Tracking of Non-Rigid Objects using RGB-D Camera

Agniva Sengupta, Alexandre Krupa, Eric Marchand

Tracking of Non-Rigid Objects using RGB-D Camera

Agniva Sengupta, Alexandre Krupa and Eric Marchand
Univ Rennes, Inria, CNRS, IRISA, France

Abstract— **A method to accurately track deformable objects using a RGB-D camera with the help of a coarse object model is presented in this paper. The deformation model is based on corotational FEM formulation. The physical model of the object does not need to be exact, nor do we require the precise physical properties for accurately tracking the object. The position of the vertices of the surface mesh of the tracked object is deformed using a set of virtual forces. A point-to-plane distance based geometric error between the pointcloud and the mesh is minimized with respect to these virtual forces. The point of application of force is determined by analysis of the error obtained from rigid tracking, which is done in parallel with the non-rigid tracking. This architecture also enables the overall system to be realtime. The proposed approach is evaluated on a synthetic data with ground-truth for deformation at every frame, as well as on real data.**

I. INTRODUCTION

Interaction with soft objects using machines remain a challenging problem in the field of robotics. To precisely interact with deforming objects using any robotic manipulator, it is necessary to track the surface of the object with high accuracy. Non-rigid object tracking also finds implementation in the field of augmented/mixed reality systems. This is an open area of research and the available literature that aims to tackle this problem is not very extensive.

Real-time tracking of the surface of non-rigid objects using RGB-D camera has become possible only very recently. However, most of the proposed approaches for non-rigid object tracking either requires a detailed model or considers tracking and reconstruction as intertwined problems. The tracking and reconstruction algorithm usually generates impressive visual results, but the accuracy of the final reconstruction does not necessarily imply accurate frame-to-frame tracking of the non-rigid object. Moreover, for model-based tracking, a detailed model of the physical properties of the object may not always be available in practice.

In this paper, we propose a method to handle the problem of accurate tracking of the surface of non-rigid objects undergoing deformation. A commodity-level RGB-D camera is used for sensing. It is assumed that we know the *visual-surface model* of the object (which can be a CAD model), but this model does not have to be precise.

A. Related Work

We propose to track the object using its *3D model*, which consists of a mathematical representation of the surface of the object in 3D. To track the complete surface of deforming objects using depth (or monocular) camera when its 3D model is available, there are three components which are generally used: **a)** an algorithm to track each face of the model w.r.t the observed 3D depth, possibly along with the corresponding image intensity values of the pixels, **b)** a mechanism to impart coherence to the motion of these individual faces, which is done either with the help of a geometric shape preserving function (also called a regularizer), such as [1], [2] or [3], or with some assumptions about the underlying physical properties of the material of the object, such as [4] or [5], and **c)** some strategy to combine the tracking of the surface with the regularizer or the physical model of the object (e.g: 'deterministic annealing' in case of [2]).

Some of the popular methods for geometrically regularizing the deformations includes as-rigid-as-possible (ARAP) regularizer [6], embedded deformation graphs [7] and thin plate spline model [8]. The physics based framework for deformation modelling includes mass-spring-damper systems [9], finite element model (FEM) [10] and kinematic chain [11]. For model based tracking of non-rigid objects, the representation of the model is usually done by a set of triangular faces representing the surface of the object. However, tetrahedral, volumetric model is preferred [12] for tracking using mass-spring-damper systems, while the FEM based systems available in the literature typically preferred to use a combination of surface model and volumetric model. Haouchine *et. al.* [13] proposed a linear tetrahedral co-rotational FEM based model for tracking large deformations. [14] uses co-rotation FEM based model to track deformation by estimating the direction and magnitude of elastic force acting on the object using depth information. As summarized in [15], FEM based approaches suffer from excessive dependency on the availability of the accurate physical properties of the object being tracked. The approach we propose in this paper removes this dependency using a novel, closed loop minimization technique.

The recent advancements in non-rigid object tracking should also include a survey of tracking and reconstruction algorithms. [16] uses ARAP to regularize a warp field from the current state of deformation to a canonical model. The reconstructed model is visually coherent and accurate. [17] improves upon this work and proposes to enhance the accuracy by introducing SIFT based sparse correspondence. [18] achieves the same objective, but uses Approximately Killing Vector Field (AKVF) to regularize the deformation. However, these approaches do not evaluate the per-frame tracking accuracy. The use of Truncated Signed Distance Field (TSDF) makes it easier to reconstruct the model, despite minor inaccuracies in tracking.

B. Contribution

We propose a method for tracking the entire visible surface of a deforming, non-rigid object in 3D, such that it:

- requires only a very approximate estimate of the physical properties (Young's modulus and Poisson ratio) of the object.
- performs accurate frame-to-frame tracking of deformation of the non-rigid object.
- has been validated on a simulated data with ground truth, as well as real data.

The method that we present in this paper is closely related to [14]. We use co-rotational FEM to simulate the physics behind the deformation of non-rigid objects and we use a tetrahedral mesh as the physical model of the object. But, unlike the approaches in the literature, we extend the method to perform a closed-loop optimization based on a point-to-plane geometric error. This additional step of minimization effectively reduces the dependency of the system on the accuracy of the physical properties of matter used for the finite element modelling. Moreover, we dissociate the rigid and non-rigid tracking into two parallel modules. The rigid tracking is purely based on the surface model, while the FEM is used strictly for tracking the non-rigid deformations. This parallel system is capable of real-time performance. Our approach has been both quantitatively and qualitatively evaluated using a combination of ground-truthed, simulated data and real data.

II. METHOD

We will now describe the details of the methodology that we use for tracking deformable objects. We begin by describing the preliminary notations and the elastic deformation model that we use for interpreting the physics behind the objects we track. This is followed by a brief description of the rigid registration module (**section** II-C). The description of matching the deformable object between consequent frames is discussed next (**section** II-D). We follow this up with the mechanism for computing the Jacobian that links the variation in the geometric error with the variation of force applied on a particular vertex. We use this Jacobian for minimizing the error using an Iteratively Reweighted Least Square (IRLS) formulation. We finish this section by summarizing the steps for minimizing the non-rigid error.

A. Notation

We use two types of 3D model for the non-rigid tracking. The first is a **visual-surface model**, denoted by $\mathcal{V}$, and the other one is a tetrahedral, volumetric, **internal model**, denoted by $\mathcal{M}$. The *visual-surface model* is described using a set of planes which are represented using the nodes defining the boundary of the surface (usually a triangle) and the connectivity between these nodes. We can represent this surface as $\mathcal{V} = \begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & ... & \mathbf{p}_M\end{bmatrix}$, where M is the number of vertices in $\mathcal{V}$ and the i-th vertex is $\mathbf{p}_i = \left(X_i, Y_i, Z_i\right)$, the 3D coordinate of a point in the reference frame of the base model. We have a corresponding, connectivity map, given by $\mathbf{X}^{\mathcal{V}} = \left\{\{a_1, b_1, c_1\}...\{a_N, b_N, c_N\}\right\}$, where $a_i, b_i, c_i \in [0..M]$ denotes the indices in $\mathcal{V}$ and N is the number of faces in $\mathcal{V}$, thereby defining connectivity between its elements. Clearly, the normal $\mathbf{n}_i = (n_i^X, n_i^Y, n_i^Z)$ and distance to origin d_i for any given face can be derived from $\mathcal{V}$ and $\mathbf{X}^{\mathcal{V}}$ using elementary geometry. We similarly define a tetrahedral, volumetric, internal model $\mathcal{M}$ along with its connectivity map $\mathbf{X}^{\mathcal{M}}$, such that every element of $\mathbf{X}^{\mathcal{M}}$ has four indices, instead of three. The *visual-surface model* and the *internal model* is maintained at the same reference frame and together, we refer to them as the **base model**.

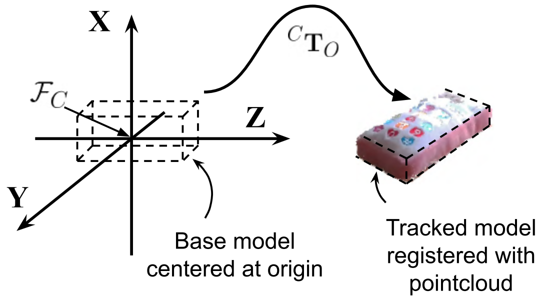

X
Z
Y
$\mathcal{F}_C$
${}^C\mathbf{T}_O$
Base model centered at origin
Tracked model registered with pointcloud

Fig. 1: The depth camera and the *base model* is centered at $\mathcal{F}_C$, the *tracked model* is used for the rigid registration of the pointcloud

In the proposed approach, a depth camera is used to track the object from the observed pointcloud. The camera is maintained at a camera-centered frame of reference $\mathcal{F}_C$. The intrinsic parameters of this camera are denoted by f_x, f_y, the focal lengths and c_x, c_y the optical centers of the image. The pointcloud captured by the depth camera is represented as a set of points $\mathbf{P}_i = (x_i, y_i, z_i)$, which are the 3D coordinate of the points in the camera's reference frame. The *visual-surface model* of the object being tracked, denoted by the tuple $\{\mathcal{V}, \mathbf{X}^{\mathcal{V}}\}$, is also based on this same frame of reference, as shown in **Fig.** 1.

We use the homogeneous matrix ${}^A\mathbf{T}_B = \left[\begin{array}{c|c} {}^A\mathbf{R}_B & {}^A\mathbf{t}_B \\ \hline 0 & 1 \end{array}\right]$ to define the rigid transformation between any two arbitrary Cartesian frame $\mathbf{F}_A$ and $\mathbf{F}_B$, where $\mathbf{T} \in \mathbb{SE}_3$. In between frames, the rigid motion of the object with respect to its previous pose is denoted by $\mathbf{q} = ({}^A\mathbf{t}_B, \theta\mathbf{u})$, where θ and $\mathbf{u}$ are the angle and axis of the rotation ${}^A\mathbf{R}_B$. The time derivative of $\mathbf{q}$ is given as $\mathbf{v} = \delta\mathbf{q}$, where $\mathbf{v} \in \mathfrak{se}(3)$ is the velocity screw.

${}^C\mathbf{T}_O$ denotes the transformation from the camera-centered coordinate frame $\mathcal{F}_C$ to the object in the pointcloud. For rigid tracking, we define a **tracked model** using the tuple $\{{}^O\mathcal{V}, \mathbf{X}^{\mathcal{V}}\}$, such that:

$$ {}^O\mathcal{V} = {}^O\mathbf{R}_C {}^C\mathcal{V} + \underbrace{\begin{pmatrix} {}^O\mathbf{t}_C & \cdots & {}^O\mathbf{t}_C \end{pmatrix}}_{3\times M} \quad (1) $$

During non-rigid deformation of the *base model* (${}^C\mathcal{V}$) centered at $\mathcal{F}_C$, the transformation of (1) is repeated, so that the model at the object centered reference frame always stays updated according to the deformation.

B. Deformation Modelling

We use *Finite Element Model* (FEM) for modelling the deformation. For the object representation, we consider the *visual-surface model* and the *internal model* to be coupled with each other in a master-slave configuration. The *internal model* acts as the mechanical model, which is the master of the system [19]. We use a function $\mathcal{J}$ to map the position of ${}^{C}\mathcal{M}$ to ${}^{C}\mathcal{V}$ such that $\mathbf{p}_i^{\mathcal{V}} = \mathcal{J}\big(\mathbf{p}_i^{\mathcal{M}}\big)$.

We use a co-rotational FEM formulation for estimating the displacement of the nodes after being acted upon by an external force (for a detailed background on FEM, interested readers may refer to [20]). Beginning from the previous description of the visual and the *internal model*, we describe the co-rotational model for a single, tetrahedral element of $\mathcal{M}$. Let the four vertices of this particular face be denoted by $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$. Let the centroid be $\mathcal{F}_C$, as shown in **Fig.** 2.

Let us assume that in the next frame, the model undergoes a rigid rotation, as well as a deformation. Let the rotated centroid at the new frame be given by $\mathbf{C}_R$. An arbitrary point $\mathbf{p}_x$ in the first frame gets deformed to $\mathbf{p}$ in the next frame. Using the corotational model [21], it can be shown that:

$$\tilde{\mathbf{d}} = \mathbf{R}^\top\big(\mathbf{U} - \mathbf{C} + \mathbf{p}_x\big) - \mathbf{p}_x \tag{2}$$

where $\mathbf{R}$ is the rotation matrix (also called the *rotator*), $\mathbf{U}$ is the total displacement of $\mathbf{p}_x$, expressed in the base frame $\mathbf{C}$, and $\tilde{\mathbf{d}}$ is the deformational displacement of $\mathbf{p}_x$, expressed in the corotated frame $\mathbf{C}_R$ (all variables with $\tilde{}$ are expressed in the corotated frame $\mathbf{C}_R$). Now, applying similar motion to the nodes of the tetrahedron, we can represent the internal, elastic forces acting on the nodes by:

$$\mathbf{F}_e = \mathbf{R}_e\mathbf{B}_e\tilde{\mathbf{U}}^R \tag{3}$$

Here, $\underset{12\times1}{\tilde{\mathbf{U}}^R = (\tilde{\mathbf{p}_1}, \tilde{\mathbf{p}_2}, \tilde{\mathbf{p}_3}, \tilde{\mathbf{p}_4})}$, $\underset{12\times12}{\mathbf{B}_e}$ is the stiffness matrix and $\underset{12\times12}{\mathbf{R}_e}$ is the block diagonal matrix of four $\mathbf{R}$ rotation matrices stacked diagonally. $\underset{12\times1}{\mathbf{F}_e}$ is the elastic forces acting on the nodes of this tetrahedral element.

To resolve the interaction between forces and their resulting displacement using the FEM, we solve a second order differential equation, given by:

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{B}\mathbf{p} = \mathbf{F}_{ext} + \mathbf{F}_e \tag{4}$$

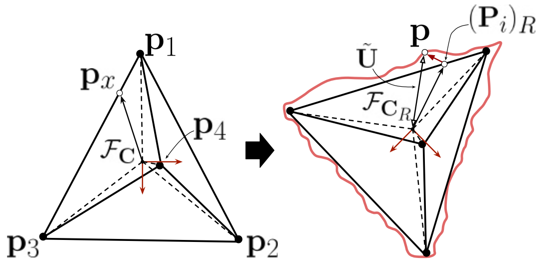


Fig. 2: Demonstration of the kinematics of a single element of a corotated, tetrahedral mesh. In the corotated frame, the red outline shows the deformed surface

where $\mathbf{M}$ and $\mathbf{D}$ are the model's mass and damping matrices respectively. $\mathbf{B}$ is the global stiffness matrix. $\mathbf{F}_{ext}$ are the external forces acting on the vertices. The method for determining $\mathbf{F}_{ext}$ is discussed in the subsequent sections. To solve this differential equation, we use a linear solver based on conjugate gradient descent [19]. To impose additional constraint on the solution of (4), it can be multiplied using a projection matrix that sets the values of certain indices to zero. We use the projective constraint to eliminate the rigid motion of the object. This is discussed in the next section

It must be noted that the force $\mathbf{F}_{ext}$ is just the deforming force acting on the model, and does not include the forces causing the rigid transformation of the body. In fact, the deformation model proposed here is completely independent of rigid motion, the effects of gravity and interaction with other contact surfaces. As described in **section** II-C, the rigid motion is tracked separately. This separation of rigid and non-rigid tracking method, along with the minimization described **section** II-D, makes the overall system independent from the inaccuracies of the physical parameters.

C. Rigid Registration

The rigid registration is a joint minimization of two error terms: depth based geometric error and keypoint based feature tracking [22]. The two error terms are explained below:

1) Depth based geometric error: Assuming that we know the accurate ${}^{C}\mathbf{T}_O^{n-1}$ at the $(n-1)$-the frame and ${}^{n-1}\mathbf{T}_n$ gives the initial estimate of transformation between previous and current frame, the error is given as:

$$\mathbf{e}^D({}^{n-1}\mathbf{q}_n) = \Big(\big({}^{n-1}\mathbf{R}_n\mathbf{P} + {}^{n-1}\mathbf{t}_n\big)\cdot\mathbf{n}_k\Big) - d_k \tag{5}$$

where $\mathbf{n}_k$ and d_k are the normal and distance to origin respectively, for the k-th planar face that corresponds with the point $\mathbf{P}$ in the pointcloud. The Jacobian that links the variation of the error $\mathbf{e}^D({}^{n-1}\mathbf{q}_n)$ with the time varying ${}^{n-1}\dot{\mathbf{q}}_n$, is given by:

$$\mathbf{J}_i^D = \Big[\mathbf{n}_k^\top \quad \big[\mathbf{n}_k\big]_\times {\mathbf{P}_i}^\top\Big] \tag{6}$$

2) Feature based minimization: The Harris corner features are used for tracking with keypoints. This is based on the classical KLT algorithm [23]. Let $\mathbf{u} = (x, y, 1)$ be the homogeneous 2D coordinate of a feature point in the $(n-1)$-th image and $\mathbf{u}^* = (x^*, y^*, 1)$ be the matched coordinate for the same point in the n-th frame. We define the error term as:

$$\mathbf{e}^K({}^{n-1}\mathbf{q}_n) = \begin{pmatrix} x({}^{n-1}\mathbf{q}_n) - x^* \\ y({}^{n-1}\mathbf{q}_n) - y^* \end{pmatrix} \tag{7}$$

where $(x({}^{n-1}\mathbf{q}_n), y({}^{n-1}\mathbf{q}_n), 1) = {}^{n-1}\mathbf{H}_n\mathbf{p}$, given that:

$${}^{n-1}\mathbf{H}_n = {}^{n-1}\mathbf{R}_n + \frac{{}^{n-1}\mathbf{t}_n}{d}\mathbf{n}^\top \tag{8}$$

where $\mathbf{n}$ and d are interpreted in the same way as (5). The corresponding Jacobian that relates the variation of $\mathbf{e}^K({}^{n-1}\mathbf{q}_n)$ with the time variation of ${}^{n-1}\mathbf{q}_n$ is given by:

$$\mathbf{J}^K = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix} \tag{9}$$

Given these two errors and their corresponding Jacobian, the combined error $\mathbf{e}({}^{n-1}\mathbf{q}_n)$ is obtained by stacking the vectors $\mathbf{e}^D({}^{n-1}\mathbf{q}_n)$ and $\mathbf{e}^K({}^{n-1}\mathbf{q}_n)$, while the combined Jacobian $\mathbf{J}$ is obtained by stacking $\mathbf{J}^D$ on $\mathbf{J}^K$. The combined error is minimized with the update given by:

$$\mathbf{v} = -\lambda\big(\mathbf{W}\mathbf{J}\big)^{+}\mathbf{W}\mathbf{e} \tag{10}$$

where $\mathbf{W}$ is the weight matrix for outlier removal using the Tukey m-estimator [24], and $\mathbf{v} \in \mathfrak{se}(3)$. This becomes an iteratively re-weighted least squares problem.

D. Non Rigid Tracking

Before starting the non-rigid tracking using a minimization technique, it is necessary to determine the possible points of application of forces that needs to be tracked. It is sufficient to track only a small subset of nodes, depending upon a set of criteria. These criteria, as detailed below, is obtained by clustering the error from the rigid tracking step. Only regions surrounding the clusters with higher value of errors are considered to be relevant for non-rigid tracking.

At the end of the rigid registration process, we can easily construct a map that links the points in the 3D pointcloud to their corresponding geometric error from (5). Given that $\mathbf{e}_i^D$ denotes the error for the i-th point $\mathbf{P}_i$ in the entire pointcloud $\mathbf{P}$, it is possible to derive a new pointcloud $\mathbf{P}^*$ using a linear thresholding operation, such that $\mathbf{P}^* = \{\mathbf{P}_i \in \mathbf{P} | \mathbf{e}_i^D \geq \theta^D\}$, where θ^D is a threshold that depends on the geometry of the object being tracked. We subject the pointcloud $\mathbf{P}^*$ to a clustering step using Eucledian distances [25]. Very small clusters and clusters spanning more than half of the size of the entire pointcloud are discarded. Let us assume that we obtain j clusters from this operation, denoted by $\mathbf{K}_1, \mathbf{K}_2, \cdots, \mathbf{K}_j$, and their corresponding centroids are represented by $\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_j$, where $\mathbf{k} = (x, y, z, 1)$ is the homogeneous 3D coordinate of the point w.r.t $\mathcal{F}_C$. These points are inverse transformed with the last estimate of ${}^C\mathbf{T}_O$ obtained after minimization with (10), such that $\mathbf{k}_i^* = \big({}^C\mathbf{T}_O\big)^{-1}\mathbf{k}_i$. For all these j centroids, we determine their nearest neighbors in ${}^C\mathcal{M}$ using the k-nearest neighbor algorithm [26]. Let the nearest neighbors of $\mathbf{k}_1^*, \mathbf{k}_2^*, \cdots, \mathbf{k}_j^*$ in ${}^C\mathcal{M}$ be denoted by ${}^C\mathcal{M}_{k1}, {}^C\mathcal{M}_{k2}, \cdots, {}^C\mathcal{M}_{kj}$.

For the following discussion, let us consider an arbitrary k_i-th cluster alone, which we refer to as ${}^C\mathcal{M}_i$ for the sake of simplicity. At this stage, we must modify (5) to accommodate a slightly different variant of the same error function. We re-define it as $\mathbf{e}^N\left({}^O\mathcal{V}_i\right) = \mathbf{n}_r \cdot \mathbf{P}_s - d_r$ assuming the r-th plane corresponds to the s-th point in the pointcloud, and the normal $\mathbf{n}_r$ and the distance to origin d_r is derived from ${}^O\mathcal{V}_i$.

Jacobian Computation: The Jacobian that relates the variation of $\mathbf{e}^N\left({}^O\mathcal{V}_i\right)$ with the variation of the applied force is computed numerically by perturbing the node ${}^C\mathcal{M}_i$ by a very small, constant force $\Delta\mathbf{F}_J$ successively along the three axes, as shown in **Fig.** 3. After applying the force on the node, a simulation of FEM is done using the minimization described in **section** II-B - (4). After the conjugate gradient solver optimizes (4), the system will attain a static condition. There will be six such configurations of the mesh, obtained by the perturbation along the positive and the negative direction of the three axes. We denote these new configurations of ${}^C\mathcal{M}_i$ as ${}^C\mathcal{M}_{J_i}^{X+}$, ${}^C\mathcal{M}_{J_i}^{X-}$, ${}^C\mathcal{M}_{J_i}^{Y+}$, ${}^C\mathcal{M}_{J_i}^{Y-}$, ${}^C\mathcal{M}_{J_i}^{Z+}$ and ${}^C\mathcal{M}_{J_i}^{Z-}$. This is immediately propagated down to the *visual-surface model* via the map ${}^C\mathcal{V}_{J_i} = \mathcal{J}({}^C\mathcal{M}_{J_i})$. We take these modified, visual mesh and transform it back to the registered pointcloud using (1). The relation of the variation of the external force with the variation in error can be expressed as $\dot{\mathbf{e}}^N({}^O\mathcal{V}_{J_i}) = \mathbf{J}_i\dot{\mathbf{F}}_{ext}$ The term $\mathbf{J}_i$ is obtained numerically by finite difference computation. The final Jacobian used is: $\mathbf{J}_i = \begin{pmatrix}\mathbf{J}_i^X & \mathbf{J}_i^Y & \mathbf{J}_i^Z\end{pmatrix}$. The temporary deformation of the model for Jacobian computation is discarded, once the Jacobian has been determined.

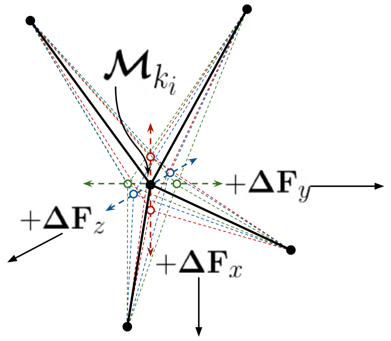


Fig. 3: The node closest to the centroid of a cluster is perturbed by a small force along the three axes in both positive and negative direction, producing six deformed configurations per node

Minimization: So far, j nodes have been selected for application of external force to deform the model. We have n points in the pointcloud, and their correspondence with the model is known. At the first iteration, the initial estimate of the vertically stacked force vector $\underbrace{\mathbf{F}_{ext}^{n-1}}_{3j\times 1}$ is set to zero. These Jacobian matrices are stacked up horizontally, such that $\underbrace{\mathbf{J}}_{n\times 3j} = \begin{pmatrix}\mathbf{J}_1 & \mathbf{J}_2 & \cdots & \mathbf{J}_j\end{pmatrix}$ The update is computed as: $\Delta\mathbf{F} = -\lambda\big(\underbrace{\mathbf{W}}_{n\times n}\mathbf{J}\big)^{+}\mathbf{W}\underbrace{\mathbf{e}^N({}^O\mathcal{V})}_{n\times 1}$ The force vector is updated by $\mathbf{F}_{ext}^{n} = \mathbf{F}_{ext}^{n-1} + \Delta\mathbf{F}$. This force is applied on the *base model* and the final node displacements are determined, once again, using (4).

III. IMPLEMENTATION

There are a few practical aspects of implementing this algorithm which needs to be discussed. It is necessary to associate every 3D point $\mathbf{P}_i$ with one of the planes of the *visual-surface model* ${}^O\mathcal{V}$. At every frame, we know the value of ${}^C\mathbf{T}_O$ obtained at the previous frame. We project the 3D points ${}^O\mathcal{V}$ for all visible planes into the image obtained in the current frame. The visibility is checked using the classical ray-casting algorithm [27]. The ray-casting is also necessary for imposing the projective constraints. The nodes invisible to the camera are considered to be immobile. We do not consider the volumetric, internal mesh to be an input to the system. The mesh is rather generated apriori, using Dirichlet tessellation, followed by Delaunay tetrahedralization of the input mesh using the Bowyer-Watson algorithm [28].

The approach proposed here does not require a visual segmentation for separating the region of interest from the background. This is done by initializing the pose of the object at the first frame using pre-trained markers on the object. It is done using the ViSP library [29], by matching the keypoints detected in the very first image with those extracted in the training images using an approach similar to [30].

The algorithm is implemented using a parallel framework, where two different processes are involved with the rigid and non-rigid tracking of the object. The rigid tracking module is capable of processing each frame at < 100 ms, thereby ensuring real-time interaction. The un-optimized, non-rigid tracking code runs at 800ms to 1.3s (approximately) per frame, depending on the size of the object and its proximity to the camera. The results reported here have been achieved using processes running on a single core of a i7-6600U CPU with 16GB of RAM. An Intel Skylake GT2 GPU has been utilized, but only for the ray-casting. The FEM solvers are implemented using the SOFA library [19], without using CUDA. The rigid tracking process runs at > 10 fps, while the non-rigid tracking module handles every 10-th/12-th frame coming from the sensor.

IV. RESULTS

The results are validated on two sets of simulated data, as well as on multiple real objects[1].

Simulation: For the simulated data, the deformation of the objects were generated using simulation of co-rotational FEM. Two objects are considered, a *cube* and a rectangular *board*. They are made to undergo simple but large deformations, as shown in **Fig.** 4. We made the dataset available at: github.com/lagadic/nr-dataset. The visual models produced as an output of the simulation were subjected to texturing, shading and rendering using the Blender[2] software, followed by the generation of the pointcloud using a RGB-D camera simulator. In the simulation, both the objects were modelled using Young's modulus (**YM**) of 50000 Pa and Poisson's ratio of 0.3. Rayleigh mass was assumed to be 0.1 and Rayleigh stiffness to be 0.3.

The simulated dataset is used not only to validate our approach, but also to compare the proposed method to a partial re-implementation of [14], where a co-rotational FEM is simulated for non-rigid object tracking. The external, elastic forces, $\mathbf{f}_{ext}$, acting on the object is given by $\mathbf{f}_{ext,i} = k_{ext,i}(\mathbf{x}_i - \mathbf{y}_i)$, where k_{ext} is the stiffness corresponding to the external elastic force and $\mathbf{y}_i$ is a point in the pointcloud which has been matched to $\mathbf{x}_i$, a vertex on the mesh. We re-implemented this method. Moreover, to reduce ambiguity, the per vertex displacement vector $(\mathbf{x}_i - \mathbf{y}_i)$ was obtained from the ground-truth in this re-implementation. In the subsequent text, the results obtained from the approach proposed in this paper are denoted by **PA**, while the results from the partial re-implementation of [14] are denoted by **SOA**.

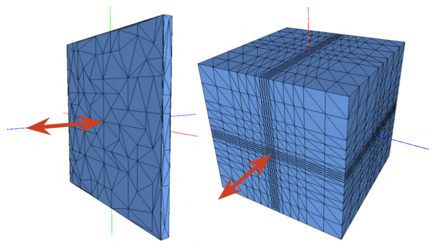


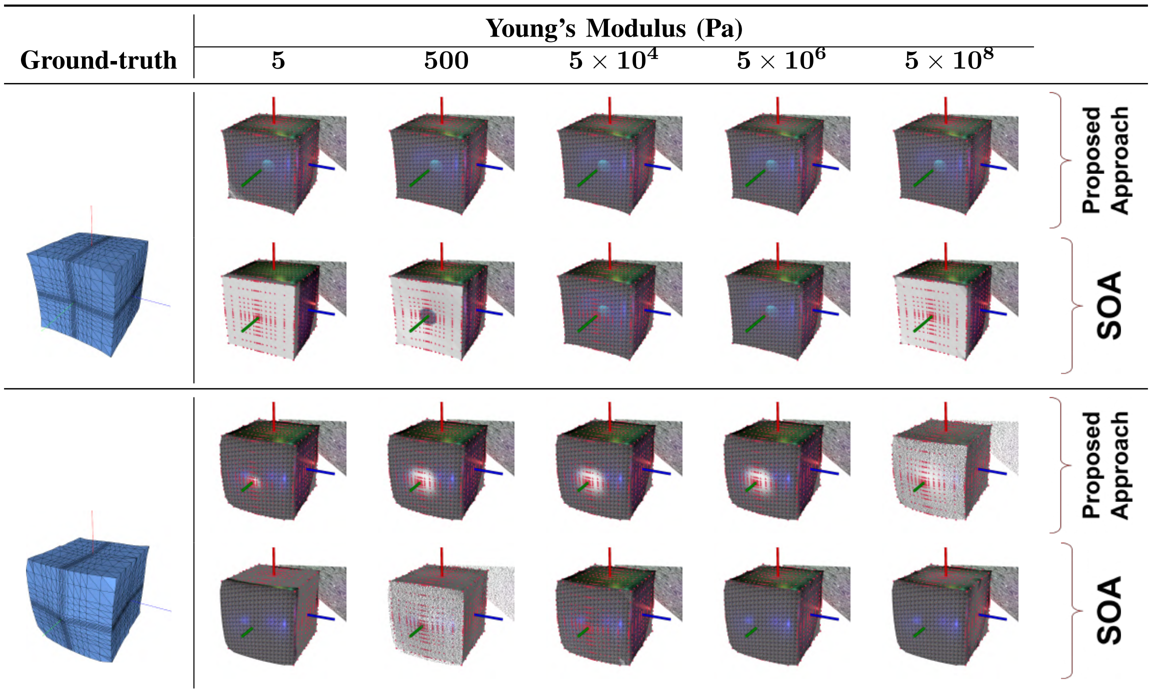
Fig. 4: The undeformed model of the *board* and the *cube*. The arrows in the image shows the approximate direction of application of force

To analyze the robustness of the two approach, experiments were repeated by varying the Young's modulus of the model for determining the variance in error with change in physical parameters. The physical parameters used for simulation are considered unknown in the implementation of the proposed approach. The tracking was repeated over the values of 5 Pa, 500 Pa, 50000 Pa, 5×10^6 Pa and 5×10^8 Pa respectively. The results are summarized in **Fig.** 5. In the figure, the ground-truth of the model undergoing deformation is provided in the left-most column. The rest of the images show the 3D model (the **tracked model**) obtained from the tracking algorithm, placed into the pointcloud with the latest estimate of ${}^{C}\mathbf{T}_{O}$.

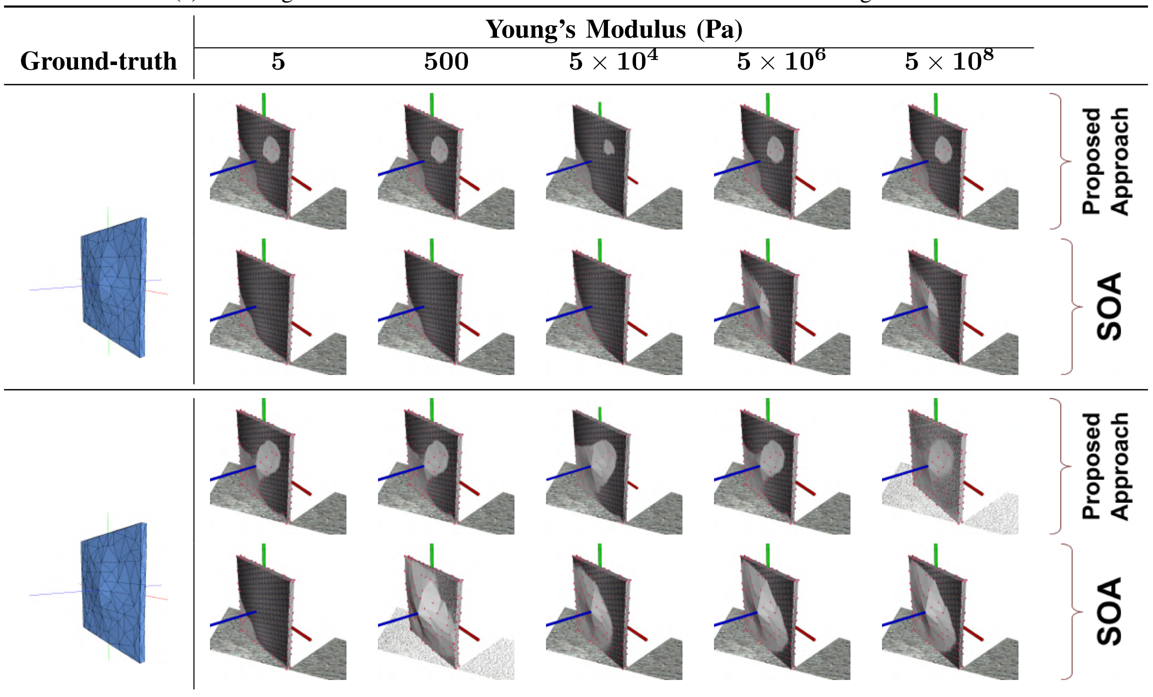
Since the ground-truth for the deformation is known, it is possible to compute the Hausdorff distance metric [31] between the output of the tracking and the ground-truth, to quantify the error between the ideal and the actual output. **Fig.** 6 delineates the Hausdorff distance of the output for various values of Young's modulus.

As given in **Table** I, the mean error in terms of Hausdorff distance using **PA** and **SOA** are 0.305 and 1.622 units for the *cube* dataset and 0.834 and 1.100 units for the *board* dataset respectively (the length of the largest diagonal is 69.28 and 55.19 units for the *cube* and the *board* models respectively). Hence, the tracking accuracy of the proposed approach is 81.18% better than **SOA** in the *cube* data and 24.20% better than **SOA** in the *board* data. The tracking accuracy of the proposed approach in terms of Hausdorff distance varies only by an average of 5.006% when the Young's modulus is increased by a factor of 10^8. It can therefore be claimed that the system is significantly robust to error in the estimation of the physical properties used to model the deforming object. It must be noted that the results shown for **SOA** contains only the error in estimating the magnitude of deformation of the objects. As stated before, the direction of application of force is provided to **SOA** from the ground-truth in our re-implementation (while **PA** makes no such exception), which explains the impressive accuracy of **SOA** when using Young's modulus equal to the ground-

[1]All results can be viewed at: youtu.be/2aqrGtLcrqU
[2]https://www.blender.org/



(a) Tracking deformation of the *cube* across different values for Young's modulus



(b) Tracking deformation of the *board* across different values for Young's modulus

Fig. 5: Comparison between the proposed approach (**PA**) and state-of-the-art (**SOA**) for the *cube* and the *board* sequence. The images show the **tracked model**, positioned in the pointcloud with the current estimate of ${}^{C}\mathbf{T}_{O}$, thereby causing the model to be partially occluded by the pointcloud. The small, red cubes are the model vertices. In the experiments, **SOA** fails to track accurately when YM is lower than the ground-truth value (5×10^4 Pa) and tends to overcompensate the deformation when YM is higher. **PA** maintains consistently accurate tracking, regardless of the changes in the value of YM

truth value for simulation (50000 Pa).

Real Data: The real data has been captured using the Intel RealSense SR300 RGB-D camera. It is a commodity level depth sensor that produces RGB-D images of reasonably good resolution between 20 to 150 cm distance. Three objects have been tracked for the validation: a) a *pizza*, b) a cuboidal soft *toy* and c) a rectangular *sponge*. A Poisson's ratio of 0.3 was assigned to all the objects uniformly. All

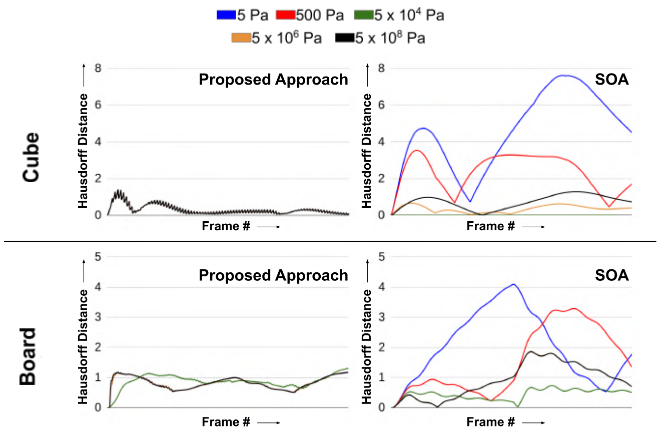


Fig. 6: Comparison of Hausdorff distance between output and ground-truth for **PA** and **SOA** for multiple values of Young's modulus

Young's Modulus:		**5 Pa**	**500 Pa**	$\mathbf{5 \times 10^4}$ **Pa**	$\mathbf{5 \times 10^6}$ **Pa**	$\mathbf{5 \times 10^8}$ **Pa**
Cube	**PA**	**0.3118**	**0.3118**	0.3009	**0.3009**	**0.3009**
	SOA	4.6448	2.3788	**0.0051**	0.3455	0.7383
Board	**PA**	**0.8239**	**0.8239**	0.8761	**0.8237**	**0.8246**
	SOA	2.0562	1.5674	**0.064**	0.9092	0.9081

TABLE I: Table for comparison of Hausdorff distance between output and ground-truth for **PA** and **SOA** for multiple values of Young's modulus

the objects were tracked using multiple values of Young's modulus which were set coarsely and empirically. The *pizza* data was tracked with an Young's modulus of 5×10^3 Pa, 5×10^4 Pa and 5×10^5 Pa, the *toy* was tracked using 5×10^4 Pa, 5×10^5 Pa and 5×10^6, and the *sponge* was tracked using 8×10^5 Pa, 8×10^6 Pa and 8×10^7 Pa. The tracking results are shown in **Fig.** 7. Given these results, the average, per pixel point-to-plane error for multiple values of Young's modulus are given in **Fig.** 8.

V. DISCUSSION

The approach proposed in this paper accurately tracked all the deforming objects that it had been experimented upon. The results are promising and the validation tests of the approach had been successful. This approach has shown significant improvement over equivalent methods seen in the literature. Using the proposed approach, FEM based deformation tracking no longer needs to be fine tuned for the precise physical parameters of the objects being tracked. The overall algorithm, especially the Jacobian computation, remains highly parallelizable. There remains significant scope for performance optimization, which could be an important future work.

VI. CONCLUSION

We present an algorithm to track deformable objects using RGB-D camera. The FEM based deformation tracking methods available in the literature suffers from the drawback of being excessively dependant on the accurate knowledge of the physical properties of the object being tracked. However, the approach proposed in this paper removes this dependency, while accurately tracking the deforming object. This will be beneficial when tracking deformation in objects with unknown physical properties. The algorithm is capable of real-time interaction using a parallelized architecture. The approach has been validated on simulated objects with ground-truth, as well on real objects of unknown physical properties. The simulated dataset with ground-truth and all real data used in this paper has been made publicly available for future research and comparison.

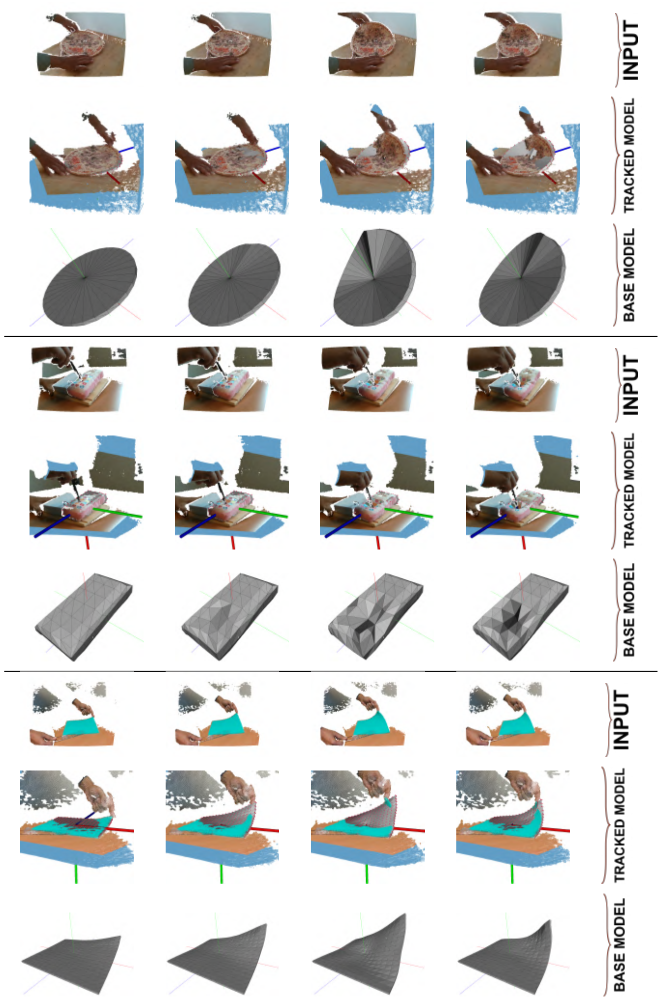
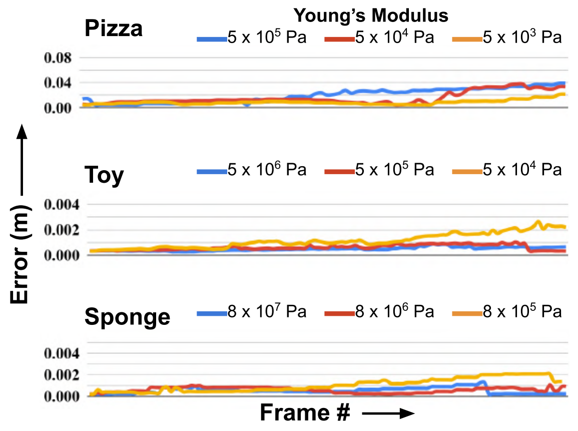


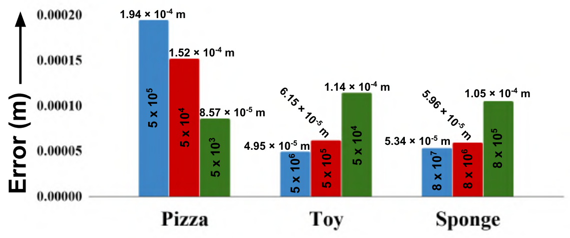
Fig. 7: The tracking out for the *pizza* (top), *toy* (middle) and *sponge* (bottom). **Input** denotes the pointcloud obtained from the RGB-D camera, projected on the image plane, the second row shows the **tracked model** placed inside the pointcloud with the current estimate of ${}^{C}\mathbf{T}_{O}$ and the third row depicts the **base model** separately

ACKNOWLEDGEMENT

We gratefully acknowledge the support from *The Research Council of Norway* through participation in iProcess—255596 project and from *Inria*.



(a) Variation of mean point-to-plane error for multiple values of Young's modulus, taken across consecutive frames



(b) The mean point-to-plane error across all the frames, plotted against the respective Young's Modulus (shown inside the bars, the values are in Pascal)

Fig. 8: Summary of point-to-plane error obtained from experiments on real data

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.

[2] H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 44–51. IEEE, 2000.

[3] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. In *ACM Transactions on Graphics (TOG)*, volume 26, page 80. ACM, 2007.

[4] L. Royer, M. Marchal, A. Le Bras, G. Dardenne, and A. Krupa. Real-time tracking of deformable target in 3d ultrasound images. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2430–2435. IEEE, 2015.

[5] A. Petit, V. Lippiello, and B. Siciliano. Tracking fractures of deformable objects in real-time with an rgb-d sensor. In *2015 International Conference on 3D Vision*, pages 632–639. IEEE, 2015.

[6] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 2007.

[7] H. Li, R. W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008.

[8] D. Lee and A. Krupa. Intensity-based visual servoing for non-rigid motion compensation of soft tissue structures due to physiological motion using 4d ultrasound. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2831–2836. IEEE, 2011.

[9] L. Royer, A. Krupa, G. Dardenne, A. Le Bras, E. Marchand, and M. Marchal. Real-time target tracking of soft tissues in 3d ultrasound images based on robust visual information and mechanical simulation. *Medical image analysis*, 35:582–598, 2017.

[10] A. Petit, V. Lippiello, and B. Siciliano. Real-time tracking of 3d elastic objects with an rgb-d sensor. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3914–3921. IEEE, 2015.

[11] T. Schmidt, R. A Newcombe, and D. Fox. Dart: Dense articulated real-time tracking. In *Robotics: Science and Systems*, 2014.

[12] C. Paloc, F. Bello, R. I Kitney, and A. Darzi. Online multiresolution volumetric mass spring model for real time soft tissue deformation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 219–226. Springer, 2002.

[13] N. Haouchine, S. Cotin, I. Peterlik, J. Dequidt, M. S. Lopez, E. Kerrien, and M. Berger. Impact of soft tissue heterogeneity on augmented reality for liver surgery. *IEEE transactions on visualization and computer graphics*, 21(5):584–597, 2015.

[14] A. Petit, S. Cotin, V. Lippiello, and B. Siciliano. Capturing deformations of interacting non-rigid objects using rgb-d data. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 491–497. IEEE, 2018.

[15] F. Nadon, A. Valencia, and P. Payeur. Multi-modal sensing and robotic manipulation of non-rigid objects: A survey. *Robotics*, 7(4):74, 2018.

[16] R. Newcombe, D. Fox, and S. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 343–352, 2015.

[17] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016.

[18] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1395, 2017.

[19] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, pages 283–321. Springer, 2012.

[20] R. D Cook. *Finite element modeling for stress analysis*. Wiley, 1994.

[21] C. A Felippa. A systematic approach to the element-independent corotational dynamics of finite elements. Technical report, Technical Report CU-CAS-00-03, Center for Aerospace Structures, 2000.

[22] S. Trinh, F. Spindler, E. Marchand, and F. Chaumette. A modular framework for model-based visual tracking using edge, texture and depth features. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 89–96. IEEE, 2018.

[23] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[24] A. E Beaton and J. W Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974.

[25] J. A Hartigan and M. A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[26] K Fukunage and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, (7):750–753, 1975.

[27] S. D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[28] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11, 2015.

[29] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.

[30] C. Choi and H. Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *IEEE Int. Conf. on Robotics and Automation, ICRA 2010*, pages 4048–4055. IEEE, 2010.

[31] J. Henrikson. Completeness and total boundedness of the hausdorff metric. *MIT Undergraduate Journal of Mathematics*, 1:69–80, 1999.