Tracking Fractures of Deformable Objects in Real-Time with an RGB-D Sensor

Antoine Petit, Vincenzo Lippiello, Bruno Siciliano

Tracking fractures of deformable objects in real-time with an RGB-D sensor

Antoine Petit, Vincenzo Lippiello, Bruno Siciliano
DIETI, Università degli Studi di Napoli Federico II, Italy
{antoine.petit, vincenzo.lippiello, bruno.siciliano}@unina.it

Abstract

This paper introduces a method able to track in real-time a 3D elastic deformable objects which undergo fractures, using the point cloud data provided by an RGB-D sensor. Our framework relies on a prior visual segmentation of the object in the image. The segmented point cloud is registered by non-rigidly fitting the mesh, based on the Finite Element Method to physically model elasticity, and on geometrical point-to-point correspondences to compute external forces exerted on the mesh. Fractures are handled by processing the stress tensors computed on the mesh of the FEM model, in order to detect fracturable nodes. Local remeshing around fracturable nodes is then performed to propagate the fracture. The real-time performance of the system is demonstrated on real data involving various deformations and fractures.

1. Introduction

Tracking non-rigid objects has aroused much interest in recent years in the computer vision, computer graphics and robotics communities, addressing various applications in fields such as augmented reality, medical imaging, robotic manipulation, by handling a huge variety of objects: tissues, paper, rubber, viscous fluids, cables, food, organs, etc.

This study comes within the scope of the RoDyMan project[2], consisting in a unified framework for robotic dynamic manipulation of deformable objects, as shown by the demonstration scenario in Fig. 1.

A particularly novel challenge lies in dealing with fractures of deformable objects, and in designing a model of both deformations and fractures, and fitting it with vision and range data. This registration problem also involves critical real-time concerns, which are especially required for instance for robotic dynamic manipulation. Although some recent studies have proposed efficient techniques to handle 3D objects which undergo isometric or elastic deformations, to our knowledge there exists no real-time 3D vision system which considers topological changes undergone by the objects, such as fractures. The aim of this paper is thus to propose, based on the method suggested in [18], a real-time tracking system able to handle elastic objects which may get fractured, using data provided by an RGB-D sensor. To cope with elastic deformations and fractures, our approach involves a physical modeling of the considered object, by relying on a Finite Element Method (FEM) model, with an implementation running in real-time. Our whole system is indeed able to run at around 30 frames per seconds. The remainder of the paper is organized as follows. Some works related to ours are presented in Sect. 2 and the system is outlined in Sect. 2.5. In Sect. 3 the physical deformation and fracture model of the object is introduced, Sect. 4 explains how the point cloud data is processed and matched with this model to perform registration. Finally, some experimental results are presented in Sect. 5.

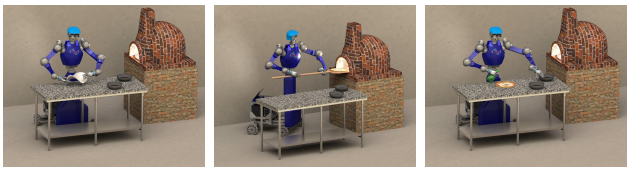

Figure 1: Artistic views of the RoDyMan robotic platform and the pizza making process.

2. Related works and motivations

In the literature, the various approaches proposed to register deformable objects, using range and vision data, could be classified according to the underlying model of the considered object and its physical realism.

2.1. Registration using implicit physical modeling

Based on implicit physical models, approaches in [11, 2, 19] use a 1D parametric curve or 2D splines models (B-splines, Radial Basis Functions) to track deformable objects

[2]http://www.rodyman.eu/ The research leading to these results has been supported by the RoDyMan project, which has received funding from the European Research Council (FP7 IDEAS) under Advanced Grant agreement number 320992. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

in monocular images. This class of methods relies on the minimization of 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. With point cloud data, methods in [10, 24] 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 [10] or an optimized GPU implementation [24], real-time performance can be achieved.

2.2. Registration using explicit physical modeling

Instead, another formulation of the problem relies on physics-based deformable models to perform registration, by modeling more explicitly elasticity. With respect to 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 be determined, by setting internal and external forces equal or, equivalently, minimizing energy functions. Physics-based methods include discrete mass-spring-damper systems [12, 5, 20], or more explicit approaches relying on the Finite Element Method (FEM), based on continuum mechanics. In [20], based on mass-spring-damper systems, 3D-3D correspondences, determined through a probabilistic inference, enable the computation of the external forces applied to the mesh. First attempts for registration employing the FEM for 3D surfaces in [3, 13] used linear elasticity FEM models. Haouchine *et al.* [9] uses a linear tetrahedral co-rotational FEM model, coping with larger elastic deformations, external forces being related to correspondences between tracked 3D feature points mapped to the 3D mesh by means of a stereo camera system.

2.3. Handling topological changes

As methods dealing with topological changes of a 3D deformable object, we can mention 3D reconstruction methods [22, 23], based on the TransforMesh system to address mesh evolution and topological changes. However these are model-free methods and reconstruct a single mesh at each time step [23] or are not suited for real-time applications [22]. Instead, the goal is here to continuously estimate the rigid transformations, the deformations and splitting topological change (or fractures) undergone by the object, modeled by a known mesh, in real-time.

2.4. Motivations and contributions

Since our system would attempt to handle elastic volumetric strains and fractures, a realistic mechanical model, based on the FEM, has been adopted. The recent suitability of these physical models for real-time applications, as demonstrated by promising approaches [20, 9], has confirmed our choice. In contrast to [22, 23], fractures could this way be modeled physically, using linear elastic fracture mechanics, through the internal forces acting on the object. We assume the prior knowledge of a consistent mesh (which could be automatically reconstructed offline) and of the material properties (through the Young modulus, the Poisson ratio, and a fracture threshold), which could be estimated offline. With respect to related approaches [10, 20, 9, 24], our main contribution has been to integrate the fracture model, along with being able to track large deformations, and handling these tasks in real time (30 fps), making our solution the first real-time vision tracking system able to deal with deformations and fractures of a non-rigid objects, using a RGB-D sensor.

2.5. Overview of the system

Our tracking system can be outlined as follows: Input : the 3D volumetric mesh of the object, a given RGB-D data, and assuming a fair registration at the previous time step.

1. Visual segmentation of the considered object, with a graph cut-based approach ensuring temporal coherence.

2. Using the resulting segmented point cloud, compute external linear elastic forces exerted on the vertices of the mesh.

3. Compute elastic stresses and internal forces, based on a tetrahedral linear co-rotational FEM model.

4. Numerical resolution of mechanical equations to compute the deformations.

5. Detect fracturable nodes in the mesh.

6. Split the fracturable nodes and perform local remeshing around them according to the fracture plane.

3. Modeling deformations and fracture: an FEM approach

Since we deal with objects which may undergo elastic deformations, a major issue lies in the definition of a relevant physical model. The Finite Element Method (FEM) [4] provides a realistic physical model, by relying on continuum mechanics, instead of finite differences for mass-spring systems for instance. It consists in tessellating the deformable object into a mesh made of elements. We rely here on a volumetric linear FEM approach with tetrahedral elements. The deformation field $\mathbf{u}_e$ over an element e is

then approximated as a continuous interpolation of the displacements $\hat{\mathbf{u}}_e$ of its four vertices, through a polynomial basis function contained in the 3×12 matrix $\mathbf{N}_e(\mathbf{x})$:

$$\mathbf{u}_e(\mathbf{x}) = \mathbf{N}_e(\mathbf{x})\hat{\mathbf{u}}_e, \tag{1}$$

with being $\hat{\mathbf{u}}_e = \mathbf{x}_e - \mathbf{x}_{e,0}$, $\mathbf{x}_e$ and $\mathbf{x}_{e,0}$ respectively the deformed and initial world coordinates of the vertices of e.

3.1. Modeling elastic deformations

In order to model deformations and elasticity, we resort to the infinitesimal strain theory to compute the Cauchy's linear strain tensor $\epsilon_\mathbf{e}$ within the tetrahedron. It can be linearly expressed with respect to $\hat{\mathbf{u}}_e$, using Voigt notations since $\epsilon_\mathbf{e}$ is symmetric:

$$\epsilon_e \quad = \quad \mathbf{L}_e\hat{\mathbf{u}}_e, \tag{2}$$

with $\mathbf{L}_e$ a constant 6×12 matrix. We then rely on Hooke's law linear elasticity theory for a continuous isotropic material, which lead us to write the infinitesimal stress tensor $\sigma_\mathbf{e}$ in the element as:

$$\sigma_e = \mathbf{C}_e\epsilon_e = \mathbf{C}_e\mathbf{L}_e\hat{\mathbf{u}}_e, \tag{3}$$

where $\mathbf{C}_e$ is a 6×6 symmetric matrix depending on two elastic parameters of the material, the Young modulus E and the Poisson ratio ν. By deriving the strain energy in e, the internal elastic forces $\mathbf{f}_e$ exerted on the four vertices of e of the mesh can be linearly related to their displacements $\hat{\mathbf{u}}_e$:

$$\mathbf{f}_e \quad = \quad \mathbf{K}_e\hat{\mathbf{u}}_e = V_e\mathbf{L}_e^T\sigma_e. \tag{4}$$

Although it is insensitive to translation transformations, the model, by using an infinitesimal approximation of the strain tensor, is however inaccurate when modeling large rotations of the elements, leading for instance to non-zero summations of the forces. A work-around consists in the co-rotational approach [6], used for registration purposes in [9], which is a good compromise between the ability to model large linear elastic deformations and computational efficiency. Since the displacement of an element can be decomposed into a rigid transformation and a pure deformation, the idea is to extract the rotation matrix $\mathbf{R}_e$ related to the rigid transformation. Then the stiffness matrix can be warped with respect to this rotation, so as to accommodate rotation transformations, giving:

$$\mathbf{f}_e = \mathbf{R}_e\mathbf{K}_e\hat{\mathbf{u}}_e^r = \mathbf{R}_e\mathbf{K}_e(\mathbf{R}_e^{-1}\mathbf{x}_e - \mathbf{x}_{e,0}), \tag{5}$$

with being $\hat{\mathbf{u}}_e^r = \mathbf{R}_e^{-1}\mathbf{x}_e - \mathbf{x}_{e,0}$, with $\mathbf{R}_e^{-1}\mathbf{x}_e$ the deformed coordinates of the vertices of e back rotated to the unrotated frame, the forces $\mathbf{K}_e\hat{\mathbf{u}}_e^r$ being then re-rotated to the current deformed element through the multiplication by $\mathbf{R}_e$. In this way, the overall forces on the whole mesh can be summed to zero, while computational efficiency is ensured since $\mathbf{K}_e$ can be computed in advance, in contrast to non-linear FEM approaches.

3.2. Modeling fractures

In the computer graphics community, various systems modeling fractures have been proposed. The detection of fractures or cracks in 3D object simulations has been initially achieved using mass-spring systems [16], by removing springs whose lengths exceed a certain threshold. Based on the FEM, [17] proposed a physically consistent method using the linear elastic fracture theory [1], through the computation, for given strains, of tensile forces undergone by the nodes. A fracture is then propagated from each detected fracturable node (or crack tip) by subdividing the neighboring tetrahedra according to a fracture plane. In a similar manner, [15, 14] have suggested a more computationally efficient solution by resorting to element wise stresses, the fracture being propagated along element boundaries. Although improvements have been achieved from these widespread approaches, for both fracture detection and propagation [8], we propose here an approach based on [17]. One reason, as justified hereafter in section 3.2.2, is to be able to continuously cope properly with the provided noisy point cloud data. Another reason is to preserve real-time performances.

3.2.1 Detecting fractures

Detecting fractures in the mesh is performed in a similar way to [17], by decomposing the internal forces exerted on the nodes into tensile and compressive forces, with the difference that [17] relies on a standard linear FEM method.

For this purpose, the stress tensor σ_e undergone by a tetrahedra is separated into a tensile and a compressive components. Here, using a co-rotational approach and based on (5) and (3), we compute σ_e as:

$$\sigma_e = \mathbf{C}_e\mathbf{L}_e\hat{\mathbf{u}}_e^r = \mathbf{C}_e\mathbf{L}_e\mathbf{R}_e^{-1}\mathbf{x}_e - \mathbf{x}_{e,0}. \tag{6}$$

Then, as in [17], the three eigenvalues $v_i(\sigma_e)$ and eigenvectors $\mathbf{n}_i(\sigma_e)$ of σ_e are computed, the positive eigenvalues ones corresponding to tensile stresses whereas negative ones to compressive stresses. Tensile and compressive tensors can be then defined as:

$$\sigma_e^+ \quad = \quad \sum_i \max(0, v_i(\sigma_e))\mathbf{m}(\mathbf{n}_i(\sigma_e)) \tag{7}$$

$$\sigma_e^- \quad = \quad \sum_i \min(0, v_i(\sigma_e)\mathbf{m}(\mathbf{n}_i(\sigma_e)), \tag{8}$$

with, for any vector $\mathbf{a} \in \mathbb{R}^3$

$$\mathbf{m}(\mathbf{a}) = \begin{cases} \frac{\mathbf{a}\mathbf{a}^T}{\|\mathbf{a}\|} \text{ if } \|\mathbf{a}\| > 0 \\ \mathbf{0} \quad \text{if } \|\mathbf{a}\| = 0. \end{cases} \tag{9}$$

Based on these tensile and compressive stresses, internal forces exerted on the vertices of e can be decomposed as

well, using (4):

$$\mathbf{f}_e^+ = V_e \mathbf{L}_e^T \sigma_e^+ \quad (10)$$
$$\mathbf{f}_e^- = V_e \mathbf{L}_e^T \sigma_e^- . \quad (11)$$

Let us first define $X = \{\mathbf{x}_j\}_{j=1}^{n_X}$ the set of vertices of the mesh. Then for each node $\mathbf{x}_j$ in the mesh, we can compute, as in [17], the Separation Tensor ζ_j, using the sets $\{\mathbf{f}_{e,j}^+ | j \in e\}_e$ and $\{\mathbf{f}_{e,j}^- | j \in e\}_e$ respectively of tensile and compressive forces exerted by the elements $\{e | j \in e\}$ attached to the node:

$$\zeta_j = \frac{1}{2}(-\mathbf{m}(\mathbf{f}_j^+) + \sum_{\mathbf{f}_{e,j} \in \{\mathbf{f}_{e,j}^+ | j \in e\}_e} \mathbf{m}(\mathbf{f}_{e,j}) + \mathbf{m}(\mathbf{f}_j^-) - \sum_{\mathbf{f}_{e,j} \in \{\mathbf{f}_{e,j}^- | j \in e\}_e} \mathbf{m}(\mathbf{f}_{e,j})) \quad (12)$$

with

$$\mathbf{f}_j^+ = \sum_{\mathbf{f}_{e,j} \in \{\mathbf{f}_j^+ | j \in e\}_e} \mathbf{f}_{e,j} \text{ and } \mathbf{f}_j^- = \sum_{\mathbf{f}_{e,j} \in \{\mathbf{f}_j^- | j \in e\}_e} \mathbf{f}_{e,j} \quad (13)$$

ζ_j is a tensor which enables to evaluate deformation imbalance between tensile and compressive forces, while being invariant to imbalance resulting in rigid motions. A fracture at node $\mathbf{x}_j$ is then detected if the largest positive eigenvalue v_j^+ of ζ_j is above a certain threshold corresponding to the toughness of the material. The eigenvector $\mathbf{n}_j$ corresponding to v_j^+ defines the normal to the fracture plane at node $\mathbf{x}_j$.

3.2.2 Fracture propagation and remeshing

In order to propagate the fracture from the detected fracturable nodes, in the direction of the fracture plane, the mesh around the considered node shall be modified. [17] proposes to subdivide the elements which are attached to the node and intersected by the fracture plane, according to the orientation of the plane. Other neighboring elements also need to be modified to maintain the consistency of the mesh. However, such a remeshing process, which is proper and suitable for simulations purposes since it manages to maintain the orientation of the fracture, is computationally costly and tends to generate an always growing number of elements, what is not desirable in our case to keep real-time performances. Also, for registration concerns, by changing locally and drastically the resolution of the mesh, matching the remeshed regions with the acquired point cloud, which is of constant and homogeneous resolution, while being noisy, may lead to heterogeneous or unstable behaviors. Another approach, suggested in [21, 15, 14], consists in propagating the fracture along the boundaries of the neighboring elements. This method, although it enables to keep the resolution constant, tends to produce jumbled fracture patterns and artifacts, which is also not desirable in our case when coping with the noisy point cloud data.

We have instead opted here for a solution preserving the mesh homogeneity and performing a neat fracture propagation. It is shown in Fig. 2. As in [17], the fracturable node is first replicated, with the same positions. Then, within the set of elements attached to the original fracturable node, the ones which are not intersected by the given fracture plane are determined and are reassigned to the original fracturable node or to the replicated one, given on which side of the plane they are located. This procedure appears coarse, affects the volume of the model and tends to degenerate the mesh. But it remains acceptable since we consider, in the experimental scenarios proposed in Sect. 5, objects undergoing simple fracture events, so that element losses in the mesh can be neglected.

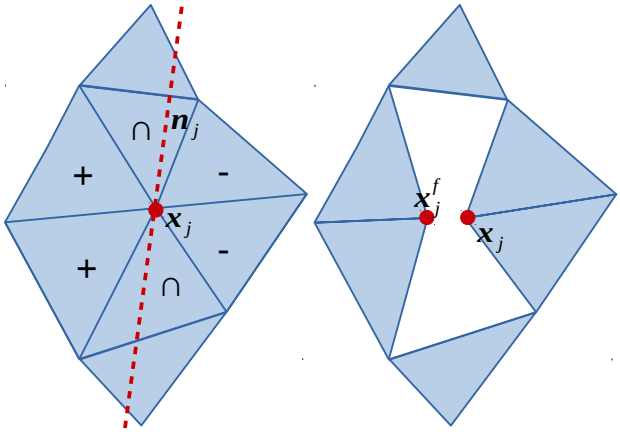

Figure 2: Remeshing procedure, represented here in the 2D case with triangles for simplicity. The fracturable node $\mathbf{x}_j$ is replicated, initially with the same position, providing a new vertex $\mathbf{x}_j^f$ for the mesh (it is here translated from $\mathbf{x}_j$ for clarity). The intersected elements are removed. The elements on the positive side of the fracture plane (determined by its normal $\mathbf{n}_j$) assigned to $\mathbf{x}_j^f$ and the others remain attached to $\mathbf{x}_j$.

4. Registration with the point cloud data

The deformable registration problem consists in fitting the point cloud data provided by an RGB-D sensor, with the tetrahedral mesh. The basic idea is to derive external forces exerted by the point cloud on the mesh and to integrate these forces, along with the internal forces computed using the physical model presented in Sect. 3, into a numerical solver solving the resulting mechanical equations.

4.1. Segmentation and deformable iterative closest point

The approach presented here uses the same methods as the ones described in the work [18] to segment the point cloud, relying on the Grabcut visual segmentation technique, in order to restrict the acquired point cloud to the considered object, so as to avoid ambiguities, in the matching process with the background or with occluding shapes.

The deformable iterative procedure worked out in [18] is also employed, first to determine the nearest neighbors correspondences, both from the segmented point cloud to the mesh and from the mesh to the segmented point cloud.

Based on the two sets of mesh-to-point cloud and point cloud-to-mesh correspondences, external elastic forces exerted on the vertices of the mesh can be computed. When computing these forces, a trade-off has to 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. Let us note that the contour weighting function proposed in [18] is not used here and that occlusions is not an issue we want to cope with in this case.

Estimating the deformations of the mesh consists in solving a dynamic system of non-linear ordinary differential equations involving the internal forces, determined with (5), and the external forces, based on Lagrangian dynamics.

4.2. Processing fractures

Once deformations are computed, topological changes due to fractures, as modeled in Sect. 3.2, are handled. For each vertex $\mathbf{x}_j$ in X, we detect if it is a fracture node, based on Sect. 3.2.1, and for fracturable nodes the remeshing procedure presented in Sect. 3.2.2. Let us not 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 X^f is obtained and will be directly employed for the next frame.

5. 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.

5.1. Implementation and experimental set-up

For the non-rigid registration phase, we have employed the Simulation Open Framework Architecture (SOFA) simulator [7], which enables to deal with various physical models and to evolve simulations 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 2.4GHz 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. 3, 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 Modulus being empirically set to $E = 2.0MPa$ and the Poisson ratio to $\nu = 0.01$. The plate object (see Fig. 3, right) 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 $E = 2.5MPa$ and $\nu = 0.01$ in this case.

Three cases of fracture are here demonstrated: fracture due to opening by excessive bending for the bar object and fractures due to shearing and to opening by stretching for the plate object, tearing the object. Let us note that in the first case the parameter λ ($0 \leq \lambda \leq 1$) which tunes the balance between mesh-to-point cloud and point cloud-to-mesh forces (see [18]) is set to $\lambda = 0.5$. In the second case $\lambda = 0.1$ so that vertices of the mesh 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 [18], the registration process is iteratively repeated (3 iterations were performed in the experiments). Here only one iteration is executed to maintain real-time performances, since fracture detection remains costly (see Fig. 1).

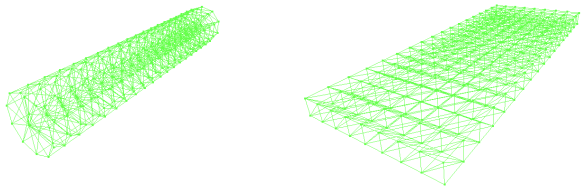

Figure 3: Meshes of both the cylindrical bar (left) and plate (right) objects.

5.2. Results

As seen in Fig. 4, opening though bending deformations can be tracked quite properly for the bar object, so tensile internal forces computed on the nodes remain physically consistent, making the fracture model valid, as shown on the third column. For the plate object, tearing it through out-of-plane shearing (see Fig. 5) can also be tracked, despite some artifacts can be noticed on some unfractured regions.

However, when opening the plate object by stretching it, as shown in Fig. 6, only vertices lying on the occluding con-

tour 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, fracture in this case appears with a certain delay, when both fractured parts split away. However, the fracture crack is quite coherent and both parts are finally consistently recovered.

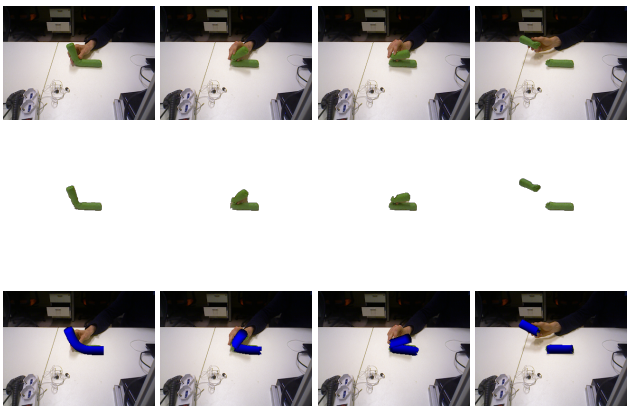

Figure 4: Results of the tracking process for the cylindrical bar object, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image.

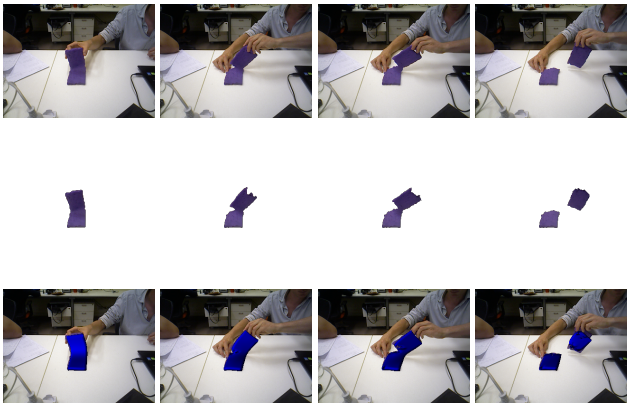

Figure 5: Results of the tracking process for the plate object, torn through out-of-plane shearing, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image.

Computational costs Regarding computational aspects, in Tab. 1 are shown the mean computation times of the various phases of the algorithm, for the sequence presented in Fig. 5. *Ext. forces* is the step involving the determination of the closest points between the mesh and the point cloud, and the computation the subsequent external forces exerted

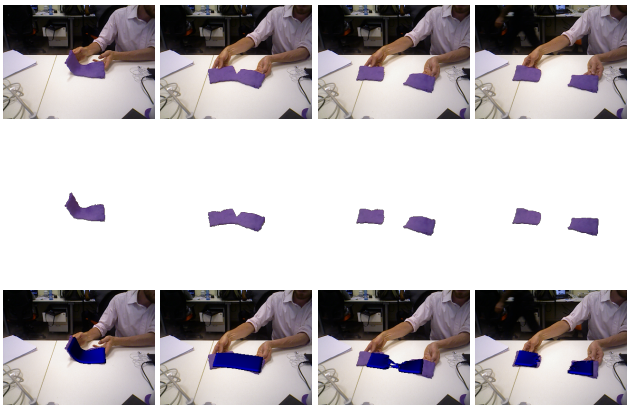

Figure 6: Results of the tracking process for the plate object, torn through opening, with the input images (first row), the segmented frames (second row), and the registered mesh reprojected in the input image.

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 in 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 fps.

Phases	Mean execution times (ms)
Segmentation	10.1
Ext. forces	3.9
Resolution	2.3
Int. forces & fracture detection	20.6
Remeshing	0.6
Total	37.5

Table 1: Mean execution times, in milliseconds, for the different phases of the approach.

6. Discussion

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), the proposed method being for the moment mostly suitable for such materials. With more elastic materials, the estimated deformations and internal forces of the mesh would be rougher, resulting in a more variable sepa-

ration tensor and so potentially spurious detections of fractures.

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

Tracking stretching deformations As observed in the experiments, 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. A method proposed in the work [18] suggests to weight the vertices of the visible surface of the mesh, given their distance to the occluding contour of the mesh. However, this technique tends to alter the physical homogeneity of the material, making in the case of a fracture model 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 a progressive degeneration of the mesh if several fracture events occur.

7. Conclusion

In this paper we propose a method to track fractures of deformable objects, as a first attempt to deal with such topological changes. The main idea has been to integrate in a physical FEM based deformation model a model of fracture events, by first detecting fracturable nodes in the considered mesh, by decomposing the internal forces into tensile and compressive components. A second step consists in a simple local remeshing of the elements around the fracture nodes. The registration process with the RGB-D data involves performing a deformable iterative closest point procedure to compute the deformations and finally investigating fracture events over the mesh and eventually carrying out remeshing. Experimental results on two different objects and three sorts of fractures show the relevance of this simple physical based method, which can however be improved by handling more properly the remeshing phase, by dealing with larger elastic, due to stretching actions for instance, and plastic deformations, and by handling occlusions by distinguishing them with fracture cracks.

References

[1] T. L. Anderson. *Fracture mechanics: fundamentals and applications*. CRC press, 2005.

[2] A. Bartoli, A. Zisserman, et al. Direct estimation of non-rigid registrations. In *British Machine Vision Conference*, pages 899–908, 2004.

[3] L. D. Cohen and I. Cohen. Deformable models for 3-d medical images using finite elements and balloons. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 592–598. IEEE, 1992.

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

[5] C. Elbrechter, R. Haschke, and H. Ritter. Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1427–1432. IEEE, 2011.

[6] O. Etzmuß, M. Keckeisen, and W. Straßer. A fast finite element solution for cloth modelling. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 244–251. IEEE, 2003.

[7] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, pages 283–321. Springer, 2012.

[8] L. Glondu, M. Marchal, and G. Dumont. Real-time simulation of brittle fracture using modal analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 19(2):201–209, 2013.

[9] N. Haouchine, J. Dequidt, I. Peterlik, E. Kerrien, M.-O. Berger, and S. Cotin. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 199–208. IEEE, 2013.

[10] A. Jordt and R. Koch. Direct model-based tracking of 3d object deformations in depth and color video. *International Journal of Computer Vision*, pages 1–17, 2013.

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[12] V. Lippiello, F. Ruggiero, and B. Siciliano. Floating visual grasp of unknown objects using an elastic reconstruction surface. In *Robotics Research*, pages 329–344. Springer, 2011.

[13] T. McInerney and D. Terzopoulos. A finite element model for 3d shape reconstruction and nonrigid motion tracking. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 518–523. IEEE, 1993.

[14] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[15] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. *Real-time simulation of deformation and fracture of stiff materials*. Springer, 2001.

[16] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. *The visual computer*, 7(4):210–219, 1991.

[17] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co., 1999.

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

[19] J. Pilet, V. Lepetit, and P. Fua. Fast non-rigid surface detection, registration and realistic augmentation. *Int. Journal of Computer Vision*, 76(2):109–122, Feb. 2007.

[20] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1130–1137. IEEE, 2013.

[21] J. Smith, A. Witkin, and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. In *Computer Graphics Forum*, volume 20, pages 81–91. Wiley Online Library, 2001.

[22] K. Varanasi, A. Zaharescu, E. Boyer, and R. Horaud. Temporal surface tracking using mesh evolution. *Computer Vision–ECCV 2008*, pages 30–43, 2008.

[23] A. Zaharescu, E. Boyer, and R. Horaud. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):823–837, 2011.

[24] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics, TOG*, 2014.