An optimized Window Memoization Architecture for High Performance and Accurate Image Processing

Mojtaba Farzmahdi^{a*}, Luo Rong^a

^a Department of Electronic Engineering, Tsinghua University, Beijing 100084, CHINA

*Corresponding Author: mfarzmahdi@mails.tsinghua.edu.cn

Abstract

In this paper, we propose a new performance technique for exact computation on window memoization algorithm instead of tolerant window memoization. Window memoization is a technique in the field of local image processing that benefit memoization method and data redundancy on images to improve speedups of computation performance by eliminating computational redundancy.

In tolerant method a number of LSBs bits of neighborhood pixels in the windows are eliminated to lead to increase the probability of having similar windows in an image and it cause to gain the speedups computations performance by skipping the computation of similar windows, but only using MSBs bits and eliminating LSBs bits of neighborhood pixels in the window cause accuracy loss in the results.

In the other hand some imaging devices make more bits per pixel, for example MRI, PET, CT in medical imaging devices and Ultra High Definition TV in commercial device, that make 10 to 16 bits per pixels, for this reasons it's necessary to have a method to support high depth of pixel without eliminating LSBs bits and have accurate results and also acceptable speedups.

We have developed this optimized method on software and applied edge detector to natural and medical images of 512x512 pixels and the typical speedups is 1.29x and 1.42x in comparison to conventional method and also without accuracy loss in the results.

Keywords: Window Memoization, Computational redundancy, Inter-pixel redundancy.

1. Introduction

In recent years the field of imaging systems and source imaging has noticeable developed, the quantity and the size of images and also the frame rate are constantly increasing, fast and optimally interactive processing of this data is major concern.

Now there is a sort of break down between making images and processing ⁽¹⁾, the imaging systems spend very little time with the measurement and making the images and then huge of time for managing and processing imaging data and it is a really big challenge, for example the processing time more than minutes are common for some medical imaging data^(2,3), or some applications in industrial or robotic systems need a fast systems to extract the features for visual feedback control systems to have stable close control loop, therefore, it is really need to have high performance image processing algorithms and implementation to handle this type of data and applications.

One important operation in image processing applications that have expensive computational process and take the huge of time to enhance the images or make suitable data for the next steps (post processing) is local image processing operation or local filtering, in this type of operation, a finite window size of weighted value is scanned through the image and applied some algorithms to the value of the pixels in the image that covered by this window, and then store the corresponding result to the frame buffer.

It is important operation for the fast processing or in the applications that use huge of image data, because a set of computations must be repeated for several times on entire of the image. The combination high volume of data and

high performance requirements leads to an urgent optimize solutions.

For optimizing solution, there are some conventional methods 1) Fast processors 2) Parallel processing and 3) Optimization Algorithms. The simplest way for optimizing solutions is using high speed processors but it's not possible to use them in all applications such as embedded systems, because the special processors have more capability that it's needed in embedded systems and expensive and also have high power consumption, for this reasons it's not suitable for some applications for example handheld systems, in the other hand volume of imaging data is growing faster than speed of processors. Another method is parallel processing, in this method one computation is divided into multiple processes with the objective of computation performance in less time and faster than conventional methods, but this type of implementation is complex and need special architecture. Another method is optimization algorithms, the goal of this method is trying to find a way to minimize the number of computations and functions for doing algorithms in less time than conventional methods.

Memoization technique is one of optimization algorithm that use the previous results in an algorithm to computation performance. Memoization method speed up computation performance by storing the results and reusing it when the same inputs incoming to algorithm again.

Window memoization is a type of memoization method that memoize window operation in local image processing algorithms and speed up the computation performance. In current (tolerant) window memoization method a number of MSBs bits of neighborhood pixels in the window are used for memoizing and other part of pixels (LSBs bits) are eliminated, by using a part of bits of pixels in the window, it lead to drastic increase the probability of having similar windows in an image and it lead to gain the speedups by skipping and avoiding the computation of similar windows, but it cause accuracy loss in the results. in some applications the accuracy loss in the results is negligible but not in all of applications for example in UHDTV that use 10 bits for removing the contouring effect or in medical imaging devices that generating 10 to 16 bits for each pixel to have accurate results.

In our proposed method in the field of window memoization, memoization apply on area in an image that have the same MSBs bits of pixels in the window where the difference between adjacent pixels are small and only the LSBs bits of pixels in the window are used by memoization algorithm.

In the reminder of this paper we present the related work of memoization technique and window memoization, Data redundancy and its relationship to computational redundancy, tolerant window memoization and our new method and finally show the results of speedups and hit-rate versus to percentage of similar windows in images.

2. Related work

There are a lot of research work about memoization on computer programming, embedded system and also in the field of image processing. The goal of memoization for all of them is improve the efficiencies and speedup computation performance and also minimize the power consumption in embedded systems and handheld devices.

For the first time Donald Michie⁽⁴⁾ introduced the concept of memoization idea for computer programing to increase efficiencies through software and speed up process, he used memory table to save the results of each function to avoid redundant computation, in Michie's algorithm before computation, all of entry in the memory table would be checked prior to find the match existence, and with this method when the memory table gets large it would take long of time to find a match and finding a match take longer than re-computation of this set of input data again.

Richardson⁽⁵⁾ proposed a technique to memoize redundant computations and eliminated trivial and redundant computations for division, multiplication and square root, he found this type of computations are more than 67% for some algorithms. for example for multiplication when at least one of the two inputs is 0, -1, 1 this calculation is trivial or in the division operation when the value of the numerator is zero or the inputs are equal, and for square root where the input is zero or one this type of computations are trivial.

Ding⁽⁶⁾ proposed a method to consider performance and power consumption in embedded software and handheld devices where the power consumption is important, in this method memoization technique apply to selected segments which frequently used with using this method he achieved speed up 1.37x and 22% reduction in power consumption.

For the first time Khalvati^(7,8) introduced the window memoization for local image processing algorithms and introduced tolerant window memoization, in tolerant window memoization only n last most significant bits (MSBs) of each pixels in the neighborhood of window are used for making symbol and memoization, in this method by reducing a number of LSBs bits, the number of similar

symbols in entire of an image and also hit-rate are increased and it cause to improve speedups computation but in the other hand this similar windows recall the same responses from reuse table and it cause to have inaccuracy in the results,

Khalvati and Agard and Tizhoosh ⁽⁹⁾ developed a model to predict the speedups on window memoization and compare this parameter by implementing memoization across different processors. they also proposed another new method to improve efficiency and increase speedups factor by using two wide super scalar pipeline to implement memoization on hardware ⁽⁸⁾, in this method two set of pixels of window incoming to memoize systems simultaneously. In this method the typical speedup factor is 1.58x with 40% less hardware to compare conventional method but also have inaccuracy in the results by considering the 4 MSBs bits of pixels for making symbols in memoization technique.

In the next section we propose a new method in window memoization technique that improve the speedup and also preserve accuracy on the results.

3. Window Memoization Technique

3.1 Concept of Memoization

Memoization is an optimization technique that speed up the computation performance by storing the results of computation for set of inputs and use the results when the same inputs occur again, in computer programming, memoization technique are used for expensive function. the benefits of using memoization are: Speed up computation performance, decreasing the power consumption on embedded system and handheld devices ⁽⁶⁾ and also Change the complex computations to simple recall operation and free the resources when all of the results have been prepared for all of possible input value and it will be only a simple recall operation when it will be needed to computation in the future.

3.2 Data and Computational Redundancy on Image

Data redundancy lead to computation redundancy in images or frames and two major types of data redundancy exist in images data, coding and inter-pixel redundancy ^(7, 10), image processing algorithms can take these advantage of redundancy to improve computation performance and also reduce the size of image or frame of data ⁽¹¹⁾.

Coding redundancy refer to this fact that the number of

bits per pixels to represent the image is more than is required and it's possible to measure it by entropy of the image. Entropy can be measured by the average information per pixel in an image, which is calculated as follow ⁽¹¹⁾:

$$H = - \sum_{i=0}^{GL-1} P_i \times \log_2 P_i \quad (1)$$

Where H is the Entropy of the image, and P_i and GL are the probability of the occurrence of particular gray level i in the image and the number of gray levels.

The coding redundancy can be determined by following formula:

$$CR = \log_2(GL) - H \quad (2)$$

Compression algorithms have taken advantage of coding redundancy to reduce the size of the image.

The advantage of this type of redundancy also are used in tolerant window memoization to eliminate computation redundancy ⁽⁷⁻¹⁰⁾.

Another type of redundancy is inter-pixel redundancy, inter-pixel redundancy is due to the correlation between neighboring pixels. In images the neighborhood pixels are highly correlated because the object in images have similar gray levels, and objects typically are characterized by similar texturing or coloring and have particular structure. This type of redundancy have the main impact on computation redundancy.

In our new method proposed we considered this type of data redundancy to find similar windows that have the same MSBs bits of pixels in the window and then apply window memoization algorithms to LSBs bits of pixels in that windows.

One of the main methods to measure inter-pixel redundancy is Auto-correlation. This method shows and measure the inter-pixel redundancy between two adjacent pixels, an image can be shifted in one direction and then measure the auto-correlation of an image and the shifted image as follows:

$$AC(\Delta_n) = \frac{\sum_{x,y} E[(I - \mu)(I_{\Delta n} - \mu)]}{\sum_{x,y} E[(I - \mu)^2]} \quad (3)$$

Where $AC(\Delta_n)$ is the auto-correlation of the input image I

and shifted image $I_{\Delta n}$, μ is the mean of the input and E is the expected value operator, higher autocorrelation means more inter-pixel redundancy and more data redundancy, the result of more data redundancy lead to more computation redundancy and more achievement speedups when window memoization technique apply to this type of images.

3.3 Window memoization & tolerant window memoization

Local image processing algorithms use convolution operator to apply filter to images. In window operation, the specific size of kernel filter apply to neighborhood of pixels in the windows in the entire of image to implement linear or nonlinear filters to enhance image or extract local features. The output image (8 bits gray level) in w neighboring of pixels is calculated as follows:

$$Q[x, y] = \max(0, \min(255, \sum_{i,j \in w} w[i, j] I[x + i, y + j])) \quad (4)$$

Where $w[i, j]$ is kernel filter, I is input image and Q is output image.

The window memoization is a technique that apply to local image processing algorithms to speed up the computation performance by minimizing the number of total computation by identifying similar input windows that previously performed and stored to the reuse table, and reuse the stored results instead of computation it again. In this method when a set of computations has to be performed for the first time, they are performed and the corresponding results are stored in the reuse table. If the same set of computations has to be performed again, the previously calculated result is reused and the actual computations are skipped. This eliminates the redundant computations leads to increase speedups factor.

The window of pixels in an image that covered by kernel filter is named *symbol*, and all the similar windows in the image are identified by one symbol, For windows of $n \times n$ pixels, n^2 dimensional symbol represents all the identical windows in the image.

In define of window memoization technique ⁽⁷⁻¹⁰⁾ and Ideal memoization a window belong to a symbol if each pixel in symbol is equal to the corresponding pixel in window and if the computation performed for that window in previous it is reused the saved result and hit occurred. The condition of ideal window memoization shows in equation 5.

$$\begin{aligned} \forall pix \in win, \forall pix' \in sym, \\ pix = pix' \Rightarrow win \in sym \end{aligned} \quad (5)$$

$$Result(win) = Saved_Result(sym)$$

In Ideal memoization hit rate will be increased by applying more inputs to the memoization algorithm to save the corresponding results of symbols and also need a huge size of memory to store all of possible symbols.

To handle this problems the tolerant memoization introduced, that can be achieved greater speed-up at the cost of introducing some error into the output results. in tolerant memoization only use a number of MSBs bits of window's pixels to represent the symbols, in other word, similar windows belong to one symbol not necessarily identical windows⁽⁷⁾:

$$\begin{aligned} \forall pix \in win, \forall pix' \in sym, \\ MSB(n, pix) = pix' \Rightarrow win \in sym \end{aligned} \quad (6)$$

$$Result(win) = Saved_Result(sym)$$

Where MSB (n,pix) represent n last MSBs of pixel in the window.

In tolerant memoization technique the hit-rate are increased and it cause to increase speedups factor but add inaccuracy in the results and it cause to decrease signal to noise ratio (SNR). after making symbol of corresponding window, the hash function is used to map incoming symbol to a suitable address in hash table, and in that address the result and symbol are saved for the next reuse if the same window incoming again.

The result save with corresponding symbol in the hash reuse table because the hash reuse table with finite size are not capable to saving all of possible input symbols and sometimes different symbols produce the same address. The previous saved result will be used and hit occurs when an input symbol and the saved symbol in that mapped address of hash table will be same.

The difference processing time between memoized algorithms and non memoized algorithms is used to determine the speedup computation performance is calculated as follows,

$$Speedup = \frac{t_{conventional}}{t_{memoization} + ((1-HR) \times t_{conventional})} \quad (7)$$

Where $t_{\text{conventional}}$ is processing time in conventional method, $t_{\text{memoization}}$ is processing time in memoization method and HR (hit rate) is the percentage of hit occurred and the result can be reused.

This formula shows the high hit-rate lead to increase speedup parameter and also this parameter is dependent on processor speed and read and write accessing time to the memory in memoization algorithms.

For increasing hit-rate and improvement speedups factor in window memoization method, the tolerant method offer eliminating LSBs bits of pixels in the window and considering only the MSBs bits of pixels in the window, In this method the windows that are similar but not identical are assigned to one symbol. This will potentially introduce inaccuracy in the result of the algorithm. However, in some applications accuracy loss in results is negligible⁽⁹⁾.

3.4 Proposed method

This is a new idea to extract the exact results without inaccuracy and also improve the speedups parameter in window operation on local image processing algorithms by using window memoization.

In this method the value of pixels in neighborhood of the window are divided into two parts, MSBs and LSBs parts and the memoization algorithms only apply to LSBs bits of pixels on windows that have the same part of MSBs bits of pixels in the window,

If all of MSBs bits of pixels in the neighborhood of window are same, then the MSBs represent a constant value that add to each pixels in that window.

in most of kernel filter in local image processing for window operation, the summation of value of filter kernel equal to zero such as edge detection kernels, Laplacian kernel and many other kernels, by considering proposed method, the result of window operation in this type of kernels is equal to weighted sum of the LSBs value of pixels within the window which is calculated as follow if all of MSBs bits of pixels in the window are equal:

$$(8)$$

$$Q[x, y] = \sum_{i, j \in W} W[i, j] I[x + i, y + j]$$

$$i, j \in W : I[x + i, y + j] = C + \Delta[x + i, y + j]$$

$$Q[x, y] = \sum_{i, j \in W} W[i, j] (C + \Delta[x + i, y + j])$$

If

$$\sum_{i, j \in W} W[i, j] = 0$$

Then

$$Q[x, y] = \sum_{i, j \in W} W[i, j] \Delta[x + i, y + j]$$

Where I is the input image, Q is the results of apply kernel to input image, w : local neighborhood centered on input image I , W is coefficients of kernel filter, C is the same part of MSBs bits of pixels in the window and it is a constant value, Δ : the part of LSBs bits of pixels in the window.

This equation shows that it is possible to divide the value of pixels of window in two parts, the constant value: the same part of MSBs bits of pixels in the window and the LSBs bits of pixels in the window.

In this propose method of window memoization, n number of last MSBs bits of pixels are considered for checking the similarity and when the all of pixels in the window have the same MSBs bits then the memoization apply or reuse to LSBs bits of pixels in that window otherwise when the MSBs bits of pixels are not identical, the result are calculated by considering the total bits of pixels in the window and using conventional method.

In some filter function the summation of coefficients value on kernel filter are not equal to zero for example in smoothing kernels, Gaussian kernel, in this type of kernels the multiplied value the summation of coefficients kernel to value of same MSBs bits part, add to final result of window that extract by considering LSBs bits in window memoization algorithm.

(9)

$$Q[x, y] = \sum_{i, j \in W} W[i, j] I[x + i, y + j]$$

$$i, j \in W : I[x + i, y + j] = C + \Delta[x + i, y + j]$$

$$Q[x, y] = \sum_{i, j \in W} W[i, j] (C + \Delta[x + i, y + j])$$

If

$$\sum_{i, j \in W} W[i, j] = d$$

Then

$$Q[x, y] = Cd + \sum_{i, j \in W} W[i, j] \Delta[x + i, y + j]$$

Where d is summation value of coefficients kernel, C is the constant value of same part of MSBs bits of pixels in the window,

In this propose window memoization method only the LSBs bits of pixels (Δ) are used to make symbol and memoized and it cause to increase probability of finding the same windows on image and increase the hit-rate and speedups computation performance and also have the exact result without inaccuracy in final result.

Figure.2 shows the flowchart of new proposed method,

Fig2. Flowchart of Proposed method for optimizing window memoization.

The number of MSBs bits of pixels for finding the similar windows are effected to total number of similar windows that can be used in window memoization algorithm.

Figure.3 shows the percentage of existence 3x3 pixels windows in a sample image that have the same MSBs bits of pixels in each window, by evaluating the n number of last MSBs bits for checking windows similarity in the image from 1 to 5 MSBs bits.

For example 67% of image have the same last 3 MSBs bits

of pixels in the windows. It means the window memoization technique can apply on 67% area of image.

The number of windows that have the same MSBs bits of neighborhood pixels in the window are increased by reducing the numbers of MSBs bits for checking windows similarity.

Fig3. White area shows the existence of the same MSBs bits of pixels in the windows.

4. Results

By using this new method for preserving accuracy in the results, only the windows which have the same MSBs bits in the window are candidate to apply window memoization. The number of windows in image that can be applied

memoization technique is related to inter-pixel redundancy in the image.

A sample image and corresponding result after applying 3x3 gradient kernels of kirsch⁽¹²⁾ edge detector are illustrated on Figure.4.

In this image 57.8 % of image contain similar windows by evaluating 4 last MSBs bits of pixels in 3x3 windows. figure. 4 shows the area of image which Hit and Miss windows of memoization occurred in the image. In this sample image the number of Hit-Rate is 39.4%.

Fig4. Sample image top-left, the result of edge detection top-right, Hit area (gray pixels) in the input image bottom-left, Miss (gray pixels) in the input image bottom-right.

The number of similar windows that have the same MSBs bits of pixels in the window effect to the Hit-Rate and Speedup computation performance. figure 5 shows the percentage of similar windows and corresponding speedup parameter, the percentage of Hit-Rate of applying window memoization to 50 natural images with evaluating 4 last MSBs bits of pixels in the window to find similar windows.

5. Conclusions

In this paper, we introduced a new performance improvement technique on window memoization to apply window memoization on local image processing where both of speedup and accuracy are important. This method use

(a) Speedup Factor versus similar windows

(b) Hit-Rate versus similar windows

Fig.5. Speedup Factor and the percentage of Hit-Rate versus the percentage of similar windows in sample natural images.

benefit of inter-pixel redundancy to define the candidate windows for memoization.

In this method the pixels in the window are divided into two parts and only the windows have the same n last MSBs bits of pixels in the window are selected by algorithm for applying memoization on the LSBs part.

The results of LSBs bits of pixels on the neighborhood of window represent the exact result of pixels on the neighborhood of window when the summation of coefficients of kernel filter is zero, otherwise add a constant value proportional to the MSBs part value and also summation of coefficients of kernel filter to the result of window memoization to have the final result, when the summation of kernel filter is not equal to zero.

We implemented this method on software where kirsch⁽¹²⁾ edge detector algorithm were applied to 50 natural images and 25 medical and the results show, typical average speedup factor is 1.29x for natural images and 1.42x for medical images without accuracy loss in the results.

References

- (1) A Sboner, X J Mu, D Greenbov, R Auerbach, M B Gerstein: "the real cost of sequencing: higher than you think", Journal of Genome of biology, doi:10.1186/gb, 2010
- (2) A. C Hodge, A.Fenster, D. B. Downey, and H.M. Ladak: "prostate boundary segmentation from

ultrasound images using 2D active shape models: optimization and extension to 3D”, computer methods and programs in biomedicine, Vol. 84, pp. 99-113, 2006.

- (3) B. Hass, T. Coardi, M. Scholz, M. H. P. Kunz, U. Oppitz, L. Andre, V. Lengkeek, D. Huyskens, A. V.Esch, and R. Reddick: “Automatic segmentation of thoracic and pelvic CT images for radiotherapy planning using implicit anatomic knowledge and organ-specific segmentation strategies”,*Phys. Med. Biol.*, Vol. 53, pp. 1751-1771, 2008.
- (4) D. Michie: “Memo functions and machine learning,” *Nature*, Vol.218, pp. 19-22, 1968.
- (5) Richardson, S.E: “Exploiting trivial and redundant computation,” *IEEE symposium on computer arithmetic*”, pp. 220-227, 1993.
- (6) Y Ding, Z Li.: “Operation Reuse on handheld devices”, *Languages and Compilers for parallel computing*, Vol. 2958, pp. 273-287, 2003.
- (7) Khalvati F, Tizhoosh Hamid R. “Perfect Window Memoization: A theoretical model of an optimization technique for image processing algorithms”, *International conference on Image Processing Computer vision & Pattern Recognition IPCV11*, 2011.
- (8) Khalvati F, Agard Mark D, Tizhoosh Hamid R.: “Window Memoization: an efficient hardware architecture for high-performance image processing,” *Journal of real time image processing*, Vol. 5, pp 195-212, 2010.
- (9) Khalvati F, Agard Mark D, Tizhoosh Hamid R.: “Window Memoization: toward high-performance image processing software,” *Journal of real time image processing*, Vol. 10, pp 5-25, 2015.
- (10) Khalvati F, Agard Mark D, Tizhoosh Hamid R.: “Accelerating Image Processing algorithms based on the reuse of spatial patterns,” *Canadian conference on computer engineering*, pp 172-175, 2007.
- (11) R.C. Gonzalez and R.E. Woods. “Digital Image Processing,” 3rd Edition, Prentice Hall 2008.
- (12) Kirsch, R.A.: “Computer determination of constituent structure of biological images,” *Computer and Biomedical. Research* 4, pp 315-328, 1971.