Two Parallel Strategies for Real-time Spatial Video Denoising for Multi-core Processors

Banpot Dolwithayakul
Department of Computing,
Faculty of Science
Silpakorn University,
Nakhon-Pathom, Thailand

Chantana Chantrapornchai
Department of Computing,
Faculty of Science
Silpakorn University
Nakhon-Pathom, Thailand

Noppadol Chumchob
Department of Mathematics,
Faculty of Science
Silpakorn University, Nakhon-Pathom Thailand and
Centre of Excellence in Mathematics, CHE, Si Ayutthaya Rd., Bangkok, Thailand, 10400

ABSTRACT
Video denoising is usually a time consuming process especially for large video files. With the advancement of the processor technology, it is possible to perform video denoising in real-time on multi-core processors. In this paper, we study parallel techniques for denoising real-time video on multi-core processor which work on both shared memory model and distributed memory model. We investigate two approaches: a block approach, which assigns a group of threads to each block of video frames; and a distributor approach, which uses one thread to distribute the frame data to each thread. Our experiments focus on the image denoising technique based on the total variation but the approach can be integrated with other image denoising algorithm like discrete wavelet transform (DWT) or diffusion technique. We found that by using the distributor strategy, we can achieve speedup which is 1.27 times faster than the block strategy and the video frame rate can be increased by 7.43%. Moreover, we also apply the prefetching technique which further enhance frame rate by 22.02% and frame rate control to stabilize frame rate and retain the original video length during denoising and playing in real-time. Our method also has good denoised quality which is better than previous work in [1] in average case.

General Terms
Algorithms, Framework, Image Processing, Video Processing, High Performance Computing, Parallel Computing

Keywords
video denoising, parallel computing, OpenMP, ROF model, total variation.

1. INTRODUCTION
In many real world applications, images and videos tend to pose some noises from sensors during retrieval, encoding and transmission. Image denoising, which involves eliminating noises to recover the original digital image, thus, plays a very important role for restoring or improve degraded images or video quality.

It is commonly known that denoising an image is an ill-conditioned problem. As such, it becomes necessary to impose a constraint on the solution via an appropriate regularization for penalizing unwanted and irregular solutions using some prior knowledge, such as smoothness of boundaries while preserving important details like edge and texture.

However, most of denoising techniques are time consuming. A lot of memory and computation power are required. Applying image denoising techniques with video sequences will be even worse since a lot of image frames needs to be processed in real-time applications like in medical equipments, multimedia live streaming etc. Thus, it will be very hard for a single processor to denoise a video in a satisfactory frame rate.

The presence of the mentioned constraint is the primary reason why variational methods are more appropriate and successful compared to other techniques for removing additive noise while preserving edge of the images. Particularly, the first total variation (TV)-based image denoising model by Rudin, Osher, and Fatemi [2] (also known as the ROF model) is among the most famous ones to offer superior image restoration quality. The ROF model uses several solving techniques, one of which is the lagged-diffusion fixed point iterative method[3,4] which solves this problem effectively and it is scalable by the number of fixed point iterations. With this method, it is possible to denoise an image without obtaining the optimal result but having the acceptable restored image quality which is suitable for real-time video denoising efficiently.

In this paper, we study proper parallel strategies for real-time spatial video denoising as a framework which is compatible with many image denoising algorithms for example total variation(TV), diffusion method or discrete wavelet transform(DWT). We select the total variation with a fixed point iterative method as a testing algorithm which works very good on the additive noise model. Refer to [5] and references therein for a review of some video denoising techniques.

The rest of this paper will be organized as follows: Section 2 presents some mathematic backgrounds related to video denoising that we use and related work in the field. We present our parallel methods in Section 3. Finally, Section 4 demonstrates the efficiency of our methods.

2. BACKGROUNDS
This section consists of two parts. The first part discusses video denoising categories and challenges in the video denoising and the second part reviews the ROF model.

2.1 Video Denoising Methods

On this research, we choose to use the spatial video denoising technique which tends to be highly parallelizable and suitable for real-time denoising on the streaming video.

Thus, it is an easy implementation to begin with and can be expanded from the image denoising [5].

In the real-time parallel computation, the following challenges are commonly encountered

2.1.1 Real-time streaming data

By working with streaming and real-time data, we have to minimize the latency from starting of the input and starting of the output. By fetching a lot of frames and store to the buffers first is not a good idea because a lot of latency will be occurred. All incoming frame need to be processed almost immediately, making the fetching any n^{th} frame from the streaming data impossible. Thus, all frames must be fetched sequentially.

2.1.2 Order of frames

When distributing each frame to different threads, the order of the frame must be retained after the computation of each thread is done.

2.1.3 Frame rate

The real time video denoising must meet the satisfactory frame rate which is normally higher than 15 frames per second. Normally, the human visual system can process 10-12 separate images per second[6]. Thus with the good frame rate, the latency of the video will not be noticed.

2.1.4 Thread utilization

Every available thread has to work efficiently by minimizing the idle threads during computation.

In this paper, we are going to design a strategy that will utilize the threads efficiently while minimizing the idle time.

2.1.5 Frame rate control

The output video has to have the same length of the original video. Users should feel no latency, delay or slow frames delivery.

2.2 ROF Model

Let $u:\Omega\subset\Re^2\to V\subset\Re^{\geq 0}=\Re\cup\{0\}$ be the image to be recovered (unknown) and $z:\Omega\to V$ be the noisy image (known), where Ω is an image domain. Assume that noises in each frame is additive and zero-mean Gaussian type. The ROF model can be represented as follows:

$$\min_u\{J(u)=\tfrac{1}{2}\int_\Omega (u-z)^2 d\Omega+\alpha TV(u)\}, \qquad (1)$$

where the integral term is the fitting or data term and

$$TV[u]=\int_\Omega |\nabla u|_\beta \, d\Omega=\int_\Omega \sqrt{u_{\mathbf{x}}^2+u_{\mathbf{y}}^2+\beta}\, d\Omega \qquad (2)$$

represents the TV regularization, $\alpha>0$ is the regularization parameter, and $\beta>0$ is a small constant to avoid the division by zero, typically β=10^{-4}. Without loss of generality we assume that $\Omega=[0,1]\times[0,1]$, $V=[0,255]$ and u_x and u_y are the partial derivative respect to x- and y-directions, respectively.

According to the calculus of variations, the minimizer of the energy functional J in Equation (1) satisfies the Euler-Lagrange equation given by the following second order and nonlinear partial differential equation (PDE)

$$-\alpha\nabla\cdot\left(\frac{\nabla u}{|\nabla u|_\beta}\right)+u=z \qquad (3)$$

subject to the homogeneous Neumann's boundary conditions

$$\partial_n u=0 \qquad (4)$$

where n is the outward unit vector normal to the image boundary $\partial\Omega$.

To solve Equation (3), an iterative algorithm is unavoidable. On a multi-core processor, performing such algorithm requires many iteration loops and is, therefore, a time-consuming task. This drawback delays the video denoising process. Generally, the execution time can be reduced in two ways: (1) by reducing the number of iterations or reducing the time that is needed per iteration. (2) by utilizing each thread effieiently. Multi-core processors are well suited for acceleration of the video denoising time.

To this end, we follow closely our previous work in [5] for solving the discrete system of Equation (3) resulting from the so-called cell-centerd finite difference discretization. This previous work uses a coupled outer-inner iteration method, also known as lagged diffusivity fixed point approach, with the inner (linear) solver provided by an efficient parallel Gauss-Seidel (GS) method in reducing the run time used per an outer iteration.

2.3 Lagged-diffusion fixed point method

We use homogeneous Neumann boundary conditions as follows:

$$u_{i,1}=u_{i,2}, u_{i,n}=u_{i,n-1}, u_{1,j}=u_{2,j}, u_{n,j}=u_{n-1,j} \qquad (5)$$

There are several methods for solving Equation (3), for example, the time marching scheme which uses a time variable *t* to transform Equation (3) into the parabolic PDE. The time marching scheme is simple to compute in each iteration. However, convergence of this method is not fast enough to use practically. Next is the Primal-Dual [7] method which use dual matrix to solve Equation (3). However, this method consumes too much memory for video in practice because of the usage of additional matrices. Another method is called *fixed-point method* which is usually a faster solver. The fixed point method is originally proposed by Vogel and Oman[13, 14] for the total variation denoising problem which can be written as follows:

$$\tfrac{\alpha}{h^2}((\textstyle\sum_{\alpha}^{[\upsilon]})_{i,j}(u_{(1)}^{[\upsilon+1]})_{i,j}-(\overline{\textstyle\sum}_{(1)}^{[\upsilon]})_{i,j}(u_{(1)}^{[\upsilon+1]})_{i,j})=(z)_{i,j} \qquad (6)$$

where

$$\begin{aligned}(\textstyle\sum)_{i,j}(u)_{i,j}&=(2D[u]_{i,j}+D[u]_{i-1,j}+D[u]_{i,j-1})(u)_{i,j},\\ (\overline{\textstyle\sum})_{i,j}(u)_{i,j}&=D[u]_{i,j}((u)_{i+1,j}+(u)_{i,j+1})\\ &\quad+D[u]_{i-1,j}(u)_{i-1,j}+D[u]_{i,j-1}(u)_{i,j-1}\end{aligned} \qquad (7)$$

and

$$(D[u])_{i,j} = \frac{1}{|\nabla u|_{\beta}} \quad (8)$$

In Equation (6) we can also write in the matrix-vector form as

Equation ($\mathbf{N}[\mathbf{u}^{[\upsilon]}]\mathbf{u}^{[\upsilon+1]} = \mathbf{z}$ (9)

We freeze the coefficient globally by keeping every point unchanged across all *k* inner iteration to solve the linear system to fix υ. For each outer step, the linear system solver like Jacobi, Gauss-Seidel, SOR including modern linear system solver like preconditioned conjugate gradient (PCG) can be used to solve the inner step.

From our previous experiments, we found that convergence of Jacobi method is too slow[8]. For SOR and PCG, they are computationally more expensive than the Gauss-Seidel method. Thus, Gauss-Seidel is a proper method to obtain satisfactory convergence.

2.4 Related Work

In this section, we present some literature reviews as they are related to our work.

Video denoising methods can be divided in three categories as follows:

2.4.1 *Spatial video denoising* method. It aims to individually remove noises in each frame.

2.4.2 *Temporal video denoising* method. It aims to find the dependency of noises between frames and then remove noises between frames.

2.4.3 *Spatial-Temporal video denoising* method. It is a combination of spatial and temporal video denoising methods.

There are some researches in all types of video denoising methods especially spatial video denoising and spatiotemporal video denoising. For the example in [1], A. Sarhan introduced spatial real-time video denoising model with two types of DWT algorithm. The proposed model is adaptive scheme for denoising noisy video which contaminated by AWGN. The author implemented two algorithms, 2D Discrete Wavelet Transform(2D DWT) 2D Dual Tree Complex Wavelet Transform(2D DTCWT) in [1]. where the second one is more efficient for removing noise but it is slow to compute. The author used the noise level estimator to estimate the noise on each frame and decided the best algorithm to denoise that frame. This model improved computation time up to 75.57% in color video compared with using DTCWT algorithm only while retain the good average PSNR result. However, this proposed model is only working on the single-core processor only and the results when converted to frame-per-second is around 0.3 – 1.2 FPS which is not fast enough.

In 2009, there is a research proposed implementing total variation regularization filter technique on video with other denoising technique[9]. The authors implemented their strategy on degraded video compressed with MPEG-4 codec which has mixed types of noise and block artifacts from lossy MPEG-4 encoding. By using deblocking[10] technique and applying the regularization filter, the authors found this technique can be used on a real-time environment for high denoised video quality denoising which does not require the optimal solution. The PSNR value is improved to 34.35 on Akiyo video[11]. However, the author did not mention about time used, parallelization, and the frame rate.

The video and image processing field in the past few years, there are many researches which use multi-core processor for process the video. Most of the research aimed for implementing on the multi-core processor for encoding and decoding the video sequence. For example in [12], the author used multi-core Xeon processor to speed up the video H.264 encoding by using the master-slave scheme which improved the speed over single thread encoder by 29%.

Though, the past researches we reviewed are interesting, none of them is working in the parallel platform like on the multi-core processor. In this research, we propose the scheme for threads and video frames management for multi-core video denoising which retain correctness of the frame order and utilizing each processor core effectively. We focus on threads management and frame data management rather than the algorithm for denoising. We have chosen Total Variation (TV) for our denoising algorithm which can remove AWGS efficiently while preserving edges and details[13]. However, our model is flexible enough to allow switching or adding another image denoising algorithm.

3. PROPOSED STRATEGIES

In this section we divided into 4 parts. Section 3.1 presents the block strategy which we design for the parallel video denoising. Then we improve the strategy in Section 3.2, called the distributor strategy to effectively utilize the threads. Next, we enhance with some technique like the prefetching (Section 3.3) to increase frame rate and frame rate control (Section 3.4) to make video smoother and retain the same length with original video by skipping some frames which cannot be denoised in time.

3.1 Block Strategy

We assume the streaming data for our real-time video denoising. The video frames have to arrive in order. Our very first model for parallel real-time denoising is to retain the frame order after denoising. We divide the computation into substeps which will denoise one block of frames at a time. We have each thread waiting for the stream data in a sequence, where n^{th} threads except the first thread will wait for the fetched signal from the $(n-1)^{th}$ thread, then it will start fetching the next frame. After a thread gets a frame from the video, it will send the fetched signal to the next thread and to start its computation.

When a thread finishes the computation for its frame, it will put that the denoised frame back to the video in the right sequence. This has to be done by the special thread synchronization.

We illustrate the diagram for computation steps of each thread as Figure 1.

Figure 1 shows the diagram of our first strategy for the parallel real-time video denoising. This strategy will retain the frame order after the video denoising. The implementation is simple and we expect less thread synchronization on each step comparing with the scheme such as the pipelining.

However, it is found that the proposed method still has a lot of idle threads from waiting to fetch a frame and thread synchronization. We propose a new strategy by introducing the frame distributor. We describe the new strategy in the next section.

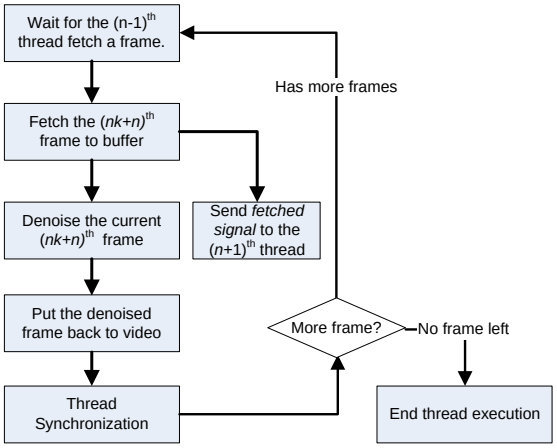

Fig 1: Block strategy for n^{th} thread on k^{th} substep

3.2 Distributor Strategy

Our idea is to eliminate the thread synchronization and idle threads by having each thread executing independently. The frame distributor is an extra thread that reads the streaming video frame and distributes frame to each thread. After a thread finished its computation, it will send back the computation results to the frame distributor and the thread will start the next computation. The frame distributor will arrange the denoised frames, put the back to the video while retaining the order of frames.

We illustrate the diagram for our distributor strategy as Figure 2, we use a thread as distributor to dynamically distribute frame to each worker thread. When computation starts, the each worker thread will request a frame from the frame distributor and frame distributor will pick an undenoised frame to send it back. After a thread finished computation, it will send denoised frame back to frame distributor and request a new undenoised frame until all frames are denoised.

In the implementation, the messages, the frame data and the signals between the worker threads and the distributor thread on OpenMP can be implemented by using shared variables and setting the pointer to a frame for each thread.

3.3 Prefetching Strategy

We also apply the prefetching strategy to the previous distributor strategy to enhance the strategy's performance. During the execution, the distributor thread will create n frames buffer where n is total number of threads. The distributor thread will read a frame and store it to a buffer of the k^{th} thread in advance. Figure 3 shows the flowchart. It is seen that we create a prefetch buffer for storing the frame. We vary several number of frames to be prefetched which is to determine the proper prefetch frame in each case. This will be shown in our experimental results.

In Figure 3 we show our modified model with the prefetching technique indicated by the rectangles. After the frame distributor sends a frame to worker threads. There will be small idle time. We make the frame distributor fetch the next frame and store in the buffer instead of fetching video on-the-fly style which causes some I/O latency.

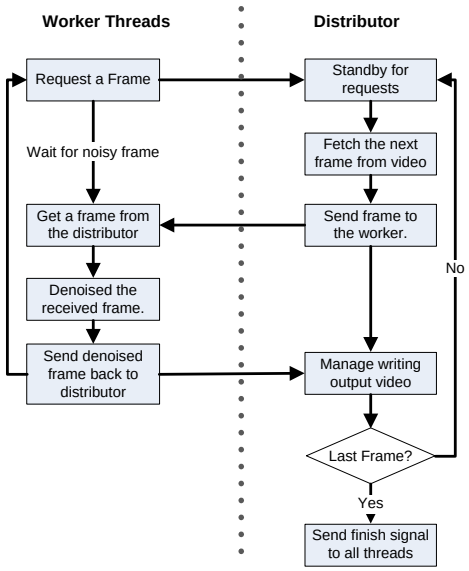

Fig 2: Distributor strategy for n^{th} thread on k^{th} substep

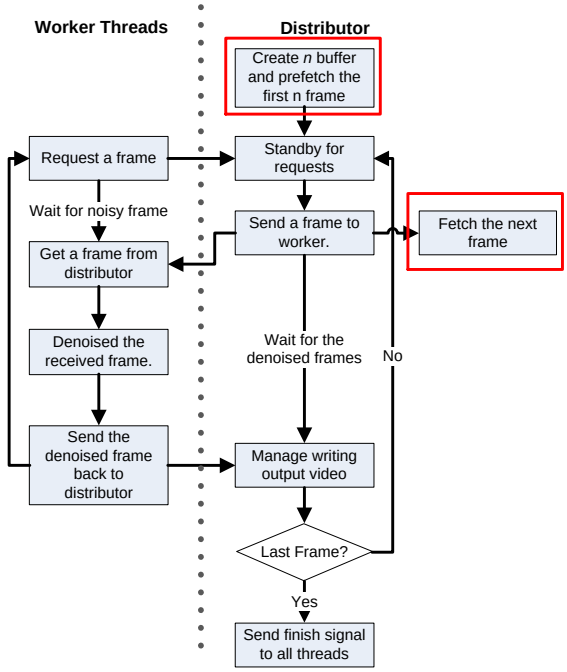

Fig 3: Distributor strategy with prefetching

3.4 Frame Rate Control Strategy

We observe that output frame rate is still not stable with our previous strategy. We apply a "frame skip" frame rate control technique by attaching an *expiration timestamp* to each frame before the distributor sends a frame to a worker thread. The worker thread will check the timestamp before sending the frame back to the frame distributor and the frame distributor will recheck before putting the frame back for output. If the time on timer greater than the timestamp on that frame. That frame will be discarded.

We compute the timestamp as Equation (10)

$$T_{end} = T_{start} + \left(\frac{N+1}{R} \right) \qquad (10)$$

where T_{end} and T_{start} is the expiration timestamp value and initial timestamp value respectively. N is the frame number and R is the original video frame rate in frames per second.

Our parallel real-time video denoising scheme can be shown as Figure 4. We make some modification as shown in the box. The job distributor checks the timestamp of the incoming frame to discard delayed frame instead of waiting for them. If any denoised frame does not come in time, the frame distributor will ignore and we need to make some modification to output system to fulfill this strategy. Moreover, the frame distributor needs to check the timestamp of the outgoing frame. Only unexpired frames will be sent out to worker threads.

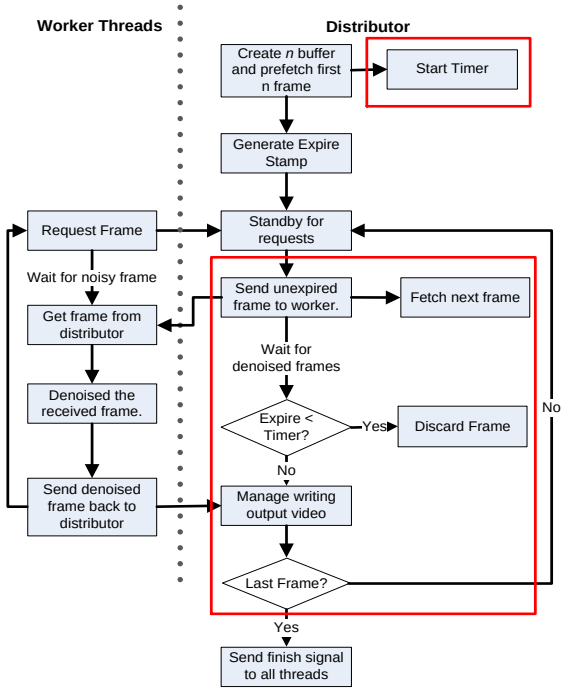

Fig 4: The distributor strategy with the frame rate control.

We also need to make some modification to output system to check the availability of denoised frame before showing a frame to screen as shown Figure 5. Every 1/*FPS* second the output system will check for availability of next frame where *FPS* is the original frame rate of original video. If the current denoised frame isn't available, the output system will use the previous frame to show on the screen and will not wait for that frame anymore to retain the video length as original.

4. EXPERIMENTAL RESULTS

We test our strategies on Intel® Core 2 Quad with four 2.5 GHz of processor cores, 4,096 MB of main memory. Our experiment platform is 64-bit version of Fedora 16 Linux, OpenCV library 2.3.1 for reading and writing video, GNU C Compiler(GCC) 4.5.5 for compiler and OpenMP library.

The experiments use the test video clips from [11] named "Miss America" with the frame rate at 15 frames per second[11], "Harbour" with the frame rate at 30 frames per second[11],. We also used video named "salesman" which has 449 frames QCIF format [15], for time and PSNR comparison with [1]. We generate 25% of Addtitive White Gaussian Noise (AWGN) randomly on each color channel individually to every video frames The original video and noisy video will be displayed in Figure 6.

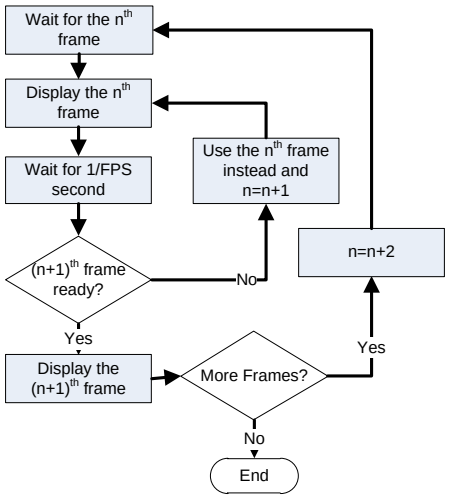

Fig 5: The modified output player with the frame rate control.

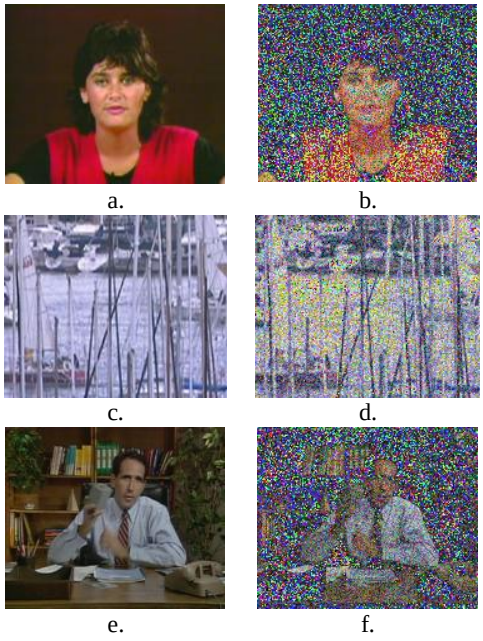

Fig 6: The sample video for experiments. The 30th frame of original Miss America(a), Miss America with 25% AWGN(b), original Harbor(c), harbor with 25% AWGN(d), salesman(e) and salesman with 25% AWGN(f)

In this paper, there are two types of evaluations. The first one is the performance for each scheme by measuring the frame rate in frame-per-second(FPS). Furthermore, we also measure denoised video quality using the PSNR value on some frames.

4.1 Performance Evaluation

4.1.1 Distributor strategy performance

We measure the frame rate for the block strategy and the distributor strategy from Harbor video varying the number of threads from 2 to 16 threads as in Figure 7.

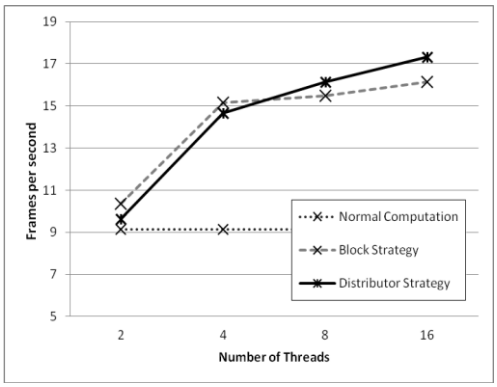

Fig 7: The number of frames per second on each strategy type varying the number of threads for Miss America video

Figure 7 shows that the job distributor works better than the block strategy when number of threads is more than four. In this method, we use one of threads to be the frame distributor (for example: 4-threads computation has 3 worker threads and 1 distributor thread). Actually, this also happens for the case of Harbor and Salesman. We also inspect the number of frames processed by each thread in the distributor model. Each thread processes about the same number of frames.

The main reason that the distributor approach works well is that the block approach needs all the threads to finish for its frame before moving to the next iteration (Figure 1 at thread synchronization step). This incurs the waiting time. Also, if we use too small number of threads, for example for the case of two threads, the performance is obviously lower than the block strategy. We can see that when the number of threads is low, the block strategy will work slightly better than the distributor strategy. This is caused by the fact that some overhead between the frame distributor and worker threads are more than the synchronization time in the block strategy.

We find that the distributor strategy boosted the frame rate up to 17.34 FPS and produces up to 7.43% better frame rate than block strategy on 16-threads computation.

4.1.2 Performance with prefetching strategy

We apply the prefetching strategy to our distributor strategy as mentioned in Section 4.1.1. The comparison between distributor strategy with and without one frame prefetching on 16-threads computation on Miss America video is depicted in Figure 8.

From Figure 8, we can see that our enhanced prefetching strategy can slightly increase FPS in the denoising process in all the test cases.

The more number of threads used, the more benefits are obtained from the prefetching strategy. It is found that our prefetching strategy can boost FPS by up to 4.38% on the 16 threads computation.

We also vary the number of prefetched frames from 1 to 8 frames. We found that the more number of prefetched frames may affect the performance. If we prefetch the proper number of frames at a time, we will get the good performance boost. If we prefetch too many frames at a time, the frame distributor will get busy with I/O rather than distributing frames to workers which will reduce the overall performance. The frame rate results of our strategy with prefetching technique is shown in Figure 9.

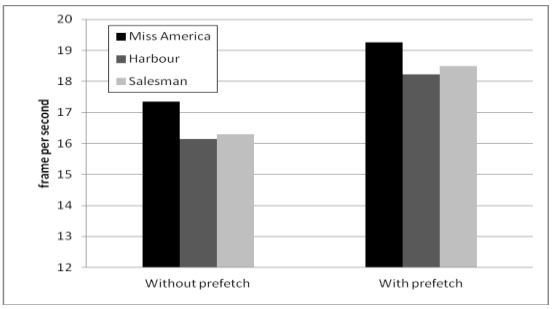

Fig 8: Frame rate comparison between the distributor strategy with and without the prefetching strategy

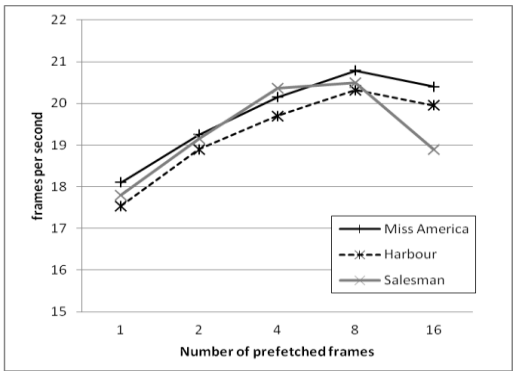

Fig 9: Frame rate for distributor strategy varying the number of prefetched frames.

From Figure 9 we can see that number of prefetched frame slightly affect the frame rate of video denoising. The best result is prefetching 8 frames at a time which can increase the frame rate up to 22.02% comparing to the distributor strategy without prefetching strategy.

We also double the size of each video and vary the number of prefetched frames each time. The results are shown as in Figure 10.

From Figure 10, we also find that the video size affects the prefetching's performance. All of our normal size videos have the best performance when the number of frames to be prefetch to 8 frames. However, when we increase the size of video at double, the best number of frame to be prefetched is 4 frames. It is clear that the performance in frame rates is affected linearly by the size of prefetched buffers and video.

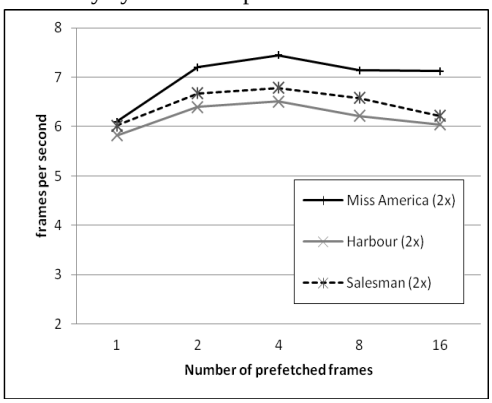

Fig 10: Frame rate of the distributor strategy varying the number of prefetched frames (double size video)

For example, the double-size video needs the half number of prefetched frames to obtain the peak performance and the frame rate drops about 3 times of that of the normal size. For the example, the case of Miss America, in Figure 9, the peak frame rate is 20 when the number of prefetched frames is four, but in Figure 10, the frame rate drops to 7.5 when the number of prefetched frames is 8. We also have the hypothesis that prefetching a lot of large frames at once will make frame distributor busy waiting for I/O and cannot distribute the frames to worker threads efficiently.

4.1.3 Frame Rate Control

We notice the unstable frame rate during the denoised video playback. This is caused by the fact that some frames are processed too slow which makes the output video player keeps waiting and make the overall video length seem to be slow and longer than it should be. We measure the MSE and the total MSE compared with the baseline 30 FPS from every 30 frames for Harbor video as in Table 1.

Table 1. Total original video length and denoised video length and error without the frame rate control.

Video Name	Original length (s)	Denoised length (s)	Error(%)
Miss America	5.00	5.61	12.20
Harbour	20.00	32.10	60.50
Salesman	29.93	33.47	11.83

From Table 1, the column "Original length" shows the length that video should be played properly. The second and third columns are the actual total time of original video and actual output time for the denoised video in seconds. The last column is the error rate computed by the difference of original length and denoised length in percent.

We noticed that Harbour video has a large error rate because the original video has the frame rate at 30 FPS in which our current hardware still cannot compute at that rate. However, in the near future denoising in 30 FPS will be possible as the advancement of the processor hardware.

We apply our frame rate control strategy to our strategy. The results are shown as Table 2. The "Frames Discarded" column shows the number of frames which is skipped by the output system because the slow denoising process cannot deliver the frames in time. The "Denoised Length" column shows the total denoised video time playback time, "Error (%)" is computed by the difference with the original video time playback time in percent and "% Improved" is the improvement of the error in the total video length compared with the error in Table 1.

Table 2. Average frame rate and the error for Harbor with the frame rate control.

Video Name	Frames Discard	Denoised Length	Error (%)	% Improve
Miss America	4	5.04	0.80	11.4
Harbour	173	21.01	5.05	55.45
Salesman	9	30.36	1.20	10.63

4.2 Denoised Image Quality Evaluation

The denoised frame is shown in Figure 11.

(a) (b) (c)

Fig 11: Denoised frame of the 30th frame of Miss America(a), Harbor(b) and Salesman (c)

From Figure 11, we can see that the denoised frame is visually satisfactory. We measure the denoised frame quality by using the average peak signal-to-noise ratio(PSNR) value for red, green and blue channels compared with the original frame. We describe PSNR value as in Equation (11)

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \quad (11)$$

where MAX is the maximum pixel value on the images, MSE is the mean square error which is described as in Equation (12),

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (12)$$

From Equation (12), *m* and *n* are image height and width in pixel respectively. *I* is the original image and *K* is the compared image. We measure the PSNR value for some frames and show the results in the Tables 3-4. It is shown that the denoised frame has a better quality in every case. The best case in the Table 3 is the improvement by 48.01%.

Table 3. PSNR values for video frames for Miss America and Harbour.

Frame Number	PSNR			
	Miss America		Harbour	
	Noisy Frame	Denoised Frame	Noisy Frame	Denoised Frame
15	38.72	**56.83**	38.70	50.10
30	38.74	55.52	38.58	**51.46**
45	38.70	56.70	38.70	51.29
60	38.73	55.73	38.76	51.39
Avg.	*38.72*	*56.19*	*38.68*	*51.06*

Table 4. PSNR values for video frames for Salesman compared with the proposed scheme in [1].

Algorithm	PSNR		
	Min	Max	Average
Our TV+FP	**33.90**	43.07	**34.80**
2D-DWT	29.84	46.79	33.33
2D DTCWT	30.45	47.53	33.89
System in [1]	30.46	**48.18**	33.63

Table 4 shows that our usage of the total variation image denoising with the lagged diffusion fixed point method can improve the PSNR from denoising AWGN noisy frames up to 43.07 or 3.48% more than [1] on average and better than using 2D-DWT and 2D DTCWT which is mentioned in [1] in all cases.

From results in Table 3 and Table 4, we conclude that our denoised video quality has a better minimum and average PSNR values on the Salesman video compared with the previous work in [1]. Also, the maximun PSNR value of our approach is about the same as the work in [1]

5. CONCLUSIONS

In this paper, we design the strategies for the parallel real-time video denoising which attempt to retain the frame order while utilizing the threads. We focus on eliminating the additive Gaussian noise by using the total variation with fixed-point iterative method which is robust, memory efficient and fast convergence rate. We present two approaches: the block approach and the distributor approach. From our experiments on the sample videos we found that our distributor strategy has more 1.20 FPS or 7.43% better frame rate over the block strategy and the better thread utilization by eliminating the thread synchronization and idle threads on the computation step.

Furthermore, we also enhanced the distributor's strategy performance by adding the prefetching buffer to make the distributor thread store some frames on its buffers before sending the frame to working threads. We found this enhancement can further boost the frame rate by 22.02% compared with the original distributor strategy. We tried to vary the video size and we found that the prefetched buffer size should be proportional to the video size. The performance obtained is also varied linearly depending on the video size as well. The double-size video needs the half number of prefetched frames to obtain the peak performance and the frame rate drops about 3 times of that of the normal size. We also apply frame rate control strategy by adding frame-skip technique to delayed denoised frames. The total playback time of the video is more accurate and closer to that of the original video.

However, there are still some issues on the denoising that will need to be addressed in the future research. First, we need to improve the algorithm by considering load balancing in the job distributor model. We found that each frame use difference computation time for denoising caused by the difference in quantity of noises and features on each frame, we are going to apply the adaptive scheme on our approach. Also, we need to improve the denoising algorithm to eliminate other types of noises rather than the Gaussian white noise. The other noise may come from video encoding like MPEG encoding [14,15]. Next, the frame rate still is not good for much larger videos like high-definition(HD) videos. In the future, we have a plan to modify this strategy to use with the more powerful processor architecture like CUDA[16] or OpenCL[17] which uses a lot of threads and cores for the computation. We expect a more sastisfiable frame rate on CUDA with the HD video from the application of our strategy in the near future.

6. ACKNOWLEDGMENTS

This work is supported in part by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program contract no. PHD/0275/2551.

7. REFERENCES

[1] A. Sarhan, M. T. Faheem, R. O. Mahmoud, "A Proposed Intelligent Denoising Technique for Spatial Video Denoising for Real-Time Applications", Intl J. Mobile Comp and Mobile Comm (IJMCMC), vol. 2 2010.

[2] L. I. Rudin, S. Osher, E. Fatami, "Nonlinear total variation based noise removal algorithms", Physica D., vol. 60, pp. 259—268, 1992.

[3] C. R. Vogel, M. E. Oman, "Fast, robust total variation-based reconstruction of noisy, blurred images", IEEE Trans. Image Process, vol.7, pp. 813—824, 1998.

[4] C. R. Vogel, M. E. Oman, "Iterative Methods for Total Variation Denoising" , SIAM J. Sci. Comput., vol.17, pp.227—238, 1996.

[5] B. Dolwithayakul, C. Chantrapornchai, N. Chumchob, "GPU-Based Total Variation Image Restoration using Sliding Window Gauss-Seidel Algorithm", Proceeding of Intelligent Signal Processing and Communication Systems (ISPACS 2011), pp. 1—6 , 2011.

[6] Paul R., M. Meyer, "Restoration of motion picture film," Conservation and Museology, Butterworth-Heinemann, 2000.

[7] T. F. Chan, G. H. Gohub, P. Mulet, "A nonlinear primal-dual method for total variation based image restoration", SIAM J. Sci. Comput., vol.20, pp. 1964—1977, 1999.

[8] R. Bagnara, "A unified proof for the convergence of Jacobi and Gauss-Seidel methods.", J. SIAM Review, vol.37(1), pp.93—97, 1995.

[9] A. Ogier, P. Hellier, C. Barillot, "Restoration of 3D medical image with total variation scheme on wavelet domain (TVW)", Proceeding of the SPIE, vol. 6144, pp. 465—473, 2006.

[10] P. Piastowski, "Image processing to reduce blocking artifacts," US Patent Application US20060274959, 2006.

[11] Xiph.org, "Xiph.org Test Media", Available via http://media.xiph.org/video/derf/, Accessed 14 May 2012.

[12] ITU-T, "H.262 Information technology – Generic coding of moving pictures and associated audio information: Video", International Telecommunication Union-Telecommunication Standardization Sector, 2000.

[13] C. Dolar, M. M. Richter, H. Schroder, "Total variation regularization filtering for video signal processing", Proceeding of 13th IEEE International Symposium on Consumer Electronics(ISCE2009), pp. 1—5, 2009.

[14] Chiariglione.org, "MPEG standards – Full list of standards developed or under development", MPEG, Retrieved 31 Oct. 2009

[15] J. Ive, "Image formats for HDTV", European Broadcasting Union Technical Review, 2004.

[16] NVIDIA® Corporation, "NVIDIA CUDA compute unified device architecture programming guide version 2.1", 2008.

[17] A. Munshi, "OpenCL Parallel Computing on the GPU and CPU", International Conference and Exhibition on Computer Graphics and Interactive Technique (SIGGRAPH 2008), 2008

[18] A. Marquina, S. Osher, "Explicit algorithm for a new time dependent model based on level set motion for nonlinear deblurring and noise removal", SIAM J. Sci. Compute., vol.22, pp. 387—405, 2000.

[19] J. Gu., Y. Sun, "Optimizing a parallel video encoder with message passing and a shared memory architecture", Tsinghua Science and Technology, vol.16(4), pp.393—398, 2011.