Efficient multiscale phase unwrapping methodology with modulo wavelet transform

DAVID BLINDER,1,2,* HEIDI OTTEVAERE,3 ADRIAN MUNTEANU,1,2 AND PETER SCHELKENS1,2

1Vrije Universiteit Brussel, Department of Electronics and Informatics (ETRO), Pleinlaan 2, B-1050 Brussels, Belgium
2iMinds, Technologiepark 19, B-9052 Zwijnaarde, Belgium
3Vrije Universiteit Brussel, Department of Applied Physics and Photonics, Brussels Photonics Team (B-PHOT), Pleinlaan 2, B-1050 Brussels, Belgium

*dblinder@etro.vub.ac.be

Abstract: Many robust phase unwrapping algorithms are computationally very time-consuming, making them impractical for handling large datasets or real-time applications. In this paper, we propose a generic framework using a novel wavelet transform that can be combined with many types of phase unwrapping algorithms. By inserting reversible modulo operators in the wavelet transform, the number of coefficients that need to be unwrapped is significantly reduced, which results in large computational gains. The algorithm is tested on various types of wrapped phase imagery, reporting speedup factors of up to 500. The source code of the algorithm is publicly available.

© 2016 Optical Society of America

OCIS codes: (100.5088) Phase unwrapping; (100.7410) Wavelets; (100.5090) Phase-only filters.

References
Phase unwrapping is a technique for inferring the absolute phase from a signal from the retrieved measured phase, which is constrained to its principal value, i.e. between $-\pi$ and $\pi$. Many imaging technologies across various scientific disciplines require phase unwrapping to produce the sought underlying absolute phase: Interferometric Synthetic Aperture Radar (InSAR) [1], optical interferometry and holography, Magnetic Resonance Imaging (MRI) [2], Fringe Projection Profilometry (FPP) [3], and others.

In the one-dimensional case, Itoh [4] has shown that the absolute phase can be recovered by integrating over the wrapped phase differences when assuming continuity. In ideal conditions, this method can be applied on higher-dimensional wrapped phase maps when the so-called Itoh condition is not violated (stating that phase differences between neighboring pixels should not exceed $\pi$ in magnitude). This is rarely the case in practice, due to a multitude of phenomena such as noise, undersampling, discontinuities, phase singularities and invalid image areas. These phenomena will induce errors in the unwrapped phase if not modeled and handled correctly. For this reason, most unwrapping algorithms are 2D (or higher-dimensional) and attempt to account for these various phenomena.

Over the years, many 2D phase unwrapping algorithms have been proposed; they can mainly be classified into path-dependent [2, 5–8] and path-independent [9–13] algorithms. Path-dependent (or path-following) methods will attempt to unwrap the phase along a specific path, relying on the assumption that Itoh’s condition holds along the chosen path. The path is determined by evaluating localized features in the wrapped phase, often combined with a quality map [14] to give lower weights to noisy or corrupted phase values for improving the fidelity of the unwrapped phase map. These methods are usually computationally efficient and can often handle discontinuities, but the selection of the unwrapping path is difficult and can lead to unwraping defects. Path-independent algorithms will attempt to minimize a global function over the complete candidate unwrapped phase maps. Often, this global function is chosen to be the $L^p$-norm over the phase differences [11]; other algorithms use regularization approaches [12] or parametric models [10] to constrain the unwrapped phase to a parametric surface. Path-
independent algorithms are usually robust to noise, but do sometimes make the unwrapped phase overly smooth and generally take longer to compute.

Robust phase unwrapping algorithms come at a the cost of being computationally demanding and can take from minutes up to hours even for moderate image sizes [14, 15]: this becomes problematic for many applications and limits the choice of available unwrappers. Some examples include real-time applications, such as live 3D-shape measurements [16] where unwrapping at video rate is needed, InSAR and InSAS [17] where increasing resolution makes computation time grow rapidly, and magnetic resonance elastography where the computation can even take entire days to process the large 3D datasets [2].

Current approaches for accelerating phase unwrapping often focus on specific existing unwrapping algorithms, either by augmenting them with additional processing steps [18], by carefully designing the data structures for representing the phase data [19] or by using GPU acceleration [20–22].

For this reason, we propose a novel multiscale unwrapping methodology, based on an earlier introduced modulo wavelet transform [23], that can be applied on the wrapped phase data and that can significantly accelerate many existing phase unwrapping algorithms. The code has been made publicly available. We have accelerated both a path-dependent [6] and a path-independent [11] unwrapping algorithm.

The paper is structured as follows: in section 2, we will introduce the modulo wavelet transform and in section 3 we will subsequently describe the multiscale phase unwrapping methodology. Section 4 will provide the experimental validation of the advocated principles and finally, section 5 will provide final conclusions.

2. Modulo wavelet transform

The problem of phase unwrapping can be formally summarized as follows: given a phase signal of the form $e^{iF(\vec{x})}$ with $F : \mathbb{R}^n \rightarrow \mathbb{R}$, we want to retrieve the function $F(\vec{x})$. Since a complete phase rotation, i.e. $2\pi$, will leave a complex number unchanged, we only know the principal, wrapped value of the phase signal $\Phi(\vec{x})$, namely:

$$\Phi(\vec{x}) = M(F(\vec{x})) \quad (1)$$

where $M$ denotes the symmetric modulo operator, returning the remainder after division of $2\pi$ centered around 0 (formally, $M(x) = 2\pi \{ \frac{x}{2\pi} \} - \pi \in [-\pi, \pi]$ with $\{x\} = x - \lfloor x \rfloor$). In the general case, it is impossible to unambiguously retrieve $F$ based upon the wrapped samples of $\Phi(\vec{x})$, unless some additional assumptions on the properties of $F$ - such as sampling density, limits on the gradient magnitude of $F$ and signal noise characteristics - are made.

To reduce complexity, the intent is to reduce the number of wrapped samples by transforming the phase signal and solving the unwrapping problem in the transform domain, so that unwrapping algorithms need only to process a fraction of the original input data. This can be achieved by utilizing a multiscale representation that allows to unwrap the signal at coarser scales for regions with smooth phase transitions and at finer scales for (noisy) regions with densely packed fringe patterns. A class of transforms highly suitable for multiscale analysis are wavelet transforms that allow for simultaneous time (or space) and frequency analysis of signals. The wavelet transform is a powerful tool that generates a sparse representation of the signal, which enables highly efficient signal analysis and compression [24]. 2D phase distributions can be processed as regular images, with sample values ranging from $-\pi$ to $\pi$ and be decomposed accordingly using standard wavelet transforms. However, local phase transitions from $-\pi$ to $\pi$ will be considered as sharp signal transitions, although these transitions are mostly attributable to the signal wrapping. Consequently, incorrect low-pass and high-pass coefficients will be produced at locations affected by phase wrapping phenomena.
To solve these problems, we have recently proposed an alternative wavelet transform, called the “modulo wavelet transform” [23]. This transform takes the wrapping into account by considering the wrapped phase values as elements of the circle group $\mathbb{T}$. In [23], we have proposed a modulo transform equivalent of the simplest discrete wavelet transform, namely the Haar wavelet. The wavelet bases of this transform are low-pass and high-pass filters, with a filter length of 2 samples, that correspond to the circular mean and the phase difference, respectively. These operations are reversible, and the modulo wavelet transformed representation has been shown to be more effective for wrapped phase data, resulting in better data compressibility over regular wavelet transforms using a JPEG 2000 codec [23].

Building on our findings in [23], in this paper we explore the idea of synthesizing the Haar modulo wavelet transform using the lifting scheme [25]. Wavelet transforms can be implemented with the lifting scheme, which allows for a very efficient computation of wavelet coefficients, as well as enabling the definition of more general classes of wavelet transforms, including potentially even non-linear operations. The (generalized) lifting scheme can be seen as a dyadic transform consisting of a cascade of reversible operations. After separating the signal into its even ($s^0$) and odd ($d^0$) samples, a series of $m$ (potentially non-linear) reversible functions $P_j$ (prediction) and $U_j$ (update) are applied alternatingly elementwise on $s^0$ and $d^0$:

$$d^{j+1} = P_j(s^j, d^j) \quad \text{and} \quad s^{j+1} = U_j(s^j, d^{j+1})$$

(2)

The inverse operations (denoted with ‘') are given by:

$$d^j = P_j'(s^j, d^{j+1}) \quad \text{and} \quad s^j = U_j'(s^{j+1}, d^{j+1})$$

(3)

with the superscript $j$ indicating the transform stage. Since the number of samples remains constant and all the constituents of the lifting scheme are reversible, the entire scheme is reversible as well. The final outputs $s^m$ and $d^m$ will correspond to the lowpass and highpass signals respectively.

In our approach, we propose to deploy the modulo operators as shown in Fig. 1. In this way, the Haar lifting scheme was empowered to handle phase data properly. The filter bank operations on a one-dimensional signal $s_i$ (contained by the low-pass band $L_i$) are defined as follows. First, a lazy wavelet transform will split the signal $s_i$ into odd $s^0_{i+1}$ and even $d^0_{i+1}$ samples. The subscript $i$ indicates the resolution level. The superscript 0 identifies the stage of the lifting scheme operations, i.e. before or after the prediction and update steps. That is:

$$s^0_{i+1}(n) = s_i(2n + 1)$$

(4)

$$d^0_{i+1}(n) = s_i(2n)$$

(5)
Subsequently, the prediction step will predict the odd set $s_{i+1}^0$ from the even set of samples $d_{i+1}^0$ and store the results, i.e. the high-pass coefficients, in $d_{i+1}$:

$$d_{i+1}(n) = M(d_{i+1}^0(n) - s_{i+1}^0(n)).$$

(6)

Finally, the update step will generate the low-pass samples $s_{i+1}$ based upon the odd samples $s_{i+1}^0$ and the high-pass coefficients $d_{i+1}$:

$$s_{i+1}(n) = M(s_{i+1}^0(n) + \frac{1}{2}d_{i+1}(n)).$$

(7)

Traditionally, the wavelet transform follows a Mallat decomposition structure [26], whereas the lifting scheme is first applied horizontally, and the resulting subbands, $L_{i+1}$ and $H_{i+1}$, are subsequently filtered vertically delivering 4 subbands: $LL_{i+1}$, $LH_{i+1}$, $HL_{i+1}$ and $HH_{i+1}$. The low-pass subband $LL_{i+1}$ can be further decomposed, resulting in $2^n - 1$ subbands for an $n$-level Mallat wavelet decomposition. For phase unwrapping, we do not need to further decompose the high-pass subband $H_{i+1}$ after the first application of the lifting scheme since the unwrapping will only be applied on the coefficients of the low-pass channel at various scales. The decomposition of the $H_{i+1}$-band is therefore omitted for performance reasons (see Fig. 2). Hence, the lifting scheme is successively applied on the phase image, whereby alternating between horizontal and vertical filtering of the low-pass subband $L_i$ at a particular decomposition level $i$.

When transforming and subsampling the phase signal, phase aliasing has to be accounted for. Since only the principal value of the phase at discrete equidistant points can be measured, we cannot distinguish a function $F(x)$ from its alias $F(x) + 2\pi K(x)$ where $K : \mathbb{Z} \rightarrow \mathbb{Z}$. If successive samples of $F(x)$ do not differ by more than $\pi$ in absolute value, Itoh’s condition holds [4] and $F(x)$ can be determined uniquely (up to a constant term). When there is no phase aliasing, the high-pass coefficients of the regular Haar wavelets $d_H$ applied on a signal $F$ will therefore be equal to the high-pass coefficients of the modulo Haar wavelets $d_{MH}$ applied on a signal $\Phi(x) = M(F(x))$, since:

$$d_H(n) = F(2n+1) - F(2n) = M(F(2n+1) - F(2n))$$

$$= M(M(F(2n+1)) - M(F(2n))) = M(\Phi(2n+1) - \Phi(2n)) = d_{MH}(n)$$

(8)

The second equality holds since $x = M(x)$ if $x \in [-\pi, \pi]$, the third equality is a property of modular addition.

Similarly, the modulo Haar low-pass coefficients $s_{MH}$ will be equal to the regular Haar low-pass coefficients $s_H$ after application of the modulo operator:

$$s_{MH}(n) = M(\Phi(2n) + \frac{1}{2}M(\Phi(2n+1) - \Phi(2n)))$$

$$= M(\Phi(2n) + \frac{1}{2}d_{MH}(n)) = M(F(2n) + \frac{1}{2}d_H(n)) = M(s_H(n))$$

(9)

We also want to determine under what circumstances Itoh’s condition will hold for the downsampled signal represented by $s_{MH}$. Indeed, consider any four successive samples $s_i(2n)$, $s_i(2n + 1)$, $s_i(2n + 2)$ and $s_i(2n + 3)$. We have that:

$$s_{i+1}(n) = M(s_i(2n) + \frac{1}{2}d_{i+1}(n)) = M(s_i(2n) + \delta_n) = M(s_i(2n+1) - \delta_n)$$

(10)

with $\delta_n = \frac{1}{2}M(s_i(2n+1) - s_i(2n))$. The last equality can be proven, as it is equivalent to proving $M(s_i(2n)) = M(s_i(2n+1) - 2\delta_n)$:

$$M(s_i(2n+1) - 2\delta_n) = M(s_i(2n+1) - M(s_i(2n+1) - s_i(2n)))$$

$$= M(s_i(2n+1) - (s_i(2n+1) - s_i(2n))) = M(s_i(2n))$$

(11)
This analogously holds for $s_{i+1}(n+1)$. The phase difference between the low-pass components at level $i+1$ is thus:

$$|M(s_{i+1}(n+1) - s_{i+1}(n))| = |M(s_i(2n + 2) - s_i(2n + 1) + \delta_{n+1} + \delta_n)|$$  \hspace{1cm} (12)

Itoh’s condition will hold for the signal $s_{i+1}$ when all its phase differences do not exceed $\pi$ in absolute value. This can be guaranteed if the successive samples of $s_i$ do not differ by more than $\pi/2$. Indeed, as $|\delta_n|, |\delta_{n+1}| < \pi/4$ the total difference in equation 12 will be no larger than $\pi/2 + |\delta_n| + |\delta_{n+1}| < \pi$, fulfilling Itoh’s condition for $s_{i+1}$.

Therefore, if successive samples of the phase signal to be decomposed do not differ by more than $\pi/2$, indicating that the phase is fairly smooth, the properties described in Eqs. (8)-(12) will all hold. In that case, if the low-pass subband of the wrapped phase would be unwrapped, the original absolute phase will be retrieved by subsequently performing a regular inverse Haar wavelet transform. This relationship can formally be described as follows:

$$(L_{i+1}, H_{i+1}) = W_M(L_i) \text{ with } W^{-1}(U(L_{i+1}), H_{i+1}) = U(L_i)$$  \hspace{1cm} (13)

where $U$ represents an unwrapping procedure, $W$ and $W_M$ are the regular and modulo wavelet transform respectively. This principle can be applied recursively on the lowpass channel as long as the equations shown above are satisfied. Since only the low-pass signal will be unwrapped, and since it consists of much fewer samples than the original image (a reduction by a factor of $2^n$ for $n$ transform steps), the unwrapping procedure will be processed much faster. The wavelet transforms are by comparison extremely fast (linear complexity) and therefore have nearly negligible computational cost. However, the aforementioned conditions will not hold if the samples differ by more than $\pi/2$ at some locations. We address this limitation in the following section.

3. Multiscale phase unwrapping methodology

When some portion of the phase image has sharp slopes or strong discontinuities, the samples at these locations will often differ with more than $\pi/2$. But even in that case, typically large portions of the image will still have smooth phase regions which could be decomposed further, allowing to further reduce the amount of coefficients needing to be unwrapped. To address this problem, we propose a multiscale unwrapping method that will unwrap samples at different levels depending on their phase differences. To determine whether phase aliasing will occur after downsampling and thus whether a given low-pass sample is suitable for unwrapping at some decomposition level, the corresponding local phase gradient-component of the low-pass signal...
has to be determined. When a low-pass coefficient is not suitable for unwrapping because of phase aliasing, we will call the coefficient “aliased”. Using this information, a pyramid of subband masks $A_i$ with $i = \{N, N - 1, \ldots, 0\}$ and $N$ representing the decomposition level, corresponding respectively with the following subbands $\{L_N, L_{N-1}, \ldots, L_1, L_0\}$ can be implicitly constructed [Fig. 2(c)], representing low-pass wavelet coefficients that are aliased at each decomposition level $i$. The intent is to unwrap coefficients at the highest possible levels, since they will cover the largest amount of pixels in the wrapped image, thus maximally reducing the amount of coefficients to be unwrapped. The unwrapping process will therefore start at the highest levels, then progressively unwrap subsequent lower levels. Wavelet coefficients can be attributed one of the following states:

0. the coefficient has already been unwrapped at some higher level.

1. the coefficient should be unwrapped at the current level.

2. the coefficient is still aliased and can only be unwrapped at a lower level.

The coefficient states 0, 1, and 2 in Fig. 2(c) are assigned black, gray and white colors respectively.

Since the effective phase gradient-components are typically not known, they have to be estimated. In our approach, the high-pass wavelet coefficients describe the local estimated phase difference between two neighboring samples (cf. Eq. (8)) and they can therefore serve as a simple phase gradient-component estimator. We consider a low-pass sample to be aliased if the corresponding estimated high-pass sample exceeds the threshold of $\pi/2$ in absolute value, matching the required conditions for Eq. (12).

Overestimating the gradient magnitude will slow down the algorithm, as an unnecessary large amount of coefficients will be unwrapped. On the other hand, underestimation might lead to unwrapping defects. Hence, the former case is preferred. To reduce the possibility of gradient underestimation at higher wavelet levels (e.g. due to noise), a dilation operator can be applied on the subband masks $A_i$ at these wavelet levels. Dilation is a morphological binary operator, defined on a mask image $M$ with a structuring element $B$ and translation $M_b$ as:

$$M \oplus B = \bigcup_{b \in B} M_b$$

The amount of required dilation will often depend on the noisiness of the phase image to be processed. Note that gradient estimator implicitly assumes that the wrapped input signal does not have neighboring pixels differing by amounts close to multiples of $2\pi$, otherwise wrong aliasing estimates will be made. But this limitation is present in most unwrapping algorithms anyway (including the two algorithms accelerated in this paper), as they tend to use the same (or very similar) phase gradient estimators.

An important requirement for the unwrapping algorithm is its ability to unwrap arbitrary shaped contiguous patches of wrapped samples, but this restriction should pose no fundamental problem for most unwrapping algorithms.

The outline of the algorithm for accelerating phase unwrapping is shown in Fig. 3. First, the image is modulo wavelet transformed $N$ times, alternating between vertical and horizontal filtering. For notational simplicity, we will use linear addressing for describing the subband coefficients; the indexes are either to be taken column-wise or row-wise matching the current transform direction.

We have to determine for every subband $H_i$ at every level marking which coefficients have a value surpassing $\pi/2$ in magnitude; then, the states $A_i$ for every level $L_i$ denote whether or not aliasing is introduced during the modulo wavelet decomposition; these are assigned using the following recursive definition:
1. \( A_0(n) = |H_1(⌊n/2⌋)| > \pi/2 \)

2. \( A_i(n) = \max(|H_{i+1}(⌊n/2⌋)| > \pi/2, \quad 2 \cdot [A_{i-1}(2n) ∨ A_{i-1}(2n + 1)] \)

We assign 0 if the above conditions are false and 1 if they are true. These aliasing masks \( A_i \) will guide the unwrapping algorithm to determine which samples should be unwrapped for every level \( i \). Coefficients are considered to be fit for unwrapping at a particular level if their corresponding coefficient in \( A \) equals to 1 as indicated before.

Although the aliasing condition threshold is chosen to be \( \pi/2 \) at every level, because of the downsampling operation at every decomposition level by a factor of 2, the phase gradients are effectively doubled at every step alternatingly horizontally and vertically in the resulting lowpass channel. Because of this recursive definition, we are actually measuring whether the phase gradient surpasses \( \pi/2, \pi/4, \pi/8, \ldots \) w.r.t. the original input samples.

Once the algorithm has determined which pixels should be unwrapped at what level, the chosen unwrapping algorithm will progressively unwrap coefficient patches starting at the highest decomposition levels (i.e. \( L_N \)). At every level, the reverse wavelet transform is applied on the data. Depending on whether the coefficient has already been unwrapped, either the regular inverse Haar transform is applied, or the modulo version (see Fig. 4):

\[
\begin{align*}
\hat{s}_{i+1}^0 &= \mathcal{M}_c(s_{i+1} - \frac{1}{2}d_{i+1}, a_{i+1}) \\
\hat{d}_{i+1}^0 &= \mathcal{M}_e(d_{i+1} + s_{i+1}^0, a_{i+1})
\end{align*}
\]  

(15)

(16)

where \( \mathcal{M}_c \) is the conditional symmetric modulo function, defined as \( \mathcal{M}_c(x, 0) = x \), otherwise \( \mathcal{M}_c(x, 1) = \mathcal{M}_c(x, 2) = \mathcal{M}(x) \), and \( a_i \) is the aliasing mask signal from \( A_i \), denoting whether the corresponding coefficients \( s_i \) are aliased or not.

The computational resources required for the wavelet decompositions and the thresholding operations are minor with respect to all but the simplest unwrapping algorithms (such as Itoh’s
Fig. 4. Inverse lifting scheme for unwrapping. Depending on whether a coefficient of $s_i$ is aliased, either the regular Haar transform or the modulo Haar transform is performed.

method [4]). However, it is not recommended to have a large number of decomposition levels, as nearly all coefficients at the higher levels will be aliased, resulting in speed loss. Moreover, every unwrapping defect occurring at these higher decomposition levels will have significant ramifications on the quality of the unwrapped phase map. By attempting to minimize the number of coefficients to be unwrapped, a significant speedup can be expected. In the next section, we present and discuss experiments performed on various wrapped phase images.

4. Experiments

The proposed acceleration procedure can in principle be applied to any unwrapping algorithm, as long as it supports unwrapping of arbitrarily shaped patches. The source code has been written in MATLAB and C++, and the MATLAB code is publicly available [27]. We tested two different unwrapping algorithms for the experiments to demonstrate the general applicability of our methodology.

The first algorithm, “Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path” (Fast 2D PU) [6], was chosen to demonstrate the speed gain for a versatile and fast algorithm, which typically does not require long computation times. It is a path-following algorithm, which attempts to minimize the errors in the resulting unwrapped phase by determining the most reliable integration path which is free of noise and discontinuities. The reliability of this path is established by a quality map, which is calculated prior to the integration process.

The second algorithm is called “Phase unwrapping via graph cuts” (a.k.a. PUMA) [11]. This robust path-independent algorithm will represent the wrapped phase map as a graph, where every pixel corresponds to a node and the edge weights are related to the phase differences between neighboring pixels. By repeatedly solving the max-flow/min-cut problem on the graph, the algorithm will decide what partition of the graph should be incremented by a multiple of $2\pi$ as to minimize the $L^p$-norm of the unwrapped solution. The max-flow/min-cut algorithm used in PUMA [28] has an average $O(n^2\sqrt{m})$ complexity, where $n$ stands for the number of vertices (pixels) and $m$ for the number of edges, indicating the large potential for speedup when reducing the coefficient (i.e. vertex) count. Interestingly, by applying modulo wavelets and grouping coefficients into patches, we are implicitly performing edge contractions by joining vertices, similarly to Karger’s algorithm [29].

Three sets of experiments were run. In the first dataset, we evaluated the performance (both in speed and qualitative accuracy) of the unwrapping accelerator framework on acquired wrapped imagery (3 experimentally acquired wrapped phase maps + 1 synthetic phase pattern). In the second set of experiments, we use computer-generated terrain maps (resembling InSAR datasets) providing us ground truth data to quantitatively determine how the unwrapping quality is affected by the acceleration procedure. In the third set, we study how noise will affect the unwrapping acceleration.
All experiments were executed on a PC with an Intel Core i5-3360M processor (2.80 GHz) with 8 GB of RAM. The PUMA implementation uses a highly optimized C++ max-flow/min-cut implementation, as well as the code for constructing the (sub)graphs matching all the patches that need to be unwrapped. The Fast 2D PU algorithm and the modulo wavelets were implemented in MATLAB. Because of that, we also report the relative speedups.

4.1. Analysis of experimentally acquired datasets

We ran the experiments on four different images (3 of which were acquired from optical setups) to test the robustness of our approach: (1) a human face, with multiple noisy regions (source: [6]), (2) a smooth computer-generated phase pattern superimposed with Gaussian noise, (3) the wrapped phase values of an array of micro-pyramids, with multiple discontinuities and phase defects and (4) the wrapped phase taken from a hologram of a MEMS lens. All images were transformed up to ten decomposition levels (five horizontal and five vertical). We used a dilation mask with radius of 1 pixel. The results are reported in Table 1 and Figs. 5 and 6.

The speedup of the algorithm depends largely on the content of the wrapped phase image. When the phase is fairly smooth, few of the modulo wavelet coefficients will be aliased, resulting in very large speedups (up to 500 times for the synthetic phase pattern). Conversely, imagery with lots of noise and high frequency content will cause more coefficients to be unwrapped at the lower decomposition levels to avoid phase aliasing, resulting in a more limited speedup. In the extreme case, when downsampling will cause phase aliasing everywhere, no speedup will be achievable; this is expected to rarely occur in practice, as it would imply that there are consistently high gradient magnitudes present over the whole image, which does not correspond to the frequency distribution of natural imagery.

Note that the resulting absolute phase without acceleration will not necessarily be identical to the one retrieved with modulo wavelet acceleration. For example, the chosen integration path in path following algorithms can slightly change since the reliability metric might return different relative reliabilities at other wavelet scales.

### Table 1. Computation times for the unwrapping algorithms for the different test images.
The third and fourth column express the computation times in seconds, respectively for the default algorithm and the best result for the accelerated version. The relative speedup is shown in the last column. Results are averaged over 10 runs.

<table>
<thead>
<tr>
<th>Image</th>
<th>Resolution (pixels)</th>
<th>Algorithm</th>
<th>Default (s)</th>
<th>Accelerated (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>768x768</td>
<td>Fast 2D PU</td>
<td>103.4</td>
<td>0.7</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMA</td>
<td>154.7</td>
<td>0.3</td>
<td>557</td>
</tr>
<tr>
<td>Face</td>
<td>512x512</td>
<td>Fast 2D PU</td>
<td>14.1</td>
<td>4.0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMA</td>
<td>13.4</td>
<td>0.6</td>
<td>25</td>
</tr>
<tr>
<td>Pyramids</td>
<td>568x728</td>
<td>Fast 2D PU</td>
<td>50.6</td>
<td>4.6</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMA</td>
<td>13.0</td>
<td>0.4</td>
<td>33</td>
</tr>
<tr>
<td>MEMS</td>
<td>1024x1024</td>
<td>Fast 2D PU</td>
<td>263.6</td>
<td>6.0</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUMA</td>
<td>140.5</td>
<td>0.5</td>
<td>292</td>
</tr>
</tbody>
</table>
Fig. 5. Computation times for the unwrapping every image on these stacked bar charts are shown for 0 up to 10 modulo wavelet decompositions. The top chart shows results for the accelerated “Fast 2D Phase Unwrapping”, the bottom chart displays the accelerated “PUMA” algorithm. The computation times for the modulo wavelet transform and the computation overhead for constructing the (sub)graphs for PUMA are added to the graph as well. The pyramid image was only decomposed up to 6 times, as its dimensions are not divisible by 16. Results are averaged over 10 runs.
Fig. 6. Results for the unwrapping procedure. Each row represents a different dataset: “Synthetic”, “Face”, “Pyramids” and “MEMS” respectively. Column (a) shows the original wrapped phase images (black $= -\pi$ up to white $= +\pi$). Columns (b) and (c) are the unwrapped figures for the “Fast 2D PU” algorithm, respectively without and with modulo wavelet acceleration. Analogously, columns (d) and (e) are the unwrapped figures for the “PUMA” algorithm, respectively unaccelerated and accelerated.
4.2. **Error metrics on synthetic datasets**

For this experiment, we generated multiple heightmaps using MATLAB code for “Automatic terrain generation” [30], resembling the acquired wrapped phase maps found in InSAR datasets. Some of the generated maps are shown in Fig. 7. This dataset was chosen as the images tend to possess complicated natural-looking structures with varied frequency content, making them challenging to unwrap. Because the datasets are synthetic, we possess the ground-truth data, allowing us to determine how the acceleration procedure quantitatively affects the unwrapping quality.

We generated 10 terrain images of 512x512 pixel resolution, with added gaussian noise. We report the averaged Mean Square Error (MSE) of the unwrapped phase maps in Table 2 (after subtraction of the mean height value) for both tested unwrapping algorithms, for 1 up to 5 modulo wavelet decompositions and with and without dilating the aliasing masks. An example is shown in Fig. 8. The corresponding average runtime speedups are reported in Table 3.

For this type of imagery, PUMA consistently outperforms the Fast 2D PU in both accuracy and acceleration. The results clearly show the importance of using the dilation operator, especially when dealing with complicated and noisy imagery.

---

**Fig. 7.** Four examples of the wrapped terrain maps generated by the “Automatic terrain generation” algorithm.

**Fig. 8.** Visual comparison of the unwrapping errors of the algorithms w.r.t. the ground truth of one of the tested height maps. (a) is the reference height map and (f) is the wrapped version. (b), (c) and (d) are the error difference images of the accelerated PUMA algorithm using dilation with radii of 0, 1 and 2 pixels respectively; (g), (h) and (i) are the same for the accelerated Fast 2D Unwrap algorithm. (e) and (j) are the errors for the original PUMA and Fast 2D Unwrap respectively as a reference.
Table 2. Reported MSE (in radians$^2$) for various combinations of the accelerated PUMA and Fast 2D PU algorithms. The reference algorithm is displayed in the second column. The columns indicate the number of used modulo wavelet levels, the rows indicate the radius of the used dilation mask. Results are averaged over the 10 generated terrain images.

<table>
<thead>
<tr>
<th>Dilation radius</th>
<th>PUMA (Ref)</th>
<th>1 level</th>
<th>2 levels</th>
<th>3 levels</th>
<th>4 levels</th>
<th>5 levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>no dilation</td>
<td>0.0029</td>
<td>0.0256</td>
<td>0.0036</td>
<td>0.0021</td>
<td>0.0009</td>
<td>0.0007</td>
</tr>
<tr>
<td>1 pixel</td>
<td>0.0640</td>
<td>0.0519</td>
<td>0.0442</td>
<td>0.0042</td>
<td>0.0004</td>
<td>0.0003</td>
</tr>
<tr>
<td>2 pixels</td>
<td>0.2058</td>
<td>0.0702</td>
<td>0.0042</td>
<td>0.0004</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>Dilation radius</td>
<td>Fast 2D PU (Ref)</td>
<td>1 level</td>
<td>2 levels</td>
<td>3 levels</td>
<td>4 levels</td>
<td>5 levels</td>
</tr>
<tr>
<td>no dilation</td>
<td>0.1260</td>
<td>0.2014</td>
<td>0.2504</td>
<td>0.2793</td>
<td>0.2978</td>
<td>0.3090</td>
</tr>
<tr>
<td>1 pixel</td>
<td>0.2014</td>
<td>0.7605</td>
<td>0.3090</td>
<td>0.3090</td>
<td>0.3090</td>
<td>0.3090</td>
</tr>
<tr>
<td>2 pixels</td>
<td>0.0021</td>
<td>0.0009</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Table 3. Relative speedup factors for the accelerated PUMA and Fast 2D PU algorithms, averaged over 10 instances.

<table>
<thead>
<tr>
<th>Dilation radius</th>
<th>PUMA (Ref)</th>
<th>1 level</th>
<th>2 levels</th>
<th>3 levels</th>
<th>4 levels</th>
<th>5 levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>no dilation</td>
<td>1.0</td>
<td>10.6</td>
<td>136.8</td>
<td>152.1</td>
<td>149.1</td>
<td></td>
</tr>
<tr>
<td>1 pixel</td>
<td>15.3</td>
<td>79.5</td>
<td>104.7</td>
<td>104.1</td>
<td>103.2</td>
<td></td>
</tr>
<tr>
<td>2 pixels</td>
<td>17.7</td>
<td>51.1</td>
<td>55.2</td>
<td>52.4</td>
<td>53.5</td>
<td></td>
</tr>
<tr>
<td>Dilation radius</td>
<td>Fast 2D PU (Ref)</td>
<td>1 level</td>
<td>2 levels</td>
<td>3 levels</td>
<td>4 levels</td>
<td>5 levels</td>
</tr>
<tr>
<td>no dilation</td>
<td>1.0</td>
<td>4.9</td>
<td>8.7</td>
<td>9.4</td>
<td>10.0</td>
<td>10.4</td>
</tr>
<tr>
<td>1 pixel</td>
<td>3.1</td>
<td>3.7</td>
<td>3.9</td>
<td>4.0</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>2 pixels</td>
<td>2.3</td>
<td>2.5</td>
<td>2.5</td>
<td>2.6</td>
<td>2.6</td>
<td></td>
</tr>
</tbody>
</table>

4.3. Effects of noise

In this series of experiments, we want to investigate the effect of Gaussian noise on the unwrapping accelerator. We used the synthetic wrapped phase pattern found in subsection 4.1, and superimposed the phase with various levels of Gaussian noise, with a standard deviation ($\sigma$) going from 0 up to $0.4 \cdot \pi$. We unwrapped the images using PUMA combined with 5 modulo wavelet levels with a dilation radius of 1 pixel. Some of the unwrapping results are shown in Fig. 9. The runtimes are shown in Table 4, consisting of the average runtimes using 10 different noise patterns per $\sigma$ value.

Table 4. Averaged runtimes over 10 instances per noise level for the accelerated PUMA algorithm using 5 wavelet decomposition levels. The runtime for the averaged runtime over all noise levels for the (unaccelerated) reference method is given in the last column, denoted “Ref”.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>0</th>
<th>0.05\pi</th>
<th>0.1\pi</th>
<th>0.15\pi</th>
<th>0.2\pi</th>
<th>0.25\pi</th>
<th>0.3\pi</th>
<th>0.35\pi</th>
<th>0.4\pi</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (s)</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td>0.9</td>
<td>1.2</td>
<td>11.9</td>
<td>86.9</td>
<td>95.7</td>
<td>96.3</td>
<td>155.2</td>
</tr>
</tbody>
</table>

The runtime grows as the noise increases, and shows an abrupt jump at a standard deviation around 0.3$\pi$. This most likely due to the fact that the noise will get so strong that it will cause many coefficients to be marked as aliased, increasing the number of coefficients needed to be
Fig. 9. Images (a)-(d) represent four of the synthetic wrapped phase images respectively superimposed with Gaussian noise with standard deviations (σ) of 0, 0.1π, 0.2π and 0.4π. Images (e)-(h) are the corresponding unwrapped images using the accelerated PUMA algorithm.

unwrapped. The modulo Haar filters are too short to be able to distinguish strong noise from sharp edges. However, the runtime is still significantly inferior to the non-accelerated algorithm as shown in Table 4. This problem could be addressed by developing larger modulo wavelet filters, using more robust gradient estimators (e.g. the “Amended matrix pencil model” [15]) or prefiltering the wrapped phase to denoise the input image prior to the unwrapping process.

5. Conclusions

We have shown that the performance of phase unwrapping algorithms can be accelerated by using modulo wavelets on imagery with heterogeneous frequency content. The algorithm is tested on various types of wrapped phase imagery, reporting speedup factors of 4 up to 500. The proposed scheme is a generic tool applicable to many different types of unwrapping algorithms. The potential for speedups is especially high for large datasets. We expect that further improvements to the algorithm could be made by using modulo wavelets with larger filter lengths as to improve the robustness to noise. Pre-filtering could be applied as well on very noisy input data. Developing more advanced gradient estimation heuristics will allow to further reduce unwrapping defects. In addition, further accelerations could be achieved by skipping corrupted phase regions entirely. This approach is directly extensible to higher-dimensional data as well. The source code of the algorithm is publicly available.

Funding

European Union’s Seventh Framework Programme (FP7/2007-2013)/ERC Grant Agreement n.617779 (INTERFERE).

Acknowledgments

We would also like to thank Prof. Anand Asundi of Nanyang Technological University, MAE school for providing the MEMS-hologram. The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme.