Generating DIC speckle patterns via ray tracing for uncertainty quantification on curved surfaces

Wiera Bielajewa [1a], Joel Hirst [1], Lorna Sibson [1,2], Adel Tayeb [1], Megan Sampson [1], Michael Atkinson [1], Alex Marsh [1], Rory Spencer [1], Rob Hamill [1], Lloyd Fletcher [1]

[1] United Kingdom Atomic Energy Authority, Culham Campus, Abingdon, OX14 3DB, UK
[2] School of Electrical and Electronic Engineering, University of Sheffield, Sheffield, S1 4DT, UK

[a] wiera.bielajewa@ukaea.uk

Abstract. The Pyvale Python package [1] is designed as a full-lifecycle toolset for sensor modelling, covering sensor behaviour simulation, Uncertainty Quantification (UQ), sensor placement optimisation, and calibration together with validation. With a specific emphasis on optical sensing, notably Digital Image Correlation (DIC) [2], the package requires precise modelling of both systematic and random imaging uncertainties. To simulate camera scenes, computer graphics generally utilise two methods: rasterisation, which is optimised for real-time speed [3], and ray tracing, which offers superior physical realism by simulating complex light interactions like shadows and reflections [4]. Because lighting and optical distortions directly impact DIC precision, these effects should be accurately captured. This research compares ray tracing with the common rasterisation-based visualisation tools, such as PyVista, for generating deformed speckle pattern images derived from solid mechanics simulations. The focus is on rendering speckled objects composed of the higher-order elements. These images are subsequently post-processed by the Pyvale DIC engine. The results indicate that conventional visualisation tools struggle to accurately simulate the interplay of reflections when rendering higher-order curved elements. This is because they employ coarse sub-tessellation, which accelerates the rendering process but produces an abundance of undesirable lighting effects. By quantifying how these rendering choices affect DIC displacement and strain field calculations, this work establishes a framework for balancing speed and accuracy within Pyvale. These insights will ultimately strengthen the validation pipeline for DIC UQ.

Introduction

The advancement of high-fidelity structural simulations has created a growing demand for accurate virtual representations of experimental setups, particularly because the calibration and construction of reliable physical setups are time-consuming and resource-intensive. Pyvale [1] is a comprehensive toolset designed to manage the entire lifecycle of sensor modelling. It integrates various functionalities, from simulating sensor behaviours and quantifying uncertainty to optimising sensor placement and performing simulation calibration/validation.

A primary focus of this package is optical sensing, specifically Digital Image Correlation (DIC) [2]. DIC relies on tracking speckle patterns on a material's surface to measure deformation. However, the accuracy of these measurements is inherently tied to the characteristics of the imaging system and the parameters used to perform the DIC analysis. To allow for quantitative use of DIC measurements, it is essential to characterise both systematic and random uncertainties. This requires the generation of synthetic images that accurately reflect the complex interplay between light, surface geometry, camera optics, and DIC processing parameters. Such an approach is particularly valuable given that DIC calibration is both time-consuming and highly sensitive, as even minor camera movements or changes in lighting conditions can necessitate repeated recalibration.

However, the visualisation tools commonly used to render simulation results (PyVista, ParaView, Netgen etc) utilise sub-tessellation, thus subdividing higher order curved surface elements into a few linear triangles. This is beneficial for rendering speed but bears detrimental consequences for the visualisation accuracy, which is highly important when generating synthetic image for DIC UQ. Therefore, in this work, a ray-tracing rendering engine has been developed as a module in Pyvale. It is specifically designed for DIC UQ and support accurate rendering of higher order finite elements necessary for DIC UQ.

Results

There are various visualisation tools available for rendering finite element (FE) meshes and simulation results, e.g. PyVista, ParaView, Netgen etc. They typically rely on the Visualization Toolkit (VTK) and therefore inherit its core rendering pipeline. VTK uses OpenGL, which is a rasterisation-based rendering engine. When rendering higher order elements, this engine sub-divides the curved surface elements into just a few linear triangles and renders these triangles instead of truly representing the original surface. Fig. 1 illustrates this problem. A sphere composed of 2nd order tetrahedra, which exactly represent the sphere's surface, is coarsely rendered by VTK-based packages. Each 2nd order curved surface triangle is split into 4 linear triangles.

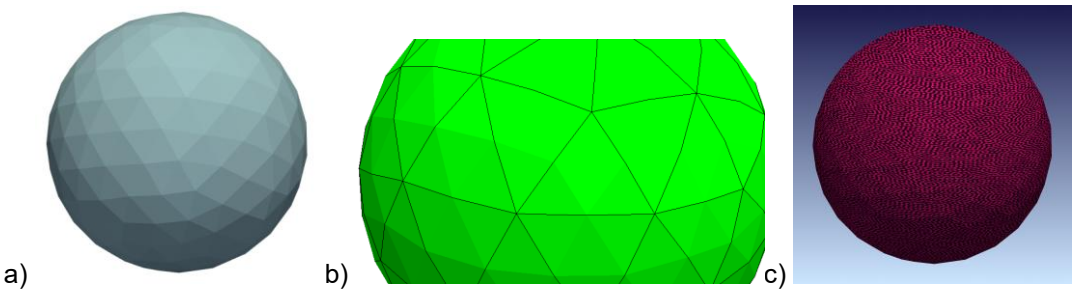

Figure 1: Visualisations of a sphere meshed using quadratic (10-node) tetrahedra illustrating a coarse division of each quadratic surface element into 4 linear triangles, a) PyVista, b) Netgen with the rough edges of the original quadratic elements outlined in black, c) sphere with applied speckle pattern rendered in PyVista with visible shadows in the lower hemisphere artificially created by sub-tessellation

Pyvale's ray tracer does not experience this issue and is able to accurately and truly render higher order elements. Fig. 2a shows the aforementioned sphere with applied speckle pattern accurately rendered by Pyvale ray tracer. The sphere's surface is assumed to be specular. On the other hand, Fig. 2b displays the same sphere rendered by Pyvale ray tracer but with sub-tessellation emulating the one performed by VTL-based packages, i.e. one quadratic triangle sub-divided into 4 linear triangles. The difference between Fig. 2a and 2b is stark as illustrated in Fig. 2c showing the absolute errors in pixel intensity levels between these two images. The sub-tessellation introduces large errors to the edges of the sphere. Furthermore, it also ensures incorrect rendering of the shadow on the lower hemisphere and reflections on the upper hemisphere from two other speckled objects located in the vicinity. Thus, the significant advantage of Pyvale ray tracer for the accurate generation of speckle patterns is showcased.

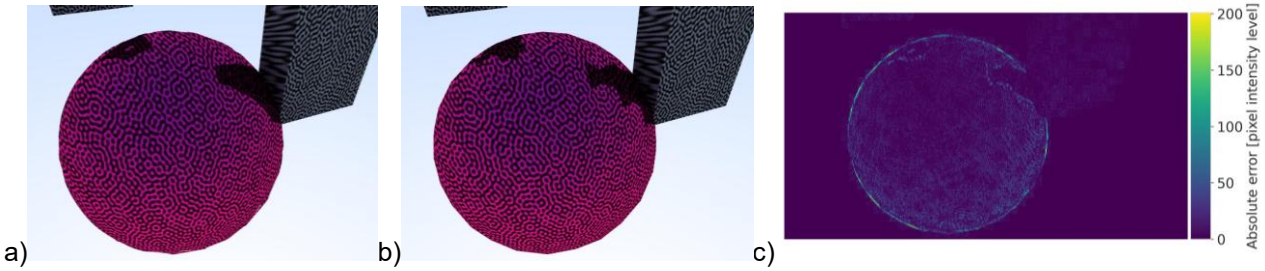

Figure 2: Visualisations of a sphere meshed using quadratic (10-node) tetrahedra with applied speckle pattern, a) Pyvale ray tracer, b) Pyvale ray tracer but with sub-tessellation emulating the one performed by VTL-based packages, i.e. one quadratic triangle sub-divided into 4 linear triangles, c) absolute errors in pixel intensity levels between a) and b), with pixel intensity levels ranging between 0 and 255

Conclusion

A custom ray tracing pipeline has been developed as a module within Pyvale. It accurately renders higher order elements and is specifically focused on DIC UQ.

References

[1] Pyvale: The python validation engine, https: //computer-aided-validation-laboratory.github. io/pyvale/index.html, release 2025.8.1, MIT License (2025).

[2] B. Pan, Digital image correlation for surface deformation measurement: Historical developments, recent advances and future goals (2018). doi:10.1088/1361-6501/aac55b.

[3] M. Schutz, B. Kerbl and M. Wimmer. Software Rasterization of 2 Billion Points in Real Time. Proceedings of the ACM on Computer Graphics and Interactive Techniques, Vol. 5, Issue 3, pp. 17, 2022.

[4] P. Shirley, Ray Tracing in One Weekend (2016).