Noname manuscript No.
(will be inserted by the editor)

Inextensible Surface Reconstruction Under Small Relative Deformations from Distributed Angle Measurements

Thibaud Talon · Sergio Pellegrino

Received: date / Accepted: date

Abstract A mathematical model to measure the shape of a 3D surface using angle measurements from embedded sensors is presented. The surface is known in a reference configuration and is assumed to have deformed inextensibly to its current shape. An inextensibility condition is enforced through a discretization of the metric tensor generating a finite number of constraints. This model allows to parameterize the shape of the surface using a small number of unknowns which leads to a small number of sensors. We study the singularities of the equations and derive necessary conditions for the problem to be well-posed as well as limitations of the algorithm. Simulations and experiments are performed on developable surfaces under relatively small deformation to analyze the performance of the method and to show the influence of the parameters used in our algorithm. Overall, the proposed method outperforms the current state-of-the-art by almost an order of magnitude.

Keywords Inextensible · Surface · Reconstruction · Distributed Sensors · Angle Measurements · Metric tensor

Thibaud Talon
thibaud.talon@gmail.com
Currently at OffWorld, Inc

Sergio Pellegrino
sergiop@caltech.edu
Joyce and Kent Kresa Professor of Aerospace and Civil Engineering; Jet Propulsion Laboratory Senior Research Scientist; Co-Director, Space-Based Solar Power Project
Graduate Aerospace Laboratories, 1200 East California Boulevard, Mail Code 105-50

1 Introduction

Recovering the shape of a 3D surface often requires a measurement system with a certain depth of view. Cameras, scanning lasers or radars are often used to generate a point cloud of the surface. Methods exist to convert this cloud into a more or less smooth 3D surface [1,2].

Many solutions focus on reconstructing the shape of a surface from a set of measurements and a reference configuration (also called a template). The 3D shape of the reference is known either from a previous measurement or from the method of construction of the structure. For instance, the surface depicted in figure 1 is known to be a sheet of paper and its dimensions are dictated by a predefined printed pattern. Monocular reconstruction of a surface is a well-known method [3,4,5,6,7,8,9,10]. For instance, Shape-from-Template (SfT) which preserves geodesic distances is widely used to reconstruct shapes by matching features in a picture of the surface and a template while allowing isometric deformation of the structure from the template. Other methods also integrate the measurement of the direction of the normal to the surface implicitly from images [8,9,10]. Such methods include Shape-from-Shading (SfS) that uses the reflection of light from the surface.

There are inherent limitations to these methods. For instance, the camera has to be held in front of the surface at a distance sufficient to accommodate the field of view or range of the system. Rigs of cameras can extend the depth of view by having each camera look at a different part of the structure and hence closer to it but add complexity and still require a minimum distance from the surface. There may be insufficient space available in front of the structure, or holding the

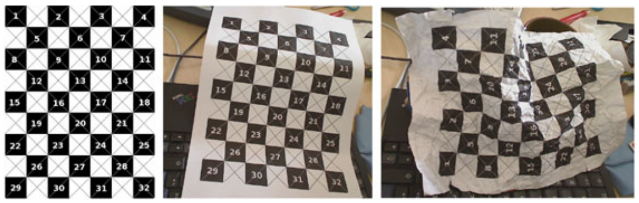


Fig. 1: Example of images used for monocular, template-based reconstruction of inextensible surface [4]. The left image shows a flat sheet of paper (reference configuration) and the other images show bent and crumpled configurations to be reconstructed.

camera may be very difficult, e.g. in the case of large space structures or rapidly moving objects. In the former case, while it is possible to attach a camera to a deployable boom or have a second spacecraft follow the primary structure, such solutions add mass and complexity to the system. Finally, camera-based methods such as Shape-from-Template require a pattern to be drawn on the surface which may be undesirable in some situations. For instance, a flexible solar array is covered with photovoltaic cells and drawing a pattern on top would decrease the efficiency of the array.

The approach investigated in this paper is based on embedding angle sensors directly on the surface to be reconstructed.

Different technologies exist to perform such measurements. Most state-of-the-art technologies use inertial sensors (a combination of accelerometers and magnetometers) to measure angles from the gravity vector and Earth's magnetic North and have been investigated to reconstruct both 3D curves and surfaces [11,12,13, 14,15,16]. Note that if the surface experiences large accelerations, the accelerometers fail to detect the direction of gravity. Magnetic fields are easily affected by magnets, currents, or ferrous materials, which limits the usage of magnetometers.

Sun sensors have recently been studied mostly for space applications where gravity is near-zero [17,18,19]. Different sensor technologies exist. The simplest ones use quad-photodiodes behind an aperture, effectively acting as a 4-pixel pinhole camera [19]. More complex architectures involve cameras with a large photosensor array that locate the centroid of the spot created by the light source using image-processing algorithms [20]. They can also identify features in the image (such as stars) to improve their accuracy.

The angle measurements of these sensors are fed in an algorithm that will be described in this paper, and the algorithm reconstructs the shape of an inextensible support surface that holds the sensors.

This study is limited to inextensible deformations, also called isometric deformations as the surface cannot stretch or contract. The term *inextensible* will be used in this paper, as it is widely used in the field of structural mechanics. This requires a reference configuration, also called template in the literature, to be fully known in order to define the conservation of lengths upon deformation.

The overall problem is described in figure 2. Imposing inextensibility of the deformation of the surface usually serves two purposes: 1) to eliminate singularities in the algorithms and 2) to improve the results by adding some knowledge of the deformation. Many methods used to reconstruct inextensible surfaces employ a triangular [5,21] or quadrilateral [22] mesh to map the surface. Each edge of the mesh can be defined as a straight, rigid line which implicitly enforces inextensibility. This method requires a fine mesh in order to achieve a smooth mapping; this means that many degrees of freedom must be computed from a large amount of data (for instance, high resolution images) which can be computationally expensive. Tangential and normal vectors are undefined at the intersection of edges which can cause issues in defining angles. Note that different methods exist to reduce the number of variables of a fine mesh [23] with some guaranteeing inextensibility of the deformation [24], eventually preserving details of the template. However, they add complexity to the reconstruction algorithm. Previous research involving embedded sensors does not strongly impose inextensibility of the deformation. The surface is reconstructed by integrating along inextensible lines where the angle sensors are placed uniformly [11,14,15]. The inextensibility is either imposed explicitly, by enforcing conservation of lengths between sensors, or implicitly by connecting sensors by means of rigid lines. The surface is then filled by different techniques such as Coon's methods in [13] or using a quad mesh in [11]. While inextensibility is imposed along the lines of integration, it is usually constrained by the placement of sensors and incomplete as shear is not taken into account.

Our approach is to reconstruct the relatively smooth shape of the surface in its current configuration by only assuming an inextensible transformation from the reference configuration (or template) and the measurement of angles at discrete locations along the structure. Because the number of sensors that can be placed on a structure is limited, constraining the amount of variables to define the shape of the surface, only smooth shapes are considered in this paper.

In order to estimate the shape of a surface, we parameterize it on a set of basis functions. This is presented in section 2. Different sets of basis functions rel-

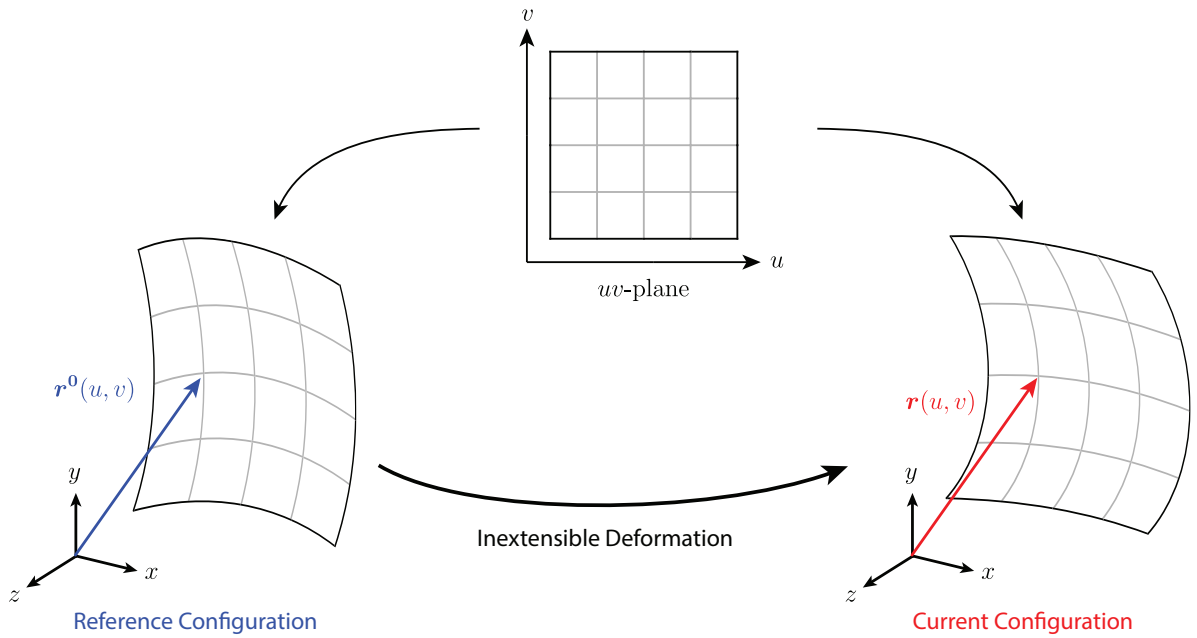


Fig. 2: Definition of the problem. The surface is parametrized by two coordinates (u, v). The 3D surface is a mapping of the 2D coordinates to 3D. The shape of the reference configuration is known while the current configuration needs to be reconstructed.

evant to this problem are introduced. We define the inextensibility of the deformation in section 3 using the conservation of the metric tensor. The general mathematical relations are discretized in order to generate a finite set of constraints. The singularities associated with this first system of equations are analyzed. A set of angle measurements is defined in section 4. Their singularities are analyzed separately from the inextensibility conditions, and we especially investigate requirements on the placement of the sensors. The complete system of equations including inextensibility and angle measurements is obtained in section 5. It is generally an overconstrained system whose least error solution can be found, given an initial estimate, using the Levenberg-Marquardt algorithm. Singularities of this complete system are studied based on the remarks done on the singularities of each set of equations. The approach is tested on simulations of a developable surface with a conical shape, in section 6. A complete study analyzing the overall error of the reconstructed shapes, the inextensibility, the influence of sensor noise and different parameters of the algorithm are investigated. In section 7, the presented method is compared to the current state of the art, which uses Inertial Measurement Units (IMUs) as sensors. It is shown that the presented method outperforms current methods. Finally, section 8 shows the results of an experimental demonstration. Sub-millimeter accuracy was achieved on a 1.2×0.2 m^2 structure under relatively small deformations, as predicted from simulations.

2 Surface Model

2.1 Definition of Parametric Surface

A 3D surface can be described explicitly by the mapping $\boldsymbol{r} : X \subset \mathbb{R}^2 \to \mathbb{R}^3$. Only two *curvilinear* coordinates (u, v) are needed to uniquely define a point on the surfaces as shown in figure 2. The image of this two-coordinate point through the mapping represents the location of that point in 3D space.

For instance, a flat surface (or plane) can be represented by:

$$\boldsymbol{r}(u,v) = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \tag{1}$$

A cylindrical surface of radius R along the z-axis can be represented by:

$$\boldsymbol{r}(u,v) = \begin{bmatrix} R\cos u \\ R\sin u \\ v \end{bmatrix} \tag{2}$$

Table 1: Nomenclature

α_S	Angle around the first tangent vector at the location of sensor S
β_S	Angle around the second tangent vector at the location of sensor S
$\mathbf{M^0}$, $\mathbf{M}$	Metric tensor in reference, and current configurations
$\boldsymbol{n}(u,v)$	Normal of surface parameterized by $\boldsymbol{r}(u,v)$
N_S	Total number of sensors
N_u	Number of control points in the u-direction
N_v	Number of control points in the v-direction
$\hat{N}_u$	Size of $[\hat{u}_i]$
$\hat{N}_v$	Size of $[\hat{v}_j]$
$\boldsymbol{q_k}$	Position of the control point k
$\boldsymbol{r}(u,v)$	Position of the surface in the current configuration
$\boldsymbol{r^0}(u,v)$	Position of the surface in the reference configuration
$\frac{\partial \boldsymbol{r}}{\partial u}(u,v)$	First tangential vector of the surface parameterized by $\boldsymbol{r}(u,v)$
$\frac{\partial \boldsymbol{r}}{\partial v}(u,v)$	Second tangential vector of the surface parameterized by $\boldsymbol{r}(u,v)$
u, v	Curvilinear coordinates of a surface
u_S, v_S	Curvilinear coordinates of sensor S
$[\hat{u}_i]$	Coordinates of inextensibility grid in the u-direction
$[\hat{v}_j]$	Coordinates of inextensibility grid in the v-direction
$\phi_k(u,v)$	Basis function k

2.2 Basis Function Decomposition

We consider a finite dimension mapping defined by basis functions. The mapping $\boldsymbol{r}$ can be written as:

$$\begin{aligned} &\boldsymbol{r} : X \subset \mathbb{R}^2 \longrightarrow \mathbb{R}^3 \\ &\quad (u,v) \quad \longmapsto \boldsymbol{r}(u,v) = \sum_{k=1}^{N} \boldsymbol{q_k} \phi_k(u,v) \end{aligned} \tag{3}$$

where $\phi_k : X \to \mathbb{R}$ are basis functions, $\boldsymbol{q_k}$ are unknown 3D points called control points and define the weight of the basis functions, and N is the dimension of the function space.

This basis representation is common for such problems [4,5,25,26]. Many sets of functions can be used to describe the mapping $\boldsymbol{r}$. Perriollat et al. [4] use Thin-Plate Splines, Metaxas et al. [25] use Finite Element basis functions which are piecewise polynomials defined over local supports. B-Splines are used to fit a surface to data points [27]. Note that B-Splines and the more complicated 2D Non-Uniform Rational Basis Splines (NURBS) are often used in computer-aided design to draw complex surfaces. Other basis functions such as rational Gaussian functions can also be used [28]. They have the advantage of being able to capture both global and local deformations with one set of basis functions

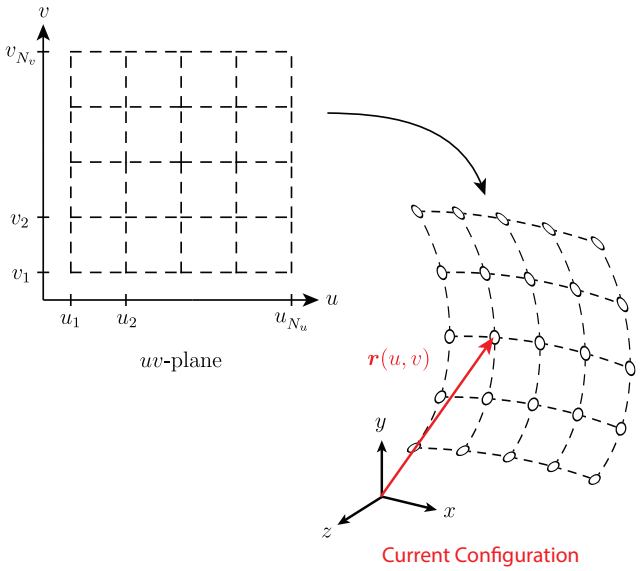


Fig. 3: Parameterized mapping defining the surface to be reconstructed. Circles represent the 3D position of the control points.

by varying the standard deviation of each Gaussian. Simple polynomial series have also been used in [17].

In order to define angles of the surface at specific locations (see Section 4), the basis functions used in our problem need to be differentiable. We limit this study to simple polynomial basis functions: 2D Lagrange polynomials. They are defined over a grid of control points aligned with the curvilinear coordinates (see figure 3). Let N_u and N_v be the size of the grid in each direction and note that $N = N_u \times N_v$. We can rewrite equation (3) as:

$$\boldsymbol{r}(u,v) = \sum_{k=1}^{N_u} \sum_{l=1}^{N_v} \boldsymbol{q_{k,l}} \phi_{k,l}(u,v) \tag{4}$$

Lagrange polynomials are often used to interpolate functions based on known values at discrete locations [29]. They are easy to compute and physically understandable as the control points lie on the surface to be reconstructed. Unfortunately, for a large number of control points, which corresponds to a large polynomial order, they are susceptible to Runge's phenomenon where a function can have large oscillations near the boundaries of the domain [30]. The basis functions are written as:

$$\phi_{k,l}(u,v) = L_k^u(u) L_l^v(v) \tag{5}$$

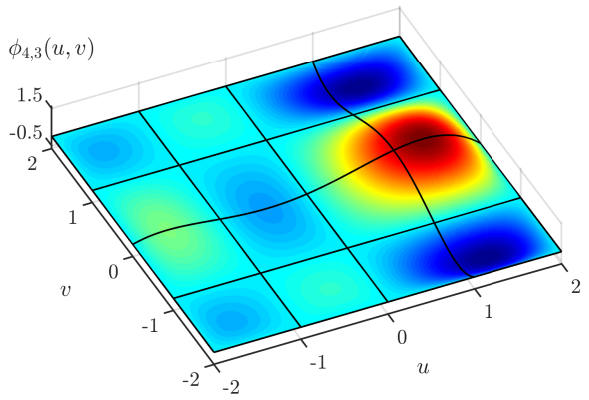


Fig. 4: Lagrange basis function $\phi_{4,3}(u,v) = L_4^u(u)L_3^v(v)$ on a regular grid of interpolation coordinates $[\boldsymbol{u_k}] = [\boldsymbol{v_l}] = (-2,-1,0,1,2)$.

where L_k^u (resp. L_l^v) is the Lagrange polynomial in the u-direction (resp. v-direction):

$$L_k^u(u) = \prod_{\substack{p=1 \\ p\neq k}}^{N_u} \frac{u-u_p}{u_k-u_p} \quad \text{and} \quad L_l^v(v) = \prod_{\substack{p=1 \\ p\neq l}}^{N_v} \frac{v-v_p}{v_l-v_p} \tag{6}$$

where u_k (resp. v_l) are interpolation coordinates defined in the uv-plane. Figure 4 shows the Lagrange basis function $\phi_{4,3}(u,v) = L_4^u(u)L_3^v(v)$ on a uniform grid of interpolation coordinates $[\boldsymbol{u_k}] = [\boldsymbol{v_l}] = (-2,-1,0,1,2)$.

3 Inextensibility Constraints

3.1 Definition of the constraints

Curvilinear distances on a surface in 3D Euclidian space can be calculated using the metric tensor [31,32]. It is the tensor representation of the first fundamental form in differential geometry [33]. For the parametric surface defined in equation (4), the associated metric tensor is:

$$\mathbf{M}(u,v) = \begin{bmatrix} \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial u} & \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial v} \\ \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial v} & \frac{\partial \boldsymbol{r}}{\partial v}\cdot\frac{\partial \boldsymbol{r}}{\partial v} \end{bmatrix} \tag{7}$$

where $\boldsymbol{a}\cdot\boldsymbol{b}$ is the inner product between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

The length of a curve defined in the uv-space by $(u(t),v(t))$, $t\in[t_0,t_1]$ can be calculated as:

$$s = \int_{t_0}^{t_1} \sqrt{\left[u'(t),v'(t)\right]\mathbf{M}(u(t),v(t))\begin{bmatrix}u'(t)\\v'(t)\end{bmatrix}}\,dt \tag{8}$$

As a result, the deformation between two surfaces defined by $\boldsymbol{r^0}$ and $\boldsymbol{r}$ is inextensible if any curve has the same length on both surfaces, or if and only if the metric tensor is conserved upon deformation $\mathbf{M} = \mathbf{M^0}$:

$$\begin{bmatrix} \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial u} & \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial v} \\ \frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial v} & \frac{\partial \boldsymbol{r}}{\partial v}\cdot\frac{\partial \boldsymbol{r}}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{r^0}}{\partial u}\cdot\frac{\partial \boldsymbol{r^0}}{\partial u} & \frac{\partial \boldsymbol{r^0}}{\partial u}\cdot\frac{\partial \boldsymbol{r^0}}{\partial v} \\ \frac{\partial \boldsymbol{r^0}}{\partial u}\cdot\frac{\partial \boldsymbol{r^0}}{\partial v} & \frac{\partial \boldsymbol{r^0}}{\partial v}\cdot\frac{\partial \boldsymbol{r^0}}{\partial v} \end{bmatrix} \tag{9}$$

Note that $\mathbf{M^0}$ is fully known since the reference configuration is known. The inextensibility of the transformation leads to the following 3 equations:

$$\frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial u} = \left\|\frac{\partial \boldsymbol{r}}{\partial u}\right\|^2 = \left\|\frac{\partial \boldsymbol{r^0}}{\partial u}\right\|^2 \tag{10}$$

$$\frac{\partial \boldsymbol{r}}{\partial v}\cdot\frac{\partial \boldsymbol{r}}{\partial v} = \left\|\frac{\partial \boldsymbol{r}}{\partial v}\right\|^2 = \left\|\frac{\partial \boldsymbol{r^0}}{\partial v}\right\|^2 \tag{11}$$

$$\frac{\partial \boldsymbol{r}}{\partial u}\cdot\frac{\partial \boldsymbol{r}}{\partial v} = \frac{\partial \boldsymbol{r^0}}{\partial u}\cdot\frac{\partial \boldsymbol{r^0}}{\partial v} \tag{12}$$

Equations (10) and (11) impose the condition that the square of the local strain in both directions u and v is zero. Equation (12) imposes the condition that the angle between the two tangent directions remains constant.

Equations (10) - (12) form a system of three nonlinear differential equations for three unknowns (the components of $\boldsymbol{r}$). The family of solutions represents all possible inextensible deformations from the initial configuration $\boldsymbol{r^0}$. The complexity of this system makes it impossible to solve analytically. As a result we derive a finite subset of constraints inspired from these PDEs.

We define a regular grid called *inextensibility grid* aligned with the curvilinear coordinates u and v. It is parameterized by the coordinates $[\boldsymbol{\hat{u}_i}] = (\hat{u}_1,...,\hat{u}_i,...,\hat{u}_{\hat{N}_u})$ and $[\boldsymbol{\hat{v}_j}] = (\hat{v}_1,...,\hat{v}_j,...,\hat{v}_{\hat{N}_v})$. This grid is shown in figure 5. It is usually different from the grid used to define the Lagrange polynomials shown in figure 3.

Equations (10) - (12) are discretized into a finite number of constraints on this grid. The strain constraints are applied on average between two nodes of the grid. This is equivalent to conserving the length of each edge of the grid. The dot product constraint is taken at the nodes of the grid. This leads to the following equations:

$$\int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\|\frac{\partial \boldsymbol{r}}{\partial u}(u,\hat{v}_j)\right\| du = \int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\|\frac{\partial \boldsymbol{r^0}}{\partial u}(u,\hat{v}_j)\right\| du \tag{13}$$

$$\int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\|\frac{\partial \boldsymbol{r}}{\partial v}(\hat{u}_i,v)\right\| dv = \int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\|\frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i,v)\right\| dv \tag{14}$$

$$\frac{\partial \boldsymbol{r}}{\partial u}^T(\hat{u}_i,\hat{v}_j)\frac{\partial \boldsymbol{r}}{\partial v}(\hat{u}_i,\hat{v}_j) = \frac{\partial \boldsymbol{r^0}}{\partial u}^T(\hat{u}_i,\hat{v}_j)\frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i,\hat{v}_j) \tag{15}$$

where the right hand-side is known.

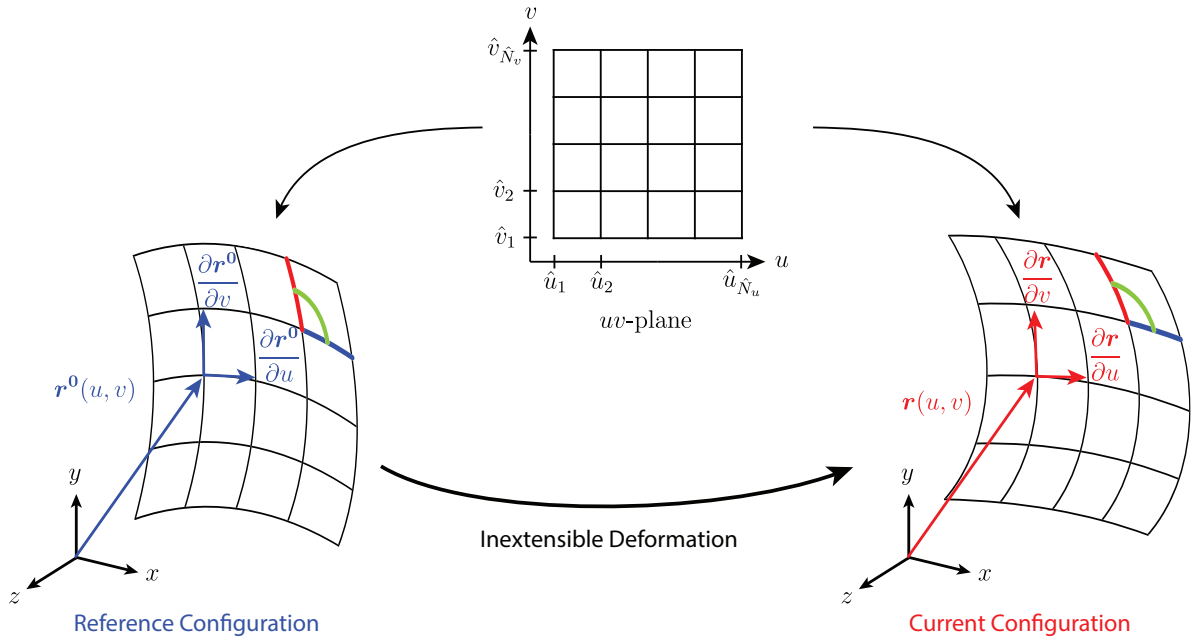


Fig. 5: Inextensibility constraints on the grid defined by $[\hat{\boldsymbol{u}}_{\boldsymbol{i}}]$ and $[\hat{\boldsymbol{v}}_{\boldsymbol{j}}]$. The red and blue segments remain the same length after deformation. The angle around the node defined by the green arc also remains the same.

These constraints can be physically interpreted by considering a 2D structure made of rods and rigid joints. The length constraints represent the inextensibility of the rods while the angle constraints represent the fixed angles between rods imposed by the joints.

A similar approach is used in [3,7,8]. The metric tensor is constrained to remain constant upon deformation at discrete locations on the surface (nodes of a grid). The length constraints in equations (13) and (14) are replaced by simply imposing the norm of the tangent vectors to remain constant. Using lengths in the constraints as opposed to the norm of the tangent vectors makes it easier to define the reference configuration, especially for more complex geometries. Distances are well defined quantities while the norm of the tangent vectors highly depends on the mapping used to define the reference configuration.

3.2 Rank Deficiencies

The inextensibility conditions defined in equations (13) - (15) applied over the whole inextensibility grid define a system of $3\hat{N}_u\hat{N}_v - \hat{N}_u - \hat{N}_v$ equations

$$\boldsymbol{f}(\boldsymbol{q_1}, ..., \boldsymbol{q_N}) = 0 \tag{16}$$

where $\boldsymbol{f}$ is a vector containing all the inextensibility conditions, written as:

$$f_{u,i,j} = \int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\| \sum_{k=1}^{N} \boldsymbol{q_k} \frac{\partial \phi_k}{\partial u}(u, \hat{v}_j) \right\| du - \int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\| \frac{\partial \boldsymbol{r^0}}{\partial u}(u, \hat{v}_j) \right\| du = 0 \tag{17}$$

$$f_{v,i,j} = \int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\| \sum_{k=1}^{N} \boldsymbol{q_k} \frac{\partial \phi_k}{\partial v}(\hat{u}_i, v) \right\| dv - \int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\| \frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i, v) \right\| dv = 0 \tag{18}$$

$$f_{A,i,j} = \sum_{k=1}^{N}\sum_{l=1}^{N} \boldsymbol{q_k}^T \boldsymbol{q_l} \frac{\partial \phi_k}{\partial u}(\hat{u}_i, \hat{v}_j) \frac{\partial \phi_l}{\partial v}(\hat{u}_i, \hat{v}_j) - \frac{\partial \boldsymbol{r^0}}{\partial u}^T(\hat{u}_i, \hat{v}_j) \frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i, \hat{v}_j) = 0 \tag{19}$$

The jacobian of this system of equations is defined as the tensor:

$$\mathbf{J} = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q_1}} \quad ... \quad \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q_N}} \right] \tag{20}$$

Let $(\boldsymbol{q_1^*}, ..., \boldsymbol{q_N^*})$ be a solution of equation (16) and $\boldsymbol{r^*}$ the associated surface. Let $\boldsymbol{q_k} = \boldsymbol{q_k^*} + \boldsymbol{\delta q_k}$ with $\|\boldsymbol{\delta q_k}\| \ll 1$. One can calculate $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q_k}}$ by calculating the

first order term of the Taylor expansion of $\boldsymbol{f}(\boldsymbol{q_1^*}, ..., \boldsymbol{q_k^*} + \boldsymbol{\delta q_k^*}, ..., \boldsymbol{q_N^*})$. This leads to:

$$\frac{\partial f_{u,i,j}}{\partial \boldsymbol{q_k}} = \int_{\hat{u}_i}^{\hat{u}_{i+1}} \frac{\frac{\partial \boldsymbol{r}^*}{\partial u}(u, \hat{v}_j)}{\left\| \frac{\partial \boldsymbol{r}^*}{\partial u}(u, \hat{v}_j) \right\|} \frac{\partial \phi_k}{\partial u}(u, \hat{v}_j) du \tag{21}$$

$$\frac{\partial f_{v,i,j}}{\partial \boldsymbol{q_k}} = \int_{\hat{v}_i}^{\hat{v}_{i+1}} \frac{\frac{\partial \boldsymbol{r}^*}{\partial v}(\hat{u}_i, v)}{\left\| \frac{\partial \boldsymbol{r}^*}{\partial v}(\hat{u}_i, v) \right\|} \frac{\partial \phi_k}{\partial v}(\hat{u}_i, v) dv \tag{22}$$

$$\frac{\partial f_{A,i,j}}{\partial \boldsymbol{q_k}} = \left(\frac{\partial \phi_k}{\partial v} \frac{\partial \boldsymbol{r}^*}{\partial u} + \frac{\partial \phi_k}{\partial u} \frac{\partial \boldsymbol{r}^*}{\partial v} \right) (\hat{u}_i, \hat{v}_j) \tag{23}$$

Singularities of the jacobian (deformations associated with a null singular value) describe the possible isometric motions of the surface around the current configuration.

Because equations (21) - (23) are functions of the basis function derivatives in u and v, the inextensibility grid has to cover the whole surface. The values of the basis function derivatives decrease as the distance from a control point increases. If the grid is localized in a certain region, the motion of a control point far from this region can create a numerical singularity as equations (21) - (23) become close to 0.

Rigid-body translations and rotations are obvious singularities of the jacobian which can be shown from equations (21) - (23). Additionally to rigid body motions, it is possible to have non-rigid singularities. These are important since they correspond to actual deformations of the surface. They are dependent on the current shape $\boldsymbol{r}^*$.

Table 2 shows the number of singularities for different shapes, number of control points and size of inextensibility grids, calculated numerically using MATLAB. Lagrange polynomials cannot exactly represent circular surfaces such as cylinders, cones, sphere, etc. These shapes were generated by placing the control points evenly on these mathematically defined surfaces. The inextensibility grid was also evenly spaced and mapped to the whole surface delimited by the control points. Different parameters where used to create the shapes (such as radii, cone opening angle, etc.) and the results were identical. Finally, the number of singularities was calculated using the MATLAB rank function on the jacobian of the system with the default tolerance parameter.

The plane yields the largest number of singularities, and the numerically calculated values are equal to $3+N_uN_v$. This result holds also for larger grids, as it is possible to see from equations (21) - (23) that the motion of each control point perpendicular to the plane creates a singularity. This set includes the rigid translation normal to the plane and the two rotations whose axes lie within the plane. The in-plane rigid translations and the rigid rotation around the normal are the 3 additional singularities, hence yielding the number $3 + N_uN_v$.

Table 2 shows that for all non-planar shapes, the actual number of singularities is much smaller than $3 + N_uN_v$.

Num. of control points	4×4	10×10	10×10
Size of inex. grid	4×4	10×10	20×20
Plane	19	103	103
Cylinder	12	30	10
Cone	9	20	6
Sphere	10	21	6
Random	8	20	6

Table 2: Numerically calculated number of singularities of the inextensibility constraints

4 Angle Measurement Constraints

4.1 Definition of the measurement

In order to measure the deformation of the surface, we introduce local angle measurements at several points on the surface. These measurements determine the angles between the normal to the surface and a specific direction.

We assume that the angles are measured to the line from the sensor location and the origin of $\mathbb{R}^3$ which coincides with the vector $\boldsymbol{r}$. This can be done in practice by placing a light source at the origin and light sensors on the surface.

At each sensor location, two angles α, β are measured along the curvilinear coordinates, that is along $\frac{\partial \boldsymbol{r}}{\partial u}$ and $\frac{\partial \boldsymbol{r}}{\partial v}$ from the normal of the surface as, shown in figure 6.

The location of the sensors is defined by $(u_S, v_S) \in \mathbb{R}^{N_S \times 2}$ where N_S is the total number of sensors. They do not need to lie on a specific grid as required by previous research [11,14,15]. Figure 7 shows an example of the location of the sensors in the uv-space and their position in the current configuration. The local coordinate system at the location of a sensor is also shown.

The angles at a sensor location (u_S, v_S) are defined as:

$$\tan \alpha_S = \frac{\boldsymbol{r}(u_S, v_S) \cdot \frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S)}{\boldsymbol{r}(u_S, v_S) \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\| \frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S) \right\|} \tag{24}$$

$$\tan \beta_S = -\frac{\boldsymbol{r}(u_S, v_S) \cdot \frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S)}{\boldsymbol{r}(u_S, v_S) \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\| \frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S) \right\|} \tag{25}$$

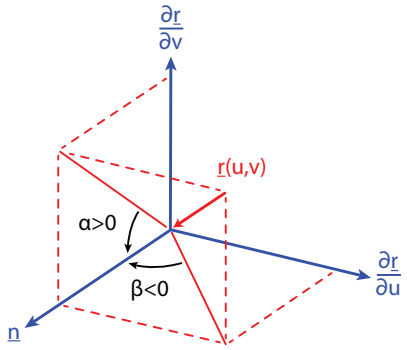


Fig. 6: Definition of the angles in the local coordinate system of a sensor. $\boldsymbol{r}\,(u_S, v_S)$ represents the direction to which the angles are measured. Note that the angle β is negative in the figure (positive angles are defined from $\boldsymbol{n}$ to $\frac{\partial \boldsymbol{r}}{\partial u}$).

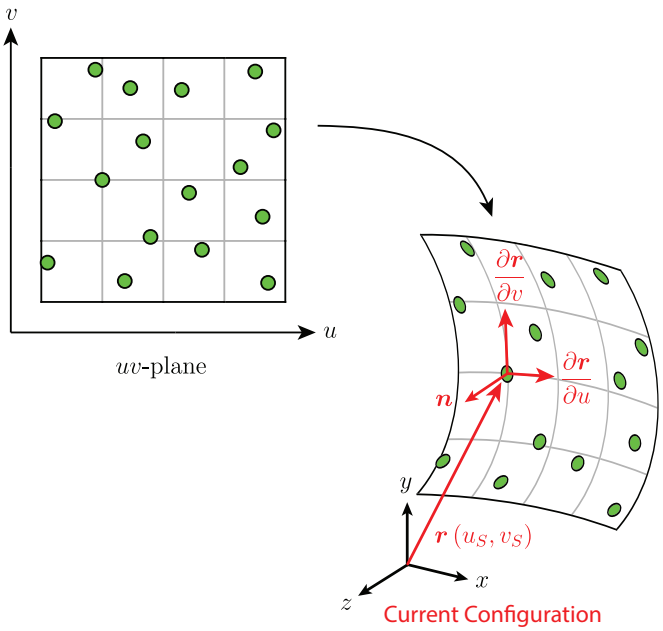


Fig. 7: Example position of the angle sensors on the surface (green circles). The vector defining the position of a sensor $\boldsymbol{r}(u_S, v_S)$ is collinear with the light ray seen by the sensor (the light source is positioned at the origin). The local coordinate system at that sensor location is also shown.

where $\boldsymbol{n}(u_S, v_S)$ is the normal to the surface:

$$\boldsymbol{n} = \frac{\partial \boldsymbol{r}}{\partial u} \times \frac{\partial \boldsymbol{r}}{\partial v} \tag{26}$$

Note that the normal of the surface needs to remain relatively close to the measurement direction, otherwise the denominator of the previous equations will approach zero. In practice, light sensors have a limited field-of-view which would limit the the measured angle in equations (24) and (25) to less than 90°.

Similar equations can be written when the angle measurement is taken from a fixed direction such as an infinitely distant light source along the vector $\boldsymbol{z}$. Then equations (24) and (25) become:

$$\tan \alpha_S = \frac{\boldsymbol{z} \cdot \frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S)}{\boldsymbol{z} \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\|\frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S)\right\|} \tag{27}$$

$$\tan \beta_S = -\frac{\boldsymbol{z} \cdot \frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S)}{\boldsymbol{z} \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\|\frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S)\right\|} \tag{28}$$

4.2 Rank Deficiency

The equations (24) and (25) are invariant for any rotation around the origin. This is because multiplying $\boldsymbol{r}$ by a rotation matrix also multiplies the local tangent and normal vectors. The dot products and norms are invariant by rotation. Uniform scaling (multiplying $\boldsymbol{r}$ by a non-zero coefficient) also creates a singularity.

Numerical rank deficiencies (singular values of the jacobian close to 0) can occur when sensors are concentrated on a specific part of the structure. This has similar effect to localizing the inextensibility grid on one part of the surface (see section 3.2). The shape functions only have a localized influence and if no sensor is located on a specific region of the structure then it can deform without modifying the measurement of the sensors.

To show this, we calculate the derivatives of the equations (24) and (25) with respect to $\boldsymbol{q_k}$. Because of the complexity of these equations, it is assumed that the deformation is perfectly inextensible. This means that $\left\|\frac{\partial \boldsymbol{r}}{\partial u}\right\|$, $\left\|\frac{\partial \boldsymbol{r}}{\partial v}\right\|$, and $\|\boldsymbol{n}\|$ are constant. Without loss of generality, they are set to 1. Additionally, the light source is assumed to be very far from the surface and along the z-direction (equations (27) and (28)).

With these simplifications, one can show that the derivatives about an initial shape $\boldsymbol{r}^*$ can be written as:

$$\frac{\partial \tan \alpha_S}{\partial \boldsymbol{q_k}} = \frac{\tan \alpha_S^*}{\boldsymbol{z} \cdot \boldsymbol{n}^*} \begin{bmatrix} \frac{\partial \boldsymbol{r}^*}{\partial v}\frac{\partial \phi_k}{\partial u} - \frac{\partial \boldsymbol{r}^*}{\partial u}\frac{\partial \phi_k}{\partial v} \\ \frac{\partial \boldsymbol{r}^*}{\partial u}\frac{\partial \phi_k}{\partial v} - \frac{\partial \boldsymbol{r}^*}{\partial v}\frac{\partial \phi_k}{\partial u} \\ \frac{1}{\tan \alpha_S^*}\frac{\partial \phi_k}{\partial v} \end{bmatrix} \tag{29}$$

$$\frac{\partial \tan \beta_S}{\partial \boldsymbol{q_k}} = \frac{\tan \beta_S^*}{\boldsymbol{z} \cdot \boldsymbol{n}^*} \begin{bmatrix} \frac{\partial \boldsymbol{r}^*}{\partial v}\frac{\partial \phi_k}{\partial u} - \frac{\partial \boldsymbol{r}^*}{\partial u}\frac{\partial \phi_k}{\partial v} \\ \frac{\partial \boldsymbol{r}^*}{\partial u}\frac{\partial \phi_k}{\partial v} - \frac{\partial \boldsymbol{r}^*}{\partial v}\frac{\partial \phi_k}{\partial u} \\ \frac{1}{\tan \beta_S^*}\frac{\partial \phi_k}{\partial u} \end{bmatrix} \tag{30}$$

From these equations, we can see that if $\left\|\frac{\partial \phi_k}{\partial u}\right\| \ll 1$ and/or $\left\|\frac{\partial \phi_k}{\partial v}\right\| \ll 1$ for all $(u_S, v_S) \in \mathbb{R}^{N_S \times 2}$ then both derivatives have components close to 0 and some

columns of the jacobian will be close to 0 leading to a numerical singularity. This would be the case if, for example, only a part of the surface is covered with angle sensors. The shape functions associated with the control points located far away from the sensors will have near-zero derivatives at the sensor locations.

5 Surface Reconstruction as a Least-Squares Problem

The numerous constraints detailed in the previous sections 3 and 4 (equations (13), (14), (15), (24), and (25)) are used to determine the shape of a surface parametrized with 2D Lagrange Polynomials (section 2). This algorithm can be written as follows:

Input: Initial Location of the Lagrange Polynomials control points $\bar{\boldsymbol{q}}_{\boldsymbol{0}}$
Location of the inextensibility grid vertices $\{(\hat{u}_i, \hat{v}_j), i = 1, ..., \hat{N}_u, j = 1, ..., \hat{N}_v\}$
Location of the sensors $\{(\hat{u}_S, \hat{v}_S), S = 1, ..., \hat{N}_S\}$
Points of interest where to reconstruct the surface $\{(u, v)\}$
Result: Optimal shape of the surface $\tilde{\boldsymbol{r}}$
$\bar{\boldsymbol{q}}^* \leftarrow \bar{\boldsymbol{q}}_{\boldsymbol{0}}$
Loop
  $(\alpha_S, \beta_S) \leftarrow$ RetrieveMeasurements();
  $\bar{\boldsymbol{q}}^* \leftarrow$ GetCPLocations$(\bar{\boldsymbol{q}}^*, \hat{u}_i, \hat{v}_j, \hat{u}_S, \hat{v}_S, \alpha_S, \beta_S)$;
  $\tilde{\boldsymbol{r}} \leftarrow$ GetShape$(\bar{\boldsymbol{q}}^*, u, v)$
EndLoop

Algorithm 1: Reconstruction of the shape of a surface from discrete angle measurements

Here, the function RetrieveMeasurements() collects the angle measurement from all sensors (or creates random measurements in the case of simulations), GetCPLocations() is a minimization function described in the next section 5.1 and GetShape() is simply equation (3). The algorithm loops indefinitely as long as measurements are available. A result, i.e., a shape of the structure is given at the end of each loop. GetCPLocations() gets initiated by the result of the previous loop except for the first loop where an initial guess is given.

5.1 System of Equations

Equations (13), (14), (15), (24), and (25) form a system that solves the problem of reconstructing a surface from angle measurements undergoing an inextensible deformation from a template:

$$\int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\| \frac{\partial \boldsymbol{r}}{\partial u}(u, \hat{v}_j) \right\| du - \int_{\hat{u}_i}^{\hat{u}_{i+1}} \left\| \frac{\partial \boldsymbol{r^0}}{\partial u}(u, \hat{v}_j) \right\| du = 0 \qquad \forall i = 1, ..., \hat{N}_u - 1, \quad \forall j = 1, ..., \hat{N}_v$$

$$\int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\| \frac{\partial \boldsymbol{r}}{\partial v}(\hat{u}_i, v) \right\| dv - \int_{\hat{v}_j}^{\hat{v}_{j+1}} \left\| \frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i, v) \right\| dv = 0 \qquad \forall i = 1, ..., \hat{N}_u, \quad \forall j = 1, ..., \hat{N}_v - 1$$

$$\frac{\partial \boldsymbol{r}}{\partial u}^T(\hat{u}_i, \hat{v}_j) \frac{\partial \boldsymbol{r}}{\partial v}(\hat{u}_i, \hat{v}_j) = \frac{\partial \boldsymbol{r^0}}{\partial u}^T(\hat{u}_i, \hat{v}_j) \frac{\partial \boldsymbol{r^0}}{\partial v}(\hat{u}_i, \hat{v}_j) \qquad \forall i = 1, ..., \hat{N}_u, \quad \forall j = 1, ..., \hat{N}_v$$

$$\tan\alpha - \frac{\boldsymbol{r}(u_S, v_S) \cdot \frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S)}{\boldsymbol{r}(u_S, v_S) \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\|\frac{\partial \boldsymbol{r}}{\partial v}(u_S, v_S)\right\|} = 0 \qquad \forall S = 1, ..., N_S$$

$$\tan\beta + \frac{\boldsymbol{r}(u_S, v_S) \cdot \frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S)}{\boldsymbol{r}(u_S, v_S) \cdot \boldsymbol{n}(u_S, v_S)} \frac{\|\boldsymbol{n}(u_S, v_S)\|}{\left\|\frac{\partial \boldsymbol{r}}{\partial u}(u_S, v_S)\right\|} = 0 \qquad \forall S = 1, ..., N_S \tag{31}$$

Replacing $\boldsymbol{r}$ with equation (3) everywhere, the system is a function of the unknown control point coordinates, and can be written as:

$$f_k(\bar{\boldsymbol{q}}) = 0 \qquad \forall k = 1, ..., N_{eq} \tag{32}$$

where $f_k(\cdot)$ represents the equations defined in (31), $N_{eq} = 3\hat{N}_u\hat{N}_v - \hat{N}_u - \hat{N}_v + 2N_S$ is the number of equations in the system, and $\bar{\boldsymbol{q}}$ is the unknown vector defined as the vertical concatenation of the control points. Since this system is (usually) overconstrained, we rewrite the problem as a least-squares minimization:

$$\bar{\boldsymbol{q}}^* = \arg\min_{\bar{\boldsymbol{q}}} \sum_{k=1}^{N_{eq}} f_k^2(\bar{\boldsymbol{q}}) = \arg\min_{\bar{\boldsymbol{q}}} \|\boldsymbol{f}(\bar{\boldsymbol{q}})\|^2 \tag{33}$$

where $\boldsymbol{f}(\bar{\boldsymbol{q}}) = [f_1(\bar{\boldsymbol{q}}), ..., f_{N_{eq}}(\bar{\boldsymbol{q}})]^T$.

5.2 Solution of Overconstrained System

In order to solve the least-squares problem defined in equation (33), a locally optimal solution, given an initial guess can be found using the Levenberg-Marquardt method in MATLAB [34].

This iterative algorithm starts with an initial guess $\bar{\boldsymbol{q}}_{\boldsymbol{0}}$ (for which details are provided in section 6.5) , finds

increments of the unknown vector, $\boldsymbol{\delta\bar{q}_k}$, such that at the next step, $\boldsymbol{\bar{q}_{k+1}} = \boldsymbol{\bar{q}_k} + \boldsymbol{\delta\bar{q}_k}$. The increment at each step solves the equation:

$$\left(\mathbf{J}^T\mathbf{J} + \lambda_k\mathbf{I}\right)\ \boldsymbol{\delta\bar{q}_k} = -\mathbf{J}^T\boldsymbol{f}(\boldsymbol{\bar{q}_k}) \tag{34}$$

where $\mathbf{J} = \frac{\partial \boldsymbol{f}}{\partial \bar{\boldsymbol{q}}}$ is the jacobian of the system and λ_k is a non-negative damping factor that is optimized at each step to maximize the decrease in the residual. If $\lambda_k = 0$ this algorithm is equivalent to the Gauss-Newton algorithm (GNA). It is however more robust than GNA when the initial guess is far from the solution. When $\lambda_k \to \infty$, the algorithm tends to the gradient descent algorithm. As a result, the Levenberg-Marquardt method is a hybrid of the two algorithms.

The Levenberg-Marquardt algorithm stops when the increment is smaller than a prescribed tolerance , i.e. when:

$$\|\boldsymbol{\delta\bar{q}_k}\| < \epsilon \tag{35}$$

It is important to note that since the system is over-constrained and the stopping criterion involves the norm of the increment, the scaling of the equations (31) matters. This will be shown through examples in the next section.

5.3 Rank Deficiencies

The jacobian of the system can be calculated as the vertical concatenation of the jacobian studied in subsection 3.2 and 4.2. In order to use the Levenberg-Marquardt algorithm, the jacobian needs to be full rank, i.e., its columns need to be independent.

A first requirement is to have more equations than unknowns: $N_{eq} = 3\hat{N}_u\hat{N}_v - \hat{N}_u - \hat{N}_v + 2N_S > 3N_uN_v$. This is only a necessary condition, observations from sub-sections 3.2 and 4.2 show that the jacobian remains singular under certain motions.

All equations are invariant for rigid-body rotations around the origin. Without loss of generality, we constrain 3 coordinates among the control points to be fixed. For instance, one point is restrained from moving along the x-axis and y-axis, and another point is constrained from moving along the x-axis.

Additionally, the inextensibility conditions can have up to N_uN_v singularities as explained in sub-section 3.2. These are mutually exclusive with the singularities of the angle equations. As a result, a minimum of N_uN_v angle measurements are needed to have a full rank jacobian, i.e. at least as many angle measurements as control points.

It is important to notice that when the distance from the origin to the surface becomes large, another numerical singularity emerges. It corresponds to a spherical motion. Figure 8 shows a schematic of this singularity in 2D. A line perpendicular to the light source can conform to any circle, hence there are infinitely many solutions. In 3D, this is not a singularity as its associated singular value is not exactly 0: conforming a section of a sphere to a sphere of different radius is an extensible deformation. This singularity is however orders of magnitude lower than the next higher which causes numerical issues. We call this singularity a *spherical singularity*.

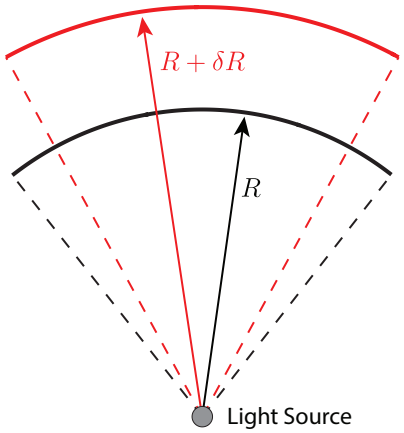


Fig. 8: 2D representation of the spherical singularity of the system of equations. The black and red lines have the same length and angles to the light source but their shapes are different.

To remedy this issue, the distance of a point on the surface to the origin can be fixed or bounded. This would ensure that the solution remains at a relative distance from the light source, removing the effect of the spherical singularity which allows large translation with relative small shape deformations.

In the case where the direction of the angle measurement is fixed (equations (27) and (28)), this numerical singularity becomes an actual singularity corresponding to the rigid-body translation along that direction. Fixing a point in 3D space can be done without loss of generality.

6 Simulation Results

A set of simulations is performed to better understand the performance of the proposed method and highlight its limitations.

The surface to reconstruct is a piece of paper of size A4 (297×210 mm^2).The sheet is initially flat (reference configuration). It is deformed to a conical shape located 1 m away from the light source. It is assumed that the

sensors have a 90° field of view so that the surface can bend to large angles relative to the light. They are also assumed to be small so that they do not constrain the deformation of the sheet otherwise.

Unless specified, a grid of 7×5 control points is used to generate the Lagrange shape functions. An identical grid is used for the inextensible grid and each vertex is the location of a sensor. Each sensor has gaussian noise which is assumed identical for all sensors and set to a standard deviation of 10 arcminutes (3σ of 0.5°). The effect of the noise on the reconstructed shape will be studied. A total of 500 measurements is generated in order to get statistically accurate results.

Reconstruction accuracy. The error of the reconstruction is calculated as the norm of the vector joining the points with equal coordinates (u, v) in the reconstructed and exact shapes:

$$\text{Error}(u, v) = \|\boldsymbol{r}(u, v) - \tilde{\boldsymbol{r}}(u, v)\| \tag{36}$$

To characterize the accuracy of the method over the whole surface for many different reconstructions, the mean error, which corresponds to the average distance between the reconstructed and true shapes is calculated:

$$e = \frac{1}{|A|} \int_A \|\boldsymbol{r}(u, v) - \tilde{\boldsymbol{r}}(u, v)\| \, dudv \tag{37}$$

where A is the surface area of the sheet of paper.

Inextensibility. The isometry of the surface deformation is evaluated by plotting the error of the 3 elements of the metric tensor from their nominal values defined in equations (10) - (12). Note that in this case, $\mathbf{M^0} = \mathbf{I}$, the 2×2 identity tensor (developable surface).

To better understand the elements of the tensor, we plot the Lagrangian normal and shear strains of the surface [35]. They are given by:

$$\epsilon_u = \frac{1}{2} (\mathbf{M_{11}} - 1) \tag{38}$$

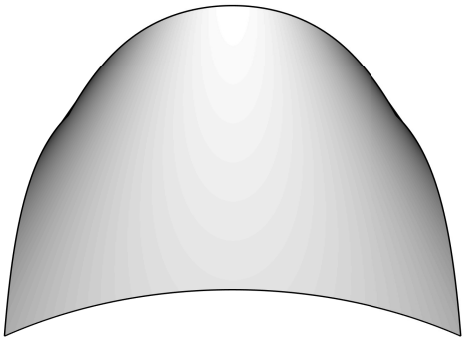
$$\epsilon_v = \frac{1}{2} (\mathbf{M_{22}} - 1) \tag{39}$$

$$\gamma_{uv} = \mathbf{M_{12}} \tag{40}$$

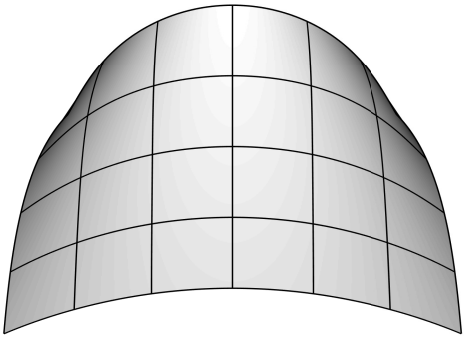
6.1 Reconstruction Errors

Figure 9 shows the average reconstructed shape from the 500 generated measurements of the angle sensors as well as the actual conical shape. They agree well qualitatively as little to no difference can be seen.

To further understand the accuracy of the method, the RMS error of the reconstructed shapes along the curvilinear coordinates is calculated and shown in figure 10. Note that rigid-body motions between the reconstructed and actual shapes were removed. The error is maximum towards the top of the sheet which is where the curvature is maximum. This is due to the



(a) Actual conical shape of the structure.



(b) Reconstructed shape of the structure. The isometry grid is shown. Each vertex also corresponds to the location of the control points and light sensors.

Fig. 9: Actual and reconstructed conical shapes viewed from the light source located 1 m in front of the structure.

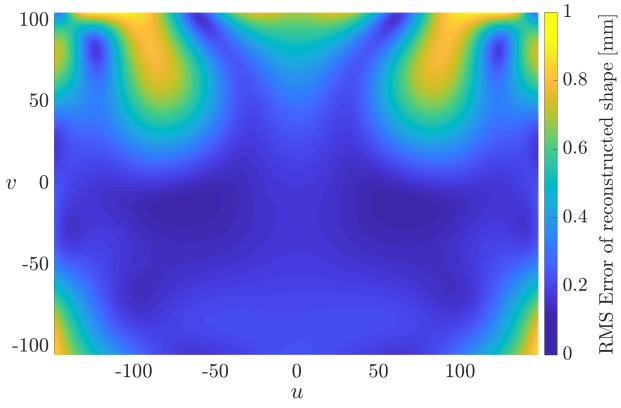


Fig. 10: RMS error of reconstructed conical shapes based on 500 simulated sensor measurements.

fact that polynomial shape functions cannot perfectly reconstruct circular segments, which introduce errors in the reconstruction. The mean error of the reconstructions is equal to 0.3 ± 0.03 mm (standard deviation from all the reconstructed surfaces).

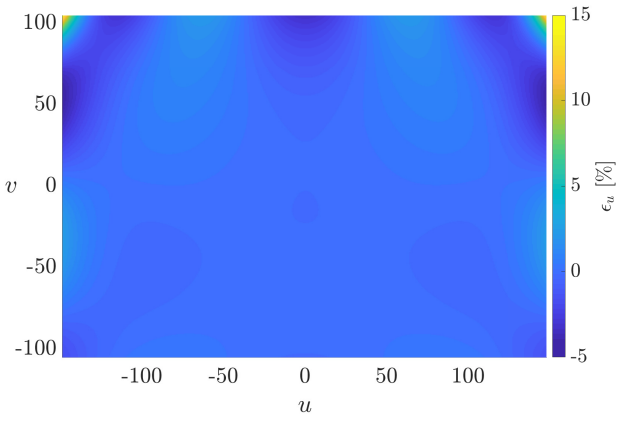
To analyze the effect of the inextensibility constraints, the normal and shear strains are plotted in figure 11. The values correspond to the average strains over the 500 reconstructed shapes.

The normal strains are close to 0 on average along each segment of the inextensibility grid. However, they still vary across the surface. A similar observation can be made for the shear strains that are also close to 0 at each node of the inextensibility grid but are non-zero elsewhere.

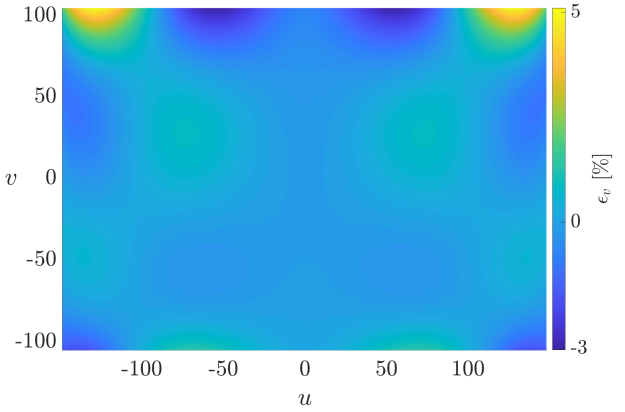
The normal strain in the u-direction is dominant and increases as the curvature increases. Since Lagrange polynomials cannot perfectly reconstruct circular segments, the algorithm needs to stretch the surface in order for the measured angles to better match their set values. More precisely, the algorithm does a tradeoff between slightly stretching the surface and adding some bias to the measured angles.

The shear strain variation is related to the normal strains: the small non-uniform accumulated strain displaces the points of the surface forcing it to shear.

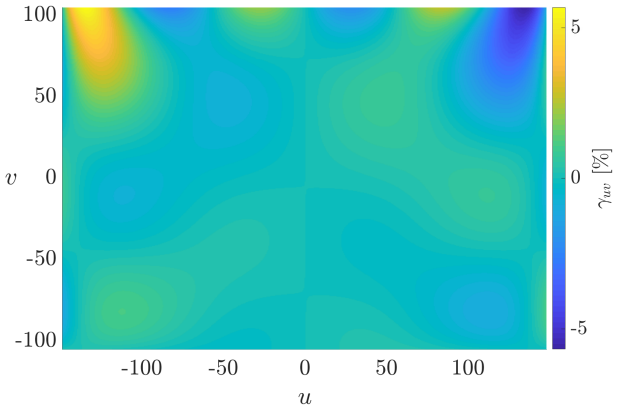
These results are similar to some monocular template-based surface reconstruction algorithm such as Second Order Cone Programs (SOCP) [6]. Some algorithms out-perform the one presented in this paper as denser meshes can be used to represent the surface since more measurement data is available.



(a) Average normal strain in the u-direction, equation (38).



(b) Average normal strain in the v-direction, equation (39).



(c) Average shear strain, equation (40).

Fig. 11: Average strains of reconstructed conical shapes.

6.2 Effect of Sensor Noise

Figure 12 shows the evolution of the mean error across the surface in function of the 3σ noise of the sensors.

For the studied structure and shape, the error rapidly increases with the noise of the sensor. For sufficiently low noise, the error does not vary significantly as it is limited by the efficiency of the shape functions to accurately represent the shape of the surface. In this case, the minimum error considering perfect sensors is 0.26 mm. This sets a bound on the accuracy of the sensors. There is no need for very accurate sensors (3σ noise below $0.1°$) as they do not provide significantly better results. More accurate shape functions (such as NURBS) could decrease this minimum error and justify the use of more precise sensors.

The uncertainty of the reconstructed shape shown in figure 12 by the gray area which represents the standard deviation of the mean error also increases as the noise of

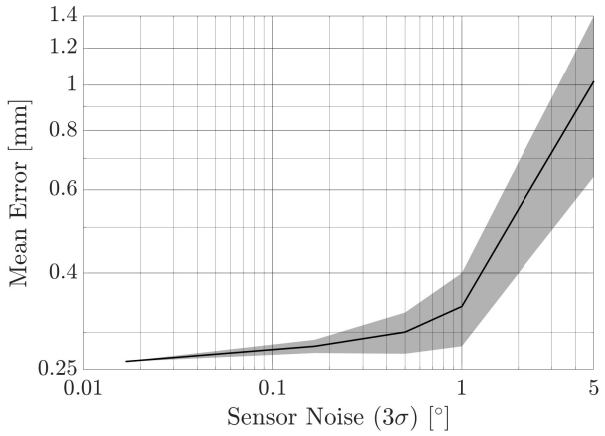


Fig. 12: Mean error of reconstructed conical shapes.

the sensors increase: not only the accuracy gets worse, it is also less predictable.

6.3 Convergence of Solution

To verify the convergence of the solution, the number of control points was increase effectively increasing the order of the Lagrange polynomials. To avoid rank deficiencies, the number of sensors needs to be increased. The same grid was used for the interpolation points of the Lagrange polynomials, inextensibility constraints and angle measurements. Its size was increased using the following sequence: 5×3, 7×5, 9×7, 11×7, 13×9, 15×11, and 17×13.

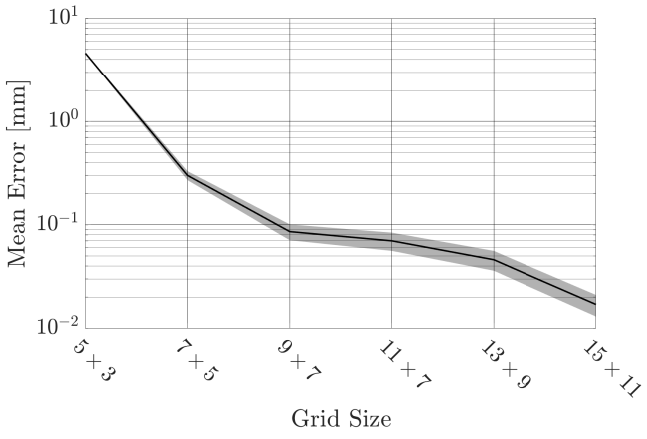


Fig. 13: Variation of the mean error between reconstructed and actual shapes by varying the grids size.

The convergence results for the conical shape are shown in figure 13. The x-axis shows the size of the grid while the y-axis shows the mean error (solid line) and standard deviation (gray area) from 500 sensor measurements. It can be seen that, as the grid gets more refined, the accuracy of the algorithm is improved, as expected.

Note that convergence issues arise for denser grids. The initial shape is modified to be close to the solution by positioning each control points at its location on the conical shape (figure 9a). More details are presented in sub-section 6.5. Sparse grids do not show convergence issues based on the initial shape.

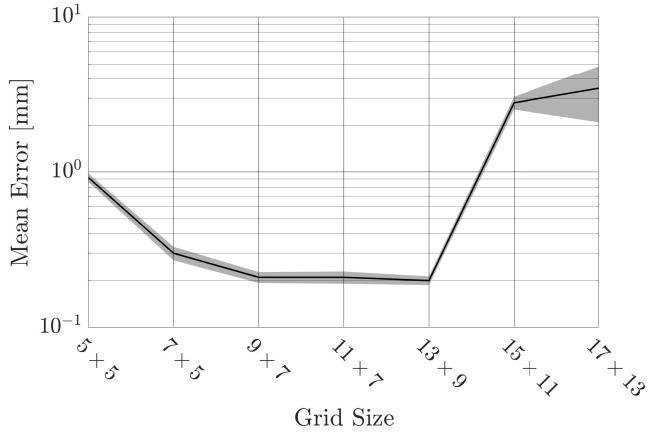
6.4 Variation of Algorithm Parameters

For a fixed number of control points, the solution depends on three sets of parameters: the number and position of the angle sensors, the coordinates of the inextensibility grid, and the weight of each equation of system (31). By changing these parameters, the algorithm converges to different solutions that will have different errors. All parameters were varied for a fixed 7×5 grid of control points, and their impact on the mean error of the reconstructed cone have been studied. The results are shown in figure 14.

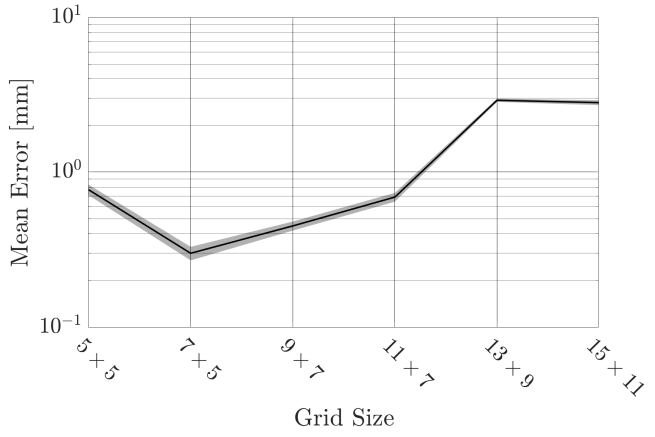
The size of the inextensibility grid was varied from 5×5 to 17×13. The minimum grid size was dictated by the minimum number of equations needed in order to have a well-posed problem. The results of varying this grid size are shown in figure 14a. The curve is not monotonic and there exists an optimum grid size that provides the lowest mean error.

Varying the number of angle measurements was performed by spreading the sensors on a uniform grid whose size ranges from 5×5 to 15×11 while using fixed 7×5 control points and inextensibility grids. The results are shown in figure 14b. Similarly, an optimal grid of sensors provides the lowest mean error.

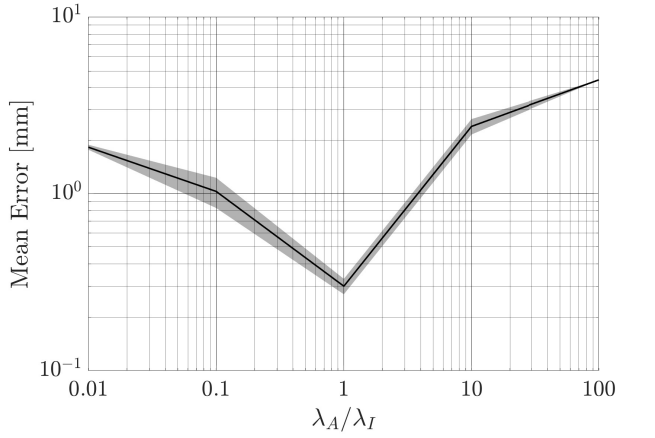
As stated in sub-section 5.2, the weight of each equation of the system is important. We investigate the difference of weight between the first 3 equations, which relate to the inextensibility of the surface and the last two, which impose constraints on the measured angles. Note that the first two equations of system (31) were made adimensional by dividing by the length of the edges of the inextensibility grid in the reference configuration. The other constraints are properly scaled and do not depend on the size of the grids. Figure 14c shows the influence of scaling the constraints. The inextensibility conditions were multiplied by a scaling factor $\lambda_I = 1$ and the angle constraints by λ_A which varies from 10^{-2} to 10^2. An optimal gain ($\lambda_A = 1$) yields an optimal reconstruction.



(a) Mean error of reconstructed shapes while varying the size of the *inextensibility grid*.



(b) Mean error of reconstructed shapes while varying the size of the grid of *angle measurements*.



(c) Mean error of reconstructed shapes while varying the *weight of the angle constraints* to the inextensibility constraints.

Fig. 14: Mean error of reconstructed shapes while varying different parameters of the algorithm using a 7×5 grid of control points.

All graphs show similar results. The error evolves as a function of the parameters. It gets higher for more sparse and denser grids and reaches a minimum in-between. This can be explained by the fact that the shape is approximated by Lagrange shape functions. Because of this approximation, the reconstructed shape is neither perfectly inextensible, nor it perfectly matches the angle measurements. Some relaxation of the constraints is needed. As a result, increasing the size of the grids leads to many constraints that cannot be met by the approximated shape. Inversely, smaller grids gives more weights to the other equations. Varying the scaling of the equations forces the algorithm to enforce constraints more or less tightly.

An optimization program could be developed to optimize the parameters of the algorithm to minimize the mean error of its solution to desired shapes. Results presented in this paper show that matching the inextensibility grid with the control points and the angle sensors leads to a near optimal scheme.

6.5 Influence of the Initial Shape

As mentioned in the previous section, the initialization of the algorithm may affect the results as the cost function is non-convex. To illustrate this point, we study the case of 9×7 grids for control points, inextensibility grids and sensor locations. It was noticed that all initial shapes for sparser grids lead to a converged solution. We study the result of the algorithm from different initial shapes ranging from a flat one (used as the initial shape in sub-section 6.1) to the solution (figure 9a):

$$\text{Initial Shape} = \alpha\,[\text{Flat}] + (1-\alpha)\,[\text{Conical Solution}] \tag{41}$$

where α ranges from 0 to 1.

Results are shown in figure 15. The algorithm does not converge for high values of α (close to the flat shape) but eventually converges when the initial shape is close enough to the solution. Note that the curve is not monotonic which may be due to the non-convexity of the problem.

To circumvent this issue, a simple solution obtained from a small number of control points can be solved first as it appears to be more reliable and less dependent on the initial shape. This solution can be used as a starting point for denser grids of control points. This can be done recursively. Once converged, the algorithm does not appear to have problems following changes in sensor inputs.

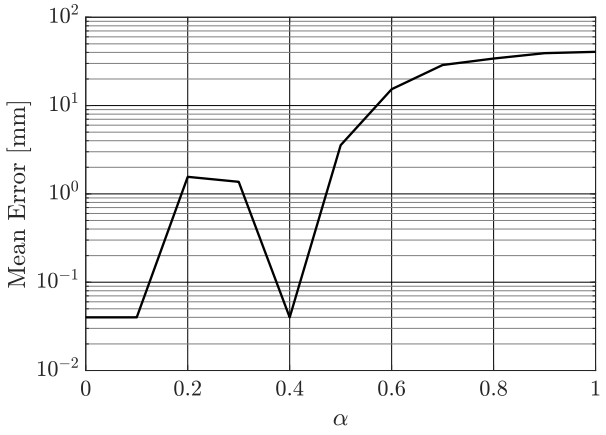


Fig. 15: Mean error of reconstructed shapes using 9×7 grids depending on the initial shape.

6.6 Computation Time

The computation time is dependent on the number of constraints and control points. Both influence the size of the jacobian matrix used in the Levenberg-Marquardt algorithm which eventually has to be inverted. No particular effort was made to optimize the speed of the algorithm except pre-computing the values of the Lagrange shape functions and its derivatives at the different locations on the surface (the different integrations used in the isometry constraints were performed using Simpson's rule).

Having the initial shape relatively far also increases the computation time as many iterations need to be performed. Once converged, the reconstructed shape can be used as the initial point for the next reconstruction using a new set of measurements. This next iteration of the algorithm becomes much quicker.

To compute the reconstructed shapes shown in figure 9b, the algorithm was run using MATLAB on an Intel Core i5-6200 CPU. The first iteration of the algorithm takes about 5 s to run as the initial shape is far from the solution. The subsequent steps take 50 ms on average as only a couple loops of the Levenberg-Marquardt method are needed.

7 Comparison to the State-of-the-Art

To the knowledge of the authors, no other methodology exists to reconstruct the shape of a surface with embedded local angle measurements such as light sensors. However, related research has used another type of sensor, Inertial Measurement Units (IMUs), which measure the direction of Earth's gravity and magnetic field, not by measuring angles but by measuring the coordinates of these vectors along the 3 axes of the sensor which correspond to the 3 local axes of the surface. The measurement equations are similar to equations (27) and (28) which can be re-written to express the unit vector collinear with the light direction in the local reference system of the sensor:

$$\boldsymbol{z}_S = \begin{bmatrix} \frac{\tan\alpha_S}{\sqrt{1+\tan^2\alpha_S+\tan^2\beta_S}} \\ -\frac{\tan\beta_S}{\sqrt{1+\tan^2\alpha_S+\tan^2\beta_S}} \\ \frac{1}{\sqrt{1+\tan^2\alpha_S+\tan^2\beta_S}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)\right\|} \\ \boldsymbol{z}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)\right\|} \\ \boldsymbol{z}\cdot\frac{\boldsymbol{n}(u_S,v_S)}{\|\boldsymbol{n}(u_S,v_S)\|} \end{bmatrix} \tag{42}$$

This expression is equivalent to equations (27) and (28) as it carries two independent pieces of information since all vectors are unit vectors (including $\boldsymbol{z}$). In the case of an IMU, two directions are measured directly:

$$\boldsymbol{g_S} = \begin{bmatrix} \boldsymbol{g}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)\right\|} \\ \boldsymbol{g}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)\right\|} \\ \boldsymbol{g}\cdot\frac{\boldsymbol{n}(u_S,v_S)}{\|\boldsymbol{n}(u_S,v_S)\|} \end{bmatrix} \tag{43}$$

$$\boldsymbol{B_S} = \begin{bmatrix} \boldsymbol{B}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial u}(u_S,v_S)\right\|} \\ \boldsymbol{B}\cdot\frac{\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)}{\left\|\frac{\partial \boldsymbol{r}}{\partial v}(u_S,v_S)\right\|} \\ \boldsymbol{B}\cdot\frac{\boldsymbol{n}(u_S,v_S)}{\|\boldsymbol{n}(u_S,v_S)\|} \end{bmatrix} \tag{44}$$

where $\boldsymbol{g_S}$ (resp. $\boldsymbol{B_S}$) is Earth's gravity (resp. magnetic field) measured by the sensor and $\boldsymbol{g}$ (resp. $\boldsymbol{B}$) is the direction of Earth's gravity (resp. magnetic field) in the global reference system (all unit vectors).

Equations (43) and (44) were used instead of the previously introduced angle measurements (equations (27) and (28)). They are very similar in nature except that two non-collinear directions are measured instead of one and the measurement is better behaved as coordinates stay bounded, unlike the tangent function.

These new measurement equations were integrated in the algorithm and simulations based on this approach were used to evaluate the performance of the presently proposed method. The results are compared to two state-of-the-art algorithms [11] and [14]. The same size of surface, location of sensors, noise of sensors, inextensibility grid and control points as the simulation in the previous section are used.

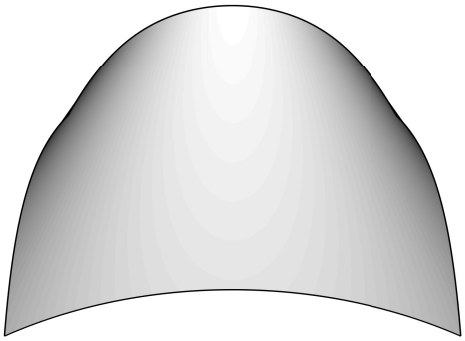
Hermanis et al. [11] propose a linear algorithm to reconstruct the shape. The two measured vectors ($\boldsymbol{g_S}$ and $\boldsymbol{B_S}$) are used to compute the tangent vectors of the surface at each sensor location without ambiguity. It is then assumed that the sensors are connected by means of rigid bars. The position of each sensor is then calculated by integrating the tangent vectors, effectively using a midpoint integration rule. The position of each

sensor is calculated as the average result from two different integration schemes: 1. the position of the sensors placed on the central row is calculated by integrating along u and using the central sensor as a reference (clamped to the origin of the coordinate system). The position of the other sensors are calculated by integrating along v using the position of the sensors on the central row as reference. 2. The position of the sensors on the central column are calculated first then the ones on each row. The rest of the surface is calculated from the position of each sensor using a bilinear interpolation scheme.

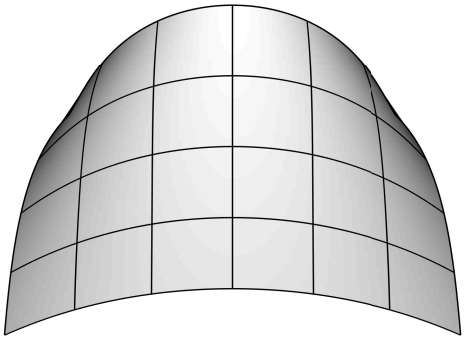
Saguin-Sprynski et al. [14] use a more complex integration scheme. The tangent vectors at each sensor location are also inferred from the measurements. Then the bottom and left edge of the surface (aligned with the bottom left sensor) are calculated by integrating the tangent vector which is interpolated from the sensor data using cubic splines on the sphere [36]. The location of the other sensors is calculated iteratively from a constrained minimization problem that enforces isometry along the curves between sensors while matching the sensor data. The surface is finally reconstructed using a partially bi-cubically blended Coons process [37].

7.1 Reconstruction Errors

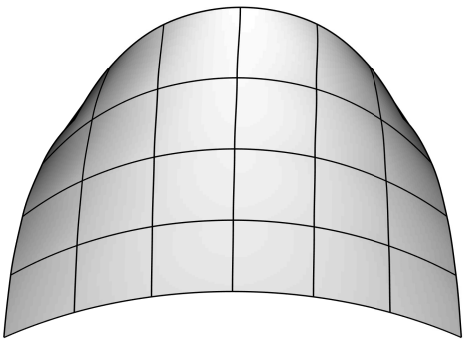
The shapes reconstructed by each algorithm are shown in figure 16. Each method is able to capture the overall shape of the structure. Figure 17 shows the RMS error of the reconstructions across the surface. Different amounts of error between algorithms are clearly visible. The one from Hermanis et al. [11] is the less accurate with a mean error of 2.01 ± 0.01 mm. While the error on the curves between sensors is relatively small, the simple filling method creates large errors when the curvature increases. The algorithm from Saguin-Sprynski et al. [14] yields a mean error of 1.3 ± 0.06 mm. While smoother, this method reconstructs the surface from the bottom left corner to the top right while strictly enforcing inextensibility on the curves between sensors. As a result, the error is maximal on the top right corner as the noise of the sensors and errors in the integration method propagates. One can see in figure 16c that the surface slightly shears to the right which is a consequence of the strict length constraints. Finally our proposed method has a mean error of 0.26 ± 0.01 mm. Note that this value is a bit lower than the mean error of 0.3 ± 0.03 obtained from light sensors, quoted in section 6.1, because each IMU provides four pieces of information instead of two. Also note that the overall pattern of the RMS error, in figure 17a, is very similar to figure 10. Overall, this result is better than the



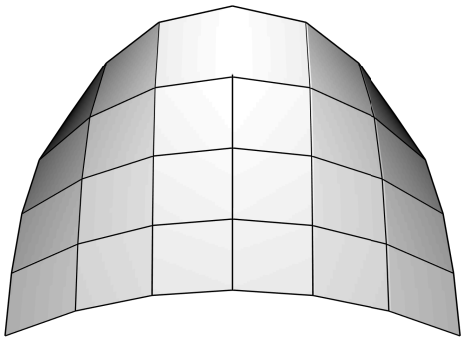
(a) Actual conical shape of the structure.



(b) Reconstructed shape using algorithm presented in this paper.

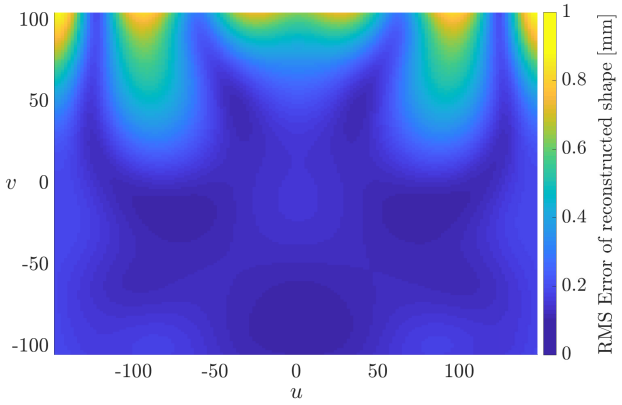


(c) Reconstructed shape using algorithm from Saguin-Sprynski et al. [14].

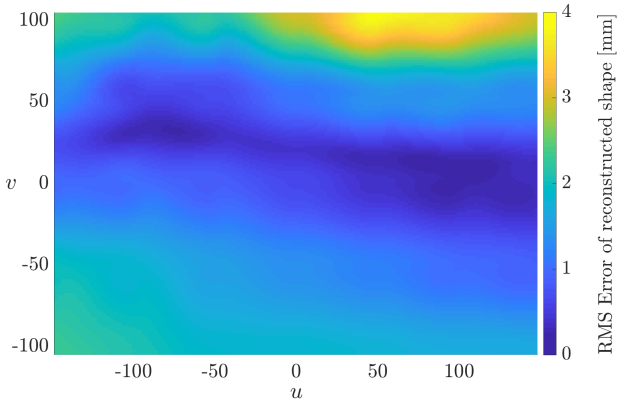


(d) Reconstructed shape using algorithm from Hermanis et al. [11].

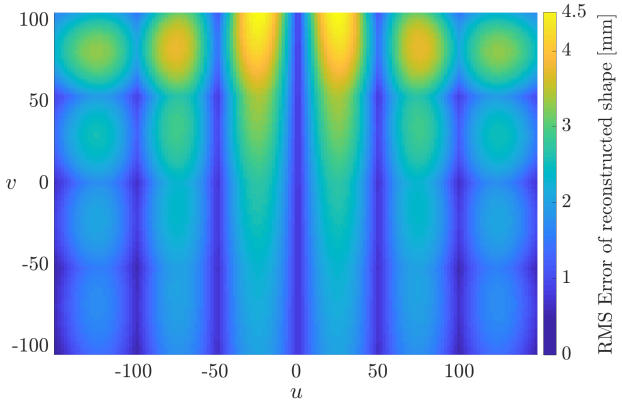
Fig. 16: Actual and reconstructed conical shapes using IMUs and different algorithms.



(a) RMS error of reconstructed shapes using algorithm presented in this paper.



(b) RMS error of reconstructed shapes using algorithm from Saguin-Sprynski et al. [14].



(c) RMS error of reconstructed shapes using algorithm from Hermanis et al. [11].

Fig. 17: RMS error of reconstructed shapes based on 500 simulated measurements using IMUs and different algorithms.

state-of-the-art by almost an order of magnitude under the same conditions.

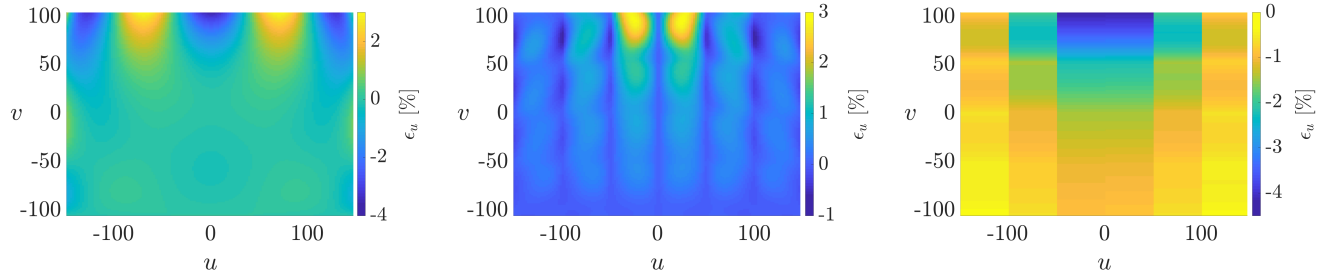
Figures 18, 19, and 20 show the average strain components of the reconstructed shapes. The normal strains are similar in terms of amplitude and overall shape for all different algorithms. They are largest at the top where the curvature is largest as the parameterization of the shapes cannot perfectly capture this effect. The main difference occurs for the shear strain. Our method outperforms the other by a factor of 2.5 or more. This is to be expected as our method includes the shear strains in the inextensibility formulation of the surface. This limits the propagation of an in-plane shift that can be seen in the reconstructed shape in figure 16c.

7.2 Effect of Sensor Noise

As in the previous section, the mean error of each algorithm is studied as a function of the noise of the sensors. The results are shown in figure 21 in logarithmic scale to compare the shape of each curve while the results are an order of magnitude different. The accuracy of each method is monotonic with the noise of the sensor, eventually reaching a plateau for low noise which corresponds to the minimum error possible with perfect sensors. This number is intrinsic to each algorithm and characterizes the efficiency of the method. In this respect, our algorithm does a much better job, as its asymptote is about an order of magnitude lower than the other ones. However, the error increases faster when the 3σ noise is greater than $1°$ which shows the importance of using precise sensors for our proposed method.

7.3 Convergence of Solution

The convergence of each method is studied by increasing the number of sensors. The results are shown in figure 22. For our method, we also increase the number of control points and size of inextensibility grid such that they all coincide as in sub-section 6.3. The mean error of the reconstructed shape using the algorithm presented in this paper decreases more rapidly as the density of sensors increases compared to the other methods. This means that adding even a few sensors has a much greater impact on the error, which can be very beneficial in practical applications. The error of the two other methods remains higher and eventually the two methods converge, as the linear integration along the curves between sensors used in Hermanis et al. [11] is a good approximation for the higher-order integration method in Saguin-Sprynski et al. [14].

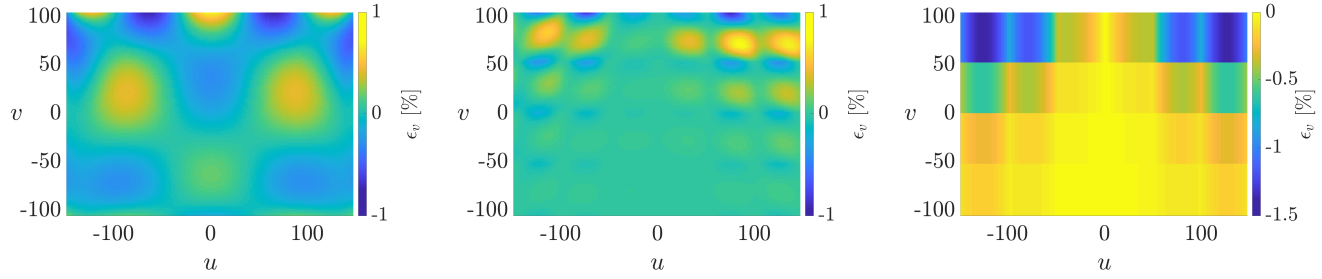


(a) Average normal strain in the u-direction using algorithm presented in this paper.

(b) Average normal strain in the u-direction using algorithm from Saguin-Sprynski et al. [14].

(c) Average normal strain in the u-direction using algorithm from Hermanis et al. [11].

Fig. 18: Average normal strain in the u-direction using IMUs and different algorithms.

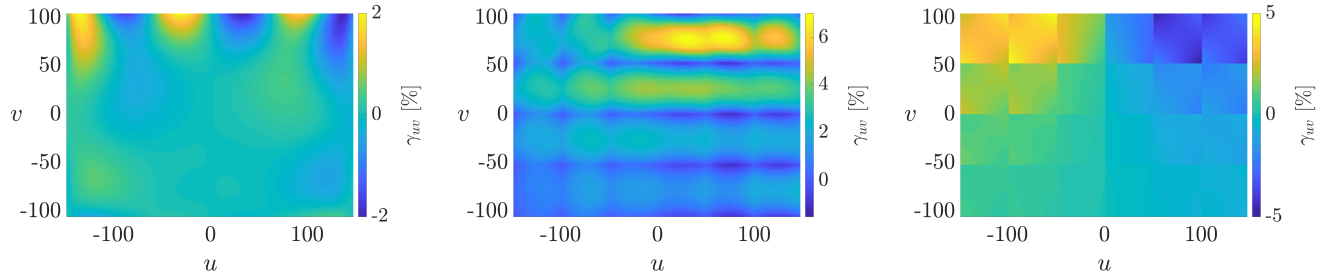


(a) Average normal strain in the v-direction using algorithm presented in this paper.

(b) Average normal strain in the v-direction using algorithm from Saguin-Sprynski et al. [14].

(c) Average normal strain in the v-direction using algorithm from Hermanis et al. [11].

Fig. 19: Average normal strain in the u-direction using IMUs and different algorithms.



(a) Average shear strain using algorithm presented in this paper.

(b) Average shear strain using algorithm from Saguin-Sprynski et al. [14].

(c) Average shear strain using algorithm from Hermanis et al. [11].

Fig. 20: Average shear strain using IMUs and different algorithms.

7.4 Computation Time

Each algorithm was implemented following the explanations given in the respective paper. No particular effort was taken to optimize their speed.

Table 3 shows the time taken by each algorithm to reconstruct a shape. Each shape is used to initialize the next step, to accelerate the process. The method proposed by Hermanis et al. [11] is the fastest by two orders of magnitude as it is based on linear calculations. The method proposed by Saguin-Sprynski et al. [14] is the slowest as it has to solve a non-linear, constrained minimization algorithm to compute the position of 24 out of the 35 sensor locations. The method proposed in this paper involves a non-linear unconstrained minimization, and yields faster results than [14] by a factor of 3.

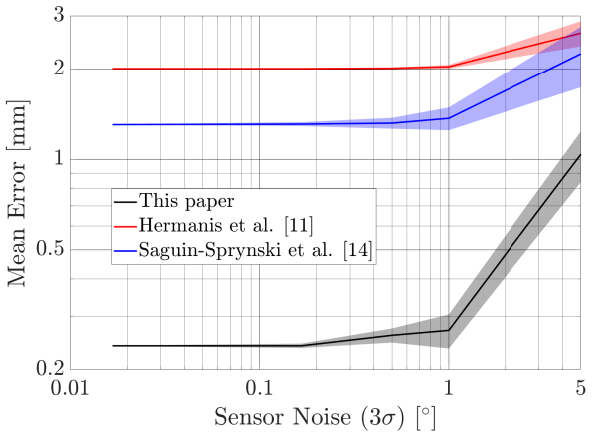


Fig. 21: Mean error of reconstructed shapes using IMUs and different algorithms.

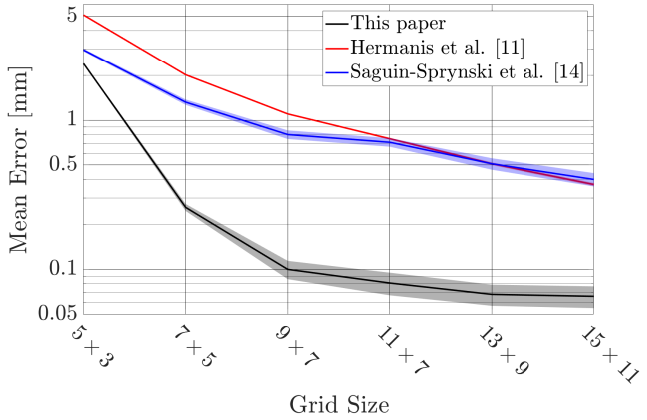


Fig. 22: Variation of the mean error between reconstructed and actual shapes by varying the grids size using IMUs and different algorithms.

Algorithm	Initial reconstruction	Converged time
This paper	800 ms	300 ms
Saguin-Sprynski et al. [14]	20 s	850 ms
Hermanis et al. [11]	1 ms	1 ms

Table 3: Computation time of different reconstruction algorithms using 7×5 IMU sensors.

7.5 Results for Multiple Reconstructions

To further show the net increase of performance of the proposed solution against the current state-of-the-art, each algorithm is used to reconstruct a set of random paper deformations, generated from the algorithm described in [38]. A set of 3 guiding rulings are randomly placed on the sheet of paper. The rotation angles along the rulings are randomly selected withing the $\pm 10°$ interval. This creates shapes with angular deformations greater than 45°. Ten extra rulings are used to create a smooth surface. From this shape generation algorithm, the position of each point on the paper as well as the jacobian of the deformation at each sensor location are retrieved. The latter is used to compute the sensor data which also includes 0.5° noise. A total of 100 shapes are generated and 10 sensors measurements are simulated for each shape yielding 1000 different reconstructions from each algorithm. All parameters used in the previous subsections remain identical. The error of each method is shown in table 4. Similarly to the results gathered from the reconstruction of a cone, the proposed solution performed better on average. Note that the algorithm from Saguin-Sprynski et al. [14] is much less stable and resulted in a higher error overall.

Algorithm	Mean	Standard Deviation
This paper	0.42 mm	0.20 mm
Saguin-Sprynski et al. [14]	2.1 mm	1.3 mm
Hermanis et al. [11]	1.3 mm	0.59 mm

Table 4: Computation time of different reconstruction algorithms using 7×5 IMU sensors.

8 Surface Reconstruction with Real Data and Small Deformations

An experiment was conducted to validate the proposed solution method using real data.

Light sensors were developed using a quad-photodiode (OPR5911) with a square aperture placed on top, effectively creating a 4-pixel pinhole camera [39,40]. A total of 14 sensors were placed on a 50" x 10" (1.27 m $\times$ 0.254 m) aluminum sheet in two rows of 7 separated by 8" (20.32 cm) in all directions. The accuracy of the sensors was 0.5° (3σ).

A LED light source was placed about 2 m in front of the aluminum sheet. A random black and white pattern was painted on the sheet in order to measure its shape with a precise Digital Image Correlation system (DIC) from Correlated Solutions, as shown in figure 23. This system has an accuracy of about 50 μm, an order of magnitude better than the expected accuracy of the light sensor system. As a result, it can be used as a ground truth to estimate the performance of the proposed solution.

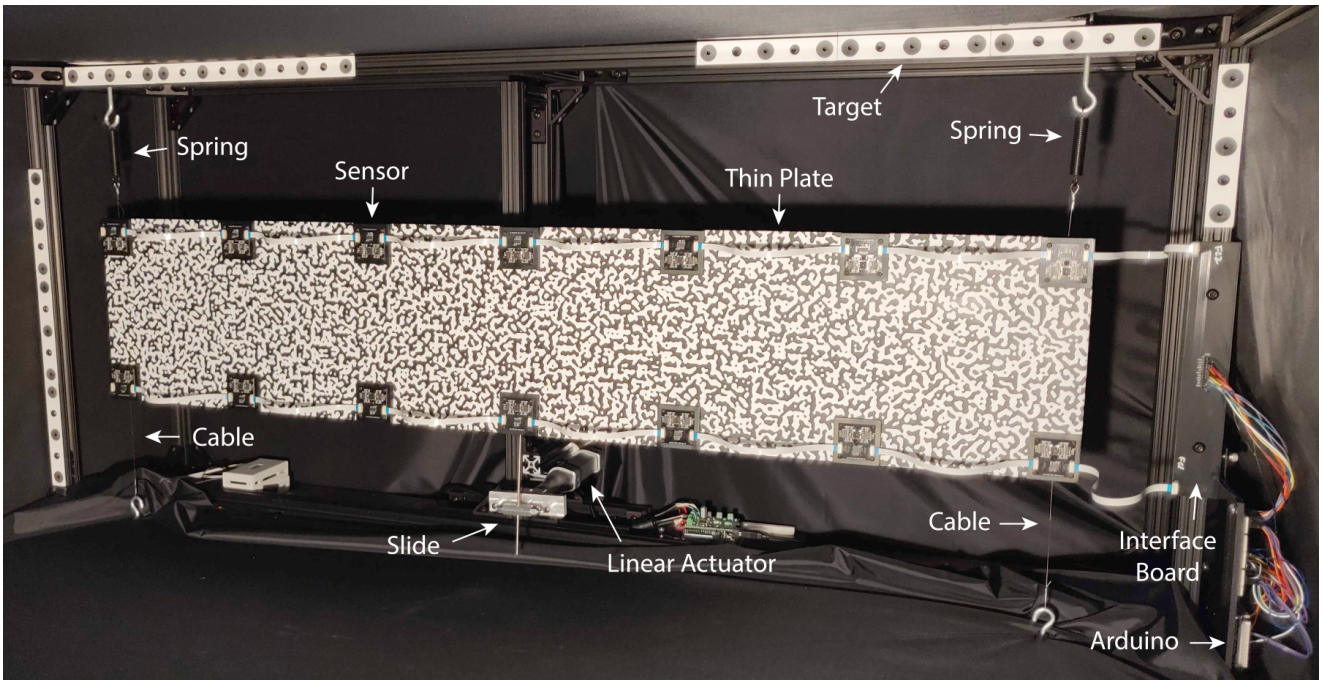
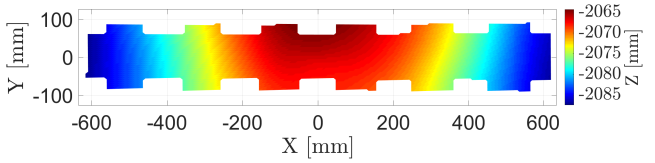
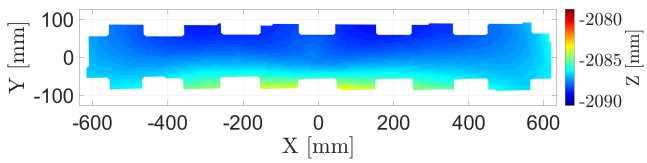


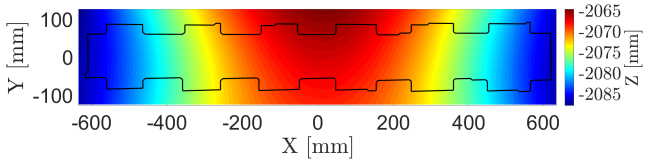
Fig. 23: Photo of the experiment. The plate with a black and white Digital Image Correlation (DIC) pattern holds 14 sensors placed on rigid supports. The plate is held by tensioned spring cables at each end and two linear actuators in the middle (the actuator attached to the center of the structure is not visible).



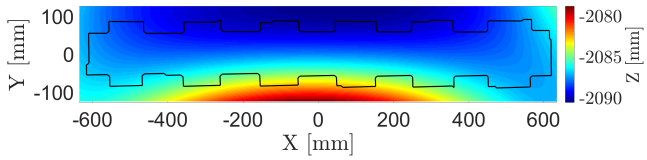
(a) 20 mm deflected shape from DIC



(b) 5° twisted shape from DIC

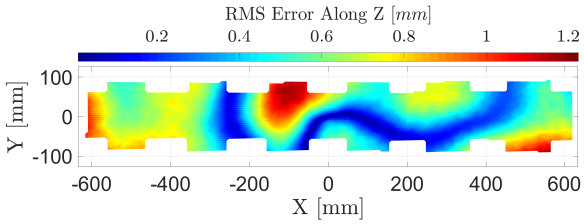


(c) Average reconstructed 20 mm deflected shape

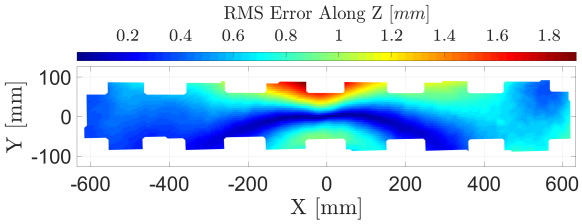


(d) Average reconstructed 5° twisted shape

Fig. 24: Results from the experiments.



(a) RMS error of 20 mm deflected shape



(b) RMS error of 5° twisted shape

Fig. 25: RMS error of the reconstructed shapes

Two different types of deformation are imposed on the structure: a 2 cm deflection and a 5° twist. Figure 24 shows the deformation of the sheet in both cases, as measured by the DIC and as computed by the presented algorithm using the light sensor data. The shape retrieved from the DIC system only images the central part of the sheet as the sensors and cables obstruct the view of the pattern. Our method can however reconstruct the whole surface and figures 24c and 24d show an excellent qualitative agreement between shapes.

The RMS error between the two shapes is shown in figure 25. A total of 1000 sets of measurements were taken in order to generate 1000 reconstructed shapes and thus provide statistically accurate results. The RMS error is lower than 1 mm on average (0.6 mm for both cases) and is localized to regions of the sheet that have more localized deformations. The proposed methodology can best reconstruct relatively smooth surfaces as the errors increase when the deformation becomes more localized. A finite-element based reconstruction algorithm that is more accurate for such cases and also for larger deformations, at the expense of much lengthier solution times, is provided in [40].

Simulations were performed by first computing the shape of the sheet from a mechanical model of the experimental setup using the commercial finite-element software Abaqus. The same algorithm was then used to estimate the accuracy of the sensor system, assuming a zero-mean, 0.5° (3σ) gaussian noise for each sensor. Such simulations predicted a 0.65 mm RMS error for the 2 cm deflection case and 1.2 mm for the 5° twist case. These results are very close to those of the experiment.

9 Conclusion and Discussion

We have presented a mathematical model to reconstruct the shape of a 3D surface based on a template and the angle measurements from embedded sensors. The template is a known configuration of the surface and it is assumed that it deforms inextensibly to its current configuration.

The performance of this method which is suitable for situations where holding a camera in front of a surface is not practical has been tested and validated through simulations and experiment. The formulation yields better results than the state-of-the-art using embedded IMU sensors.

The formulation is similar to SfT-SfS algorithms which also assume inextensibility of the deformation from a template and integrate the shape from the computation of the surface normal. More work could be done to use the explicit measurement of the normal from the embedded sensors together with these algorithms which are usually energy-based.

Aerospace is one of the application areas for this method. Sun sensors are already widely used in this industry for attitude control, and together with this algorithm, could be used to measure the shape of a deployable or reconfigurable structure in space (such as solar sails or large antenna arrays). Wearable technologies used for augmented reality, medical purposes or robotics are other applications of this method using such sensors (usually IMUs).

The choice of shape functions used to define the shape of the surface can also limit the accuracy of the algorithm. More work could be done to study a wider range of functions such as NURBS or dense, mesh-based functions.

Only developable surfaces have been considered in this study. More work should be done to understand the performance of the presented algorithm on more complex shapes such as doubly curved surfaces and surfaces with localized deformations like kinks or buckles.

Acknowledgements Financial support from the Space Solar Power Project at Caltech is gratefully acknowledged.

References

1. Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A. and Silva, C.T., A survey of surface reconstruction from point clouds, In Computer Graphics Forum, Vol. 36, No. 1, pp. 301-329 (2017)
2. Khatamian, A. and Arabnia, H.R., Survey on 3D surface reconstruction, Journal of Information Processing Systems, 12(3) (2016)
3. Brunet, F., Hartley, R., Bartoli, A., Navab, N. and Malgouyres, R., Monocular template-based reconstruction of smooth and inextensible surfaces, Asian Conference on Computer Vision (pp. 52-66). Springer, Berlin, Heidelberg (2010)
4. Perriollat, M., Hartley, R. and Bartoli, A, Monocular template-based reconstruction of inextensible surfaces, International journal of computer vision, 95(2), pp.124-137 (2011)
5. Salzmann, M., Pilet, J., Ilic, S. and Fua, P., Surface deformation models for nonrigid 3D shape recovery, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(8), pp.1481-1487 (2007)
6. Brunet, F., Bartoli A. and Hartley, R.I., Monocular template-based 3D surface reconstruction: Convex inextensible and nonconvex isometric methods, Computer Vision and Image Understanding, 125, pp.138-154 (2014)
7. Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T. and Pizarro, D., Shape-from-template, IEEE transactions on pattern analysis and machine intelligence, 37(10), pp.2099-2118 (2015)
8. Chhatkuli, A., Pizarro, D., Bartoli, A. and Collins, T., A stable analytical framework for isometric shape-from-template by surface integratio,. IEEE transactions on pat-

tern analysis and machine intelligence, 39(5), pp.833-850 (2016)
9. Gallardo, M., Collins, T. and Bartoli, A., Dense non-rigid structure-from-motion and shading with unknown albedos, In Proceedings of the IEEE International Conference on Computer Vision, pp. 3884-3892 (2017)
10. Varol, A., Shaji, A., Salzmann, M. and Fua, P., Monocular 3D reconstruction of locally textured surfaces, IEEE transactions on pattern analysis and machine intelligence, 34(6), pp.1118-1130 (2012)
11. Hermanis, A., Cacurs, R. and Greitans, M., Acceleration and magnetic sensor network for shape sensing, IEEE Sensors Journal, 16(5), pp.1271-1280 (2016)
12. Hoshi, T. and Shinoda, H., 3D shape measuring sheet utilizing gravitational and geomagnetic fields, 2008 SICE Annual Conference, pp. 915-920, IEEE (2008)
13. Huard, M., Sprynski, N., Szafran, N. and Biard, L., Reconstruction of quasi developable surfaces from ribbon curves, Numerical Algorithms, 63(3), pp. 483-506 (2013)
14. Saguin-Sprynski, N., Jouanet, L., Lacolle, B. and Biard, L., Surfaces reconstruction via inertial sensors for monitoring, EWSHM - 7th European Workshop on Structural Health Monitoring, IFFSTTAR, Inria, Université de Nantes, pp.702-709 (2014)
15. Sprynski, N., Lacolle, B. and Biard, L., Motion capture of an animated surface via sensors' ribbons, PECCS 2011-1st International Conference on Pervasive and Embedded Computing and Communication Systems, pp. 421-426, SciTePress (2011)
16. Stanko, T., Hahmann, S., Bonneau, G.P. and Saguin-Sprynski, N., Shape from sensors: Curve networks on surfaces from 3D orientations, Computers & Graphics, 66, pp. 74-84 (2017)
17. Boden, R.C. and Hernando-Ayuso, J., Shape estimation of gossamer structures using distributed sun-angle measurements, Journal of Spacecraft and Rockets, 55(2), pp. 415-426 (2017)
18. Talon, T. and Pellegrino, S., In-space shape measurement of large planar structures, 4th AIAA Spacecraft Structures Conference, AIAA 2017-1116 (2017)
19. Talon, T. and Pellegrino, S., Shape measurement of large structures in space: Experiments, 2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace, pp. 581-584, IEEE (2018)
20. Trebi-Ollennu, A., Huntsberger, T., Cheng, Y., Baumgartner, E.T., Kennedy, B. and Schenker, P., Design and analysis of a sun sensor for planetary rover absolute heading detection. IEEE Transactions on Robotics and Automation, 17(6), pp.939-947 (2001)
21. Salzmann, M., Moreno-Noguer, F., Lepetit, V. and Fua, P., Closed-form solution to non-rigid 3D surface registration, European conference on computer vision (pp. 581-594). Springer, Berlin, Heidelberg (2008)
22. Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M. and Grinspun, E., Efficient simulation of inextensible cloth, ACM Trans. Graph., 26(3), p.49 (2007)
23. Sumner, R.W., Schmid, J. and Pauly, M., Embedded deformation for shape manipulation, In ACM SIGGRAPH 2007 papers, pp. 80 (2007)
24. Sorkine, O. and Alexa, M., As-rigid-as-possible surface modeling, In Symposium on Geometry processing, Vol. 4, pp. 109-116 (2007)
25. Metaxas, D. and Terzopoulos, D., Constrained deformable superquadrics and nonrigid motion tracking, In Proceedings, 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 337-343). IEEE (1991)
26. Torresani, L., Hertzmann, A. and Bregler, C., Learning non-rigid 3D shape from 2D motion, Advances in Neural Information Processing Systems, pp. 1555-1562 (2004)
27. Dierckx, P., Curve and surface fitting with splines. Oxford University Press (1995)
28. Goshtasby, A., Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces. International Journal of Computer 11263on, 10(3), pp.233-256 (1993)
29. Abramowitz, M. and Stegun, I.A., Handbook of mathematical functions: with formulas, graphs, and mathematical tables (Vol. 55), Courier Corporation (1965)
30. Fornberg, B. and Zuev, J., The Runge phenomenon and spatially variable shape parameters in RBF interpolation, Computers & Mathematics with Applications, 54(3), pp.379-398 (2007)
31. Kreyszig, E., Introduction to differential geometry and Riemannian geometry, University of Toronto Press (1968)
32. Ricci, M.M.G. and Levi-Civita, T., Méthodes de calcul différentiel absolu et leurs applications, Mathematische Annalen, 54(1-2), pp.125-201 (1900)
33. Abbena, E., Salamon, S. and Gray, A., Modern differential geometry of curves and surfaces with Mathematica, Chapman and Hall/CRC (2017)
34. Floudas, C.A. and Pardalos, P.M. eds., Encyclopedia of optimization (Vol. 1), Springer Science & Business Media (2001)
35. Malvern, L. E., Introduction to the mechanics of continuous medium, Prentice Hall (1969)
36. Nielson, G.M., ν-Quaternion splines for the smooth interpolation of orientations, IEEE transactions on visualization and computer graphics, 10(2), pp.224-229 (2004)
37. Coons, S.A., Surface patches and B-spline curves, In Computer Aided Geometric Design, Academic Press, pp. 1-16 (1974)
38. Perriollat, M. and Bartoli, A., A Computational Model of Bounded Developable Surfaces with Application to Image-Based Three-Dimensional Reconstruction, Computer Animation and Virtual Worlds, 24(5), pp.459-476 (2013)
39. Talon, T., Surface Reconstruction from Distributed Angle Measurements, PhD diss., California Institute of Technology (2020)
40. Talon, T., Chen, Y. L. and Pellegrino, S. Shape Reconstruction of Planar Flexible Spacecraft Structures Using Distributed Sun Sensors, Acta Astronautica, 180, pp.328-339 (2021).