An Integrated Image Motion Sensor for Micro Camera Module

Fabrice Gensolen, Guy Cathébras, Lionel Martin, Michel Robert

An Integrated Image Motion Sensor for Micro Camera Module

F. Gensolen[1,2], G. Cathebras[2], L. Martin[1], M. Robert[2]
[1] *STMICROELECTRONICS, ZI de Rousset, BP 2, 13106 Rousset, France*
[2] *LIRMM, Univ. Montpellier II / CNRS, 161 rue Ada, 34392 Montpellier, France*
fabrice.gensolen@lirmm.fr

Abstract

This paper deals with the building of a vision sensor which provides standard video capture and the associated global motion between consecutive frames. We aim at proposing embedded solutions for mobile applications. The application we focus on is video stabilization. Therefore the global motion we consider here is the one typically produced by handheld devices movement. We extract this global motion from local motion measures at the periphery of the image acquisition area. Thanks to this peculiar and "task-oriented" configuration, we take advantage of CMOS focal plane processing capabilities without sacrificing the sensor fill factor. We have at first validated the technique by software. Then we have designed peripheral custom pixels dedicated to motion detection and we are implementing it in a CMOS 0.13µm technology.

1. Introduction

Although CMOS image sensors only appeared in the middle of the 90's, they are expected to reach half of the total image sensors market this year 2005. This is mainly due to the increasing new market of mobile phones camera [1].

Our objective is to develop a smart CMOS image sensor for mobile systems (PDA, mobile phone). Such handheld devices are very shake prone and often provide trembling video; also we focus in this paper on video stabilization.

The best way to stabilize video is to perform an optical correction using gyro or inertial sensors and mobile optics/sensors, stabilizing directly the incidence of the light onto the focal plane [2]. Nevertheless, this is a costly and burdening solution for embedded devices.

The electronic approach of this task consists in analyzing the displacement between two consecutive frames of the video, so called global motion (G.M.) or camera motion. In order to separate the intentional motion from the unwanted one, one can apply a temporal filtering [3]. The high frequency component stands for the unintentional movement and must be compensated (Figure 1). This compensation corresponds to slip the visualized window in the original larger image in order to make the video stable. This is the stabilization scheme we have adopted. We focus in the present paper on the crucial global motion estimation stage of the processing.

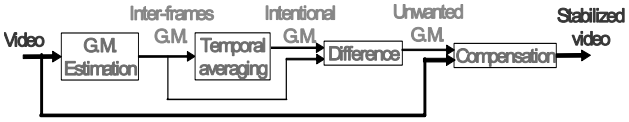


Figure 1 : Block diagram of the video stabilization.

Considering the signal processing architecture, such a motion estimation task can be realized as a post-processing of digital images coming from the imager (Figure 2). In a time to market point of view, this is a very efficient way to implement an image processing on silicon. However, that means to process the huge amount of video data serially, which is very time and power consuming [4].

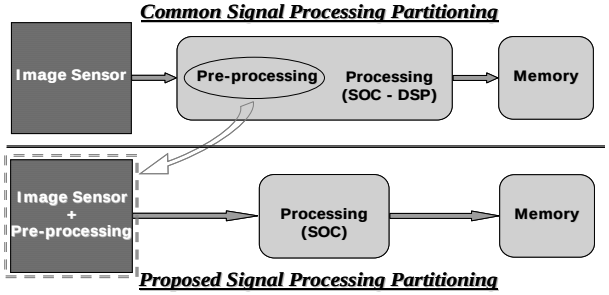


Figure 2 : Signal processing partitioning.

In this paper, we investigate another way to perform motion estimation task by reporting part of the processing at pixel level. This approach speeds up the processing time and alleviates the computing power by making use of parallelism of the pixels needed in image sensors for light spatial sampling. The main drawback of this kind of silicon integration is the increased area per pixel, which decreases the fill-factor and the image resolution. But we present and validate in this paper a new global motion estimation technique based on local motion measures at the periphery of the image acquisition area. Thanks to this peculiar and "task-oriented" configuration, we take advantage of CMOS

focal plane processing capabilities without sacrificing the sensor fill factor. Indeed, the silicon area has become the main contribution to the cost of image sensors, accounting for around 70% of the overall cost.

We describe in section 2 our global motion estimation technique, based on peripheral local motion measures. Section 3 is dedicated to its validation by software implementation, and section 4 to the evaluation and the partitioning of the processing. Then we describe in section 5 the transfer of part of it onto focal plane, performing pixel level processing.

2. Global motion estimation

2.1. Principle and basis

In order to describe the global motion between two consecutive frames, we make use of a four components parametric model, called similarity model. This model allows us to describe the main inter-frames global movements perceived in the focal plane. That is to say horizontal and vertical translations, rotations around the optical axis, and zoom. It is also a good tradeoff between complexity, noise sensitivity, and description of the inter frames movement.

Two kinds of motion are generally present in common video captures with handheld devices like cell phones: the one due to mobile elements in the scene, and the background one (Figure 3). For video stabilization purpose, this last background movement is of main interest as it informs directly about the camera/global motion.

Moreover we point out that the periphery of images is particularly interesting for this task. Indeed, this area contains local motions that better constrain the global motion parameters. Also, these local motions are well distributed in the images and background is often on the periphery of images. Therefore, we only focus on this area to extract the desired global motion (Figure 3).

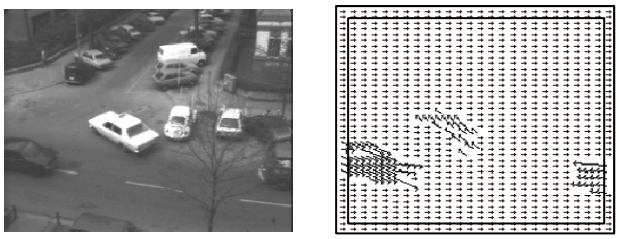


Figure 3 : Example of a scene, with the associated local motion vectors for a left panning of the camera. Edges on the right point out the area of interest.

2.2. Global motion estimation procedure

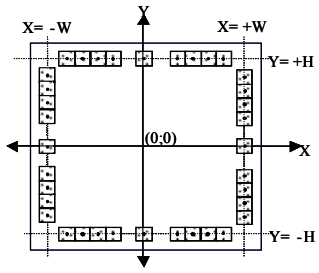
Let us suppose that a picture "n" is transformed according to a geometric combination of a rotation θ, a zoom factor α, and two translations Tx and Ty, in another picture "n+1": a pixel "j" with cartesian coordinates (Xj(n),Yj(n)) in frame "n" become the pixel with cartesian coordinates (Xj(n+1),Yj(n+1)) in frame "n+1". The system in Figure 4 describes such a geometric transformation. Applying this transformation to all the points of the area of interest, that leads to the following linear over determined system:

$$P = K \times M \quad (1)$$

Where P is the positions matrix (Xj(n+1),Yj(n+1)), K the matrix linking these positions to the ones of frame "n", composed of (Xj(n),Yj(n)) coordinates with "0" and "1", and M contains the four transformation parameters. These parameters are Tx, Ty, $\alpha.\cos\theta$, and $\alpha.\sin\theta$. Then knowing P and K, we are able to determine the four global motion parameters M thanks to an optimization process.

Matrix P is obtained summing original cartesian positions of pixels in frame "n" with local motions of this pixels between frame "n" and "n+1". The optimization operation is performed here in a least squares sense [5], and the resulting estimation can be written as:

$$M = (K^T \times K)^{-1} \times K^T \times P \quad (2)$$



$$\begin{cases} Xj(t+dt) = \alpha * \cos\theta * Xj(t) + \alpha * \sin\theta * Yj(t) + Tx \\ Yj(t+dt) = -\alpha * \sin\theta * Xj(t) + \alpha * \cos\theta * Yj(t) + Ty \end{cases}$$

Figure 4 : Geometric system setting.

3. Software implementation

In order to validate our approach, we first evaluate the performances by software. Hence, we need to compute the local movement estimations at the periphery on the two consecutive images.

3.1. Local motion estimation

Several algorithms exist to perform local motion estimation, but one of the most efficient ways is to carry out pixel, or area, correspondence. We have chosen two algorithms for extracting local motion vectors: the first is the well known full search block matching (FSBM), and the other comes from [6]. The last technique, called Census transform, consists in a local texture coding which results in a binary code for each pixel that is then tracked from one frame into the next.

3.2. Performances

Using Matlab software, we have characterized our global motion estimation technique with respect to outdoor and indoor scenes. Firstly we built synthetic video sequences starting from a high resolution picture which we transformed and from which we picked CIF ones with known displacements. Then we also grabbed real video sequences thanks to the same digital camera, providing a 15 im/s CIF video. Both synthetic and real sequences contain the same image texture. They have been captured in the same illumination conditions with various constant amplitudes of movement. These amplitudes are always lower than: 5% of the image size for translation, 3° for rotation, 2% for zoom. Both represent indoor and outdoor scenes.

The inter frame global motion being unknown in real sequences, we apply the precise and robust algorithm developed by [7] in order to obtain our reference motions[1]. This algorithm is proven reliable in various applications [8].

We report in the following Table 1 the first results of our characterizations in terms of error percentage to the reference, or known, motion. As we can see, raw Census transform motion estimation gives lower results than block matching.

Table 1 : First performances achieved in terms of error percentage to the reference motion.

	*Census 5*5*	*Block matching*
Synthetic outdoor	5.8 %	0.02 %
Real outdoor	71 %	12%
Synthetic. Indoor	6.2 %	0 %
Real indoor	82 %	15 %

4. Towards a vision system on chip

Our final objective is to integrate the proposed technique to an image sensor. The main constraint, as discussed in the introduction of the paper is the silicon area. But the sensor is dedicated to embedded devices, hence additional constraints have to be taken into account. Power consumption takes a place of choice among them, and on top of all, we have to perform the global motion estimation in real time.

4.1. Complexity analysis

In the hierarchy of vision sensor design flow, complexity analysis is the first step to optimize the digital hardware required.

Firstly we can point out that the lines number for all matrix involved in our technique can be half the original one, while keeping exactly the same estimation robustness. Indeed, let us suppose that we consider N positions of local motion estimations, the resulting over determined system of equation (1) contains 2N lines, as each position is described in the image plane by two coordinates X and Y. Then if a local motion is erroneous[2], it constitutes a proportion of 2/2N=1/N of the system, which is the same proportion as if we consider only one of the two new coordinates (1/N). Therefore we choose this last solution and we will involve in the global estimation only the coordinate parallel to the considered side of the image periphery. This is the same as computing one-dimensional motion estimation, and the overall processing load in terms of elementary operations number is half of the original one. We have quantified the total number of elementary operations to perform the global motion estimation, leading to 42N+207 operations, where N is the number of local motion considered in the periphery of the image.

Let us now consider the local motion estimation processing load which constitutes the main part of the processing load. Indeed, as described earlier, we compute local movement estimations thanks to matching algorithms, just as the well-used MPEG scheme. Unfortunately, these algorithms lead to highly regular low-level tasks, and a huge amount of data access through frame buffer is also required. In a typical video encoder for example, it accounts for as much as 60% of total CPU cycles [4]. In our case, FSBM and Census algorithms are both quadratic algorithms, respectively equal to $2\times M^2\times S^2\times N$ and $S^2\times N$. Where M×M are blocks of pixels, S is the search area in pixels, and N the number of local movements considered. Let us consider that we perform one local motion measurement each 10 pixels of the SVGA video, that means that N=280. In that case, it accounts for as much as the overall number of operations presented in the following Table 2.

Table 2 : Processing load of local motion estimation in elementary operations, and the associated ratio to the total computational load of the global motion estimation (with N=280 local motions on the periphery).

Search area	FSBM 5×5 ~ $2\times M^2\times S^2\times N$	*Census 5*5* ~ $S^2\times N$
+/-16 pixels	~ 15 232 068 op 99.92 %	~ 4 394 495 op 99.73 %
+/- 12 pixels	~ 8 736 042 op 99.87 %	~ 3 369 087 99.65 %
+/- 8 pixels	~ 4 032 002 op 99.31 %	~ 2 330 367 99.49 %

4.2. Hardware requirements and partitioning

As pointed out in Table *2*, local motion estimations accounts for around as much as 99% of the total processing load required to extract the global motion between two consecutive frames.

[1] source codes are available on the IRISA website: http://www.irisa.fr/Vista/Motion2D/index.html.

[2] or considered as erroneous in the case of a mobile object in the scene.

On the other hand, we can perform in CMOS technology numerous kinds of analog and/or mixed signal processing, as soon as the phototransduction [9], avoiding to perform the computation in the processing stage (Figure 2). This computation saving being then allocated to higher level tasks as image segmentation for tracking for example. This is the processing partitioning that we propose, performing the peripheral local motion measurements at pixel level thanks to dedicated motion detectors, and adding the global motion estimation to the post-processing hardware (Figure 2). As it is discussed in the following section, two proposed vision sensor architectures are considered (Figure 5).

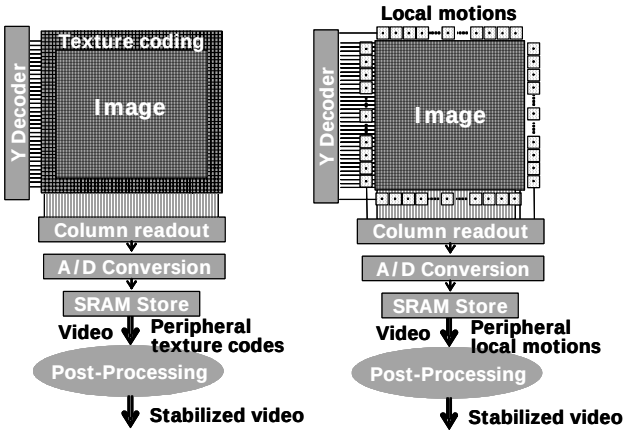


Figure 5: Two proposed architectures of CMOS image sensors.

5. Focal plane local motion measures

Focal plane signal processing is often composed of elementary operations in order to keep pixels simple and preserve the sensor fill-factor while benefiting from the intrinsic massive parallelism of image sensors to obtain powerful computing architecture [10]. It is also possible to design specific architectures dedicated to particular tasks.

Several local motion detectors have been designed for the purpose of motion estimation [11]. Some of them are inspired from biology and constitute silicon models of elementary biological functions. Each of these smart pixels embeds additional electronics (around 20 transistors). This leads to larger silicon area per pixel, hence lower resolution sensors and higher cost compared to common image sensors with 3 transistors per pixel. That is the main reason why these kind of smart pixels are not widely used in industry.

However, fixing our task of global motion estimation considering only the periphery of the image acquisition area avoids this antagonism between pixel level processing and silicon area required. Also, we propose in the following section two kinds of focal plane architecture estimating the desired local motions. In the first solution we integrate a modified version of the census transform (left in Figure 5), and in the second solution we propose the integration of local motion estimators (right in Figure 5).

5.1. Ternary census transform

We firstly consider the silicon integration of the census transform, which has been previously integrated in [12]. Then, starting from these circuit characterizations, we have carried out further validations on the census transform. That brought us to introduce a new version of it, especially dedicated to pixel level processing. Such an image processing approach imposes to work on raw data, without noise reduction (due to silicon area saving). This new census transform is shown to be more robust to fixed pattern noise[3] [13]. The resulting pixel architecture involves a hysteresis comparator with 3 transistors active pixel (Figure 6). Each pixel results in a 10×10µm² area, instead of 4×4µm² for pixels specialized in image acquisition only (3 transistors per pixel). Pixel characteristics are summarized in Table 3. The comparator performs threshold luminance comparisons between pixels in order to get local texture codes in each frame. Then local motions are obtained, in a post-processing stage, by tracking each code from one frame to the other. This implies to consider a searching zone limited by the magnitude of the inter frame displacements we want to measure. Therefore, we need to integrate several lines of ternary census pixels around the image (left in Figure 5).

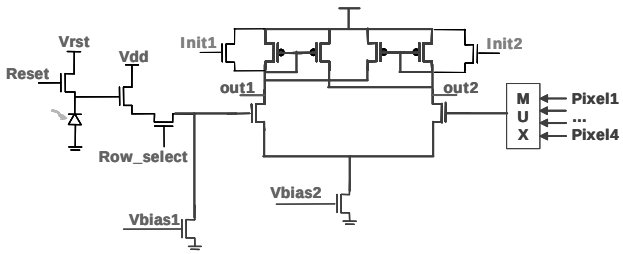


Figure 6: Pixel architecture integrating the ternary census transform.

Table 3 : Pixel characteristics.

CMOS 0.35µm *1 photodiode with 3T APS* *1 comparator* *1 analog MUX 4:1*	*20 transistors*
	Area = 10×10 µm²
	Fill factor = 22%
	Dynamic range = 46 dB

For a SVGA video module for example (sensor size which is currently on the market), the magnitude is about 3% of the image size, which equals about 21 pixels (in terms of image pixels size, i.e. 4×4µm² area). However in terms of ternary census pixels, it is equal to a displacement between 8 and 9 pixels in both directions. Therefore we need to integrate 2*9+1=19 lines of ternary

[3] This noise is due to components mismatches.

pixels, increasing the image area by 22% of the original SVGA image acquisition area. This integration induces 50% saving of the overall computation load to perform global motion estimation (Table 2).

5.2. Local motion detector

As introduced at the beginning of this section, several motion detectors have been designed to measure local motions. We focus currently on the ones described in [14]. The main advantage of such a processing is the continuous time mode of measuring local motion. Indeed it avoids the problem of temporal aliasing [11]. The principle illustrated in Figure 7 is to measure the time Δt of travel of a spatial or temporal feature of the scene (an edge for example) between two photo elements distant of L, resulting in a crossing speed of:

$$V = \frac{L}{\Delta t} \tag{3}$$

This measure is asynchronous, that is why we are currently developing a technique to process a temporal integration over the inter frame period in order to synchronize our data with the video.

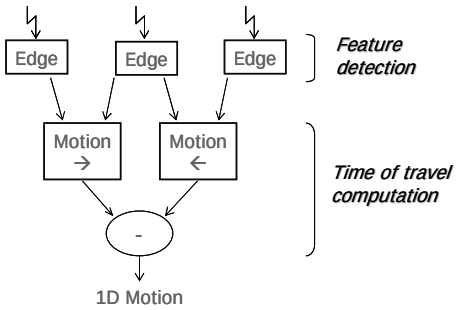


Figure 7 : Local motion measurement.

This elementary motion sensor is integrated in a 30×30µm² pixel. Moreover, thanks to the continuous time processing, only one line of such detectors are necessary, resulting in an increase in silicon area of 5% of the original VGA image acquisition area. These local motion measurements induce around 99% saving of the overall computational load (Table 2).

6. Conclusion

We have presented in this paper a new approach in performing video stabilization. The global motion required to perform this task is extracted from local motion estimations at the periphery of the image acquisition area. We have firstly validated the approach by software implementation. Then we have described the building of a vision sensor able to provide video capture and the associated global motion. The main advantage of the proposed technique, in a vision system architecture point of view, is to perform each task of image acquisition and motion estimation independently. Indeed it avoids the sensible tradeoff between image pixel area and pixel level processing. The proposed integration induces a saving in the processing load required to estimate the global motion of around 99%, with an increase in silicon area of 5% of VGA image acquisition surface.

We are finalizing the integration of the system in a CMOS 0.13µm technology. Our final objective is to embed this vision sensor in micro camera modules for mobile devices such as PDA and mobile phones.

7. References

[1] In-Stat/MDR, "Image Sensors 2004: Camera Phones & Digital Still Cameras Drive Market for CMOS and CCDs", no. IN0401157MI, Sept. 2004.

[2] A. Dutta, "An Image Stabilizer for a Microcamera Module of a Handheld Device and Method for Stabilizing a Microcamera Module of a Handheld Device", U.S. Patent 2003,076,421, Issued April 2003, Nokia Corporation, 2003.

[3] C. Morimotto and R. Chellappa, "Fast Electronic Digital Image Stabilization", *in Proceedings of the 13th Int. Conf. on Pattern Recognition*, Aug. 1996, vol. 3, pp. 284-288.

[4] P. M. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation", *in Proceedings of the Int. Conf on Visual Communications and Image Processing*, San Jose, Jan. 1998, vol. SPIE 3309, pp. 498-509.

[5] N.R. Draper and H. Smith, "*Applied Regression Analysis*", 3rd Ed., John Wiley & Sons, New York, 1998.

[6] R. Zabih and J. Woodfill, "Non-Parametric Local Transforms For Computing Visual Correspondance", *in 3rd European Conference on Computer Vision*, 1994, pp 151-158.

[7] J.M. Odobez, P. Bouthemy, "Robust multiresolution estimation of parametric motion models", *Journal of Visual Communication and Image Representation*, December 1995, 6(4), pp.348-365.

[8] F. Spindler, P. Bouthemy, "Real-time estimation of dominant motion in underwater video images for dynamic positionning", *IEEE Int. Conf. on Robotics and Automation, ICRA'98*, May 1998, Leuven, Belgium, vol. 2, pp. 1063-1068.

[9] C. Mead, "Analog VLSI and neural systems," *Addison-Wesley*, 1989.

[10] G. Linan, S. Espejo, R. Dominguez-Castro and A. Rodriguez-Vazquez, "Architectural and Basic Circuit Considerations for a Flexible 128×128 Mixed-Signal SIMD Vision Chip", *Analog Integrated Circuits and Signal Processing*, *Kluwer Academic Publishers*, 2002, 33, pp. 179-190.

[11] R. Sarpeshkar, J, Kramer, G. Indiveri and C. Koch "Analog VLSI Architectures for Motion Processing: from Fundamental Limits to System Applications", *Proceedings of the IEEE*, Vol. 84, Issue: 7, , July 1996, pp. 969-987.

[12] D. Navarro, G. Cathébras and F. Gensolen, "A block matching approach for movement estimation in a CMOS

retina: principle and results," *proceedings of the European Solid-State Circuits Conference ESSCIRC'03*, 2003, pp. 615-618.

[13] F. Gensolen, G. Cathebras, L. Martin, M. Robert, «Pixel level silicon integration of motion estimation », *IEEE workshop on Design and Diagnostic of Electronic Circuits and System*s, Sopron, Hungary, April 2005, pp. 93-98.

[14] J. Kramer, "Compact integrated motion sensor with three-pixel interaction", in IEEE trans. On Pattern Analysis and Machine Intelligence, April 1996, vol. 18, Issue 4, pp. 455-460.