Learning-Based Non-rigid Video Depth Estimation Using Invariants to Generalized Bas-Relief Transformations

Matteo Pedone, Abdelrahman Mostafa, Janne Heikkilä

Abstract
We present a method to locally reconstruct dense video depth maps of a non-rigidly deformable object directly from a video sequence acquired by a static orthographic camera. The estimation of depth is performed locally on spatiotemporal patches of the video, and then, the full depth video is recovered by combining them together. Since the geometric complexity of a local spatiotemporal patch of a deforming non-rigid object is often simple enough to be faithfully represented with a parametric model, we artificially generate a database of small deforming rectangular meshes rendered with different material properties and light conditions, along with their corresponding depth videos, and use such data to train a convolutional neural network. Since the database images are rendered with an orthographic camera model, linear deformations along the optical axis cannot be recovered from the training images. These are known in the literature as generalized bas-relief (GBR) transformations. We address this ambiguity problem by employing the invariant-theoretic normalization procedure in order to obtain complete invariants with respect to this group of transformations, and use them in the loss function of a neural network. We tested our method on both synthetic and Kinect data and experimentally observed that the reconstruction error is significantly lower than the one obtained using conventional non-rigid structure from motion approaches and state-of-the-art video depth estimation techniques.

Keywords depth video estimation · Invariant · Moving frame · Deep learning · Bas-relief ambiguity

1 Introduction
The human visual system has a remarkable ability of discerning the three-dimensional shape of an observed scene. Although its internal mechanisms are still not entirely clear, it is known that our visual system relies simultaneously on several cues to perceive shape [15,40], such as the local changes in shading and texture of an object, or the temporal change in appearance of an object while seen from different angles. Researchers in the last few decades have attempted with varying degrees of success to formulate mathematical models that would approximately describe such mechanisms in order to emulate them on machines. Some of the most prominent outcomes of these efforts are: algorithms for structure from motion (SfM) [45], monocular depth estimation [39], photometric stereo (PS) [48], shape from shading [21], and shape from texture [5]. Each of these methods has a vast literature of its own. Interesting surveys can be found in [1,30,35,37].

The aforementioned methods have been typically approached under the strong assumption that the observed scene is static. In recent years, we have witnessed an increasing interest in extending such techniques to the case of dynamic scenes made of rigidly and non-rigidly moving objects. However, there are currently still many challenges to be faced due to the ill-posedness of the inverse problems related to the estimation of three-dimensional shape of a dynamic scene from two-dimensional images. Research efforts in this direction have led to the introduction of methods for dynamic photometric stereo [18,23,47], non-rigid structure from motion (NRSfM) [43], and more recently video depth estimation [31].

Although NRSfM methods differ significantly in terms of the inputs they rely on and in terms of the outputs they
produce, compared to dynamic PS or to video depth estimation, we can nonetheless argue that NRSfM algorithms are often preceded by tracking algorithms that in fact rely on information extracted from the pixel values of a video sequence. Given the recent success of deep-learning algorithms in many different settings, it is natural to investigate the feasibility of deep-learning-based approaches that operate directly on a video sequence acquired from a static camera as its input, and that would automatically yield a dense representation of the scene depth for each frame as its output. This has potentially the advantage of overcoming the frequently imposed assumptions on the illumination conditions of the scene and on the reflectance properties of the observed object (as in PS), and that of avoiding to explicitly solve ill-posed geometric problems, as in NRSfM.

The main challenge is clearly represented by the difficulty of acquiring large scale training data in which each sample would consist of a video sequence of an object (or a portion of it), along with its corresponding 3D representation in space. Although nowadays it is theoretically possible to acquire RGB video sequences along with their corresponding depth maps using inexpensive equipment, producing a sufficiently large database involving non-rigid motion of many different subjects, materials, textures, and light conditions would be a daunting and time consuming process.

In this paper, we try to address this problem by showing that it is possible to train a convolutional neural network to perform the direct estimation of a depth video from the pixel intensities of a video sequence by generating a database consisting of short, rendered, synthetic video sequences, each depicting the movement of a small patch of a non-rigidly deforming object, along with the corresponding depth video (Fig. 1). Potential applications can be found in industry, computer graphics, and safety systems, where the method could be used, respectively, for folds/wrinkles detection and analysis in automated garment manufacturing processes, acquisition of dynamic wrinkle maps for realistic rendering of animated human/animal skin, and sea state monitoring with cameras mounted on buoys, saildrones, boats, or naval ships.

The contributions of this paper are the following ones:

- We propose a network architecture to estimate the depth map for each frame of a video sequence of a deforming object acquired by a static orthographic camera, as well as a computationally fast mathematical model to synthetically generate the training data.
- Since the training samples are often ambiguous with respect to certain linear transformations along the optical axis, known as generalized bas-relief (GBR) transformations, we derive two novel sets of complete invariants for this particular subgroup of affine transformations: point-cloud GBR-invariants and differential GBR-invariants. We use GBR-invariants to formulate a loss function in the training stage.
- We utilize the theory of the moving frame to obtain GBR-invariants in a relatively simple and constructive manner, therefore following the opposite strategy often found in the computer vision literature, in which unmotivated formulas are first introduced, and then proven to satisfy an invariance property.
– We obtain better performance than common state-of-the-art non-rigid structure from motion methods when reconstructing depth videos of deforming objects that are representable as surfaces without holes.

The rest of this paper is organized as follows. In Sect. 2, we give a brief overview of related work in the literature. Section 3 describes the proposed method from a general point of view and introduces the main assumptions that are invoked in the next sections. Section 4 describes the ambiguity scenario that emerges in our application, where each training sample may correspond to an infinitely large class of ground truths. In Sects. 5 and 6, we provide the mathematical background to describe such problem, and in Sects. 7 and 8, we propose a strategy to address it. In Sect. 9, we describe in details the generation of our database of spatiotemporal patches with their corresponding ground truth depth video. The architecture of the convolutional neural network (CNN) utilized for 3D shape estimation is described in Sect. 10. Section 11 contains implementation details related to the estimation of the depth video. Sections 12 and 13, respectively, show the experimental results and give concluding remarks.

2 Related Work

We now give an overview of some of the works in the literature that can be considered related to ours. In SfM, it is assumed that the features in the image plane of some interest points are acquired from several points of view, and under the assumption of a static scene, the geometric relationships between corresponding points can be utilized to recover the original 3D structure of the scene, as well as the camera pose. The problem of estimation of non-rigid structure from motion (NRSfM) considers the case where the scene itself can move non-rigidly; hence, the geometric relationships that are utilized in ordinary SfM, where the scene is static, are more difficult to exploit.

Researchers have introduced priors on the camera model [7], on the type of trajectories of points in space-time [3], or on the shape of the object [14,26,28,44,49] in order to make the problem mathematically tractable. Algorithms for NRSfM mainly differ from each other in terms of the prior they impose on the shape and motion of the objects. However, most current methods rely on the assumption that the scene is observed through an orthographic camera (see Chapter 5 of [38] for some exceptions using the perspective camera model), and they are essentially generalizations of the popular factorization method, which was originally employed for traditional orthographic SfM [43].

Agudo et al. [2] show that it is feasible to solve the NRSfM problem from video frames of an orthographic monocular camera if one assumes that the non-rigid motion can be accurately described by the equations governing the motion of elastic bodies. After determining the 3D locations of the tracked points, the authors propose a scheme to construct a triangular mesh approximating the global shape of the surface. An extensive comparison between NRSfM methods can be found in [22].

In photometric stereo, one or more images of the same subject need to be acquired under different illuminations in order to estimate its 3D shape, and some authors have addressed the problem of moving objects by using specialized hardware [18,23,47]. In addition to that, the physical surface of the observed object is assumed to have certain reflectance properties (usually Lambertian).

In monocular depth estimation, one solely relies on the information carried by a single image in order to extract the depth at each pixel, which is generally insufficient to discern far pixels from near pixels; therefore, such methods typically rely on statistics about the appearance of the image at several scales and on training data [39].

Attempts at generalizing these techniques to the case of spatiotemporal data resulted in some promising works. Kumar et al. [27] recently proposed a method based on superpixels and motion constraints in order to produce animated depth maps for a video sequence of moving subjects. In contrast with other recent approaches in the same category, their method does not rely on machine learning; moreover, the estimated depth maps are always piecewise planar within the superpixel boundaries, affecting the accuracy of the reconstruction.

In [46], a machine learning system is presented that uses pairs of consecutive frames of a video sequence to produce an accurate perspective depth map and camera position. The method works on static scenes and therefore is not able to handle the more general problem of NRSfM.

In [25], the authors show a deep-learning-based approach to perform NRSfM from 2D hand-annotated landmarks and obtain good performance on articulated motion and when recovering the different structures of rigid and static objects of the same type photographed from different perspectives.

In [31], the authors propose a deep learning based technique that is capable of obtaining good quality video depths from challenging scenes taken with a handheld camera. Their method is able to handle a moderate amount of non-rigid motion in the scene.

Despite the promising results of existing methods, to the best of our knowledge, there is still a lack of available methods in the literature explicitly targeted at recovering the three-dimensional geometry of a non-rigidly deforming object directly from a video sequence.
3 Overview of the Proposed Method

We consider the problem of reconstructing the spatiotemporal depth map (depth video) from the video sequence of a non-rigidly moving object seen from a static orthographic camera. Our approach relies on the following assumptions:

1. The scene/object is observed by a static orthographic camera.
2. The deformation of the object observable within fixed spatiotemporal windows of the video sequence is non-negligible.
3. Locally, the 4D structure of the object (i.e., its 3D shape evolving in time) can be approximated with a parametric model controlled by a relatively small amount of parameters.

The orthographic camera assumption is justified in those scenarios where the field of view is narrow and the observed object is sufficiently far from the camera [29]. Note that these assumptions, despite being restrictive, are analogous to the assumptions made by current state-of-the-art NRSfM methods [28]. The proposed method essentially consists in estimating the depth video locally, within local “patches” in the space-time domain, and then in reconstructing the entire depth map of the scene by “stitching” together the local depth videos. The estimation is done by training a neural network to learn the animated depth map within small spatiotemporal patches of a video that have size $64 \times 64 \times 16$ pixels in our implementation. Given the practical difficulty of acquiring a large amount of real data consisting of small video clips of deforming subjects, along with their corresponding depth videos, we generate a database of synthetic data.

The data consists of deforming planar meshes rendered with variable textures, reflectance properties, and light conditions, and we obtain their respective depth videos by setting the gray intensity of each pixel to the corresponding value of the $z$-coordinate of the surface in camera coordinates. The database is utilized to train a CNN to directly estimate $64 \times 64 \times 16$ depth videos from RGB video sequences of the same size. In the final step, the depth videos of each patch are combined together in order to reconstruct the entire depth video. The reconstruction scheme that we utilize is inspired by the well-known method of signal reconstruction using the constant overlap-add (COLA) decomposition. Since the training data does not contain videos in which the surface presents holes, our method is mainly suitable to recover surfaces with genus 0. Note also that the assumption of orthographic camera removes the need to train the network with different camera parameters; however, it introduces also an ambiguity problem where same video sequences may correspond to different depth maps. This issue is addressed in detail in Sect. 4.

4 Ambiguity-invariant Representation of Depth Maps

Adopting an orthographic camera model has the advantage of avoiding the need to train the network with different camera parameters; however, scenes rendered with orthographic projection are prone to ambiguities. In the particular case of Lambertian surfaces illuminated by directional light sources, the resulting ambiguity is the generalized bas-relief ambiguity (GBR) [6], that has been studied extensively in the context of shape from shading, and which is essentially a composition of shears and stretches along the optical axis $z$. Although the rendered surfaces in our database are not always perfectly Lambertian, the reflectance model adopted in the rendering stage is not sufficient to resolve entirely the ambiguity [11], even in presence of texture and specular highlights. Therefore, any rendered scene in our database is virtually indistinguishable from all the other hypothetical rendered versions undergoing arbitrary GBR transformations (Fig. 2).

This issue would surely impair the training stage; hence, it is crucial to find a representation of smooth depth map surface that is invariant to the group of GBR transformations, which form exactly the subgroup of affine transformations that lead to ambiguities in the orthographic camera images.

Once such an invariant representation is obtained, it is possible to utilize it in the loss function of the training stage of the network. This has the big advantage of letting the network treat all the GBR-transformed versions of one depth map as an entire equivalence class: in fact, the invariants of two depth maps related by a GBR transformation will be exactly the same; hence, the loss function will yield always zero in such cases.

In the following sections, we provide a self-contained explanation on how to derive GBR-invariant representations for depth maps. Since there is no unique way of defining this representation, we consider two main types of invariants that are calculated by, respectively, treating depth maps as...
point clouds, and as smooth 2D functions in 3D space. Finding complete (i.e., maximally discriminative) invariants for a given transformation are more than often a non-trivial mathematical problem. In order to simplify considerably this step, we employ the normalization construction, which is a well-known technique in the mathematical literature of invariant theory [33] aimed at deriving algorithmically the invariants with respect to a given transformation. The material presented in Sects. 5 and 6 is a brief review of selected topics in the mathematical literature of invariant theory, needed for the scope of this paper. Readers who are familiar with the theory can skip directly to Sects. 7 and 8 that contain novel material, and in which we give the formulation of the invariants used in this paper for our applications (Theorems 1-2).

5 Brief Review of Group Actions

A group is a set $G$ equipped with an associative binary operation between elements of $G$, and such that an identity element and inverse elements exist for each $g \in G$. The types of groups we are concerned with in this article are mainly groups where the elements are matrices that represent linear geometric transformations, and where the binary operation is given by the ordinary matrix multiplication. To be more precise, we will deal with certain types of Lie groups, that are essentially groups with a manifold structure, and therefore, their elements can be uniquely identified through a smooth parametrization. A typical basic example of a Lie group representation is the group of $2 \times 2$ rotation matrices parametrized by the angle $\theta$. Groups can act on a given set $X$ by producing transformed versions of the elements of $X$.

Definition 1 A group action of $G$ on a set $X$ is a map $\ast: G \times X \to X$ satisfying the following properties:

1. (identity) $1 \ast x = x$
2. (associativity) $g_1 \ast (g_2 \ast x) = (g_1g_2) \ast x$,

for all $x \in X$ and $g_1, g_2 \in G$, where the juxtaposition $g_1g_2$ denotes the group operation.

A concrete example is that of the group of $2 \times 2$ rotation matrices acting on points $x \in \mathbb{R}^2$, given by $R \ast x = Rx$, where $x$ is assumed to be a column vector. Such action produces new rotated versions of $x$ that lie on a circle centered at the origin and passing through $x$. In general, given an element $x \in X$, the subset $Gx = \{y \in X : y = g \ast x\}$ is called orbit of $x$ under the action of $G$. Given a group action on $\mathbb{R}^3$, we can always define three important types of induced actions: the induced action on point clouds, the induced action on smooth functions, and the prolonged action on smooth functions.

Definition 2 Given a group action of $G$ on $\mathbb{R}^3$, the induced action on point clouds is the map $\ast: G \times (\mathbb{R}^3 \times \ldots \times \mathbb{R}^3) \to \mathbb{R}^3 \times \ldots \times \mathbb{R}^3$ defined as:

$$g \ast (p_1, \ldots, p_N) := (g \ast p_1, \ldots, g \ast p_N).$$

where $p_1, \ldots, p_N \in \mathbb{R}^3$ and $N$ is the number of points.

Definition 3 Given a group action of $G$ on $\mathbb{R}^3$, the induced action on smooth functions is the map $\ast: G \times C^\infty(\mathbb{R}^2, \mathbb{R}^3) \to C^\infty(\mathbb{R}^2, \mathbb{R}^3)$ defined as:

$$(g \ast f)(x) := f(g \ast x).$$

Before introducing the next type of induced action, it is important to recall that the $n$-th order jet (prolongation) of a smooth function $f$ at a point $x$ is the vector $f^{[n]}(x) = [a^\alpha f(x)]_{|\alpha|=0,\ldots,n}$ whose entries are given by all the partial derivatives of $f$ with multi-indices $\alpha \in \mathbb{N}_0^n$ up to order $n$, evaluated at $x$. Note that the $n$-th order prolongation of a function at a certain point is therefore an element of a space denoted as $J^n$, whose dimension is higher than that of the codomain of $f$ for $n > 1$.

Definition 4 Given a group action of $G$ on $\mathbb{R}^3$, the $n$-th order prolonged action on smooth functions is the map $\ast: G \times J^n(\mathbb{R}^2, \mathbb{R}^3) \to J^n(\mathbb{R}^2, \mathbb{R}^3)$ defined as:

$$(g \ast f^{[n]})(x) := \left[a^\alpha (g \ast f)(x)\right]_{|\alpha|=0,\ldots,n}.$$  

Whenever one considers the action of a generic element $g \in G$, it is often unnecessary to explicitly include $g$ in the formulas; hence, it is common practice to adopt the notation $\tilde{x} = g \ast x$.

6 The Moving Frame and the Normalization Procedure

In this section, we give a brief review of the theory of the moving frame and the normalization procedure, which are powerful tools by which invariant quantities can be obtained in a constructive manner.

When a group action on a set is considered, it is often useful to derive an invariant operator with respect to that action. This is essentially equivalent to finding an operator that maps each element of the set to its corresponding orbit. Finding such an operator is usually not straightforward; therefore, we now introduce to the reader some well-known techniques that will allow us to derive complete invariants in a constructive manner. One key observation is that it is often easier to find an anti-equivarient map with respect to the group action than directly finding an invariant map. The invariant can then be
constructed easily from the knowledge of the anti-equivariant map. We clarify this fact in details in the next lines.

**Definition 5** *(moving frame)* Given a group action of $G$ on a set $X$, a map $\rho : X \rightarrow G$ is called *moving frame* if it is anti-equivariant with respect to the action, i.e., if it satisfies the following property:

$$\rho(g \ast x) = \rho(x)g^{-1}. \quad (4)$$

The term *moving frame* originates from the work of the French mathematician Elie Cartan [9,10] who developed the theory underlying the normalization technique in the context of differential geometry. Later, his theory and techniques were given a rigorous foundation and were extended to a more abstract setting by Fels and Olver [16,17], although the original term used by Cartan “repére mobile” (i.e., “moving frame”) survived also in contemporary mathematical literature. Assuming one has found a moving frame for a given action, it is straightforward to find a corresponding invariant operator.

**Lemma 1** *(invariantization)* Given a moving frame $\rho : X \rightarrow G$ for the action of $G$ on $X$, the operator $\iota : X \rightarrow X$ defined as follows:

$$\iota(x) := \rho(x) \ast x \quad (5)$$

is invariant with respect to the group action, i.e., $\iota(g \ast x) = \iota(x)$ for all $g \in G$, and (5) is called invariantization formula.

**Proof** The proof of the above statement follows easily from the identities below:

$$\begin{align*}
\iota(g \ast x) &= \rho(g \ast x) \ast (g \ast x) \quad \text{(by Eq.}(5)) \\
&= (\rho(x)g^{-1}) \ast (g \ast x) \quad \text{(by Eq.}(4)) \\
&= (\rho(x)g^{-1})g \ast x \quad \text{(groupactiondefinition)} \\
&= \rho(x) \ast x \quad \text{(groupoperation)} \\
&= \iota(x) \quad \text{(by Eq.}(5)).
\end{align*}$$

□

Now that we can constructively derive invariants from the knowledge of the moving frame, it is natural to ask if it is in turn possible to obtain the moving frame constructively. This can be often accomplished by using the normalization procedure, which consists in defining a system of $k$ equations where $k$ is the number of parameters of the Lie group in question:

$$\psi_i(\vec{x}) = 0 \quad ; \quad i = 1, \ldots, k, \quad (6)$$

and solving for the group parameter $g$. Geometrically speaking, the normalization equations define a *cross section* for the action of $G$, which is essentially an implicit surface that intersects all the $G$–orbits of $X$. Since the equations are expressed in terms of $g$ and $x$, solving for the variable $g$ yields the group element that “projects” any $x \in X$ onto its corresponding point on the cross section (Fig. 3). Since the choice of the cross section is arbitrary, there is typically no unique way of defining the normalization equations. One guiding principle is that of defining a set of equations $\psi_i$ that makes the calculations as simple as possible. However, defining the normalization equations and finding the moving frame is typically easier than trying to find an invariant operator by trial and error and proving its completeness.

As a toy example, one can consider the action $\vec{p} = R_0 \vec{p}$ of the $2 \times 2$ rotation matrices on the points in $\mathbb{R}^2$. Since the group of rotation matrices is parametrized only by $\theta$, it is sufficient to define only one normalization equation. One sensible choice is: $\vec{p}_x = 0$ that yields $p_x \cos \theta - p_y \sin \theta = 0$, and whose solution is given by $\theta = \arctan_2(p_x, p_y)$, which represents the explicit form for the moving frame $\rho(\vec{p})$. As an exercise, the reader can verify that by the invariantization equation we obtain $\iota(\vec{p}) = R_{\arctan_2(p_x, p_y)} \vec{p} = \left( 0 \sqrt{p_x^2 + p_y^2} \right)^T$, which is a complete invariant to rotation, as expected. In the following sections, we utilize the normalization techniques to derive complete invariants for the action of the group of GBR transformations on point clouds and on smooth 2D functions in $\mathbb{R}^3$. Since it is outside of the scope of this paper to give an exhaustive overview on the theory of Lie group actions and the moving frame, the interested reader can refer to [32] for an accessible introduction to the subject, and to [33,34] for a more detailed treatment of the subject.
7 GBR-invariants for Point Clouds

Let us consider the Lie group of GBR transformations parametrized by \((\alpha, \beta, \lambda, \tau)\), where \(\alpha, \beta \in \mathbb{R}\) are the two parameters for shearing along the \(z\)-axis, \(\lambda \in \mathbb{R}^+\) is the parameter for scaling (stretching) along the \(z\)-axis, and \(\tau \in \mathbb{R}\) is the parameter for \(z\)-translation. Consider a point cloud with \(N\) points \((\mathbf{p}_1, \ldots, \mathbf{p}_N) \in \mathbb{R}^{3 \times N}\) and the action of the GBR transformations on \(\mathbb{R}^{3 \times N}\). For convenience of notation, we will denote a given point cloud in the following three equivalent ways:

\[
P = \begin{pmatrix} X^T \\ Y^T \\ Z^T \end{pmatrix} = (\mathbf{p}_1, \ldots, \mathbf{p}_N). \tag{7}
\]

A GBR-transformed point cloud will then be given by the concatenation of \(z\)-translation, \(z\)-shear, and \(z\)-scaling:

\[
\tilde{P} = \begin{pmatrix} 1 & 1 & \lambda \\ 1 & 1 & \alpha \\ \alpha & 1 & 1 \end{pmatrix} \left( P + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right)^T, \tag{8}
\]

where \(1\) denotes the \(N \times 1\) column-vector of 1’s. From Equation (8), it is possible to verify that the GBR group is equivalent to the (semidirect) product of the three aforementioned subgroups of transformations. In such cases, it is possible to use a result from [24] that essentially states that a complete invariant for the action of a semidirect product is given composing the invariants for each subgroup action. Since it is well known that complete invariants with respect to translation and scaling can be easily obtained by, respectively, translating the \(z\)-centroid of the point cloud to 0, and by normalizing the standard deviation of the \(z\)-coordinates of the point cloud, the original problem now reduces to that of finding a point cloud invariant for the action of the subgroup of \(z\)-shears parametrized by \((\alpha, \beta)\). Let us consider a \(z\)-sheared point cloud:

\[
\tilde{P}_{\text{shear}} = \begin{pmatrix} \tilde{X}^T \\ \tilde{Y}^T \\ \tilde{Z}^T \end{pmatrix} = \begin{pmatrix} 1 & 1 & \alpha \\ 1 & 1 & \beta \\ \alpha & 1 & 1 \end{pmatrix} \begin{pmatrix} X^T \\ Y^T \\ Z^T \end{pmatrix} = \begin{pmatrix} X^T + \beta Y^T + Z^T \end{pmatrix}. \tag{9}
\]

We now employ the normalization procedure described in Sect. 6 and define a cross section with the following normalization (linear) equations:

\[
\begin{align*}
\sigma_{\tilde{X}, \tilde{Z}} &= 0 \\
\sigma_{\tilde{Y}, \tilde{Z}} &= 0,
\end{align*} \tag{10}
\]

where \(\sigma_{., .}\) stands for the sample covariance operator. One might argue that it would have been possible to replace the two normalization equations in (10) with \(\tilde{z}_1 = \tilde{z}_2 = 0\), which is considerably simpler than our current formulation involving cross-covariances; however, note that such a system of normalization equations would lead to solutions which only contain information about two arbitrarily chosen points of the point cloud. Clearly this would have an impact in terms of robustness, and for this reason we opted for two normalization equations in which all the points of the point cloud appear. By plugging the corresponding terms in (9) into \(\tilde{X}, \tilde{Y}, \tilde{Z}\), and solving for \((\alpha, \beta)\), one obtains:

\[
\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = -\Sigma^{-1}_{\tilde{X}, \tilde{Y}} (\sigma_{\tilde{X}, \tilde{Z}} \sigma_{\tilde{Y}, \tilde{Z}}), \tag{11}
\]

where \(\Sigma_{\tilde{X}, \tilde{Y}}\) stands for the covariance matrix of \(\tilde{X}\) and \(\tilde{Y}\). We can plug the right term of Equation (11) into the invariantization formula (5) and obtain:

\[
\iota_{\text{shear}}(P) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_{\tilde{X}, \tilde{Z}} \sigma_{\tilde{Y}, \tilde{Z}} \end{pmatrix} \text{diag}(\Sigma_{\tilde{X}, \tilde{Y}}^{-1}) P. \tag{12}
\]

Lemma 1 guarantees that \(\iota_{\text{shear}}(\tilde{P}_{\text{shear}}) = \iota_{\text{shear}}(P)\); thus, the point cloud \(\iota_{\text{shear}}(P)\) is shear-invariant. In order to obtain the GBR-invariant representation of the point cloud, it is sufficient to apply (12) to the \(z\)-centered version of \(P\) and then standardize the result across \(z\). This is stated more precisely in the following theorem:

**Theorem 1** Given a 3D point cloud \(P \in \mathbb{R}^{3 \times N}\) with \(N\) points, the resulting point cloud obtained by:

\[
\iota(P) = \iota_{\text{scale}} \circ \iota_{\text{shear}} \circ \iota_{\text{trans}}(P), \tag{13}
\]

where:

\[
\iota_{\text{trans}}(P) = P - \begin{pmatrix} 0 \\ 0 \\ \frac{1}{N} 1^T Z \end{pmatrix}, \tag{14}
\]

\[
\iota_{\text{scale}}(P) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \Sigma_{\tilde{Z}}^{-1} P \tag{15}
\]

and \(\iota_{\text{shear}}\) is defined as in Equation (12), is invariant to the action of the group of GBR transformations.
Fig. 4 Examples of point cloud invariants and differential invariants. (Top row) From left to right, the original point cloud, one GBR-transformed version of it, and a transformed version perturbed by noise. (Middle row) The corresponding point cloud invariants obtained by (13). Note that the invariant point cloud obtained from the noisy data appears multiplicatively biased (squeezed) along $z$, compared to its non-noisy version, due to an over-estimation of the scaling parameter in the moving frame equations. (Bottom row) The corresponding differential invariants obtained by (23) and visualized as normalized-Hessian maps, in which the three components $\begin{bmatrix} z_{xx} & z_{yy} & z_{xy} \end{bmatrix} \| \nabla^2 z \|^{-1}_F$ were, respectively, assigned to the $L^*$, $a^*$, $b^*$ channels of the CIE $L^*, a^*, b^*$ color space.

Note that the quantity $\iota(P)$ is singular whenever the determinant of the covariance matrix $\Sigma_{X,Y}$ vanishes, or when $\sigma_Z = 0$. Note, however, that if the point cloud is an arbitrary depth map scattered on a regular grid, $\Sigma_{X,Y}$ is always constant and non-singular, thus the only degenerate case happens when the point cloud $P$ is entirely scattered across a plane perpendicular to the $z$-axis. For this reason, when $\sigma_Z \approx 0$ small perturbations in $P$ can make the computation of the normalized point cloud $\iota(P)$ unstable. Note also that $\iota$ implicitly calculates four global parameters (the moving frame $\rho$) which are then applied to all points in order to project $P$ on the cross section. If the point cloud $P$ is perturbed, then all the moving frame estimates for $\alpha, \beta, \lambda, \theta$ may contain noise and the point cloud does not get projected exactly on the cross section. From Equations (12)-(15), we can deduce that $\iota$ might be affected by additive and/or multiplicative bias (see Fig. 4). For this reason, in the next section, we consider also another type of invariants (differential invariants) in which...
the moving frames are defined locally and the invariant operators are calculated for each location of the surface.

8 GBR-invariants for Smooth 2D Functions

Let us consider the space $S \subset C^\infty(\mathbb{R}^2, \mathbb{R}^3)$ of smooth functions (spatial patches) $f : \mathbb{R}^2 \to \mathbb{R}^3$ defined as follows, as well as $\tilde{f}$, the GBR-transformed version of $f$:

$$f(x, y) = \begin{pmatrix} x \\ y \\ z(x, y) \end{pmatrix}$$

$$\tilde{f} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \alpha x + \beta y + \lambda z + \tau \end{pmatrix}$$ (16)

where $z \in C^\infty(\mathbb{R}^2)$ is the corresponding depth map, $\alpha, \beta, \tau \in \mathbb{R}$, and $\lambda \in \mathbb{R}^+$. We use the normalization construction described in Sect. 6 when considering the prolonged action of the GBR transformations on $f^{[n]}$, the $n-$th order jets of $f$. Note that to alleviate the notation, we will write partial derivatives with the subscript notation, e.g., $f_{xy} = \partial^3 f$.

The prolonged action on the $n-$jets of $f$ is thus given by:

$$\tilde{f}^{[n]} = (\tilde{f}^{[n]} f_{x}^{[n]} f_{y}^{[n]} f_{xx}^{[n]} f_{xy}^{[n]} f_{yy}^{[n]} \cdots) \in J^n(\mathbb{R}^2, \mathbb{R}^3)$$ (17)

where the vector in the right term contains partial derivatives up to the $n-$th order. By plugging the definition of $\tilde{f}$ into (17), we obtain:

$$\tilde{f}^{[n]} = \begin{pmatrix} \tilde{x} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \cdots \\ \tilde{y} \\ \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{z}_{xx} \\ \tilde{z}_{xy} \end{pmatrix}.$$ (18)

Since the GBR group is parameterized by four parameters, we consider the following system of four normalization equations:

$$\begin{cases}
\tilde{z} = 0 \\
\tilde{z}_x = 0 \\
\tilde{z}_y = 0 \\
\tilde{z}_{xx} + \tilde{z}_{xy} + \tilde{z}_{yy} + \tilde{z}_{xy} + \tilde{z}_{yy} = 1.
\end{cases}$$ (19)

The reader could argue that the normalization equation $\tilde{z}_{xx} = 1$ would have led to simpler calculations; however, including the remaining terms of the Hessian matrix both increases the robustness and avoids the degenerate cases in which only $\tilde{z}_{xx} = 0$. By plugging the explicit formulas for the partial derivatives in (18) into (19) yields:

$$\begin{cases}
\alpha x + \beta y + \lambda z + \tau = 0 \\
\alpha + \lambda z_{xx} = 0 \\
\beta + \lambda z_{xy} = 0 \\
(\lambda z_{xx})^2 + (\lambda z_{xy})^2 + (\lambda z_{yy})^2 + (\lambda z_{xy})^2 = 1
\end{cases}$$

and solving for the parameters $\alpha, \beta, \lambda, \tau$, one obtains the moving frame:

$$\begin{align*}
\alpha &= -\tilde{z}_x \\
\beta &= -\tilde{z}_y \\
\lambda &= 1/\sqrt{\tilde{z}_{xx}^2 + \tilde{z}_{xy}^2 + \tilde{z}_{yy}^2} \\
\tau &= -\left(\alpha \beta \lambda\right)f.
\end{align*}$$ (20)

By the invariantization lemma, plugging the parameter values of the moving frame into (18) yields a differential invariant for the GBR-action:

$$I_f = \begin{pmatrix} 0 & 0 & 0 & \tilde{z}_{xx} & \tilde{z}_{xy} & \tilde{z}_{yy} & \cdots \end{pmatrix} \sqrt{\tilde{z}_{xx}^2 + \tilde{z}_{xy}^2 + \tilde{z}_{yy}^2}.$$ (21)

It is possible to verify that the entries of $I_f$ that contain derivatives of degree higher than 2 are obtainable from the first three nonzero invariants in (21). Furthermore, we recognize $\tilde{z}_{xx}, \tilde{z}_{xy}, \tilde{z}_{yx}, \tilde{z}_{yy}$ as the entries of the $2 \times 2$ Hessian matrix of $\tilde{z}$; therefore, we can conclude that a complete differential invariant for the GBR-action on $S$ is given by:

$$I_f = \frac{1}{\sqrt{\tilde{z}_{xx}^2 + \tilde{z}_{xy}^2 + \tilde{z}_{yy}^2}} \begin{pmatrix} \tilde{z}_{xx} & \tilde{z}_{xy} & \tilde{z}_{yx} & \tilde{z}_{yy} \end{pmatrix}.$$ (22)

We can now make the following statement.

**Theorem 2** Given a smooth function $f \in S$ as defined in (16), the normalized Hessian

$$\iota_f := \frac{\nabla^2 z}{\|\nabla^2 z\|_F},$$ (23)

where $\|\cdot\|_F$ denotes Frobenius norm, is a complete differential invariant for $f$ with respect to the action of the group of GBR transformations.

Note that $\iota_f(x, y)$ has singularities only in those point where all the second derivatives vanish. The only surfaces for which $\iota_f$ is singular at every point are the planar ones. In our implementation, where the domain of $z$ is discretized, we estimate the partial derivatives of $z$ in (23) with second derivative filters. It is interesting to note that if $z$ is locally perturbed in the neighborhood of a point $(x, y)$, then only the corresponding invariants $\iota_f(x, y)$ at those locations that are spatially close to $(x, y)$ will be affected by the perturbation (see Fig. 4).
9 Dataset Generation

In this section, we give a detailed explanation on how we generate a database of video sequences to train a neural network in estimating depth information. Consider a deforming surface in space-time $f : (-1, 1)^2 \times (0, 1) \rightarrow \mathbb{R}^3$ having the following form:

$$f(x, t) = E_t (x + d(x, t)),$$  \hspace{1cm} (24)

where the term $E_t$ represents Euclidean transformations, i.e., roto-translations, smoothly parameterized by $t$, whose effect is that of changing the orientation and position of the coordinate frame across time, and the quantity inside brackets in (24) can be seen as a 2D surface in $\mathbb{R}^3$ obtained by applying a displacement $d$ to each point of the bi-unit square $(-1, 1)^2$ (Fig. 5). Note that the function $f$ can be interpreted as an homotopy (continuous deformation) between the surfaces $f(\cdot, 0)$ and $f(\cdot, 1)$.

In order to define the vector displacement function $d$, we utilize an approach inspired by some ideas found in the literature of computer graphics for simulating in real-time the appearance of a planar surface with given physical properties, animated by means of realistic cloth simulation. We stress that in our application, it is not strictly necessary to estimate depth information. Consider a deforming surface by the map $x + d(x, t)$ which is then re-oriented in space through a rigid transformation $E_t$, resulting in the green surface in Fig. 5. The vector displacement $d(\cdot, t)$ is depicted $\kappa$ as a color texture whose red, green, and blue components correspond, respectively, to the $x$, $y$, $z$ components of the displacement (middle gray denotes zero displacement). The appearance of the vector displacement texture varies smoothly in time thanks to the effect of the term $\psi$ in (25)-(26) (flexibility), respectively, control the overall strength of the displacement, how much the square patch remains anchored at its border, how much it tends to create folds, and how much it behaves like a soft or hard surface. The matrix $\Sigma$ controls the distribution of orientations of the displacements, while $\phi, \theta : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ are functions that determine the speed by which each frequency component changes in time and their initial phase angle (Fig. 6). In our implementation, $\phi$ and $\theta$ are initialized randomly.

Our approach of defining a moving surface by means of (26) can be seen as a computationally cheap way to emulate the appearance of a planar surface with given physical properties, animated by means of realistic cloth simulation. We stress that in our application, it is not strictly necessary to obtain physical correctness of the movement, and for this reason, we adopted colored noise to generate the geometry of the patch and its movement. Note that in a practical scenario, the domain of $f$ would be obviously discretized into a 3D array.

After the geometry of a moving surface is defined, we render 153600 video clips of space-time surfaces $f$ by varying the parameters in (26), the light source parameters, the texture of the surface, its reflectance properties, and the variance of the additive noise that is added after the render. For rendering, we adopt the Phong reflectance model [36]. More accurate reflectance models could be used to increase real-
Fig. 6 Illustration of the effect of the parameters of (26) on the surfaces used as training data. From left to right: a patch obtained by setting $v, \xi, \eta$ to very low values and $\Sigma = I$; the same patch with increased constraint; with non-identity $\Sigma$; with increased flexibility; with increased folding.

Fig. 7 Two examples of scenes rendered with our method. From left to right, the 3D mesh, rendered object, and depth map are depicted.

10 Network Architecture

In order to estimate a depth video from its corresponding sequence of grayscale video frames, we use an architecture similar to the 3D U-net [12] (see Fig. 8). The input for the network is a 16 frames video and each frame size is $64 \times 64$ pixels. The output of the network is the corresponding depth video, which has size $64 \times 64 \times 16$. To enlarge the receptive field and get richer features, a context module [50] with atrous spatial pyramid pooling is introduced which performs parallel dilated convolutions with different dilation rates. These feature maps are then concatenated, and the output of the module is the convolution of these feature maps. For the first two stages in the network, three dilation rates are used (1, 2, 3). For the following two stages, two dilation rates are used (1, 2) as the feature maps dimensions get smaller. We use $3 \times 3 \times 3$ kernels for all convolutions. The leaky ReLU activation function is used for all layers, except for the last one, which has a linear activation function to predict the depths. The motivation behind this choice of architecture was that of using the 3D U-Net to get feature maps across time domain to learn local motion, which we observed to be better than using a 2D U-Net and consider the frames as channels or features. Also, using atrous convolution effectively enlarged the receptive field, and experimentally determined an increase in performance.

11 Reconstruction of a Depth Map from Local Patches

The proposed network architecture is designed to recover only depth videos whose frames have dimension $64 \times 64$. When the input video has larger dimensions, an additional step is required in order to reconstruct the full-sized depth map from the local patches. To address this issue, we split a frame of the input video into $64 \times 64$ squares with constant overlap and run our algorithm to estimate the depth in each block. Since the recovered depth maps are ambiguous with
The network architecture: the input is a grayscale video and the output is the corresponding sequence of depth maps. Below each layer, the number of channels is reported.

Fig. 8

| Table 1 | Average and standard deviation of MAE_	ext{sn} calculated from 15360 video sequences for the two loss functions considered |  |
|---------|---------------------------------------------------------------|---|
|         | \( \ell \) | \( \ell_\partial \) |
| Align each frame | 0.5226 ± 0.1123 | 0.3832 ± 0.1305 |
| Align to first frame | 0.7090 ± 0.1971 | 0.6133 ± 0.2591 |

Bold values indicate the minimum MAE among the entries of the same row of the table.

Table 2: Average and standard deviation of MAE_	ext{sn} calculated from 1000 video sequences using our method, two state-of-the-art NRSfM reconstruction algorithms (CSF2, KSTA), and one state-of-the-art method for video depth estimation (CVDE).

| Method | 0.8746 ± 0.6372 | 0.8738 ± 0.6369 | 0.8377 ± 0.6207 | 0.5907 ± 0.4536 |

Bold value indicates the minimum MAE among the entries of the same row of the table.

12 Experiments

In this section, we describe the experiments that were carried out to validate our method. We split this discussion in two parts in which we, respectively, test the performance of our method on synthetic images and on real data.

12.1 Experiments on Synthetic Data

Two main experiments were run on synthetic images. We generated a database of 153600 entries as explained in Sect. 9 and split it into three partitions: 80% for training, 10% for validation, 10% for testing. We used this database to train a CNN with the architecture described in Sect. 10.

In the first experiment, we considered two different loss functions during the training stage, which we call \( \ell_\partial \) and \( \ell \). The first one is based on differential invariants and is defined as:

\[
\ell_\partial = \text{MSE}(\iota_f, \iota_{f^*}),
\]

where \( \iota_f \) and \( \iota_{f^*} \) are, respectively, the estimated and the ground truth depth maps, and \( \iota \) is the differential invariant operator as defined in Equation (23). The second loss function, which we call \( \ell \), is defined analogously, and the only difference with \( \ell_\partial \) is that we replace the differential invariant \( \iota \) with the point cloud invariant of Equation (13). Since the differential invariants can be singular at some locations, we consider only those pixels where the quantities of \( \iota_f \) and \( \iota_{f^*} \) are well defined. To quantitatively evaluate the performance, we align the estimated depth maps with the corresponding ground truth by a linear transformation and calculate the mean absolute spatially normalized error:

\[
\text{MAE}_\text{sn}(f, f^*) = T^{-1} \sum_{t=1}^{T} \sigma_{f^*}^{-1}(\cdot, t) \text{MAE}(f(\cdot, t), f^*(\cdot, t))
\]

where \( T \) is the number of frames (16 in our experiments) in each spatiotemporal patch, and \( \sigma_{f^*}^{-1}(\cdot, t) \) denotes the standard deviation of the ground truth depth values of the frame \( t \). The use of Equation (27) is justified by the fact that the depth videos estimated with our method are scale-ambiguous; thus, a scale-invariant error metric like MAE_	ext{sn} allows one to measure...
Fig. 9 Results on synthetic data. (First row) five frames of an animation generated with the method described in Sect. 9. (Second row) the ground truth depth maps visualized both as a textured 2D surface in space and as color mapped image. (Third row) estimated depth maps using our method. (Fourth row) using CSF2. (Fifth row) KSTA. (Last row) CVDE
sure the quality of the reconstruction with respect to the ground truth depth, up to a multiplicative scalar. We repeat the same experiment by re-using the parameters for the alignment of the first frame for all the other frames. In a practical scenario, one could alternatively recover the parameters for the alignment by tracking points on a planar surface of the scene. The results of the above experiment are summarized in Table 1.

In the second experiment, we synthetically generate 1000 samples \((f_{i,\text{render}}, f_{i,\text{depth}})\) having size \(256 \times 256 \times 16\) using the method in Sect. 9 while using significantly different parameters in Equation (26) than the ones used in the training stage. We then automatically select 1089 points scattered uniformly on the 3D mesh and track their \(xy\)-coordinates on the image plane. We use these coordinates as inputs for two popular state-of-the-art non-rigid structure from motion methods: CSF2 [20], and KSTA [19], and obtain the \(z\)-coordinates of the tracked points for each frame. Note that according to the benchmark published in [22], the most recent current state-of-the-art method [28] performs very similarly to CSF2 [20]. To obtain a dense \(256 \times 256\) depth map, we use scattered interpolation of the estimated depth values of the videos obtained directly from the input video sequence. For the two NRSfM methods, CSF2 and KSTA, we performed point tracking on the two video sequences using the minimum eigenvalue algorithm [41], which yielded the highest amount of correctly tracked points and manually calibrated their parameters to obtain best results. The depth videos were then obtained as in the previous experiment, i.e., by recovering for each frame a dense depth map from scattered interpolation of the estimated depth values of the tracked points. CVDE needs a similar pre-processing step in which correspondences between frames are automatically found using optical flow. In this experiment, we had to establish 30 correspondences manually in the non-static region of interest of the video, since the automatic algorithm was unable to detect them. The results are given in Table 3, and the estimated depth maps are shown in Fig. 10.

### 12.2 Experiments on Real Data

In this experiment, we used real data, consisting of a grayscale video along with the corresponding depth map, acquired using Microsoft Kinect v2, which is equipped with a pair of RGB and depth perspective cameras. The video sequence was obtained by recording a video in an indoor environment of a curtain moved by a person located at about 2 meters from the camera. The video clip contained 60 frames with \(512 \times 424\) pixel resolution recorded at 30 frames per second. For each corresponding pixel, a finite depth value was recorded by the sensor. As in the previous section, we compare our method, trained with the same database as in the previous experiment, with three other methods in the literature, CSF2, KSTA, and CVDE. In our method, the depth videos are estimated directly from the input video sequence. For the two NRSfM methods, CSF2 and KSTA, we performed point tracking on the two video sequences using the minimum eigenvalue algorithm [41], which yielded the highest amount of correctly tracked points and manually calibrated their parameters to obtain best results. The depth videos were then obtained as in the previous experiment, where correspondences between frames are automatically found using optical flow. In this experiment, we had to establish 30 correspondences manually in the non-static region of interest of the video, since the automatic algorithm was unable to detect them. The results are given in Table 3, and the estimated depth maps are shown in Fig. 10.

### 13 Conclusion

We presented a deep-learning-based approach to recover a depth video directly from a video sequence of a non-rigidly deforming object acquired with a static orthographic camera. The problem of training the network was addressed by synthetically generating a large database of short videos depicting the local shape of deforming 3D meshes rendered with realistic textures, and paired with their corresponding depth videos. Since with an orthographic camera model there are potentially infinitely large classes of depth maps corresponding to the same rendered image, we addressed this issue by mathematically deriving maximally discriminative representations of surfaces that are invariant to the group of generalized has-relief transformations, and used them in the loss function during the training stage. We tested our method on both synthetic and real videos and experimentally observed that the quality of the estimated depth videos outperforms that of state-of-the-art NRSfM and video depth estimation algorithms for scenes that are mostly dominated by non-rigid motion.

| CSF2   | KSTA   | CVDE   | ours with \(\ell_\beta\) |
|--------|--------|--------|-------------------------|
| 4.7    | 3.6    | 3.1    | 2.2                     |

Bold value indicates the minimum MAE among the entries of the same row of the table.
Fig. 10 Results on real data. (First row) a frame and the corresponding depth map of a video sequence captured with Kinect v2. (Second row) close-ups of the region of the video containing the largest amount of motion extracted from six frames. (Third row) ground truth depth maps obtained with Kinect v2. (Fourth row) depth maps recovered with our method. (Fifth row) using CSP2. (Sixth row) KSTA. (Last row) CVDE. The contrast of the depth maps in the figure was enhanced for visualization purposes.
Acknowledgements The authors would like to thank Prof. Peter J. Olver for his technical advice, and Academy of Finland for the financial support for this research (grant no. 297732).

Funding Open Access funding provided by University of Oulu including Oulu University Hospital.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Ackermann, J., Goesele, M.: A survey of photometric stereo techniques. Founda- tions and Trends® in Comput. Graph. Vis. (2015), 9(3–4):149–254. issn: 1572- 2740, 1572-2759. https://doi.org/10.1561/0600000065.

2. Agudo, A., et al.: Sequential non-rigid structure from motion using physical priors. IEEE Trans. Pattern Anal. Mach. Intell. 38(5), 979–994 (2016). https://doi.org/10.1109/TPAMI.2015.2469293. (issn: 0162-8828, 2160-9292, 1939-3539)

3. Allen, J.B., Rabiner, L.R.: A unified approach to short-time fourier analysis and synthesis. In: Proceedings of the IEEE (1977), 65(11):1558–1564. issn: 1558–2256. https://doi.org/10.1109/PROC.1977.10770

4. Allen, J.B., Rabiner, L.R.: A unified approach to short-time fourier analysis and synthesis. In: Proceedings of the IEEE, vol. 65(11), pp. 1558–1564. issn: 1558–2256. https://doi.org/10.1109/PROC.1977.10770 (1977)

5. Aloimonos, J.: Shape from texture. Biol. Cybern. 58(5), 345–360 (1988). https://doi.org/10.1007/BF00363944. (issn: 1432-0770)

6. Belliveau, P.N., Kriegman, D.J.: and A.L. Yuille. The bas-relief ambiguity. In: Proceedings of IEEE Computer Society Conference on Com- puter Vision and Pattern Recognition. San Juan, Puerto Rico: IEEE Comput. Soc. pp. 1060–1066. https://doi.org/10.1109/PROC.CVPR.1997.609461 (1997)

7. Bregler, C., Hertzmann, A., Biermann H.: Recovering Non-Rigid 3D Shape from Image Streams. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662). Vol. 2. 690-696. https://doi.org/10.1109/CVPR.2000.854941 (2000)

8. Bridson, R., Hourihm, J., Nordenstam, M.: Curl-noise for procedural fluid flow. ACM Trans. Graph. 26(3), 46 (2007). https://doi. org/10.1145/1276377.1276435. (issn: 0730-3031)

9. Elie, Cartan,LA Méthode de Repère Mobile, La Théorie Des Groupes Continus, et Les Espaces Généralisés. In: Exposés de géométrie No. 5, Hermann, Paris (1935)

10. Cartan, E.: La Théorie Des Groupes Finis et Continus et La Géométrie Différentielle Traitées Par Méthode Du Repere Mobile. Cahiers sci- entifiques 18, Gauthier-Villars, Paris (1937)

11. Chandraker, M.K., Kahl, F., Kriegman, D.J.: Re ections on the Generalized Bas-Relief Ambiguity. In: 2005 IEEE Computer Society Confer- ence on Computer Vision and Pattern Recogni- tion (CVPR’05). Vol. 1, 788–795. https://doi.org/10.1109/CVPR.2005. 299. (2005)

12. özgün, Çiçek, et al.: 3D U-Net: learning dense volumetric segment- mentation from sparse annotation. In: International Conference on Medical Image Computing and Computer-Assisted Inter- vention. Springer, pp. 424–432 (2016)

13. Cimpoi, M., et al.: Describing textures in the wild. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recogni- tion.(CVPR) (2014)

14. Yuchao, D., Huizhong, D., Mingyi, H.: Dense non-rigid structurefrom-motion made easy|A spatial-temporal smoothness based solution. In: IEEE International Confer- ence on Image Processing (ICIP). IEEE. IEEE. 2017, pp 4532–4536 (2017)

15. Sven Dickinson, J., Pizlo, Z.: Shape perception in human and com- puter vision. Springer, (2013)

16. Fels, M., Olver, P.J.: Moving coframes: I. A Practical Algorithm. Acta Applicandae Math. 51(2), 161–213 (1998)

17. Fels, M., Olver, P.J.: Moving coframes: II. Regularization and theoretical foundations. Acta Applicandae Math. 55(2), 127–208 (1999)

18. Fyffe, G., Yu, X., Debevec, P.: Single-Shot Photometric Stereo by Spectral Multiplexing. In: 2011 IEEE International Conference on Computational Photography (ICCP), pp. 16. doi: 10 . 1109 / ICCPHOT . 2011 . 5753116 (2011)

19. Gotardo P.F.U., Martinez, A.M.: Kernel Non-Rigid Structure from Motion. In: 2011 International Conference on Computer Vision. pp. 802809. https://doi.org/10.1109/ICCV.2011.6126319 (2011)

20. Gotardo P.F.U., Martinez, A.M.: Non-rigid structure from motion with complementary rank-3 spaces. In: CVPR 2011, pp. 3065–3072. https://doi.org/10.1109/CVPR.2011.5995560 (2011)

21. Ikeuchi, K., Horn, B.K.P.: Numerical Shape from Shading and Occluding Boundaries. Artif. Intell. 17(1), 141–184 (1981). https://doi.org/10.1016/0004-3702(81)90023-0. (issn: 0004-3702)

22. Hoppe, S., Jensen, N., et al.: A benchmark and evaluation of non-rigid from motion. arXiv preprint arXiv:1801.08388. arXiv: 1801.08388 (2018)

23. Kim, H., Wilburn, B., Ben-Ezra, M.: Photometric stereo for dynamic surface orientations. In: Computer Vision-ECCV 2010. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 59–72. isbn: 978-3-642-15549-9. https://doi.org/10.1007/978-3-642-15549-9-5 (2010)

24. Irina, K.A.: Inductive construction of moving frames. Contemp. Math. 285, 157–170 (2001)

25. Kong, C., Lucey, S.: Deep Non-Rigid Structure from Motion. In: Proceedings of the IEEE International Conference on Computer Vi- sion, pp. 1558–1567 (2019)

26. Kumar, S., Dai, Y., Li, H.: Multi-body non- rigid structure-from-motion. In: 2016 Fourth International Conference on 3D Vision, pp. 148–156. https://doi.org/10.1109/3DV.2016.23 (2016)

27. Kumar, S., et al.: Dense depth estimation of a complex dynamic scene without explicit 3D motion estimation. In: arXiv:1902.03791 [cs]. arXiv: 1902.03791 [cs] (2019)

28. Kumar, S., et al.: Scalable dense non-rigid structure-from-motion: a grassmannian perspective. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 254–263 (2018)

29. Larsson, V., Pollefeys, M., Oskarsson, M.: Orthographic- perspective epipolar geometry. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5570–5578 (2021)

30. Liu, F., et al.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Trans. Pattern Anal. Mach. Intell. 38(10), 2024–2039 (2016). https://doi.org/10.1109/TPAMI. 2015.2505283. (issn: 1939-3539)
