High speed „real-time“ multi-λ PS and FT interferometry P33

Joachim Wesner, J. Heil, W. Müller, M. Netsch, H.M.Heuck
Leica Microsystems CMS GmbH
Ernst-Leitz-Strasse 17-37
D-35578 Wetzlar, Germany

“Real-time” temporal phase shifting interferometry (TPSI)

- Any Interferometer is a highly parallel instrument, with “lightspeed” response
- The resulting direct feedback can be extremely versatile, especially in fine-adjusting single lenses or whole lens-systems
- However, what is lacking is the quantitative analysis capability only achievable with digital processing of the interference signal, which is commonplace today
- However, with a few exceptions it seems that interferometric software has not yet really benefited from the continuing increase in available processing power, leaving the cycle time for a full analysis still at typically > 1sec, only using the speed benefit for data averaging[1]
- A real-time display allows a much more direct estimate by the operator of drifts, measurement reproducibility, “RMS noise” and glitches without any elaborate statistical analysis
- **TPSI, while representing a very well established technique, is usually considered especially hampered due to low speeds and extra noise sensitivity during the multiple buckets that need to be acquired. However as we will show, TPSI with full analysis can be pushed to a total cycle times of less ≤ 70 ms @ 512x512 pixels using current PC hardware**

Related or earlier work

- (1990) **Zeiss DIRECT 100**: 4..20ms camera + 40 ms processor @ 480x480 Pixels, Spatial PSI[2]
  - **Hardware**: Real-time raw phase calculation (PSI & atan2, averaging, reference subtraction, artificial interferogram of higher sensitivity). → Massively pipelined hardware solution, ~ 60 ms/cycle
  - **Software**: Unwrapping, PSF, MTF, Aberrations, Zernike-Coefficients etc., Speed unknown but cycle time likely > 1 s
- (2011) **Zygo Dynafiz**[3] (Fringe processing FTI or possibly also Static Phase Shifting interferometry SPSI?), 1200x1200 pixels, 82 Hz camera, hardware?

Current advances of (desktop) digital processing speed

- Double precision FFTW[4] “MFlops” 256x256

Athlon XP 3000+	ARM-A9 @1 GHz	Intel i7 @ 3.6GHz
~1000 MFlops	~300 MFlops	~80 Gflops (all cores)

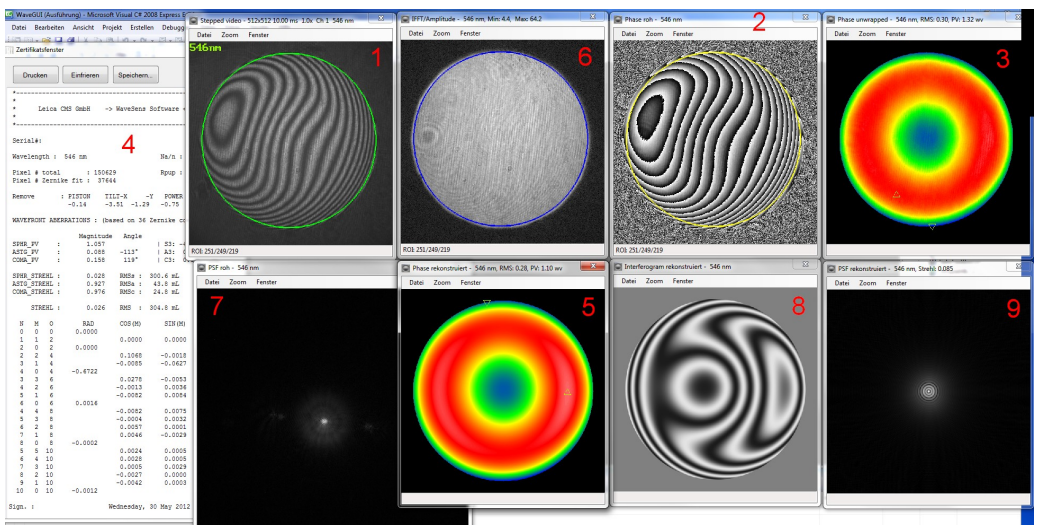
➔ **Today, a *single 3GHz core* can do the work of the DIRECT 100 hardware at similar or higher speed (30ms, see below)**, (and a current smartphone with a single core is comparable to a decent workstation of the year ~2000!!!)

Basic quantitative interferometry processing blocks

- Calculate Re and Im of complex wavefront (from PSI or FTI-Input (**1**))
- Determine wrapped phase (atan2) (**2**, see figure below for labels)
- Optionally detect pupil rim and inconsistent/low quality regions (Residues, phase derivative variance)
- Phase unwrapping (**3**), determine RMS and PV values
- Zernike fitting and removal of piston, tilt and defocus
- Output Zernikes, Seidels, Strehl ratios (**4**), reconstructed wavefront (**5**)

Extended quantitative interferometry processing blocks

- Determine intensity (**6**) resp. modulation of the complex wavefront to check for any apodization and to better detect the pupil rim
- Calculate **raw** PSF from **complex wavefront** (**7**)
- Subtract a (previously) acquired or defined reference wavefront
- Calculate artificial interferograms (**8**) and PSFs (**9**) from (selected) Zernikes



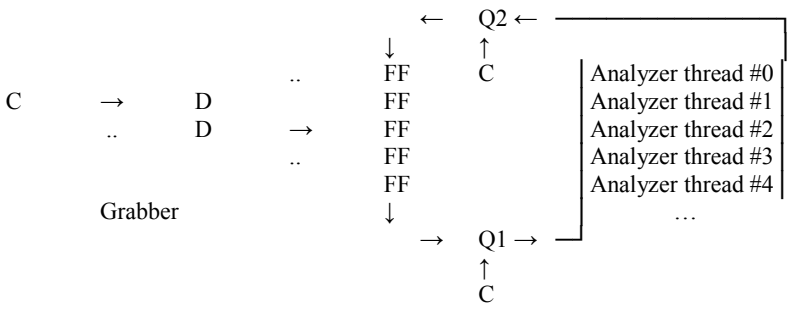
Example of what may be measured and calculated within 70 ms

Multi-threaded processing

- As there is no data dependency between individual PSI frames (of N buckets), the problem decomposes nicely in one high-priority “grabber” thread, that controls camera and phase shifter and communicates with up to **M** low priority “analyzer” threads
- This is a low-overhead “coarse-grained” solution, in contrast to other more “automatic” approaches that parallelize on the basis of a “for/do loop” (“Fine-grained” multi-threading as for example in Matlab)
- The grabber threads assembles all buckets in a PSI frame and, together with other data that uniquely describe the frame and all relevant hardware settings, puts the result in a common waiting queue for the analyzers
- Even if the grabber thread does not do any heavy calculations, it is nevertheless very time critical, otherwise apparent “shift errors” might appear.

Data buffering and queuing, command injection

- Camera data are acquired in a tightly coupled double buffer **D** (MIL assisted)
- Data are assembled in **N**-bucket frames **F** and put in a thread-safe waiting queue **Q1** (“Producer-Consumer problem with > 1 consumers”[5])
- “Used” Resp. “Empty” frame buffers are put back in a second thread-safe waiting queue **Q2**
- Commands to the grabber or the analyzers can be injected in both queues as special entries **C**
- The number of buffers in the waiting queues is strictly limited to a very small numbers (typ. 3) to reduce latency, for better “real-time” response.



Camera, Shifter and PC synchronization

- There is no direct synchronization (trigger) from PC to camera, which runs “free” at highest speed. (This is backward compatible to analog video cameras)
- The PC is synchronized on the data ready event from the camera, which also controls the restart of the piezo ramp in the right moment (after **N+1** buckets)
- Timing has to include the transfer time over the FW bus from camera to PC
- Any piezo ramp is independently output in the background and can be calculated to compensate for piezo nonlinearities.
- For a shift range ≥ 2π rads, only buckets 2π rads apart are shown in the “live” window, so the operator can anytime check for correct piezo drive, i.e. when any apparent “jitter” completely disappears. Similarly for a shift range > 2π rads, the compensation of piezo nonlinearity can be verified in 1st order.

Properties of HW and SW employed in our system

- Dell Optiplex 790 with 4 GB Ram and Intel i7 2700k, 4 Cores, 8 Threads (HT)
- Matrox MIL 9.2 for frame grabbing (FW/GigE) and display (thread-safe!)
- NI 6014 Multifunction ADDA interface card to generate ramp in background
- Program is written in C# 3.5 with most interferometry related functions in native C-Dlls, including FFTW 3.0 library, highly optimized Zernike Routines and two fast path-following unwrappers similar to Goldstein's algorithm.
- Uses managed (.NET based) multi-threading[5]

Timing results for one i7 core (4+1 buckets, 72 Hz camera, 14.4 Hz overall, 512x512 data, r_{pupil} = 220 Pixels)

PSIanalysis	Intensity	PSFRaw	Unwrap	ZernFit	ZernRec	ZernIntf	PSFRec	
24	5	32	38	28	15	25	24	ms

- Σ = 191 ms, Σ/4 = 48 ms → Processing speed is actually limited by camera speed, not by CPU speed (overall CPU load ~ 46% due to HT cores, actually 8 analyzer threads are started)

[1] www.trioptics-berlin.de, “Knowledge Base: Phase Shifting”
[2] M. Küchel & B, Dörband, “Jahrbuch für Optik und Feinmechanik 1992“, 90-111
[3] D. M. Sykora & M.L.Holmes, Proc. SPIE 8082, 80821R-1, (2011)
[4] www.ffw.org
[5] J. & B. Albahari, “C# 4.0 in a Nutshell”, Chap. 21 “Threading”, Safari Books