

Non-rigid Tracking Using RGB-D Data



Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano

Abstract In this chapter, the real-time non-rigid tracking system to continuously estimate the deformations of the manipulated objects is described, using visual and range data provided by an RGB-D sensor. Based on the models described in the previous chapter, the method enables to deal with various deformations (elastic and plastic), fractures, and contacts, while ensuring physical consistency, handling rigid motions, occlusions, and addressing these tasks in real-time. It relies on a prior visual segmentation of the object in the RGB image. The mesh is registered first in a rigid manner with a classical ICP algorithm between the visible surface of the mesh and the segmented point cloud. A non-rigid fitting phase is then performed by determining geometrical point-to-point correspondences with the point cloud, used to compute external forces exerted on the mesh. Deformations are computed by solving mechanical equations balancing these external forces with internal forces provided by the FEM models. A technique to estimate the elastic parameters of the object is proposed by minimizing a fitting error between the simulated deformations, actuated by the input operator force provided by a force sensor, and the deformations captured by the RGB-D camera. Conversely, estimating a contact force exerted on the object can be carried out using point cloud data by minimizing the deviation between the registered and the simulated deformations. The system has been evaluated on synthetic and real data, with various objects, deformation, and interaction scenarios, and by integrating it into manipulation experiments on the RoDyMan humanoid robotic platform. This chapter is based on the works presented in [1–3].

A. Petit (✉)

Therenga, 74F rue de Paris, 35000 Rennes, France

e-mail: antoine.a.petit@inria.fr

V. Lippiello · B. Siciliano

Department of Electrical Engineering and Information Technology, CREATE Consortium &

University of Naples Federico II, PRISMA Lab, Via Claudio 21, 80125 Naples, Italy

e-mail: vincenzo.lippiello@unina.it

B. Siciliano

e-mail: bruno.siciliano@unina.it

© Springer Nature Switzerland AG 2022

B. Siciliano and F. Ruggiero (eds.), *Robot Dynamic Manipulation*,

Springer Tracts in Advanced Robotics 144,

https://doi.org/10.1007/978-3-030-93290-9_2

1 Brief Introduction

The problem of capturing the deformations of non-rigid objects is an active research field in computer vision that finds prospective applications in robotic manipulation, human/machine, tangible interactions, and motion capture.

For the perception system of the RoDyMan platform, this objective of sensing deformations led to handle several challenges such as modelling the considered manipulated objects and their material and aligning the designed models with the vision and range data provided by an RGB-D sensor.

As introduced in the previous chapter, a mechanical model based on the FEM was employed. The recent suitability of these models for real-time registration and tracking problems, as demonstrated by promising approaches [4–7], has corroborated our choice. Thus, we assume the prior knowledge of consistent meshes of the considered objects (which could be automatically reconstructed offline) and their material properties (through the Young’s modulus and the Poisson’s ratio). As a vision sensor, we rely on an off-the-shelf RGB-D camera.

The deformable registration and tracking method that we propose consists of fitting in real-time, frame-by-frame, the known mesh of the considered object with the point cloud data provided by an RGB-D sensor. The basic idea is to derive

Table 1 Main symbols used in this chapter

Definition	Symbol
Input image	$\mathcal{I}$
Number of pixels	$N > 0$
Binary pixel label	α_i
Set of the binary pixel labels	$\alpha = \{\alpha_i, i = 1, \dots, N\}$
Pixel of the image	$\mathbf{p}_i \in \mathbb{R}^2$
Input depth map	$\mathcal{D}$
Segmented image	$\mathcal{I}^s$
Segmented depth map	$\mathcal{D}^s$
Number of points in the cloud	$N_Y > 0$
3D point in the sensor frame	$\mathbf{y}_j \in \mathbb{R}^3$
Segmented point cloud	$\mathcal{Y} = \{\mathbf{y}_j, j = 1, \dots, N_Y\}$
Number of vertices of the mesh	$N_X > 0$
Coordinates of a mesh’s vertex	$\mathbf{x}_j \in \mathbb{R}^3$
Set of the vertices of the mesh	$\mathcal{X} = \{\mathbf{x}_j : j = 1, \dots, N_X\}$
Set of the vertices of the visible surface	$\mathcal{X}_V$
Size of $\mathcal{X}_V$	$N_{\mathcal{X}_V} > 0$
Vector collecting all the vertices of the mesh	$\mathbf{x} \in \mathbb{R}^{3N_X}$
Young’s modulus	$E \in \mathbb{R}$
Poisson’s ratio	$\nu \in \mathbb{R}$

external forces exerted by the point cloud on the mesh and balance these forces with the internal forces computed from the FEM physical model by solving mechanical equations for the deformations. The FEM model thus acts as a regularizer over the computed deformations.

The external forces are computed from geometrical point-to-point correspondences between the point cloud and the mesh, relaxing the assumption of having a textured object or an object with a rough surface, for which 2D or 3D key points can be extracted and matched. We assume the mesh to be manually initialised correctly.

Based on this framework, other physical behaviours have been addressed besides pure elastic deformations, such as capturing fractures undergone by objects and interactions, which constitute an open research area in the field of 3D computer vision. For fractures, our approach involves the FEM physical model for fractures described in the previous chapter.

Several challenges must be overcome when considering multiple interacting objects to estimate the non-rigid objects' deformations jointly. The main challenge regards inter-occlusions between objects and deformations due to contacts. Tracking the different objects independently would result in inter-penetrations and physically unrealistic behaviours between objects due to inaccuracies and ambiguities in the visual or point cloud information. Instead, a simultaneous registration method relying on modelling the contacts between the objects would inherently handle these issues and reach a reliable behaviour.

As direct applications of this perception system for the RoDyMan project, we show its relevance when performing a simple planning task on the RoDyMan robotic platform (see Sect. 10). A method to estimate a point-wise force exerted on a deformable object is also described in Sect. 11.3 by using the external RGB-D sensor to confront the deformations measured by the registration technique with simulated deformations generated with respect to the force to be estimated. This method requires an accurate preliminary estimation of elasticity parameters of the material (Sect. 11), also based on the deviation between simulated deformations, generated from a measured force, and observed deformations, using RGB-D point clouds.

2 Related Work and Motivations

In this section, various works related to registration and tracking of non-rigid objects using vision sensors are reported, considering 3D/2D and 3D/3D modalities and the various estimation strategies to register isometric, elastic deformations, topological changes, and interactions, with regards to the motivations of the system designed for the RoDyMan project.

In the literature, the various approaches proposed to register deformable objects using vision and/or range data could be classified according to the underlying model of the considered object and its physical realism. Let us first clarify our scope and distinguish it from non-rigid reconstruction methods, for which at each frame provided by the vision/range sensor a single mesh is reconstructed as in [8, 9]. Instead, here

the goal is to continuously estimate the rigid transformations and the deformations undergone by the object, modelled by a known mesh.

2.1 Registration Using Implicit Physical Modeling

Based on implicit physical models, approaches in [10–12] use a 1D parametric curve or 2D splines models (B-splines, radial basis functions) to track deformable objects in monocular images. This method relies on minimizing an energy function, involving an external energy term related to some image features, and an internal energy term regularizing curvature, bending, or twisting, compelling the model to vary smoothly. Adapting these techniques to register with 3D shapes or surfaces in monocular images is much more complex since 3D deformations can imply ambiguous 2D transformations, resulting in an underconstrained problem. A first attempt in [13], relying on 3D splines and inspired by the work in [11], densely processes gradient features to compute the data energy term. Less ambiguous feature-based approaches [14] have been preferred, and additional constraints are often added to solve ambiguities. With point cloud data, methods in [15, 16] employ an RGB-D sensor to register the acquired point cloud to a surface mesh by minimizing an error function accounting for geometric or direct depth and color errors and a stretching penalty function for the mesh. By means of a NURBS parametrization [15] or an optimized GPU implementation [16], real-time performance can be achieved. Although these two systems have shown promising and impressive results, they are still bounded to isometric or slightly elastic deformations, employing regularization functions proportional to squared distances between nodes of the mesh. In contrast, we wish to model elastic in a more physically realistic manner to handle volumetric effects and more significant strains. Another limitation of these methods is that they process mesh-to-input point cloud correspondences in their data error functions and are thus sensitive to missing data or unobserved areas of the considered object due to occlusions. We consider in this chapter also correspondences from the input point cloud to the mesh, through the use of a segmentation method, to restrict the input point cloud to the observed areas of the object, and based on these correspondences, the occluded or unobserved areas would not affect registration.

We can also mention template-free non-rigid reconstruction methods for which at each frame provided by the vision/range sensor, a single mesh of the sensed scene is reconstructed and tracked, as in [8, 9, 17, 18]. However, these dense and exhaustive methods, besides often failing fulfilling real-time constraints, are limited to isometric slightly elastic deformations.

2.2 Registration Using Explicit Physical Modeling

Instead, another formulation of the problem relies on explicit physics-based deformable models to perform registration. Concerning implicit methods, other sorts (such as non-linear elasticity) and magnitudes of deformations can be treated, inferring more consistently shape and/or volumetric regularization. Statistically, the solution can then be determined by setting internal and external forces equal or, equivalently, minimizing energy functions. Physics-based methods include discrete mass-spring-damper systems [4, 7, 19], or more straightforward approaches relying on the FEM, based on continuum mechanics. In [7], based on a mass-spring-damper system, 3D-3D correspondences, determined through a probabilistic inference, enable the computation of the external forces applied to the mesh. However, mass-spring systems are limited in elastic deformations' magnitude and do not guarantee specific mechanical properties, such as volume preservation, failing to handle volumetric effects. First attempts for registration employing the FEM for 3D surfaces in [20, 21] used linear elasticity FEM models. More recently, registration in monocular images is addressed in [22] by designing a stretching/shrinking energy using continuous mechanical constraints on 2D elements assuming linear elasticity and some 3D boundary conditions. A linear tetrahedral co-rotational FEM model, coping with more significant elastic deformations, is used in [6] through external forces being related to correspondences between tracked 3D feature points mapped to the 3D mesh employing a stereo camera system. This system has been then extended in [5] by involving a non-linear St-Venant Kirchhoff FEM model to address hyperelastic behavior efficiently. At the same time, the registration was addressed through monocular keypoint-based external forces. This latter method proposes the most realistic physical elastic model within a real-time vision-based tracking system to the best of our knowledge. However, both approaches in [5] and [6] do not address issues such as capturing rigid motions and require textured materials.

2.3 Handling Topological Changes, Fractures and Cuts

Regarding the methods dealing with topological changes of a 3D deformable object, we can mention the 3D reconstruction methods in [23, 24] based on the *TransforMesh* system to address mesh evolution and topological changes. However, these are model-free methods, and they reconstruct a single mesh at each time step [24], or they are not suited for real-time applications [23]. More recently, some works [25, 26] addressed the issue of tracking cuts of deformable objects. They rely on the detection of changes in the configurations or structures of neighbouring visual features to predict a cut over a registered surface [26] or volumetric physics-based model [25], which is propagated on the considered meshes.

2.4 *Multiple Objects Registration*

Simultaneously estimating the 3D transformations undergone by multiple objects with vision data is already a well-studied problem for rigid or articulating objects. This consists of retrieving the 6D pose of all rigid objects or the various joint angles in articulated objects. Common approaches address the task by independently processing the different objects [27, 28]. Other works propose to register objects and capture interactions and cope with inter occlusions. In the case of articulated hand-hand, hand-object, and object-object interactions, methods are numerous [29–36]. In these works, the interactions are modelled through physics-based constraints, enabling to properly deal with collisions and occlusions, using single or multiple RGB or RGB-D cameras. However, to our knowledge, handling multiple deformable objects in interaction is quite a leftover problem. We can mention the system designed in [37], which introduces a collision detection process in its non-rigid reconstruction pipeline to deal with auto-collisions. However, this technique falls in the class of non-rigid surface reconstruction of the whole observed scene. Our method, instead, as a template tracking technique, would explicitly consider different known deformable interacting entities and in a volumetric manner.

2.5 *Motivations and Contributions*

Among the methods based on RGB-D or point cloud data having the closest goals, motivations, and constraints to ours, we can mention [6, 7, 15, 16]. Several contributions are proposed against them, such as handling various large deformations like elastic ones while ensuring physical consistency, handling rigid motions, occlusions, fractures, interactions, and addressing all these tasks with real-time or near real-time performance. For interactions, introducing the collision detection and response model into the registration framework has enabled to cope with contacts and with inter occlusions in the point cloud matching processes of the different considered objects. For this latter contribution, another improvement lies in the estimation of rigid transformations into the FEM resolution by integrating rigid ICP-based elastic forces. Another novelty has consisted of using the designed point cloud matching techniques and registration method for elasticity parameter and single contact force estimation and standard planning for manipulation on the RoDyMan platform.

2.6 *Overview of the System*

Our base tracking system for a single elastic object is represented in Fig. 1. As input data, we have the known 3D volumetric mesh of the object, a given RGB-D data. The procedure is sketched out as follows.

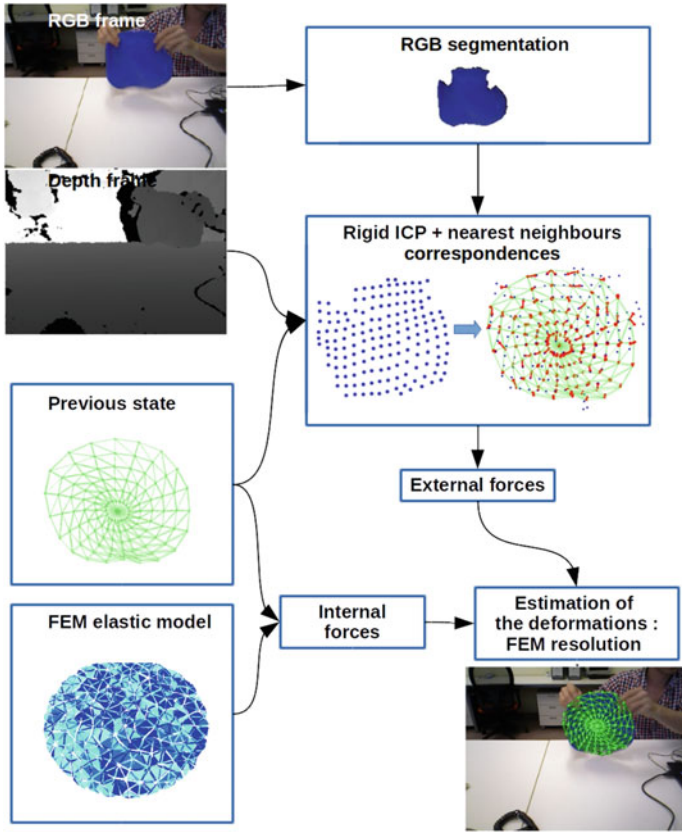


Fig. 1 Overview of our approach for deformable object tracking for a single object

1. Visual segmentation of the considered object, with a graph cut-based approach ensuring temporal coherence.
2. Using the resulted segmented point cloud, perform a rigid ICP to estimate a rigid transformation using closest point correspondences, from the point cloud to the mesh.
3. Using the resulting segmented point cloud, compute external linear elastic forces exerted on the vertices of the mesh from the point cloud to the mesh and conversely, based on closest point correspondences.
4. Numerical resolution of mechanical equations to compute the deformations, based on a tetrahedral linear co-rotational FEM model.

The different steps of this pipeline will be described in the following sections. The visual segmentation will be presented in Sect. 3, point cloud processing in Sect. 4, the rigid registration technique in Sect. 5, the non-rigid one in Sect. 6 and the resolution strategy will be introduced in Sect. 7.

Being the base for our different registration/tracking algorithms, these described methods will then be extended to the cases of fractures and interacting multiple object, as described respectively in Sects. 8 and 9. Finally, they will be applied for planning for robotic manipulation (Sect. 10), and for elasticity parameters and contact force estimation (Sect. 11).

3 Visual Segmentation

In this work, we advocate the use of a prior visual segmentation step in order to restrict each successively acquired point cloud to the object of interest (see Sect. 4 for a more detailed justification).

3.1 Grabcut Segmentation

We rely here on the efficient and widespread *GrabCut* method [38], based on graph cuts. In its original formulation, the *Grabcut* algorithm addresses the visual bilayer segmentation task as an energy minimization problem, based on statistical models of the foreground (the object) and the background.

For an input image $\mathcal{I}$, we denote within α , the set of the binary pixel labels, $\alpha_i = 0$ for the background pixels and $\alpha_i = 1$ for the foreground. The labels' estimation, $\hat{\alpha}$, can be formulated as the minimization of an energy-based Markov random field objective function, $E(\alpha)$, defined as

$$\hat{\alpha} = \arg \min_{\alpha} E(\alpha) \quad (1)$$

$$\text{with } E(\alpha) = E_{data}(\alpha) + \gamma E_{smooth}(\alpha), \quad (2)$$

$$\text{and } E_{data}(\alpha) = \sum_i U_i(\alpha_i), \quad (3)$$

where E_{data} is the data energy term, $U_i(\alpha_i)$ a unitary term accounting for the observation probability $p(\mathbf{p}_i | \alpha_i)$ for a pixel $\mathbf{p}_i$ to belong to the foreground or to the background, based on some image data (intensity, color, location, and so on) observed on the pixel, using the statistical models built for the background and the foreground. More formally, we have $U_i(\alpha_i) = -\log(p(\mathbf{p}_i | \alpha_i))$. Finally, E_{smooth} is the smoothness energy term whose goal is to favour smoothness or spatial coherence within the pixels.

In order to compute the optimal solution of the previous minimization problem and determine $\hat{\alpha}$, a *graph cuts* minimization algorithm [39] is employed, providing the segmented frame $\mathcal{I}^s$.

Statistical models for the data energy function are GMMs based on color distributions, learned for both the foreground and background layers, which are initially

determined by the user through a bounding box, manually defined around the foreground on the initial image. Besides, pixels outside this bounding box are definitely assigned to the background layer ($U_i(\alpha_i = 0) = \inf$), whereas inside their label is unknown, so that energy minimization only has effects inside the bounding box.

3.2 Temporal Coherence and Real-Time Issues

Once the initial image is segmented through user interaction, the following frames are treated similarly, except that, instead of segmenting the whole frame, the area to segment effectively is updated frame-by-frame around the silhouette contour of the segmented area, providing temporal coherence in the segmentation process. More precisely, as shown in Fig. 2, the silhouette contour of the previous segmented foreground is extracted and the distance transform is computed over it, providing a signed distance map $\mathbf{d}$ to these contours (negative outside, positive inside). According to a fixed threshold, $d_t > 0$, on this distance map, we define a narrow strip around the contour. Such a narrow strip (grey area) has the property that $d_i < d_t$, with $d_i > 0$ is an element of $\mathbf{d}$, and in which the label of a pixel $\mathbf{p}_i$ is unknown, with $U_i(\alpha_i) = -\log(p(\mathbf{p}_i | \alpha_i))$. On the other hand, the inner side of the strip is assigned to the foreground (white area), if $d_i < -d_t$ and by setting $U_i(\alpha_i = 1) = \inf$, otherwise it is assigned to the background (black area) if $d_i > d_t$ and by setting, in this case, $U_i(\alpha_i = 0) = \inf$.

In this manner, temporal consistency is ensured since energy minimization is only effective within this strip, in the vicinity of the previous segmentation boundary, avoiding some outliers outside or inside, and reducing significant computations. Let us note that the GMMs are determined initially, making this segmentation process valid for sequences for which color distributions of both the foreground and the background are assumed relatively constant, which is a fair assumption for the robotic manipulation applications considered in this chapter.

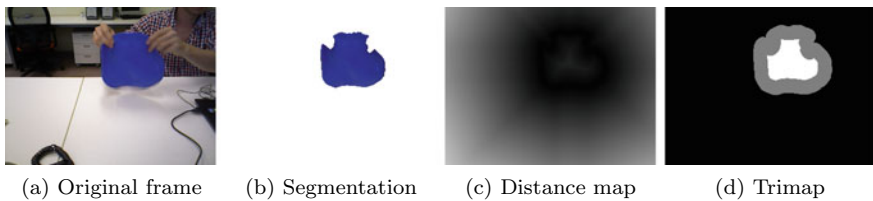


Fig. 2 Temporal consistency for segmentation. Segmentation will be effective on the strip (grey area on $\mathbf{d}$) around the contour of the previous segmented frame ($\mathbf{b}$), through the distance map to the contour ($\mathbf{c}$)

4 Segmented and Sampled Point Cloud

We use the acquired RGB image sequence to visually segment the object of interest from its background and occlusions. Since we do not rely on some distinctive visual features, the point cloud provided by the depth sensor is indeed restricted to the considered object to avoid ambiguities in the matching process with the background or with occluding shapes, and be able to process correspondences from the input point cloud to the mesh. At frame k , using $\mathcal{I}^s$, a segmented depth map $\mathcal{D}^s$ is obtained by aligning and intersecting the original input depth map $\mathcal{D}$ with the segmented area in $\mathcal{I}^s$. Then, by back-projecting $\mathcal{D}^s$ in the sensor frame, the desired segmented point cloud $\mathcal{Y}$ is determined. For computational purposes, we limit the size of $\mathcal{Y}$ by firstly sampling $\mathcal{D}^s$ on a regular grid in the image plane.

5 Rigid Registration

The first step in our method is to register $\mathcal{Y}$ in terms of rigid translation and rotation transformations, initially considering the mesh of the object as rigid. Let us consider the set of vertices of the mesh $\mathcal{X}$, initially in their previous computed states, for the frame $k - 1$. We suggest a classical rigid ICP algorithm [40] between $\mathcal{Y}$ and the vertices of the visible surface $\mathcal{X}_V$ of the mesh, transformed with respect to the previous RGB-D data. $\mathcal{X}_V$ is determined by performing a visibility test on the rendered depth map of the projected 3D mesh of the object, for the frame $k - 1$. Through this procedure, which converges rapidly, fast rigid motions can be tracked and a fair initialisation for the non-rigid process can be obtained.

6 Point Cloud Matching for Non-rigid Registration

In order to register the segmented point cloud with the mesh in a non-rigid manner, we suggest an ICP-like procedure, given the sets $\mathcal{X}$ and $\mathcal{X}_V$ updated by the rigid transformation estimated as described above.

6.1 Nearest Neighbor Correspondences

By means of a k -dimensional tree search, nearest neighbour correspondences are determined both from the segmented point cloud to the visible surface of the mesh and from the visible surface of the mesh to the segmented point cloud. This step provides us with the sets of nearest neighbours $\mathcal{N}_{\mathcal{X}_V} = \{\mathbf{NN}_{\mathcal{Y}}(\mathbf{x}_j) : \mathbf{x}_j \in \mathcal{X}_V\}$ and

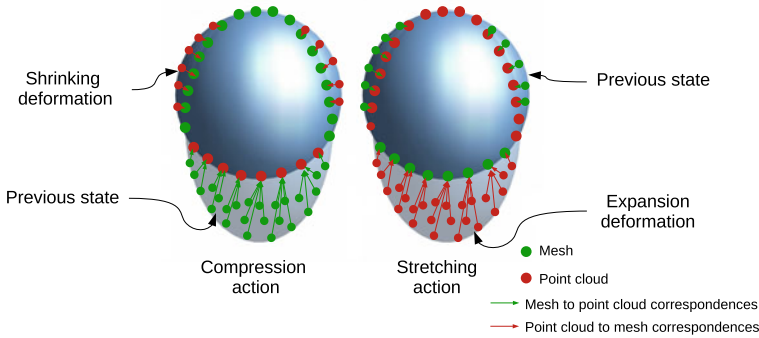


Fig. 3 Point cloud to mesh and mesh to point cloud correspondences. With shrinking deformations, mostly mesh to point cloud correspondences can attract the mesh towards the compressed point cloud, and conversely for expansion deformations

$\mathcal{N}_{\mathcal{Y}} = \{\mathbf{NN}_{\mathcal{X}}(\mathbf{y}_j) : j = 1, \dots, N_Y\}$ in $\mathcal{Y}$ for $\mathcal{X}_V$, with the 1-NN function $\mathbf{NN}_{\mathcal{Y}}$, and in $\mathcal{X}_V$ for $\mathcal{Y}$, with the 1-NN function $\mathbf{NN}_{\mathcal{X}_V}$, respectively.

Both sets of correspondences are employed since relying on the sole geometrical proximity may lead to inconsistent matches using single point-to-point matches. Indeed, as illustrated in Fig. 3, from the segmented point cloud to the mesh, the correspondences enable to track expansion deformations under stretching forces, for which the observed segmented point cloud $\mathcal{Y}$ would spread over the visible surface of the mesh $\mathcal{X}_V$. The extended areas of $\mathcal{Y}$ with respect to $\mathcal{X}_V$ can be matched with the outer areas of $\mathcal{X}_V$ (red arrows on the right side of Fig. 3). These correspondences also enable to deal with occlusions and segmentation errors since the corresponding unobserved areas of the object would not affect the underlying areas of $\mathcal{X}_V$. Conversely, from $\mathcal{X}_V$ to $\mathcal{Y}$, the correspondences are instead more suited to track shrinking deformations under compression actions. The outer areas of $\mathcal{X}_V$ are coherently matched with the outer areas of the observed point cloud $\mathcal{Y}$ of the compressed object (green arrows on the left side of Fig. 3). As a drawback, unobserved areas (occlusions and segmentation errors) would affect the underlying areas $\mathcal{X}_V$ which would match with the closest areas of $\mathcal{Y}$.

As described hereafter, a trade-off must be found between these two sets of correspondences, whether the application deals with stretching or compression actions on the object and whether occlusions or segmentation errors are to be dealt with.

6.2 Computation of External Forces

Based on the two sets of mesh-to-point cloud and point cloud-to-mesh correspondences, given by $\mathcal{N}_{\mathcal{X}_V}$ and $\mathcal{N}_{\mathcal{Y}}$, an external elastic force $\mathbf{f}_{ext}(\mathbf{x}_j) \in \mathbb{R}^3$ can be computed for each $\mathbf{x}_j \in \mathcal{X}_V$ as follows

$$\mathbf{f}_{ext}(\mathbf{x}_j) = k_{ext}(\mathbf{x}_j - \mathbf{y}_j^f) \quad (4)$$

with $k_{ext} > 0$ the stiffness of these external spring elastic forces and

$$\mathbf{y}_j^f = \begin{cases} \lambda \mathbf{NN}_{\mathcal{Y}}(\mathbf{x}_j) + (1 - \lambda) \frac{1}{N_K} \sum_{\mathbf{y}_j \in \mathcal{K}_j} \mathbf{y}_j & \text{if } N_K > 0, \\ \lambda \mathbf{NN}_{\mathcal{Y}}(\mathbf{x}_j) + (1 - \lambda) \mathbf{x}_j & \text{if } N_K = 0, \end{cases} \quad (5)$$

where $\lambda \in \mathbb{R}$ is a fixed scalar, $\mathcal{K}_j = \{\mathbf{y}_i \in \mathcal{Y} : \mathbf{NN}_{\mathcal{X}_v}(\mathbf{y}_i) = \mathbf{x}_j\}$ is the set of points in $\mathcal{Y}$ whose nearest neighbours are $\mathbf{x}_j$, $\mathbf{y}_j^f$ acts as a virtual point attracting $\mathbf{x}_j$, and $N_K \geq 0$ is the size of $\mathcal{K}_j$. The component of $\mathbf{y}_j^f$ resulting from the multiple point cloud to mesh correspondences is scaled by the reciprocal of N_K to be homogeneous with the component resulting from the single mesh to point correspondence. The fixed scalar λ tunes the balance between the mesh-to-point cloud and point cloud-to-mesh correspondences, as a trade-off between the stretching or compression actions to be tracked, as suggested in Sect. 6.1. If $\mathcal{K}_j$ is empty, the missing point cloud-to-mesh correspondences are replaced by a self-contribution for the vertices $\mathbf{x}_j$, compelling it to remain at its current position. The stiffness k_{ext} can be set accordingly to the Young's modulus, using the equivalence between a mass spring model and a linear FEM model, as described in Sect. 7.2. However, modelling this way the external forces, using equivalent springs, can be coarse since vision data affects the visible vertices only. Some outliers in the point cloud may result in aberrant correspondences and thus in aberrant forces exerted on some vertices. A simple solution has been to discard points in the cloud whose point-to-point distances with their nearest neighbors in the mesh are above a certain threshold with respect to the mean value and the standard deviation of the whole set of point-to-point distances. In this case, for the considered vertices $\mathbf{x}_j$, we have $\mathbf{f}_{ext}(\mathbf{x}_j) = 0$.

Finally, regarding points $\mathbf{x}_j$ in $\mathcal{X}$ which are not visible, we also set $\mathbf{f}_{ext}(\mathbf{x}_j) = 0$. The whole set of forces is finally concatenated in a vector $\mathbf{f}_{ext}$ of size N_X .

6.3 Weighting Forces Using Contours

A limitation of this method lies in tracking large elastic deformations due to stretching efforts, for instance. In this case, since correspondences are established based on 3D geometry, only vertices lying on the occluding contour of the mesh are attracted to the extended area in the point cloud. As a consequence, the forces attracting the contours are weak. We propose to emphasize them by weighting the vertices of the visible surface of the mesh given their distance to the occluding contour of the projected mesh. Based on the depth map $\mathcal{D}^M$ of the projected mesh, we compute the distance map of the occluding contour of the mesh. Then, the weight $w_j \in \mathbb{R}$ for the vertex $\mathbf{x}_j$ is set proportional to

$$w_j \propto e^{-\frac{d_j^M}{\sigma}}, \quad (6)$$

where $d_j^M > 0$ is the distance from $\mathbf{x}_j$ to the nearest contour of the projected mesh, $\sigma > 0$ is a parameter which is empirically set. Then, w_j is normalized so that we get an observation probability. Finally, the forces are computed as

$$\mathbf{f}_{ext}(\mathbf{x}_j) = w_j k_{ext}(\mathbf{x}_j - \mathbf{y}_j^f). \quad (7)$$

7 Solver

Estimating the deformations of the mesh consists in solving a dynamic system of ordinary differential equations, involving the internal and external forces, based on Lagrangian dynamics

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{f} = \mathbf{f}_{ext} \quad (8)$$

$$\text{with } \mathbf{f} = \mathbf{K}\mathbf{x} + \mathbf{f}_0, \quad (9)$$

where $\mathbf{M} \in \mathbb{R}^{3N_x \times 3N_x}$ and $\mathbf{C} \in \mathbb{R}^{3N_x \times 3N_x}$ are the mass and damping matrices, respectively, $\mathbf{K} \in \mathbb{R}^{3N_x \times 3N_x}$ is the global stiffness matrix which sums the element-wise *rotated* stiffness matrices $\mathbf{K}_e = \mathbf{R}_e \mathbf{K}_e \mathbf{R}_e^{-1} \in \mathbb{R}^{3N_x \times 3N_x}$, written with respect to whole set of vertices, and $\mathbf{f}_0 = \mathbf{R}_e \mathbf{K}_e \mathbf{x}_0$ the corresponding global offset summing the element-wise ones, with $\mathbf{x}_0 \in \mathbb{R}^{3N_x}$ the position of the vertices before the deformation. An Euler implicit integration scheme is used to solve the system, along with a conjugate gradient method. Finally, $\mathcal{X}$ can then be updated using the resulting estimated positions $\mathbf{x}$ of the vertices of the mesh.

Algorithm 1 summarizes the whole process presented above.

7.1 Experimental Results

In order to evaluate the performance of our method and contributions, some experimental results are shown in this section, on some computer-generated data, and some real data. Various objects, deformations, and conditions are tested.

For the non-rigid registration phase, we have employed the SOFA simulator [41], which enables us to deal with various physical models and to evolve simulations in real-time.

Algorithm 1: Registration process

Data: Tetrahedral mesh of the object, RGB-D data

Result: Fitting the mesh with the RGB-D data

```

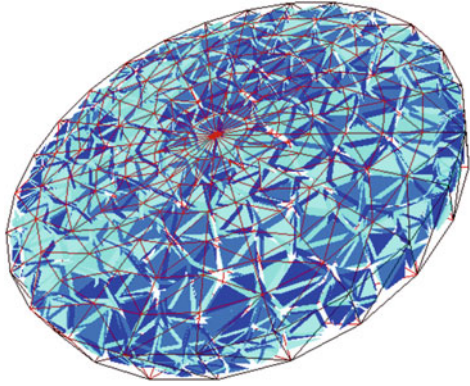
1 Initialization of the vertices  $\mathcal{X}$  in the mesh;
2 Initialization of segmentation;
3 for each new  $\mathcal{I}$  and  $\mathcal{D}$  do
4   Segmentation in  $\mathcal{I}$ , providing  $\mathcal{D}^s$  ;
5   Sampling and back-projection, giving  $\mathcal{Y}$ ;
6   Determine  $\mathcal{X}_V$ ;
7   Rigid ICP between  $\mathcal{X}_V$  and  $\mathcal{Y}$ ;
8   Rigidly transform  $\mathcal{X}$ ;
9   Non-rigid process:
10  3D correspondences from  $\mathcal{Y}$  to  $\mathcal{X}_V$ ;
11  3D correspondences from  $\mathcal{X}_V$  to  $\mathcal{Y}$ ;
12  Computation of external forces  $\mathbf{f}_{ext}$ ;
13  In SOFA simulator:
14  Computation internal forces  $\mathbf{f}$ ;
15  Computation of the new positions  $\mathbf{x}$ :
16  by solving  $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{f} = \mathbf{f}_{ext}$  (Euler implicit + conjugate gradient);
17 end
  
```

7.2 Results for Tracking on Synthetic Data

Relying on the SOFA simulator, we have first generated a sequence involving the deformations of a cylindrical elastic object, modeled by the FEM corotational approach. The sequence involves the elastic deformations of a cylindrical elastic object modelled by the FEM corotational approach. It has a Young's modulus of $E = 800$ Pa and Poisson's ratio of $\nu = 0.3$. Based on a manually designed cylindrical surface triangular mesh with radius/height dimensions of 0.11×0.02 m, the volumetric tetrahedral mesh was generated using the 3D Delaunay triangulation through the CGAL library.¹ A disordered placement of the nodes has been adopted here on this simple cylindrical geometric topology, not to lose generality to an arbitrary object topology. A 3D Delaunay triangulation would be preferred as a mesh generation routine. The resulting mesh is made of 1369 elements and 497 vertices and featured in Fig. 4a.

In order to simulate deformations and generate the considered sequence, an elastic stretching force is applied in the $-z$ direction of $\mathcal{W}$ (see Fig. 5), on one point on the border of the object (point 1) and few other points being fixed on the opposite border (points 4,5,6). Two compression forces are applied along y and $-Y$ (points 2,3) directions in $\mathcal{W}$. The applied forces result in a fast elongation deformation of the object, with a maximum elongation above 50%, and in a critical bending deformation along y -axis, as it can be seen in Fig. 7. For the tracking phase, segmentation aspects are not considered in these experiments. We only process the visible vertices of the rendered object in the sequence, and as ground truth, the positions of the whole set of

¹ <http://www.cgal.org>.

Fig. 4 Processed mesh

points are stored for evaluation. Results can be visually observed in Fig. 5, featuring the original target (red) and the tracking 3D mesh (blue), and in Fig. 6 the 3D errors between the vertices of the registered mesh and the corresponding points in the point cloud are plotted.

The following models and methods have been compared:

- volumetric mass-spring model;
- standard FEM model;
- corotational FEM model;
- corotational FEM model along with contour weighting.

On these results, we set the Young's modulus and the Poisson's ratio for the FEM approaches equal to the values assigned for the simulated object ($E = 800$ Pa, $\nu = 0.3$). We set $\lambda = 0.5$, balancing point cloud to mesh and mesh to point cloud forces. Also, we set $k_{ext} = 10 \text{ Nm}^{-1}$ for the stiffness of the external forces, which appeared to provide satisfying results. We consider a volumetric mass-spring model on the tetrahedrons of the generated mesh by attaching springs to connected vertices.

Determining the spring parameters of a mass-spring model equivalent to the FEM model in terms of elasticity behavior is not trivial, especially when dealing with unordered and irregular tetrahedrons, as pointed out in [42, 43]. Using the analytical expression proposed in [43] to derive the stiffness $k_m(i, j)$ of an equivalent spring between connected vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ s of the tetrahedral mesh

$$k_m(i, j) = \sum_e \frac{2\sqrt{2}}{25} \left(v_e \frac{12}{\sqrt{2}} \right)^{\frac{1}{3}} E, \quad (10)$$

with the sum over the elements e adjacent to the edge (i, j) . The resulting stiffnesses are too large to register the investigated deformations. The performance of a mass-spring model with spring stiffnesses scaled by 0.05 (defined as k_{tuned}), which appeared to fit best on this sequence, is thus also shown.

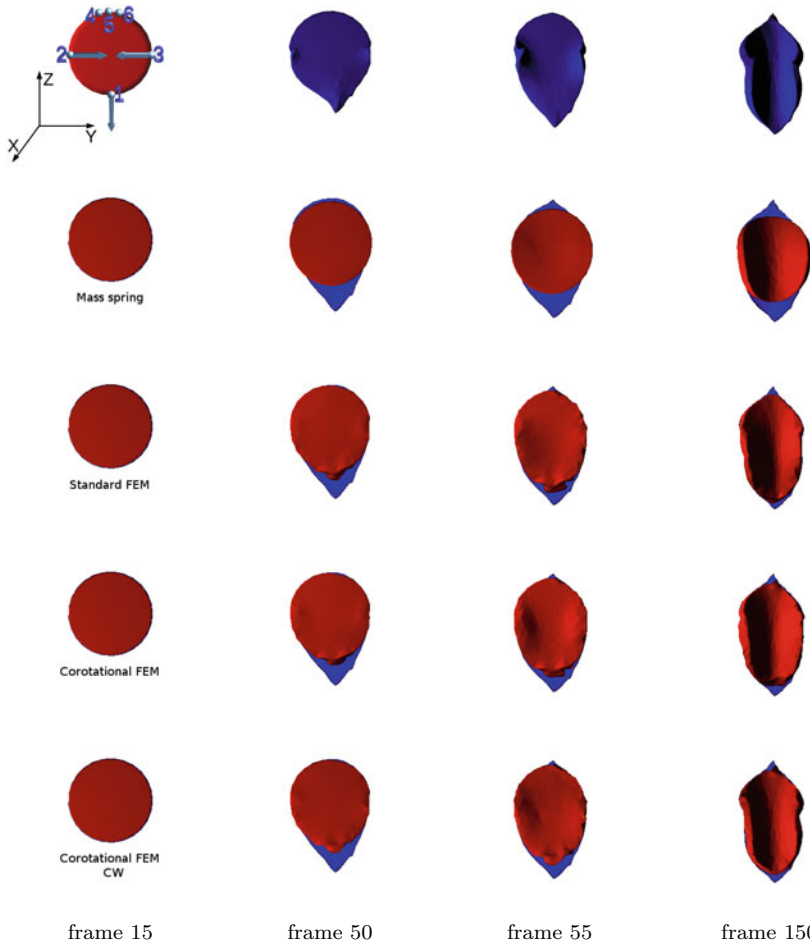


Fig. 5 Results of the deformable tracking process. In the first row, it is featured the ground truth. In the second row, the tracking with the equivalent mass spring system is displayed. In the third row, the linear FEM is depicted. In the fourth row, the corotational FEM is shown. Finally, in the fifth row, the contour weighting is added

We observe that the equivalent mass-spring model outperformed by the FEM approaches and the mass-spring model with manually tuned stiffnesses. This latter performs worse than the FEM models on the first frames when triggering the deformations (frame 50) due to the stretching deformations, which are severe over this phase. However, the standard linear FEM tends then to fail to track the large bending deformations which occur until steady-state, enhancing the sensitivity of this approach regarding rotation transformations within elements, as it can also be observed in Fig. 7. On the whole sequence, the benefit of using a co-rotated approach

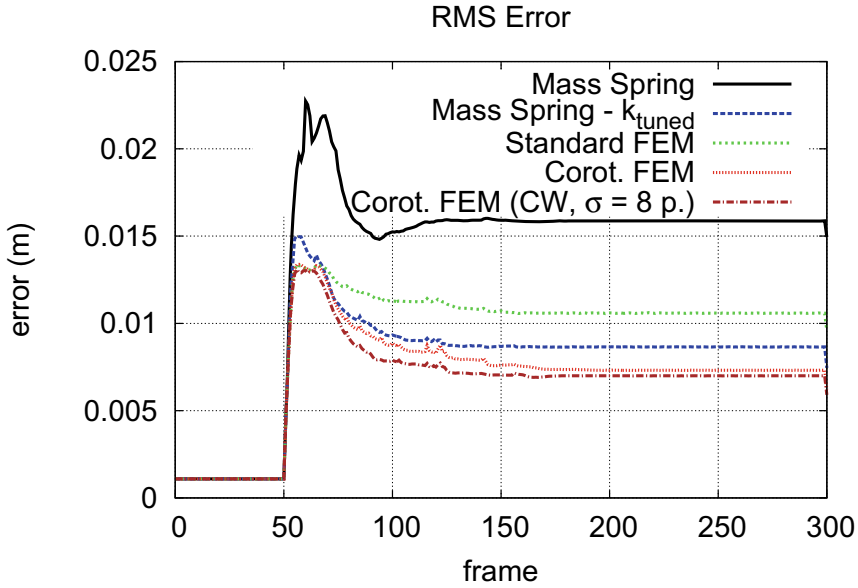


Fig. 6 Errors of the deformable tracking process with the different tested approaches, the mass-spring system with the equivalent stiffness, the mass-spring system with tuned stiffness (k_{tuned}), the standard FEM, the corotational FEM, and the corotational FEM with contour weighting (with $\sigma = 8$ pixels)



Fig. 7 Deformations on the generated sequence (a), on the tracked mesh with the standard FEM (b), and with the corotational FEM (c)

and the advantage of using the contour weighting function (with $\sigma = 8$ pixels) to track significant bending and stretching deformations can thus be particularly stressed out.

In order to evaluate the sensitivity with respect to tuning parameters, in Fig. 8 the errors for different external stiffnesses k_{ext} and for different values of λ are represented. For the external stiffness, the value of k_{ext} acts as a gain. It tunes the balance between tracking large stretching deformations, especially on the silhouette borders, and regularization effects. Its influence is especially observable on the large initial deformations. With λ a good compromise is found with $\lambda = 0.7$, showing the relevance of point cloud to mesh correspondences in this case for which stretching actions prevail.

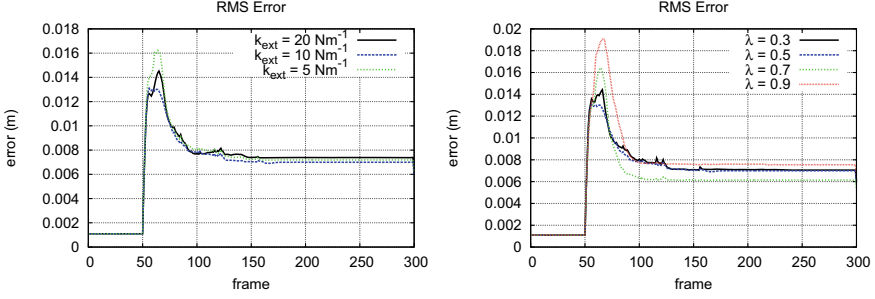


Fig. 8 Errors with respect to k_{ext} (a) and λ (b), tuning the balance between mesh to point cloud and point cloud to mesh forces

7.3 Results on Real Data

In order to carry out experiments on real data, the point cloud of the investigated scene is acquired from a calibrated RGB-D camera Asus Xtion, 320×240 RGB. Depth images were processed using a standard laptop with an NVIDIA GeForce 720M graphic card, along with a 2.4GHz Intel Core i7 CPU. Here, the segmentation process is involved in the loop and, since fast real-time performance is required, it relies on a CUDA implementation. The results presented here deal with a textureless and smooth elastic object made with silicon.

Concerning the “pizza-like” object, the idea was to test motions and deformations similar to those involved in the pizza-making process. The involved mesh is generated in a similar manner to Sect. 7.2, with radius/height dimensions of 0.12×0.01 m, and consisting in 574 vertices and 1675 elements. Note that the point cloud’s resolution is chosen through sampling to coincide approximately with the resolution of the visible mesh.

On the first featured sequence, the object undergoes large rigid motions and various isometric and elastic deformations. The Young’s modulus and the Poisson’s ratio of the material are unknown and good compromises have been found with $E = 300$ Pa and $\nu = 0.3$, by setting $k_{ext} = 1 \text{ Nm}^{-1}$.

Qualitative results are presented in Fig. 9, comparing our method using the corotational FEM approach with other models. In the first row, input RGB images are shown. The second row features the corresponding segmented frame, while the third row shows the 3D mesh tracking the object with the mass-spring model. The fourth row shows the standard FEM model, while the fifth shows the proposed approach (with $\sigma = 30$ pixels).

We can visually notice the ability of the proposed method to segment the visible part of the object correctly. Besides, it tracks rigid motions, in contrast to the mass-spring, to accurately register stretching deformations. Finally, in contrast to the standard linear FEM, it is robust to folding/bending deformations thanks to the

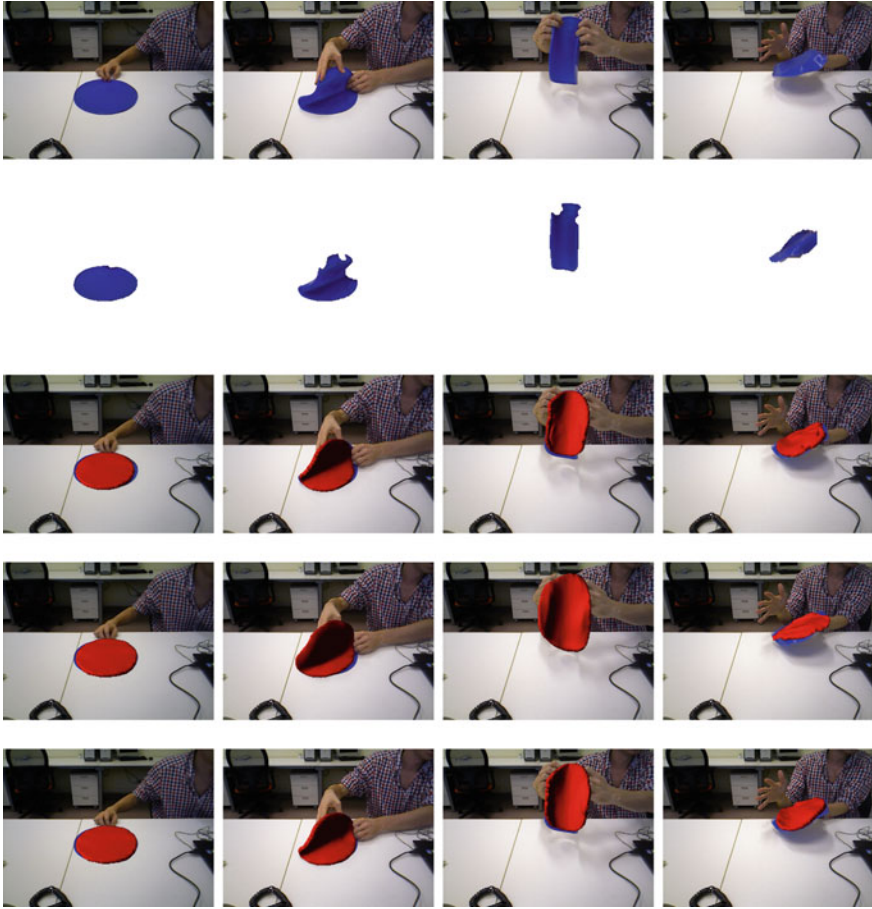


Fig. 9 Results of the tracking process for the pizza-like object, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image, for: the mass-spring model (third row), the standard FEM model (fourth row), and the corotational model with the contour weighting technique (fifth row)

corotational approach. Robustness to occlusions due to the hands manipulating the object (third column in Fig. 9) or segmentation errors can also be observed.

The second sequence has been worked out to stress the robustness of the proposed method under large elastic deformations due to stretching actions. The results for the proposed method are shown in Fig. 10, while they are compared to different models during stretching in Fig. 11. Since the object is essentially stretched along the image plane, involving few bending deformations along the optical axis, we propose an evaluation criterion to compute the mean 3D distance between the contour of the projected mesh in the image and the contour of the segmented silhouette of the object. This is achieved by searching from the 3D vertices, corresponding to the

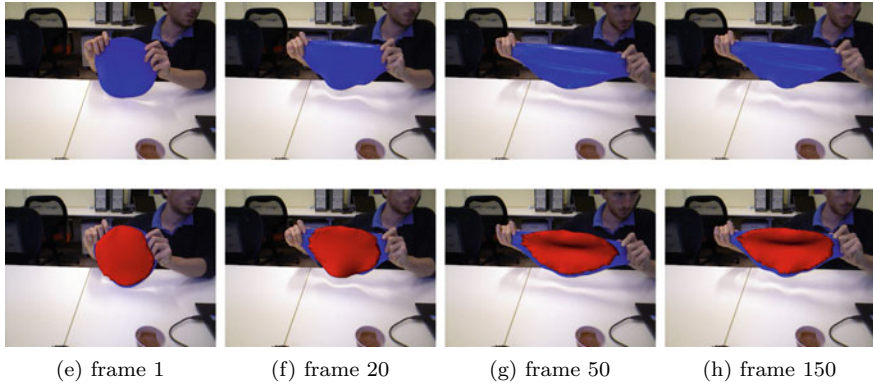
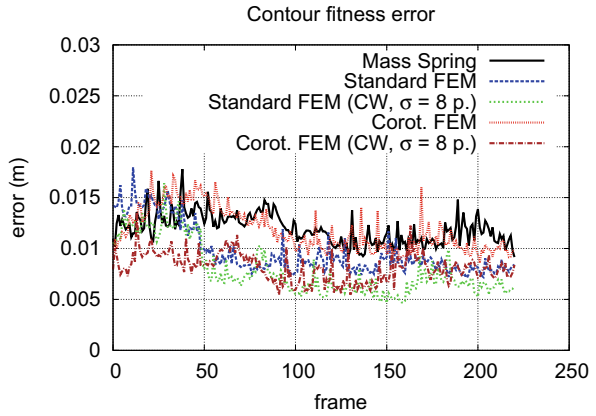


Fig. 10 Results of the tracking process for the silicon pizza, with the input images (first row), and the tracking results in the second row



Fig. 11 Results of the tracking process for the silicon pizza during stretching for the mass spring model (a), the standard FEM model, with (b) and without (c) contour weighting, with the corotational model, with (d) and without the contour weighting technique (e), on frame 150

Fig. 12 Contour fitting error for the different tested approaches (mass spring, standard and corotational FEM)



extracted 2D contour of the mesh, for the nearest 3D points, corresponding to the 2D contour of the segmented silhouette. This way, the benefit of using the corotational approach along with the contour weighting function can also be stressed out (see Fig. 12).

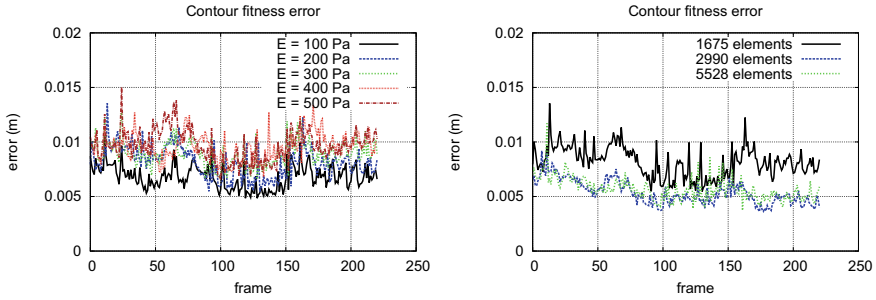


Fig. 13 Contour fitting error, for different stiffnesses (a) and resolutions (b)

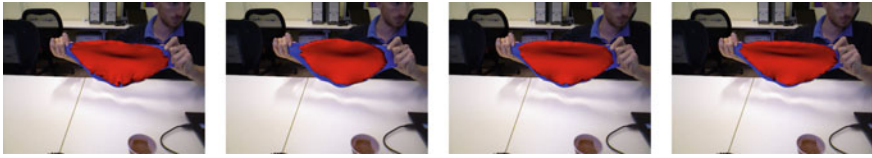


Fig. 14 Results of the tracking process with the proposed approach, for $E = 100$ Pa (a) and $E = 500$ Pa (b) with the standard resolution, and for resolutions of 2990 (c) and 5528 (d) elements, with $E = 100$ Pa

Concerning sensitivity to material and tuning parameters, different stiffnesses of the material have been tested, $E = 300$ Pa appearing, as stated before, to be a good compromise between the ability to register large deformations, while providing sufficient regularization and robustness to noise, which can result in spurious registrations, and self-collisions, especially around the borders of the mesh, as seen in Fig. 14a, b with the contour fitting errors. Different resolutions of the mesh have been tried adjusting the resolution of the point cloud accordingly through the manually set sample step. It can be observed in Fig. 13b that higher resolutions manage to fit better with the contour, and also, as observed in Fig. 14c, d to capture higher frequency details, up to a certain point for which the benefit becomes minor.

7.3.1 Computational Costs

Regarding computational aspects, the computation times of the various phases of the algorithm, for the different methods compared in this chapter, are shown in Table 2. *Visibility* corresponds to the process of determining the visible vertices of the rendered mesh and, in the case of using the contour weighting mode, extracting the vertices lying on the contour. *Ext. forces* is the step involving the determination of the closest points between the mesh and the point cloud and the computation of the subsequent external forces exerted on the mesh. *Resolution* consists instead of the resolution of the Lagrangian mechanical equations to compute the deformations. The presented figures are the averages of the execution times per frame (in milliseconds)

Table 2 Execution times, in milliseconds, for the different phases of the approach, and the various models and methods employed in this paper

	Mass Spring	Stand. FEM	Corot.	Corot. - CW
Segmentation	10.7	10.5	10.7	10.7
Rigid ICP	3.0	2.5	2.7	2.6
Visibility	8.1	8.2	7.6	7.4
Ext. forces	3.4	3.5	3.5	4.0
Resolution	2.8	3.3	4.0	4.1
Total	28.0	27.9	28.6	28.8

for the sequence presented in Fig. 9. As noticed, the suggested method (corotational model with the contour weighting mode) runs on the sequence at around 35 frame per second. We can also observe that the computational costs for the resolution phase being relatively small within the whole process, overall execution times are relatively independent of the selected model.

8 Fractures

8.1 A Pure Physics-Based Approach

For fractures, once deformations are computed, topological changes due to fractures, as modelled in the previous chapter, are handled. For each vertex $\mathbf{x}_j \in \mathcal{X}$, we detect if it is a fracture node, based on Sect. 1.4.1. Fracturable nodes undergo the remeshing procedure presented in Sect. 1.4.2. Let us note that, for remeshing, an element attached to different fracturable nodes is treated only once by the first investigated node. At the end of the procedure, a new mesh $\mathcal{X}$ is obtained and it will be directly employed for the next frame.

8.2 Experimental Results

In order to evaluate the performance of our method and contributions, some experimental results are shown in this section, in a qualitative manner on some real data.

8.2.1 Implementation and Experimental Set-Up

For the non-rigid registration phase, we have employed the SOFA simulator [41], which enables us to deal with various physical models and to evolve simulations

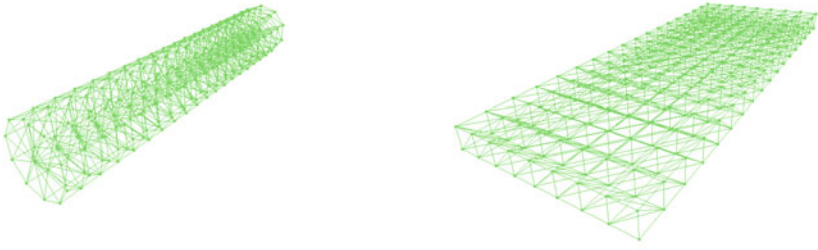


Fig. 15 Meshes of both the cylindrical bar (left) and plate (right) objects

in real-time. In order to carry out experiments on real data, the point cloud of the investigated scene is acquired from a calibrated RGB-D camera Asus Xtion, 320×240 RGB and depth images being processed. A standard laptop with an NVIDIA GeForce 720M graphic card has been used, along with a 2.4 GHz Intel Core i7 CPU. The segmentation process relies on a CUDA implementation. The results presented here deal with elastic bar and plate objects made of modeling clay. For the cylindrical bar object, the involved mesh, depicted in Fig. 15a, has a circumferential/radial/height resolution of $10 \times 20 \times 2$, resulting in 220 vertices and 570 tetrahedral elements. The material is here poorly elastic, the Young's modulus being empirically set to $E = 2.0$ and the Poisson's ratio to $\nu = 0.01$. The plate object (see Fig. 15b) instead has a length/width/height resolution of $18 \times 9 \times 2$, resulting in 324 vertices and 816 elements. For material properties, we set $E = 2.5$ and $\nu = 0.01$, in this case. Three fracture cases are demonstrated here: fracture due to opening by excessive bending for the bar object and fractures due to shearing and opening by stretching for the plate object, tearing the object. Let us note that in the first case the parameter $0 \leq \lambda \leq 1$, which tunes the balance between mesh-to-point cloud and point cloud-to-mesh forces (see [44]), is set to $\lambda = 0.5$. In the second case, $\lambda = 0.1$, so that mesh's vertices lying on the fracture crack of the object will be driven towards the nearest observed point on the border of the crack. Let us note that in [44], the registration process is iteratively repeated (three iterations were performed in the experiments). Here, only one iteration is executed to maintain real-time performance since fracture detection remains costly.

8.2.2 Results

As seen in Fig. 16, opening though bending deformations can be tracked quite correctly for the bar object. In this way, internal tensile forces computed on the nodes remain physically consistent, making the fracture model valid, as shown in the third column. For the plate object, tearing it through out-of-plane shearing (see Fig. 17) can also be tracked, despite some artifacts that can be noticed on some unfractured



Fig. 16 Results of the tracking process for the cylindrical bar object, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image (third row)

regions. However, when opening the plate object by stretching it, as shown in Fig. 18, only vertices lying on the occluding contour of the mesh are attracted to the expanded areas in the point cloud since correspondences are established based on 3D geometry, without any discriminative descriptors. As a consequence, forces attracting the contours and stretching the mesh are weak. Therefore, in this case, fracture appears with an unavoidable delay when both fractured parts split away. However, the fracture crack is relatively coherent, and both parts are finally consistently recovered.

Regarding computational aspects, in Table 3 are shown the mean computation times of the various phases of the algorithm for the sequence presented in Fig. 17. *Ext. forces* is the step involving the determination of the closest points between the mesh and the point cloud and the computation of the subsequent external forces exerted on the mesh. *Resolution* corresponds to the computation of the internal forces and in the resolution of the Lagrangian mechanical equations to compute the deformations based on the computed external and internal forces. *Fracture detection* consists of computing the separation tensors to detect fractures on the nodes of the mesh. Finally, *Remeshing* shows the time dedicated to remesh the mesh around the fracturable nodes. As noticed, the suggested method runs on this sequence at around 27 frame per second.

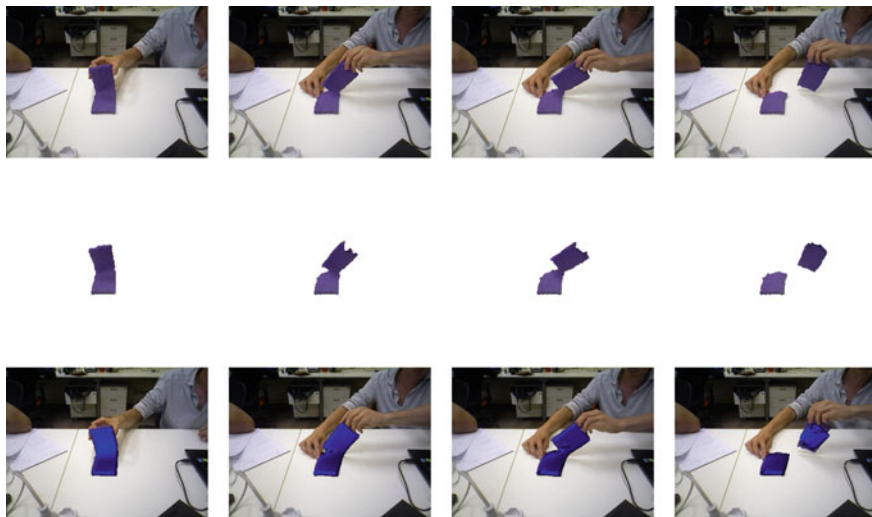


Fig. 17 Results of the tracking process for the plate object, torn through out-of-plane shearing, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image (third row)



Fig. 18 Results of the tracking process for the plate object, torn through opening, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image (third row)

Table 3 Mean execution times, in milliseconds, for the different phases of the approach

Phases	Mean execution times (ms)
Segmentation	10.1
Ext. forces	3.1
Resolution	2.3
Int. forces and fracture detection	2.3
Remeshing	0.6
Total	37.5

8.3 *Comments on the Results*

The proposed approach is a first simple attempt to deal with fractures of deformable objects. Although the results presented above are promising and show the relevance of employing a physically rigorous method to detect fracture, several issues shall be discussed.

Stiffness of the materials. Experiments are shown on two different simple objects with particular material properties (high stiffness). With more elastic materials, the mesh's estimated deformations and internal forces would be rougher, resulting in a more variable separation tensor and so potentially spurious detections of fractures.

Tuning material parameters. A cumbersome process can lie in determining the object's coherent material properties, which consist in the Young's modulus, the Poisson's ratio, and the fracture threshold. There are set empirically in this chapter, and having a fair physical realism may require some effort.

Tracking stretching deformations. As observed, our method faces some problems when tracking opening tearing fractures due to the expansion of the area covered by the object since only the vertices lying on the contour of the mesh are attracted. The method proposed in [44] suggests to weight the visible surface's vertices of the mesh, given their distance to the occluding contour of the mesh. However, this technique tends to alter the material's physical homogeneity, making fractures to likely occur around the contour of the mesh.

Sensitivity to occlusions. Since in the segmentation no distinction is made between a fracture crack and unobserved areas of the object due to occlusions or segmentation errors, our method remains sensitive to these latter events.

Remeshing. Handling the propagation of a fracture by removing the elements attached to a fracturable node and intersected by the fracture plane leads to progressive degeneration of the mesh if several fracture events occur.

9 Multiple Objects

The frame-by-frame registration problem, described in Sect. 2.6 can be extended to multiple objects that can interact and collide.

As illustrated in Fig. 19, the solution consists in integrating the suggested collision detection and response models (Sects. 1.5 and 1.5.2) into the rigid and non-rigid registration framework based on successive RGB-D data.

Thus, point cloud segmentation, matching, rigid and non-rigid registration techniques are adapted as described below.

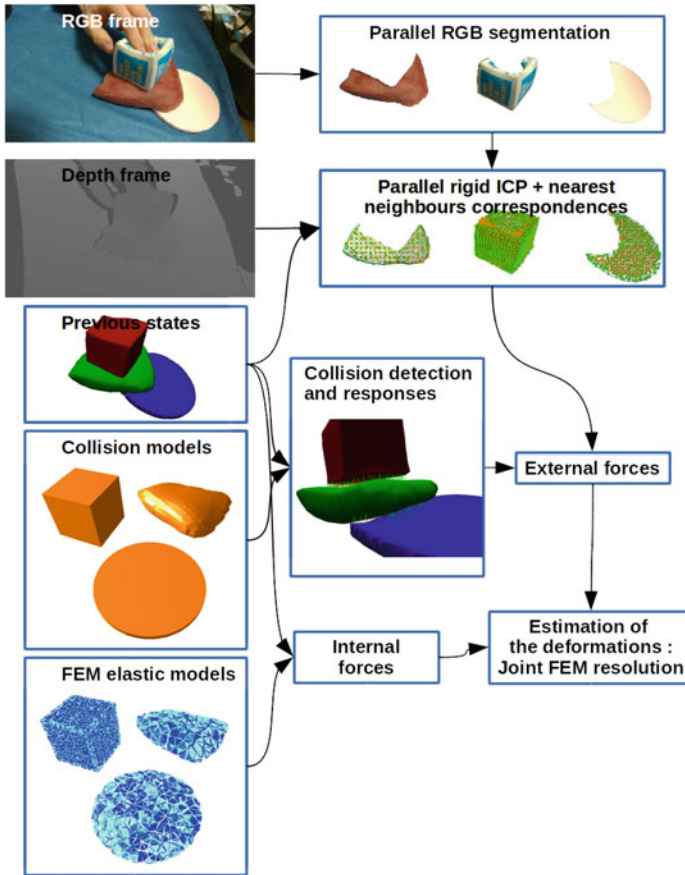


Fig. 19 Overview of our approach for deformable object tracking for multiple interacting objects

9.1 Preliminary Parallel Visual Segmentation

The visual segmentation step presented section 3.1, based on a graph cut minimisation over foreground/background colour models, is carried out in a parallel manner on each considered object $\mathcal{O}_i$. This step provides us with a 3D target point cloud Y_i for each $\mathcal{O}_i$. We limit the size of each Y_i by sampling it on a regular grid in the image plane.

9.2 Parallel Rigid Pose Estimation

Each mesh $\mathcal{X}_i$, initially in its state computed for the previous frame, is then rigidly registered to each observed segmented point cloud $\mathcal{Y}_i$. In contrast to the approach proposed for a single object, for which the rigidly transformed meshes are used to initialise the non-rigid phase, we derive elastic forces from the rigidly transformed meshes. The idea is to integrate these rigid transformations in a smooth and simultaneous manner into the FEM simulation framework and thus to properly cope with collisions.

We actually do not directly consider $\mathcal{X}_i$ but its vertices which appears visible from the previous estimated state in the camera frame. The set $\mathcal{X}_{V,i}$, associated to the object $\mathcal{O}_i$, is thus determined through a visibility test on the rendered mesh with respect to its previous state. This visibility test is performed on the whole rendered scene so that occlusions between the interacting object are considered.

As introduced in Sect. 5, a classical rigid ICP algorithm [40] is then carried out for each object, between $\mathcal{Y}_i$ and $\mathcal{X}_{V,i}$. For each object, we can compute the rigidly transformed mesh $\mathcal{X}'_i = \{\mathbf{x}'_{i,j} : j = 1, \dots, N_{X_{i,t}}\}$, with $N_{X_{i,t}} > 0$ the dimension of $\mathcal{X}'_i$, as follows

$$\mathbf{x}'_{i,j} = \mathbf{T}_i \mathbf{x}_{i,j}, \quad (11)$$

with $\mathbf{T}_i \in SE(3)$ the rigid transformation estimated with the ICP.

We then derive elastic forces $\mathbf{f}_{ext,i}^r$ between both sets $\mathcal{X}'_i$ and $\mathcal{X}_i$ as

$$\mathbf{f}_{ext,i}^r(\mathbf{x}_{i,j}) = k_{i,ext}^r(\mathbf{x}'_{i,j} - \mathbf{x}_{i,j}), \quad (12)$$

with $k_{ext,i}^r > 0$ a stiffness parameter.

9.3 Parallel Point Cloud Matching for Non-rigid Registration

Following the single object approaches, we derive the external forces exerted by the observed segmented point clouds Y_i on the different corresponding sets $\mathcal{X}_{V,i}$. We use sets of external forces $\mathbf{f}_{ext,i}^d \in \mathbb{R}^3$ related to geometrical information as introduced

in Sect. 6.2. For each object $\mathcal{O}_i$, based on the two sets of mesh-to-point cloud and point cloud-to-mesh correspondences, we can compute an external elastic force $\mathbf{f}_{ext,i}^d$ exerted on each $\mathbf{x}_{i,j}$ in $\mathcal{X}_{V,i}$ as follows

$$\mathbf{f}_{ext,i}^d(\mathbf{x}_{i,j}) = k_{ext,i}^d(\mathbf{x}_{i,j} - \mathbf{y}_{i,j}^f), \quad (13)$$

where, as described in [44], $\mathbf{y}_{i,j}^f \in \mathbb{R}^3$ is a linear combination of points in the point clouds which are matched to $\mathbf{x}_{i,j}$, either from mesh-to-point cloud and from point cloud-to-mesh correspondence sets, while $k_{ext,i}^d > 0$ is the stiffness of these external elastic forces, which is here common to each object.

9.4 Resolution

The objective of the resolution process is to balance the sets of external forces capturing rigid and non-rigid transformations of the objects and the interactions between themselves, with the internal forces based on the different deformation FEM models presented above. The estimation consists in solving for each object the dynamic system of linear ODEs involving the internal and the different external forces, based on Lagrangian dynamics

$$\begin{aligned} \mathbf{M}_i \ddot{\mathbf{x}}_i + \mathbf{C}_i \dot{\mathbf{x}}_i + \mathbf{f}_i &= \mathbf{f}_{ext,i}^d + \mathbf{f}_{ext,i}^r + \mathbf{f}_i^c, \\ \text{with } \mathbf{f}_i &= \mathbf{K}_i \mathbf{x}_i + \mathbf{f}_{0,i}, \end{aligned} \quad (14)$$

where $\mathbf{x}_i \in \mathbb{R}^{3N_{X,i}}$ is the vector containing the positions of the vertices of the object $\mathcal{O}_i$, $\mathbf{M}_i \in \mathbb{R}^{3N_{X,i} \times 3N_{X,i}}$ and $\mathbf{C}_i \in \mathbb{R}^{3N_{X,i} \times 3N_{X,i}}$ are the mass and damping matrices, respectively, $\mathbf{K}_i \in \mathbb{R}^{3N_{X,i} \times 3N_{X,i}}$ is the global stiffness matrix which sums the element-wise *rotated* stiffness matrices $\mathbf{K}_e^r = \mathbf{R}_e \mathbf{K}_e \mathbf{R}_e^{-1} \in \mathbb{R}^{3N_{X,i} \times 3N_{X,i}}$, written with respect to whole set of vertices, and $\mathbf{f}_{0,i}$ the corresponding global offset summing the element-wise ones $\mathbf{R}_e \mathbf{K}_e \mathbf{x}_{e,0}$. The terms $\mathbf{f}_{ext,i}^d \in \mathbb{R}^{3N_{X,i}}$, $\mathbf{f}_{ext,i}^r \in \mathbb{R}^{3N_{X,i}}$ and $\mathbf{f}_i^c \in \mathbb{R}^{3N_{X,i}}$ are vectors containing the external forces, the ones resulting from the interaction model, and the collision responses between object $\mathcal{O}_i$ and the other objects, respectively. The Euler implicit integration scheme is used to solve the system with respect to each $\mathbf{x}_i$, along with a conjugate gradient method.

9.5 Experimental Results

Based on the multiple interacting objects tracking algorithm (see Sect. 9), the results presented here report different scenarios involving objects of different shapes, potentially textureless, and with different interactions.

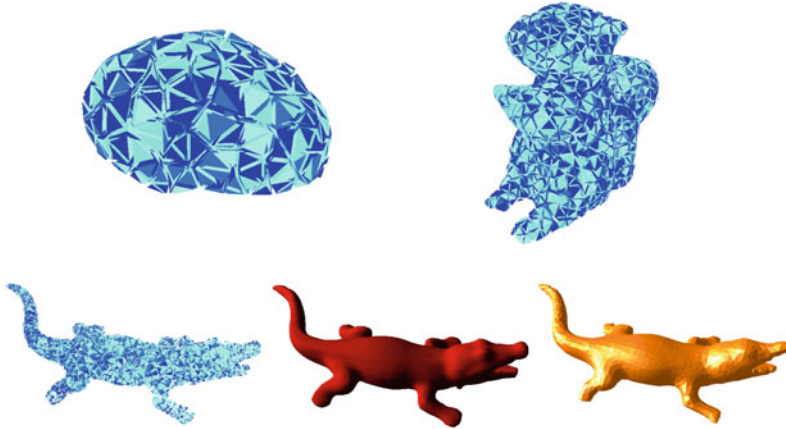


Fig. 20 Tetrahedral meshes for the brain, squirrel and crocodile, plus the visual and collision models for the crocodile toy

The first sequence shows a stuffed toy and an elastic rubber toy brain colliding. The second is a silicon elastic disk wrapping a silicon crocodile puppet filled with plastic balls. In the third sequence, the disk, a silicon object, and a foam cube are stacked and squeezed together.

The point clouds of the investigated scene are acquired using different structured light commodity RGB-D cameras. An Asus Xtion, 320×240 RGB and depth images, is employed for the first sequence. An Intel RealSense SR300 (960×540 images) is employed for the second and third test sequences.

In order to create the geometric and deformation models of some of the considered objects (the squirrel, the crocodile, the toy brain, the light red silicon object), surface meshes of the undeformed objects were first reconstructed offline. We have used the RGB-D based dense 3D reconstruction techniques KinectFusion [45] and InfiniTAM [46] which is implemented for the RealSense SR300. The large resolution and short range of the RealSense SR300, with respect to the Asus Xtion, enables capturing and modeling finer grain geometries, such as the one of the crocodile. Finally, some re-meshing and smoothing tasks have been carried out with a modelling engine to get fair, closed, and clean surface meshes of the objects. The corresponding volumetric tetrahedral meshes have been generated by achieving 3D Delaunay triangulations within the surface meshes, thanks to the CGAL library. As an approximation, we assume the isotropy of the materials in order to apply the deformation model described before. Some of the considered tetrahedral meshes are featured in Fig. 20, as well as the visual and collision surface models for the crocodile. For computational efficiency, the collision mesh is made coarser than the visual mesh.

The software implementation relies on the SOFA simulator [41], which provides the FEM elastic and the collision models. Qualitative results are shown in Figs. 21 and 22 for the different scenarios, as in the provided video. Note that initialization

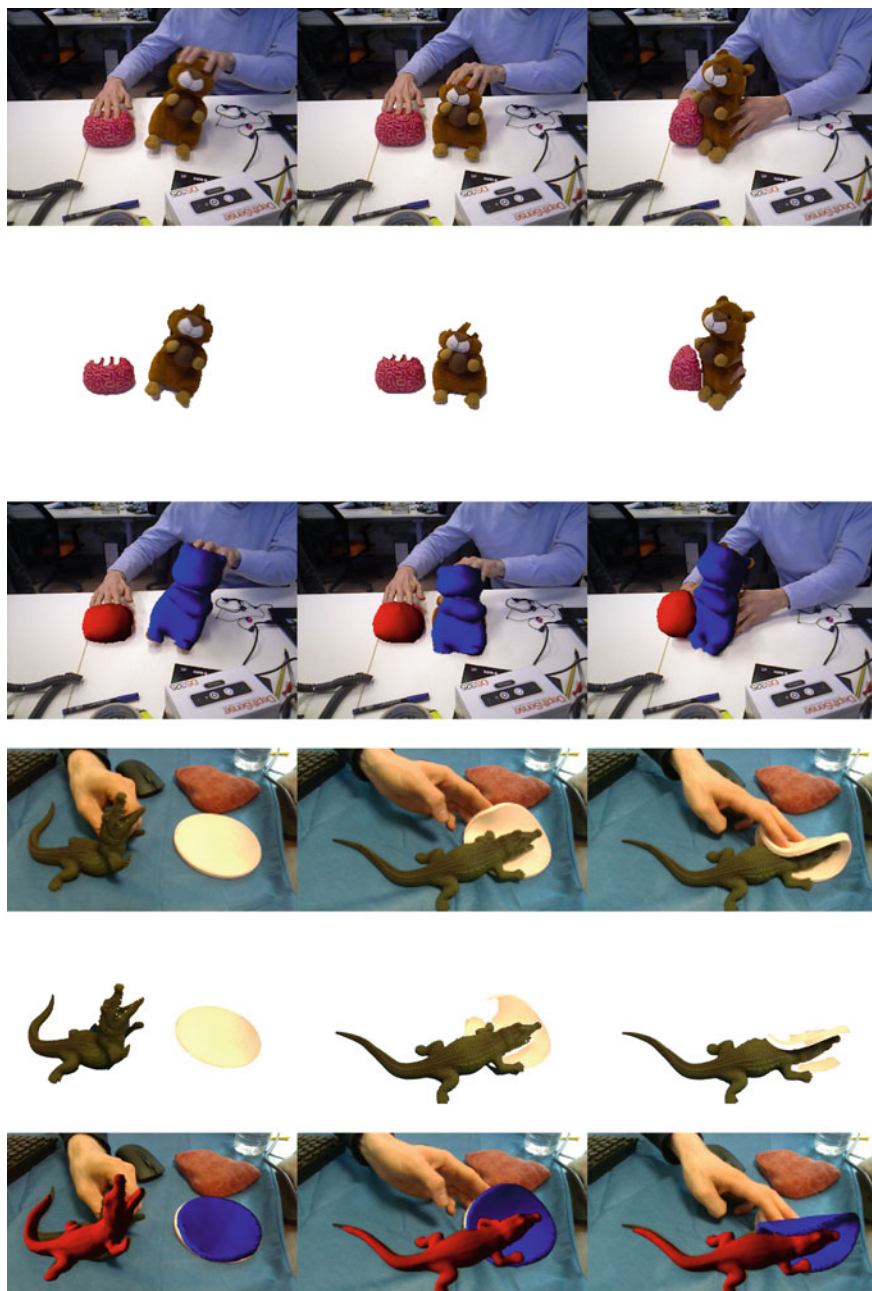


Fig. 21 Results of the proposed registration method for two interacting objects. For each block, the sequence of rows is: input images, the segmented images, and the registered mesh re-projected in the input image

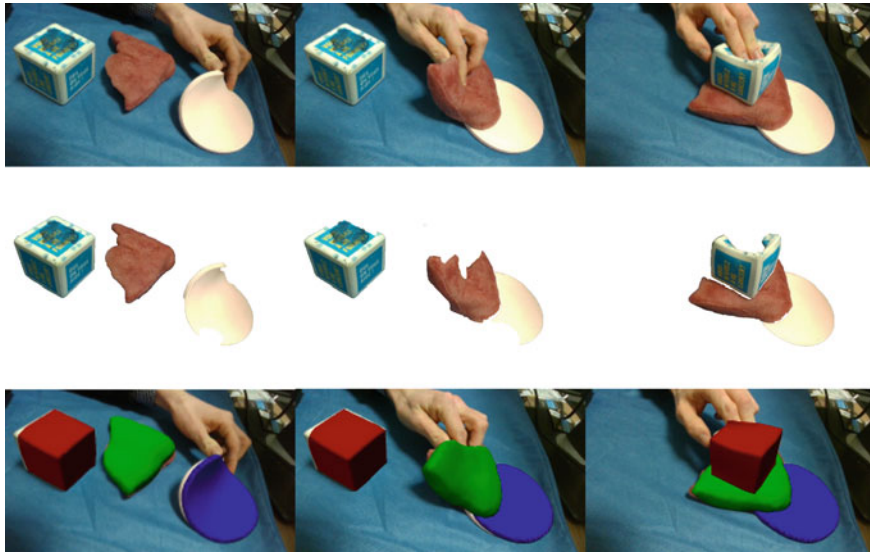


Fig. 22 Results of the proposed registration method for three interacting objects. For each block, the sequence of rows is: input images, the segmented images, and the registered mesh re-projected in the input image

is performed manually, rigidly, and close to the actual initial poses. We assume that the objects are not in contact initially.

The case of the silicon disk wrapping the crocodile toy is examined. The collision between both objects enables to constraint sufficiently the silicon disk so that it gets registered coherently, even with severe bending deformations, occlusions by the crocodile, and errors in the segmentation.

In Fig. 23, we can visualize the vertices of the visible surface of the meshes (green squares), which are processed for registration with the segmented point clouds (brown squares). Different samplings for the point clouds are used according to the resolutions of the meshes. The detected collisions between the interacting objects can be seen on the third row with segments between the found contact points.

The focus on the collision regions in Fig. 24 with wireframe views illustrates that contacts and occlusions between the objects can be properly handled when objects are squeezed, wrapped, or stacked together. Independently tracking them leads to inter-penetrations. In Fig. 25 are shown some failure situations when forces resulting from the rigid ICP registration are not integrated with the resolution.

As for computational issues, segmentation being addressed in a parallel manner, and collision requiring extra computations when objects are getting into proximity, the method does not achieve fast performance: 5–10 frame per second are reached, depending on the number of objects and on the resolutions of the collision surface meshes.

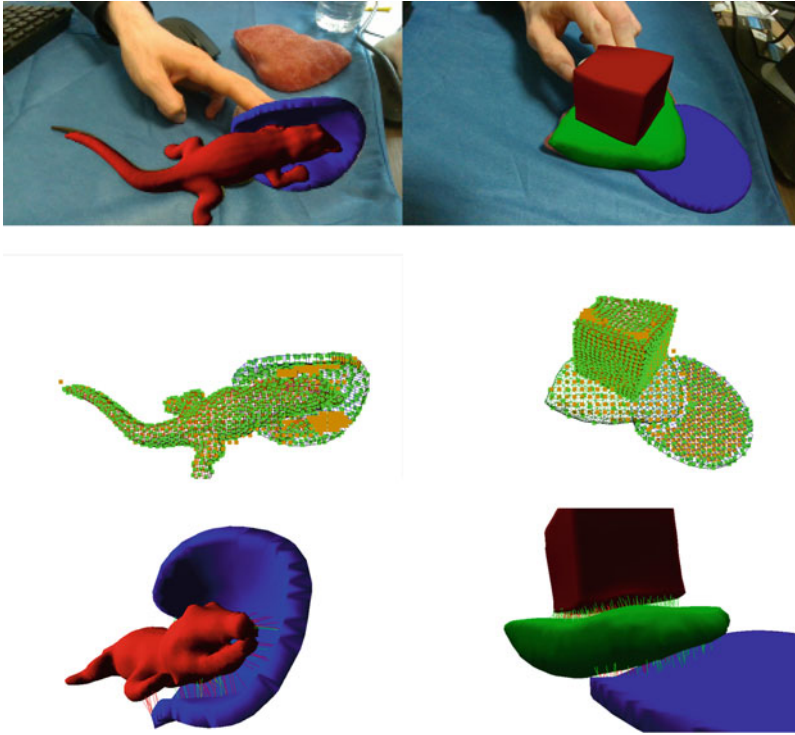


Fig. 23 Visualization of the registered point clouds (brown squares) for the meshes (greens squares as vertices on the visible surfaces) of examples with two and three interacting objects. The displayed segments on the third row show the detected collisions

9.6 Comments on the Results

The experiments presented above show some promising results for the method. However, a few limitations of the method shall be discussed. The system requires some tuning for both the FEM models of the objects and the collision detection and response models to obtain stability and proper behaviour. Some instabilities or ripples can indeed appear when the involved object undergoes fast rigid motions or when too many interactions are involved i.e., when two many objects of complex shapes and collision models are involved). In these cases, the addition of numerous and potentially significant and opposite external forces coming from the point cloud correspondences, the rigid motion estimation, and the collision responses can make the Lagrangian dynamics equations hardly to be solved using the classical implemented conjugate gradient solver. Hence, some tuning can be required to achieve a successful resolution of the whole system. Besides, contact detection distances need to be set relatively high in some cases (see Fig. 23, third row).

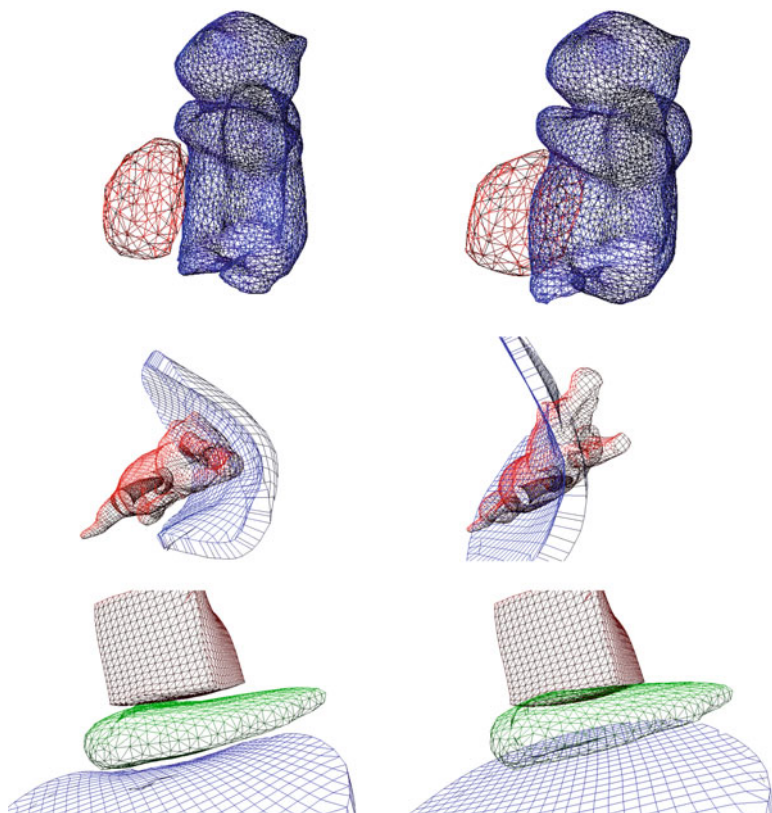


Fig. 24 For the different examples, comparison between registration based on a collision detection model (left column), and without such a model, resulting in large inter-penetrations (right column)



Fig. 25 For the different examples, failure cases when the proposed rigid registration approach is not included in the resolution

The collision detection and response models are also very classical, and a more elegant approach could be undertaken guaranteeing a more physical response by resorting to Signorini's model and by solving the contacts as a linear complementary constraint problem [47]. Including friction through Coulomb's law could also provide some improvements.

Although the system has proven to be fast when considering a single object, multiple object computations linearly increase with the number of objects and even more when contacts occur. One crucial burden in the computations lies in the segmentation process, which is handled independently for each considered object. A simultaneous segmentation method, in a pyramidal manner, could be designed to obtain better efficiency.

10 Application to Robotic Manipulation

In the scope of the RoDyMan project, we present in this section some experiments integrating the tracking method described and validated above into a robotic manipulation task involving a deformable object, that is, the silicon pizza dough. Based on the proposed registration method, the scenario consists of a kinematic control in the operational space to follow a trajectory aiming at manipulating the pizza from one hand to the other on the RoDyMan robotic platform.

10.1 Trajectory Planning and Control

For each hand, the idea is to follow a path that can be decomposed into three paths. The first one is a circular path with constant orientation to make the hand supporting the pizza dough approaching the other one. The second path is also circular, while this time performing a rotation of 180° around the x -axis of $\mathcal{W}$ and a rotation of 50° around the z -axis of $\mathcal{W}$ until the pizza gets overturned. The third is again a circular path that wants to move back the hand to its initial position. In a feed-forward manner, the final position of the second path is computed based on the data provided by the registration algorithm initially, which enables to retrieve the position of the vertex in the mesh corresponding to the object's CoM in its rest shape (see the point C_P in Fig. 26). In detail, it is given by the hand's CoM, C_R , on which the pizza shall be laid, plus a translation along the z -axis of $\mathcal{W}$, and an offset, as seen in Fig. 26. This offset is approximated as the length of the path between C_R and C_P and passing through P_C . The CLIK-based controller controls the joints of the robot's torso and head so that the optical axis of the RGB-D sensor intersects throughout the manipulation task.

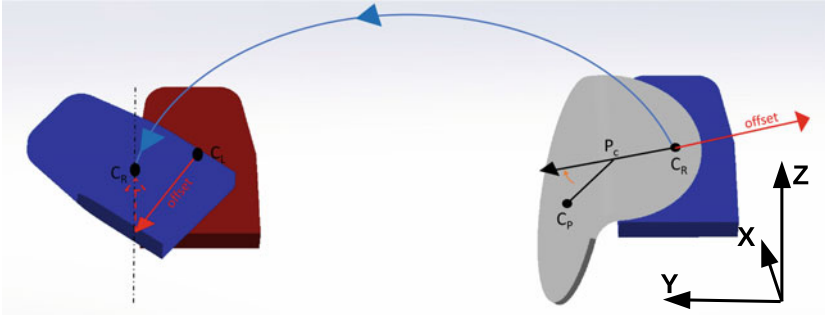


Fig. 26 Trajectory planning for the second circular path

10.2 Experimental Set-Up

The images and the point clouds of the investigated scene are provided by the Asus Xtion RGB-D sensor mounted on the head (320 *times* 240 RGB and depth images). The computer hardware and software have the same characteristics as given in Sect. 7.3.

10.3 Comments on the Results

Two passes, way and back, are performed to bring the silicon pizza dough from the right hand to the left and conversely. The resulting trajectories in the $y - z$ -plane of the CoM of both hands can be observed in Fig. 28a. Starting from the label dot 1, three phases are addressed. At first, a circular path brings the right hand close to the left (label dot 2). Then, the corrected circular path flips the pizza (label dot 3). Finally, a third circular path brings the hand back to its initial position. The reverse trajectory is then achieved by the left hand.

For the registration phase of the pizza dough during the manipulation phase, the results are presented in Fig. 27. The input RGB images are shown in the first row. The corresponding segmented frames are placed in the second row. The 3D mesh tracking the object is finally depicted in the third row. The trajectory of C_P is shown in Fig. 28b. Note that the position of C_P is smoothed using a constant velocity Kalman filter.

When the pizza gets flipped, the occlusions caused by the hands prevail too much: the deformation registration process is thus stopped to avoid spurious deformations on the mesh that may not be recovered afterwards when the pizza gets reasonably visible again. Hence, only the rigid ICP process is maintained during these phases, from step 3 to step 4 and step 7 to 8.

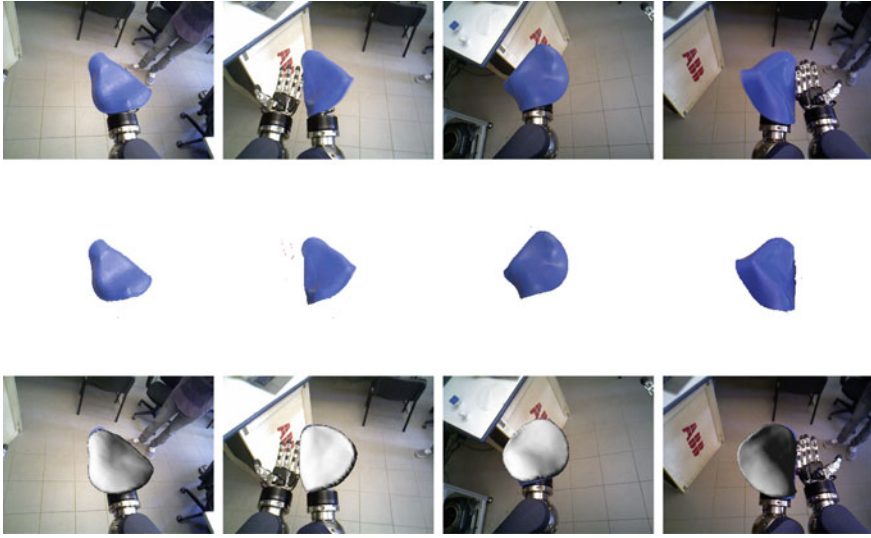


Fig. 27 Results of the tracking process for the silicon pizza, with the input images (first and fourth rows), the segmented frames (second row), and the registered mesh re-projected in the input image with the proposed method

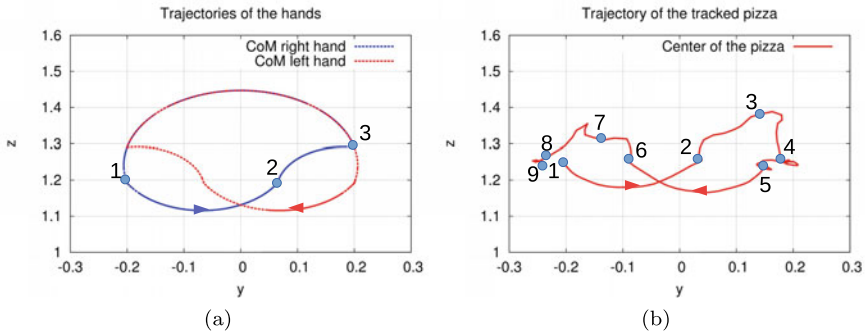


Fig. 28 Trajectories in $y - z$ -plane of both hands' COMs (a), and trajectory of the vertices corresponding to the object's CoM at rest (b)

11 Application to Elasticity Parameter Estimation and Contact Force Estimation

The measure of the contact forces can be a crucial requirement in various applications such as capturing and synthesising human manipulation tasks or controlling robotic hands. Classical techniques use mechatronics sensors, such as force transducers mounted on the manipulator, the manipulated object, or human hands through gloves. This may represent a severe problem in applications like minimally invasive

surgical robots, where sensors cannot be easily installed on surgical instruments due to sterilisation or electrification.

Capturing interactions in manipulation based on computer vision has aroused much interest recently and may represent a convenient, minimally invasive, and cheap sensing set-up. Some efforts have been focused on sensing interactions between rigid objects, but the field remains open when considering deformations.

The purpose of this section is to propose a methodology enabling the measurements of contact forces between the operator and a deformable object through an RGB-D camera. The task is very challenging since a multiplicity of interactions can generate a given deformation.

Here we demonstrate that, by comparing a physical deformation model based on continuum mechanics and known material properties of the object with deformations measured through vision, it is possible to retrieve a single point-wise contact force exerted by an operator (a human hand, a robot end-effector, and so on) on the considered object.

Material properties, represented by elasticity parameters, are estimated in a preliminary step using a force sensor and an optimisation technique. Then, assuming that the object lies on a flat surface and that the tool/object interaction consists of a known single contact point, a similar optimisation technique is used to infer the value of the 3D force exerted on that point by fitting the simulated deformations with those estimated by the vision system. This approach is based on the techniques suggested in [3].

11.1 Related Work

Force sensing for robotics or virtual reality applications is typically based on transducers [48]. These devices can be placed either on the object to be manipulated or on the operator, embedded on skins or gloves [49–51]. Otherwise, force is estimated through transduction mechanisms mounted on the robot joints.

While most of these technologies rely on pressure sensing, other recent approaches use visual information as a cue to measure force. For instance, the changes in the appearance of the fingertip are measured through photodetectors or an external camera, and they are processed to estimate contact forces using statistical models, as in [52–57]. These technologies are limited to measure the normal force and cannot simultaneously consider shear or slip. A promising approach [58], also based on visual observations, relies on the GelSight sensor [59]. In this case, the deformations, measured by marker-based visual tracking techniques on an embedded elastomer medium, are interpreted as known responses to the external load exerted on the sensor.

All the above force-sensing methods require precise calibration and providing the interacting tool/hand and/or the manipulated object with cumbersome and expensive equipment, limiting the range of motion. Often they are tailored to particular objects,

making difficult the generalisation, whereas we wish to design a generic data-driven system.

Force sensing based on external sensing devices represents an appealing alternative. Vision sensors appear as the most simple, cheap, convenient technology to propose. In the literature, several works have suggested using an external vision system to capture the interactions and sense contact forces in the case of object manipulation by a human or any manipulation tool. In the motion capture field, this approach has been demonstrated by using marker-based or markerless vision tracking to capture and synthesise hand/object interaction [32, 60, 61], providing a kinematic analysis of the interactions through discriminative or generative processes and introducing some physics-based constraints to deal with occlusions or collision detection. Some other approaches [62, 63] propose to bridge the gap between the kinematics provided by the motion capture systems and the contact forces by linking physical constraints to visual observations.

The techniques proposed in [34, 64] go further by employing rigid body and contact dynamics to fully model interactions and to link these models with kinematics provided by external vision systems. In this way, the contact forces between a human and the ground [64] or between a hand and a manipulated object [34] can be estimated in a physically realistic manner. To the best of our knowledge, these works are the only cases proposed in the literature where external visual tracking is used as a cue for contact force estimation in rigid and articulated body interactions.

What we propose in this chapter is to use an external vision system to infer the contact forces deriving from the manipulation of deformable objects. This is achieved by confronting the object's physical properties and its deformation model with the output of the vision system. Our approach is close to the idea in [58] of relating measured deformations to forces. However, the method presented in [58], however, relies on an embedded sensing device and addresses force estimation through empirical relationships between deformations and forces. In our approach, as a significant contribution, we propose to use realistic physical modelling of a deformable object to infer interaction forces exerted on it by processing external vision data. We focus here on a static case, for which the general idea is to estimate a point-wise contact force for which the resulting deformations best fit the deformations measured by the vision system. We assume the contact point to be known and the deformable object to be isotropic. Relying on a physical model implies knowing the mechanical properties of the object. Here, we employ the FEM to model the object and its elasticity described by the Young's modulus and the Poisson's ratio. As a second contribution of this work, we suggest estimating these two parameters. This is handled during a preliminary step by exploiting the vision data and a force sensor, which is also used to validate our vision-based force estimation framework.

11.2 Elasticity Parameter Estimation

Our system consists of estimating the elasticity parameters of the object using the point cloud data (provided by an RGB-D sensor and observing the deformations) and a force sensor. For the estimation of these parameters, we follow the data-driven approaches described in [65, 66]. These approaches minimise a fitting error between the simulated deformations, based on the designed deformation and interaction models presented above and generated by the input operator force acquired from the sensor, and the deformation captured by the RGB-D sensor. These two methods also employ finite elements for the deformation model. The work [66] goes further by proposing a framework that sequentially tracks the shape and estimates both material and dynamic parameters (i.e., the damping) through the dynamic deformation model and solely based on a complete vision capture set-up with various RGB-D sensors around the scene. Here, we reason statically for modeling, and we limit this preliminary process as an estimation process of the Young's modulus and the Poisson's ratio of the material, employing a single RGB-D sensor and the force sensor on a robotic operator, with a set-up similar to the one proposed in [65]. The deformations are indeed generated by applying an effort (a compression, in our case) on the deformable object, and we observe the deformations with the vision sensor once static equilibrium is reached. These deformations can be simulated, starting from the same initial rest shape, provided the deformation and interaction model presented above, the elasticity parameters, and the input measured contact force. As a data-driven approach, our problem is thus addressed by minimising the deviation between these simulated deformations and the observed ones with respect to the elasticity parameters. This deviation is defined by a fitting function accounting for the sum of squared distances between the measured real displacements, observed on the acquired object point cloud, and the simulated deformations. Such a function is defined as

$$e(E, \nu) = \text{dist}(\text{sim}(E, \nu, \mathbf{f}^{op}, \mathbf{x}_{i_c}), Y), \quad (15)$$

where $\mathbf{f}^{op} = [f_x^{op} \ f_y^{op} \ f_z^{op}]^T \in \mathbb{R}^3$ is the measured force exerted by the operator on the object, $\mathbf{x}_{i_c} \in \mathbb{R}^3$ is the i -th contact point, Y is the acquired point cloud, $\text{dist}(\cdot)$ is the distance function, and $\text{sim}(\cdot)$ is the function simulating deformations. The point cloud Y is segmented on the considered object by running the segmentation phase described in Sect. 3 to obtain $\mathcal{Y}$.

For a relevant significant error function, we design it by employing the matching technique presented in Sect. 6 between the segmented point cloud and the visible part of mesh of the virtual object and conversely. Based on the two sets of mesh-to-point cloud and point cloud-to-mesh correspondences, the error function is calculated as:

$$e(E, \nu) = \frac{1}{N_{\mathcal{X}_V}} \sum_{i=0}^{n_{\mathcal{X}_V}} (\mathbf{x}_i - \mathbf{NN}_{\mathcal{Y}}(\mathbf{x}_i))^2 + \quad (16)$$

$$\frac{1}{n_Y} \sum_{j=0}^{N_Y} (\mathbf{y}_j - \mathbf{NN}_{\mathcal{X}_V}(\mathbf{y}_j))^2. \quad (17)$$

The optimization problem for (E, ν) is nonlinear, and the evaluation of the objective function is expensive. At the same time, its gradients are non-trivial to compute, making gradient-based optimization methods prohibitive. We thus employ the gradient-free Nelder-Mead method, which is an extension of the downhill simplex method to the nonlinear case. We point out that, during the Nelder-Mead processs, for each evaluation of the objective function, the mesh is initially reset to its rest shape before applying the virtual contact force $\mathbf{f}^{op}$ on the known vertex of the mesh, given the elasticity parameters (E, ν) . For each evaluated value of (E, ν) , a simulation is then started until a static equilibrium is reached. From this static equilibrium, the matching process to derive equation (16) is handled.

11.3 Contact Force Estimation

By inverting the method for parameter estimation, the problem of recovering the contact force exerted on the manipulated object relies on the interaction model between the object, the deforming operator, and the ground, presented in Sect. 1.5. Similarly, we then employ a fitting process, this time with respect to the force, between simulated deformations given the interaction model and the deformations observed using the vision. Since the material parameters are now determined, we can use the deformation model and the registration technique described before as the visual observation, providing a regularized and complete observation of the deformations.

Here, we consider the static case to estimate the interaction force, for which the static equilibrium state of the deformed material due to the exerted contact force is reached. Our system consists in determining the force for which the resulting simulated deformations best fit the mesh deformed by the vision data. Formally, we minimize the least square error e between the deformations sensed through vision, $\mathbf{x}_i^{vision} \in \mathbb{R}^3$, and the deformations simulated based on the interaction model involving, $\mathbf{x}_i^{sim}(\mathbf{f}^{op}) \in \mathbb{R}^3$, with respect to $\mathbf{f}^{op}$

$$e(\mathbf{f}^{op}) = \sum_i (\mathbf{x}_i^{vision} - \mathbf{x}_i^{sim}(\mathbf{f}^{op}))^2 \quad (18)$$

Given the full interaction model presented above, the derivation of the error (18) is non-trivial, and it requires the inversion of the model, making iterative methods (i.e., Newton's one) tricky to handle. Instead, as for the estimation of the material parameters, since we aim at performing a quite global process without any strong

guess on this force, we suggest a Nelder-Mead optimization framework to minimize e with respect to $\mathbf{f}^{op}$. In practice, to compute e for a given force $\mathbf{f}^{op}$ in the Nelder-Mead process, we start from the state of the mesh deformed through the vision registration. We then substitute all the external forces due to the point cloud data by $\mathbf{f}^{op}$ on the known vertex in contact with the operator. A simulation is evolved based on this force and the interaction model presented in Sect. 1.5. After a few iterations in the simulation process, e can be computed and thus measures the ability of $\mathbf{f}^{op}$ to reproduce the actions of the forces provided by vision on the object.

11.4 Experimental Results

The results presented here involve a deformable object, a stuffed toy undergoing a compression deformation effort applied by a tool fixed on the end-effector of a Kuka LWR arm, equipped with a force sensor at the wrist (see Fig. 29). The point clouds of the investigated scenes are acquired using a calibrated RGB-D camera Asus Xtion, with 320×240 RGB and depth image resolution. To estimate the elasticity parameters and the applied contact compression force, we process the data of a single RGB-D camera taken at static equilibrium.

A surface mesh of the undeformed object was reconstructed offline to build the deformation model of the stuffed toy by using an RGB-D based dense 3D recon-

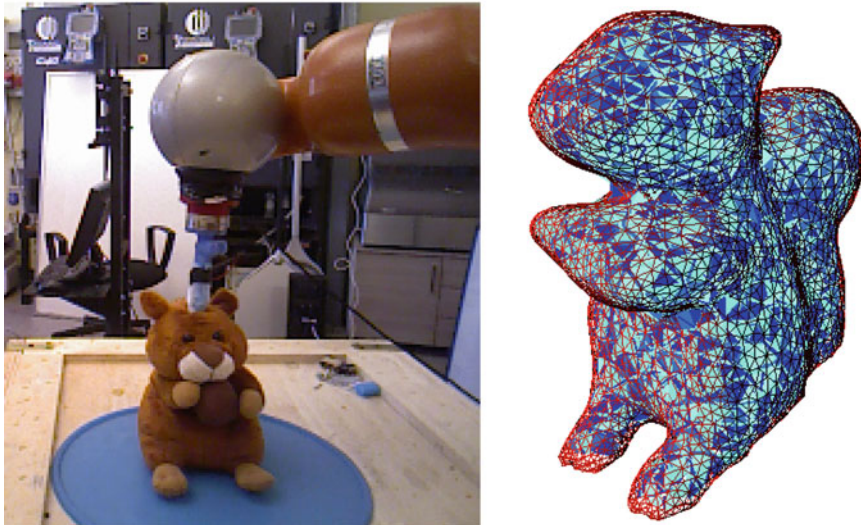


Fig. 29 On the left, the experimental set-up with the tool mounted on the robotic arm, equipped with a force sensor, to compress the object. On the right, the surface triangular (in red) and the volumetric tetrahedral mesh (in blue) of the stuffed toy

struction technique [45] and flying around the object with the Xtion sensor. Then, we manually segment the part of the scene featuring the object. Finally, some re-meshing and smoothing procedures are carried out with a modeling engine to get a fair, closed, and clean surface mesh of the object. The volumetric tetrahedral mesh was generated by carrying out a 3D Delaunay triangulation on the surface mesh with the CGAL library. As a compromise between modeling accuracy and real-time constraints, we generated a volumetric mesh with 951 vertices and 5015 tetrahedral elements (see Fig. 29). As an approximation, we assume the isotropy of the material of the stuffed toy to apply the suitable deformation model. We employed the SOFA simulator for modeling purposes, which enables us to deal with various physical models and evolve simulations in real-time. In terms of hardware, a standard laptop with an NVIDIA GeForce 720M graphic card has been used, along with a 2.4 GHz Intel Core i7 CPU.

11.4.1 Elastic Parameters Estimation

For the estimation of the elastic parameters, we first measure the contact force exerted by the tool mounted on the robotic arm to compress the object, that is $\mathbf{f}_{mech}^{op} = [0.17 \ 1.125 \ 4.006]^T$ N. Due to the particular shape of the considered object, the application of this pointwise contact force in simulation may result in the loss of the static equilibrium. For this reason, we constrain the system by fixing some vertices' position on the lower part of the shape, close to the contact area with the flat surface. In this way, the object may not get bent excessively or turned over, and its base remains quite rigidly attached to the flat contact surface.

Following the basic implementation of the Nelder-Mead algorithm, since the parameter space for (E, ν) is dimension two, three candidate samples are sorted after each iteration of the optimization while performing the reflection, expansion, contraction, and shrinking steps. This arrangement provides the best, a good, and the worst candidate. We also integrate the specific boundaries for both E and ν in the process, in the sense that inequalities $E > 0$ and $0 < \nu < 0.5$ should be preserved during the different steps. If an inequality is violated, E or ν is reset slightly below or above.

We tested our parameters estimation technique with two different initial configurations. In Figs. 31 and 32 we can observe the trajectories of the three sorted candidates for these two configurations, along E and ν , throughout the iterations of the Nelder-Mead process. The figures also show the fitting errors for the three candidates after each iteration. In the first case, the initial values are pretty far from the actual estimated one, stressing out the robustness of the estimation for rough initial guesses. At the same time, in the second configuration, the process starts closer to the solution. For both configurations convergence is achieved respectively towards $(E, \nu) = (4268.65 \text{ Pa}, 0.412031)$ and $(E, \nu) = (4328.12 \text{ Pa}, 0.415625)$. As it can be noticed on the plot of the fitting error in Fig. 30, the non-convexity can be espe-

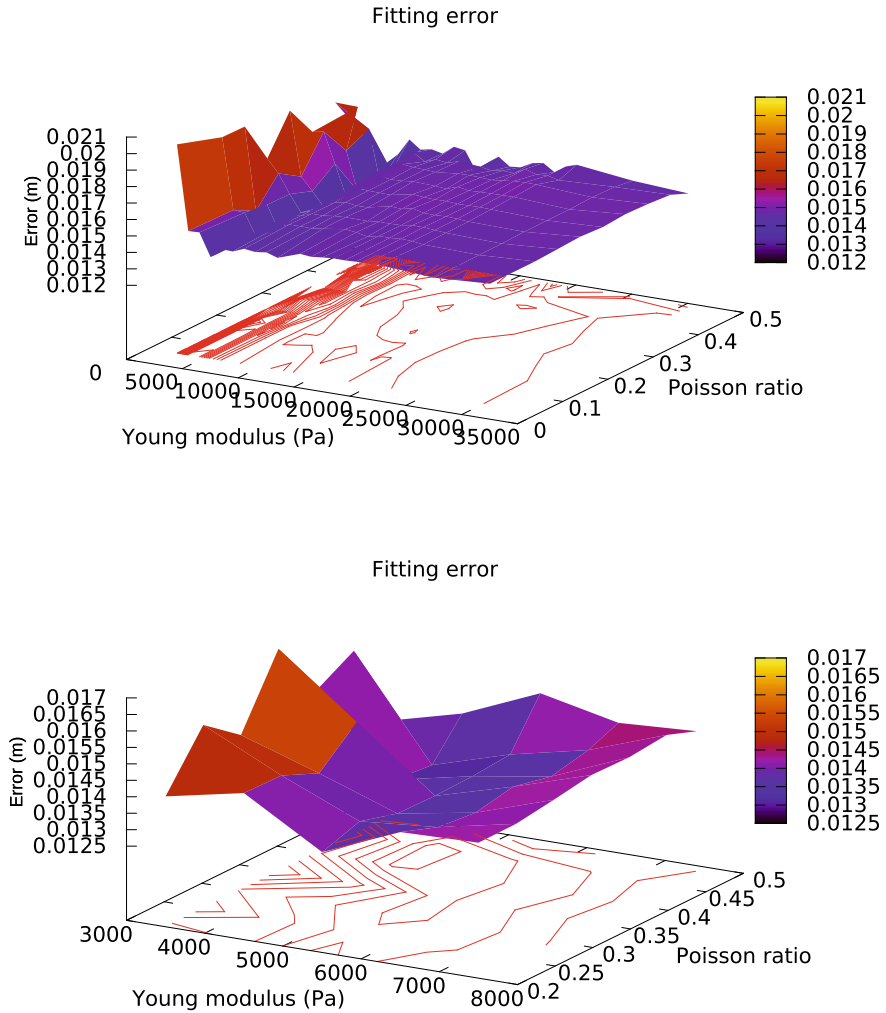


Fig. 30 The top figure shows the fitting error with respect to the elasticity parameters. The bottom one depicts a closer view around the global minimum

cially observable for the ν parameter, along which the error is relatively flat, resulting in some local minima.

11.4.2 Contact Force Estimation

Intending to test the operator contact force estimation based on the vision tracking system, we first proceed by setting the material parameters of the deformation models

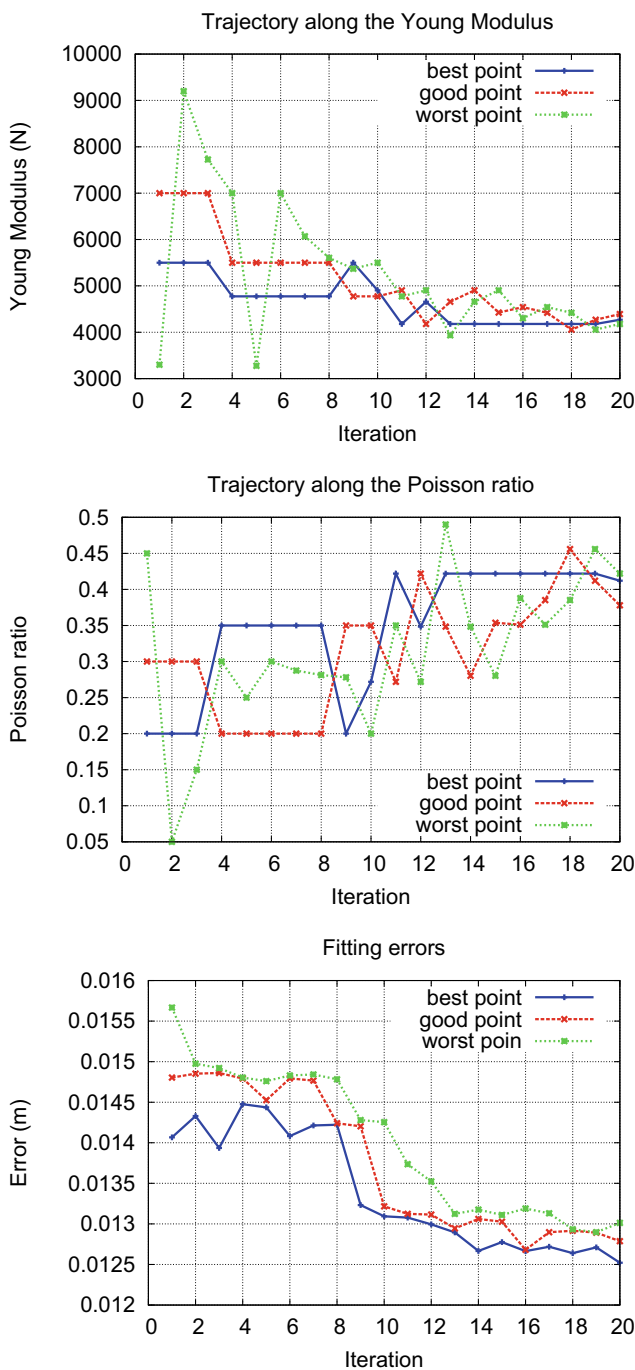


Fig. 31 Nelder-Mead process for elasticity parameters estimation for the first initial configuration

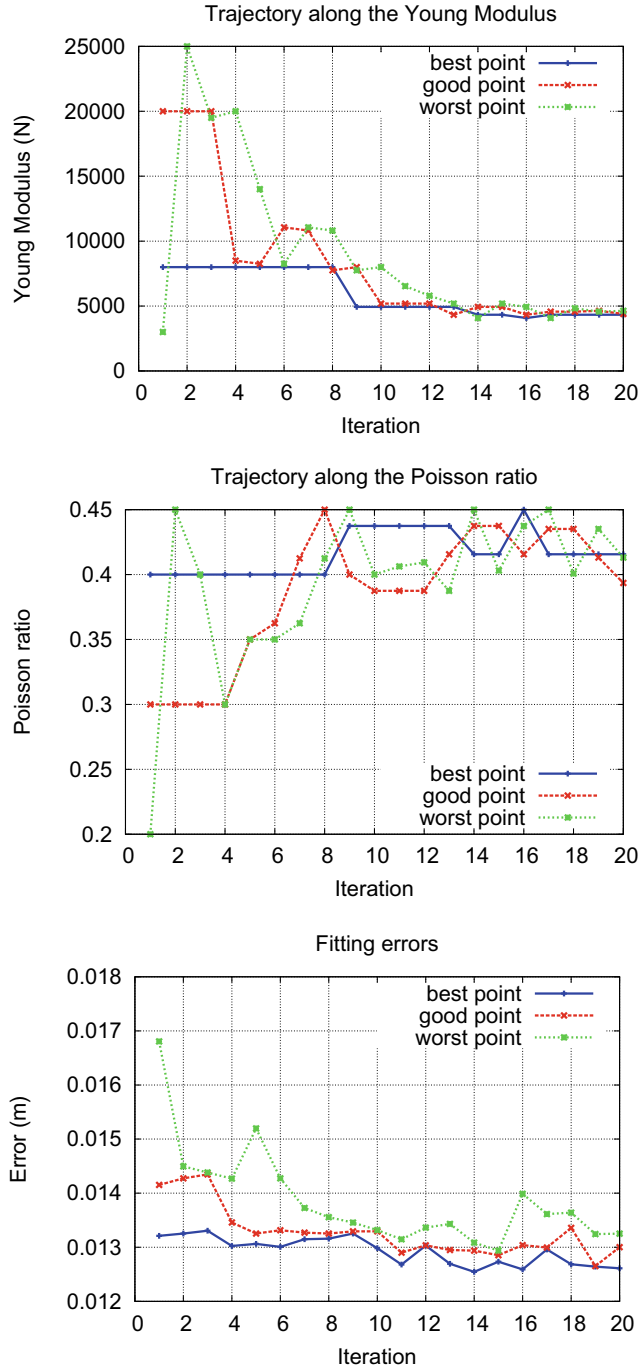


Fig. 32 Nelder-Mead process for elasticity parameters estimation for the second initial configuration

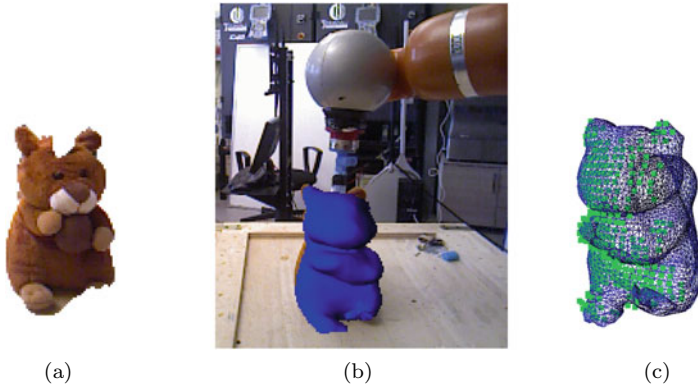


Fig. 33 Registration process, with: **a** Preliminary segmentation, **b** Fitting result between the mesh and the segmented point cloud, **c** Registered mesh reprojected in the image

used in the vision system. Here, we use $d(E, \nu) = (4268.65 \text{ Pa}, 0.412031)$. The result of the registration process can be observed in Fig. 33.

For each evaluation of the error function, the registered mesh is relaxed from the forces exerted by the vision while applying the point-wise contact force to evaluate the known vertex. Fixing vertices as boundary conditions to constraint the simulation is not necessary since we measure here the ability of this force to keep the static equilibrium already reached by the action of the vision forces. Since the parameter space is three-dimensional, four samples are sorted after each iteration, namely the best, the worst, and two intermediate samples. Two different initial configurations are tested here. The former starts quite far, without any particular guess on the intensity and direction of the force. The latter is close to the actual value of the force given by the sensor $\mathbf{f}_{mech}^{op} = [0.17 \ 1.125 \ 4.006]^T \text{ N}$. Figures 34 and 35 show in both cases the trajectories of the four candidates for the estimate of the contact force $\mathbf{f}^{op}$ and the corresponding fitting errors. In both cases, the algorithm converges respectively towards a force of $\mathbf{f}^{op} = [0.618 \ 0.0687929 \ 3.54801]^T \text{ N}$ and $\mathbf{f}^{op} = [-0.462414 \ 0.247626 \ 3.71292]^T \text{ N}$ which are relatively close to the value sensed by the force sensor mounted on the robot, thus validating our whole model. Convergence is of course reached much faster in the second case.

11.5 Comments on the Results

The proposed framework consists of (i) estimating the material parameters based on a known exerted force to develop a deformation model, and (ii) estimating the force based on the known deformation model and on a registration technique that allows measuring deformations. The results presented in this chapter are promising, but several issues shall be discussed.

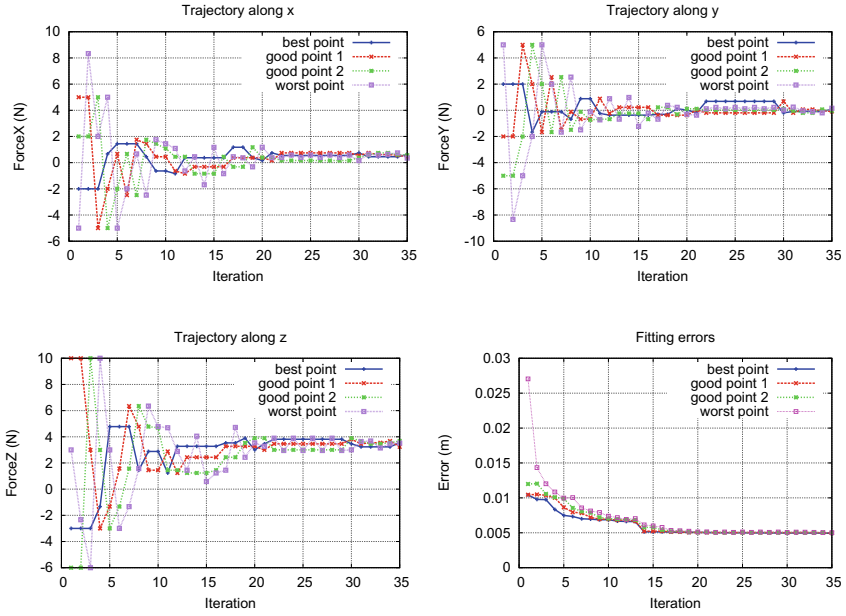


Fig. 34 Nelder-Mead process for force estimation, for the first initial configuration

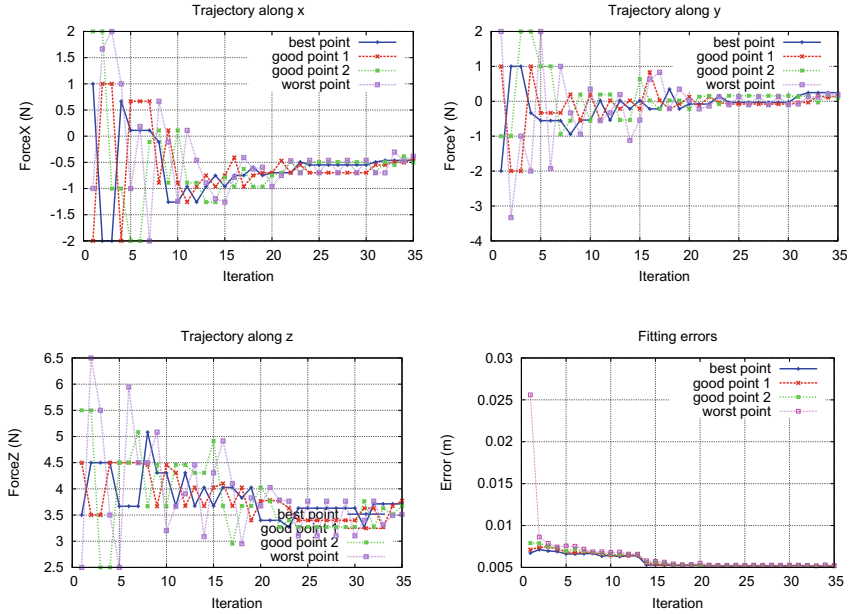


Fig. 35 Nelder-Mead process for force estimation, for the second initial configuration

Our imaging set-up is based on a single Asus Xtion RGB-D sensor, providing relatively low resolution, partial and noisy point cloud data around the object. It results in non-convex shapes for the fitting error function in the elasticity estimation process, especially concerning ν , or in registration errors. A more sophisticated set-up to capture deformations, such as the one proposed in [66], with a set of RGB-D sensors at different viewpoints, would give more accurate results for the estimation of both the mechanical parameters and the contact force.

Besides, we assumed the contact point between the object and the operator to be known, as well as the contact between the object and the underlying flat surface. Further development of our approach would be, following the method described in Sect. 9, the design of a vision system able to capture the interaction between the object and its interacting environment: the manipulation tool, the table, and so on, enabling the detection of contact points, some proper priors for the segmentation and registration of the different entities.

In the estimation of the force exerted by the operator, we use a gradient-free Nelder-Mead optimization method. This has the advantage of being relatively easy to implement and robust to a coarse initialization. It is, however, relatively slow to run. Indeed, for force estimation, each evaluation of the error function requires at least five successive simulations to obtain a reliable error for deformations of the registration process. An iteration in the Nelder-Mead algorithm requires, in this case, seven evaluations, around 350 ms, given that one simulation takes around 10 ms, making the process quite far from being real-time if considering a stream of successive RGB-D data. A possible improvement would be to investigate an efficient inversion of the whole interaction model and some local optimization techniques such as quadratic programming, as proposed in [67].

Finally, our system is designed for the static case, for which deformations have reached a static equilibrium. The presented method might be adapted to a dynamic case by benefiting from a measure, through vision, of the kinematics of the object or interacting entities. Hence, based on Lagrangian dynamics, the system could track online both the deformations and the force.

12 Discussion and Conclusion

Real-time perception is a crucial issue for robotic nonprehensile dynamic manipulation tasks, especially when considering deformable objects such as tissues, organs, clothes, or food, which has to be manipulated in a very challenging way, as in the case of a pizza being stretched and tossed by a humanoid pizza chef robot.

In this chapter, we have described a series of contributions to register and track deformable objects undergoing fast rigid motions, large deformations, and behaviours such as fractures and interactions with other objects or with the environment, based on the deformations models introduced in the previous chapter.

The recent development of physics-based modelling methods for deformable elastic objects for registration purposes and the availability of real-time implementations

led us to choose such an approach to track a textureless and smooth object subjected to various large deformations with an RGB-D sensor. The main idea is to continuously fit the volumetric deformation models onto the successive 3D point clouds provided by the RGB-D camera. At the same time, the FEM modelling acts as a regularizer for the registration.

For the primary case of a single object, the use of the corotational FEM model, an efficient segmentation method, and a classical point cloud registration techniques have made our system a promising real-time tracking method able to handle various deformations and motions while being robust to segmentation errors or occlusions.

Regarding fractures, our approach, as the first attempt to deal with such topological changes in a computer vision and registration problem, consists of integrating into the FEM deformation model a model for fracture events. Fracturable nodes are firstly detected in the considered mesh by decomposing the internal forces into tensile and compressive components. In a second step, we perform a simple local re-meshing of the elements around the fracture nodes. The registration process with the RGB-D data is the same as for a single object to compute the deformations. The investigation of fracture events over the mesh and the eventual re-meshing is purely physics-based. Experimental results on two different objects and three sorts of fractures show the relevance of this simple physical-based method. However, this can be improved by handling more properly the re-meshing phase by dealing with more significant elastic and plastic deformations and handling occlusions by distinguishing them with fracture cracks.

We have also designed a system able to register multiple interacting deformable elastic objects simultaneously. This system also relies on the corotational FEM elastic models to accurately and efficiently model elasticity and an interaction model consisting of collision detection and response models to cope with contacts between objects. Based on RGB-D input data, point cloud matching and registration processes are achieved on segmented data in a parallel manner, enabling to deal with occlusions. Rigid motions, deformations, and contacts of all the considered objects are then estimated in a common solver, providing a realistic behaviour and coherently constraining the registration of the involved entities. The method has been tested on simple scenarios with elastic objects but could be extended to other physical models to articulate objects, such as for robotic manipulation applications.

A first step towards robotic manipulation of such objects has been achieved by performing a simple manipulation task on the RoDyMan platform.

Finally, another application has focused on designing a system able to recover a single contact force exerted on a deformable object by resorting to an external vision and registration system and fitting simulated deformations with the observed ones. This is achieved by taking advantage of the physical FEM modelling and employing a basic interaction model with the manipulator and the environment, proposing a simple manipulation scenario. The deformation model requires two mechanical parameters, Young's modulus and Poisson's ratio, which we initially estimate by fitting simulated deformations with the point cloud given by an RGB-D sensor. The entirely determined elastic model then feeds the registration system and the simulator, which are matched to the contact force. We address these two

optimization problems using gradient-free Nelder-Mead methods. Some promising results have been obtained on a simple case of a single applied compression force on a known contact point at static equilibrium.

References

1. A. Petit, S. Cotin, V. Lippiello, B. Siciliano, Capturing deformations of interacting non-rigid objects using RGB-D data, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 491–497, Madrid, E (2018)
2. A. Petit, V. Lippiello, G.A. Fontanelli, B. Siciliano, Tracking elastic deformable objects with an RGB-D sensor for a pizza chef robot. *Robot. Auton. Syst.* **88**, 187–201 (2017)
3. A. Petit, V. Lippiello, B. Siciliano, Tracking fractures of deformable objects in real-time with an RGB-D sensor, in *2015 International Conference on 3D Vision*, pp. 632–639, Lyon, F (2015)
4. C. Elbrechter, R. Haschke, H. Ritter, Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling, in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1427–1432 (2011)
5. N. Haouchine, J. Dequidt, M.-O. Berger, S. Cotin, Monocular 3D reconstruction and augmentation of elastic surfaces with self-occlusion handling. *IEEE Trans. Vis. Comput. Gr.* **21**(12), 1363–1376 (2015)
6. N. Haouchine, J. Dequidt, I. Peterlik, E. Kerrien, M.-O. Berger, S. Cotin, Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery, in *2013 IEEE International Symposium on Mixed and Augmented Reality*, pp. 199–208 (2013)
7. J. Schulman, A. Lee, J. Ho, P. Abbeel, Tracking deformable objects with point clouds, in *2013 IEEE International Conference on Robotics and Automation*, pp. 1130–1137 (2013)
8. A. Bartoli, V. Gay-Bellile, U. Castellani, J. Peyras, S. Olsen, P. Sayd, Coarse-to-fine low-rank structure-from-motion in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8 (2008)
9. A. Weiss, D. Hirshberg, M.J. Black, Home 3D body scans from noisy image and range data, in *2011 IEEE International Conference on Computer Vision*, pp. 1951–1958 (2011)
10. A. Bartoli, A. Zisserman, Direct estimation of non-rigid registrations, in *British Machine Vision Conference*, pp. 899–908 (2004)
11. M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contour models. *Int. J. Comput. Vis.* **1**(4), 321–331 (1988)
12. J. Pilet, V. Lepetit, P. Fua, Fast non-rigid surface detection, registration and realistic augmentation. *Int. J. Comput. Vis.* **76**(2), 109–122 (2007)
13. D. Terzopoulos, A. Witkin, M. Kass, Constraints on deformable models: recovering 3D shape and nonrigid motion. *Artif. Intell.* **36**(1), 91–123 (1988)
14. M. Salzmann, J. Pilet, S. Ilic, P. Fua, Surface deformation models for nonrigid 3D shape recovery. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(8), 1481–1487 (2007)
15. A. Jordt, R. Koch, Direct model-based tracking of 3D object deformations in depth and color video. *Int. J. Comput. Vis.* **102**, 1–17 (2013)
16. M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, M. Stamminger, Real-time non-rigid reconstruction using an RGB-D camera. *ACM Trans. Gr.* **33**(4), 1–12 (2014)
17. B. Allain, J.-S. Franco, E. Boyer, An efficient volumetric framework for shape tracking, in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 268–276 (2015)
18. R.A. Newcombe, D. Fox, S.M. Seitz, Dynamic fusion: reconstruction and tracking of non-rigid scenes in real-time, in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 343–352 (2015)

19. V. Lippiello, F. Ruggiero, B. Siciliano, Floating visual grasp of unknown objects using an elastic reconstruction surface, in *Robotics Research: The Fourteenth International Symposium, in Springer Tracts in Advanced Robotics 70* ed. by C. Pradalier, R. Siegwart, G. Hirzinger (Springer, 2011), pp. 329–344
20. L.D. Cohen, I. Cohen, Deformable models for 3-D medical images using finite elements and balloons, in *1992 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 592–598 (1992)
21. T. McInerney, D. Terzopoulos, A finite element model for 3D shape reconstruction and nonrigid motion tracking, in *1993 IEEE International Conference on Computer Vision*, pp. 518–523 (1993)
22. A. Malti, R. Hartley, A. Bartoli, J.-H. Kim, Monocular template-based 3D reconstruction of extensible surfaces with local linear elasticity, in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1522–1529 (2013)
23. K. Varanasi, A. Zaharescu, E. Boyer, R. Horaud, Temporal surface tracking using mesh evolution, in *European Conference on Computer Vision*, pp. 30–43 (2008)
24. A. Zaharescu, E. Boyer, R. Horaud, Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(4), 823–837 (2011)
25. C.J. Paulus, N. Haouchine, S.-H. Kong, R.V. Soares, D. Cazier, S. Cotin, Handling topological changes during elastic registration. *Int. J. Comput. Assist. Radiol. Surg.* **12**(3), 461–470 (2017)
26. A. Tsoli, A.A. Argyros, Tracking deformable surfaces that undergo topological changes using an RGB-D camera, in *2016 Fourth International Conference on 3D Vision*, pp. 333–341 (2016)
27. I. Badami, J. Stückler, S. Behnke, Depth-enhanced Hough Forests for object-class detection and continuous pose estimation, in *Workshop on Semantic Perception, Mapping and Exploration* (2013)
28. A. Tejani, D. Tang, R. Kouskouridas, T.-K. Kim, Latent-Class Hough Forests for 3D object detection and pose estimation, in *European Conference on Computer Vision*, pp. 462–477 (2014)
29. H. Hamer, K. Schindler, E. Koller-Meier, L. Van Gool, Tracking a hand manipulating an object, in *2009 IEEE International Conference On Computer Vision*, pp. 1475–1482 (2009)
30. K. Kim, V. Lepetit, W. Woo, Keyframe-based modeling and tracking of multiple 3d objects, in *2010 9th IEEE International Symposium on Mixed and Augmented Reality*, pp. 193–198 (2010)
31. N. Kyriazis, A. Argyros, Scalable 3D tracking of multiple interacting objects, in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3430–3437 (2014)
32. I. Oikonomidis, N. Kyriazis, A.A. Argyros, Full DoF tracking of a hand interacting with an object by modeling occlusions and physical constraints, in *2011 IEEE International Conference on Computer Vision*, pp. 2088–2095 (2011)
33. I. Oikonomidis, N. Kyriazis, A.A. Argyros, Tracking the articulated motion of two strongly interacting hands, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1862–1869 (2012)
34. T.-H. Pham, A. Kheddar, A. Qammar, A.A. Argyros, Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces, in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2810–2819 (2015)
35. M. Salzmann, R. Urtasun, Physically-based motion models for 3D tracking: A convex formulation, in *2011 IEEE International Conference on Computer Vision*, pp. 2064–2071 (2011)
36. S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, C. Theobalt, Real-time joint tracking of a hand manipulating an object from RGB-D input, in *European Conference on Computer Vision*, pp. 294–310 (2016)
37. K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, Y. Liu, Real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. *ACM Trans. Gr.* **36**(4), 32:1–32:13 (2017)
38. C. Rother, V. Kolmogorov, A. Blake, Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Gr.* **23**, 309–314 (2004)

39. Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(11), 1222–1239 (2001)
40. Y. Chen, G. Medioni, Object modelling by registration of multiple range images. *Image Vis. Comput.* **10**(3), 145–155 (1992)
41. F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, S. Cotin, Sofa: A multi-model framework for interactive physical simulation, in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, pp. 283–321 (2012)
42. A.V. Gelder, Approximate simulation of elastic membranes by triangulated spring meshes. *J. Gr. Tools* **3**(2), 21–41 (1998)
43. B.A. Lloyd, G. Székely, M. Harders, Identification of spring parameters for deformable object simulation. *IEEE Trans. Vis. Comput. Gr.* **13**(5), 1081–1094 (2007)
44. A. Petit, V. Lippiello, B. Siciliano, Real-time tracking of 3D elastic objects with an RGB-D sensor, in *2015 IEEE International Conference on Intelligent Robots and Systems*, pp. 3914–3921 (2015)
45. R.A. Newcombe, A.J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, A. Fitzgibbon. Kinect Fusion: real-time dense surface mapping and tracking, in *10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136 (2011)
46. O. Kähler, V.A. Prisacariu, D.W. Murray, Real-time large-scale dense 3d reconstruction with loop closure, in *European Conference on Computer Vision*, pp. 500–516 (2016)
47. J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, P.G. Kry, Volume contact constraints at arbitrary resolution, in *ACM SIGGRAPH 2010 Papers*, pp. 1–10 (2010)
48. B. Siciliano, O. Khatib, *Springer Handbook of Robotics*, 2nd edn. (Springer Science & Business Media, 2016)
49. R.S. Dahiya, G. Metta, M. Valle, G. Sandini, Tactile sensing-from humans to humanoids. *IEEE Trans. Rob.* **26**(1), 1–20 (2010)
50. N. Wettels, J.A. Fishel, Z. Su, C.H. Lin, G.E. Loeb, Multi-modal synergistic tactile sensing, in *Tactile Sensing in Humanoids-Tactile Sensors and Beyond Workshop, 9th IEEE-RAS International Conference on Humanoid Robots* (2009)
51. H. Yousef, M. Boukallel, K. Althoefer, Tactile sensing for dexterous in-hand manipulation in robotics-A review. *Sens. Actuat. A* **167**(2), 171–187 (2011)
52. T.R. Grieve, J.M. Hollerbach, S.A. Mascaró, Force prediction by fingernail imaging using active appearance models, in *2013 World Haptics Conference*, pp. 181–186 (2013)
53. D. Hristu, N. Ferrier, R.W. Brockett, The performance of a deformable-membrane tactile sensor: Basic results on geometrically-defined tasks, in *2000 IEEE International Conference on Robotics and Automation*, pp. 508–513 (2000)
54. S.A. Mascaró, H.H. Asada, Photoplethysmograph fingernail sensors for measuring finger forces without haptic obstruction. *IEEE Trans. Robot. Autom.* **17**(5), 698–708 (2001)
55. Y. Sun, J.M. Hollerbach, S.A. Mascaró, Predicting fingertip forces by imaging coloration changes in the fingernail and surrounding skin. *IEEE Trans. Biomed. Eng.* **55**(10), 2363–2371 (2008)
56. Y. Sun, J.M. Hollerbach, S.A. Mascaró, Estimation of fingertip force direction with computer vision. *IEEE Trans. Rob.* **25**(6), 1356–1369 (2009)
57. S. Urban, J. Bayer, C. Osendorfer, G. Westling, B.B. Edin, P. Van Der Smagt, Computing grip force and torque from finger nail images using gaussian processes, in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4034–4039 (2013)
58. W. Yuan, R. Li, M.A. Srinivasan, and E.H. Adelson. Measurement of shear and slip with a GelSight tactile sensor, in *2015 IEEE International Conference on Robotics and Automation*, pp. 304–311 (2015)
59. K. Sato, K. Kamiyama, N. Kawakami, S. Tachi, Finger-shaped gelforce: Sensor for measuring surface traction fields for robotic hand. *IEEE Trans. Haptics* **3**(1), 37–47 (2010)
60. L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points, in *European Conference on Computer Vision*, pp. 640–653 (2012)

61. N. Kyriazis, A. Argyros, Physically plausible 3D scene tracking: the single actor hypothesis. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9–16 (2013)
62. Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, J. Chai, Video-based hand manipulation capture through composite motion control. *ACM Trans. Gr.* **32**(4), 43 (2013)
63. W. Zhao, J. Zhang, J. Min, J. Chai, Robust realtime physics-based motion control for human grasping. *ACM Trans. Gr.* **32**(6), 1–12 (2013)
64. M.A. Brubaker, L. Sigal, D.J. Fleet, Estimating contact dynamics, in *2009 IEEE International Conference on Computer Vision*, pp. 2389–2396 (2009)
65. B. Frank, R. Schmedding, C. Stachniss, M. Teschner, W. Burgard, Learning the elasticity parameters of deformable objects with a manipulation robot, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1877–1883 (2010)
66. B. Wang, L. Wu, K. Yin, U. Ascher, L. Liu, H. Huang, Deformation capture and modeling of soft objects. *ACM Trans. Gr.* **34**(4), 94:1–94:12 (2015)
67. F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, C. Duriez, Real-time control of soft-robots using asynchronous finite element modeling, in *2015 IEEE International Conference on Robotics and Automation*, pp. 2550–2555 (2015)