Spatial-Temporal Motion Estimation for Image Reconstruction and Mouse Functionality with Optical or Capacitive Sensors

Fabrizio S. Rovati, Pieluigi Gardella, Paolo Zambotti and Danilo Pau*

Abstract — *This paper describes an innovative solution to perform mouse functionality ("navigation") and image reconstruction ("scanning"). The application uses a recursive block-matching motion estimation algorithm and an optical or capacitive sensor. Results show this solution provides integrated mouse and scanner functionality at a very low cost.*

Index Terms — Capacitive finger sensor, image reconstruction, motion estimation, optical mouse, scanner.

I. Introduction

TODAY, an optical CMOS sensor with resolution of tens of pixels for each dimension can be integrated on a chip with processing logic at a very low cost. A capacitive sensor, used to scan fingerprints, is also affordable, if we use a 'stripe' version able to capture a small part of the finger image. The sensor can be as small as 256 pixels by 2 lines; this compactness makes it ideal for applications where power consumption, component dimensions and cost must be minimized.

Ideal target applications are navigation/mouse and/or a portable scanner. We can also use the fingerprint sensor for non-critical security features; for example, to gain access to a consumer device, or simply to recall preferred personal settings. The user would move the sensor on a surface (see fig. 1) and by processing the input images, we are able to extract the movement occurred and/or to reconstruct the underlying global image.

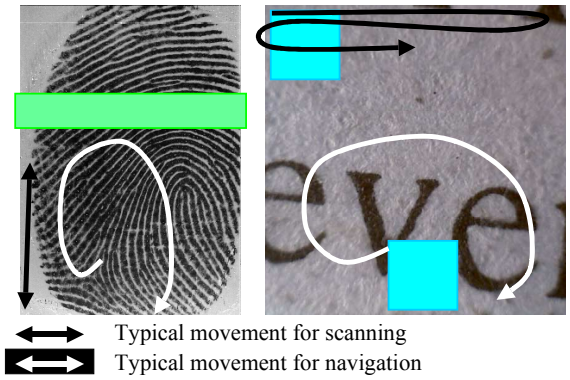

Fig. 1: Application example. User moves finger upon capacitive sensor, or slides optical sensor on an image or surface

* All authors are with STMicroelectronics Advanced System technology Labs, Agrate Brianza, Italy

To be able to reconstruct accurately a complete image from such small snapshots, or to estimate the motion imposed to the sensor we need a reconstruction algorithm. Simply stacking input images will not produce satisfactory scanning results, and will not give any information about navigation, as fig. 2 demonstrates.

Fig. 2: Example of fingerprint scanning. By simply stacking the sensor output images (on the right), we have an unacceptable result

II. State of the Art Review

This paper describes an innovative application using a spatial-temporal block-matching motion estimator to reconstruct bigger images starting from a small optical or capacitive sensor output, and to provide mouse functionality. Exactly the same algorithm is able to achieve both functions. Applications range from optical mice to user identification for small PDA's or mobile phones. This double integrated functionality is very novel in the art, as we target applications that are, if at all present today, implemented without a unified approach. Example for optical mouse can be found in [1]. We will therefore review the state of the art of the two single applications; keeping in mind we also have the advantage of offering an integrated approach.

Manuscript received June 10, 2003

A. *Optical mouse state of the art*

Nowadays optical mouse navigation functionality is based mainly on edge detection and/or phase correlation, algorithms much more complex than the proposed one. Complexity of the processing involved for finding out the motion limits the frame rate to around 1000 frames per second, especially in wireless mice, where power consumption is very important. Having a high number of pictures per unit time is beneficial to be able to follow hi-speed motion, for example when we use the mouse to play games. It also helps when scanning a document or a picture, as it makes the process less tedious by being able to make it more quickly.

Correlation between hi-speed motion detection and high me rate is simply explained: there is an intrinsic threshold to respect to be able to estimate motion between two frames, whatever the technology used. This limit is the maximum displacement occurred between the frames, e.g. P pixels in each direction. Let's assume this as a constant. To find out maximum detectable speed, we must multiply the displacement by the number of times we are able to measure it in a second. This is given by the frame rate, F. Having a higher frame rate F means covering a wider displacement per unit time, P*F, which grants the ability to follow faster motion.

B. *Fingerprint sensor state of the art*

Capacitive sensors are composed by a bi-dimensional array of capacitors laid out on a silicon surface in order to be able to capture finger's ridges and valleys. Ridges are the pieces of skin in relief; valleys are the remainders of it. Vicinity of finger's skin modifies the capacitance of each of such devices. The varied capacitance can be sensed in terms of different voltage resulting from applying a fixed electrical charge. After A/D conversion of this physical measure, the detection gives an array of values (one for each capacitor) that can be seen as a picture of the finger just scanned. Each pixel is monochromatically encoded into 8 bits: 0 if totally on a ridge, 255 if totally on a valley. We therefore have a picture composed of 8-bit pixels, the same output of a monochromatic optical sensor.

State of the art capacitive fingerprint sensors are 'complete image' sensors, with large cost in term of silicon area. A typical array [2] is composed by 256x360 sensing elements, with a pixel pitch of 50 μm. This geometry leads to an area of around one squared inch, enough to capture the entire fingerprint at once. Such a device has some distinctive drawbacks: cost (silicon area is huge), power consumption and physical size. The latter is important if this sensor has to be used into miniaturized terminals, like a Personal Digital Assistant (PDA) or a mobile phone, where area as well as cost and power consumption are key factors.

To overcome these disadvantages, instead of capturing a whole fingerprint at once, in our approach we scan it by letting the finger slip over a 'stripe' sensor. The difference in this case is sensor size: the horizontal dimension stays the same, but it will be only a few lines high, ideally only two. Therefore the sensor area is reduced by a factor of (360/2) = 180, with evident benefits in terms of area and power consumption. To date, this is a very high lines reduction ratio. References [3]-[6] can be made to other devices that were able to reduce the size by factors of 11 at most, i.e. using 32 lines. Using a stripe sensor, of course, we have to introduce the step of reconstructing the whole fingerprint starting from the partial images taken. The algorithm we propose performs this task. Once this function is present in the device, it can also be used as mouse output, just reusing displacement information and discarding the reconstructed image.

III. MOTION ESTIMATION

The algorithm we present is based on spatial/temporal recursive motion estimation. Motion estimation is a very much known technique in the digital video-processing field, where it is used to exploit temporal redundancy (i.e. similarities among different pictures) during video compression. Another use is for interpolation of missing information, for example for frame rate up conversion.

Various types of motion estimations have been proposed in literature [7], [8]. The most successful have been pixel-by-pixel motion estimation (where each single pixel is estimated by its own) and block-matching motion estimation, where a bi-dimensional array of pixels is estimated at once, to decrease computational complexity.

A. *Block-matching motion estimation*

The basic principle of block matching motion estimation (see fig. 3) is to use two images A and B. We select an area C inside first picture A that needs to be 'motion estimated' on the other and we look for an area D inside picture B that is as similar as possible to area C. The position of each area C, D is identified by the position of its upper-left pixel. The difference in position of these two pixels in the respective pictures is called motion vector (E), and it is the expression of the motion that area C underwent between the two images. The motion

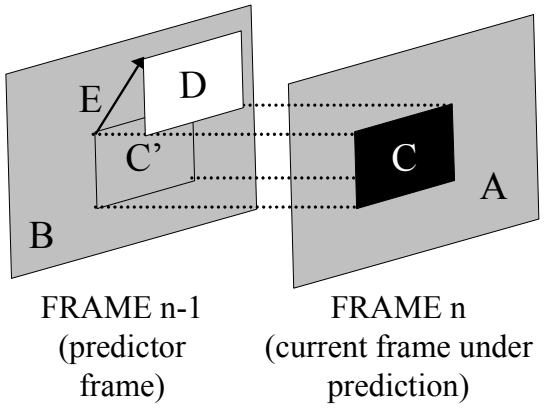

Fig. 3: Block-matching motion estimation technique

vector can also be imagined as the vector joining the projection of upper-left C pixel into B (C') and upper-left D

pixel. The area D is often called the predictor, as we are trying to 'predict' area C from D.

Ideally, we could test all the D areas that we can extract from the B frame, to provide a matching as accurate as possible. This approach is called Full Search. This generally provides very good estimations, but at the expense of great computational complexity. For example, to perform an estimation of an X by Y area from an N by M frame, we would require to test (M-X+1)(N-Y+1) areas, each X*Y pixels wide. Moreover, normally matches need sub-pixel accuracy, i.e. movements of ½ or ¼ of pixel or even less need to be tested. The formula above would be multiplied by, respectively, a factor of 4 and 16. Each test usually consists in computing the Mean Absolute Error (MAE) function:

$$\mathrm{MAE} = \frac{1}{X * Y} \sum |p(i,j) - q(i,j)| \qquad (1)$$

With i = 0 to X-1, and j = 0 to Y-1; p(i,j) is a pixel from area C , position i,j; q is the pixel in the same position from area D. If areas are perfectly identical, MAE = 0. In general, the lower the MAE, the more similar we can assume areas C and D are. The inner part of (1) is often called Sum of Absolute Differences, SAD. One MAE is then the average of X*Y SAD values

B. Reduced-complexity block-matching motion estimation

In literature [7]-[13] there are other approaches that try to decrease the computational complexity by a focusing on the selection of candidate predictors. One among the most successful is the spatial-temporal approach [14]-[16]. It exploits the principle that when estimating a sequence of video images, results of successive estimations are not independent, but they instead tend to be strongly correlated. This is true spatially and temporally. Temporally means that if we perform an estimation of picture N over N-1, and then of picture N+1 from N, the results will be highly correlated, i.e. the vectors will be similar. Spatially means that, in case we have more than one area C that we want to estimate in picture A, results of areas C1, C2, Cn are usually similar, at least if the blocks are neighbors. We will call motion vectors that are taken from estimations of different parts of the same picture "spatial vectors"; vectors that are taken from estimations of parts of different pictures will be referenced as "temporal vectors".

Algorithms following this approach exploit the correlation among motion vectors to decrease the number of matches and increase consistency of the results. It has to be noted that these algorithms have been developed for TV (one PAL image is 720x576 pixels) sequences digital compression. In this application motion estimation is performed on 16x16 pixels blocks, so there are a lot of vectors to be taken as spatial-temporal references (1620 for each frame). The algorithm will test a certain amount of such temporal and spatial vectors as the starting point of the estimation. A second 'refine' phase will test small variations of the first phase winner in order to see if a better match is available in some neighboring position (see fig. 4).

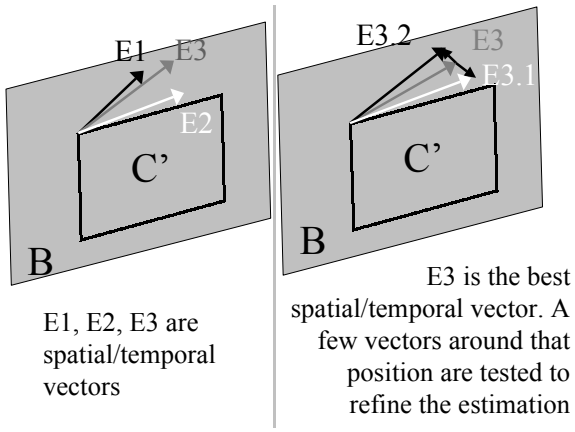

Fig. 4: Spatial/temporal recursive block-matching motion estimation

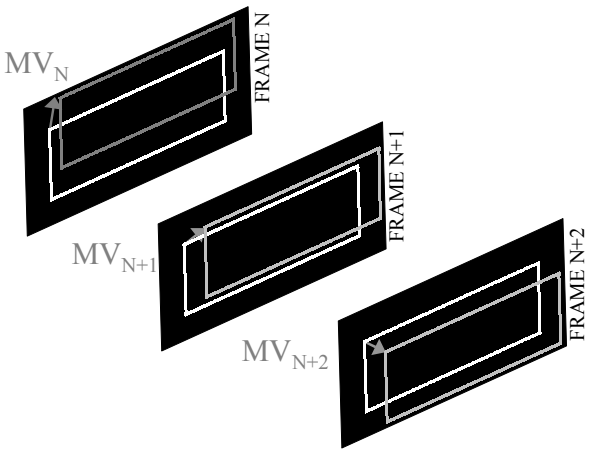

Fig. 5: Recursive motion estimation applied to capacitive stripe (or optical) sensor output: we have only one vector per frame, but high frame rate

IV. PROPOSED ALGORITHM

In our application, however, there is not plenty of such vectors among which to select the best starting point for new estimations. Pictures used for estimations are very limited in dimensions: 2 lines by 256 columns for capacitive 'stripe' sensor, or 20 by 20 for the optical one. We choose this dimension for the optical sensor as it yields a similar number of pixels per frame as the capacitive one, even if aspect ratio is different. Another constraint comes from the application target, which is to estimate a single global motion, not a plurality of local motions. The net result of the above constraints is that we can inherit only one vector per previous

picture, and no vectors from previous estimations of current picture, as shown fig. 5. It is a theoretical challenge to prove that a recursive algorithm can work also in this 'depleted' environment (1 temporal and no spatial vectors per frame).

Obviously, the correlation between previous motion vectors and the current estimation to be performed is inversely proportional to the temporal distance between the current frame and the one from which vectors are taken. The high frame rate, anyway, makes us infer that we could use temporal vectors from more pictures in the past, not only the preceding one.

We modified the core of the spatial-temporal approach to suit application's conditions better, becoming a temporal-only approach. We added test of linear combinations of previous-frames temporal vectors as potential 'seeds' to overcome the scarcity of candidates. In particular, considering the underlying physics of the application, 'physically sound' candidates have been generated, such as constant speed, constant acceleration, and so on. After having tested these candidates, an 'update' phase of the algorithm would refine the best among the vectors tested so far to see if the prediction could be bettered. A patent application for this technique has been filed [17].

Note that in this case the goal of the estimation is not to generate a 'prediction' for the current image, but rather to compute the shift between the two images to stitch them together appropriately. In some respect, this application is more difficult than the prediction in video encoding. Even a small percentage of errors would be very visible in the reconstructed image or movement, whereas a single sub-optimal block prediction is not critical in video compression, as it is concealed by encoding the block as intra, i.e. without prediction.

V. SIMULATIVE RESULTS

We developed a bit-accurate fixed-point algorithmic model and we created a simulation test bench to assess the performances. The verification environment comprised two different tests, one for the optical sensor, and another for the capacitive one. In both cases, inputs of the complete chain were full pictures taken with high-resolution sensors. We used fingerprint samples captured using a conventional 256 by 360 sensor, and optical images taken with a Megapixel CMOS sensor.

These images formed the input to a 'strip images generator'. This program takes as input the big image, and a motion trajectory, and extracts from the full picture the equivalent stream of samples a small optical or stripe capacitive sensor would generate, if passed on the big image with the specified motion. Trajectories are defined as a sequence of points, or by a mathematical formula. In particular, thanks to an internal interpolation feature, movements can be programmed to be of any fractional amount, for example 0.12 pixels per frame, to better test fast and slow motion on the sensor.

These images were then given as input to the reconstruction motion estimator, which would then work on them, compute the relative movement among successive frames and then stitch them one over the other according to the results of the estimation, as shown in. fig. 6. We could then compare the estimated with the original movement given in input to the strip image generator, and compute the percentage of correct or wrong estimations.

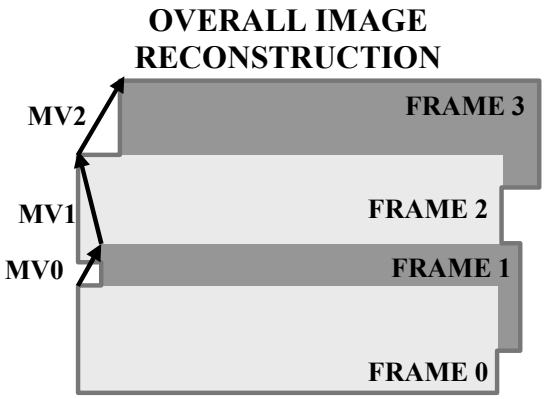

Fig. 6: Method for stitching sensor output images together to obtain the reconstructed picture

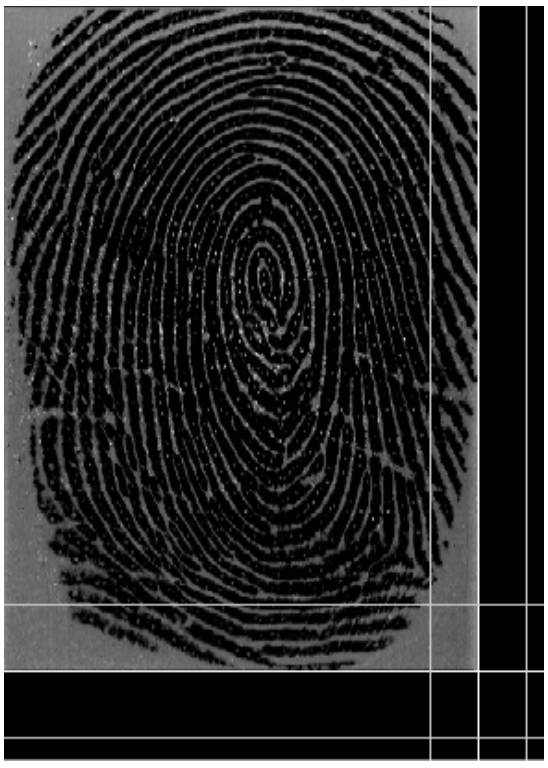

Fig. 7: Result of fingerprint reconstruction for 0.37 pixels/frame vertical motion, no horizontal motion. Starting image is particularly dark. White central lines show where picture would stop if reconstruction was perfect. Surrounding lines show +/- 10% stretching error marks

Results have demonstrated the validity of the algorithm, with both optical and capacitive sensors, for mouse and scanner application. Reconstructed fingerprints match in height and width very accurately the original ones, and trajectory of estimated motion follows very closely the original one. Fig. 7 through 9 show several fingerprints with different characteristics (bright, dark, noisy, smeared) scanned with a range of different speeds in horizontal and vertical directions.

Fig. 8 Result of reconstruction for 0.6 pixels/frame vertical and 0.37 pixels/frame horizontal motion, starting from a smeared image. Grey area on the right is due to padding of the sequence generator, to be able to generate horizontally moving images.

To avoid stretching or compressing the fingerprint is a particularly important feature in order to be able to recognize correctly the owner of the fingerprint via automatic matching algorithms.

Results are also good when the optical sensor is used. About scanning function, even starting from very small (20 pixels by 20 pixels) input images, results are acceptable, as shown in fig. 10. A small drift is visible in scanning intersections, because estimation errors add up, so even a very small percentage of errors can lead to an imperfect crossing. This problem appears because we need thousands (2704 theoretical minimum) of sensor images to obtain a 1024x1024 resolution image. During all these estimations, even a very small percentage of errors lead to a visible artifact. Applying additional matches can solve the problem when crossing parts of the reconstructed image already acquired to better center them. This is a topic for future research. Another obvious solution is to increase the sensor resolution to reduce the number of input images necessary to cover one full picture.

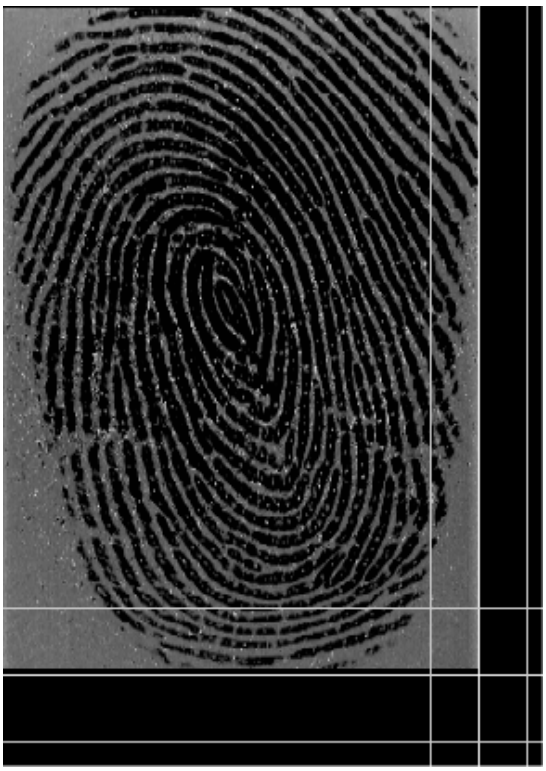

Fig. 9: Result of reconstruction for 0.89 pixels/frame vertical motion on a dark, noisy fingerprint image

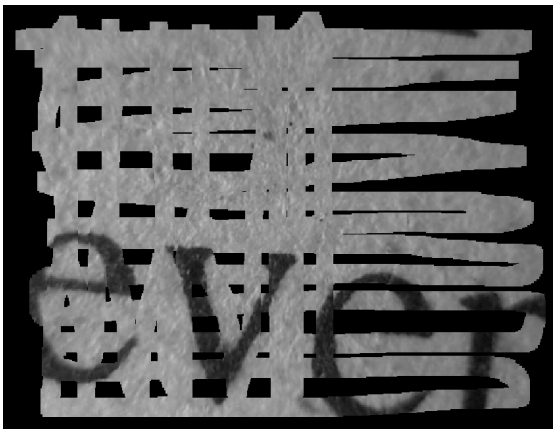

Fig. 10: Results of simulation of scanning a printed paper with the optical sensor and the combining the pictures with the proposed algorithm

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.