A 64 × 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization

Bo Zhao, *Student Member, IEEE,* Xiangyu Zhang, *Student Member, IEEE,* Shoushun Chen, *Member, IEEE,* Kay-Soon Low, *Senior Member, IEEE,* and Hualiang Zhuang

***Abstract*—This paper presents a 64×64 CMOS image sensor with on-chip moving object detection and localization capability. Pixel-level storage elements (capacitors) enable the sensor to simultaneously output two consecutive frames, with temporal differences digitalized into binary events by a global differentiator. An on-chip, hardware-implemented, clustering-based algorithm processes events on the fly and localizes up to three moving objects in the scene. The sensor can automatically switch to region of interest mode and capture a picture of the object. The proposed image sensor was implemented using UMC 0.18 μm CMOS technology with a die area of 1.5 mm × 1.5 mm, and power consumption was only 0.4 mW at 100 FPS.**

***Index Terms*—Motion detection, moving object localization, on-the-fly clustering, smart image sensor.**

I. Introduction

CAMERA systems that localize and track fast-motion objects are important to a number of applications, including surveillance, security monitoring, and road traffic enforcement. Despite aggressive advancements in CMOS technology, existing implementations of these systems remain bulky [1]–[4]. In addition to complex signal processing algorithms, these systems usually need high-speed imaging to avoid motion blur and multiple sensors to cover a large surveillance area from different view angles. Cameras that have no or few feature extraction capabilities produce large amounts of unimportant data that must be read and processed to obtain features of interest [5]. Thus, huge computation resources are demanded for real-time operation, and this consequently can only be done on high-performance computing platforms. Translation of these systems into low-cost and lightweight platforms is therefore a challenging task.

Smart image sensors combine focal-plane signal processing and implement novel approaches to improve the computation efficiency, when compared to conventional discrete sensor-processor systems. Among these are various image sensors for motion detection, adaptive resolution, and even object tracking [6]–[15]. An architecture of an object tracking CMOS image sensor was presented in [9]. The sensor can switch between two operation modes: object acquisition and tracking. It first finds the N most salient targets in the field of view and defines windows around their centroid coordinates, then only the regions inside the windows are processed. A spatial-temporal multi-resolution CMOS image sensor was reported in [15]. This image sensor can simultaneously generate two outputs: one at a low frame rate with maximum spatial resolution for stationary backgrounds and the other at a high frame rate with reduced spatial resolution for moving objects in the region-of-interest (ROI). Based on 1-bit motion information, the centroid of the moving object is computed by an external CPLD device and sent back to the sensor. The row decoder and the column selector later only access ROI pixels to track the object. The overall system requires two external CPLDs and the algorithm suffers from background noise and cannot adaptively change the size of the region of interest. A more advanced object tracking algorithm was proposed in [16] and [17]. This tracking system was developed using an asynchronous, temporal-contrast, address-event-representation-type vision sensor [14] and an off-chip digital signal processor. The algorithm continuously clusters incoming events based on a mean-shift approach. Each new event is assigned to a cluster based on a distance criterion, and the event is used to update this cluster's weight and position. Though this system removes the need for a buffer to record every motion event, computation tends to be extensive due to the long list of clusters to be maintained and updated for every event. This is caused by the random spatial and temporal distribution of motion events out of the vision sensor, in which the pixels belonging to one object arrive in nonconsecutive order. An individual motion pixel cannot easily be treated as "noise" or "a part of an interested object" without waiting a significant amount of time to see whether it is surrounded by enough nearby neighbors.

In this paper, we propose an image sensor with on-chip moving object detection and localization. The algorithm was implemented *on-chip* without the need for any external computation and storage. This image sensor performs frame differencing [18] to generate binary motion events, which are processed on the fly to build clusters based on a distance criterion. After the position and size of the active object is obtained, the sensor switches to ROI intensity mode and reports a picture of the object. The aim of this paper is to develop a real-time tracking system for low-power applications. Our major

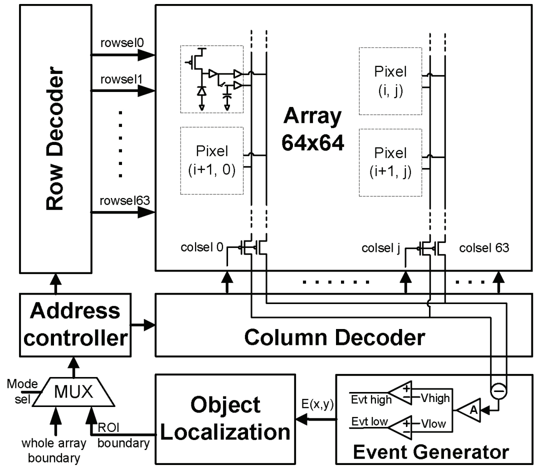


Fig. 1. Block diagram of the single-chip image sensor with moving objects detection and localization functionality. The event generator digitalizes temporal differences into sequential motion events, which are processed by the moving object localization unit on the fly to find the boundary addresses of the moving object region. With these addresses, the address controller enables the image sensor to extract the image in the ROI.

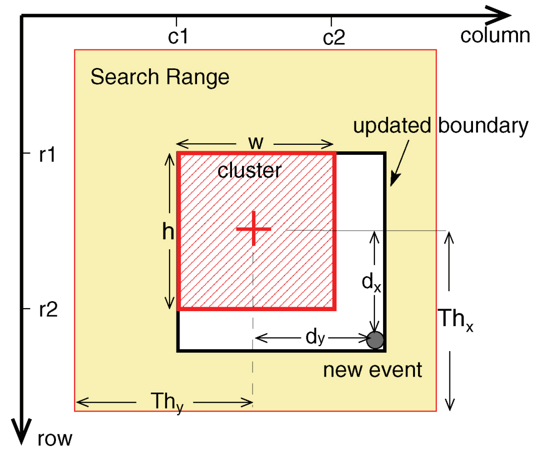


Fig. 2. Updating an existing cluster upon the arrival of a new active motion event (pixel value equals "1"). $(r1, c1)$ and $(r2, c2)$ illustrate the cluster's original boundary, w and h are its width and height, and Th_x and Th_y define the search range of the cluster. d_x and d_y show the row distance and column distance between the cluster center (red cross) and the incoming event, respectively. If the new event resides within the search range of the cluster, then it is defined as belonging to that cluster, and the cluster's information will be updated correspondingly.

contribution resides in two areas: 1) an efficient clustering-based object localization algorithm; and 2) single-chip fusion of motion detection and object localization.

The rest of this paper is organized as follows. In Section II, we introduce the system architecture. Section III describes the moving object localization algorithm. The very large scale integration (VLSI) implementation of motion detection and object localization is illustrated in Section IV. Section V reports the experimental results and Section VI concludes this paper.

II. System Architecture

Fig. 1 shows the system architecture of the proposed image sensor. Main building blocks include a 64×64 pixel array, temporal difference event generator, object localization unit, address controller, and decoders. Each pixel is equipped with an analog memory (capacitor) and can output both the new integration voltage on its photodiode and the previous voltage stored on its capacitor. The event generator computes the difference between the two voltages and compares it to a positive and negative threshold. A motion event is generated if this difference exceeds the thresholds. If the scene illumination and object reflectance are constant, the changes in scene reflectance only result from object movements or camera motion. The background information is thus filtered by the camera, avoiding the background subtraction computation needed with systems that employ standard intensity cameras [19], [20]. The motion events are processed on the fly to build clusters based on a distance criterion. At the end of the frame, the position and size of the active object is obtained. The sensor can automatically switch to ROI intensity mode and capture a picture of the object.

III. Object Localization Algorithm

Integration of a signal processing algorithm onto silicon involves trading off between performance and complexity. Increased complexity usually translates into higher system cost and power consumption. On the other hand, an over-simplified algorithm will downgrade system performance and lose ground for practical usage. On the basis of these considerations, we proposed an efficient object localization algorithm and seamlessly integrated it into the above-mentioned motion detection sensor. The key processing element is a so-called "cluster," which is a block of pixels belonging to the same object. As illustrated in Fig. 2, a cluster is visually represented as a rectangle shape and is uniquely defined by its boundary coordinates $(r1, c1), (r2, c2)$. Besides these boundary coordinates, parameters of a cluster also include number of events (NoE). By trading off between performance and implementation complexity (explained in Section III-B), we employ three clusters and the chip is therefore able to track up to three objects at the same time.

A. Clustering of Motion Events

The algorithm is implemented in an on-the-fly fashion. It continuously monitors the incoming pixel data (motion event), in which "1" stands for a pixel on a motion object (active motion event) and "0" for a still background pixel (inactive motion event). For the sake of clarity, the "event" mentioned in the following part always refers to the active motion event.

At the beginning of each frame, all clusters are reset (i.e., number of events of each cluster is cleared to 0). A cluster is initiated upon the arrival of the first active motion event. Later on, the algorithm examines the distance of a new event to the existing clusters (nonempty) based on the following criterion:

$$d_x < Th_x \quad \text{and} \quad d_y < Th_y \tag{1}$$

where d_x and d_y are the row distance and column distance between the event and the cluster center, respectively. Th_x and Th_y define a *search range* (the largest rectangular area in Fig. 2) with respect to the center of the existing cluster.

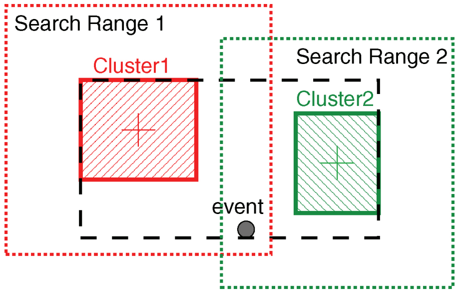


Fig. 3. Illustration of instant merging. If the incoming event belongs to more than one cluster at the same time, then all corresponding clusters and current event will be merged together.

Th_x and Th_y are not constants; they instead grow with the cluster. Let w and h be the width and height of the cluster, then Th_x and Th_y can be described as

$$\begin{aligned} Th_x &= Th0 + h/2 \\ Th_y &= Th0 + w/2 \end{aligned} \tag{2}$$

where $Th0$ is a user-defined *threshold*, which stands for the distance from the boundary of the cluster to the boundary of search range.

The clusters will then be updated according to the following procedure.

1) If the event falls within the boundary of an existing cluster, the cluster simply increases its number of events by one. If the event falls out of the cluster boundary but is still within the search range, it is still considered to be part of this cluster, and the cluster therefore grows its boundary to enclose the event. This procedure is illustrated in Fig. 2.
2) If the incoming event is out of the search range, a new cluster needs to be initiated which is centered at the address of the event.
3) In case the event belongs to more than one cluster at the same time, these clusters are merged into a single larger cluster. As illustrated in Fig. 3, when the search ranges of cluster 1 and cluster 2 overlap and an event occurs in the common region, we merge the two clusters into a single one and store the updated information into cluster 1. We name this strategy "instant merging." In this way, the resource of cluster 2 can be reused, effectively reducing required total number of clusters.
4) If the event belongs to none of the existing clusters, a new cluster needs to be initiated. Due to limited resources, there is a chance that all clusters are deployed. In this case, a discarding strategy is adopted. The cluster containing the least number of events is considered a noise object and discarded. At the same time, it is re-initiated according to the address of the event.

Fig. 4 shows the intermediate clusters during the sequential scanning of a binary image, which models the output data stream from the image sensor. It illustrates the above-mentioned clustering process, including initiation of a new cluster, cluster growing, merging, and discarding.

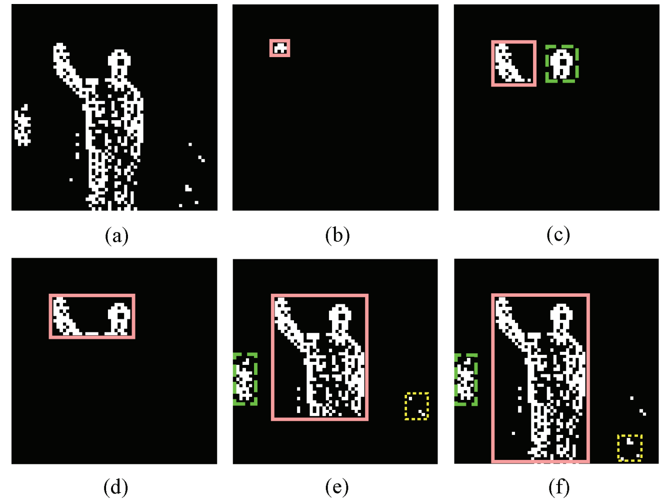


Fig. 4. Evolution of the intermediate clusters during the sequential scanning of a temporal difference image. (a) Test image which models the output data stream from the sensor. (b) First motion event initiates cluster 1 (solid box). (c) As more events are received and processed, cluster 1 is enlarged, and cluster 2 is initiated (dashed box). (d) Instant merging: a new event belongs to both clusters 1 and 2, and the two clusters are therefore considered to be parts of one object and merged. (e), (f) Discarding strategy. A new event belongs to none of the existing clusters and all cluster resources are taken, then the cluster containing the least number of events (cluster 3, in this example, shown as a dotted box) is considered a noise object and discarded. At the same time, this cluster is re-initiated at the address of the event.

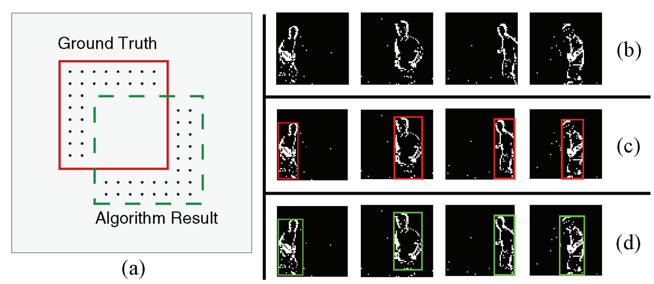


Fig. 5. (a) Figure that is used to illustrate the definition of localization error. Dotted regions represent the nonoverlapping area. (b) Lists some sample test images. (c) and (d) present corresponding ground truth and simulation results (at certain $Th0$ and number of clusters), respectively.

B. Selection of Optimal Parameters

In order to evaluate the proposed algorithm and select optimal parameters (search range threshold $Th0$ and number of clusters), we have defined a quantitative criterion. As illustrated in Fig. 5(a), for a test image with a resolution of 64×64, the boundary of the largest object in the scene is manually marked as the reference ground truth, and is compared to the boundary computed by the algorithm (at certain $Th0$ and number of clusters). The localization error is defined as follows:

$$err = N_d/N_t \tag{3}$$

where N_d is the number of pixels in the nonoverlapping area of the two bounding boxes, and N_t is the number of pixels in the ground truth box.

We have built a library of 120 motion images, including various scenarios such as road traffic, pedestrians, and laboratory

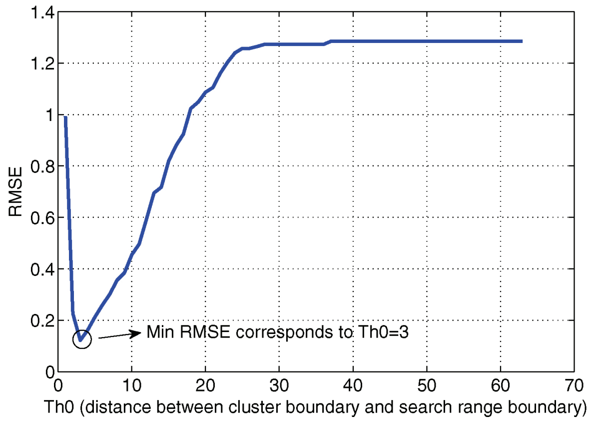


Fig. 6. RMSE versus Th0 for a test image sequence (given three clusters). The optimal Th0 is 3.

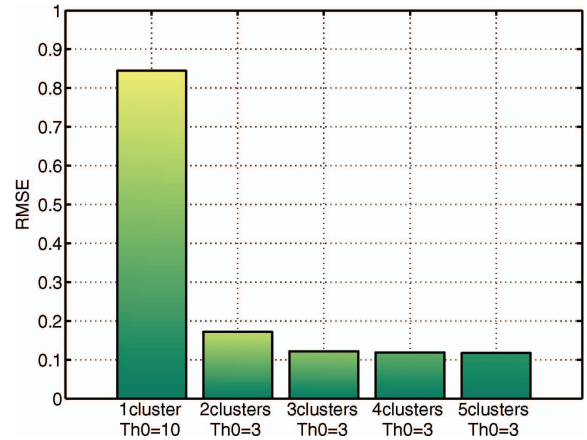


Fig. 7. RMSE versus number of clusters (at corresponding best threshold $Th0$). Three clusters are enough for the given test image library.

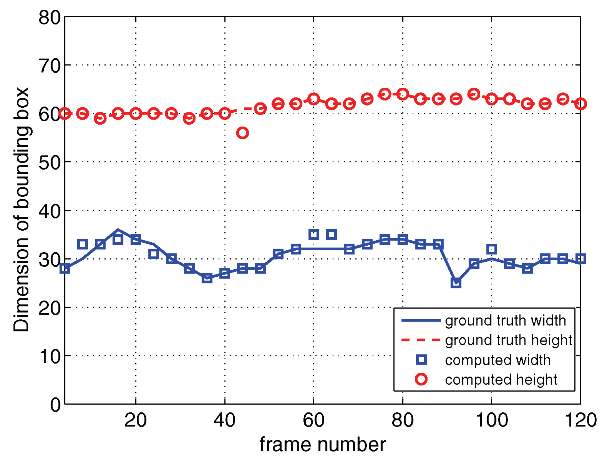


Fig. 8. Comparison of the width and height of the computed bounding box to the ground truth box for the test library.

activities. A few examples are shown in Fig. 5(b). Based on the test image library, we first evaluated the choice of search range threshold $Th0$. Given a fixed number of clusters (e.g., $1, 2, 3, \cdots$), for a certain threshold, $Th0 = k,\ \ 1 \leq k < 64$, each image of the test sequence produces an error ($err_i,\ i = 1, 2, ..., N, N = 120$), and root mean square error (RMSE) is calculated based on the following performance metric:

$$RMSE_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N} err_i^2}. \tag{4}$$

On one hand, a smaller threshold (corresponding to a conservative boundary expansion rate) leads to better noise rejection and allows production of a "cleaner" bounding box for an object. On the other hand, a larger threshold (corresponding to an aggressive boundary expansion rate) is required to merge discontinuous regions of one object, which are commonly found in temporal difference images due to the nonuniform motion speed of parts belonging to one object. With the given library, we tried different numbers of clusters (1 to 5); and for each number, we sweep the search range threshold Th0 to find an optimal value, based on the minimum RMSE rule. When using only one cluster, the optimal threshold Th0 is found at 10; while for two to five clusters, the optimal thresholds are all coincidentally equal to 3. Fig. 6 shows the simulation result for three clusters. The best clustering performance (i.e., the minimum RMSE) for each cluster number is reported in Fig. 7. More clusters undoubtedly improve performance; however, the improvement almost saturates when cluster number is more than three. In addition, seen from the description of the algorithm, larger number of clusters will involve more operations such as distance measurement and comparison, and hence end up with more hardware resources. With the above considerations, we employ three clusters in the system.

With the obtained parameters (i.e., three clusters and $Th0 = 3$), we execute the algorithm on a short motion video sequence. For each frame, the computed bounding box is compared with a manually marked ground truth. The simulation result is shown in Fig. 8.

C. Performance Under Different Noise Conditions

In order to quantitatively analyze the influence of noise on the algorithm's performance, we added salt and pepper noise to all the library images. We tried different noise densities, ranging from 0.01 to 0.10, with a step length of 0.01. The RMSE for each noise density was calculated and shown in Fig. 9. As expected, the RMSE gradually increases with the increase of noise density. Fig. 10 shows samples of localization results under different noise conditions. It demonstrates the robustness of our algorithm against low to medium level noise interference.

IV. VLSI Implementation

A. Motion Detection

The image sensor consists of a 64×64 pixel array, and each pixel is equipped with an analog memory (capacitor). The whole array is hence capable of storing the previous frame as a reference image. The rows are first sequentially selected for reset. Later, at another round of scanning, the rows of pixels are sequentially selected for readout. Each pixel will output both the new integration voltage on its photodiode and the

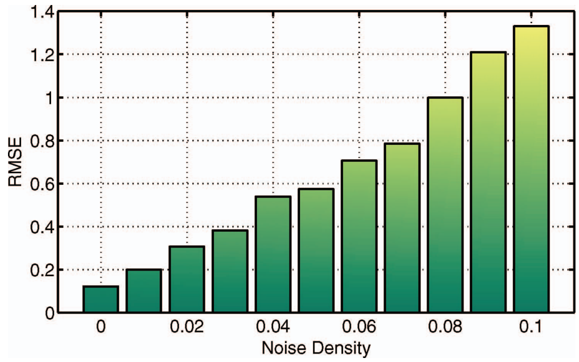


Fig. 9. RMSE versus noise density. (Algorithm parameters: three clusters, search range threshold Th0 equals 3.)

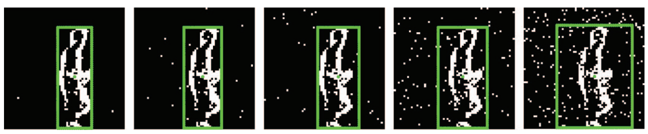


Fig. 10. Localization results of an example image under different noise conditions. The first one is the result before adding noise. The others are results under different levels of salt and pepper noise interference (from left to right, noise densities are 0.01, 0.02, 0.05, and 0.10, respectively).

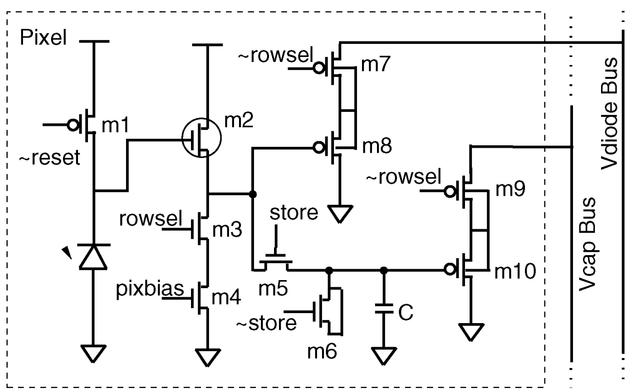


Fig. 11. Schematic of the pixel. Low threshold NMOS follower m2 is highlighted with a circle. Capacitor C is the memory device that stores the previous-frame pixel value.

previous voltage stored on its capacitor. The two voltages are fed into a global event generator circuit which is composed of a global amplifier with a temporal-difference computation circuit based on dual comparison [18]. The event generator computes the difference between the two voltages and compares it to a positive and negative threshold. A digital motion event is generated if this difference exceeds the thresholds.

Fig. 11 shows the schematic of the pixel. Building blocks include a photo sensing element (photodiode), reset transistor (m1), an internal unity gain buffer made of a NMOS source follower (m2–m4), a sample-and-hold circuit (m5, m6, C) for storing the previous photo sensing voltage, and two sets of PMOS source follower readout paths (m7, m8 and m9, m10). In order to address the body effect and improve linearity, PMOS source follower transistors are designed in dedicated Nwells. This comes at the cost of a minor increase in silicon area.

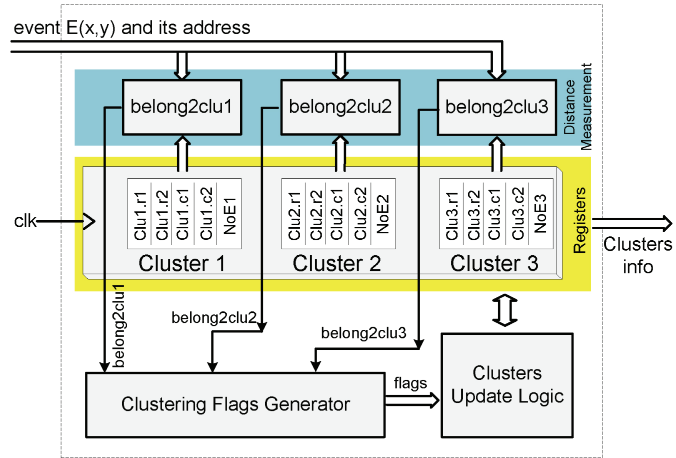


Fig. 12. Block diagram of the object localization unit.

B. Object Localization

Fig. 12 shows the block diagram of the object localization unit. Major building blocks include three sets of registers for storing clusters' information and a batch of computational logics, including Distance Measurement Unit ("belong2clu1," "belong2clu2," and "belong2clu3"), "Clustering Flags Generator," and "Clusters Update Logic."

1) *Distance Measurement:* The motion event and its address are sent in parallel to the distance measurement unit. Based on the criterion described in (1), three bits of comparison results are produced, namely "belong2clu1," "belong2clu2," and "belong2clu3," respectively.

For instance, signal "belong2clu1" indicates whether the new event falls within the search range of cluster 1. Let (x, y), (x_c, y_c), $(r1, c1)$, and $(r2, c2)$ denote the addresses of the motion event, cluster-1's center, top left corner, and bottom right corner, respectively; then the distance between the event and the cluster center, the search range, and the final comparison results are computed by

$$\begin{aligned}
&h = r2 - r1\\
&w = c2 - c1\\
&Th_x = Th0 + h/2\\
&Th_y = Th0 + w/2\\
&x_c = (r1 + r2)/2\\
&y_c = (c1 + c2)/2\\
&d_x = abs(x - x_c)\\
&d_y = abs(y - y_c)\\
&belong2clu1 = (d_x < Th_x)\&\&(d_y < Th_y).
\end{aligned} \tag{5}$$

Therefore, the computational core for measuring the distance to each cluster consists of ten adders. Fig. 13 shows the implementation of "$a < b$" and "$abs(a - b)$" operation.

2) *Clustering Flags Generator:* Seen from the description of the algorithm, one event may simultaneously belong to multiple clusters. The three bits of comparison results are fed to "Clustering Flags Generator" to make further judgements.

For instance, the combination of 'b100 ("belong2clu1" = 1, "belong2clu2" = 0, "belong2clu3" = 0) indicates that the motion event only belongs to cluster-1 and hence cluster-1 should be enlarged. As shown in Fig. 14, this can be simply implemented in hardware by a "AND3" gate, which outputs

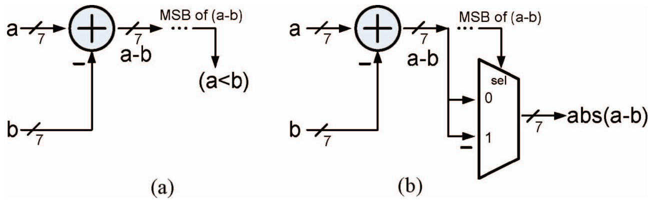


Fig. 13. Arithmetic operations for $(a < b)$ and $abs(a - b)$. (a) $(a < b)$. (b) $abs(a - b)$.

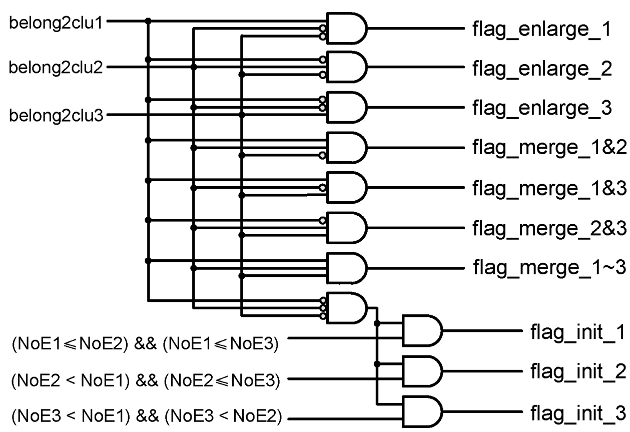


Fig. 14. Generation of the clustering flags. The 3 bits of comparison results ("belong2clu1," "belong2clu2," and "belong2clu3"), together with the information of NoE in each cluster, generate ten clustering flags.

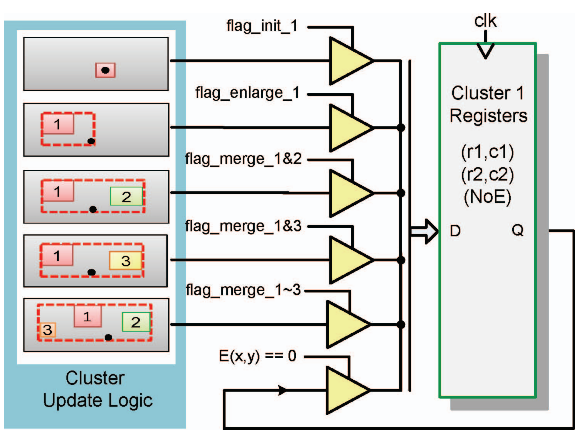


Fig. 15. Update of cluster-1 registers. Clustering flags control a series of tri-state buffers to choose the right boundary update results.

a flag signal, denoted as "flag_enlarge_1." Similarly, the sequence of 'b111 asserts a signal namely "flag_merge_1∼3." In particular, 'b000 means the event belongs to none of the clusters. In this case, the cluster containing the least number of events will be re-initialized.

3) *Clusters Update Logic:* The aforementioned flags will determine the clusters' update operations, i.e., initiation, growing, merging, and so on. In hardware (shown in Fig. 15), a batch of dedicated arithmetic blocks first compute every possible future cluster information (i.e., boundary address and number of events). The clustering flags then control a series of tri-state buffers to select one out of the future values.

The computational core in each arithmetic block only consists of adders. For instance, enlarging cluster-1 involves the following arithmetic operations:

$$\begin{aligned} r1' &= \min(r1, x)\\ r2' &= \max(r2, x)\\ c1' &= \min(c1, y)\\ c2' &= \max(c2, y)\\ NoE1' &= NoE1 + 1 \end{aligned} \tag{6}$$

where $(r1', c1')$, $(r2', c2')$, and $NoE1'$ represent the updated boundary and number of events.

4) *Discussion:* We examined the implementation complexity of the localization block with respect to scaled number of clusters.

a) Storage of the clusters (i.e., flip-flops) will linearly increase. Each cluster records its two corner addresses $(r1, c1)$ and $(r2, c2)$, and NoE. This needs 41 bits of flip-flops in total.
b) Since a new event has to check whether it belongs to each cluster, the complexity of the distance measurement unit (i.e., +, −, and < operations) will linearly increase.
c) More inter-cluster operations are needed. For instance, when there are three clusters, an incoming event will check whether it belongs to only one of them or simultaneously two of them, or even three of them. These operations are translated to "AND" gates (as shown in Fig. 14). For N clusters, the total number of "AND" operations is $C(N, 0)+C(N, 1)+C(N, 2)+...+C(N, N) = 2^N$. Moreover, in order to find out which cluster has the least number of events, another $2N$ "AND" gates and $C(N, 2)$ comparators are required.

In summary, building larger number of clusters on-chip allows us to track more objects and offers better noise rejection performance; however, this is at the expense of more hardware resources.

V. Experimental Results

The single-chip smart vision sensor consisting of a temporal difference imager and an object localization unit was implemented using UMC 0.18 μm CMOS process (one poly and six metal layers). Fig. 16 shows the chip microphotograph with main building blocks highlighted. The chip has a total area of 1.5 mm × 1.5 mm (inclusive of I/O pads). The 64×64 pixel array, motion event generator, and row and column decoders were implemented using a full custom approach. Each pixel features an area of 14×14 μm^2 with a fill-factor of 32%. The object localization unit was designed from register transfer level using Verilog HDL, and it was implemented as synchronous digital circuits using standard cells. It occupies a relatively small silicon area of 600×220 μm^2. Guard rings were extensively used to limit substrate coupling and shield the pixels from the outer-array digital circuitry. Power and ground buses were routed using top layer metal.

In order to test the chip, we developed a field-programmable gate array (FPGA) based testing platform as shown in Fig. 17. The vision sensor is interfaced with an Opal-Kelly XEM 3005 FPGA board. The FPGA is configured to provide input control signals (clock, reset and switch between analog/motion mode), temporarily store the cluster data to an

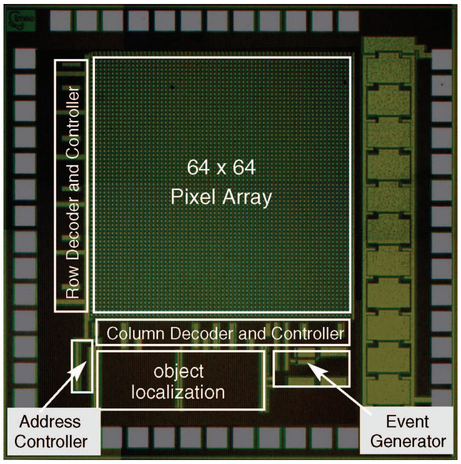


Fig. 16. Chip microphotograph with main building blocks highlighted.

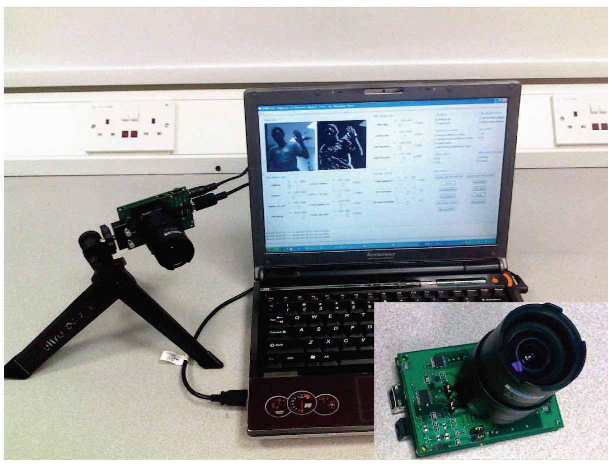


Fig. 17. FPGA based testing platform. The vision sensor is interfaced with an Opal-Kelly XEM 3005 FPGA board.

on-board SDRAM, and communicate with a PC through a high-speed USB link. On the PC side, a graphic user interface is developed which translates operational parameters such as frame rate and motion sensitivity into FPGA signals. At the end of each frame, the three clusters are read out sequentially. Based on the features of object size, position, and motion density (which can be derived from the cluster size and number of events), external processors can effectively track the object of interest. The sensor can automatically switch to ROI intensity mode, and the on-chip address unit will only read out the ROI pixels. An on-board 12-bit ADC (AD7476) is used for analog image conversion.

Fig. 18 reports a few sample images of road traffic. The demo video can be accessed from our lab website [21]. In this example, we assume the largest object in the scene is the target of interest, and one can note that the sensor can locate and extract the running person, motorcycle, and car quite well. During data acquisition, the sensor was mounted still on the second floor and faces the road with an angle of $\pm 45°$. Due to the limited resolution of 64×64 pixels, the image quality of the ROI is not very satisfying when the object goes far. With the successful proof of this concept sensor, we plan to adopt a multi-resolution strategy [15] in the future. The proposed

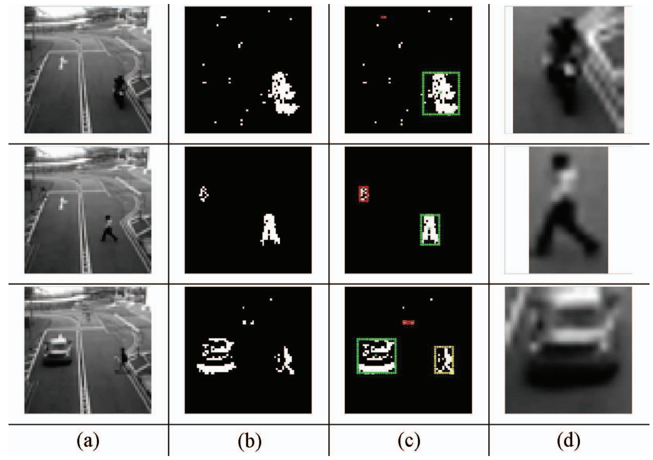


Fig. 18. Sample images of the prototype image sensor. (a) Gray level images captured in normal intensity mode. (b) Corresponding images captured in temporal difference mode. (c) Localization results on top of the temporal difference images. (d) Snapshot of the ROI (images are resized in software for better visualization).

TABLE I

CHIP CHARACTERISTICS

Process Technology	UMC 0.18 μm 1P6M CMOS, 1.8 v Supply
Die size	1.5×1.5 mm^2
Pixel array	64×64
Pixel size	14×14 μm^2
Number of trans/pixel	10
Fill factor	32%
Readout strategy	Sequential scan
Fixed pattern noise	0.4%
Dark current	6.7 fA
Sensitivity	0.11 v/(lux·s)
Max frame rate	100 fps
Power consumption	Pixels array + motion detection 0.4 mW, object localization 8.63 μW (at 100 fps)

clustering algorithm works at low resolution and high frame rate. Once the object of interest is determined, the sensor can switch to higher resolution and only reports intensity image of the ROI.

Other characteristic parameters of the chip are summarized in Table I. In particular, the sensor features low power consumption. The analog part only dissipates 0.4 mW, and the clustering part consumes 8.63 μW when operating at 100 f/s.

VI. CONCLUSION

This paper reported the theory, simulation, VLSI design, and experimental measurements of a single-chip CMOS image sensor with moving object detection and localization capability. Motion events are first detected using a frame differencing scheme; then they are processed by an on-the-fly clustering processor to localize the motion objects in the scene. Unlike existing systems relying on external FPGAs or CPLDs to perform object localization, our system does not require any external computation or storage. The proposed algorithm is integrated on chip, featuring compact silicon implementation and little power consumption. The proposed design is an ideal candidate of wireless sensor network node, for applications such as assisted living monitors, security cameras, and even robotic vision. Future improvements include adoption of a dynamic resolution pixel array and event based object tracking.

REFERENCES

[1] C.-H. Chen, Y. Yao, D. Page, B. Abidi, A. Koschan, and M. Abidi, "Heterogeneous fusion of omnidirectional and PTZ cameras for multiple object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1052–1063, Aug. 2008.

[2] A. Blumberg, L. Keeler, and A. Shelat, "Automated traffic enforcement which respects 'driver privacy'," in *Proc. IEEE ITSC*, Sep. 2005, pp. 941–946.

[3] Y. Cho, S. O. Lim, and H. S. Yang, "Collaborative occupancy reasoning in visual sensor network for scalable smart video surveillance," *IEEE Trans. Consumer Electron.*, vol. 56, no. 3, pp. 1997–2003, Aug. 2010.

[4] Y.-J. Wu, F.-L. Lian, and T.-H. Chang, "Traffic monitoring and vehicle tracking using roadside cameras," in *Proc. IEEE Int. Conf. SMC*, vol. 6. Oct. 2006, pp. 4631–4636.

[5] H. Zhuang, K.-S. Low, and W.-Y. Yau, "On-chip pulse based parallel neural circuits for object-tracking system," *IET Electron. Lett.*, vol. 46, no. 9, pp. 614–616, Apr. 2010.

[6] M. Gottardi, N. Massari, and S. Jawed, "A 100 μW 128 $\times$ 64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *IEEE J. Solid-State Circuits*, vol. 44, no. 5, pp. 1582–1592, May 2009.

[7] E. Artyomov and O. Yadid-Pecht, "Adaptive multiple-resolution CMOS active pixel sensor," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 53, no. 10, pp. 2178–2186, Oct. 2006.

[8] A. Teman, S. Fisher, L. Sudakov, A. Fish, and O. Yadid-Pecht, "Autonomous CMOS image sensor for real time target detection and tracking," in *Proc. IEEE ISCAS*, May 2008, pp. 2138–2141.

[9] A. Fish, L. Sudakov-Boresha, and O. Yadid-Pecht, "Low-power tracking image sensor based on biological models of attention," *Int. J. Inform. Theory Applicat.*, vol. 14, no. 2, pp. 103–114, 2006.

[10] S. Chen, F. Boussaid, and A. Bermak, "Robust intermediate read-out for deep submicron technology CMOS image sensors," *IEEE Sensors J.*, vol. 8, no. 3, pp. 286–294, Mar. 2008.

[11] S. Chen, A. Bermak, and Y. Wang, "A CMOS image sensor with on-chip image compression based on predictive boundary adaptation and memoryless QTD algorithm," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 4, pp. 538–547, Apr. 2011.

[12] Y. Chi, U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings, "CMOS camera with in-pixel temporal change detection and ADC," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2187–2196, Oct. 2007.

[13] S. Mizuno, K. Fujita, H. Yamamoto, N. Mukozaka, and H. Toyoda, "A 256 $\times$ 256 compact CMOS image sensor with on-chip motion detection function," *IEEE J. Solid-State Circuits*, vol. 38, no. 6, pp. 1072–1075, Jun. 2003.

[14] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[15] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon, "A spatial-temporal multiresolution CMOS image sensor with adaptive frame rates for tracking the moving objects in region-of-interest and suppressing motion blur," *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, Dec. 2007.

[16] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schon, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Proc. Digit. Signal Process. Workshop*, Sep. 2006, pp. 173–178.

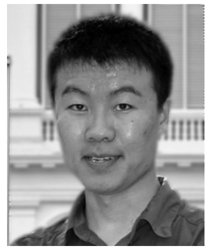
[17] M. Litzenberger, A. Belbachir, N. Donath, G. Gritsch, H. Garn, B. Kohn, C. Posch, and S. Schraml, "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *Proc. IEEE ITSC*, Sep. 2006, pp. 653–658.

[18] S. Chen, W. Tang, and E. Culurciello, "A 64 $\times$ 64 pixels UWB wireless temporal-difference digital image sensor," in *Proc. IEEE ISCAS*, Jun. 2010, pp. 1404–1407.

[19] H. Su and F.-G. Huang, "Human gait recognition based on motion analysis," in *Proc. Int. Conf. Mach. Learning Cybern.*, vol. 7. Aug. 2005, pp. 4464–4468.

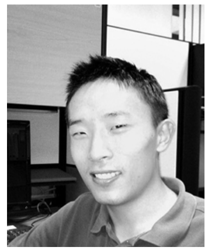
[20] J. Triesch and C. von der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 12, pp. 1449–1453, Dec. 2001.

[21] B. Zhao. (2011, Jan.). *Demo Video* [Online]. Available: http://www3.ntu.edu.sg/home2009/zhao0130/1/demo.htm



Bo Zhao (S'11) received the B.S. and M.S. degrees in electronic engineering from Beijing Jiaotong University, Beijing, China, in 2007 and 2009, respectively. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His current research interests include smart vision sensors and energy-efficient algorithms for object recognition.



Xiangyu Zhang (S'11) received the B.S. degree in electronic engineering from Southeast University, Nanjing, China, in 2009. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His current research interests include mixed-signal integrated circuit design for smart image sensors.

Shoushun Chen (M'05) received the B.S. degree from Peking University, Beijing, China, the M.E. degree from the Chinese Academy of Sciences, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, in 2000, 2003, and 2007, respectively.

He was a Post-Doctoral Research Fellow with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology for one year after graduation. From February 2008 to May 2009, he was a Post-Doctoral Research Associate with the Department of Electrical Engineering, Yale University, New Haven, CT. In July 2009, he joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as an Assistant Professor. His current research interests include mixed signal integrated circuits design for sensors, feature extracting biomimetic sensors for sensor networks, energy-efficient algorithms for object recognition, smart vision sensors, asynchronous very large scale integration circuits, and systems.

Dr. Chen serves as a Technical Committee Member of Sensory Systems, IEEE Circuits and Systems Society, an Associate Editor of the *Sensors Journal*, and the Program Director (Smart Sensors) of VIRTUS, IC Design Center of Excellence.

Kay-Soon Low (M'88–SM'00) received the B.Eng. degree in electrical engineering from the National University of Singapore, Singapore, and the Ph.D. degree in electrical engineering from the University of New South Wales, Sydney, Australia.

He joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 1994 as a Lecturer and subsequently became an Associate Professor. He has served as a consultant to many companies and has a number of granted patents on nonlinear circuits and ultrawideband systems. His funded projects are in the field of wearable wireless sensor networks, solar energy systems, pulse neural networks, and satellite systems. Currently, he is the Center Director of the Satellite Research Center, Nanyang Technological University.

Hualiang Zhuang received the B.Eng. degree in electrical engineering from Zhejiang University, Zhejiang, China, and the M.Eng. degree in electrical engineering from Nanyang Technological University, Singapore. He is currently pursuing the Ph.D. degree in electrical engineering with Nanyang Technological University.

He is currently a Research Associate with Nanyang Technological University. His current research interests include process control and neural networks.