A Quad-Tree Based Phase Unwrapping Algorithm

A. Baldi, F. Bertolino, F. Ginesu

Department of Mechanical Engineering, University of Cagliari
Piazza D'armi, 1 – 09123 Cagliari, Italy
e-mail: baldi@iris.unica.it

Abstract

One problem to be tackled when using interferometric phase shifting technology, is how the phase can be reconstructed. On account of the fact that an arc tan function appears in the formulation, the final data is not the phase, but the phase modulo 2π. In this paper the authors present a new phase unwrapping algorithm based on a two-step procedure. In the first step, the digital image to be analysed is divided into patches using a Quad-Tree like recursive procedure, in the second step the single patches are joined together to obtain larger, continuous ones. The basic idea of the procedure is to simplify the problem by factoring the complete image into "homogeneous areas" (i.e. areas where each pixel phase differs by less than π from its neighbours) so that only interfaces need to be dealt with. On account of the consistency of the data, the macro-areas assembled in this way are also homogeneous, so the procedure ends up with a fully unwrapped representation of the image.

After a complete description of the algorithm, some examples of its use on synthesised digital images, with and without noise, are illustrated. Images derived from experimental data are also shown. Lastly the solution time has been examined. As the algorithm combines the local and global approach, it has proved to be fast and reliable.

1. INTRODUCTION

Reconstructing the correct phase field from a grid of modulo 2π data is the final step in many algorithms for digital image analysis. Most interferometric techniques generate a fringe pattern—a cosinusoidal phase function modulated by the physical quantity being measured—for which only the main values, lying in the [-π,π) interval, are known. Because trigonometric functions are used in the analysis procedure, a final step, known as "phase unwrapping", must be performed on the experimental data to obtain indirectly the original continuous field by removing the 2π discontinuities.

The phase unwrapping problem can be easily solved for phase maps obtained from good quality fringe data. If there is no aliasing problem, all the absolute phase differences are less than π except for the expected 2π discontinuities. Thus, starting from an image corner, it is possible to follow each row/column adding/subtracting 2π every time a jump greater than -π/π is detected. This procedure can fail if noise is present: a mis-detected jump sometimes

propagates through the rest of the image producing a completely wrong result. In this case a much more robust algorithm needs to be used.

Over the past years much work has been done to improve the noise immunity of unwrapping algorithms and several authors have proposed solutions: the algorithms can be grouped into four classes: **1**) path-dependent algorithms, **2**) path-independent algorithms, **3**) tile/regions combining algorithms and **4**) global algorithms.

All the algorithms belonging to the first class (linear scanning, spiral scanning, multiple scan directions…)[1] use an a-priori-defined search path. They are usually very fast, but unable to handle noisy images because of the fixed data evaluation order. The fixed path limitation is overcome by the 2nd class algorithms[2–6]: again they work on a sequential base, but the unwrapping order is established by the pixel reliability. In this way the analysis of unreliable data can be postponed and error propagation probability is minimised. They perform well, even if they are computationally and algorithmically intensive.

The algorithms in the 3rd class[7–8] adopt a completely different approach dividing the data field into homogeneous areas which are subsequently re-assembled so as to minimise the phase differences at their interfaces. How these areas are identified varies: the *regions* family generates homogeneous areas based on phase gradients while the *tile* family divides the original image into a grid of smaller areas which are unwrapped using a simpler algorithm (usually a path dependent one). Both families are known to be very robust even if the image division procedure can be difficult, computationally intensive and error prone (an interrupted fringe could generate just one region where two should be present).

The algorithms in the 4th class formulate the unwrapping problem in terms of the least square minimisation of a global functional[9–16]. While the error function is quite similar for all the methods, the "normal equation" solution algorithm differs, ranging from an algebraic solution[9–10], to a simulated annealing method[11] to a discrete transform method[12–13,15]. This class usually includes the *cellular automata method*[16][1] even if it has no conceptual relation with the others. All the algorithms of this class are computationally intensive but their global approach compensates for the longer execution time.

In the following we will describe a new unwrapping algorithm which could be classified in the *region/tile* family. Similarly to the *tile algorithm* it divides the original data into square sets, but these sub-areas are internally homogeneous and assembled in a *region algorithm* like way.

2. THE QUAD-TREE UNWRAPPING ALGORITHM

As previously mentioned, the relation between the wrapped phase $\varphi(i, j)$ and the unwrapped phase $\psi(i, j)$ is quite simple:

$$\psi(i, j) = \varphi(i, j) + 2\pi n(i, j) \tag{1}$$

where $n(i, j)$, an integer multiplier field, should be chosen in such a way as to minimise the phase error ε over the full $N \times M$ domain:

$$\begin{aligned} \delta_x &= [-1,0,1,1,1,0,-1,-1] \\ \delta_y &= [-1,-1,-1,0,1,1,1,0] \\ \varepsilon &= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{8}\left|\psi(i, j) - \psi\left(i + \delta_x(k), j + \delta_y(k)\right)\right| \end{aligned} \tag{2}$$

where terms outside the working domain do not contribute to the summation. Within an homogeneous areas—a i^*, j^* not simply connected sub-domain, where $\forall i \in i^*, j \in j / |\varphi(i,j) - \varphi(i \pm 1, j \pm 1)| < \pi$ — all $n(i,j)$ will be the same, so that only a global n that minimises phase differences at the interfaces has to be chosen. This is the basic concept of the *region algorithms*, but it is evident that in presence of noise, the region identification is a difficult and error prone procedure, usually handled by statistical methods[7].

The *tile method*, based on the assumption that smaller areas should be easier to unwrap, by-passes the region identification problem using a simpler unwrapping algorithm to artificially generate square homogeneous areas. This procedure does away with the region identification step and simplifies the evaluation of interfaces energy, but binds the overall performance to the low level unwrapping algorithm and to the user defined tile size.

The *region method* uses an advancing front algorithm to identify homogeneous areas: starting from a pixel, it analyses the relation between the boundary pixels and their external neighbours, evaluating the probability that a pixel belongs to the current area on the base of both phase jumps and number of current class neighbours. It is evident that this algorithm is complex (one must maintain multiple lists to describe region boundaries) and that two areas could coalesce in the case of missing fringes.

To overcome these problems one could adopt a bottom-up approach: noting that a single pixel is obviously homogeneous, one can build a 2×2 homogeneous area just checking the "interfaces" of these 4 areas; a 4×4 area can be built using four 2×2 macro-pixels, a 8×8 area can be built using four 4×4 macro-pixels and so on. The procedure terminates with a single unwrapped region resulting from the combination of 4 square homogeneous sub-regions. Using this approach, region identification is not a problem and there is no dependency on tile unwrapping method and size. On the other hand one has to perform interface checking even in homogeneous areas because of the bottom-up approach.

However, the same algorithm can be implemented by checking the full image for phase jumps and if necessary dividing it. By recursively performing the same checks on each sub-image (and dividing it, if necessary), one can identify, at the end of the division procedure, four homogeneous, square, sub-areas (at the most 4 pixels), that can be assembled using the previously described algorithm. This homogeneous area can then be combined with its three same-size neighbours (eventually resulting from a similar process) to generate a larger, almost homogeneous area. The latter can be assembled with its neighbours to generate an even larger area, and so on up to the full image. In particular the algorithm works as described below:

a) For each pixel within a working area, check all the neighbours for phase jumps; **if** there is no $|\Delta\varphi| > \pi$ **then** return. **else**
b) Divide current area into 4 sub-areas,
c) for each sub-area recursively call the unwrapping algorithm,
d) assemble the 4 homogeneous areas minimising phase jumps at the interfaces only.
e) **return**.

Note that a single pixel always succeeds at step **a**) and that, on account of the recursive call at **c**),all the sub-areas are homogeneous once we arrive at step **d**): in fact either they are homogeneous (return from step **a**) or they result from a deeper level combination (step **e**).

Using this algorithm the initial image is divided into square, variable size tiles. Each tile is homogeneous and can easily be combined with neighbours of the same size by checking phase jumps at the interfaces only.

The tile sizing problem and the dependency on the low level unwrapping algorithm of the *tile method* are completely solved. The homogeneous area identification problem of the *region algorithm* is also solved, while the misidentification problem of homogeneous areas in presence of missing fringe is only partially solved. In any case, even if a tile can be erroneously assumed to be homogeneous by the dividing algorithm, the recombination method, which at each step compares the interfaces of 4 areas, intrinsically limits the error propagation probability; moreover, since each tile, once assembled, forms part of a larger one, it moves with it, so that the overall result is almost always correct and the errors remain local.

A minor drawback of the Quad-Tree algorithm is that it requires square images with a power of two size. Although this could appear to be a serious problem this is not so because of the weighting procedure (described below) which allows to disable added borders.

2.1. Tile combining algorithm

The above algorithm can be viewed as a two step method: the *dividing* step, to identify homogeneous areas in the original image, and the *recombining* step, to generate a larger, homogeneous area. Although this classification is somewhat artificial—image tessellation and tile recombination are not sequential but intermixed—these operations are nevertheless essentially different. In the first step, the algorithm checks for phase jumps looking for homogeneous areas; in the second step these areas are combined to obtain larger ones.

The area recombination step attempts to minimise the phase jumps at the interfaces, that is it attempts to find, for each sub-area, a global n (n_1, n_2, n_3, n_4) so as to minimise $\sum \Delta\varphi$.

Since we are not concerned with external interfaces, we have to take care of relative position only, so n_1 can safely be assumed to be null. By contrast n_2, n_3 and n_4 can in principle be any integer value as any sub-area is processed separately, so we chose an incremental approach:

a) put $N_2 = N_3 = N_4 = 0$;
b) for each $n_2, n_3, n_4 \in [0,1,-1]$, calculate $E = \sum \left|\Delta\varphi\left(n_2 + N_2, n_3 + N_3, n_4 + N_4\right)\right|$;
c) if the E minimising combination is not $n_2 = n_3 = n_4 = 0$ then
   d) add n_2, n_3, n_4 to N_2, N_3, N_4;
   e) goto b;
f) add N_2 (N_3, N_4) to each *n(i,j)* of the sub-area.

Note that since each n_i can take one of three possible values, we have to perform 27 (3^3) phase jump evaluations at step **b**), which is therefore one of the most time consuming phases of the Quad-Tree algorithm.

2.2. Improving unwrapping by weighting

Experimental images often show low modulation/highly noisy areas so that a pixel disabling procedure is necessary. The procedure is easily implemented by means of a phase weighting matrix *W(i,j)*.Since the unwrapping process is driven by the minimisation of phase jumps at sub-areas interfaces, the use of a nil pixel weight automatically nullifies its contribution to the summation. Given $\vartheta(i, j) = W(i_0 + i, j_0 + j)\psi((i_0 + i, j_0 + j))$, we obtain

$$
\begin{aligned}
E = & \sum_{k=1}^{N}\left\{\left|\theta\left(k,\frac{N}{2}\right)-\theta\left(k,\frac{N}{2}+1\right)\right|+\left|\theta\left(\frac{N}{2},k\right)-\theta\left(\frac{N}{2}+1,k\right)\right|\right\} \\
+ & \sum_{i=1}^{N-1}\left\{\left|\theta\left(i,\frac{N}{2}\right)-\vartheta\left(i+1,\frac{N}{2}+1\right)\right|+\left|\theta\left(i+1,\frac{N}{2}\right)-\vartheta\left(i,\frac{N}{2}+1\right)\right|\right\} \\
+ & \sum_{j=1}^{N-1}\left\{\left|\theta\left(\frac{N}{2},j\right)-\theta\left(\frac{N}{2}+1,j+1\right)\right|+\left|\theta\left(\frac{N}{2},j+1\right)-\theta\left(\frac{N}{2}+1,j\right)\right|\right\}
\end{aligned}
\tag{3}
$$

where N is the size of the area to be assembled and i_0, j_0 are the co-ordinates of its origin.

Note that **a**) by defining an appropriate weighting mask it is possible to isolate one (or more) area from the others and **b**) inside a completely masked out area there will be no unwrapping since all the sub-area configurations will give a null E.

The weighting matrix can be used to improve unwrapping too. If phase reliability data are available, it is possible to use a continuously varying weight, rather than a digital one, to drive the unwrapping process. Numerical experiments show that a good weight can be obtained combining phase modulation $\gamma(i,j)$ and local phase regularity:

$$
W(i,j)=\gamma(i,j)\left[1-\frac{\left|\varphi(i,j)-\frac{1}{8}\sum_{k=1}^{8}\varphi\left(i+\delta_x(k),j+\delta_y(k)\right)\right|}{2\pi}\right] \tag{4}
$$

Since φ and γ result from the same calculation, an erroneous phase is sometimes coupled with an erroneous modulation. This formulation requires a high γ *in* a locally regular area thus balancing both errors. Note that the geometrical term in (4) needs to be calculated twice: if the pixel to be weighted is near a 2π discontinuity, there will be a large gap between pixel phase and mean local phase even if we analyse a noise free image. To avoid this problem it suffices to add π to all phases, thus shifting phase discontinuities, and re-calculate W. Of the two possible results, the greatest will obviously be used.

2.3. Implementation Notes

The performance of the Quad-Tree unwrapping algorithm described in the previous sections can be easily optimised. The attention should be focused on two main points:

1) during the division step, the algorithm carries out redundant homogeneity tests. Suppose we analyse a 128×128 image. A phase discontinuity between pixel (2,2)-(2,3) will be checked during the 1st pass (causing a recursive call), then during the analysis of the (1,1--64,64) sub-image, during the analysis of the (1,1--32,32) sub-image and so on up to the (1,1--2,2) sub-area.
2) During each (sub)assembling step, the algorithm performs several interface "energy" evaluations, each requiring significative computation time due to phase jumps calculation and data retrieval.

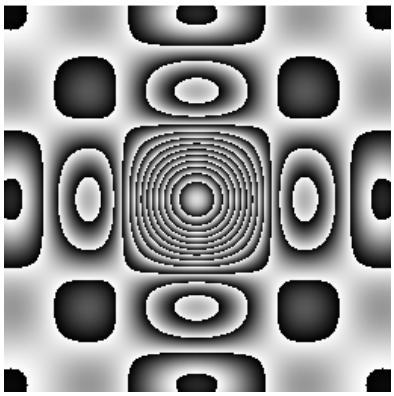
Concerning the first point, it should be observed that, although the algorithm requires the detection of phase jumps to correctly unwrap the phase field, a knowledge of the effective jumps, once they are known to be larger than π, is useless. For each point, it suffices to know which of the eight neighbours cannot be grouped with the current one without phase correction. This information can be efficiently stored in a byte matrix $C(i,j)$, which can be

initialised before the recursive step. In this way, to conduct the homogeneity check it suffices to verify that $C(i, j) \neq 0$, inside the working area, while on the boundaries, a slightly more complex analysis, $(C(i, j) \& k_s) \neq 0$, must be performed (k_s is a constant bit mask depending on working side and data encoding).

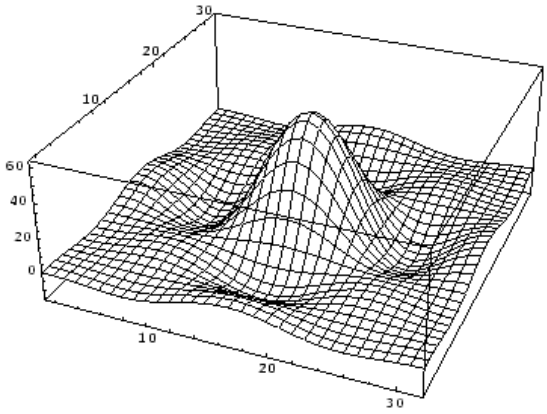
Interface energy evaluation cannot be optimised in the same way because phase jumps must be known. However, for each assembly, it is possible to store phase values in local arrays. This data indexing simplification, coupled with careful coding, greatly enhances code

2. NUMERICAL AND EXPERIMENTAL RESULTS

Let us start with a simple synthetic image: in figure 1 and 2 we show a 256×256 *sinc()* function:



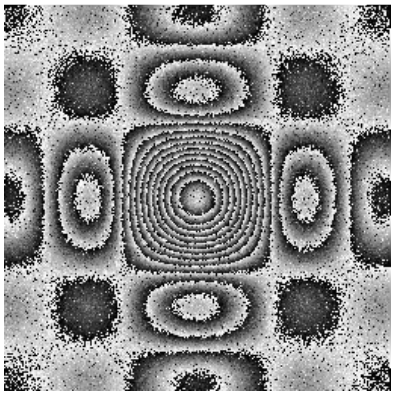
1. Wrapped sinc function



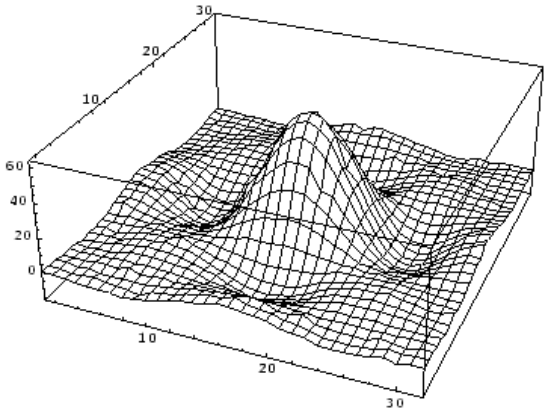
2. Unwrapped sinc function

we did not add any noise to the phase modulo 2π since our aim was to show the theoretical correctness of the algorithm. Nevertheless the test is significant because of the variable fringe densities.

Adding some noise to the image (figure 3) does not noticeably alter the previous result: there are some small errors near the sinc top (figure 4), but they are only deviations from the overall trend. This behaviour is typical of the Quad-Tree algorithm because of the method used to assemble the tiles.



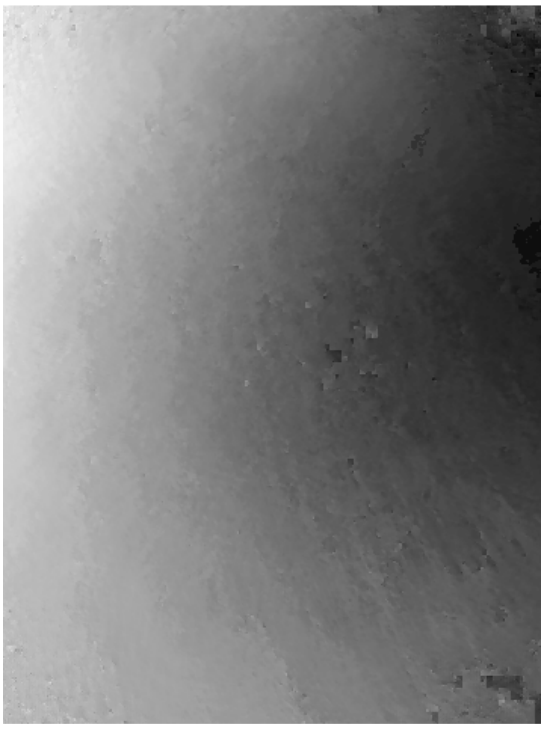
3. Wrapped noisy sinc function



4. Unwrapped noisy sinc function

Figure 5 shows an experimental fringe field of a mirror surface. This was obtained using a Twiman-Green interferometer, a He-Ne laser (λ = 632.8 nm) and a 4 images, π /2 phase shifting algorithm.

5. Mirror surface fringe system



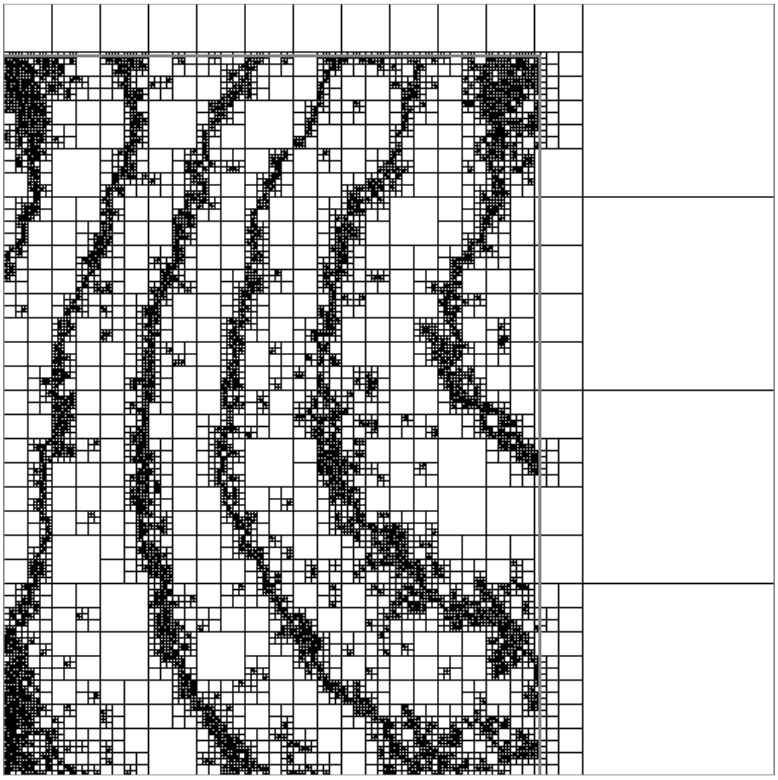
6. Unwrapped mirror surface

The unwrapped result (figure 6) is fairly good even compared with the original wrapped phase.

2. SOME CONSIDERATIONS ON TIME AND CONCLUSIONS

Short solution time is a minor requirement for unwrapping algorithms. Nevertheless the Quad-Tree algorithm uses a unique recursive approach, requiring little analysis to be performed.

As mentioned previously, each tile is divided during the homogeneity-test step until all phase differences are less than π. This means that, even in the same image, the end-of-recursion tile dimensions are not the same since they obviously depend on signal-to-noise ratio and fringe density. Figure 7 shows the Quad-Tree generated when analysing the mirror surface. The image side has been expanded to fulfil algorithm requirements, nevertheless the expansion procedure is clearly not a problem. On the contrary, the fringe density and the signal-to-noise ratio can significantly affect the execution speed. Even if tree depth can never be greater than $\log_2(n)$ (n is the size of image side) it expands and shrinks several times depending on phase smoothness, so that it is not possible to predict execution time without input data analysis



7. Quad-Tree generated during mirror surface analysis

Unwrapping time for the mirror image (477×356 pixel, mapped to a 512×512 square) was about 1.2~sec. on a K6/300 MHz processor, compared to 1.95 sec required for a single iteration of the Ghiglia and Morero DCT method[13], using the same machine.

We have presented a two-dimensional phase unwrapping algorithm based on a two-step procedure: **a**) the division of higher fringe density/noisy areas into smaller, almost homogeneous parts and **b**) their recombination using global minimisation at the interfaces. Preliminary numerical experiments show the new algorithm to be very fast and robust: its performances on images with low and medium noise level are very good; comparative studies with other algorithms for highly noisy images are in progress.

REFERENCES

[1] D. W. Robinson “Phase unwrapping methods” in *Interferogram analysis: Digital Fringe Pattern Measurement Techniques*, D. W. Robinson and G. T. Reid Eds., pp. 194–229, Institute of Physics Publishing, Bristol and Philadelphia, 1993.

[2] J. M. Huntley “Noise immune phase unwrapping algorithm” *Applied Optics* **28** (15) pp. 3268–3270, 1989.

[3] N. H. Ching, D. Rosenfeld and M. Braun “Two-dimensional phase unwrapping using a minimum spanning tree algorithm” IEEE Trans. Image process.**1** pp. 355–365, 1992.

[4] J. M. Huntley and H. Saldner “Temporal phase unwrapping algorithm for automatic interferogram analysis” *Applied Optics* **32** pp. 3047–3052, 1993.

[5] R. Cusack, J. M. Huntley and H. T. Goldrein “Improved noise-immune phase unwrapping algorithm” *Applaid Optics* **28** pp. 781–789, 1995.

[6] M. Takeda and T. Abe “Phase unwrapping based on maximum cross-amplitude spanning tree algorithm: a comparative study” in *Interferometry VII: Techniques and Analysis* M. Kujawinska, R. J. Pryputniewicz and M. Takeda Eds., *Proc. SPIE 2544* pp. 122–129, 1996.

[7] J. J. Gierloff “Phase unwrapping by regions” in *Current Development in Optical engineering II*, R. E. Fischer and W. J. Smith Eds., *Proc. SPIE* **818**, pp. 2–9, 1987.

[8] K. M. Hung and T. Yamada “Phase unwrapping by regions using least-square approach” *Optical Eng.* **37**(11),pp. 2965–2970, 1998.

[9] B. R. Hunt “Matrix formulation of the reconstruction of phase values from phase differences” *J. Opt. Soc. Am.* **69**pp. 393–399, 1979.

[10] H. Takajo and T. Takahashi “Noniterative method for obtaining the exact solution for the normal equation in least square phase estimation from phase difference” *J. Opt. Soc. Am. A* **5**, pp. 1818–1827, 1988.

[11] L. Guerriero, G. Nico, G. Pasquariello and S. Stramaglia “New regularization scheme for phase unwrapping” *Applied Optics* **37 (**14) pp. 3053–3058, 1998.

[12] G. Fornaro, G. Franceschetti, R. Lanari and E. Sansosti “Robust phase-unwrapping techniques: a comparison” *J. Opt. Soc. Am. A* **13**, pp. 2355–2366, 1996.

[13] D. C. Ghiglia and L. A. Romero “Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods” *J. Opt. Soc. Am. A* **11**, pp. 107–117, 1994.

[14] M. Servin, J. L. Marroquin, D. Malacara and F. J. Cuevas “Phase unwrapping with a regularized phase-tracking system” *Applied Optics* **37**(10), pp. 1917–1923, 1998.

[15] G. H. Kaufmann, G. E. Galizzi and P. D. Ruiz “Evaluation of a preconditioned conjugate-gradient algorithm for weighted least-square unwrapping of digital speckle-pattern interferometry phase maps” *Applied Optics* **37** (14), pp. 3076–3084, 1998.

[16] D. C. Ghiglia, G. A. Mastin and L. A. Romero “Cellular-automata method for phase unwrapping” *J. Opt. Soc. Am. A* **4** (1), pp. 267–280, 1987.