Implementation of A High Accuracy Phase Unwrapping Algorithm using Parallel-hybrid Programming Approach for Displacement Sensing using Self-Mixing Interferometry

Tassadaq Hussain[1], Saqib Amin[1], Usman Zabit[2], Eduard Ayguadé[3]
[1]Riphah International University Islamabad, Pakistan
[2] National University of Sciences and Technology (NUST), Islamabad, Pakistan
[3] Barcelona Supercomputing Center, Spain
Email: {tassaduq.hussain} @riphah.edu.pk

***Abstract*—Phase unwrapping is an integral part of multiple algorithms with diverse applications. Detailed phase unwrapping is also necessary for achieving high accuracy metric sensing using laser feedback based self-mixing interferometry (SMI). Among SMI specific phase unwrapping approaches, a technique called Improved Phase Unwrapping Method (IPUM) provides the highest accuracy. However, due to its complex, sequential, and compute-intensive nature, this method requires a high performance computing architecture, capable of scalable parallel processing so that such a high accuracy algorithm can be used for high bandwidth sensing applications. In this work, the existing sequential IPUM C program is parallelized by using hybrid OpenMP/MPI (Open Multi-Processing/Message Passing Interface) parallel programming models and tested on Barcelona Supercomputing Center Nord-III Supercomputer. The computational performance of the proposed parallel-hybrid IPUM algorithm is compared with existing IPUM sequential code by executing multi-core and uni-core processor architecture respectively. While comparing the performance of sequential IPUM with the parallel-hybrid IPUM algorithm on 16 nodes of Nord-III supercomputer, the results show that the parallel-hybrid algorithm gets 345.9x times performance improvement as compared to IPUM's standard, sequential implementation on a single node system. The results show that the parallel-hybrid version of IPUM gives a scalable performance for different target velocities and a different number of processing cores.**

I. Introduction

Phase unwrapping is an integral part of multiple algorithms with applications such as fingerprint re-construction [1], synthetic aperture radar (SAR) interferograms [2], magnetic field mapping [3], 3D shape reconstruction [4], digital photo-elasticity [5], imaging [6] etc. Accurate phase unwrapping is also a fundamental requirement for high accuracy metric sensing using self-mixing interferometry (SMI) or optical feedback interferometry [7]. As opposed to conventional interferometry techniques requiring high optical part count (e.g. mirrors, beam-splitters, and external photo-detectors), SMI is attractive for real-world applications due to self-aligned, low-cost and compact nature of the SMI sensor (see Figure 1). SMI has been thus demonstrated for vibration [8], [9], displacement [10], [11], gas spectroscopy [12], embedded [13] and biomedical applications [14] etc.

SMI based retrieval of remote target's displacement, however, is not simple due to rich diversity in the shape, amplitude, and hysteresis of SMI signals under variable optical feedback coupling (see Figure 2). In this regard, various SMI phase unwrapping methods have been proposed [15, 10, 16, 17, 18, 19, 20, 21, 22] with displacement retrieval accuracy ranging from $\lambda_0/8$ to $\lambda_0/60$, where λ_0 is laser's wavelength. Among these methods, the Improved Phase Unwrapping Method (IPUM) [22] delivers the best performance.

However, this high accuracy algorithm is still not being used for real-world, high bandwidth applications (such as monolithic multi-channel laser array imaging [23] and measurement of ultrasonic vibrations [24], etc.) due to its complex, compute-intensive, sequential and time-consuming blocks involving signal segmentation, max./min. searches, phase reconstruction, and iterative joint-estimation of key optical feedback based parameters [22]. This then requires that a sequential and compute-intensive algorithm such as IPUM needs to be parallelized and then implemented on High Performance Computing (HPC) platforms so that scalable performance and high accuracy sensing can be achieved for real-world, high bandwidth applications.

In this work, we present Open Multi-Processing (OpenMP) and Message Passing Interface (MPI) based parallelized parallel version of IPUM called parallel-hybrid IPUM and executed on Barcelona Supercomputing Center Nord-III supercomputer. The methodology of parallel-hybrid IPUM algorithm uses earlier published sequential IPUM C program [22], find control and computation by using Control Data Flow Graph (CDFG) model, based on the CDFG distribute the algorithm into multiple tasks, schedule the tasks on distributed processing system using MPI programming environment and execute each task on a shared memory multi-core processor system using OpenMP. Performance, scalability, and portability have been considered while programming the parallel-hybrid implementation of the IPUM algorithm. The results show that parallel parallel-hybrid programming gives a scalable performance.

The parallel-hybrid implementation of IPUM on distributed systems shows 345.9x times performance improvement. The results show that the parallel-hybrid IPUM algorithm gives a scalable performance for different target velocities and the different number of distributed nodes.

II. Related Work

An SMI based real-time vibrometer was developed by Magnani et al. [17] which was based on a digital signal processor (DSP) card (F28335 by Texas Instruments) performing signal acquisition at 1 M samples/s. The implemented algorithm then estimated the optical feedback coupling condition and carried out appropriate optical feedback regime specific SMI signal elaboration.

Using SMI for vibrometry application, Melchionni et al. [25] proposed an FPGA-based prototype. For this purpose, a detailed analysis of the feedback loop was carried out to achieve better performance under unstable optical feedback conditions.

Zabit et al. [26] proposed a real-time SMI displacement sensor for parasitic-motion-compensated displacement sensing that is capable of retrieving correct remote target's displacement when in the presence of corruptive local motion. This was achieved by coupling a solid-state accelerometer with the SMI. Data-fusion between accelerometer and SMI sensor was achieved in real-time by using a micro-controller by Analog Devices called ADuC7020 and a DSP by Microchip called dsPIC33FJ128GP.

Using SMI for flow measurement, Norgia et al. [27] demonstrated a laser diode-based flow sensor, based on Doppler-shift caused by fluid-flow. The prototype of the SMI based flow sensor was based on a DSP model TMS320F28044 with a sampling frequency of 6 MHz.

Cavedo et al. [28] developed an SMI based distance sensor for the characterization of steel pipes. A 32-bit ARM Cortex-M4 core operating at 72 MHz along with Micro-controller STM32F303 was used for low-cost and fast system design.

Likewise, an FPGA-based real-time optical SM interferometer for speed and direction measurements was proposed by Magnani et al. [29]. Altera DE0-nano FPGA board was utilized for system development.

Based on four laser interferometers in a specific arrangement, Ducourtieux et al. [30] developed a metrological atomic force microscope for calibrations and dimensional measurements. An FPGA-based controller along with an embedded PXI controller was used to accomplish real-time control of XYZ positions.

A high speed, real-time, laser displacement sensor was developed by Ji et al. [31] by using a charge-coupled device. It used a high-speed analog to digital converter with a conversion rate up to 40 MHz, along with first-in-first-out (FIFO) and digital signal processor. The intensity of light was controlled by an adaptive technique to make the system suitable for various specific target objects.

Lastly, using heterodyne interferometry, Wang [32] designed an FPGA based prototype of high speed, 4-channel phase-meter, capable of performing commercial phase measurements. The system was designed and tested on the Altera DE2 FPGA development board.

III. SMI And Improved Phased Unwrapping Method

As already mentioned, Improved Phased Unwrapping Method (IPUM) [22] is an advanced, high-accurate algorithm capable of retrieving remote target displacement information by processing the SM interferometric signal acquired through the SM sensor of a remote target. This section is further categorized into two subsections the *Self-Mixing Fundamentals* and the *Working Principle of IPUM*.

A. Self-Mixing Fundamentals

Self-Mixing (SM) laser sensor works on the principle of optical feedback interferometry. The laser beam is generated in the optical cavity and a portion of the laser light is reflected back from the target surface and re-enters the active laser package affecting the transmitted light by mixing with the generated light. This Self-Mixing changes spectral and optical laser properties, which can be detected at built-in photodiode to capture the corresponding signal, known as the SM signal [7]. This SMI signal can then be processed for remote sensing purposes, as elaborated below. The deviations in the optical output power (OOP) of the laser $P(t)$ due to optical feedback can be represented as:

$$P(t) = P_0[1 + m.\cos(\phi_F(t))] \tag{1}$$

P_0 is the transmitted power under no feedback condition, and m represents the modulation index. $\phi_F(t)$ represents the output phase of laser under feedback, given by Equation 2:

$$\phi_F(t) = 2\pi \frac{D(t)}{\lambda_F(t)/2} \tag{2}$$

Where $D(t)$ represents the moving target displacement. The emitted wavelength under feedback $\lambda_F(t)$ is shown by the equation known as excess phase equation 3:

$$\phi_0(t) = \phi_F(t) + C\sin[\phi_F(t) + arctan(\alpha)] \tag{3}$$

α is the line-width enhancement factor and $\phi_0(t)$ represents output laser phase in no feedback condition, which can be obtained by substituting $\lambda_F(t)$ in the Equation 2 by $\lambda_0(t)$. The optical feedback coupling factor C is represented by the Equation 4:

$$C = \frac{\tau_D}{\tau_L}\gamma\sqrt{1+\alpha^2}\kappa_{ext} \tag{4}$$

Where τ_L and τ_D are the round trip times of the inside and outside cavities respectively, whereas γ represents the coupling efficiency, and κ_{ext} linearly depend on the reflectiveness of the target surface.

C value plays a fundamental role in SM interferometry and determines the regime in which an SM sensor operates [7]. C $<$ 1 describes the so-called weak feedback regime, where equation 3 has a unique solution. 1 $<$ C $<$ 4.6 characterizes the moderate optical feedback regime with multiple solutions

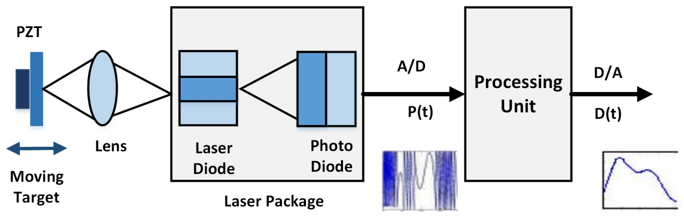


Fig. 1. Block diagram of a basic Self-Mixing (SM) sensing set-up requiring only a laser diode package with its built-in photo-diode and a focusing lens. A piezoelectric transducer (PZT) has been used as target. Variations in the optical output power of the laser diode $P(t)$ are processed to retrieve the target motion $D(t)$

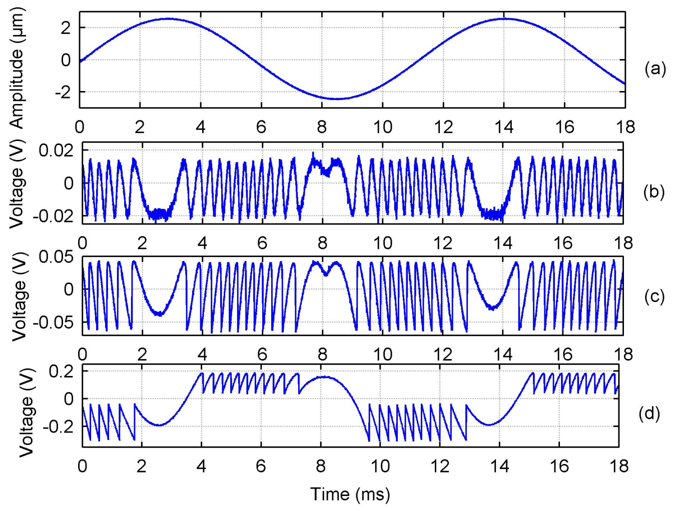


Fig. 2. (a) Vibration of remote target (a piezoelectric transducer (PZT) acting as reference senor with 2 nm resolution) with peak to peak amplitude of 5 micrometers, and corresponding SM interferometric signals acquired using Sanyo DL7140 laser diode with λ = 785nm belonging to : (b) weak optical feedback regime, (c) moderate optical feedback regime, and (d) strong optical feedback regime.

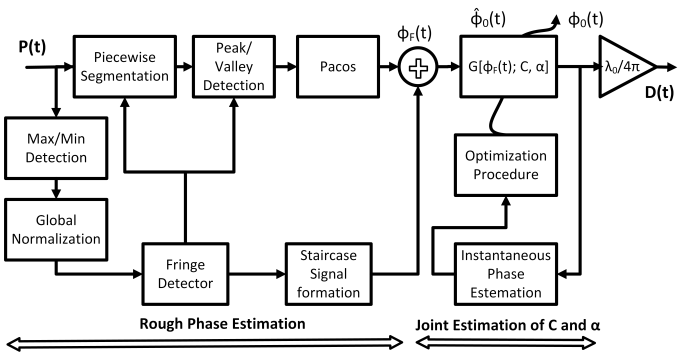


Fig. 3. Block diagram of Improved Phase Unwrapping Method (IPUM) [22]

to equation 3, while C $>$ 4.6 encompasses the strong feedback regime with increasingly chaotic behavior for higher C values. Using an SM sensor with Sanyo DL7140 laser diode with λ = 785nm, Figure 2 (b) presents an experimentally acquired SM signal corresponding to weak feedback regime with characteristic quasi-sinusoidal fringes, Figure 2 (c) represents a moderate optical feedback regime SM signal with saw-tooth shaped fringes, while Figure 2 (d) represents a strong feedback regime SM signal with hysteresis and fringe-loss [33].

Let us now summarize how remote target's displacement D(t) can be retrieved from P(t) by using IPUM which is based on solution of Equations 1, 2, 3.

B. Working Principle of IPUM

The working principle of the IPUM algorithm is schematized in Figure 3 [22] and explained below in detail.

First, global normalization of $P(t)$ is done in order to obtain $P(t)/P0$ in range of ± 1 interval, then an arcos function is utilized to retrieve $\phi_F(t)_{mod\pi}$ (Equation 1). Later, the SMI signal fringe is identified. For fringe detection without speckle [34, 35], a straightforward method based on derivative [20] is enough. However, in the case of seackles, SM fringe detection needs more advanced techniques e.g. customized Wavelets [36], Hilbert Transform [37] or advance neural networks [38]. The rough phase $\phi_F(t)_{mod\pi}$ is then processed through a piece-wise segmentation on identified peaks and valleys from fringe locations. In parallel, a stair-case signal is formed by using an integrator to continuously add or subtract 2π at each fringe location depending upon its direction.

This stair-case signal is added with P_{acos} signal to form a rough phase ϕ_F.

This rough signal is then used in the joint estimation process of finding the key parameters C and $\alpha = \phi_F(t=0) + \arctan(\alpha)$, by using a bi-dimensional optimization routine on the excess equation of phase in equation 3. An accurate estimation of $\phi_0(t)$ related to target displacement $D(t)$ concludes the IPUM algorithm.

IV. IPUM Programming Methodology

In this section we discussed programming methodology of sequential and parallel-hybrid IPUM algorithms. The section is further subdivided into three subsections; the *Sequential IPUM*, *Parallel-Hybrid IPUM*, and the *Processing System*.

A. Sequential IPUM

This section discusses the working principle of the sequential IPUM algorithm. The pseudo code of sequential IPUM has 13 function (T1 : T13) (shown in Figure 4), short description of each function (T) is mentioned below:

```
Begin
Input: $\mathbf{P_{in}}$
Constant: FIR[coeffs]
T1: For j ← 0: i-1
    $P[j] \leftarrow P_{in}$
End T1
$P_{max}, P_{min} \leftarrow 0$
T2: For j ← 0: i-1
     If $P[j] > P_{max}$
          $P_{max} \leftarrow P[j]$
      Else If $P[j] < P_{min}$
          $P_{min} \leftarrow P[j]$
      End If
End T2
T3: For j ← 0: i-1
     $P_{norm}[j] \leftarrow 2 * \left[\frac{(P[j]-P_{min})}{P_{max}-P_{min}}\right] - 1$
End T3
T4: For j ← 0: i-1
     $\varphi_{Fmod\pi}[j] \leftarrow \arccos(P_{norm}[j])$
End T4
T5: For j ← 1: i-1
     $P_{diff}[j] \leftarrow P_{norm}[j] - P_{norm}[j-1]$
End T5
Constant: $th_{pos} \leftarrow A$,  $th_{neg} \leftarrow B$
Int K=0;
T6: For j ← 0: i-1
     IF $P_{diff}[j] < th_{neg}$
          Fringe_val[k] ← -1
          Fringe_loc[k] ← j
          k← k+1
     ELSE IF $P_{diff}[j] > th_{pos}$
          Fringe_val[k] ← 1
          Fringe_loc[k] ← j
          k← k+1
      End If
End T6
Constant: band ← C,
T7: For j ← 0: k
    Fringe_amp[j] ← 0;
    T7-1: For m ←Fringe_loc[j]– band : Fringe_loc[j] + band
          IF P[m] > Fringe_amp[j]
               Fringe_amp[j] ← P[m]
                    Peak_loc[j] ← m
          End If
     End T7-1
     Fringe_amp[j] = 1;
     T7-2: For m ←Fringe_loc[j]– band :Fringe_loc[j] + band
          IF P[m] < Fringe_amp[j]
               Fringe_amp[j] ← P[m]
                    Valley_loc[j] ← m
          End If
     End T7-2
End T7
T8: For j ← 0: k
      IF Fringe_val[j] ← -1
          T8-1: For m ← Valley_loc[j] : Peak_loc[j]
               $\varphi_{Fmod\pi}[m] \leftarrow \arccos(-1 * P[m])$
          End T8-1
     Else IF Fringe_val[j] ← 1
          T8-2: For m ← Peak_loc[j] : Valley_loc[j]
               $\varphi_{Fmod\pi}[m] \leftarrow \arccos(-1 * P[m])$
          End T8-2
     End IF
End T8
$P_{staircase}[0] \leftarrow 0$
T9: For j ← 1: i-1
   IF $(\varphi_{Fmod\pi}[j] - \varphi_{Fmod\pi}[j-1]) > \pi/2$ )
        $P_{staircase}[j] \leftarrow P_{staircase}[j-1] - \pi$
   Else IF $(\varphi_{Fmod\pi}[j] - \varphi_{Fmod\pi}[j-1] < -\pi/2)$
        $P_{staircase}[j] \leftarrow P_{staircase}[j-1] + \pi$
   Else
        $P_{staircase}[j] \leftarrow P_{staircase}[j-1]$
  End IF
End T9
T10: For j ← 1: i-1
  $\hat{\Phi}_F[j] \leftarrow P_{staircase}[j] + \varphi_{Fmod\pi}[j]$
End T10
$C\_val \leftarrow C_{start}$,  $\alpha\_val \leftarrow \alpha_{start}$
T11: Loop $C_{ind} \leftarrow 0$: i1-1
     T11-1: Loop $\alpha_{ind} \leftarrow 0$: i2-1
          T11-1-1: Loop  j ← 0: i-1
            $\hat{\Phi}_0[j][C_{ind}][\alpha_{ind}] \leftarrow \hat{\Phi}_F[j] + C_{val} * \sin(\hat{\Phi}_F[j] + \arctan(\alpha_{val}))$
          End T11-1-1
          T11-1-2: Loop  j ← 1: i-1
            $\hat{\Phi}_{0\_diff}[j] \leftarrow \hat{\Phi}_0[j][C_{ind}][\alpha_{ind}] - \hat{\Phi}_0[j-1][C_{ind}][\alpha_{ind}]$
          End T11-1-2
          $J[C_{ind}][\alpha_{ind}] \leftarrow 0$
          T11-1-3: Loop  j ← 0: i-1
              IF  j < i - coeffs+1      % filtering
                   Accum = 0;
                   T11-1-3-1: For  f ← 0: coeffs
                        Accum = Accum + FIR[f] * $\hat{\Phi}_{0\_diff}[j+f]$
                   End T11-1-3-1
                   $\hat{\Phi}_{0\_diff}[j] \leftarrow$ Accum
              Else
                   $\hat{\Phi}_{0\_diff}[j] \leftarrow 0$
              End IF
          $J[C_{ind}][\alpha_{ind}] \leftarrow J[C_{ind}][\alpha_{ind}] + \mathrm{rms}\{\hat{\Phi}_{0_{diff}}[j]\}$
          End T11-1-3
      $\alpha\_val \leftarrow \alpha\_val + \alpha_{step}$
    End T11-1
  $C\_val \leftarrow C\_val + C_{step}$
End T11
$J_{min} \leftarrow P_{max}$,  $C_{opt} \leftarrow 0$ , $\alpha_{opt} \leftarrow 0$
T12: Loop $C_{ind} \leftarrow 0$: i1-1
   T12-1: Loop2 $\alpha_{ind} \leftarrow 0$: i2-1
        IF $J[C_{ind}][\alpha_{ind}] < J_{min}$
          $J_{min} \leftarrow J[C_{ind}][\alpha_{ind}]$
          $C_{opt} \leftarrow C_{ind} * C_{step}$,    $\alpha_{opt} \leftarrow \alpha_{ind} * \alpha_{step}$
       End IF
    End T12-1
End T12
T13: For j ← 0: i-1
Output:         $D[j] = \frac{\lambda_0}{4\pi} * \hat{\Phi}_0[j][C_{opt}][\alpha_{opt}]$
End T13
END
```

Fig. 4. Pseudo-code: Generic Sequential Improved Phased Unwrapping Method

T1: The algorithm starts processing by storing input data sample vectors stored in local memory P_{in}.

T2: Identify max/min values of data using max/min search.

T3: Perform global normalization.

T4: Calculate arccosine of normalized data.

T5: Takes derivative of normalized data

T6: Threshold applied on derivative to detect ± fringes.

T7: Identify Peak/valley locations (shown Figure 3). It uses

```
Begin
Input: $\mathbf{P_{in}}$
Constant: FIR[coeffs]
$P_{max}$, $P_{min} \leftarrow 0$
T1: For j ← 0: i-1
    $P[j] \leftarrow P_{in}$
     If $P[j] > P_{max}$
          $P_{max} \leftarrow P[j]$
      Else If $P[j] < P_{min}$
           $P_{min} \leftarrow P[j]$
      End If
End T1
Constant $th_{pos} \leftarrow$ A, $th_{neg} \leftarrow$ B, band← C
Int k=0, $P_{staircase}[0] \leftarrow 0$
T2: For j ← 0: i-1
     $P_{norm}[j] \leftarrow 2 * \left[\frac{(P[n]-P_{min})}{P_{max}-P_{min}}\right] - 1$
     $\varphi_{Fmod\pi}[j] \leftarrow \arccos(P_{norm}[j])$
     If n>0 then
          $P_{diff}[j] \leftarrow P_{norm}[j] - P_{norm}[j-1]$
          IF $P_{diff}[j] < th_{neg}$
                Fringe_val[k] ← -1
                Fringe_loc[k] ← j
          Else IF $P_{diff}[j] > th_{pos}$
                Fringe_val[k] ← 1
                Fringe_loc[k] ← j
          End If
          Fringe_amp[k] ← 0;
          T2-1: For m ←Fringe_loc[k]–band:Fringe_loc[k] + band
                   IF P[m] > Fringe_amp[k]
                           Fringe_amp[k] ← P[m]
                           Peak_loc[k] ← m
                   End If
          End T2-1
            Fringe_amp[k] = 1;
          T2-2: For m ←Fringe_loc[k]–band:Fringe_loc[k] + band
                   IF P[m] < Fringe_amp[k]
                           Fringe_amp[k] ← P[m]
                           Valley_loc[k] ← m
                   End If
          End T2-2
          IF Fringe_val[k] ← -1
                   T2-3: For m ← Valley_loc[j] : Peak_loc[j]
                               $\varphi_{Fmod\pi}[m] \leftarrow \arccos(-1 * P[m])$
                   End T2-3
           Else IF Fringe_val[k] ← 1
                   T2-4: For m ← Peak_loc[j] : Valley_loc[j]
                               $\varphi_{Fmod\pi}[m] \leftarrow \arccos(-1 * P[m])$
                   End T2-4
          End IF
           K ← k+1
          IF $(\varphi_{Fmod\pi}[j] - \varphi_{Fmod\pi}[j-1]) > \pi/2$ )
                $P_{staircase}[j] \leftarrow P_{staircase}[j-1] - \pi$
          Else IF $(\varphi_{Fmod\pi}[j] - \varphi_{Fmod\pi}[j-1] < -\pi/2)$
                $P_{staircase}[j] \leftarrow P_{staircase}[j-1] + \pi$
          Else
                $P_{staircase}[j] \leftarrow P_{staircase}[j-1]$
         End IF
         $P_{stair}[j] \leftarrow 2\pi$*Fringes[j] + $P_{stair}[j-1]$
         $\widehat{\Phi}_F[j] \leftarrow P_{stair}[j] + \varphi_{Fmod\pi}[j]$
     End If
End T2
C_val ← $C_{start}$, α_val← $\alpha_{start}$, $J_{mii} \leftarrow P_{max}$, $C_{opt} \leftarrow 0$ , $\alpha_{opt} \leftarrow 0$
T3: Loop $C_{ind} \leftarrow$ 0: i1-1
     T3-1: Loop $\alpha_{ind} \leftarrow$ 0: i2-1
      $J[C_{ind}][\alpha_{ind}] \leftarrow 0$
         T3-1-1: Loop j ← 0: i-1
     $\widehat{\Phi}_0[j][C_{ind}][\alpha_{ind}] \leftarrow \widehat{\Phi}_F[j] + C\_val * \sin(\widehat{\Phi}_F[j] + \arctan(\alpha\_val))$
           If n >0 then
             $\widehat{\Phi}_{0\_diff}[j] \leftarrow \widehat{\Phi}_0[j][C_{ind}][\alpha_{ind}] - \widehat{\Phi}_0[j-1][C_{ind}][\alpha_{ind}]$
                IF j < i - coeffs+1      % filtering
                   Accum = 0;
                   T3-1-1-1: For f ← 0: coeffs
                     Accum = Accum + FIR[f] * $\widehat{\Phi}_{0\_diff}[j+f]$
                  End LT3-1-1-1
                  $\widehat{\Phi}_{0\_diff}[j] \leftarrow$ Accum
               Else
                  $\widehat{\Phi}_{0\_diff}[j] \leftarrow 0$
               End IF
             $J[C_{ind}][\alpha_{ind}] \leftarrow J[C_{ind}][\alpha_{ind}] + rms\{\widehat{\Phi}_{0_{diff}}[j]\}\}$
           End If
         End T3-1-1
         IF $J[C_{ind}][\alpha_{ind}] < J_{min}$ then
           $J_{min} \leftarrow J[C_{ind}][\alpha_{ind}]$
          $C_{opt} \leftarrow C_{ind}*C_{step}$,   $\alpha_{opt} \leftarrow \alpha_{ind}*\alpha_{step}$
        End IF
        $\alpha$_val ← $\alpha$_val + $\alpha_{step}$
    End T3-1
   C_val ← C_val + $C_{step}$
End T3
T4: For j ← 0: i-1
Output:                    $D[j] = \frac{\lambda_0}{4\pi} * \widehat{\Phi}_0[j][C_{opt}][\alpha_{opt}]$
End T4
END
```

Fig. 5. Pseudo-code: Parallel-Hybrid Improved Phased Unwrapping Method

two conditional branches (T7-1 and T7-2) identify the peak- and valley-locations.

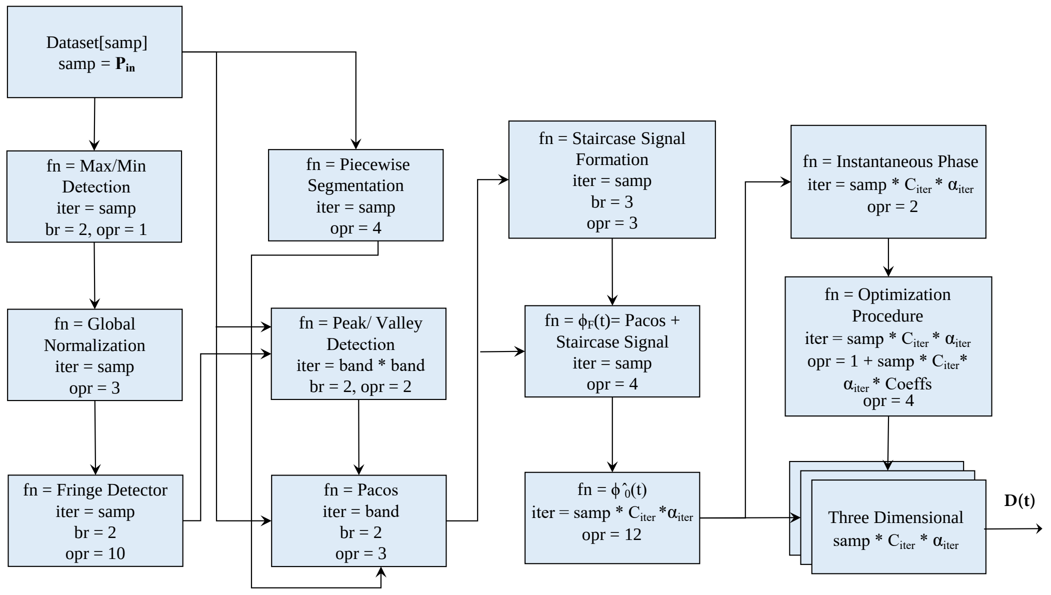
T8: Local phase inversion correction is performed using piecewise segmentation around each detected fringe (having two branches T8-1 and T8-2) leading to $\phi_{Fmod\pi}$.

T9: Implement staircase signal formation block, which is based on derivative of $\phi_{Fmod\pi}$ followed by addition of $\pm\ \pi$.

T10: Provide rough phase ϕ_F retrieval.

T11: Performs joint estimation of C and α parameters (shown in Figure 3). The function solves excess phase equation 3 for different values of C and α. It generates instantaneous RMS power by taking derivative of phase equation solution and applies low pass filtering that removes undesired high-frequency components, and then calculate the RMS resulting in $J[C_{ind}][\alpha_{ind}]$. Complex function having multiple nested loops and conditional branches. T11-1-1 performs the solution of phase equation operation, and T11-1-2 performs the operation of derivative of solution from the previous operation, while T11-1-3 performs the operation of filtering of the derivative signal followed by instantaneous power calculation $J[C_{ind}][\alpha_{ind}]$.

12: Performs minimization routine on $J[C_{ind}][\alpha_{ind}]$ enabling



Dataset[samp] samp = $\mathbf{P_{in}}$
fn = Max/Min Detection iter = samp br = 2, opr = 1
fn = Global Normalization iter = samp opr = 3
fn = Fringe Detector iter = samp br = 2 opr = 10
fn = Piecewise Segmentation iter = samp opr = 4
fn = Peak/ Valley Detection iter = band * band br = 2, opr = 2
fn = Pacos iter = band br = 2 opr = 3
fn = Staircase Signal Formation iter = samp br = 3 opr = 3
fn = ϕ_F(t)= Pacos + Staircase Signal iter = samp opr = 4
fn = $\hat{\phi_0}$(t) iter = samp * C_{iter} *α_{iter} opr = 12
fn = Instantaneous Phase iter = samp * C_{iter} * α_{iter} opr = 2
fn = Optimization Procedure iter = samp * C_{iter} * α_{iter} opr = 1 + samp * C_{iter}* α_{iter} * Coeffs opr = 4
Three Dimensional samp * C_{iter} * α_{iter}
D(t)

Fig. 6. Control Data Flow Data Graph of IPUM: Presenting Tasks/Functions, Iterations, Branches and Operations.

C and α estimation.

T13: Retrieves the desired target displacement D(t) using Equation 2.

B. Parallel-hybrid IPUM

The pseudo-code of parallel-hybrid IPUM is shown in Figure 5. Thorough understanding of major functions, their iterations, branches and operations of generic pseudo-code has helped to reduce complexity, functions, branches, and iterations. It was observed that pipe-lining was possible in some major functions to reduce the complexity and iterative nature of the algorithm. Parallel-hybrid IPUM pseudo code uses 4 functions (T1: T4), details of each function are mentioned below.

T1: The parallel-hybrid IPUM algorithm starts performing parallel max/min operations on incoming data, eliminating the need for separate function having iterations for the operation of max/min detection and normalization. Unlike sequential IPUM, only two consecutive samples are used (at a time) to take the derivative. This improves the input data pipelining delay. Furthermore, threshold and fringe detection operations also do not require a complete vector for processing.

T2: Peak/valley location identification operation and local phase inversion correction operation are performed on individual fringes rather than waiting for the identification of all fringes. Staircase signal formation operation and rough phase retrieval are also pipelined inside the function T2, as these require only two samples of data at a time, rather than the complete data vector to start their processing.

T3: Performs pipelined implementation of a joint estimation of C and α. Branch T3-1 provide different combinations of C and α values for the operation of the solution of phase equation (Equation 3). The derivative operation is pipelined with the phase equation solution. The operation of filtering of the derivative signal is also now performed in a pipelined manner with derivative operation leading to the instantaneous power calculation (T3-1-1-1). The minimization replaces instantaneous power calculations of all the values of C and α and iterative minimization routine with a simple if-else branch inside the function T3. Which compares the current value of instantaneous powers $J[C_{ind}][\alpha_{ind}]$ after each iteration to estimate optimum C and value. This replacement not only reduces the complexity but also makes the algorithm more suitable for real-time parallelized implementation due to reduction and subsequent unrolling of iterative functions.

T4: It provides the estimation of target displacement by using the optimized C and α values, thereby concluding the processing steps.

C. Parallel-Hybrid IPUM Programming

The parallel-hybrid structure of IPUM uses Control Data Flow Graph (CDFG) to identify the available parallelism IPUM. The CDFG performs the partitioning of the IPUM algorithm and generates smaller tasks/functions. Each task has data dependencies and cannot be portioned more. Later each task is assigned a data link and links all the tasks together. Each task The CDFG of IPUM algorithm is shown in the Figure 6 representing the processing and control details. The parallel-hybrid program of IPUM takes the CDFG and partitioned into tasks/functions. Each task is further divided into iterations (*iter*), branches (*br*) and operations (*opr*). The iterations define the data intensity, branches and operations present the control complexity and computation respectively for each task. The IPUM takes an input data set and after performing processing it stores the output in a 3D data structure. Each task (shown in the Figure 6) can be processed independently on a processor core of a multi-core system. The complexity of IPUM can be seen from its CDFG (Figure 6), each graph (arrow) represents data-flow and data-dependencies. The IPUM algorithm requires 5 thousand operations to process a 16-bit sample, therefore its operational intensity (Floating Point Operations per Second / Byte) (FLOPS/B) would be $(5 \times 10^3 FLOPS/2Byte)$ 2.5×10^3 FLOPS/B.

The parallel-hybrid IPUM program takes the benefits of both OpenMP and MPI programming models. The MPI model takes the main program and performs parallelization at the top level by converting the main algorithm into multiple processes. The MPI processes control the data of each task/function and transfer it to multiple nodes (distrusted memory multi-core processing systems). The MPI processes run the tasks concurrently on multiple nodes by providing communication between them. The implementation of parallel IPUM uses a parallel-hybrid (OpenMP/MPI) programming model approach. The MPI programming model schedule the IPUM tasks by using MPI processes (MPI1 to MPI7) on different processing nodes. Each node has shared memory system architecture that can run a task on multiple parallel processing cores. A single process executes multiple tasks on shared memory processing core using OpenMP. To perform fine-grain parallelism on a single node OpenMP programming model is used. The OpenMP takes a single task and executes on a shared memory multi-core processing system.

D. *Processing System*

In this section, we describe the Nord-III supercomputer system architecture that is used to execute the parallel-hybrid IPUM algorithm. Nord-III is an intel sandy bridge processors based supercomputer system having idataplex compute racks interconnected by using InfiniBand interconnection and utilizes SUSE Linux Operating System. The parallel-hybrid processing system uses 16 nodes of Nord-III, each node is equipped with 16 Intel processing cores working at a frequency of 2.6 GHz.

V. Results and Discussion

In this section, the performance and scalability of the proposed parallel-hybrid IPUM algorithm are measured by executing it on a different number of cores and nodes of Nord-III supercomputer. (Discussed in Section IV-D). The section is further subdivided into four subsections; experimental setup, accuracy, performance comparison and scalability.

A. *Experimental Setup*

The Interferometry Systems are experimentally tested for the IPUM algorithm by using the experimental laboratory setup shown in Figure 7. The target (Figure 7 (a)) is mounted on a mechanical wave driver PASCO SF-9324 having a frequency range of 0.1 Hz to 5 kHz. The function generator is used to excite the target at a frequency of 80 Hz with a peak to peak amplitude of 5 μm. To obtain the SM signal, a Hitachi HL6501MG laser diode is used (shown in Figure 7 (c)) having an emission wavelength of 658 μm, the threshold current of 75 mA and with the optical output power of 35 mW. An oscilloscope is attached to the sensor electronic circuitry to observe the generated SM signal.

TABLE I
Sequential IPUM's processing time as a function of remote target's velocity (leading to different number of input data-sets (samples/second))

$v_t(t)$ meter/sec	10x10^{-6}	100x10^{-6}	1x10^{-3}	10x10^{-3}	100x10^{-3}
Input Samples	400	4×10^3	40×10^3	400×10^3	4×10^6
Time (sec)	0.026	.286	2.98	30.6	320.8

The displacement retrieval results are compared from a reference commercial piezoelectric (PZT) sensor from Physik Instrument (P753.2CD) having 2nm accuracy. Displacement retrieval for a target motion of 80 Hz and its comparison with the reference PZT sensor is shown in Figure 8. Figures 8 (a) and (b) show the SM signal and retrieved target displacement using the PZT sensor respectively. Figures 8 (c) and (d) show the displacement retrieval using the IPUM algorithm and its error comparison with the PZT sensor. The IPUM retrieves the target displacement with an RMS error of 10.2 nm. The error is calculated by taking the difference of retrieved displacement using IPUM (shown in Figure 8 (c)) and reference PZT sensor displacement (Figure 8 (b)).

B. *IPUM Performance in case of High Bandwidth Sensing Applications*

To measure the performance of the sequential and parallel-hybrid versions of IPUM as a function of remote target's velocity $v_t(t)$, the following strategy is used. Assume for the sake of simplicity that the SM laser sensor uses a laser diode with a wavelength (λ) of 1 μm. As 20 samples per fringe are at-least required to accurately detect a fringe [39], so the SM signal's sampling rate needs to be at-least 40 samples per second to accurately measure $v_t(t) = \lambda m/s = 1\mu m/s$. Consequently, performance of algorithm in processing SM signals corresponding to $v_t(t)$ of 10 μm/s, 100 μm/s, 1 mm/s, 10 mm/s and 100 mm/s, the systems require 400, 4K, 40K, 400K and 4M samples per second respectively. Likewise,

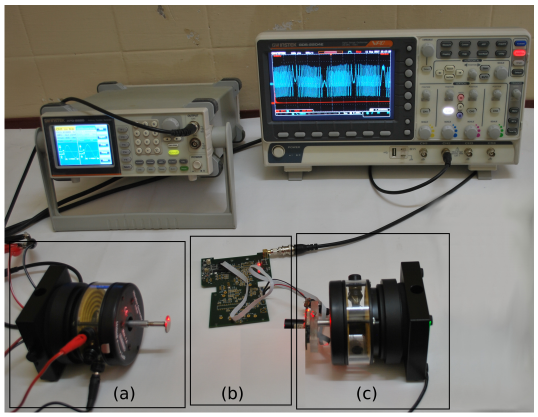


Fig. 7. SMI Laboratory Setup: (a) PASCO SF-9324 Mechanical Shaker Used as Moving Target (b) DL7140 Laser Diode Based Sensor Electronics Circuitry (c) DL7140 Based Laser Sensor Mounted on Mechanical Shaker

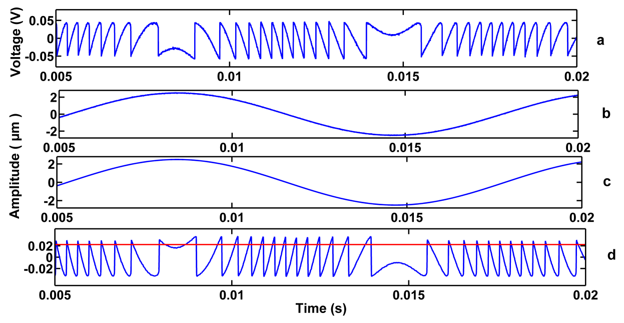


Fig. 8. Experimental Results: (a) Experimentally Acquired SM Signal (b) Displacement Measured by the Reference PZT Sensor (c) Displacement Retrieval from the IPUM Algorithm (d) IPUM Error With Respect to the Reference PZT Sensor

TABLE II
PARALLEL-HYBRID IPUM: EXECUTION TIME (IN SECONDS) AGAINST DIFFERENT NUMBER OF PROCESSING CORES OF A NODE

Number of Cores	1	2	4	8	12	16
Time (sec)	320.5	161.5	83.3	43.29	29.17	22.1

for accurate estimations of coupling factor C and linewidth enhancement factor α, higher values of iterations of C_iter and α_iter of 200 and 100, respectively are used.

The parallel-hybrid and sequential IPUM algorithms are tested on 400 MegaByte of dataset stored in external memory. During execution time, the algorithm takes different input samples. Table II shows the averaged time while executing the input sample data (shown in Table I). Table I shows the execution time of the sequential IPUM algorithm on a uni-core of a node for a different number of the input sample. The results show the sequential IPUM algorithm is compute-intensive and its performance requirements increase with the increased input number of samples.

TABLE III
PARALLEL-HYBRID IPUM ALGORITHM SCALABLITY: EXECUTION TIME AGAINST DIFFERENT NUMBER OF DISTRIBUTED NODES

Nodes	1	2	4	8	16	32
Execution Time (Sec)	22.1	11.9	6.38	3.21	1.66	0.98

C. Performance Comparison and Scalability of Parallel-hybrid IPUM

In this experiment, the performance of the parallel-hybrid IPUM algorithm is measured by executing it on a different number of processing cores of Nord-III Supercomputer (discussed in Section IV-D) and then compared with sequential IPUM algorithm. The parallel-hybrid and sequential IPUM algorithms are tested on 400 MegaByte of dataset stored in external memory. During the experiments, we mention the average time taken by the parallel-hybrid and sequential algorithms to execute 100 copies of each having 4 Mega samples of data. The algorithms use C_iter and α_iter of 200 and 100, respectively.

Table II shows performance improvement while executing 4 Mega samples of input data on a different number of processing cores. The average time includes local, shared memory and external memories read/write, processing and interprocess communication time of shared memory system. The results show that the sequential IPUM algorithm on a uni-core system takes 2780 seconds. When compared to the performance of parallel-hybrid IPUM against the sequential IPUM algorithm, the results show that parallel-hybrid IPUM on 2, 4, 8, 12 and 16 cores improve 1.9, 3.8, 7.2, 11, and 14.54 times performance respectively.

The scalability of the parallel-hybrid IPUM algorithm is measured by running the algorithm on a different number of nodes.The execution time of the Parallel-IPUM includes processing time on multiple processing cores, memory read-/write and MPI communications time between the distributed processing system. Table III shows that the execution time of parallel-hybrid IPUM reduces with an increase in the number of distribution nodes. The results show that parallel-hybrid IPUM is highly scalable and can perform load balancing. The parallel-hybrid IPUM on a distributed system manages processing resources and allocates more processing cores to the compute-intensive IPUM tasks. While comparing the execution time of parallel-hybrid on uni-node with 2, 4, 8, 16 and 32 multi-node execution, the results show that the algorithm achieves 1.84, 3.6, 6.85, 13.2, and 23.78-times performance improvement respectively. The results confirm that the parallel-hybrid on multi-node gives 345.9x times performance improvement against the sequential IPUM.

VI. CONCLUSION

In this work, we have implemented a parallel hybrid version of the phase retrieval algorithm called parallel-hybrid IPUM and executed it on a supercomputing system. The existing sequential IPUM C program has been parallelized by using parallel-hybrid OpenMP/MPI (Open Multi-Processing/Message Passing Interface) parallel programming models and executed on Nord-III supercomputing system. The implementation involves a thorough understanding and segmentation of the compute-intensive parts of the algorithm leading to their pipelining, parallelizing, and mapping on multiple cores of the distributed computing system architecture. To validate the performance and scalability of parallel-hybrid IPUM, the algorithm is tested on multiple distributed nodes of Nord-III Supercomputer of Barcelona Supercomputing Center. While executing parallel-hybrid IPUM on a different number of processing cores the algorithm achieves up to 345.9x times performance improvement. This shows that the parallel-hybrid version of IPUM gives a scalable high performance for different target velocities.

REFERENCES

[1] Jianjiang Feng and Anil K Jain. Fingerprint reconstruction: from minutiae to phase. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):209–223, 2011.

[2] Curtis W Chen and Howard A Zebker. Phase unwrapping for large sar interferograms: Statistical segmentation and generalized network models. *IEEE Transactions on Geoscience and Remote Sensing*, 40(8):1709–1719, 2002.

[3] R Cusack and N Papadakis. New robust 3-d phase unwrapping algorithms: application to magnetic field mapping and undistorting echoplanar images. *Neuroimage*, 16(3):754–764, 2002.

[4] Song Zhang, Xiaolin Li, and Shing-Tung Yau. Multilevel quality-guided phase unwrapping algorithm for real-time three-dimensional shape reconstruction. *Applied Optics*, 46(1):50–57, 2007.

[5] M Ramji and K Ramesh. Adaptive quality guided phase unwrapping algorithm for whole-field digital photoelastic parameter estimation of complex models. *Strain*, 46(2):184–194, 2010.

[6] Jos M Bioucas-Dias and Gonalo Valadao. Phase unwrapping via graph cuts. *IEEE Transactions on Image processing*, 16(3):698–709, 2007.

[7] Thomas Taimre, Milan Nikolic, Karl Bertling, Yah Leng Lim, Thierry Bosch, and Aleksandar D. Rakic. Laser feedback interferometry: a tutorial on the self-mixing effect for coherent sensing. *Advances in Optics and Photonics*, 7(3):570–631, 2015.

[8] Olivier D Bernal, Usman Zabit, Francis Jayat, and Thierry Bosch. Sub-$\lambda/2$ displacement sensor with nanometric precision based on optical feedback interferometry used as a non-uniform event-based sampling system. *IEEE Sensors Journal*, 2020.

[9] L. Lu, W. Zhang, B. Yang, J. Zhou, H. Gui, and B. Yu. Dual-channel self-mixing vibration measurement system in a linear cavity fiber laser. *IEEE Sensors Journal*, 13(11):4387–4392, Nov 2013.

[10] O. Bernal, U. Zabit, and T. Bosch. Robust method of stabilization of optical feedback regime by using adaptive optics for a self-mixing microinterferometer laser displacement sensor. *Selected Topics in Quantum Electronics, IEEE Journal of*, 21(4):1–8, July 2015.

[11] Yongbing Zhang, Yingbin Wei, Chenxi Chen, Wencai Huang, Xiulin Wang, and Huizhen Xu. Self-mixing interferometer based on frequency analysis method for accurate refractive index measurement. *IEEE Photonics Journal*, 8(2):1–6, 2016.

[12] T Hagelschuer, M Wienold, H Richter, L Schrottke, K Biermann, HT Grahn, and H-W Hübers. Terahertz gas spectroscopy through self-mixing in a quantum-cascade laser. *Applied Physics Letters*, 109(19):191101, 2016.

[13] U. Zabit, O. D. Bernal, and T. Bosch. Design and analysis of an embedded accelerometer coupled self-mixing laser displacement sensor. *Sensors Journal, IEEE,*, 13(6):2200–2207, June 2013.

[14] Silvano Donati and Michele Norgia. Self-mixing interferometry for biomedical signals sensing. *IEEE Journal of Selected Topics in Quantum Electronics*, 20(2):104–111, 2014.

[15] Ayesha Ehtesham, Usman Zabit, Olivier D Bernal, Gulistan Raja, and Thierry Bosch. Analysis and implementation of a direct phase unwrapping method for displacement measurement using self-mixing interferometry. *IEEE Sensors Journal*, 17(22):7425–7432, 2017.

[16] Usman Zabit, Olivier D Bernal, Saqib Amin, Muhammad Farrukh Qureshi, Arsalan Habib Khawaja, and Thierry Bosch. Spectral processing of self-mixing interferometric signal phase for improved vibration sensing under weak-and moderate-feedback regime. *IEEE Sensors Journal*, 19(23):11151–11158, 2019.

[17] Alessandro Magnani, Alessandro Pesatori, and Michele Norgia. Self-mixing vibrometer with real-time digital signal elaboration. *Applied Optics*, 51(21):5318–5325, 2012.

[18] A. L. Arriaga, F. Bony, and T. Bosch. Real-time algorithm for versatile displacement sensors based on self-mixing interferometry. *Sensors Journal, IEEE*, 16(1):195–202, 2016.

[19] S. Merlo and S. Donati. Reconstruction of displacement waveforms with a single-channel laser-diode feedback interferometer. *IEEE Journal of Quantum Electronics*, 33(4):527–531, 1997.

[20] C. Bes, G. Plantier, and T. Bosch. Displacement measurements using a self-mixing laser diode under moderate feedback. *Instrumentation and Measurement, IEEE Transactions on*, 55(4):1101–1105, Aug 2006.

[21] Y. Fan, Y. Yu, J. Xi, and J. F. Chicharo. Improving the measurement performance for a self-mixing interferometry-based displacement sensing system. *Applied Optics*, 50(26):5064–5072, 2011.

[22] O.D. Bernal, U. Zabit, and T. Bosch. Study of laser feedback phase under self-mixing leading to improved phase unwrapping for vibration sensing. *Sensors Journal, IEEE*, 13(12):4962–4971, Dec 2013.

[23] Yufeng Tao, Ming Wang, and Dongmei Guo. Compound cavity theory of resonant phase modulation in laser self-mixing ultrasonic vibration measurement. *Optical Engineering*, 55(7):074107–074107, 2016.

[24] Yah Leng Lim, Russell Kliese, Karl Bertling, Katsuyoshi Tanimizu, PA Jacobs, and Aleksandar D Rakić. Self-mixing flow sensor using a monolithic vcsel array with parallel readout. *Optics express*, 18(11):11720–11727, 2010.

[25] Dario Melchionni, Alessandro Magnani, Alessandro Pesatori, and Michele Norgia. Development of a design tool for closed-loop digital vibrometer. *Applied optics*, 54(32):9637–9643, 2015.

[26] U. Zabit, O. D. Bernal, A Chamorro-Coloma, and T. Bosch. Real-time accelerometer coupled self-mixing laser displacement sensor for embedded applications. *Sensors, 2012 IEEE*, pages 1,4,28–31, Oct. 2012.

[27] Michele Norgia, Dario Melchionni, Alessandro Magnani, and Alessandro Pesatori. High-speed self-mixing laser distance sensor. In *11Th International Conference On Vibration Measurements By Laser And Noncontact Techniques-Aivela 2014: Advances And Applications*, volume 1600, pages 422–425. AIP Publishing, 2014.

[28] Federico Cavedo, Alessandro Pesatori, Michele Norgia, Politecnico di Milano, and Gabriel E Solari. Laser rangefinder for steel pipes characterization. In *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, pages 1387–1390. IEEE, 2015.

[29] Alessandro Magnani, Alessandro Pesatori, and Michele Norgia. Real-time self-mixing interferometer for long distances. *IEEE Transactions on Instrumentation and Measurement*, 63(7):1804–1809, 2014.

[30] Sebastien Ducourtieux and Benoit Poyet. Development of a metrological atomic force microscope with minimized abbe error and differential interferometer-based real-time position control. *Measurement Science and Technology*, 22(9):094010, 2011.

[31] Chen Ji, Wang Xin, Cao Dajiu, and Zhou Zhaofeng. Development of high-speed ccd laser displacement sensor [j]. *Optics and Precision Engineering*, 16(4):616, 2008.

[32] Chen Wang. *FPGA-based, 4-channel, High-speed Phasemeter for Heterodyne Interferometry*. 2013.

[33] Olivier D. Bernal, Usman Zabit, and Thierry Bosch. Classification of laser self-mixing interferometric signal under moderate feedback. *Appl. Opt.*, 53(4):702–708, Feb 2014.

[34] S. Donati, G. Martini, and T. Tambosso. Speckle pattern errors in self-mixing interferometry. *IEEE Journal of Quantum Electronics*, 49(9):798–806, Sept 2013.

[35] Reza Atashkhooei, Santiago Royo, Francisco Azcona, and Usman Zabit. Analysis and control of speckle effects in self-mixing interferometry. In *Sensors, 2011 IEEE*, pages 1390–1393. IEEE, 2011.

[36] Olivier Bernal, Han Cheng Seat, Usman Zabit, Frédéric Surre, and Thierry Bosch. Robust detection of non regular interferometric fringes from a self-mixing displacement sensor using bi-wavelet transform. *IEEE Sensors Journal*, 16(22):7903, 2016.

[37] Antonio Luna Arriaga, Francis Bony, and Thierry Bosch. Speckle-insensitive fringe detection method based on hilbert transform for self-mixing interferometry. *Applied Optics*, 53(30):6954–6962, 2014.

[38] Imran Ahmed, Usman Zabit, and Ahmad Salman. Self-mixing interferometric signal enhancement using generative adversarial network for laser metric sensing applications. *IEEE Access*, 7:174641–174650, 2019.

[39] Usman Zabit, OD Bernal, and Thierry Bosch. Self-mixing sensor for real-time measurement of harmonic and arbitrary displacements. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pages 754–758. IEEE, 2012.