2. Match the pictures using different pixel (or sub-pixel) displacements along the motion direction
3. Find the displacement corresponding to the best matching
4. Map the displacement to a correct metric displacement using calibration data

The algorithm that we propose is similar to this algorithm. However, we do not measure the displacement in term of pixels between two consecutive images, but rather measure the time the object stays within a pixel distance. This is the same idea as was used in the former TTI algorithm. As for the TTI algorithm we do not measure the whole object. Instead we locate single pixel features which we denote local extreme points (LEPs) and track for how many frames they stay within the same pixel. This gives the following basic algorithm:

1. Locate a number of feature points (LEP, Local Extreme Points)
2. Keep track of how many frames each LEP stays inside the same pixel ("LEP run")
3. Compute an average value of the runs of all LEPs
4. Map the average value to a correct metric displacement using calibration data

We propose a "1.5D" architecture consisting of N columns and n rows, where N >> n, shown in Figure 3 where n is equal to three. This means that we have full resolution in one dimension and a sufficient resolution in the other dimension to obtain a local neighborhood.

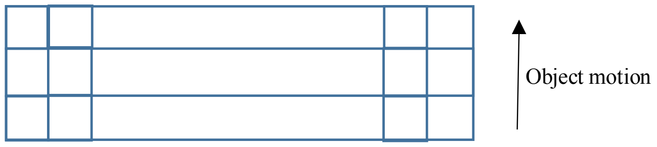

Figure 3, 1.5D Sensor layout

Each pixel processor that handles a pixel at the center line consists of a photodiode, a comparator, a simple logic unit, and some memory. The pixels adjacent to the center line do not have a conventional pixel processor. Each such pixel only consists of a photodiode and a comparator. The output from these pixels are processed by the processors along the center line. The LEP's are computed from the n pixels in the vertical (motion) direction.
Although this algorithm replaces the traditional image matching procedure, it is still unsuitable for the NSIP architecture, mainly because the number of generated runs is image dependent and the length of each run needs to be stored until the averaging can be done over all the runs. This is due to the SIMD character of the NSIP processor. For this reason we introduce a further simplifying step, namely to retain only the longest run in each of the N pixel processors after a suitable number of frames. The averaging is then computed only over the L retained runs.
The resulting average of the LEP runs gives a first estimate of the motion. However, as the runs have integer length it is necessary to compensate for the bias that occurs towards longer runs as only the longest run from each pixel processor is retained. As an example, if the true motion corresponds to 5.5 frames per pixel, runs of length 5 and length 6 should occur equally often. But if the number of runs seen by a pixel processor happens to be e.g. 4, the probability that a run length of 6 is reported from this processor will be as high as $1\text{-}0.5^4 = 0.94$. With similar behavior from other pixel processors, the average run length as reported from the NSIP processor will then be 5*0.06+6*0.94 = 5.94, far from the correct value of 5.5. It is thus necessary to apply a correction to the measured value. The correction will be primarily based on the number of LEPs seen by each pixel processor. This number can be computed by the NSIP processors during the exposures and used in the final estimation.
Given the algorithm and the architecture described above we obtain the following performance figure from a very simple example system:

Field of view	D [m]
Camera resolution	N [pixels]
Subpixel resolution	M [ratio]
Speed	v [m/s]
Pixel form factor	F =a/b [ratio]

Minimum Line rate is given by:

$$f_{LR} = \frac{v}{F\frac{D}{N}}M = \frac{v\,N\,M}{F\,D}$$

Assuming a field of view of 1 m, a subpixel resolution of 5, and a square shaped pixel (F = 1) we need to achieve the minimum line rates described in Table 1, for different camera resolutions and object speeds.

Table 1, Minimum line rate

N	v	Min Line rate	Measurement output rate (M=5)
500	1	2,5 kHz	40 Hz
500	10	25 kHz	400 Hz
5000	1	25 kHz	400 Hz
5000	10	250 kHz	4 kHz

The program complexity for each loop, which runs for each line is in the order of 100 cycles. This means that the circuit must be clocked at least at 25 MHz, which is reasonable based on known implementations of NSIP sensors [7,8,9].

If we adjust the pixel form factor to 10, i.e. the width of the pixel along the array is 10 times smaller than its height, the line rate can be reduced by a factor of 10. However, this requires that the LEPs are sufficiently separated in the motion direction so that they do not appear in the same pixel at any time.

Improving noise immunity with a modified architecture

Based on simulations we have seen that the algorithm works well and gives good performance. However, when adding noise to the sensor readout, the performance degrades quickly. This is consistent with our experience with real implementations of the TTI algorithm as well. The problem is that LEP runs break up and tend to be shorter than they should be, which results in a too high estimate of the speed.
Therefore, we propose the following extension to the NSIP architecture.

The number of sensor rows is extended to n=5 which will allow to use larger filters. We will also add, to each pixel, an extra comparator, the reference level of which can be set externally. Typically the reference level of the second comparator will be very close to the main comparator, such as 1% higher depending on the noise level that needs to be handled. This is shown in Figure 4. Alternatively, to avoid the problem of accurate matching of the two comparators, the same comparator could be used twice, through a quick change of the reference value.

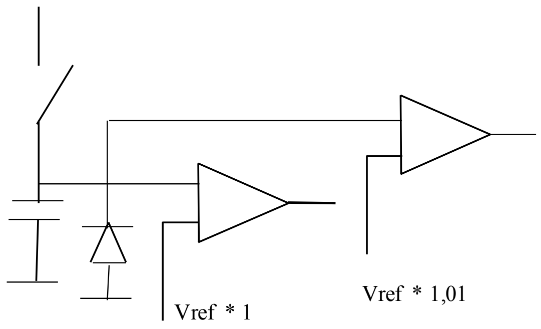

Figure 4, New sensor processing element design

Table 2 considers the effect of using two different voltages for two neighboring pixels. The table lists all possible states and their corresponding interpretation. Here, A is the output from the comparator with the higher voltage reference (V_{ref} * 1.01) from the center pixel and B is the corresponding output from the comparator with the lower reference voltage. C is the output from the comparator, with the higher reference voltage, from neighboring pixel, and D is the output from the neighboring pixel's comparator with the lower reference voltage.

Table 2, Possible states given two neighboring pixels with two reference voltages

A	B	C	D	Interpratation
0	0	0	0	No hit in either pixel
0	0	1	0	A brighter neighbor
0	0	1	1	A much brighter neighbor
1	0	0	0	Brighter than neighbor
1	0	1	0	Equal bright, little difference
1	0	1	1	Brighter neighbor
1	1	0	0	Much brighter than neighbor
1	1	1	0	Brighter than neighbor
1	1	1	1	Both bright

In the original algorithm for LEP-detection, a pixel was assigned to be a LEP when its two nearest neighbors had lower intensities than the center pixel. This can be described as

$$L_{X,0} = (P_{X,0} > P_{X,-1}) \cap (P_{X,0} > P_{X,+1})$$

Using the improved architecture we can modify the above condition to make it more insensitive to noise. Here, P is the output from the comparator with the lower voltage, V_{ref} * 1 in Figure 4, and Q is the output from the comparator with the higher voltage, V_{ref} * 1,01. Thus, we combine the extended neighborhood with the use of two reference voltages.

$$L_{X,0} = (P_{X,0} > P_{X,-1}) \cap (P_{X,0} > P_{X,+1}) \cap (P_{X,0} > Q_{X,-2}) \cap (P_{X,0} > Q_{X,+2})$$

This gives much better immunity to noise.

Simulations

To simulate the performance we used a segment of an image shown in Figure 5. The sensor architecture used was a 512 by 5 sized sensor array. The processors in the middle row have a complete architecture as shown in Figure 1. The other four rows have a simplified design as shown in Figure 4. The speed of the image along the vertical direction of the sensor array is 1/5.5 pixel per sample. I.e. after 55 samples the image has moved 10 pixels. It follows that the 32 image rows in Figure 5 correspond to 176 scan lines.

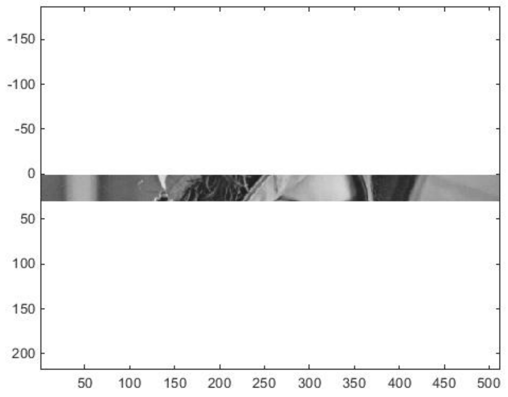

Figure 5, Test image

To obtain the LEPs we used the logical function described earlier

$$L_{X,0} = (P_{X,0} > P_{X,-1}) \cap (P_{X,0} > P_{X,+1}) \cap (P_{X,0} > Q_{X,-2}) \cap (P_{X,0} > Q_{X}$$

based on a 1x5 neighborhood and two reference voltages. The result from 160 line scans is shown in Figure 6.

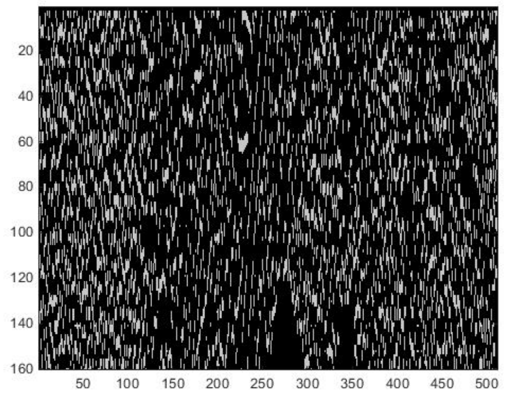

Figure 6, LEP image

If we magnify a section of the LEP-image we get Figure 7.

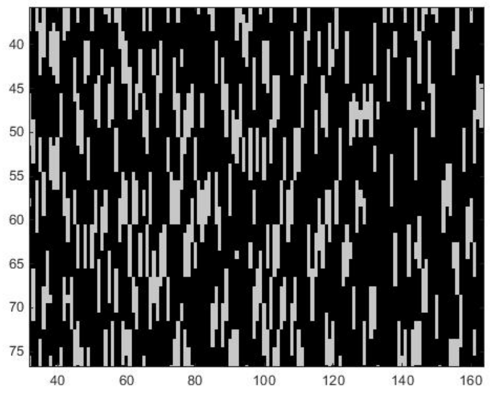

Figure 7, Magnified LEP image

With increasing number of line scans we will have a larger value of M. This is shown in Figure 8. As mentioned earlier, we keep the longest run in each column. These runs are then averaged across the array.

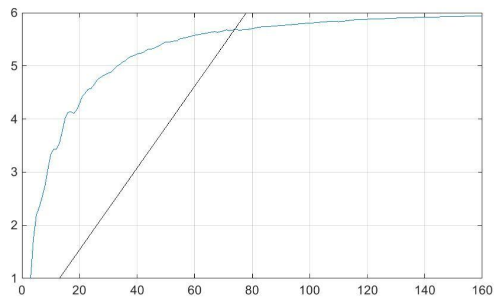

Figure 8, Different values for M depending on the number of lines scanned (L). Speed is 1/5.5 pixels per scan

We have found empirically that a good value for the averaging window, L, is 13 times the subpixel ratio, M. This is related to the previously described bias towards larger runs. The line M = L/13 is therefore inserted in figure 8 to yield the estimate of M. In Figure 8 we can see that an L-value of 70 gives a good estimate of M.

Below follows the result at a slower speed. In this case the speed is reduced from 1/5.5 pixels per scan to 1/9.5 pixels per scan. In Figure 9 it is seen that the LEPs have grown larger.

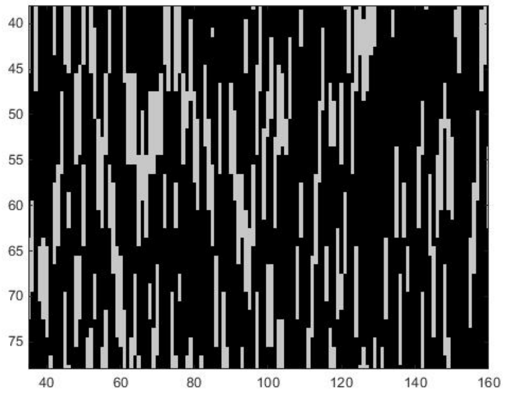

Figure 9, Magnified LEP image with size 9.5

From Figure 10 it is seen that an averaging window of 13*9.5 again gives a good estimate of M.

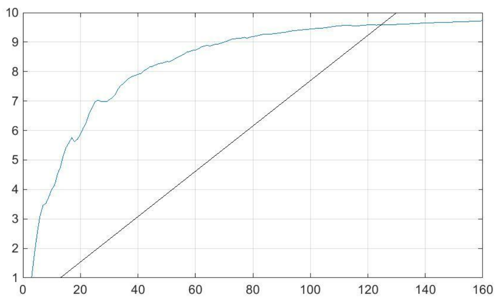

Figure 10, Different values for M depending on the number of lines scanned (L). Speed is 1/9.5 pixels per scan

Applications

We have used the conveyor belt encoder as an example of a typical application of the described motion encoder. There are of course many other applications where the features of the sensor are useful, such as in a paper mill, in the textile industry or as skid sensors for cars.

Conclusions

We have shown that the "inverse" computation of object motion based on stability of LEPs within pixels results in a non-complex solution suitable for the NSIP smart sensor architecture. This opens up for fast and compact implementations that include both the optical sensor and processing unit. The concept is general and has earlier been demonstrated in a time-to-impact sensor. A suggestion for improving the noise immunity is presented based on two simultaneous readings of the sensor data with slightly different reference levels.

References

[1] Åström A, Forchheimer R., "Low-complexity, high-speed, and high-dynamic range time-to-impact algorithm" Journal of Electronic Imaging 21(4), 043025 (2012)

[2] Åström A, Forchheimer R, "Time-to-impact sensors in robot vision applications based on the near sensor image processing concept," Proc. SPIE 8298, 829808 (2012).

[3] Åström A, Forchheimer R, "A High Speed 2D Time-to-Impact Algorithm Targeted for Smart Image Sensors." Proc. SPIE 9022, 90220Q (2014).

[4] Åström A, Forchheimer R., "Near-Sensor image Processing" Advances in Imaging and Electron Physics, Vol 105, (1999)

[5] Åström A, Forchheimer R., "Impact time from image sensing" Patent pending, WO 2013/107525, (2012)

[6] Åström A, Forchheimer R., Danielsson P-E, "Intensity Mappings within the Context of Near-Sensor image Processing," IEEE Trans. Image Processing, 7, 12, December, (1998).

[7] Forchheimer R, Åström A, "Near-Sensor Image Processing. A New paradigm." IEEE Trans Image Processing, 3, 6, 735-746 (1994).

[8] Eklund J-E, Svensson C, and Åström A, "Implementation of a Focal Plane Processor. A realization of the Near-Sensor Image Processing Concept" IEEE Trans. VLSI Systems, 4, (1996).

[9] El Gamal A., "Trends in CMOS Image Sensor Technology and Design," International Electron Devices Meeting Digest of Technical Papers, pp. 805-808 (2002).

[10] Guilvard A., et al., "A Digital High Dynamic Range CMOS Image Sensor with Multi-Integration and Pixel Readout Request", in Proc. of SPIE-IS&T Electronic Imaging, 6501, (2007).

Author Biography

Anders Astrom was born in 1964. He received the M.S. degree in computer engineering in 1988 and the Ph.D. degree in 1993, both from Linkoping University, Sweden. His research areas cover SIMD architecture for image and radar signal processing and architecture and algorithms for smart image sensors. He was an associated professor at Linkoping University until 1999. He is vice president of Combitech AB, head of Industry, which is a subsidiary to Saab AB. He is responsible for the image processing at Combitech. He holds several patents.

Robert Forchheimer received the M.S. degree in electrical engineering from the Royal Institute of Technology, Stockholm (KTH) in 1972 and the Ph.D. degree from Linköping University in 1979. During the academic year 1979 to 1980, he was a visiting research scientist at University of Southern California where he worked in the areas of image coding, computer architectures for image processing and optical computing. Forchheimer's research areas have involved data security, packet radio communication, smart vision sensors, image coding, optical networks and organic electronics. He has authored and coauthored papers in all of these areas and also holds numerous patents. He is the cofounder of several companies within the University science park. Forchheimer is currently Professor Emeritus at Linköping University. His current work concerns computer networks and organic electronics.