Real-Time Median Filtering for Embedded Smart Cameras

Yong Zhao
Brown University
Yong_Zhao@brown.edu

Gabriel Taubin
Brown University
taubin@brown.edu

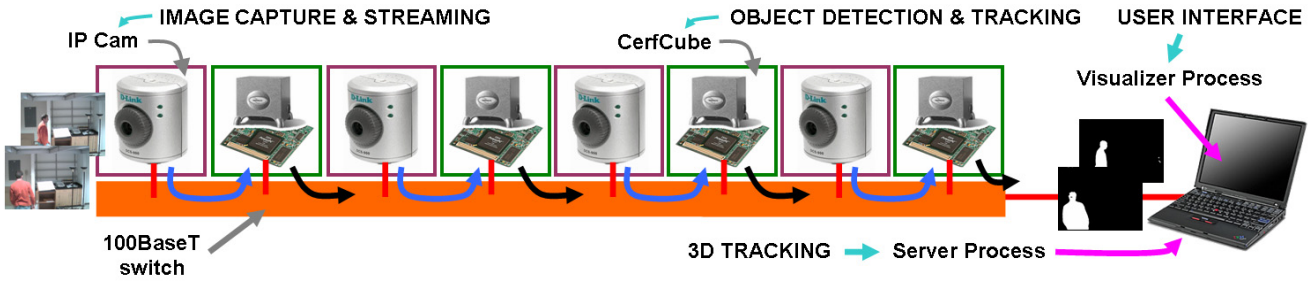

Figure 1: System architecture of our proof-of-concept Visual Sensor Network (VSN) built with off-the-shelf IP cameras and embedded single-board-computers (SBCs) as image processors which communicate over common ethernet switch fabric amongst themselves and with the server. Each SBC serves as image processor for one IP camera. The algorithm described in this paper runs in the SBCs as part of an indoors person detection and tracking application. Results presented include the overhead incurred by each SBC decoding JPEG frames sent by a corresponding IP camera.

Abstract

This paper describes a new median filter algorithm optimized for real-time performance in smart cameras with embedded processors. As in the JPEG and MPEG compression algorithms, each frame of the video stream is first partitioned into a regular array of non-overlapping square blocks. The median value for each block is then computed and compared with corresponding values of neighboring blocks. If the magnitude of the difference does not exceed a threshold, the output value for all the pixels in the block is set to the median value. Otherwise, the output value for each pixel in the block is computed as the median value within a window of the same size centered at this pixel. We describe variations for binary and grayscale images. The algorithm has been implemented and tested in an embedded single-board-computer (SBC) with no hardware acceleration, as a component of a Visual Sensor Network (VSN) system for real-time indoor person detection and tracking. In this system, where the SBCs have the additional overhead of decoding JPEG frames from IP cameras, our new algorithm is 5 to 20 times faster than the traditional algorithms for typical window sizes. We expect further speed-ups to frame-rate performance on smart cameras with embedded image sensors and reconfigurable hardware.

1 Introduction

The two dimensional median filter has been extensively used for smoothing operations in image processing since its introduction by Tukey [11]. The result of applying the median filter to an $N \times N$ image I using a $W \times W$ window, where $W = 2w+1$ is an odd number, is a new image M of the same size[9]. The output pixel value $M[i,j]$ is computed as the median of all the input values within the $W \times W$ window centered at the pixel $I[i,j]$:

$$M[i,j] = \text{median}\{I[a,b] \, : |a-i| \leq w \wedge |b-j| \leq w\} \, .$$

Computing this value requires sorting the input pixel intensity values within the window (the filter kernel), and this has to be done for each pixel in the image, resulting in high computational cost. Compared with a linear averaging filter, which evenly diffuses impulsive noise to neighboring pixels, the median filter removes impulsive noise by ignoring it. Consequently, median filtering is hardly affected by impulsive noise smaller than the filter kernel.

The algorithm described in this paper was developed and implemented as a basic low-level operation for the proof-of-concept Visual Sensor Network (VSN) described in Figure 1. This platform was used to implement a real-time indoor person detection and tracking application. Our long term goal is to build large scale VSNs for real-time operations with very large number of cameras (1000s). For this kind of scalability, bandwidth constraints requires the cameras to

Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS 2006)
0-7695-2506-7/06 $20.00 © 2006 IEEE

be *smart*, i.e., to host the most high data rate intensive image processing operations. Figure 7 shows the architecture and current state of our first smart camera design, which is not yet operational. The experimental results reported in this paper are based on the same SBC platform, but the image capture is simulated using off-the-shelf IP cameras, which imposes on the SBCs the overhead of communicating with their corresponding camera over the same network used to communicate with the server, and decoding compressed JPEG frames before applying the new median filter algorithm. All of this overhead is included in the results.

The paper is organized as follows. In Section 2 we describe the new fast median filter algorithm in detail, first for binary images, and then for graylevel images. In Section 3 we provide brief descriptions of prior related work, and comparisons with our new approach. In Section 4 we describe the platform where we performed our experiments. In Section 5 we describe our experimental results. Finally, in section 6 we end the paper with conclusive remarks and summary.

2 The Algorithm

We first explain the new algorithm for binary images, and then show how it has to be modified for graylevel images.

2.1 Fast Binary Median Filter

Binary image smoothing is used widely in many applications. For example, in most object tracking applications foreground objects are detected by comparing each frame with a background model. Each pixel in the image is classified as foreground or background. Various kinds of background statistical models have been proposed because plain differencing is not robust enough. However, noise can not be avoided mainly due to variations in illumination or background/foreground characteristics, etc. Therefore, for most approaches the resulting images need to be smoothed prior to further processing.

Figure 2 shows an example of using our new fast median filter algorithm to smooth a binary image in a person tracking application. Image (a) is a frame captured from a video camera, showing that a man is walking in the room. Image (b) is the binary image obtained by comparing (a) with the background model. The value of each pixel (black or white) indicates whether it belongs to the background or the foreground. As can be seen, there is too much noise and too many fuzzy edges in this image. In order to remove this ”salt and pepper” noise and smooth the edges of the foreground region, we apply median filtering on this image. Image (c) shows the output of the traditional median filter.

Our algorithm first divides the binary image (b) into a regular grid of non-overlapping square blocks the size of the kernel window of median filter. Image (d) shows the divided image (for illustration purposes the blocks shown in image (d) are much bigger than the ones actually used). The median value of each block is then computed, and assigned to all the pixels in the block. The result of this step is shown in image (e). That is, the algorithm evaluates the traditional median filtering only at the center points of each block and these values are replicated to the rest of the pixels in the

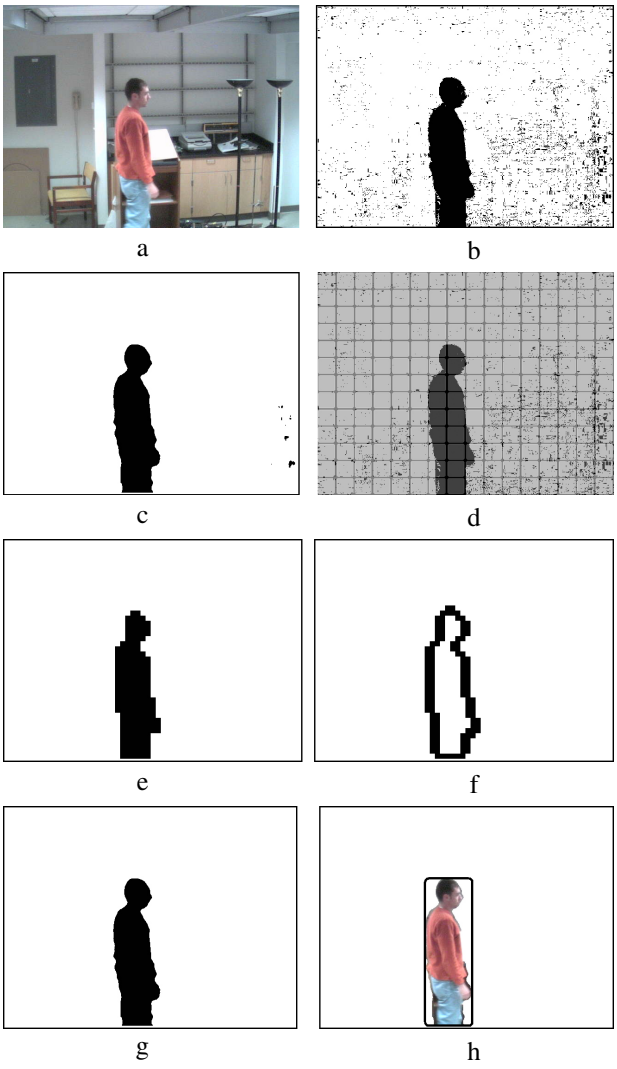

Figure 2: Fast binary median filter. (a) Original image captured by the sensor; (b) Noisy foreground segmentation obtained by comparing the input image with the background model; (c) Binary image smoothed with traditional median filter. (d) Noisy image partitioned into non-overlapping blocks (big blocks shown for illustration purposes; actual blocks are smaller); (e) Low resolution foreground image generated by *block approximation*; (f) Detected *edge blocks*; (g) Binary image smoothed with the algorithm described in this paper; (h) Object of interest segmented out from input image.

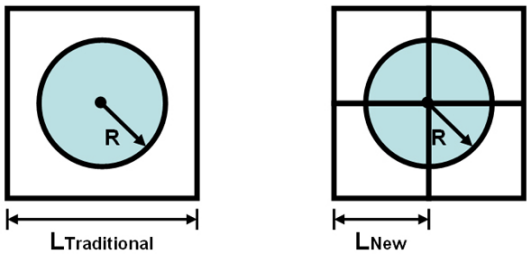

Figure 3: Different kernel size is needed to remove the same size noise by traditional median filter and new approach.

block. As can be seen in image (e), the noise is successfully removed by this step, but details on the region edges are lost. In the following step the algorithm recovers the lost details.

To decide which blocks require additional boundary details, the median value assigned to each block in image (e) is compared with the corresponding values assigned to the eight-connected neighboring blocks. If not all of these vales coincide, the central block is labeled as an *edge block*. Edge blocks are blocks which contain both background and foreground pixels (one of them dominates the whole block after the approximation). Image (f) shows the result of this step. After all the edge blocks are found, the traditional median filter algorithm is evaluated on each pixels of image (b) belonging to an edge block, and the corresponding output pixel is set to the computed median value. This means that overall, the median value is computed on all the block centers, and on all the edge block pixels. Image (g) shows the final result of fast noise removal and edge smoothing, and image (h) shows the result of segmenting image (a) according to the foreground classification of image (g).

2.2 Algorithm Analysis

Comparing images (c) and (g) in Figure 2, we find that the noise removal and edge smoothing quality of the new approach are better than those obtained using the traditional medial filter algorithm with the same filter kernel size. This phenomenon can be easily explained by the following example:

As illustrated in Figure 3-(a), in order to entirely remove a round-shaped noise disk of radius R, the minimum size of kernel window needed is $L_{\text{Traditional}} \times L_{\text{Traditional}}$, where $L_{\text{Traditional}} = (2\pi R^2)^{1/2} \approx 2.506\,R$, in the traditional median filter algorithm. While for the new approach, the minimum filter kernel size depends on the position of the noise disk. The best case is when the center of the noise disk is at the vertex of blocks. In this case $L_{\text{new}} = (0.25\pi R^2)^{1/2} \approx 1.253\,R$. The worse case is when the center of the noise disk is at the center of block. In this case $L_{\text{new}} = L_{\text{Traditional}}$. In general, in new approach a smaller filter kernel size is needed to remove the noise features of same dimension. In other words, with the same filter kernel size, the the new approach has stronger image smoothing power than the traditional algorithm.

W(pixels)	3	5	7	9	11	13	15
σ (%)	0.4	1.8	2.7	3.2	3.6	4.5	6.3
θ (%)	11.5	5.8	4.7	4.4	4.4	5.1	6.7

Table 1: Computational cost comparison of two median filters.

Our primary concern for real-time embedded applications is to reduce the computational cost. The computational cost in the traditional median filter algorithm is determined by the image size and kernel window size. For example, applying median filtering with a $W \times W$ window to a $N \times N$ digital image involves N^2 median value calculations; each median value calculation involves data sorting for W^2 values. The temporal complexity of the traditional median filtering: $T_{\text{Traditional}} = O(N^2 W \log(W))$.

The computational cost of the new median filter algorithm with the same kernel window size on the same image is much lower because we only evaluate the median values on the center pixels of every block, and on all the pixels in edge blocks. The total temporal complexity of new median filtering is:

$$T_{\text{New}} = O\left(\left(\frac{N^2}{W^2} + A_{\text{edge\_blocks}}\right) W \log(W)\right)$$

Where $A_{\text{edge\_blocks}}$ denotes the area of all the edge blocks in pixels.

In order to show how fast the new approach is, we use θ to denote the computational cost ratio of the two median filters:

$$\theta = \frac{T_{\text{New}}}{T_{\text{Traditional}}} = \frac{1}{W^2} + \frac{A_{\text{edge\_blocks}}}{N^2} = \frac{1}{W^2} + \frac{\sigma}{100}$$

Where σ denotes the percentage of edge blocks area in the whole image area. The value σ depends on the noise level, size and shape of foreground region, and the size of kernel window.

We should notice that for binary image, the data sorting computation is much cheaper than that of intensity image case. And the temporal complexity is no longer $O(n \log(n))$. But the computational cost ratio θ of two median filters is the same. For the image showed in Figure 2, the value σ and the corresponding θ for different S are displayed in Table 1.

We see that when $W = 5$, the area of "edge blocks" is just about 1.8% of the whole image area for the case showed in Figure 3. Therefore the computational cost of new approach is expected to be only about 5.8% of that of traditional one.

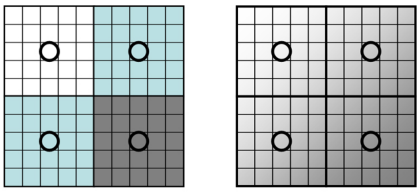

Figure 4: Example of block approximation. (a). Simple approximation of four adjacent 5×5 blocks. In each block, median values of only the central pixels (marked by the circle) are computed and shared by all non-central pixels. (b). Smoother approximation using bilinear interpolation algorithm with the computed median value for central pixels.

2.3 Fast Intensity Median Filter

The basic idea of the new median filter described for the binary case could be easily extended to the graylevel image case. The only difference is at *block approximation* step.

In the binary case we simply approximate the median filter algorithm results of non-center pixels within each block, with the values assigned to the center pixel of the same block.

In the graylevel image case this simple approximation does not work well because it produces block boundary artifacts. This problem is illustrated in Figure 4-(a). In order to get a smoother approximation, we smoothly interpolate values from median values of adjacent blocks, as shown in Figure 4-(b). We have tried some commonly-used continuous interpolation approaches, such as nearest neighbor, bilinear, spline, cubic and sinc interpolation. The smoothness performances produced by these approaches do not differ too much, but their computational costs vary dramatically. Because what we need is a fast approximation to avoid computing the median value on every pixel, we have chosen bilinear interpolation as our *block approximation* step.

We then use a scheme similar to the one used in the binary case. Median values of adjacent blocks are compared, and the magnitudes of the differences are thresholded to detect the *edge blocks*. For edge blocks, the median value is computed for every pixel in the block. For the other blocks, the bilinear interpolation is used to set the output pixels. The results from two steps are combined and then we have the final result of well-smoothed image.

Figure 5 illustrates the steps of our fast median filter for graylevel images. 25% *salt and pepper* noise is added to input image (a), resulting in image (b). Image (c) shows the output from the traditional median filter with a 3×3 window. Image (d) shows the *edge blocks* detected in (b). Image (e) is the output of our fast median filter with a 3×3 window. When the noise level is increased to 40% in image (f), the traditional algorithm with a 5×5 window outputs image (g), and our new algorithm outputs image (h).

The example presented in Figure 5 shows that, in addition to the significant speedup, the new median filter algorithm produces very good quality output images in the case of intensity image. With the same size of filter kernel, the new approach delivers better noise removal quality than the traditional median filter algorithm, especially when the noise is dense. Additionally, in the above example, the computational cost ratio θ (defined in Section 3.2) is about 15%, which means that the new approach is expected to be at least 6 times computationally more efficient that the tradition algorithm.

3 Related Work

Since its introduction, median filtering has been extensively used to remove impulsive *salt and pepper* noise from images. It is a more robust method than the traditional linear filter because it preserves sharp edges.

One of the major flaws of the median filter is that it is very computationally expensive when the filter kernel is big. The temporal complexity of a straightforward serial computation of applying median filtering on a $N \times N$ digital image with $W \times W$ kernel is $O(N^2W^2)$. This can reduced to $O(N^2W \log(W))$ by Quicksort [1]. A fast two-dimensional median filtering algorithm developed by Huang, Yang and Tang [4] provides a $O(N^2W)$ performance. A separable median filter introduced by Narendra [8] and latter improved by Basu and Brown [2] takes only $O(N^2 \log W)$ time. Various kinds of hardware solutions using configurable logic or a pyramid computer have been introduced to achieve real-time performance [10, 3, 7]. Dedicated hardware does provide high speed but it is too expensive and rarely available in most application systems.

Note that all these previous works have concentrated on how to accelerate the computing of the median number for every pixel. The focus of the approach presented in this paper is on how to decrease the number of median number computations and still achieve equally good smoothing results. As mentioned in section 2.2, for most images only 0.4 6.3% the median number computations performed by other algorithms are evaluated by our algorithm. Our approach is complementary those listed above, and we plan to incorporate some of these ideas in future versions of our algorithm to achieve further speedups.

4 Platform

As mentioned in Section 1, and illustrated in Figure 1, we constructed a VSN for real-time indoor people detection and tracking. In this first system We used CerfCube405EP SBCs powered by IBM PowerPC System-on-Chip (SoC) processors [6] as image processors, and D-Link DCS-900

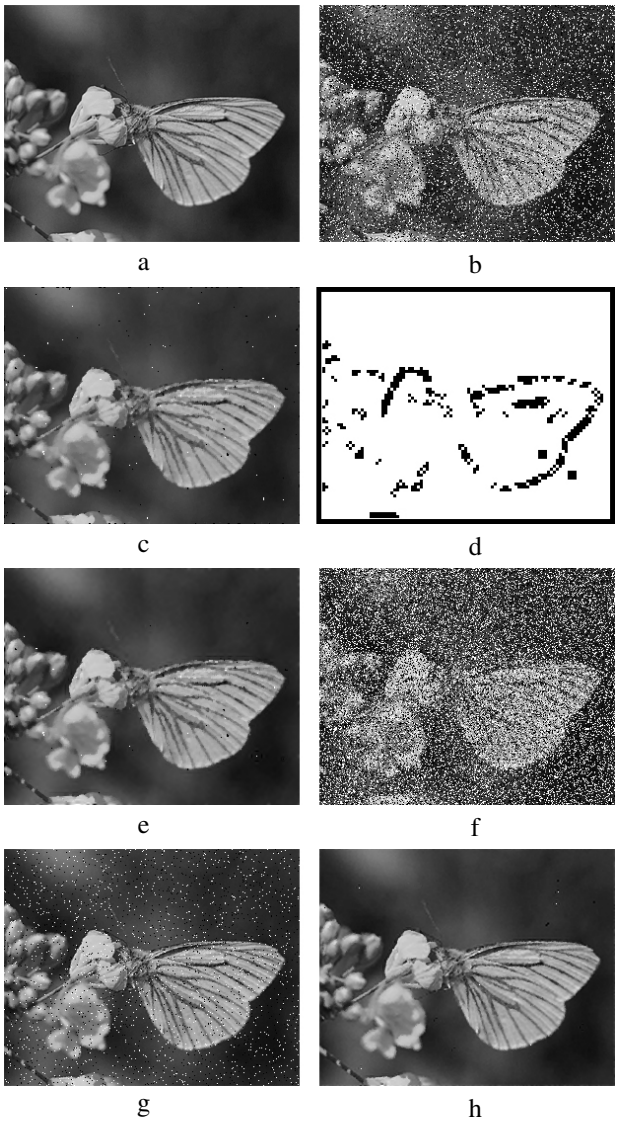

Figure 5: Example of fast intensity median filtering. (a). Original grayscale image. (b). Image imposed with 25% impulsive noise. (c). Smoothed image by traditional image. (d) "Edge blocks" detected in new approach; (e) Smoothed image by new approach; (f) Image imposed with 40% impulsive noise. (g). Smoothed image of (f) by traditional image; (h) Smoothed image of (f) by new approach.

IP cameras for image capture [5]. The common 100BaseT Ethernet switch fabric existing in the building was used for communication between each camera and its corresponding image processing SBC, and between each SBC and the server. This approach reduces the bandwidth requirements, distributes the computation, and allows the server to concentrate on other jobs, such as calibration, synchronization, information integration and visualization. Performing all the computation at the server would have been impossible

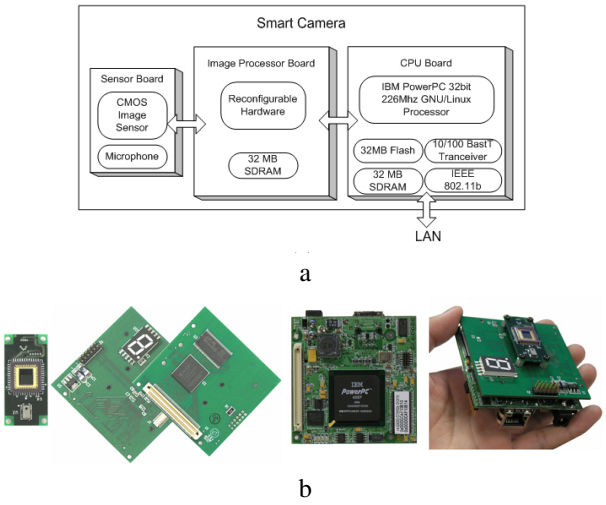

Figure 6: (a). Architecture of embedded smart camera; (b). Picture of our current camera hardware. Each smart camera consists of 3 functional PCB boards: a sensor board, a reconfigurable hardware image processor board and a CPU board.

with this architecture, both in terms of computation and communications. With this structure only small amounts of data generated by the embedded image processors need be transferred over the network to the server. It is then possible to construct a real-time, large-scale visual sensor network with the compact, power-saving embedded cameras, and one regular PC working as the server.

Figure 6 illustrates the smart camera we will use in our next VSN system. The high level system architecture is showed in (a), and pictures of the three circuit which constitute this smart camera are shown in (b). This embedded camera has been designed and is currently under assembly and debugging. In order to develop and test the image processing software, such as the algorithm presented in this paper, we implemented the image capture device driver for the SBC emulating the hardware platform that we are building. A mass-market IP camera was used to capture images, which were transmitted to the SBC over the Ethernet interface. Once a frame is received, decoded, and in local memory, the SBC runs the image processing operations at the same speed it will on our future smart camera. The only difference is that the raw data in JPEG format is transferred from the sensors to the image processors by Ethernet instead of inter-PCBs connections, and therefore the image processors has the overhead of communication and decompressing the JPEG data data before processing it.

5 Experimental Results

The main purpose of our new median filter algorithm is to overcome the high computational cost of the traditional ap-

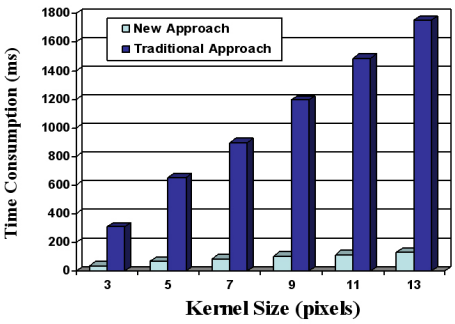

Figure 7: Time consumption of two approaches in binary case on embedded system.

Kernel Size	3	5	7	9	11
New Approach	35	47	72	149	189
Traditional Approach	323	862	1614	3426	4195

Table 2: Time consumption of two approaches in intensity case (unit: ms).

proach. In previous sections we demonstrated that the quality of the images produced with the new approach is equal or better than with of traditional approach, especially when the noise level is high. The algorithm analysis in provides reasons for the improvement of smoothing quality and reduction of computational cost. In this section, we present speed test results for the new algorithm running in the VSN system described above. We first tested the speed performance of traditional and new median filters for binary images. The median filtering works as foreground smoother in this application. The time costs of the traditional median filter vs. the new algorithm are displayed in Figure 7.

On our current emulating embedded cameras, the objects detection and tracking application runs at 8 to 10 frames per second in QVGA mode, which is an encouraging result. We expect a much higher frame rate, probably 30 fps, when we remove the overhead associated with interfacing with the IP cameras over the common network, and run the algorithm in our new embedded cameras after the assembly and debugging work is finished. This is so because in that case, the video data will be transferred from the image sensor to the processor core through a private dedicated inter-PCB connections instead of the Ethernet, and no decompression by the image processor will be required.

We also tested the fast intensity median filtering on some 1600×1200 resolution images on a regular PC running at 3.2 GHz. The average comparison results are listed in Table 2.

6 Conclusion and Discussion

In this paper we propose a new high-speed median filtering for real-time embedded applications. The main feature of this new approach is its much lower computational cost. In addition, it provides better smoothing quality than the traditional approach.

We discussed how to apply the new approach both to binary and graylevel images, and we get very good experimental results. Further investigation could be done in how to apply the same idea on color images, which is a vector median filtering problem. However, the higher discrimination of color information might generate too high δ value (*edge block* area ratio) to achieve satisfactory speed performance.

7 Acknowledgements

To be written after the review period.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[2] A. Basu and M. Brown. Algorithms and hardware for efficient image smoothing. *Computer Vision, Graphics and Image Processing*, 40:131–146, February 1987.

[3] J. G. R. Delva, A. M. Reza, and R. D. Turney. Fpga implementation of a nonlinear two dimensional fuzzy filter. In *Proceedings of the International Conference on ASSP*, Phoenix, Arizona, March 1999.

[4] T. S. Huang, G.Y. Yang, and G. Y. Tang. A fast two dimensional median filtering algorithm. In *IEEE Transaction on ASSP*, volume ASSP-27, pages 13–18, February 1979.

[5] D-Link Inc. Dcs-900 ip camera. `http://www.dlink.com`.

[6] Intrinsyc Inc. Cerfcube 405ep embedded single board computer. `http://www.intrinsyc.com`.

[7] G. Louverdis, I. Andreadis, and A. Gasterato. A new content based median filter. In *EUSIPCO 2004*, Vienna, Austria, September 2004.

[8] P.M. Narendra. A separable median filter for image noise smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):20–29, 1981.

[9] S. Ranka and S. Sahni. Efficient serial and parallel algorithms for median filtering. In *Proceedings of the International Conference on Parallel Processing*, pages 56–62, 1989.

[10] S. L. Tanimoto. Algorithms for median filtering of images on a pyramid machine. In M.J.B. Duff, editor, *Computing Structures for Image Processing*. Academic Press, London, 1983.

[11] J.W. Tukey. *Exploratory Data Analysis (preliminary ed.)*. Reading, MA: Addison-Wesley, 1971.