ELSEVIER

Optics and Lasers in Engineering 30 (1998) 487–502

OPTICS and LASERS in ENGINEERING

The Tapestry Cellular Automata phase unwrapping algorithm for interferogram analysis

H.Y. Chang[a], C.W. Chen[a], C.K. Lee[b,*], C.P. Hu[c]

[a]*Department of Civil Engineering, National Taiwan University, Taipei, Taiwan, Republic of China*
[b]*Institute of Applied Mechanics, National Taiwan University, Taipei, Taiwan, Republic of China*
[c]*Atomic Energy Council, Executive Yuan, Taiwan, Republic of China*

Abstract

A newly developed phase-unwrapping algorithm, which is termed Tapestry Cellular Automata, are presented. Fundamental restrictions of traditional path-dependent phase unwrapping algorithms such as noise propagation and inconsistent data reconstruction are discussed first. The advantages and drawbacks of a path-independent algorithm, Cellular Automata, are then examined. The parallel and distributed processing nature of Tapestry Cellular Automata is shown to be able to keep the merits of traditional Cellular Automata algorithm while taking advantage of the rapid advancement of personal computers such as distributed computing over internet or intranet and multi-tasks operating environment. Both numerical simulation and experiments used to examine the effectiveness of this newly developed algorithm are presented in detail as well.

1. Background

Phase-shifting techniques are often used to convert interferograms obtained from optical interference techniques such as electronic speckle pattern interferometry (ESPI) [1], holographic interferometry [2], optical wavefront measurement, etc., into wrapped phase maps. Phase unwrapping is then used to convert the wrapped phase maps into unwrapped phase maps which corresponds to physical meaningful quantity such as displacement, strains, etc. As phase unwrapping algorithms have such a wide

*Corresponding author. Tel.: (+886) 2 2365-9436; fax: (+886) 2 2365-9436; e-mail: cklee@memsu1.iam.ntu.edu.tw

0143-8166/98/$ - see front matter
PII: S0143-8166(98)00054-2

range of scientific and engineering applications, many algorithms [3–8] have been developed over the years to accommodate all sorts of data processing needs. However, immunity to noise propagation and computational effectiveness are the two main conflicting issues that all of these phase unwrapping algorithms must try to balance. For example, conventional phase-dependent unwrapping algorithms were plagued by noise propagation and the users have to pay a lot of attention to keep the unwrapping results reliable and consistent.

The Cellular Automata method for phase unwrapping was first published in 1987 by Ghiglia [5], which brought a brand new path-independent way to unwrap the phase maps and can overcome many of the critical problems mentioned above easily. With the rapid advancement of personal computers, which include technologies such as multi-tasking and distributed computing environments, an innovative phase unwrapping algorithm evolved from the original Cellular Automata algorithm is developed to take advantage of these advancements. This newly developed computation algorithm was named Tapestry Cellular Automata (TCA) due to its domain segregation method. The phase masks concept, which is useful to handle many practical data processing needs, was also introduced into this newly developed TCA phase unwrapping algorithm. Advantages of the TCA algorithm such as easy implementation, straightforward error checking, fully parallel processing nature, and adaptable to distributed computing environment are all to be discussed in detail.

Phase unwrapping is the process of recovering the absolute value of the phase angle of a continuous function that extends over a range of more than 2π (relative to a starting point). In general cases, fringes possess a sinusoidal variation in intensity related to the phase angle θ. As the value of θ extends beyond 2π to say $2n\pi$ with n as the number of fringes, the absolute value of the phase angle can be recovered once the n is determined. In a noise-free pattern, we can determine the integer n by a simple fringe counting algorithm. However, once severe noise is present, such a "fringe counting" algorithm fails to reveal the correct integer n.

To overcome the fundamental shortcomings of the fringe counting algorithm, many alternative strategies that deal with noisy data sets have been proposed. The basic thinking behind most of these alternatives lies in the fact that phase data must be self-consistent to be unwrapped. For a path-dependent phase unwrapping method, the correct phase data is retrieved by using many different paths so as to circumvent the noisy data area. As the success of these types of phase unwrapping processes is highly dependent on the paths taken, many restrictions exist in applying these path-dependent methods in practical situations where high percentage of noise are often present in the data set obtained experimentally. Even though identifying all the locations where severe noise is present and taking all possible paths to unwrap the phase may improve the accuracy of the result, this often used tactic has seriously lowered the speed for phase unwrapping.

Compared to the above-mentioned path-dependent algorithms, path-independent unwrapping methods take a global view of the data array by unwrapping the phase along all possible paths between any two points. Cellular Automata [4], which concerns the evolution of discrete mathematical systems by simple mathematical rules, was developed as an effective path-independent unwrapping method [5,6].

Unlike conventional path-dependent methods, which need to take many complex paths to unwrap the phase maps line by line, Cellular Automata has a natural parallelism to unwrap the whole array of pixels. It is also because of the fundamental parallelism properties, Cellular Automata are immune to noise propagation and will not create distortion due to artificial smoothing often needed for conventional path-dependent phase unwrapping algorithms. With many of the merits mentioned above, the main drawbacks of Cellular Automata algorithm lie in its slow convergence to the final unwrapped phase maps. This lack of speed is the main reason that Cellular Automata did not receive more attention in dealing with practical phase-unwrapping requirements.

2. Cellular Automata

It is known that Cellular Automata can exhibit complex behavior resulting from the collective effects of a large number of cells, which evolves in discrete time steps according to simple local neighborhood rules [4]. The advantages of adopting the Cellular Automata in phase unwrapping [5,6] can be best understood by describing the unwrapping algorithm itself and the implementation issues such as the boundary conditions needed.

2.1. Algorithm

In Cellular Automata phase unwrapping algorithm, the phase data at all pixels are unwrapped at the same time. The two fundamental steps, the local iteration and the global iteration, form the backbone of this important algorithm. In local iteration, for each local iteration every pixel is compared with their four neighborhood points to decide if $+2\pi$ or -2π is needed to bring each neighborhood point to within the 2π range. In all the calculations done in this paper, the wrapped phase value is located within the $(-\pi, \pi]$ range. More specifically, this process first computes

$$k_m(x,y) = \sum_{ij} \operatorname{int}\left[\frac{|p(x,y) - p(x-i,y-j)|}{\pi}\right]\left[\frac{|p(x,y) - p(x-i,y-j)|}{p(x,y) - p(x-i,y-j)}\right], \tag{1}$$

where the subscript "m" means the mth iteration, the symbol "int" means integer, $i = 0$, for $j = 1$, or -1, and $j = 0$, for $i = 1$, or -1. If k_m is not equal to zero, a number of $+2\pi$ or -2π are added to the phase value of the original pixel according to the sign of k_m, i.e.,

$$L_m = \frac{2\pi|k_m|}{k_m}. \tag{2}$$

More specifically, if the original pixel after the mth local iteration is expressed as $p_m(x,y)$, the pixels modified after the above process becomes

$$p_{m+1}(x,y) = p_m(x,y) + L_m. \tag{3}$$

This step is repeated until the data set converges to a period 2 oscillation, which completes each local iteration cycle.

Once the map converges to a period 2 oscillation, the global iteration step is performed. The global iteration basically takes the average of the two period 2 oscillated phase maps I_m and I_{m+1}, both of which were obtained after each local iteration is completed. More specifically, the jth global iteration G_j is computed by [5]

$$G_j = (I_m + I_{m+1})/2. \tag{4}$$

If the phase map is not completely unwrapped after the jth global iteration, the local iteration steps must then be repeated again. By executing the local and the global iterations as mentioned above alternatively, the fully unwrapped phase map can then be obtained. Figs. 1 and 2 are examples of the intermediate phase maps obtained after local and global iterations. Please note that all the unwrapped phase maps used in this paper were obtained from an experiment that used a simple Michelson interferometer setup to measure the deformation of a cantilever beam loaded by a point force at the tip.

It is worthwhile to take a closer look of how Cellular Automata algorithm treats singular points. Most of the conventional path-dependent methods remove points such as the local dislocations, which may be induced in the experimental process, by using the closed-loop checking. This closed-loop checking procedure basically examines whether the phase value summed over a closed-loop equals to zero or not. Anytime the phase value does not sum to zero, the pixels within the loop are viewed as noise or local dislocation and are either neglected or deleted. There is another approach which uniformly adds the calculated phase difference value obtained after each closed-loop checking to every pixel around the loop to have all pixels of the phase map meet the phase self-consistent requirements [9,10]. However, this error checking and correcting process can sometimes cause troublesome artificial cross-sectional cut to the modified/corrected wrapped phase maps [9,10]. In contrast, the mathematical steps adopted by Cellular Automata algorithm minimizes the influence of these dislocation points by isolating them at their original locations [11].

The main advantage of the Cellular Automata method is its immunity to noise and artifacts, which can be well understood by its fundamental property such as path-independence, and non-propagation of noisy data. In the presence of noise, path dependent methods demand highly complex algorithms, which require extensive programming efforts. On the other hand, Cellular Automata algorithm can be easily programmed due to its simplicity.

Although the Cellular Automata algorithm is superior in preventing the noise from propagating and in dealing with the discontinuity problems, the long processing time needed makes it to this day far from being extensively implemented in commercially available systems. That is, despite its natural parallel-processing feature, the Cellular Automata algorithm does need tremendous computation effort which significantly lengthens the time needed to perform phase unwrapping. Implementing the Cellular Automata algorithm and then examining the time required for every step indicates that the traditional Cellular Automata algorithms waste too much time in local iterations to reach the period 2 oscillations. This phenomenon is especially severe

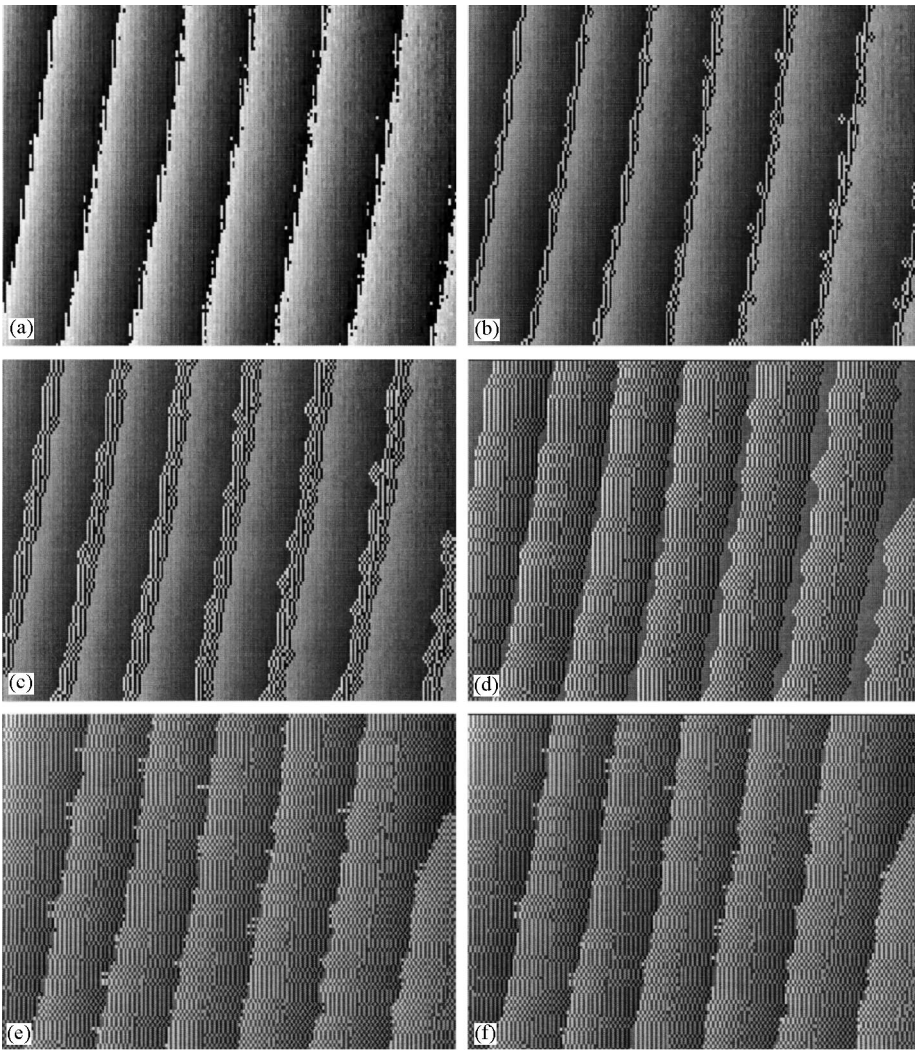

Fig. 1. Examples of Cellular Automata phase unwrapping local iteration results. (a) The original experimentally obtained wrapped phase map, which is 200 pixels wide and 100 pixels high. (b) Modified phase map after the first local iteration step. (c) Modified phase map after the second local iteration step. (d) Modified phase map after the 20th local iteration step. (e) and (f) are the two modified phase maps obtained when the local iteration reaches the period-two state requested by the Cellular Automata local iteration process.

when the phase maps contain serious inconsistencies, which may be induced by crack or obstruction on the specimen. Another thing that should be noted is that when several regions of the wrapped phase maps violate the phase consistency requirement, the local maps will fail to converge to period 2 oscillation during local iteration.

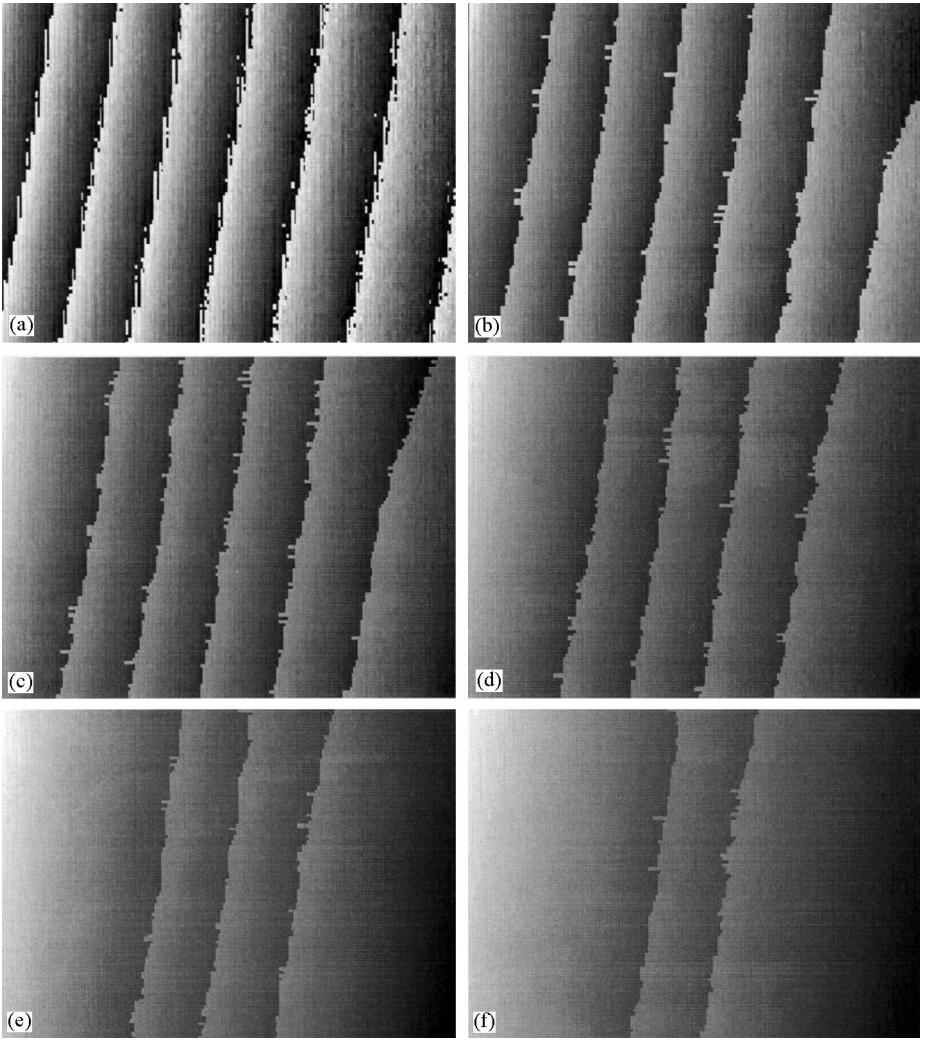

Fig. 2. Examples of Cellular Automata phase unwrapping global iteration results. (a) The original wrapped phase map, which is 200 pixels wide and 100 pixels high. (b)–(f) Modified phase maps after each successive global iteration process. It is clear that each global iteration process removes one fringe from the data set.

Instead the local iteration will converge to more than period 2 iteration, say period 3 iteration. Practical experience indicates that the number of periods actually provide some indication on the number of regions that the whole phase maps should be split in order to obtain individual phase maps that are self-consistent. This experience is important in introducing the phase masks concept commonly implemented in traditional path-dependent phase unwrapping algorithms into the Cellular Automata algorithm.

2.2. Implementation of Cellular Automata

The Cellular Automata algorithm was implemented by using the LabVIEW™ program [12] running under Microsoft Windows™ 95 platform [13]. In order to speed up the calculations, array processing techniques were implemented. All elements mentioned in Eq. (1) are introduced as arrays. The subtraction process required by Eq. (1) is performed by first creating four new arrays which are obtained by shifting the elements in the original unshifted arrays up, down, right, left. Subtracting the original unshifted map from each of the four new arrays and then summing up the final results we arrive at the results required by Eq. (1). This approach can be easily implemented by hardware such as array processors to significantly improve the computation speed. The parallel processing nature of the Cellular Automata algorithm guarantees the possibility of the above mentioned implementation approaches.

2.3. Boundary condition of the phase maps

The processing steps needed to implement boundary conditions were not discussed in detail in published literature regarding Cellular Automata. However, the boundary conditions are important to determine how the phase maps will evolve and can ultimately determine if the Cellular Automata algorithm can be effective or not. Even though the difference between the central pixel and its neighboring four pixels can be calculated by shifting and subtracting for pixels that are not located on the boundary, this process fails to operate when the central pixel is located on the boundary according to Eq. (1). This problem can be solved by setting the phase difference equal to zero whenever no data set exists to be subtracted. Thus the phase difference for all central pixels located on the boundary will be set to zero. This condition holds equally true no matter if the boundary comes from the data field itself or from the phase masks implemented by the user. With these approaches, the whole phase map can be unwrapped. The Cellular Automata algorithm will make all the jumps or defects propagate through the boundary and then disappear.

3. TCA

As it was mentioned above that Cellular Automata has many advantages against traditional path-dependent phase unwrapping algorithms despite some of its drawbacks. The TCA to be presented below tries to deal with some of the above-mentioned drawbacks while keeping the merits of the traditional Cellular Automata algorithm. The fundamental rationale lying behind the development of TCA is that most of the inefficiency of Cellular Automata algorithms comes from the time needed to move the fringes gradually to the boundary. More specifically, the local iterations of Cellular Automata gradually move the fringes to the boundary and each global iteration basically unwraps a single fringe. If the fringe is far from the boundary or if there is some dislocation points appear on the wrapped phase maps, the local iterations will either take a very long time to converge or not converge at all. Thus, if

we can split the whole wrapped phase maps into an array of sub-maps and perform steps similar to the traditional Cellular Automata algorithms to each sub-map, the time needed to move a single fringe will become much shorter. Once every sub-map is unwrapped, a stitching process can then be used to connect all the sub-maps into the whole phase maps. To make this TCA approach even more powerful, the phase masks concept will also be introduced to further effect the isolation of the noise or the dislocation points. On a higher level thinking, the TCA algorithm works like a mixed breed between the traditional Cellular Automata algorithm and the traditional path-dependent phase-unwrapping algorithms.

The new TCA process as shown in Fig. 3 will consume much less time than the original Cellular Automata method since we do not have to wait for the typically relatively smaller regions having serious discontinuities to converge on many local iterations. Furthermore, the shorter distance between each fringe and the boundary makes the local iterations time needed become much shorter. The introduction of the phase mask process and the least-squared sub-maps stitching method are part of the overall TCA algorithm that allows the user to take advantage of some information known a priori.

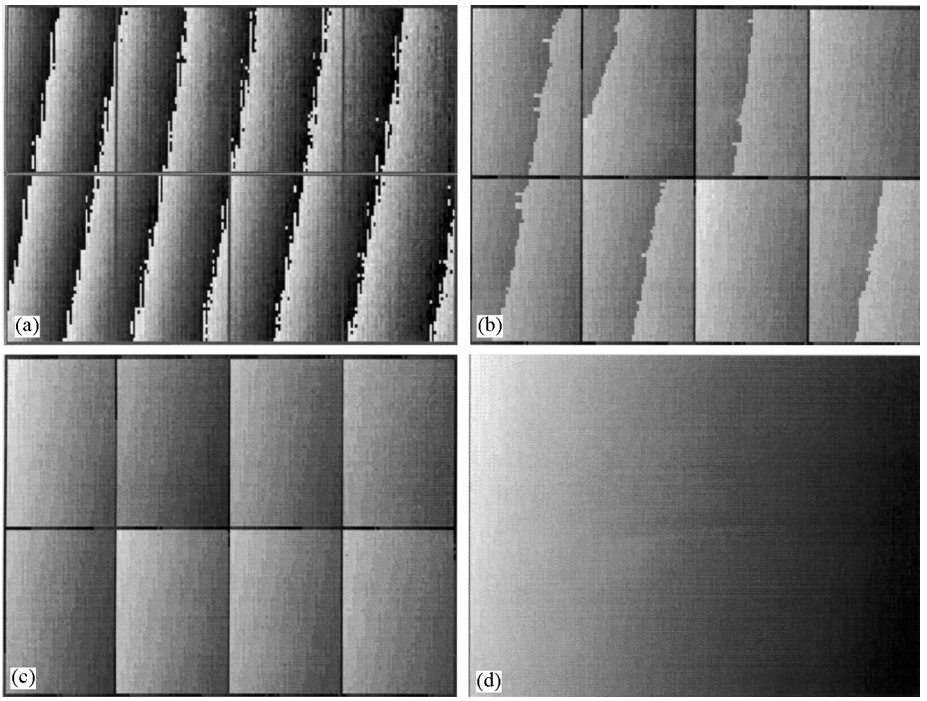

Fig. 3. Examples of TCA implementation procedures. (a) The original 100 pixels by 200 pixels experimental phase map was split into eight 50 pixels by 50 pixels sub-phase maps according to the TCA algorithm. (b) The sub-phase maps after the first global iteration. (c) The unwrapped sub-phase maps obtained after only two global iteration steps. (d) The unwrapped phase map obtained after stitching the sub-phase maps shown in (c).

3.1. Phase mask

All the issues mentioned above fail to examine the problems whenever a large area of defects or phase inconsistency over 2π caused by lateral shearing or other reasons are present in the wrapped phase maps. The time needed for each local and global iteration will be significantly increased at the presence of the fringe noise or errors. Thus, if an algorithm can provide the user with a way to mask out the bad data area before the phase unwrapping process begins in order to avoid precious processing efforts spent on these areas, the processing time can be significantly improved.

The phase mask approach presented in Fig. 4 is a programming technique typically adopted to separate the masked bad data area from the valid parts of the phase map. Please note that the phase mask is the thick rectangular line in Fig. 4a. It is used to

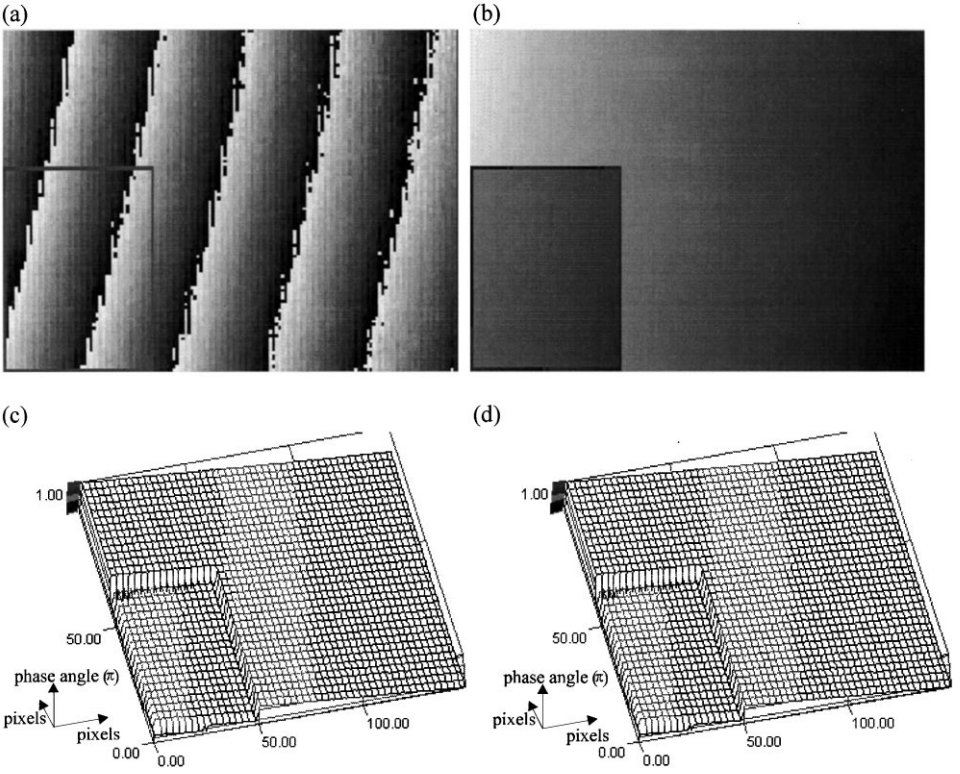

Fig. 4. Implementing the phase mask and stitching process. (a) Designating phase mask area on the 200 pixels wide and 100 pixels high wrapped phase map. Note that the rectangle designated by the dark line is the phase mask, which means that the two sub-phase regions separated by the phase mask should not be connected. (b) Unwrapped phase map of (a). It is clear that the two unwrapped phase maps are not connected. (c) 3-D presentation of (b). (d) Implementing the stitching technique to connect the two regions. This process can be adopted when the two regions are actually a whole phase map but the data area designated by the phase mask are too deterioated to be used to represent valid data set.

depict that the two sub-phase regions in Fig. 4a are not connected. Once the phase mask is designated, the user then has a chance to determine which regions of the whole wrapped phase map need to be processed. In other words, this process provides the user with an opportunity to remove the bad data set before the unwrapping process starts. For the case shown in Fig. 4, the phase mask is used to split the wrapped phase map. The TCA process is then adopted to unwrap the two regions independently. Finally, the two sub-regions are stitched as the user determines that the data set are valid and only the area covered by the phase mask are too contaminated by noise to be used.

The TCA process essentially treats the masked boundary by using the same steps as that of the regular phase maps boundary mentioned above. This strategy prevents the valid data located near the masked boundary from getting connected and influenced by the data sets located across the user designated phase masks.

3.2. TCA for phase unwrapping

The steps for TCA are proceeded by masking out the bad data areas or the areas that need not perform phase unwrapping. Once the phase maps are masked, the remaining part of the phase maps is split into an array of sub-maps (see Fig. 3a). The traditional Cellular Automata algorithm was then adopted to unwrap every sub-map in a distributed manner. The unwrapped sub-maps are then connected through each boundary in a least-squared sense. More specifically, the two neighboring sub-map edges should have minimum variations if the sub-phase maps are unwrapped correctly. The variations of the phase values along the neighboring edge can be represented by

$$\sigma^2 = \sum_{x,y\in l} [z_{i+1}(x,y) - z_i(x,y)]^2, \tag{5}$$

where the edge intersected by the two phase maps is designated by l, z_i means the unwrapped phase value of the ith sub-map, and z_{i+1} means the unwrapped phase value of the $(i+1)$th sub-map. Noted is that only the phase boundary located on the neighboring edges should be used to calculate the variations in Eq. (5).

Representing the mean value of a sub-map z_i as

$$m_i = \frac{1}{\#(l)} \sum_{x,y\in l} z_i, \tag{6}$$

where $\#(l)$ means the number of points located within the neighboring edge l. The mean value of z_i should equal the mean value of z_{i+1} in order to minimize the variation. However, if the two sub-maps are unwrapped in two different bases, the mean value may differ. The difference of mean value can be expressed as

$$\Delta m_i = m_{i+1} - m_i. \tag{7}$$

The stitching process of TCA minimizes the variations defined in Eq. (5) by adding Δm_i to all values of the sub-map z_{i+1}. That is, the stitched sub-phase map $z'_{i+1}(x,y)$ are

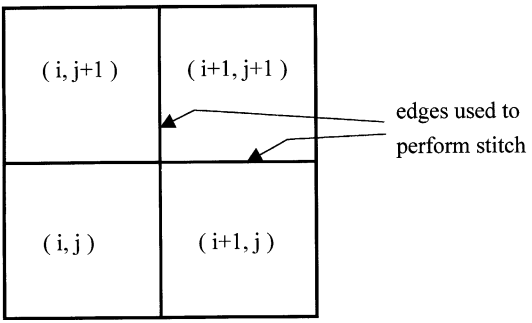

Fig. 5. Stitching sub-phase maps by fitting the edges in a least-square sense.

related to the original sub-phase map $z_{i+1}(x, y)$ by

$$z'_{i+1}(x, y) = z_{i+1}(x, y) + \Delta m_i \tag{8}$$

for every pixel within the sub-phase map z_{i+1}. This basic stitching process is executed path by path for every sub-phase map. For example, as there are four neighboring edges for the case shown in Fig. 5, four similar stitching processes must be executed. One thing that should be noted is that the paths taken to perform the stitching process will influence the final results if phase inconsistency exists within the phase map. This path-dependent effect present in the TCA stitching process is similar to that of the traditional path-dependent phase unwrapping algorithms.

Despite that some path dependency exists within the TCA process, there are some fundamental distinctions on removing phase singularity when compared to the traditional path-dependent phase unwrapping algorithms. Whenever a local iteration fails to yield a period 2 oscillation in TCA, the divergent sub-phase map obtained from the previous global iteration process can either be divided into smaller sub-phase maps or be removed by the phase mask. For the case where smaller sub-phase maps are generated, the TCA process demands local iterations be further performed to each and every new sub-phase maps generated to identify ever more precisely the locations of the phase inconsistency. This approach will not only save a lot of iteration time but also will increase the possibility of unwrapping the phase maps even if phase inconsistency exists in some regions of the phase maps.

4. Discussion of implementation

As it was mentioned above, the process used to stitch each sub-phase map into a full phase map is an extremely important issue in the TCA algorithm. During the process of stitching, an error checking process can also be implemented to further guarantee the correctness of the final unwrapped phase map. Certainly, the parallel and distributed data processing nature of the TCA algorithm move in harmony with the major trend of today's computer industry which is multi-tasking operating systems, parallel

processing, and distributed computing through the net. All of these merits make TCA a versatile algorithm for phase unwrapping.

4.1. Stitch strategy and error checking

Along every stitch line, the TCA algorithm first compares the mean boundary value of the two sides across the fictitious line generated when the full phase map was split. The difference between the two mean values was then calculated. The TCA stitches the two neighboring parts by fitting the edges using the least-squared sense if both sub-phase maps were unwrapped successfully. Whenever a closed-loop stitch fails within a zone, a global singularity region is found as shown in Fig. 6b. Once a global singularity is found, two possible strategies can be adopted to treat the singularity. The first one is that the erroneous parts be re-divided into smaller sub-phase maps for further TCA processing as shown in Fig. 6. The second strategy is to discard the whole area in question by the TCA phase mask process. The user will have a chance to trade the small erroneous areas with higher efficiency and better accuracy.

When faced with serious noise or partially invalid data, the TCA phase mask can be used to achieve speed and ease of processing. In addition, it is also a good and efficient error checking method when compared to the traditional pixel-by-pixel closed-loop error checking methods. In order to speed up the processing time and have the most noise-free area for phase unwrapping process, the TCA algorithm provides the user with a way to mask out all of the visible defects of the wrapped phase map manually before the whole unwrapping process starts.

Another situation that needs error checking is when local iteration fails to yield a period two oscillation, which is called divergent. When serious defects or singularities are encountered in the sub-phase maps unwrapping process, the sub-phase map fails to converge to a stable state. In these cases, further sub-division is a feasible way to limit the divergent area to a smaller area to be further error checked and masked out. This TCA implementation process can be performed as an intelligent error checking strategy (see Fig. 7).

4.2. Parallel and distributed processing

As the TCA algorithm possesses parallel and distributed data processing nature, the computation platform adopted to implement the TCA will certainly influence the algorithm performance. To keep pace with the rapidly advancing PC industry, which includes the multi-tasking operating system, Internet and Intranet, etc., the graphic user interfaced LabVIEW™ [12] program running under Microsoft Windows™ 95 [13] environment was chosen as our implementation platform.

By neglecting the time needed to perform TCA stitching process and by assuming that fringes distributed uniform across the whole wrapped phase map, the theoretical computation speed of TCA is compared to that of the traditional Cellular Automata in Table 1. It is clear from Table 1 that after sub-dividing the N_x pixels by N_y pixels phase map into a_x pixels by a_x pixels sub-phase maps, the computation speed in TCA is improved by $(N_x \bullet N_y)/(a_x \bullet a_y)$ times when compared with that of the original Cellular

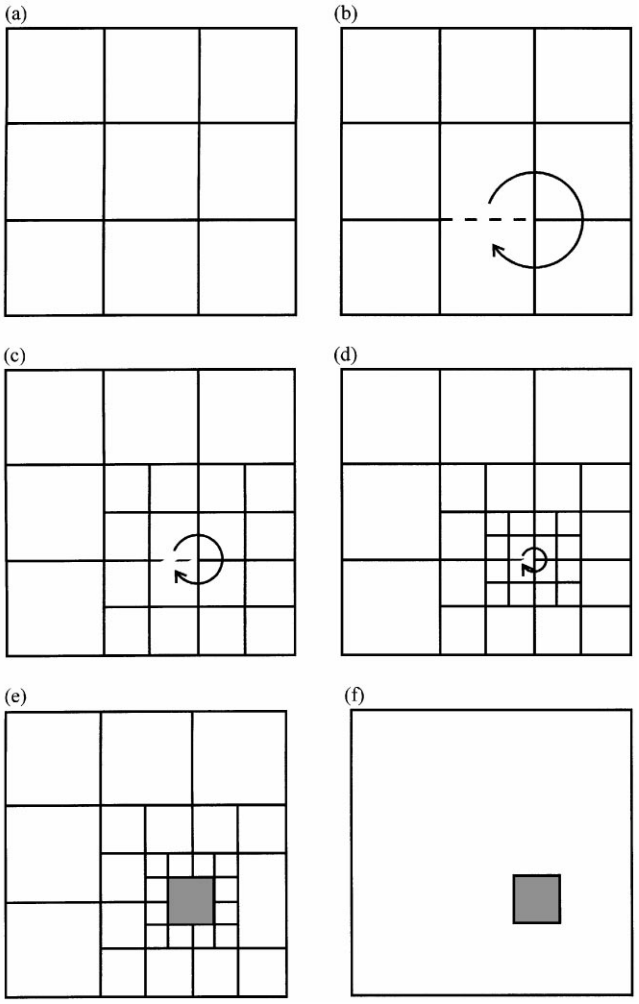

Fig. 6. A graphic representation of the TCA stitching strategy. (a) Divide the full wrapped phase map into several sub-phase maps first. (b) A closed-loop inconsistency, which was shown by the dotted line, was found while stitching the fully unwrapped sub-phase maps. (c) and (d) Further sub-dividing and closed loop error checking were performed to limit the error to a small area. (e) and (f) Once the erroneous area is recognized and masked out, the remaining phase map can be unwrapped by the TCA process.

Automata. Please note that the above-mentioned sub-dividing approach makes each sub-phase map identical in size, however, it is not necessary to make every sub-phase map identical within TCA. The optimum strategy of sub-dividing the phase map in TCA is to make the number of fringes in each sub-phase map identical as each TCA global iteration essentially removes one fringe within the data set. In other words, the fringes should be distributed uniformly within each sub-phase map. This assertion is

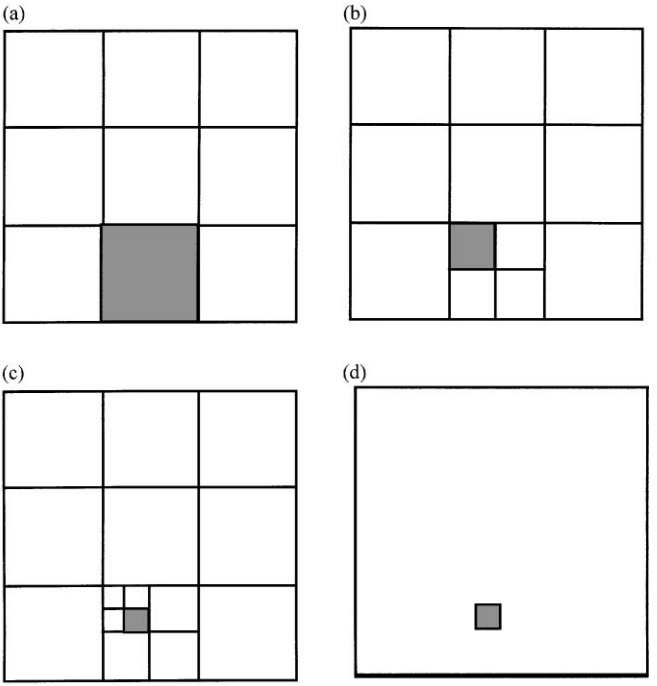

Fig. 7. A graphic representation of the TCA automatically restricting process for the divergent sub-phase map, which is any sub-phase map that fails to achieve period 2 oscillations. (a) The divergent sub-phase map is shown in black. (b) and (c) Further sub-dividing to smaller sub-phase maps. Local iterations and stitching are performed until the divergent area is equal to or smaller than the default area size. (d) The divergent area is finally recognized and masked out.

further examined by actually unwrapping the 100 pixels by 200 pixels phase map in the LabVIEW platform (Table 2). The TCA algorithm is implemented by dividing the original phase map into eight 50 pixels by 50 pixels sub-phase maps. According to Table 1, the TCA processing speed should be eight times faster than that of the Cellular Automata algorithm. The true processing speed improvement achieved by TCA is shown in Table 2 and indicates that only a factor of five improvement is achieved, which is nevertheless noteworthy. The difference can be understood as the theoretical calculation was based on uniform fringe distribution and no stitching time is needed.

In addition to the above-mentioned speed improvement merits, TCA also has the potential of being implemented in distributed computer systems. In many practical applications, very large phase maps may have to be unwrapped. By first splitting the large phase map into many smaller sub-phase maps and then computing each sub-phase map in distributed computer systems can significantly reduce the computation time and significantly lower the cost needed to construct one such phase unwrapping system. For example, to implement this strategy in a client-server personal computer

Table 1
Theoretical comparison of computation speed between Cellular Automata (CA) and TCA when each sub-phase map is split to be identical in size

	No. of pixels per sub-phase map (A)	No. of local iterations (B)	No. of global iterations (C)	Total no. of iterations (equals $B*C$)	Speed ratio (equals $A*B*C$)
CA	$N_x \bullet N_y$	$\sqrt{N_x \bullet N_y}$	L	$L \bullet \sqrt{N_x \bullet N_y}$	1
TCA	$a_x \bullet a_y$	$\sqrt{a_x \bullet a_y}$	$L \bullet \frac{\sqrt{N_x \bullet N_y}}{\sqrt{a_x \bullet a_y}}$	$L \bullet \sqrt{N_x \bullet N_y}$	$\frac{N_x \bullet N_y}{a_x \bullet a_y}$

Table 2
Experimental comparison of computation speed between CA and TCA

	No. of pixels per sub-phase map	No. of local iterations (B)	No. of global iterations (C)	Total no. of iterations (equals $B*C$)	Speed ratio
CA	100∗200	68	7	476	1
TCA	50∗50	48	2	96	4.95

environment, a centralized personal computer acting as the server can be used to split the large phase map into many sub-phase maps. Each set of sub-phase maps was then transmitted to each client personal computer by using the TCP/IP protocol through either the internet or intranet computer network. After each client personal computer finished unwrapping the sub-phase maps, the server then retrieved the unwrapped sub-phase maps from each client to be stitched into a full phase map.

5. Conclusion

It is clear from the results presented in this paper that a new path-independent phase unwrapping algorithm TCA has been developed. TCA was successfully implemented by using a graphic user interface programming language LabVIEW for ease of maintenance and for ease of system upgrade. Significant processing speed improvement was also demonstrated when compared to the traditional Cellular Automata phase-unwrapping algorithm.

Compared to the traditional path-dependent phase unwrapping algorithm, TCA can be used even when severe noise are present. In addition, the phase mask concept was also introduced into the TCA algorithm. Since phase mask operation provides users with a way to identify the erroneous data region, the success on the integration of phase mask with phase unwrapping operations is shown to both significantly

shorten the processing time and increase the accuracy of phase unwrapping operations. Furthermore, the parallel and distributed nature of TCA was shown to be in tune with the main development trend of the whole personal computer industry. It is estimated that the TCA processing time will be significantly improved with the rapid advancement of the personal computer industry and can be widely adopted in many areas such as non-destructive testing, remote sensing, holographic interferometry, etc.

Acknowledgements

This research was supported partially by the National Science Council, Taiwan, Republic of China under the contract numbers NSC 85-2622-E-002-017R, NSC 86-2622-E-002-023R, NSC 86-2815-C-002-006-E, NSC 86-2815-C-319-001-E.

References

[1] Creath K. Phase shifting speckle interferometry. Appl Opt 1985;24:3053–58.
[2] Robinson DW, Reid GT. Interferogram analysis: digital fringe pattern measurement techniques. Bristol, Great Britain: IOP Publishing Ltd. 1993.
[3] Tribolet JM. A new phase unwrapping algorithm. IEEE Trans Image Process ASSP-1977;25(2):170–7.
[4] Wolfram S. Cellular Automata, Los Alamos Science, No. 9. Los Alamos, NM: Los Alamos National Laboratory, 1983.
[5] Ghiglia DC, Mastin G, Romero LA. Cellular-automata method for phase unwrapping. J Opt Soc Am 1987;4:267–80.
[6] Spik A, Robinson DW. Investigation of the Cellular Automata method for phase unwrapping and its implementation on an array processor. Opt Lasers Engng 1991;14:25–37.
[7] Ghiglia DC, Romero LA. Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods. J Opt Soc Am 1994;11(1):107–17.
[8] Marroquin JL, Rivera M. Quadratic regularization functionals for phase unwrapping. J Opt Soc Am 1995;12(11):2393–400.
[9] Takajo H, Takahashi T. Least-squares phase estimation from the phase difference. J Opt Soc Am 1988;5(3):416–25.
[10] Takajo H, Takahashi T. Noniterative method for the exact solution for the normal equation in least-squares phase estimation from the phase difference J Opt Soc Am 1988;5(11):1818–27.
[11] Scivier MS, Fiddy MA. Phase ambiguities and the zeros of multidimensional band-limited functions. J Opt Soc Am 1985;2(5):693.
[12] LabVIEW for Windows User Manual. National Instruments Corporate Headquarters, Austin, TX, December 1993.
[13] Microsoft Windows™ 95 Resource Kit. Microsoft Corporation, Redmond, Washington. Also, U.S. Patent No. 4,955,066, 1995.