


# Deformation tracking of truss lattices under dynamic loading based on Digital Image Correlation

**Journal Article****Author(s):**

Radi, Kaoutar; Allamand, Fabian; [Kochmann, Dennis M.](#) 

**Publication date:**

2023-08

**Permanent link:**

<https://doi.org/https://doi.org/10.3929/ethz-b-000612865>

**Rights / license:**

[Creative Commons Attribution 4.0 International](#)

**Originally published in:**

Mechanics of Materials 183, <https://doi.org/10.1016/j.mechmat.2023.104658>



Research paper

# Deformation tracking of truss lattices under dynamic loading based on Digital Image Correlation

Kaoutar Radi, Fabian Allamand, Dennis M. Kochmann\*

Mechanics &amp; Materials Lab, Department of Mechanical and Process Engineering, ETH Zürich, 8092 Zürich, Switzerland

## ARTICLE INFO

Dataset link: [https://gitlab.com/kaoutar\\_radi/nodes-displacement-tracking](https://gitlab.com/kaoutar_radi/nodes-displacement-tracking)

## Keywords:

Digital Image Correlation  
Truss  
Metamaterials  
Dynamics  
3D printing

## ABSTRACT

Three-dimensional (3D) truss-based metamaterials (or architected materials) have been the subject of an increasing amount of studies, owing to their impressive and tunable mechanical properties. Unfortunately, experimentally studying their mechanical multiscale behavior under dynamic loading presents a challenge due to the complex time-dependent correlation between the overall truss response and the deformation of individual beams and beam junctions. Tracking the time-dependent deformation of 3D-printed low-density trusses is challenging for traditional techniques such as Digital Image Correlation (DIC), owing to the discrepancy between typical feature (strut) sizes and the field of view of the overall truss, further difficulty in applying speckle patterns, as well as the lack of a stable bright background during testing—especially when high rates and large 3D deformation are involved. As a remedy, we present an efficient DIC-based technique, which admits tracking the nodal displacements of points of interest (such as, e.g., the strut junctions) in periodic and non-periodic truss networks across a range of loading rates. We use this technique to identify the time-dependent nodal displacements of different truss architectures, whose large deformation is of interest for energy absorption capabilities of truss metamaterials. The quality of results is assessed by performing multiple trackings on each truss topology, which reveals that errors are negligible for the reported range of conditions.

## 1. Introduction

Metamaterials and, in particular, truss-based architected materials have allowed for the exploration of previously uncharted regions in the material property space, pushing the boundaries of material property combinations that can be achieved with conventional, monolithic materials. Truss metamaterials owe their properties to their multiscale architecture with design principles ranging from the geometry of individual struts and junctions to the topology of periodic or non-periodic truss networks, combined with the properties of the base material and, possibly, material size effects at small scales (Meza et al., 2014a; Meza and Greer, 2014; Sun and Litchinitser, 2016; Mohsenizadeh et al., 2018). Apart from high stiffness and strength at low mass density, the macroscopic, effective deformation behavior of such architected materials differs significantly from that of bulk materials of comparable density and, most importantly, can be controlled through the multiscale truss architecture. Examples of tunable performance include, among others, large elastic strains in otherwise brittle materials (Meza et al., 2014b), negative Poisson's ratio (Yasuda and Yang, 2015; Babaee et al., 2013), negative compressibility (Grima et al., 2008), and mechanical cloaking (Bückmann et al., 2015). Many studies have focused on the effective mechanical properties for a variety of load scenarios from

quasistatics to impact testing, from which the effective, macroscopic response is obtained in the form of, e.g., load–displacement relations. By contrast, the microstructural deformation mechanisms responsible for the macroscopic behavior (which involves the deformation of, e.g., individual struts within a large three-dimensional (3D) truss network) are significantly more challenging to characterize quantitatively in an experiment—especially at high rates.

To experimentally characterize displacements, a variety of methods have been developed. For example, interferometry has been used for precise displacement measurements in one dimension (1D), with further distinctions such as shearography (Bai et al., 2015), Moiré and speckle interferometry for additional information on two-dimensional (2D) stress fields (Chen and Basaran, 2008; Chen et al., 2003). For 3D displacement fields, Bhaduri et al. used a combination of Digital Speckle Pattern Interferometry (DSPI) and Digital Speckle Photography (DSP) (Bhaduri et al., 2011), while others utilized Particle Image Velocimetry (Yuan et al., 2019), Phase Contrast Magnetic Resonance Imaging (Draney et al., 2002) – even for in-vivo measurements – or X-ray computer tomography in combination with Digital Image Correlation (DIC) (Roux et al., 2008) and ready-to-use RGB-D cameras, which are often based on infrared light (Abdelbarr et al., 2017).

\* Corresponding author.

E-mail address: [dmk@ethz.ch](mailto:dmk@ethz.ch) (D.M. Kochmann).<https://doi.org/10.1016/j.mechmat.2023.104658>

Received 26 October 2022; Received in revised form 9 March 2023; Accepted 10 April 2023

Available online 9 May 2023

0167-6636/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Owing to the rapid advances in optical and computer vision technologies, DIC has been extensively employed in various engineering applications, mainly to measure surface deformation. The principle of DIC is the application of a correlation analysis to compare digital images taken before and after deformation. During experiments, artificially applied (or naturally existing) speckle patterns on an otherwise bright and uniform specimen surface are recorded by cameras. As the patterns are approximately random, they can be assumed unique at two different spatial positions. Roughly speaking, the speckles within a region of an undeformed reference image are mathematically translated, rotated, and deformed, until the best fit to the same region in the deformed image is obtained. The quality of the speckle patterns significantly affects the accuracy of the correlation results.

Several studies have demonstrated that DIC can be used for tracking the deformation of low-density metamaterials under certain conditions. Maraghechi et al. (2020) used DIC on miniaturized cellular elastic metamaterials to analyze size effects and to calculate full-field displacements. In a different experiment, the pivots of a pantographic metamaterial were marked and a random speckle pattern was created by spraying black and white paint on a truss to measure the dynamic behavior of the sheet undergoing sinusoidal displacements (Dell'Isola et al., 2019). A DIC-based technique to measure the in-plane elastic wave propagation in truss lattices was developed by Schaeffer et al. (2017), who analyzed the motion of trusses undergoing small deformation by tracking the positions of the truss junctions from images taken by a high-speed camera. To avoid the manual application of speckles, Niu et al. (2017) developed an alternative, which uses DIC to measure displacement fields by projecting virtual patterns that reflect light differently depending on the deformation. Rouwane et al. (2022) developed a DIC-based technique that used complex random structural compositions to create a traceable pattern.

Despite the wide range of existing experimental techniques, they all have limitations when it comes to tracking the deformation of complex networks of slender struts, such as in truss-based metamaterials. The latter comes with a number of challenges. (i) Struts are slender and may have diameters on the order of a few millimeters and below so that applying speckles is non-trivial (besides, traditional paint used for applying speckle patterns sticks poorly on 3D-printed polymer samples). In fact, most of the studies cited above reported that finding an appropriate speckle pattern is tedious and strongly depends on the base material, the loading conditions, and the specific architectural features of the structure. (ii) Even if small-scale speckles can be applied (such as in Micro-DIC), the discrepancy between the typical strut thickness to the overall field of view of the entire truss is tremendous, so that resolving the deformation within each beam cross-section while tracking thousands of beams and junctions becomes prohibitively expensive. (iii) The background of a truss' front surface is not bright and constantly changing in dynamic truss experiments (the background involves trusses at many different depth levels, which are deforming significantly over time). (iv) Most of the developed methods are limited to the detection of small displacements that do not cause significant changes to the lattice configuration, which is why they cannot track the dynamic behavior of structures that undergo large deformation. (v) Most of the available DIC codes fail at high rates, especially when tracking 3D deformation. For these reasons, there is a lack of experimental techniques for the characterization of the displacements of 3D truss networks undergoing large time-dependent deformation.

In this work, we present an efficient and simple open-source software tool, which is based on DIC and the Tomasi–Kanade algorithm (Tomasi and Takeo, 1991) and which tracks a set of points of interest (such as the truss junctions) and computes their displacements on the surface of complex 3D truss networks undergoing large deformation. In Section 2, we first present the experimental setup, before describing the process of generating an artificial mask (or grid) on the surface of the truss along with the point tracking algorithm. The method

is applied to 3D-printed periodic and non-periodic (spatially graded) trusses deformed quasistatically and dynamically under compressive and impact loading, respectively, whose results are summarized in Section 3. We further quantify the measurement error and the effect of the strain rate and frame rate on the quality of results. Finally, Section 4 concludes our study. The code described in this manuscript is available online (see the Data Availability Statement).

## 2. Methods

### 2.1. Experimental setup

In order to characterize the dynamic behavior of truss-based architected materials, we designed a simple droptower test frame, which ensures a collinear impact of a free-falling impactor onto the tested samples. The experimental setup, shown in Fig. 1, attaches the impactor to a hollow rod contained within a guiding tube. The rod is held by a release system consisting of an electromagnet that can be switched off to ensure a well-controlled dropping of the impactor. The impactor at the lower end of the guiding tube carries an Injector Control Pressure (ICP) force sensor to measure the force between the impactor, and the sample is clamped onto a platform, which contains a second ICP sensor to measure the force between the sample and the ground. To control the velocity of the impact, the height of the guiding rod is adjustable. Although our impact experiments shown in the following use a flat impactor, the setup is sufficiently general to accommodate other impactors. The position of the impactor and the deformation of the sample are captured by a high-speed camera (Photron FASTCAM Mini AX200), while the forces are registered by the two ICP sensors and extracted via an oscilloscope (Tektronix TBS 2000 series). Using this setup, we performed tests with varying impact velocities on four different truss architectures, which were 3D-printed by selective laser sintering (SLS) using a Sintratec S1 and a Thermoplastic Elastomer (TPE) as the base material—an elastomer that results in flexible, rubber-like parts with an elongation at break of up to 438%.

### 2.2. Creating the artificial grid

Rather than aiming for full-field measurements that characterize the detailed deformation within structural members, we focus on tracking the positions of well-defined characteristic points across the entire truss network. For example, we may track all nodes (i.e., the structural junctions) of a truss over time. Our approach is sufficiently general to apply to any chosen set of points, which is not restricted to the truss junctions but may also include, e.g., points along each strut. The exact choice of the points to be tracked depends on the application. We refer to this set of characteristic points as the *artificial grid* of points to be tracked, and to those points as *nodes* (see, e.g., Fig. 2). In many scenarios tracking such a grid of characteristic points is sufficient to characterize the behavior of the truss, as detailed information about the deformation of individual struts can be extracted from the configuration of the artificial grid by assuming a constitutive behavior and a kinematic model (see, e.g., Portela et al., 2018; Glaesener et al., 2021 for models of 3D truss-based architected materials).

As identifying the initial node positions automatically may be challenging for asymmetric, non-periodic, and irregular truss topologies, we exploit a-priori knowledge about the location of the nodes in the (ideal) undeformed truss, which can be imported, e.g., directly from the CAD model used for 3D-printing the truss. Alternatively, if the actual coordinates of the points of interest are not available but the truss topology is known, one may calculate the artificial grid coordinates based on the truss topology information. (In our code, four types of artificial grids – octahedron, bicructured octahedron (BiOct), octet, and a spatially variant BiOct – are available, while other grids can be created following the method described below.) The example in

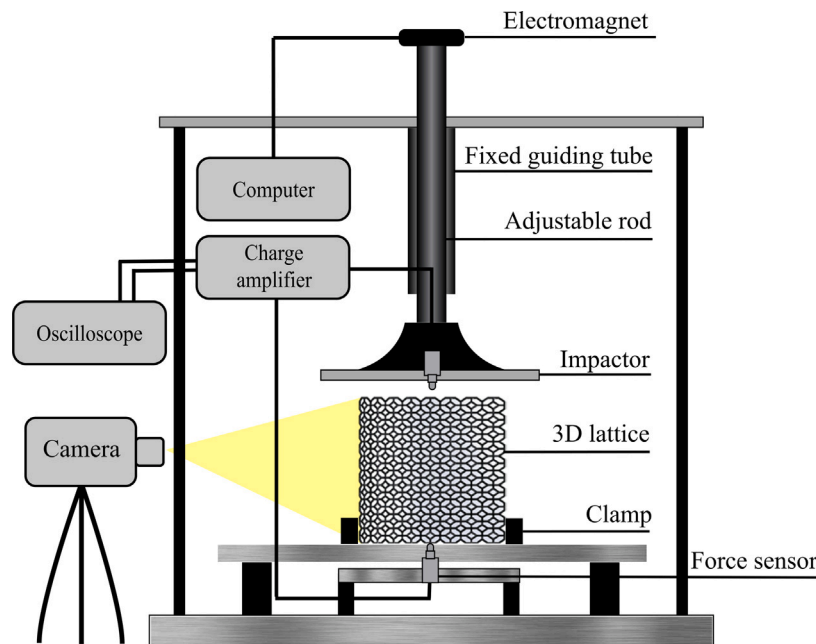


Figure 1. Experimental setup including the in-house designed drop tower, the force sensors, and the high-speed camera.

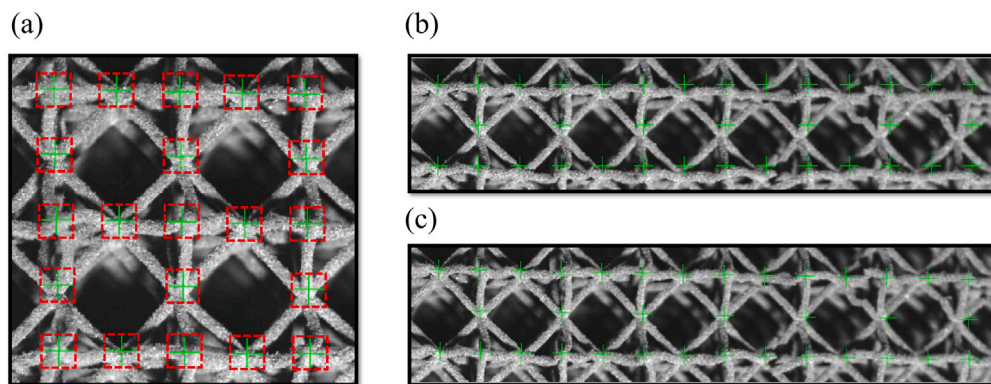


Figure 2. Example of node distribution in a subsection of an octahedron truss lattice. (a) Array assembled for a small number of unit cells generated by the function described in Section 2.2, showing the windows (red dashed lines) of each of the tracked nodes (green crosses). Fractions of the top right corner of the impacted truss, showing (b) the original distribution when using only two points to define the interest region in the image versus (c) the improved version of the grid locations (shown as green markers) after choosing four points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 2a shows an artificial grid of 21 points within four unit cells of an octahedral truss.

Creating the artificial grid based on ideal (as-designed) node positions miss that 3D-printed samples generally suffer from imperfections and defects, which lead to deviations in the actual node locations. Therefore, we apply a correction scheme and rescale the coordinates of the artificial grid according to an image taken from the as-built, undeformed truss. To this end, we select a rectangular area of interest that contains the artificial grid and retrieve the pixel coordinates of all four corners (point  $A$  in the top left corner,  $B$  in the top right corner,  $C$  in the bottom left corner, and  $D$  in the bottom right corner of Fig. 3). Based on the pixel information of those four corner points, we apply an affine interpolation to all points in the artificial grid to properly identify the pixel information of each and every point in the artificial grid, closely matching their positions in the reference image. Details of the interpolation are summarized in Appendix A. Full information of all four corner points is important, as a simple linear rescaling in the horizontal and vertical directions (Fig. 2c) does not match the as-built node positions as well as the full interpolation (Fig. 2b). Fig. 3 shows the resulting artificial grids on a bitruncated octahedron sample with two different node densities.

Once the scaling has been applied and all initial node positions have been identified, the artificial grid is described by an array  $P$  (see Appendix A) containing the initial, undeformed 2D coordinates of each and every point to be tracked. (Coordinates can be defined either as pixel information or as physical coordinates, both of which can uniquely be converted based on the chosen area of interest). Since the artificial grid is defined by the user, it can be adapted to areas of interest (not necessarily requiring the entire truss to be tracked) as well as to varying node densities (e.g., tracking only the strut junctions or also points along each strut) without a noticeable impact on the tracking accuracy, as explained below. Note that the node positions in the artificial grid are computed, following the above procedure, only once before the start of an experiment and based on a static reference image. Once this step has been completed, all nodes of the artificial grid are tracked from frame to frame, as outlined in Section 2.3.

### 2.3. Tracking algorithm

A challenge in finding the displacement vector of any point from one frame to the next in a sequence of images (taken, e.g., at different

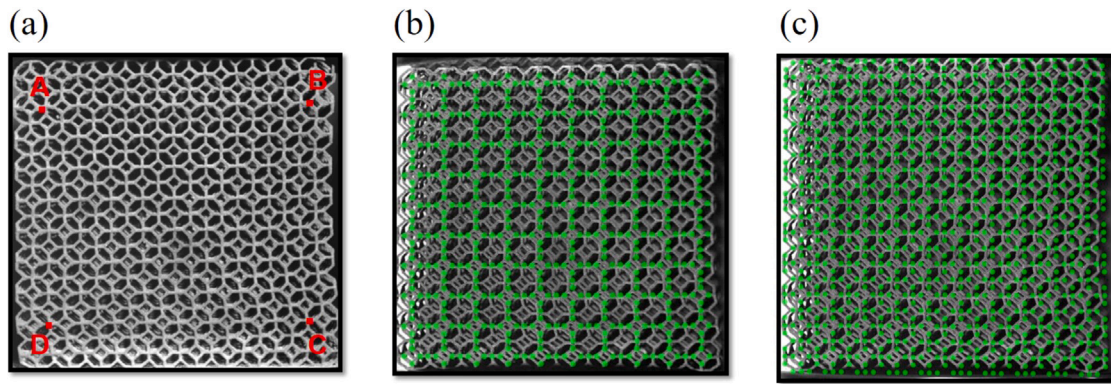


Figure 3. An example of the grid generation in a Bitruncated Octahedron. (a) Reference points selections (b, c) and the resulting point distributions with two different densities.

load or time steps) is that a single pixel cannot be tracked unless it has a distinctive brightness with respect to all of its neighbors. In practice, the pixel value can easily change due to noise and be confused with adjacent pixels. As a consequence, it is hard, if not impossible, to determine where a pixel went from one frame to the next, based only on local information. Therefore, we do not track single pixels but *windows* of pixels, and we aim for windows that contain sufficient texture—one window for each point to be tracked (see Fig. 2a). This effectively breaks the overall tracking problem down into many independent DIC problems, which track the deformation of each window and thereby determine the displacements of the corresponding nodes (such as the truss junction or chosen points along each strut). This implies that we define a window for each node to be tracked.

The displacements experienced by different points as well as their pixel intensity may vary quite distinctly within a window. For example, the surface of a 3D truss may be slanted (and the surface of a strut with a general cross-section is generally not flat), so its intensity pattern can become warped from one frame to the next. Or the window may be along a blocking boundary so that points move at significantly different velocities or even disappear and/or re-appear. This gives rise to two questions. First, how do we know that we are following the same window, if its content changes over time? Second, if we measure the displacements in a window at discrete times, how do we accurately translate those into velocities? The solution to the first problem will be to monitor the residue  $E$ , a metric that checks if the appearance of a window has changed significantly. If it has, the window is to be discarded. The second issue can be solved by modeling the changes in the window as complex transformations rather than simple translations. In this way, different velocities can be attributed to different points of the window (Lucas, 1981). However, due to the high number of parameters to estimate, the tracked window must be large. Therefore, Tomasi and Takeo (1991) suggested a different approach, which estimates the (approximately uniform) displacement vector for sufficiently small windows—considering that any discrepancy between successive windows that cannot be explained by an affine translation is considered to be error. Here, we adopt this approach to track the displacements of all points in the artificial grid.

Let  $F_n(x)$  denote the function that defines for the  $n$ th frame the pixel value at location  $x$  within a given window  $\mathcal{W}$  (Lucas, 1981). Furthermore, let  $h(x)$  be the 2D displacement vector<sup>1</sup> of the point located at  $x$ . In going from frame  $n$  to frame  $n + 1$ , we define the residue as

$$E_{n \rightarrow n+1} = \sum_{x \in \mathcal{W}} [F_n(x + h(x)) - F_{n+1}(x)]^2, \quad (1)$$

<sup>1</sup> Here and in the following, we adopt the common notation in image processing (Tomasi and Takeo, 1991), denoting the displacement field by  $h$ , whereas mechanicians would rather refer to  $u$ .

which is summed over all pixel locations  $x$  within window  $\mathcal{W}$  (assuming that the windows contained within  $F_n$  and  $F_{n+1}$  contain the same numbers of pixels). If the displacements  $h(x)$  are identified correctly, then this residue vanishes. Otherwise, its value is a helpful error metric to indicate the quality of an identified displacement field.

If we assume that the window is sufficiently small to undergo approximately rigid-body motion, then  $h(x) = h$  is constant for all points in the window. To minimize  $E$  with respect to  $h$ , we insert the linear expansion  $F(x + h) \approx F(x) + hF'(x)$  and arrive at

$$\frac{\partial E_{n \rightarrow n+1}}{\partial h} \approx \sum_{x \in \mathcal{W}} 2F'_n(x)[F_n(x) + hF'_n(x) - F_{n+1}(x)] = 0, \quad (2)$$

which can be solved for  $h$ . As this only accounts for rigid-body translation, we expand the description to account for rigid-body rotations and brightness fluctuations within the window. To this end, the residue is re-defined as

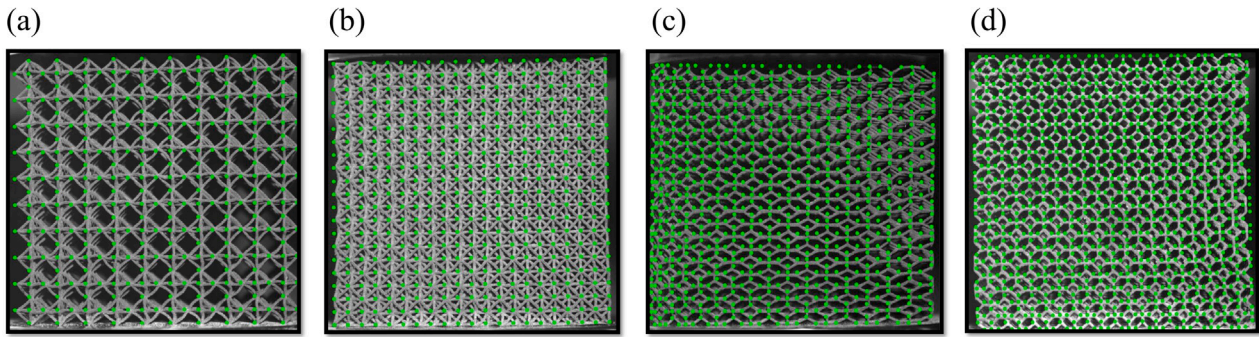
$$E_{n \rightarrow n+1} = \sum_{x \in \mathcal{W}} [F_n(A(x) + h) - (\alpha F_{n+1}(x) + \beta)]^2, \quad (3)$$

where  $\alpha$  and  $\beta$  are scalar contrast and brightness adjustment variables, while  $A(x)$  is a linear spatial transform that allows us to rotate, scale, or shear to find a minimum. The displacement vector  $h_{n \rightarrow n+1}$  of this window between frames  $n$  and  $n + 1$ , along with the transformation for this step, are found from

$$E_{n \rightarrow n+1} \rightarrow \min_{h, \alpha, \beta, A(x)}. \quad (4)$$

This is repeated for every node in the artificial grid  $\mathcal{P}$ , for each of which a suitable window  $\mathcal{W}$  must be defined.

Each window is chosen as the neighborhood around a tracked point, centered at the tracked point and having a size specified as a 2D vector (height, width). The window size must be chosen wisely and can be adjusted for accuracy. Starting from a single pixel, the tracking generally becomes more precise with growing window size, yet this also incurs growing computational expenses. In addition, windows cannot be too large, as this would break the assumption of an approximately constant displacement across the window. Our experience shows that choosing a size of only a few pixels fails, while a window size that is comparable to the width of a strut (and hence roughly to the width and height of a strut junction) is an ideal compromise between accuracy and efficiency. Larger window sizes do not significantly affect the accuracy but can incur considerable computation times. In our examples (see Fig. 2a), we choose a window size of  $31 \times 31$  pixels, which proved sufficiently large to contain the desired feature albeit not too large to result in a residual error greater than 1 pixel (the strut diameter of the 3D-printed samples in our experiments measured approximately 30 pixels). Note that the implementation of this method uses the image pyramids scheme (Tomasi and Takeo, 1991), which implies that the



**Figure 4.** The four tested truss samples with their artificial grids of tracked points. The periodic truss topologies are those of the (a) octahedron, (b) octet, and (c) bitruncated octahedron (BiOct). (d) The fourth lattice is a spatially-variant bitruncated octahedron (SV-BiOct), whose (undeformed) units are uniformly stretched from left to right.

resolution is reduced by a factor of 2 for each level of the pyramid. Starting at the lowest level, this increases the chances of detecting larger displacements.

When minimizing the residue  $E$  for a given tracked point in a frame, we consider not only the original window but also all shifted windows, obtained by horizontally and vertically shifting the window while keeping its size constant. As in the original DIC, we choose the window which minimizes the residue  $E$ . Especially when exposed to large deformation, the neighborhoods of the tracked points can change significantly (see Fig. 5), which makes the aforementioned update of windows necessary. In addition, sufficient lighting is required for the tracking to be accurate (which is a general DIC necessity).

#### 2.4. Node tracking code

The theory described above has been implemented in Matlab (Matlab, 2021). Our point tracking code contains four scripts: (A) A main script, which provides the user interface to input the most relevant information (e.g., the location of the input images or video, the type of truss, number of units cells, number of skipped frames) to be passed to the tracking scripts B (for images) and C (for videos as source data). These in turn call script D to generate a normalized distribution of relative positions of all points within a truss to be tracked. The tracking algorithm records the coordinates of all points in the artificial grid for all frames and saves those into respective output files. Using this output, the nodal displacements are extracted by subtracting the initial coordinates from all subsequent coordinates.

### 3. Results: application to truss-based metamaterials

Using the experimental setup in Fig. 1, we tested four different truss samples, which include both periodic and spatially-variant (SV) trusses (all are shown in Fig. 4). Each sample was tested five times at five different impact speeds  $v_i = \{1, 10, 50, 500, 1000\}$  mm/s. The algorithm described above was applied to the images taken during all conducted tests. As the node tracking resulted in similar results for all samples, we here discuss the case of the octahedron lattice as a representative example in more detail. Examples of the tracking videos of the octahedron and the spatially variant bitruncated octahedron – showing the captured images with the tracked points overlaid – are enclosed as Supplementary Videos S1 through S4.

Fig. 5 illustrates how all tracked points move, as the truss is deforming under compression—both by hand at relatively low speeds and by the impact setup of Fig. 1 at a high compression rate. At a low rate (Fig. 5a), tracking of all points succeeds even when the struts come into contact and the lattice is highly deformed. However, when increasing the impact velocity (Fig. 5b) while keeping the same frame rate, some of the points are no longer tracked and are estimated out of the region of interest (visibly outside the lattice). Since the algorithm is DIC-based, it only scans for matching sections to correlate a set of points in a given

vicinity of the reference location, which means that, if the difference in location of a point/node from one frame to another becomes too high, the point cannot be correlated and thus drops out of the interest region. This issue can be solved by increasing the frame rate during acquisition, as seen in Fig. 5c, where the frame rate has been increased from 2000 (in Fig. 5b) to 4000 frames per second.

To study the effect of the strain rate on the number of non-tracked points (i.e., points dropped out of the region of interest), we tested the four lattices shown in Fig. 4 under four impact velocities, while recording images at the four frame rates of  $\{1000, 2000, 4000, 6000\}$  fps. Fig. 7 compares the number of dropped points during the node tracking process as a function of impact velocity and camera frame rate. As may be expected, the percentage of dropped points increases with increasing impact velocity; however, this percentage drops considerably when increasing the image frame rate. Even when impacting the lattices at high velocities (up to 1000 mm/s) and a relatively low frame rate (e.g., 1000 fps), the number of dropped points is still low and does not surpass 5.5% of all points to be tracked.

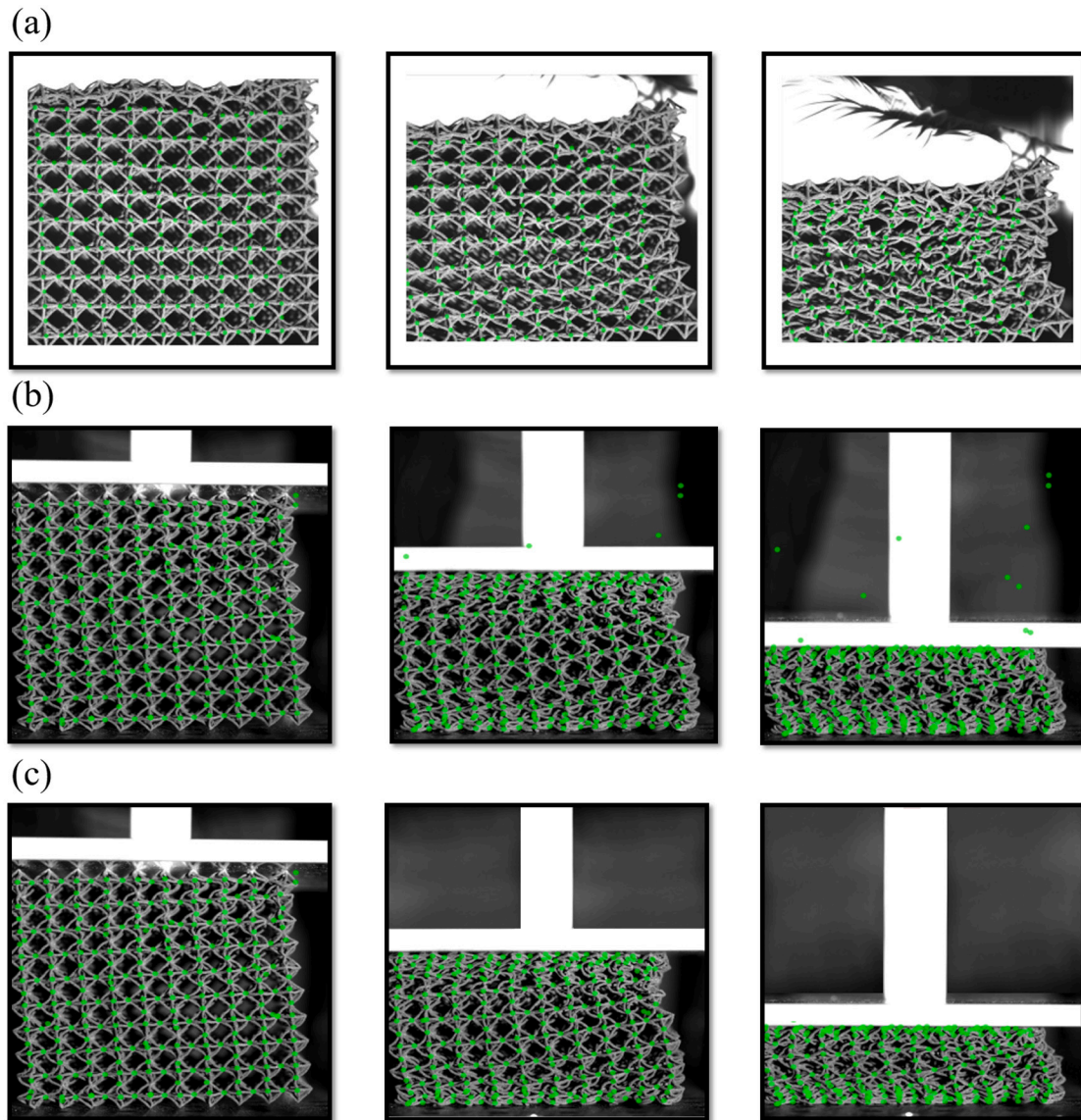
As an example of the tracked nodal displacements, Fig. 6 shows the vertical displacement component (parallel to the impact direction) of three points in the octahedron truss chosen in the top, bottom, and central regions. Fig. 6 shows that the maximum variation between the different trackings (represented by the error bars) is less than 3 mm throughout, which confirms the excellent reproducibility of results.

To quantitatively assess the repeatability of the tracking results, we repeated the process of selecting the outer four reference nodes A through D six times (each time choosing different reference nodes) for each sample and calculated the average error

$$x_{\text{err,avg}} = \frac{\sum_{i=1}^n \sum_{j=1}^{n_{\text{points}}} \sum_{k=1}^{n_{\text{frames}}} |x_0(j, k) - x_i(j, k)|}{n_{\text{points}} n_{\text{frames}} n}, \quad (5)$$

where  $x_i(j, k)$  denotes the position of point  $j$  in the  $k$ th frame during the  $i$ th sequence of measurement.  $n$  is the total number of measurement sequences (excluding the first one, whose positions  $x_0(j, k)$  are chosen as a reference). Matrix  $x$  is structured such that each row contains the displacement data of one tracked point, with the columns corresponding to the subsequent frames analyzed. The results are in pixels and the corresponding values in millimeters are retrieved via calibration. Over the course of  $n = 6$  measurements, the calculated average position error  $x_{\text{err,avg}}$  was less than one pixel per node in both directions, which is translated into the average position errors in millimeter shown in Table 1. Note that there is a difference in the  $x$ - and  $y$ -directions, which depends on the truss geometry and is attributed to the uncertainty in manually choosing the reference points, which may or may not fall onto a node in the various runs.

As an error metric, we introduce the maximum bidirectional error (and define a maximally allowable bidirectional error in our code). The bidirectional error is computed by tracking a feature from one frame to



**Figure 5.** Tracking of the nodes in the octahedron lattice. (a) Manually compressed lattice with an instantaneous loading velocity  $v = 10$  mm/s and a frame rate of 2000 fps. (b, c) Lattices subjected to impact loading with the same impact velocity  $v_i = 1000$  mm/s but two different frame rates: (b) 2000 fps, (c) 4000 fps. (See Supplementary Videos S1 and S2 for the full tracking history).

**Table 1**

Average position errors (as defined in Eq. (5)) in the  $x$ - and  $y$ -directions in units of pixels and millimeters for each of the four studied lattices, determined from six independent measurement rounds and a total of  $n_{\text{points}} = \{290, 440, 1060, 1060\}$  (for the octahedron, octet, bitruncated octahedron, and the spatially graded bitruncated octahedron, respectively) points over  $n_{\text{frames}} = 480$  frames.

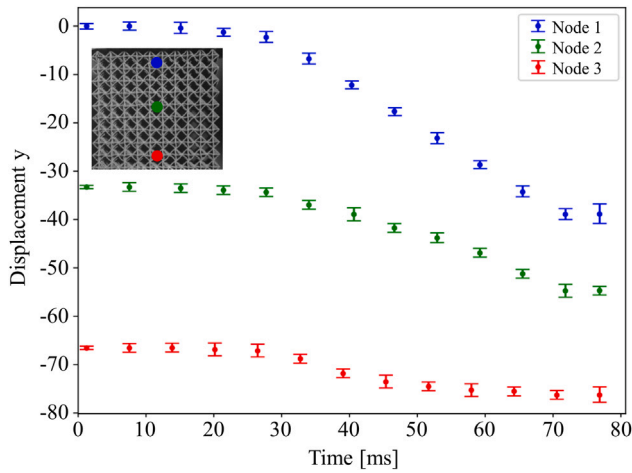
Error	Octahedron	Octet	BiOct	BiOct SV
$x_{\text{err,avg}}$ [px]	0.3245	0.2288	0.3628	0.5231
$x_{\text{err,avg}}$ [mm]	0.0377	0.0266	0.0422	0.0714
$y_{\text{err,avg}}$ [px]	0.3529	0.30424	0.2664	0.4755
$y_{\text{err,avg}}$ [mm]	0.0410	0.0354	0.0310	0.0681

the next through a sequence of images, and then tracking it back to the initial frame to assess where this feature lands in the initial frame. The resulting coordinates are compared to the actual initial coordinates, and the discrepancy is reported in values of pixels. For all results shown in this work, we chose an allowable maximum of 1 pixel, which was maintained throughout all experiments reported here—meaning that all points that were tracked until the end of the sequence have sub-pixel

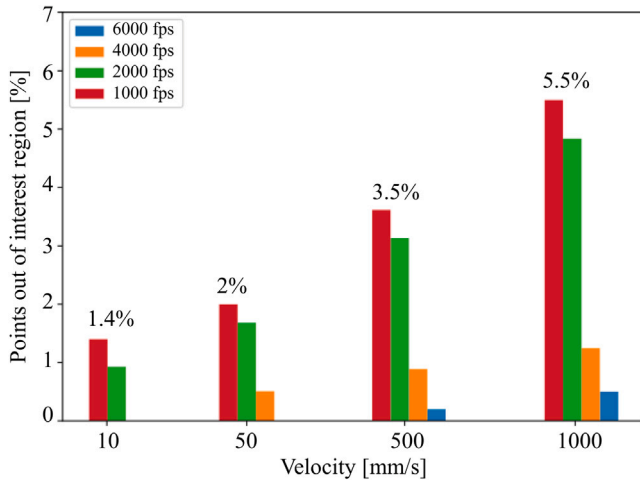
precision between individual frames. (This parameter is adjustable in our code along with other parameters, such as the size of the integration window, the number of pyramid levels and the number of iterations per frame). The low maximum bidirectional error is evidence of the accuracy of the tracking scheme (in addition to the aforementioned proof of reproducibility of results for different reference nodes). A quantitative validation by comparison with an alternative technique would be ideal to report but is challenging due to a lack of alternatives for the experimental conditions of large deformation, high rates, and low-density cellular structures (cf. the challenges of conventional DIC and related techniques, as discussed in Section 1). A further visual confirmation of the accuracy of the method is given by the comparison of recorded videos and tracked nodes in the Supplementary Videos.

#### 4. Conclusions

We have presented a versatile point-tracking scheme specifically tailored to track the displacements on the surface of complex truss lattices undergoing large deformation. For an artificial grid of points of interest (such as strut junctions or other characteristic points), we



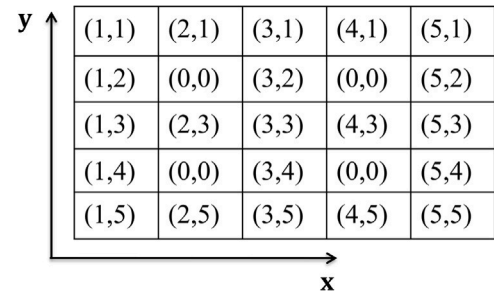
**Figure 6.** Tracked vertical displacements of three points in the octahedron truss of Fig. 5c during impact. The three nodes were chosen near the top (blue marker), in the center (green), and near the bottom of the sample (red). Error bars represent the standard deviations computed from the six experiments conducted. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**Figure 7.** Number of points/nodes dropping out of the interest region of the lattices during tracking. The maximum value of the dropped points related to each velocity is displayed on the top of the frame rate 1000 fps.

track each point by defining a window for each point and applying DIC-based techniques to identify the displacement vector from frame to frame. We demonstrated the successful application of this method to a variety of periodic and spatially graded trusses undergoing quasistatic to high-rate impact, which reliably and repeatably characterizes the displacements of (almost all) points in the artificial grid—even when the truss undergoes severe deformation upon compaction. We showed that the choice of the imaging frame rate is crucial to obtaining optimal results. In addition, we evaluated the measurement error (in terms of the maximum bidirectional variation of pixels across all frames), which was proven to be less than 1 pixel across all tested trusses.

The same framework has been used successfully to track small displacements on lower-density truss lattices (see the Supplementary Videos), and it can even be used as an alternative to classical DIC to compute strain fields of general samples (not necessarily trusses or cellular solids); our approach makes the process simpler, since it does not require a speckle pattern. We note that the developed tool at present only tracks 2D displacements. However, by adding extra cameras during experiments and in a postprocessing step computing the



**Figure A.8.** Array assembled for the small number of unit cells shown in Fig. 2, generated by the function described in Section 2.2.

depth map using stereo vision, the same tools can be used to extract 3D displacements of the tracked nodes.

The algorithms used in this study are freely available from the link provided in the data availability statement.

**Declaration of competing interest**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Data availability**

The tracking algorithms along with video examples used in this study are available as MATLAB code from this link [https://gitlab.com/kaoutar\\_radi/nodes-displacement-tracking](https://gitlab.com/kaoutar_radi/nodes-displacement-tracking).

**Appendix A. Artificial grid rescaling**

Each point of interest in the artificial grid is defined by its (undeformed, ideal)  $x$ - and  $y$ -coordinates, which leads to an array of the nodal positions of points  $P_i$  with  $i = 1, \dots, n$  and  $n$  denoting the total number of points to be tracked:

$$P = \begin{bmatrix} P_{1,x} & P_{1,y} \\ P_{2,x} & P_{2,y} \\ \vdots & \vdots \\ P_{n,x} & P_{n,y} \end{bmatrix} \tag{A.1}$$

These  $x$ - and  $y$ -coordinates can be in arbitrary units, e.g., in units of the unit cell strut length. For example, for the artificial grid shown in Fig. 2a the array is shown in A.8.

The coordinates are normalized by the total number of grid points in the horizontal and vertical directions, respectively, which results in a set with components within  $[0, 1]$ :

$$P_{i,x} \leftarrow \frac{P_{i,x}}{\max P_x}, \quad P_{i,y} \leftarrow \frac{P_{i,y}}{\max P_y}, \tag{A.2}$$

where vectors  $P_x = \{P_{1,x}, \dots, P_{n,x}\}$  and  $P_y = \{P_{1,y}, \dots, P_{n,y}\}$  denote, respectively, the first and second column of the original array in Eq. (A.1). For convenience, we re-scale the normalized coordinates such that the reference point A (Fig. 3a) is located at  $(0, 0)$  in the undeformed configuration and such that the artificial grid is affinely transformed to best match the actual locations of the respective points of interest in the as-printed sample. This is accomplished by rescaling according to

$$P_{i,x} \leftarrow P_{i,x}(C_x - D_x)M_{i,y} + N_{i,x} \tag{A.3}$$

and

$$P_{i,y} \leftarrow P_{i,y}(A_y - D_y)M_{i,x} + N_{i,y} \tag{A.4}$$

for  $i = 1, \dots, n$  with

$$M_{i,y} = 1 - \frac{P_{i,y}}{\max P_y}, \quad N_{i,y} = \frac{(B_x - C_x)P_{i,y}}{\max P_y}. \quad (\text{A.5})$$

$M_{i,x}$  and  $N_{i,x}$  are defined analogously. Finally, we re-scale according to

$$P_{i,x} \leftarrow P_{i,x} + \frac{(A_x - D_x)P_{i,y}}{\max P_y} \quad (\text{A.6})$$

and

$$P_{i,y} \leftarrow P_{i,y} + \frac{(C_y - D_y)P_{i,x}}{\max P_x} \quad (\text{A.7})$$

This results in an array  $P$  of the normalized relative positions of all points to be tracked on the surface of a truss sample. Note that this approach is equally applicable to periodic and non-periodic trusses, as will be demonstrated in our examples in Section 3.

## Appendix B. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.mechmat.2023.104658>.

## References

- Abdelbarr, M., Chen, Y.L., Jahanshahi, M.R., Masri, S.F., Shen, W.M., Qidwai, U.A., 2017. *Smart Mater. Struct.* 26 (12).
- Babae, S., Shim, J., Weaver, J.C., Chen, E.R., Patel, N., Bertoldi, K., 2013. *Adv. Mater.* 25 (36), 5044–5049.
- Bai, P., Zhu, F., He, X., 2015. *Opt. Laser Technol.* 73, 29–38.
- Bhaduri, B., Tay, C.J., Quan, C., Niu, H., Sjö Dahl, M., 2011. *Opt. Commun.* 284 (10–11), 2437–2440.
- Bückmann, T., Kadic, M., Schittny, R., Wegener, M., 2015. *Proc. Natl. Acad. Sci. USA* 112 (16), 4930–4934.
- Chen, B., Basaran, C., 2008. *Exp. Mech.* 48 (5), 665–673.
- Chen, F., Luo, W.D., Dale, M., Petniunas, A., Harwood, P., Brown, G.M., 2003. *Opt. Lasers Eng.* 40 (5–6), 459–485.
- Dell'Isola, F., Seppecher, P., Spagnuolo, M., Barchiesi, E., Hild, F., Lekszycki, T., Giorgio, I., Placidi, L., Andraus, U., Cuomo, M., Eugster, S.R., Pfaff, A., Hoschke, K., Langkemper, R., Turco, E., Sarikaya, R., Misra, A., De Angelo, M., D'Annibale, F., Bouterf, A., Pinelli, X., Misra, A., Desmorat, B., Pawlikowski, M., Dupuy, C., Scerrato, D., Peyre, P., Laudato, M., Manzari, L., Göransson, P., Hesch, C., Hesch, S., Franciosi, P., Dirrenberger, J., Maurin, F., Vangelatos, Z., Grigoropoulos, C., Melissinaki, V., Farsari, M., Muller, W., Abali, B.E., Liebold, C., Ganzosch, G., Harrison, P., Drobnicki, R., Igumnov, L., Alzahrani, F., Hayat, T., 2019. *Contin. Mech. Thermodyn.* 31 (4), 1231–1282.
- Draney, M.T., Herfkens, R.J., Hughes, T.J., Pelc, N.J., Wedding, K.L., Zarins, C.K., Taylor, C.A., 2002. *Ann. Biomed. Eng.* 30 (8), 1033–1045.
- Glaesener, R.N., Bastek, J.H., Gonon, F., Kannan, V., Telgen, B., Spöttling, B., Steiner, S., Kochmann, D.M., 2021. *J. Mech. Phys. Solids* 156 (June), 104569.
- Grima, J.N., Attard, D., Gatt, R., 2008. *Phys Status Solidi (B) Basic Res* 245 (11), 2405–2414.
- Lucas, B.D., 1981. pp. 121–130.
- Maraghechi, S., Hoefnagels, J.P., Peerlings, R.H., Rokoš, O., Geers, M.G., 2020. Experimental full-field analysis of size effects in miniaturized cellular elastomeric metamaterials. *Mater. Des.* 193.
- Matlab, 2021. *Matlab Documentation*. vision.pointtracker.
- Meza, L.R., Das, S., Greer, J.R., 2014a. *Science* 345 (6202), 1322–1326.
- Meza, L.R., Das, S., Greer, J.R., 2014b. *Science* 345 (6202), 1322–1326.
- Meza, L.R., Greer, J.R., 2014. *J. Mater. Sci.* 49 (6), 2496–2508.
- Mohsenizadeh, M., Gasbarri, F., Munther, M., Beheshti, A., Davami, K., 2018. *Mater. Des.* 139, 521–530.
- Niu, Y., Shao, S., Park, S.B., Kao, C.L., 2017. *IEEE Trans. Compon. Packag. Manuf. Technol.* 7 (2), 276–284.
- Portela, C.M., Greer, J.R., Kochmann, D.M., 2018. *Extreme Mech. Lett.* 22, 138–148.
- Rouwane, A., Bouclier, R., Passieux, J.C., Périé, J.N., 2022. *Int. J. Solids Struct.* 234–235.
- Roux, S., Hild, F., Viot, P., Bernard, D., 2008. *Composites A* 39 (8), 1253–1265.
- Schaeffer, M., Trainiti, G., Ruzzene, M., 2017. *Sci. Rep.* 7 (February), 1–9.
- Sun, J., Litchinitser, N.M., 2016. *Fundam. Appl. Nanophotonics* 253–307.
- Tomasi, C., Takeo, K., 1991. *Tech. Rep.* 37 (1), 165–168.
- Yasuda, H., Yang, J., 2015. *Phys. Rev. Lett.* 114 (18), 1–5.
- Yuan, B., Sun, M., Wang, Y., Zhai, L., Luo, Q., Zhang, X., 2019. *Int. J. Geomech.* 19 (5), 1–8.