Visual Tracking of Deformable Objects with RGB-D Camera

Agniva Sengupta

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE UNIVERSITE BRETAGNE LOIRE

ÈCOLE DOCTORALE N° 601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Agniva SENGUPTA

Visual Tracking of Deformable Objects with RGB-D Camera

Thèse présentée et soutenue à l'IRISA, le 29 juin 2020
Unité de recherche : INRIA - Rainbow

Rapporteurs avant soutenance :

Andrea CHERUBINI	Maitre de conférences HDR, Université de Montpellier
Gilles SIMON	Maitre de conférences HDR, Université de Lorraine

Composition du jury :

Présidente :	Luce MORIN	Professeure des Universités, INSA de Rennes
Examinateurs :	Stéphane COTIN	Directeur de recherche Inria, Strasbourg
	Andrea CHERUBINI	Maitre de conférences HDR, Université de Montpellier
	Gilles SIMON	Maitre de conférences HDR, Université de Lorraine
	Eric MARCHAND	Professeur des Universités, Université de Rennes 1
	Alexandre KRUPA	Chargé de recherche Inria, HDR, Rennes
Dir. de thèse :	Alexandre KRUPA	Chargé de recherche Inria, HDR, Rennes
Co-dir. de thèse :	Eric MARCHAND	Professeur des Universités, Université de Rennes 1

ACKNOWLEDGEMENT

This thesis would not have been possible without the guidance and supervision of Alexandre Krupa and Eric Marchand. This thesis is a result of their efforts, suggestions and timely encouragement. I would also remember them for being extremely nice, especially Alexandre for his discernment and wit and Eric for his occasional words of wisdom that were extremely helpful. I would also like to thank Maud Marchal for her insightful comments and help during our brief interaction.

I would like to thank the members of the jury, Stéphane Cotin, Andrea Cherubini, Gilles Simon and Luce Morin for reading and reviewing my thesis as well as for the invigorating discussion during the defense.

I have to thank François Chaumette and Paolo Robuffo Giordano for welcoming and supporting me in Lagadic and Rainbow. Fabien Spindler had been wonderfully helpful to me on multiple occasions and I thank him for his efforts. And a lot of thanks to Hélène de la Ruée for the numerous helps during the entirety of my PhD. I thank Ekrem Misimi for all the wonderful interactions over the years, as well as for hosting me in Trondheim.

A lot of thanks to the entire team of Rainbow for the wonderful memories throughout the years. I would fondly remember the co-workers in my office including Alexander, Fabrizio, Souriya and Aly, as well as Sunny, Daniel and Steven. I want to thank Pedro, Usman, Ide-flore, Firas, Hadrien, Marco & Marco, Bryan, Rahaf, Quentin, Wouter, Samuel, Javad, Ramana and Florian for all the good moments. A special thanks to Romain Lagneau for not only being an incredible collaborator but also a good friend.

I want to take this opportunity to thank my friends at the AGOS for some wonderful experiences, including Sebastien, Raphaël and everyone else. And a special thanks to Patrick and Pierre for putting up with me in Fontainbleau. I must also mention that I will fondly remember the chats at the cafét with Vijay and Sudhanya, thanks for the wonderful time.

Thanks to my wife for supporting me throughout the thesis and bearing with me through this often difficult journey of PhD. And I thank my parents for enabling me to pursue my dreams in the first place. However, it is not nearly possible to thank my family sufficiently enough with a few words of acknowledgement.

TABLE OF CONTENTS

INTRODUCTION

Salvador Dalí's illustrious career of over six decades was well known for his famous artworks that interpret common, rigid objects as flexible. However, little is said or known about an obscure and unfinished project of his own, the *hundred-meter horse*. In 1980, Salvador Dalí conceptualized the model of a horse with its shoulder and rump stretched apart over a distance of several meters while it appeared to be proportionally appropriate when viewed from the right perspective [Banchoff et al., 2014]. Due to reasons which remains unclear, this project of designing the '*hundred-meter horse*' never really took off. By 1982, Dalí had rechristened his original idea as *Horse from the Earth to the Moon*. In this renewed project, the shoulder of the horse was supposed to be on a mountaintop while the rump shall be on the moon. Evidently, none of these ideas ever materialized. However, despite the obvious absurdity of the concept, it is likely that Dalí's model of the horse stretched from the earth to the moon, if constructed and viewed from the right place at the right time, would appear to be in perfect proportions.

This example is representative of the conundrum faced by all systems that involve viewing deformed objects from an arbitrary point in space. The visual perception of a deformed shape is highly dependant on the perspective of the viewer. Any attempt made towards geometrically explaining these shapes tend to become a highly unconstrained problem. If the system used for observing the deformation involves a projection operation, we loose another dimension of space in the captured data. Projection makes it even more challenging to resolve the structure of the object that is being viewed.

The fundamental principles of computer vision were originally developed with rigid objects and rigid scene as the only input. However, the actual physical world is made up of a very large amount of non-rigid objects. To be of practical utility, computer vision and vision based control systems needed to evolve techniques for dealing with non-rigid objects as well. This is especially true when robotic devices attempt to interact with soft objects.

To deal with the analysis of deforming objects in an observed visual data, it is necessary to have a mathematical prior to explain the deformation. The mathematical priors form a deformation model, which is a set of rules that the deforming object can

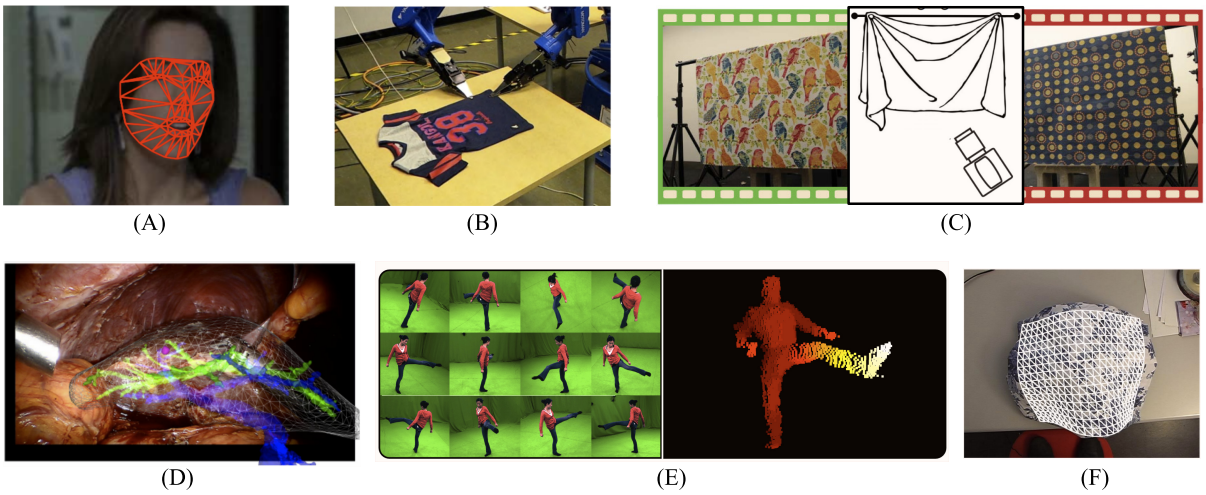


FIGURE 1 – Different approaches and techniques for deformation tracking have been used in diverse application areas such as face tracking [Agudo et al., 2014] (A), manipulation of soft objects using robotic grippers [Sun et al., 2018] (B), estimation of material properties of deformable surfaces [Davis et al., 2015] (C), augmented reality on deformable objects, including surgical applications [Haouchine et al., 2013] (D), motion capture [Helten et al., 2013] (E) or deformation tracking of generic objects [Salzmann et al., 2008] (F)

be expected to follow. Every deformation tracking approach needs to have some deformation model for explaining the observed behaviour of the object it intends to track. In the literature, a combination of geometric, visual and physics-based models have been used in conjunction with standard computer vision techniques to track deformations with varying degree of accuracy. In the last few decades, the field of non-rigid object tracking using visual information has evolved rapidly. Fig. 1 shows a pictorial summary of some of the relevant application areas that have been addressed by the non-rigid tracking research in the recent years. Moreover, the advent of modern GPUs has enabled the tracking of deforming surfaces at frame rate. However, despite these advances, there are many challenges that need to be addressed before deformation tracking using computer vision can be considered as a problem that has been 'solved'. In this thesis, we will look at those problems and offer some novel methods to address some of those challenges.

This thesis addresses the problem of tracking the temporal evolution of the surface of a deforming object using visual information only. The visual information is captured using a depth camera. The thesis has been developed with future industrial applications in mind, such that the methodologies developed here can be utilized to aid and

enable the manipulation of soft objects by robotic grippers and actuators.

Deformation tracking using depth cameras has many practical utilities. Although robotic manipulation using deformation tracking remains the primary application area of this thesis, similar or closely-related approaches can also be utilized for surgical robotics [Moerman et al., 2009], motion capture [Hughes, 2012], augmented reality [Huang et al., 2015] and non-rigid SLAM [Agudo et al., 2011].

In the recent years, many approaches have been proposed to address the problem of tracking non-rigid objects. However, due to reasons discussed in details throughout the dissertation, we chose to utilize physics based approaches for deformation modelling. Not only does this approach impart physical reality to the deformation tracking methods proposed in this thesis, this approach also enables us to extend the deformation tracking algorithms towards robotic applications. These applications could have significant industrial use-cases by themselves.

Motivation

In most industrial robotic applications, monocular or depth camera is used as the primary sensor (apart from tactile sensing). To manipulate complex objects in real-time, it is necessary to know the position of the object along all 6 degrees-of-freedom (DoF) with respect to the robot. If the object happens to be deforming and changing its shape, it is necessary to track not just the 6-DoF of the object, but to track the entire surface. This can be done with or without a CAD model of the object. However, without a model, the problem statement becomes more unconstrained. Moreover, without the object model, it is not possible to have any idea about the occluded or hidden surface of the object. In an industrial setup, an approximate CAD model of the object that needs to be manipulated is usually readily available. There are many industrial robotic assemblies which deal only with known objects that have been identified beforehand.

This leads us to opt for model based tracking of deforming objects. However, we ensure that all the approaches proposed in this thesis work accurately with a coarse model of the object, since the process of reconstructing a fine mesh of a random, complex shaped object is significantly more complex (and expensive) than acquiring an approximate 3D model. Moreover, the tracking accuracy has to be high enough and robust enough to handle occlusion and sensor noise.

Over the second half of the dissertation, we progressively arrive towards addressing

a secondary problem statement. This involves estimating the elasticity parameters of an object while we visually track its deformation. This has been made possible largely due to the presence of the physics based model for deformation tracking.

Contributions

This dissertation makes the following contributions to the field of non-rigid object tracking:

— A rigid object tracking methodology is developed to track complex shaped objects using a coarse object model ;

— A physics based model is utilized to track deforming objects by minimization of depth based error by estimating the gradient of error using repeated simulation of deformations ;

— A similar physics based model is used for minimization of geometric and photometric cost functions using a novel approach to analytically approximate the gradient of error, thereby avoiding the expensive simulations for gradient estimation at every iteration ;

— Having developed a non-rigid object tracking methodology, we utilize this approach to estimate the physical elasticity parameter of a deforming object using a robotic actuator and also use the estimated parameter to track the forces being applied on the object using visual information only

Structure of the thesis

The manuscript of this dissertation is organized as follows:

Chapter 1 introduces some of the fundamental concepts of computer vision and geometry. The first part of this chapter recalls some of the classical algorithms for rigid object tracking. The second half devotes itself to analyzing the existing methods for non-rigid object tracking, as available in the literature. We explain some of the relevant concepts in details and analyze some of the important results and observations from the pertinent literature.

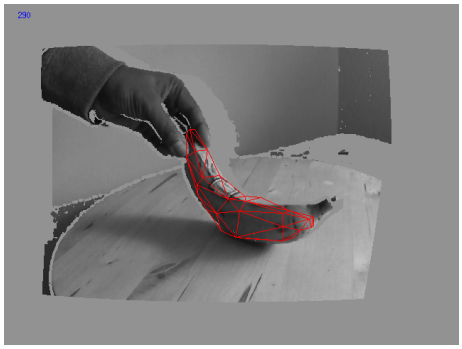


FIGURE 2 – A banana being tracked by the approach of Chapter 2

In **Chapter 2**, we describe a method to accurately track rigid objects of complex shapes using a coarse object model with the help of RGB-D cameras. We describe the methodology in details and validate the approach on simulated and real data. On simulated data, we compare our approach to a similar approach from state-of-the-art and present the quantitative comparison.

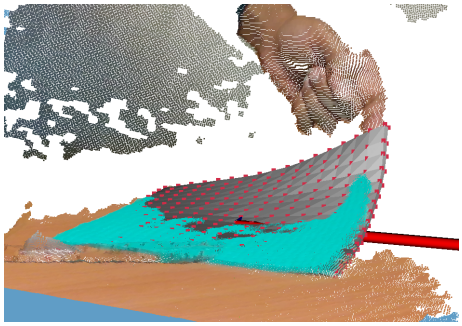


FIGURE 3 – The deformation of a sponge being tracked by the approach of Chapter 3

Chapter 3 presents the first approach to track non-rigid objects using RGB-D data. We describe the methodology, the details of the cost function and the proposed strategy for minimization. We validate our approach on simulated and real data, demonstrating that the approach is invariant to the accuracy of the physical parameters provided as an input to the deformation model.

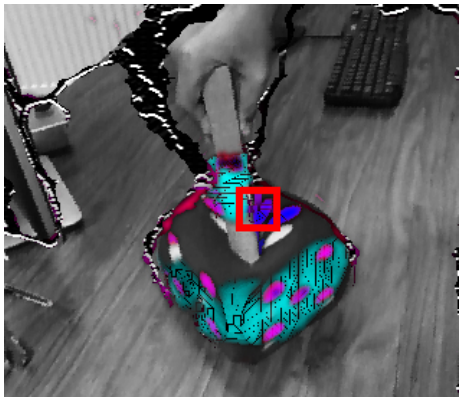


FIGURE 4 – The deformation of a soft dice being tracked by the approach of Chapter 4

In **Chapter 4**, we reformulate the deformation tracking approach to minimize a combination of photometric and geometric error using an approximately analytic gradient estimation technique. We first explain the methodology in a generalized sense, such that the framework remains valid for any kind of visual error minimization. We then proceed to explain the specific example of minimization using a combined geometric and photometric error term. We compare this proposed approach to state-of-the-art methodologies, as well as the method introduced in the previous chapter. We also validate the method on real objects as well as extend the approach to track the force applied on an object, when the physical elasticity parameters of the object is known.

Chapter 5 deals with robotic applications of deformation tracking. We address the problem of estimating the elasticity parameter of an object using a combination of deformation tracking and force measurements from a robotic actuator. Once the elasticity parameter has been measured, we proceed to utilize the approach proposed in Chapter 3 for estimating deforming forces acting on the object, using visual information only. The approach proposed in Chapter 5 was done as a joint work with Romain Lagneau.

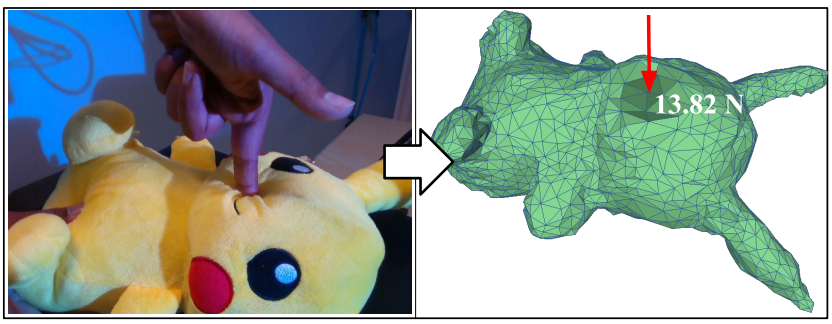


FIGURE 5 – Tracking of contact force using the approach of Chapter 5

Contributions

We present a brief summary of the contributions that were disseminated during this Ph.D.

Publications

This thesis resulted in the following **publications** being accepted in international conferences :

— Sengupta, A., Krupa, A. and Marchand, E., 2019, September. RGB-D tracking of complex shapes using coarse object models. In *2019 IEEE International Conference on Image Processing (ICIP)* (pp. 3980-3984)

— Sengupta, A., Krupa, A. and Marchand, E., 2019, October. Tracking of Non-Rigid Objects using RGB-D Camera. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC) (pp. 3310-3317)*

— Sengupta, A., Lagneau, R., Krupa, A., Marchand, E. and Marchal, M., 2020, June. Simultaneous Tracking and Elasticity Parameter Estimation of Deformable Objects. In *IEEE Int. Conf. on Robotics and Automation, (ICRA)*

Dataset

The following **dataset** were created and made publicly available for aiding future research and for enabling future comparison with our proposed approaches :

— github.com/lagadic/nr-dataset.git

— github.com/lagadic/VisualDeformationTracking.git

Videos

The following **videos** have been made publicly available for demonstrating our results from the methods proposed throughout this thesis :

— Demonstration of rigid object tracking methodology : youtu.be/_TPJkleBu3w

— Demonstration of non-rigid object tracking using depth information only : youtu.be/RFd-Ix9hcdg

— Demonstration of non-rigid object tracking using depth and photometric information : youtu.be/y4uuXpWFrvE

— : Robotic applications of deformation tracking : youtu.be/k1MPnmqmovQ

CHAPTER 1

BACKGROUND

Non-rigid object tracking is a complex problem to solve. It becomes even more challenging when it is required to be done in real-time. This has only been made possible by the recent developments over the last decade or so. Rigid object tracking, on the other hand, is a very well-developed research area. There are multiple alternative approaches available in the state-of-the-art for handling the problem of tracking deforming objects using visual sensors. However, a many existing non-rigid tracking methods utilize some format of rigid object tracking as an initial registration step. Given this context, we commence our study of non-rigid object tracking by first analyzing some fundamental principles involving rigid object tracking. This is followed by a survey of the existing literature of deformation tracking, as well as a brief explanation of some underlying principles.

The first half of this chapter (Sec. 1.1) introduces some of the preliminary mathematical notations required for a detailed study of rigid object tracking (Sec. 1.2), while the second-half (Sec. 1.3) dives into the state-of-the-art for deformation tracking.

1.1 Preliminary Mathematical Definitions

Since the state-of-the-art is significantly diverse in terms of approaches used for tracking deformations, we confine ourselves to describing only those preliminary notations and fundamentals which are strongly relevant to the method proposed in the thesis. Some additional approaches are discussed in section 1.3, as required.

Since the overall approach for deformation tracking proposed in this paper broadly involves visual tracking using mechanical object models, the Sec. 1.1 is splitted into two sub-parts, one which recalls the classical mathematical tools used in visual tracking of rigid objects and the other describing the details of the mechanical models.

1.1.1 Frame Transformation

In this thesis, the Euclidean transformation of points, lines and planes are almost always restricted to the rotation group $\mathrm{SO}(3)$ and the special Euclidean group of transformation $\mathrm{SE}(3)$. An element $\mathbf{R} \in \mathrm{SO}(3) \subset \mathbb{R}^3$ is a matrix denoting the group of 3D rotation. An element of $\mathrm{SE}(3)$ is usually denoted by the homogeneous matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3) \Big| \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3 \tag{1.1}$$

where $\mathbf{t} \in \mathbb{R}^3$. The tangent space of $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$, denoted by $\mathrm{so}(3)$ and $\mathrm{se}(3)$ respectively, is obtained using the logarithmic map:

$$\boldsymbol{\omega} = \frac{\theta}{2\sin\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{1.2}$$

where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1.3}$$

and

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) \tag{1.4}$$

such that $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) \in \mathrm{so}(3)$ and the corresponding skew-symmetric matrix is given by

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{1.5}$$

An element $\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\nu}^\top & \boldsymbol{\omega}^\top \end{pmatrix} \in \mathrm{se}(3)$ can be derived by:

$$\boldsymbol{\nu} = \left(\mathbf{I}_{3\times 3} + \left(\frac{1-\cos\theta}{\theta^2}\right)[\boldsymbol{\omega}]_\times + \left(\frac{\theta - \sin\theta}{\theta^3}\right)[\boldsymbol{\omega}]_\times^2\right)^{-1}\mathbf{t} \tag{1.6}$$

The setup for visual tracking (in the context of this thesis) would always contain at-least three cartesian reference frames, one centered at the camera (or the visual sensor) $\mathcal{F}_\mathrm{c}$, another centered at the object being tracked $\mathcal{F}_\mathrm{o}$ and the last one depic-

ting the world reference frame $\mathcal{F}_{\mathrm{w}}$. The coordinate frames belong to the 3D Euclidean space $\mathbb{E}^3$. An arbitrary 3D point $\mathbf{P}$ in real coordinate space $\mathbb{R}^3$ is represented in homogeneous coordinates as a 4-vector $\overline{\mathbf{P}} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1)$, while the point itself is represented as $\mathbf{P} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z}) \in \mathbb{R}^3$ w.r.t $\mathcal{F}_{\mathrm{c}}$. The same point, when expressed w.r.t the world or object coordinate, gets represented as ${}^{\mathrm{w}}\mathbf{P}$ and ${}^{\mathrm{o}}\mathbf{P}$ respectively. Using standard convention, a transformation from an arbitrary reference frame A to another arbitrary reference frame B is denoted by ${}^{\mathrm{B}}\mathbf{T}_{\mathrm{A}}$. With this representation, a 3D point can be transformed from the frame A to B using:

$$ {}^{\mathrm{B}}\mathbf{P} = {}^{\mathrm{B}}\mathbf{T}_{\mathrm{A}}\,{}^{\mathrm{A}}\mathbf{P} \tag{1.7} $$

If $\mathbf{T}$ is represented as $\mathbf{q} = (\mathbf{t}, \theta\mathbf{u})$, where $\mathbf{u}$ is the axis of the rotation $\mathbf{R}$ and θ its relative angle, the time derivative $\dot{\mathbf{q}}$ can be used to represent the same expression as a velocity twist $\boldsymbol{\xi} = \dot{\mathbf{q}}$, provided θ and $\mathbf{t}$ are very small.

1.1.2 Perspective Projection Model

When a 3D point $\mathbf{P}$ is imaged using the *pinhole camera model* of the *general projective camera* [Hartley and Zisserman, 2003], a 2D image point $\mathbf{p} = (u, v)$, with the corresponding homogeneous coordinate $\overline{\mathbf{p}} = (u, v, 1)$, is formed on the image plane $\mathcal{I}$, where

$$ \begin{aligned} u &= f_x \frac{\mathrm{X}}{\mathrm{Z}} + c_x \\ v &= f_y \frac{\mathrm{Y}}{\mathrm{Z}} + c_y \end{aligned} \tag{1.8} $$

(f_x, f_y) are the focal lengths of the camera, expressed in pixel units, whereas (c_x, c_y) denotes the 2D coordinates of the *principal point* on the image plane.

For a point $\mathbf{p}$, the projective transformation $\mathbf{K}_p$ can be expressed as the mapping:

$$ \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f_x\mathrm{X} + c_x\mathrm{Z} \\ f_y\mathrm{Y} + c_y\mathrm{Z} \\ \mathrm{Z} \end{bmatrix} = \overline{\mathbf{p}} = \mathbf{K}_p \boldsymbol{\Pi}_p \overline{\mathbf{P}} \tag{1.9} $$

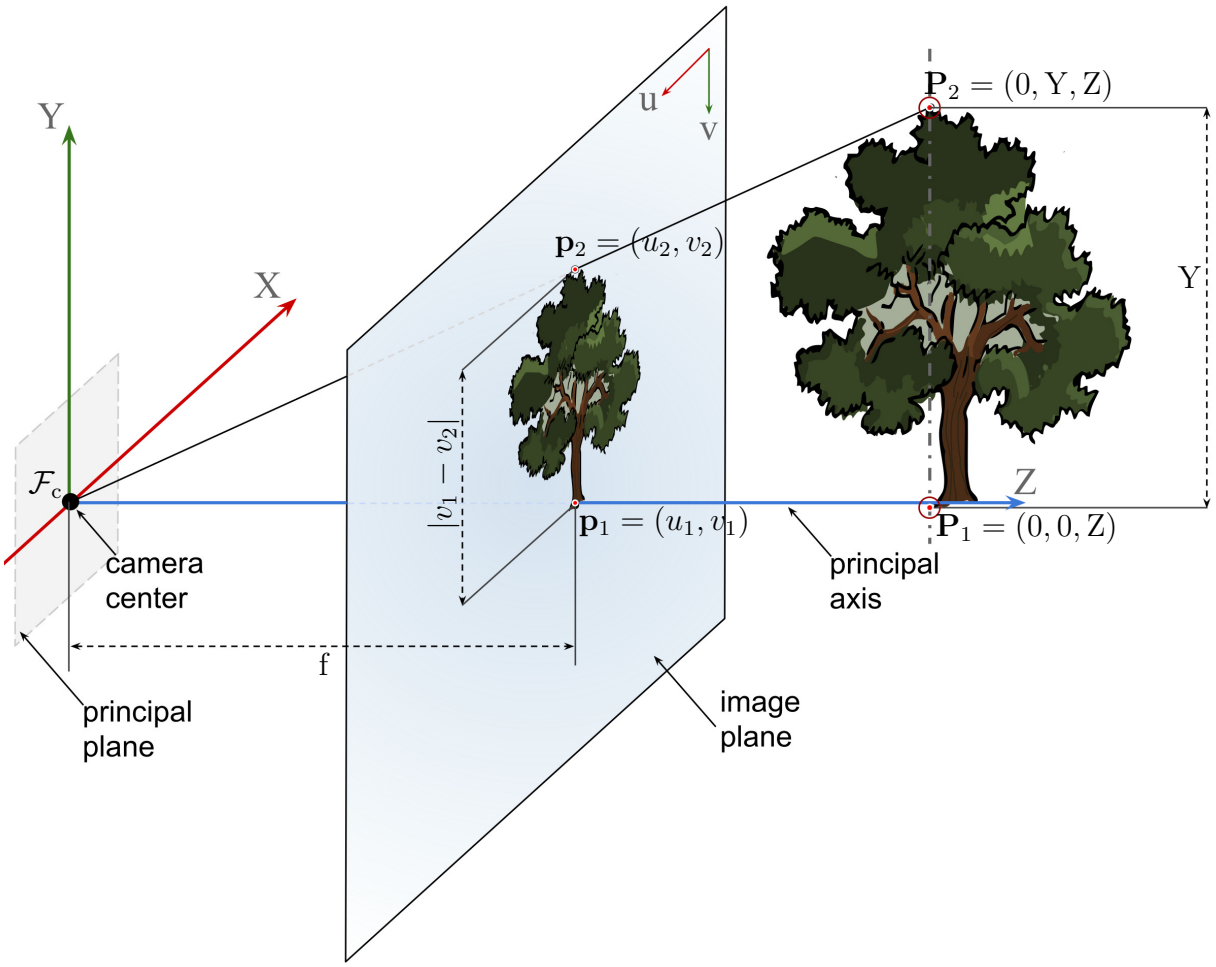


FIGURE 1.1 – Pinhole camera projection model. We represent image formation for two 3D points $\mathbf{P}_1$ and $\mathbf{P}_2$ that are co-planar and lies parallel to the image plane

$$\Rightarrow \overline{\mathbf{p}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_p} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{\Pi}_p} \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix} \tag{1.10}$$

The matrix $\mathbf{K}_p$ is called the *camera calibration matrix* and denotes the intrinsic parameters for a CCD or CMOS type sensor. Note that generalized projection could be

modelled with an additional skew parameter s_p, such that:

$$\mathbf{K}_p = \begin{bmatrix} f_x & s_p & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.11}$$

s_p is an axial skew parameter which denotes the shear distortion in the projected image. However, it can be assumed that $s_p = 0$ for most normal cameras. Moreover, to generalize the projection even further, if $\overline{\mathbf{p}}$ is not maintained at the camera-centered reference frame, an additional transformation can be utilized to orient the point ${}^{\mathrm{o}}\overline{\mathbf{P}}$ from its object-centered reference frame to that of the camera, such that:

$$\overline{\mathbf{p}} = \underbrace{\mathbf{K}_p \mathbf{\Pi}_p {}^{\mathrm{c}}\mathbf{T}_{\mathrm{o}}}_{\text{camera projection matrix}} {}^{\mathrm{o}}\overline{\mathbf{P}} \tag{1.12}$$

The $\mathbb{R}^3 \mapsto \mathbb{R}^2$ operation corresponding to the perspective projection of 3D points to 2D pixel on the image plane can be simplified by the projection operator $\mathbf{\Pi}\big(\ \cdot\ \big)$ such that $\mathbf{p} = \mathbf{\Pi}\big(\mathbf{P}\big)$ can be expressed using the non-homogeneous coordinates of the 3D and 2D points.

$\mathcal{I}(u, v)$ gives the grayscale intensity of the pixel at coordinates (u, v) of $\mathcal{I}$. The pinhole camera model used for this projection operation is shown in Fig. 1.1. This perspective projection ideally allows us to represent light rays by straight lines, but with most cameras in real-life, certain radial distortions are observed in the pixel coordinates. These radial distortions can be modelled easily [Faugeras, 1993] [Hartley and Zisserman, 2003], and is represented as:

$$\begin{aligned} u &= u_d(1 + k_1 s^2 + k_2 s^4) \\ v &= v_d(1 + k_1 s^2 + k_2 s^4) \end{aligned} \tag{1.13}$$

where (u, v) is a point in the image plane which can be obtained using perspective projection only while (u_d, v_d) is the corresponding point with distortion. Here, $s^2 = u_d^2 + v_d^2$ and k_1 and k_2 are the parameters of distortion coefficient, which can be estimated using classical calibration techniques, such as [Brown, 1971] [Stein, 1997].

1.1.3 Object Model and Pointcloud

The 3D model depicting the surface of an object is expressed using the tuple $\mathscr{M} = \left({}^{\mathrm{A}}\mathcal{V}, \mathbf{X}^{\mathcal{V}}\right)$, where:

$$ {}^{\mathrm{A}}\overline{\mathcal{V}} = \begin{bmatrix} \overline{\mathbf{P}}_1 & \overline{\mathbf{P}}_2 & \cdots & \overline{\mathbf{P}}_{\mathrm{N}} \end{bmatrix} = \begin{bmatrix} \mathrm{P}_{1_{\mathrm{X}}} & \mathrm{P}_{2_{\mathrm{X}}} & \cdots & \mathrm{P}_{\mathrm{N}_{\mathrm{X}}} \\ \mathrm{P}_{1_{\mathrm{Y}}} & \mathrm{P}_{2_{\mathrm{Y}}} & \cdots & \mathrm{P}_{\mathrm{N}_{\mathrm{Y}}} \\ \mathrm{P}_{1_{\mathrm{Z}}} & \mathrm{P}_{2_{\mathrm{Z}}} & \cdots & \mathrm{P}_{\mathrm{N}_{\mathrm{Z}}} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{1.14} $$

and ${}^{\mathrm{A}}\mathcal{V} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_{\mathrm{N}} \end{bmatrix}$ is a set of 3D points of a model with N vertices (Fig. 1.2a), maintained at an arbitrary reference frame A.

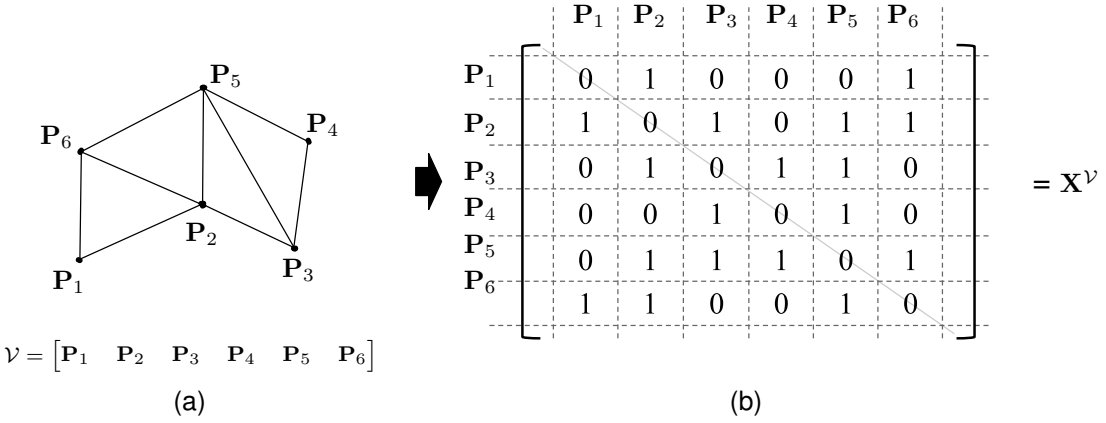
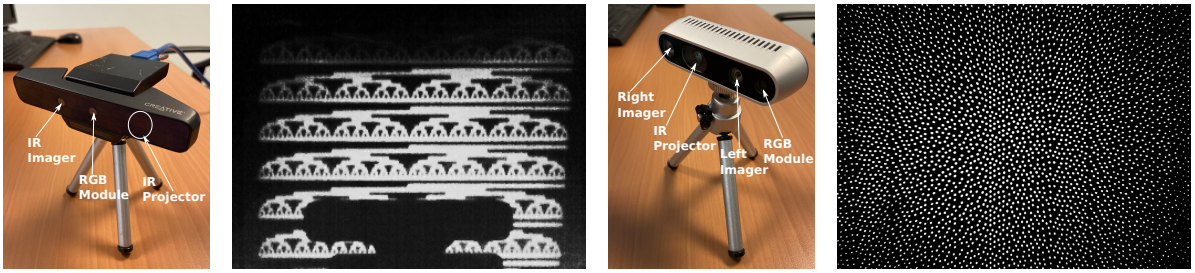


FIGURE 1.2 – Example of the mesh model setup used in this thesis, showing a mesh consisting of six vertices

A pointcloud comprising of multiple 3D points is denoted by $\boldsymbol{\psi}$, which has the same matrix structure as ${}^{\mathrm{A}}\mathcal{V}$, i.e., $\boldsymbol{\psi} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_{\mathrm{K}} \end{bmatrix}$ and ${}^{\mathrm{A}}\overline{\boldsymbol{\psi}} = \begin{bmatrix} \overline{\mathbf{P}}_1 & \overline{\mathbf{P}}_2 & \cdots & \overline{\mathbf{P}}_{\mathrm{K}} \end{bmatrix}$, where K denotes the number of points in the pointcloud.Note that $\mathcal{V}$ and $\boldsymbol{\psi}$ denotes similar matrices that represent a set of 3D points. However, throughout this thesis, $\mathcal{V}$ has been used to denote the vertices of a mesh whereas $\boldsymbol{\psi}$ has been used to denote pointclouds. $\mathbf{X}^{\mathcal{V}}$ denotes the adjacency matrix of the simple, undirected mesh such that:

$$ \mathbf{X}^{\mathcal{V}}_{ij} = \begin{cases} 1 \text{ if } (\mathbf{P}_i, \mathbf{P}_j) \text{ is connected} \\ 0 \text{ else} \end{cases} \tag{1.15} $$

thereby depicting a model with N 3D points and M number of $\mathrm{Q}-$ hedral surfaces (denoting a polygon with Q edges), as shown in Fig. 1.2b.



(a) SR300 (b) Snapshot of SL from SR300 (c) D435 (d) Snapshot of SL from D435

FIGURE 1.3 – The two depth sensors from Intel: RealSense SR300 and RealSense D435. (b) shows the color-coded Structured Light (SL) pattern emitted by the IR projector of SR300, as viewed when projected on a flat surface. (d) shows the Structured Light for D435, used to improve the quality of stereo based active depth sensing by the sensor

This enables us to transform the model to another arbitrary reference frame B by:

$$^{\mathrm{B}}\overline{\mathcal{V}} = {}^{\mathrm{B}}\mathbf{T}_{\mathrm{A}}{}^{\mathrm{A}}\overline{\mathcal{V}} \tag{1.16}$$

while $\mathbf{X}^{\mathcal{V}}$ remains independent of the inertial reference frame.

1.1.4 Sensor Setup

While discussing the representation of pointcloud, it is worthwhile to briefly discuss the depth sensor setup that has been used in this thesis to obtain the pointcloud data. The non-rigid tracking approaches proposed in this thesis have been developed using RGB-D sensors only. The capability to extract depth information along with color (RGB) information from the scene can be availed by using multiple sensor setups, such as a stereo rig, time-of-flight (ToF) sensors or structured-light based depth sensor. In this thesis, two different depth cameras are used for the experiments conducted in this thesis: Intel RealSense SR300 and Intel RealSense D435, as shown in Fig. 1.3. SR300 is a coded-light based depth camera [Zabatani et al., 2019], wherein a special temporal optical code is projected by an infrared (IR) projector and the triangulation between projected patterns and images, captured by a dedicated IR sensor, is used to generate a dense depth map. The D435 uses an *Active Stereo Depth* system [Grunnet-Jepsen et al.,], by projecting a pattern to assist the stereoscopic depth estimation using the right and left IR sensor. The color camera of SR300 is rolling shutter, while D435 is equipped with a global shutter color camera.

1.2 Rigid Object Tracking

While tracking rigid objects using visual information, there are multiple approaches that can be utilized for frame-to-frame registration. We first describe the classical approaches for rigid object tracking and then take a look at some of the recent methods from the state-of-the-art for rigid object and scene tracking.

1.2.1 Classical Approaches

Since we utilize a depth camera, we can choose to track objects in terms of *3D-3D* registration only, we can restrict ourselves to just the image data and track objects in a *2D-2D* sense, we can switch back and forth between *2D-3D* registration or we can use the *hybrid* data consisting of depth and image information together in RGB-D format. We shall present the basic principles related to the different approaches for registration of rigid objects:

1.2.1.1 3D-3D Registration

The first approach for rigid object tracking involves the registration of a pair of 3D data obtained from some suitable depth sensor. Given a stream of 3D points from the depth sensor, one of the simplest approach to align the points is using an Iterative Closest Point (ICP) algorithm which was first proposed as a method for registration of 3D shapes in [Besl and McKay, 1992]. It was later summarized by [Pomerleau et al., 2015], outlining more than two decades of refinement since it was first proposed.

In its basic form, ICP has two variants, one involving the registration of 3D points with another set of 3D points (*point-to-point* ICP), while the other variant involves the minimization of distance between a set of 3D points with planar surfaces (*point-to-plane* ICP). The *point-to-point* ICP can be expressed as the minimization of the following cost function w.r.t the transformation between two pointclouds:

$$\mathrm{E}(\boldsymbol{\psi}_{\mathcal{P}}, \boldsymbol{\psi}_{\mathcal{S}}) = \sum_{(\mathbf{p},\mathbf{s})\in\mathbf{N}_{\boldsymbol{\psi}}} \|\mathbf{p} - \mathbf{s}\|_2 \tag{1.17}$$

where $\boldsymbol{\psi}_{\mathcal{P}}$ and $\boldsymbol{\psi}_{\mathcal{S}}$ are two pointclouds located close to each other, and the set $\mathbf{N}_{\boldsymbol{\psi}}$

denotes the association between the pointclouds, defined as:

$$\mathbf{N}_{\psi} = \text{match}(\boldsymbol{\psi}_{\mathcal{P}}, \boldsymbol{\psi}_{\mathcal{S}}) = \left\{ (\mathbf{p}_n, \mathbf{s}_m) : \forall \mathbf{p}_n \in \boldsymbol{\psi}_{\mathcal{P}}, \mathbf{s}_m = \arg\min_j \left(\text{d}(\mathbf{p}_n, \mathbf{s}_j) \right) \right\} \tag{1.18}$$

where $\text{d}(\mathbf{p}, \mathbf{s}) = \|\mathbf{p} - \mathbf{s}\|_2$. This matching operation is repeated iteratively. For this form of problem, there exists a closed form solution for determining the transformation between $\boldsymbol{\psi}_{\mathcal{P}}$ and $\boldsymbol{\psi}_{\mathcal{S}}$ that minimizes the error in Eqn. 1.17. Solutions have been proposed based on singular value decomposition (SVD) [Arun et al., 1987], quaternions [Horn, 1987], orthonormal matrices [Horn et al., 1988] and dual quaternions [Walker et al., 1991] (however, it must be noted that [Horn, 1987] and [Walker et al., 1991] does not utilize exactly the same error function as given in Eqn. 1.17, interested readers may refer to [Eggert et al., 1997]). Despite the existence of closed form solution, it must the noted that Eqn. 1.17 is differentiable and therefore it can be fitted into a *steepest gradient descent* or Gauss-Newton optimization scheme to arrive at the optimal transformation [Solomon, 2015].

The *point-to-plane* ICP is the registration between a 3D pointcloud and a surface model, and it is expressed as the minimization:

$$\text{E}(\boldsymbol{\psi}_{\mathcal{P}}, \mathcal{M}_{\mathcal{S}}) = \sum_{k=1}^{\text{K}} \|(\mathbf{p}_k - \mathbf{s}_l) \cdot \mathbf{n}_l\|_2 \tag{1.19}$$

where $\mathbf{n}_l$ is the surface normal of the model around the point $\mathbf{s}_l \in \mathcal{V}_{\mathcal{S}}$ and the model itself is represented as $\mathcal{M}_{\mathcal{S}} = (\mathcal{V}_{\mathcal{S}}, \mathbf{X}^{\mathcal{V}_{\mathcal{S}}})$ and has l vertices. K represents the total number of points in the pointcloud $\boldsymbol{\psi}_{\mathcal{P}}$ and the association between $\mathbf{p}_k$ and $\mathbf{s}_l$ is done using an external algorithm for *point-to-plane* correspondence. Interested readers may refer to [Ramalingam and Taguchi, 2013] for an overview of correspondence methods. [Pomerleau et al., 2015] provides a recap of a classical closed form solution to Eqn. 1.19, given in the form of:

$$\underbrace{\mathbf{G}\mathbf{G}^{\top}}_{\mathbf{A}} \underbrace{\boldsymbol{\xi}_p^{\top}}_{\mathbf{x}} = \underbrace{\mathbf{G}\mathbf{h}}_{\mathbf{b}} \tag{1.20}$$

where $\boldsymbol{\xi}_p = \begin{pmatrix} \boldsymbol{\omega} & \boldsymbol{\nu} \end{pmatrix}^{\top}$, the transformation aligning $\boldsymbol{\psi}_{\mathcal{P}}$ with $\mathcal{M}_{\mathcal{S}}$, can be obtained by solving Eqn. 1.20 for $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ using Cholesky decomposition [Cholesky, 2005] of the

matrix denoted by $\mathbf{A}$. Here:

$$\mathbf{G} = \underbrace{\begin{bmatrix} \cdots & \mathbf{p}_k \times \mathbf{n}_k & \cdots \\ & \mathbf{n}_k & \end{bmatrix}}_{[6\times \mathrm{K}]} \tag{1.21}$$

and:

$$\mathbf{h} = \underbrace{\begin{bmatrix} \vdots \\ (\mathbf{s}_k - \mathbf{p}_k) \cdot \mathbf{n}_k \\ \vdots \end{bmatrix}}_{[\mathrm{K}\times 1]} \tag{1.22}$$

However, it is also possible to reach at a similar solution with non-linear least squares optimization, where the Jacobian relating the variation of $\boldsymbol{\xi}_p$ with the change of $\mathrm{E}(\boldsymbol{\psi}_{\mathcal{P}}, \mathcal{M}_{\mathcal{S}})$ is given by:

$$\mathbf{J}_{\mathrm{icp}} = \frac{\partial \mathrm{E}(\boldsymbol{\psi}_{\mathcal{P}}, \mathcal{M}_{\mathcal{S}})}{\partial \boldsymbol{\xi}_p} = \begin{bmatrix} \mathbf{p}_k \times \mathbf{n}_k & \mathbf{n}_k \end{bmatrix} \tag{1.23}$$

while the update $\Delta\boldsymbol{\xi}_p$ can be computed in the Gauss-Newton sense:

$$\Delta\boldsymbol{\xi}_p = -(\mathbf{J}_{\mathrm{icp}}^{\top}\mathbf{J}_{\mathrm{icp}})^{-1}\mathbf{J}_{\mathrm{icp}}^{\top}\mathrm{E}(\boldsymbol{\psi}_{\mathcal{P}}, \mathcal{M}_{\mathcal{S}}) \tag{1.24}$$

The update is combined with the initial estimate by $(\boldsymbol{\xi}_p)_t = (\boldsymbol{\xi}_p)_{t-1} + \Delta\boldsymbol{\xi}_p$ where $(\boldsymbol{\xi}_p)_t$ is the estimate of $\boldsymbol{\xi}_p$ at t-th iteration of the minimization. This iterative gradient estimation and update of $\boldsymbol{\xi}_p$ is continued till a certain convergence criterion is reached. The exact convergence criterion is often determined empirically and depends on the application area. Interested readers are directed to [Salzo and Villa, 2012] for a detailed analysis of convergence properties and criteria for Gauss-Newton method.

1.2.1.2 Dense 2D-2D Registration

Having summarized some of the traditional methods for 3D-3D registration, we now take a look into some of the 2D-2D registration techniques based on just the image data. This is a more classical problem statement, since monocular cameras were popularized much earlier than affordable depth sensors. Tracking with images can be

usually expressed as the minimization of the sum of squared error term:

$$\mathrm{E}(\mathbf{p}) = \arg\min_{\mathbf{x}} \sum_{\mathbf{p}} \Big[\boldsymbol{\mathcal{I}}(\mathbf{W}(\mathbf{p},\mathbf{x})) - \boldsymbol{\mathcal{I}}^{*}(\mathbf{p})\Big]^{2} \tag{1.25}$$

With classical image based tracking using the Lucas-Kanade algorithm [Baker and Matthews, 2004], $\mathbf{p}$ is usually the image pixel coordinates, where $\boldsymbol{\mathcal{I}}^{*}(\cdot)$ is the target/template image and $\boldsymbol{\mathcal{I}}$ is the current image, $\mathbf{W}$ is an image warping function based on the warp parameters $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, a vector of parameters defining the warp. Minimizing the expression in Eq. 1.25 is a non-linear optimization task, even if $\mathbf{W}$ is linear in $\mathbf{x}$. To optimize Eq. 1.25, the current estimate of $\mathbf{x}$ is assumed to be known. The increment $\Delta\mathbf{x}$ is iteratively solved by minimizing

$$\mathrm{E}(\mathbf{p}) = \sum_{\mathbf{p}} \Big[\boldsymbol{\mathcal{I}}\Big(\mathrm{W}(\mathbf{p},\mathbf{x}+\Delta\mathbf{x})\Big) - \boldsymbol{\mathcal{I}}^{*}\Big(\mathbf{p}\Big)\Big]^{2} \tag{1.26}$$

and the parameters are updated by

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x} \tag{1.27}$$

The Lucas-Kanade algorithm is specifically a *Gauss-Newton* based gradient descent algorithm. However, *steepest gradient descent*, *Newton-Raphson* [Acton, 1990] or *Levenberg-Marquadt* [Marquardt, 1963] [Solomon, 2015] based minimization can be equally applicable for a minimization of this nature. The expression for intensity difference can be linearized using the Taylor expansion:

$$\boldsymbol{\mathcal{I}}(\mathrm{W}(\mathbf{p},\mathbf{x}+\Delta\mathbf{x})) - \boldsymbol{\mathcal{I}}^{*}(\mathbf{p}) = \boldsymbol{\mathcal{I}}(\mathrm{W}(\mathbf{p},\mathbf{x})) + \nabla\boldsymbol{\mathcal{I}}\frac{\partial\mathbf{W}}{\partial\mathbf{x}}\Delta\mathbf{x} - \boldsymbol{\mathcal{I}}^{*}(\mathbf{p}) \tag{1.28}$$

where $\nabla\boldsymbol{\mathcal{I}} = \begin{pmatrix}\frac{\partial\boldsymbol{\mathcal{I}}}{\partial u} & \frac{\partial\boldsymbol{\mathcal{I}}}{\partial v}\end{pmatrix}$, the image gradients along x and y axis of the image plane. The term $\frac{\partial\mathbf{W}}{\partial\mathbf{x}}$ is the *Jacobian* of the warping function and is given by:

$$\frac{\partial\mathbf{W}}{\partial\mathbf{x}} = \begin{bmatrix}\frac{\partial\mathbf{W}_x}{\partial x_1} & \frac{\partial\mathbf{W}_x}{\partial x_2} & \cdots & \frac{\partial\mathbf{W}_x}{\partial x_n} \\ \frac{\partial\mathbf{W}_y}{\partial x_1} & \frac{\partial\mathbf{W}_y}{\partial x_2} & \cdots & \frac{\partial\mathbf{W}_y}{\partial x_n}\end{bmatrix} \tag{1.29}$$

The formulation of eqn. 1.26 is termed the *forward-additive* formulation for direct image intensity based visual tracking. *Forward-compositional* formulation [Shum and Sze-

liski, 2001], *inverse-additive* formulation [Hager and Belhumeur, 1998] and *inverse-compositional* formulation [Baker and Matthews, 2001] are used as well. The comparative outline of these methods are summarized below:

1. **Forward-additive method** The error function is given as by Eqn. 1.26, while the update is given by Eqn. 1.2.1.2.

2. **Forward-compositional method** The error function is given by:

$$\mathrm{E}(\mathbf{p}) = \sum_{\mathbf{p}} \Big[\boldsymbol{\mathcal{I}}\Big(\mathrm{W}\big(\mathrm{W}(\mathbf{p}, \Delta\mathbf{x})\big), \mathbf{x}\Big) - \boldsymbol{\mathcal{I}}^*\Big(\mathbf{p}\Big)\Big]^2 \tag{1.30}$$

   and the update is given by:

$$\mathrm{W}(\mathbf{p}, \mathbf{x}) \leftarrow \mathrm{W}(\mathbf{p}, \mathbf{x}) \circ \mathrm{W}(\mathbf{p}, \Delta\mathbf{x}) \tag{1.31}$$

   where the $\circ$ operator denotes the composition of two warps, such that:

$$\mathrm{W}(\mathbf{p}, \mathbf{x}) \circ \mathrm{W}(\mathbf{p}, \Delta\mathbf{x}) \equiv \mathrm{W}\Big(\mathrm{W}(\mathbf{p}, \Delta\mathbf{x}), \mathbf{x}\Big) \tag{1.32}$$

3. **Inverse-compositional method** The error function is given by:

$$\mathrm{E}(\mathbf{p}) = \sum_{\mathbf{p}} \Big[\boldsymbol{\mathcal{I}}^*\Big(\mathrm{W}(\mathbf{p}, \Delta\mathbf{x})\Big) - \boldsymbol{\mathcal{I}}\Big(\mathrm{W}(\mathbf{p}, \mathbf{x})\Big)\Big]^2 \tag{1.33}$$

   while the update is given by:

$$\mathrm{W}(\mathbf{p}, \mathbf{x}) \leftarrow \mathrm{W}(\mathbf{p}, \mathbf{x}) \circ \mathrm{W}(\mathbf{p}, \Delta\mathbf{x})^{-1} \tag{1.34}$$

4. **Inverse-additive method** The error function is the same as Eqn. 1.26, while the role of the template and the image is switched. This reformulates the Taylor expansion of Eqn. 1.28 into:

$$\mathrm{E}(\mathbf{p}) = \sum_{\mathbf{p}} \Big[\boldsymbol{\mathcal{I}}\Big(\mathrm{W}(\mathbf{p}, \mathbf{x})\Big) + \nabla\boldsymbol{\mathcal{I}}^*\Big(\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Big)^{-1} \frac{\partial \mathbf{W}}{\partial \mathbf{x}} \Delta\mathbf{x} - \boldsymbol{\mathcal{I}}^*(\mathbf{p})\Big]^2 \tag{1.35}$$

   and the parameter update is given by $\mathbf{x} \leftarrow \mathbf{x} - \Delta\mathbf{x}$

The Lucas-Kanade algorithm described above enables us to track an image based on a template. We now focus on deriving a relationship between a small number of

image points in two different images when a one-to-one correspondence between the image positions are given. Given a set of four 2D to 2D point correspondences ${}^{\mathrm{a}}\overline{\mathbf{p}_i} \leftrightarrow {}^{\mathrm{b}}\overline{\mathbf{p}_i}$, where ${}^{\mathrm{a}}\overline{\mathbf{p}_i} = ({}^{\mathrm{b}}u_i, {}^{\mathrm{a}}v_i, 1)$ and ${}^{\mathrm{b}}\overline{\mathbf{p}_i} = ({}^{\mathrm{b}}u_i, {}^{\mathrm{b}}v_i, 1)$ are two points on the image plane in homogeneous coordinates, it is possible to derive a 3×3 matrix $\mathbf{H}$ such that ${}^{\mathrm{b}}\overline{\mathbf{p}_i} = \mathbf{H}{}^{\mathrm{a}}\overline{\mathbf{p}_i}$. To enforce a scale constancy, an additional constraint can be imposed in the form of ${}^{\mathrm{b}}\overline{\mathbf{p}_i} \times (\mathbf{H}{}^{\mathrm{a}}\overline{\mathbf{p}_i}) = 0$. This leads to a set of three equations of the form:

$$\underbrace{\begin{bmatrix} \mathbf{0}^\top & -{}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top & {}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top \\ {}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top & \mathbf{0}^\top & -{}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top \\ -{}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top & {}^{\mathrm{a}}\overline{\mathbf{p}_i}^\top & \mathbf{0}^\top \end{bmatrix}}_{3\times9} \underbrace{\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix}}_{9\times1} = 0 \tag{1.36}$$

where $\mathbf{h}_n$ gives the n-th row of $\mathbf{H}$. With four point correspondences, a non-zero solution of $\mathbf{H}$ can be determined up to a non-zero scale factor [Hartley and Zisserman, 2003]. This forms the simplified 2D representation of the *Direct Linear Transform* (DLT) algorithm, and can be utilized in a 2D-2D object tracking framework. However, it can be also utilized in tandem with 3D data resulting in a more stable variant of tracking, as explained in the next paragraph.

1.2.1.3 2D-3D Registration

For a calibrated camera, i.e., when $\mathbf{K}_p$ from Eqn. 1.12 is estimated using an external calibration method, the transformation ${}^{\mathrm{c}}\mathbf{T}_{\mathrm{o}}$, that describes the relative pose between the object frame and the camera frame, can be determined using a small number of 2D-3D point correspondences and this method is more stable and reliable for object tracking applications, as compared to classical DLT based techniques. [Fischler and Bolles, 1981] coined the term Perspective-n-Point (PnP) problem, where n stands for the number of correspondences, to describe this class of algorithms and a vast amount of research has been done to come up with various solutions to this PnP problem.

The PnP problem can be solved either non-iteratively [Fischler and Bolles, 1981, Dhome et al., 1989, Gao et al., 2003, Kneip et al., 2011] or through iterative techniques [Lowe, 1991, Dementhon and Davis, 1995, Lu et al., 2000]. Interested readers may refer to [Marchand et al., 2016] for a detailed survey of 3D pose estimation techniques.

1.2.2 State-of-the-art for Rigid Object Tracking

The rigid object tracking literature is extensive and heterogeneous. An exhaustive survey of all such techniques is beyond the scope of this thesis. Therefore, we restrict the discussion to a few of the more important and relevant object tracking methodologies that were proposed in the last three decades. A brief summary of some of the notable research in the field of rigid object tracking is given below.

As mentioned in Sec. 1.2, [Besl and McKay, 1992] is one of the earliest paper to explain the ICP in details. The approach involves calculating the closest distance from a point to either a parametric entity: viz., curves or surfaces, or implicit geometric entity: like a vector-valued, multivariate function. A Newtonian minimization approach is used for the former, while an augmented Lagrange multiplier system [Luenberger et al., 1984] is used for the later. The closest point calculation and registration is done iteratively till the mean square error falls below a preset threshold. This forms the basic ICP algorithm described in [Wikipedia, 2016]. Alternate methods of point cloud registration has been proposed by [Pottmann et al., 2004] for point cloud registration without using the original ICP proposed by [Besl and McKay, 1992]. [Pottmann et al., 2004] proposes a kinematic model-based approach to 3D point cloud registration.

[Newcombe et al., 2011] and [Izadi et al., 2011] provided a major advance in real-time tracking and mapping. It allowed users to create very accurate map of the scene while tracking the camera reliably, leading to implementation in applications like 3D modelling, high quality, tracking of robots mounted with these depth cameras, etc. The technique used for *KinectFusion* is broadly divided into four steps: A) measurement, B) pose estimation, C) Reconstruction and D) Surface Prediction. The measurement of the raw data coming in from the sensor is stored as a dense vertex map and a pyramid of normal map. The reconstruction of the scene, at every step, is stored as a Truncated Signed Distance Function (TSDF). It eases the integration of new measurement into the reconstructed model. If the transformation of the camera (or Kinect) with respect to the world is known, the appearance of the depth map at the next iteration can be predicted by transforming the reconstruction using the camera's rotation and translation, and then projecting it into the image plane using ray-casting techniques. Pose estimation, on the other hand, is done using ICP.

Many subsequent papers have tried to improve upon the accuracy of *KinectFusion*. [Henry et al., 2013] is one of the primary research work dealing with improvement of tracking accuracy of *KinectFusion* by implementing global optimization on the

pose frame graph. This was done using *g2o* [Kümmerle et al., 2011]. [Roth and Vona, 2012], [Nießner et al., 2013], [Zeng et al., 2012] and [Whelan et al., 2013] improves the mapping capabilities of *KinectFusion*. [Choi and Christensen, 2013] proposed a particle filtering based approach to track the degrees of freedom of an object from RGB-D data, wherein the likelihood of each particle was evaluated using a combination of geometric and photometric information.

Extending this approach, [Ren et al., 2017] proposes to express the probability of each pixel as a joint distribution over a combined function of the camera pose, *Signed Distance Function* (SDF) [Chan and Zhu, 2005] of the object, RGB value of the pixel and a step function which determines if the object belongs to background or foreground. The pose is determined by a Levenberg - Marquardt minimization of the Negative Log-Likelihood (NLL) of this distribution. [Slavcheva et al., 2016] proposes a real-time SDF to SDF registration method using a combination of geometric and surface normal based constraints to generate a colored pointcloud which is globally registered.

Among the recent approaches, [Kehl et al., 2016] uses a convolutional auto-encoder for matching scene-patch descriptors with synthetic model patches to detect and localize all 6 DoF of an object. [Garon and Lalonde, 2017] proposes a CNN based 6DoF object tracker using Microsoft Kinect. In 2017, [Xiang et al., 2018] proposed PoseCNN, a CNN based semantic labelling and 6DoF object tracking framework. [Wu et al., 2018] proposes a realtime technique for 6DoF object localization using CNN based on RGB images only.

1.3 Non-rigid Object Tracking

We shall now present a brief overview of the existing state-of-the-art on non-rigid object tracking, as well as the summary of some of the relevant and notable methodologies used in the literature. However, we begin by defining the objective of this thesis.

Problem Statement

Given a 3D mesh model $\mathscr{M}$ (which can be a triangular or tetrahedral mesh) and a 3D point cloud ψ, we define a cost function $\mathrm{d}(\mathscr{M}, \psi)$ that provides a similarity mea-

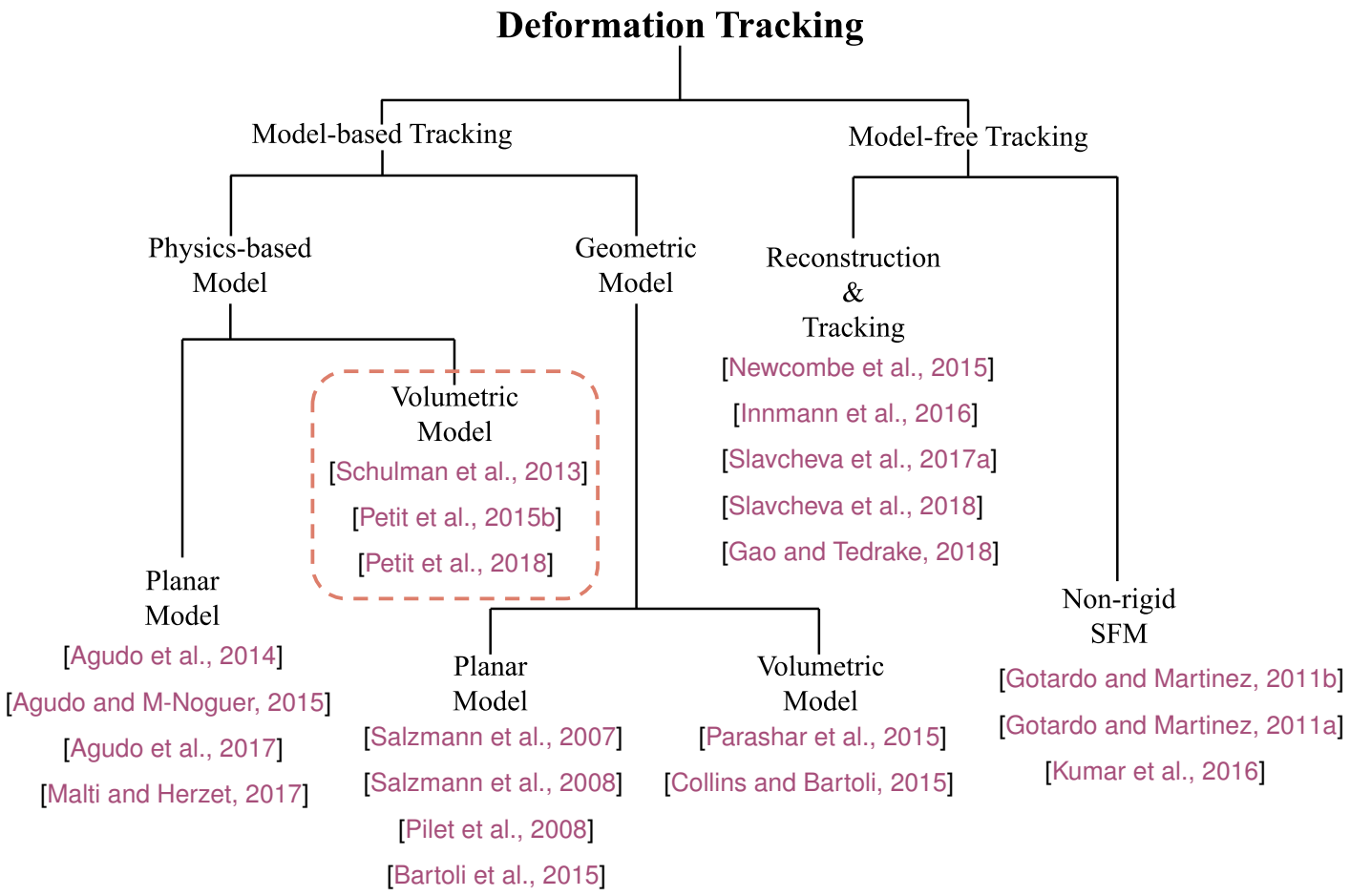


FIGURE 1.4 – Categorization of some of the selected literature about deformation tracking using visual and/or depth information. The highlighted region denotes the area of interest of this thesis

sure between the pointcloud and the model. We also define a deformational function $\mathscr{F}$ which acts on the model $\mathscr{M}$ to produce a deformed model $\mathscr{F}(\mathscr{M})$. $\mathscr{F}$ can be parameterized using a deformation or force field. The objective of the non-rigid tracking method is to minimize the cost function:

$$\mathrm{E}(\boldsymbol{\psi}) = \arg\min_{\mathscr{F}} \mathrm{d}\Big(\mathscr{F}(\mathscr{M}), \boldsymbol{\psi}\Big) \tag{1.37}$$

With this objective, tracking of deforming objects can be handled using multiple approaches, based on the application area and the practical constraints involved with the targeted application. The available literature on this subject can be broadly classified

into two categories: **a)** template/model based tracking, and **b)** model-free reconstruction and tracking of deforming objects. Even though this thesis primarily deals with model based tracking, we also present the details of the research from the relevant reconstruction and tracking literature as well.

For the sake of completeness, we discuss the state-of-the-art for both these paradigms. An overview of some of the notable approaches that we plan to discuss are given in Fig. 1.4. The explanation and the relevant discussion about these research articles follow in the subsequent paragraphs. In this thesis, we use a physically based deformation model, the co-rotational FEM, for modelling the deformation of objects. The following paragraphs are an attempt to explain the rationale behind this choice and to highlight the advantage of using physically based models for deformation tracking, over other alternative approaches.

1.3.1 Physically-based Models

Mechanical models for modelling bio-mechanical deformation of organic objects can be done using various advanced mechanical models, such as Finite Element Method (FEM), Smoothed-Particle Hydrodynamics (SPH) or porous media model [Wei et al., 2009] based fluid flow simulation. However, in this thesis, we confine our study of physics based deformable model to FEM.

FEM aims to discretize a dynamic system involving elastic materials (in the context of mechanical engineering), such that the temporal evolution of the state of the system when subjected to external, unbalanced forces can be simulated across time steps. To do so, multiple formats of mechanical models can be used, such as triangular, tetrahedral or hexahedral meshes. This mechanical model alone is sufficient to evaluate the evolution of the FEM. However, since we are interested in visual tracking, it is important to map this mechanical model with a visual model of the surface of the object being tracked.

The mapping between the vertices of the mechanical mesh $\mathcal{V}_m$ and the visual model $\mathcal{V}_s$, as shown in Fig. 1.5, can be expressed by $\mathcal{V}_s = \mathscr{J}(\mathcal{V}_m)$ (the notation for the mesh vertices is defined in Sec. 1.1.3) while the velocities are related by $\mathcal{U}_s = \mathbf{J}_{sm}\mathcal{U}_m$, where $\mathcal{U} = \dot{\mathcal{V}}$, the differentiation w.r.t time, and the Jacobian:

$$\mathbf{J}_{sm} = \frac{\partial \mathcal{V}_s}{\partial \mathcal{V}_m} \tag{1.38}$$

relates the displacement of the vertices of $\mathcal{V}_m$ to $\mathcal{V}_s$. Here, $\mathscr{J}$ represents the barycentric mapping.

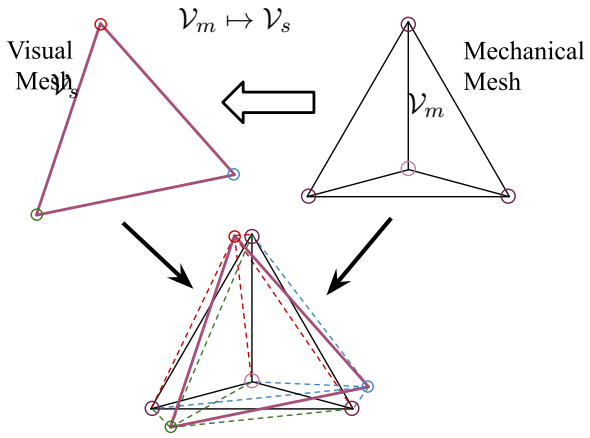


FIGURE 1.5 – The mapping between the visual and the mechanical mesh

One of the purpose of using FEM is to convert the harmonic analysis of free or damped vibrations of physical objects in continuous domain, which is usually expressed as Partial Differentials Equations (PDEs), into a discrete formulation such that the same system can be expressed by Ordinary Differential Equations (ODEs), which enables it to be solved efficiently by modern computers.

Frame 1: Example of FEM in a 1D Bar

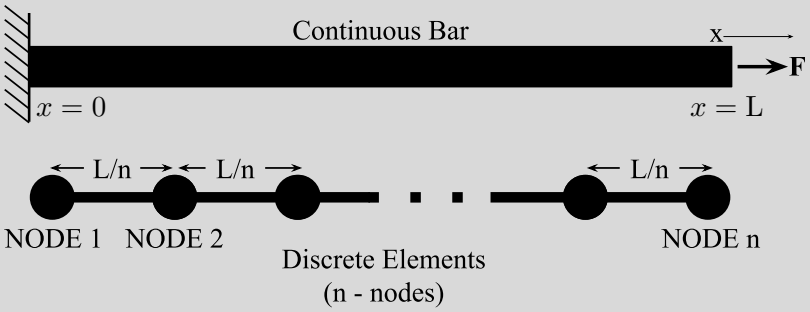


FIGURE 1.6 – A continuous bar (*top*), represented by discretizing it into n nodes (*bottom*)

A simple example of FEM for a 1-dimensional bar is summarized below. Let us assume an uniform, homogeneous bar of length L, as shown in Fig. 1.6. The *one dimensional wave equation* of this bar [Wood and Zienkiewicz, 1977], under an axial load, is given by $\frac{\partial^2 u_s}{\partial t^2} = \mathrm{V}_b^2 \frac{\partial^2 u_s}{\partial x^2}$ such that $\mathrm{V}_b = \sqrt{\frac{\mathrm{E}}{\rho}}$ and u_s is the displacement in x direction at time t.

E and ρ are the Young's modulus and density of the bar respectively, while V_b gives the wave propagation velocity. For FEM based analysis, it is necessary to represent the force-displacement relationship in a matrix representation of the format:

$$\mathbf{F} = \mathbf{K}\mathbf{u}_s \tag{1.39}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}_s$ is the vector of displacements and $\mathbf{F}$ is the external force vector. It can be shown [Rao, 1995] that for the i-th node of the bar:

$$\mathbf{K}_i = n\mathrm{C}_k \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{1.40}$$

and:

$$\mathbf{F} = \int_0^l \mathrm{F}_b(x,t)\boldsymbol{\phi}\mathrm{dx} \tag{1.41}$$

Here, A is the cross-sectional area of the bar, $\mathrm{C}_k = \mathrm{EA}$, l is the length of the current element, $\mathrm{F}_b(x,t)$ is a time-varying distributed load and $\boldsymbol{\phi}$ is the *shape function* [Logan, 2011]. Representing the stiffness matrix for individual elements as:

$$\mathbf{K}_i = \begin{bmatrix} \mathrm{K}_i^{11} & \mathrm{K}_i^{12} \\ \mathrm{K}_i^{21} & \mathrm{K}_i^{22} \end{bmatrix} \tag{1.42}$$

we can *assemble* the global stiffness matrix as:

$$\underset{n\times n}{\mathbf{K}} = \begin{bmatrix} \mathrm{K}_1^{11} & \mathrm{K}_1^{12} & & & \\ \mathrm{K}_1^{21} & \mathrm{K}_1^{22}+\mathrm{K}_2^{11} & \mathrm{K}_2^{12} & & \\ & \mathrm{K}_2^{21} & \mathrm{K}_2^{22}+\mathrm{K}_3^{11} & & \\ & & & \ddots & \\ & & & & \mathrm{K}_n^{22} \end{bmatrix} \tag{1.43}$$

Since the left end of the bar in Fig. 1.6 is fixed, the row and column corresponding to node 0 will be set to zero, thereby enforcing the *boundary condition*. If the forces are known, the nodal displacement vector can be obtained as:

$$\mathbf{u}_s = \mathbf{K}^{-1}\mathbf{F} \tag{1.44}$$

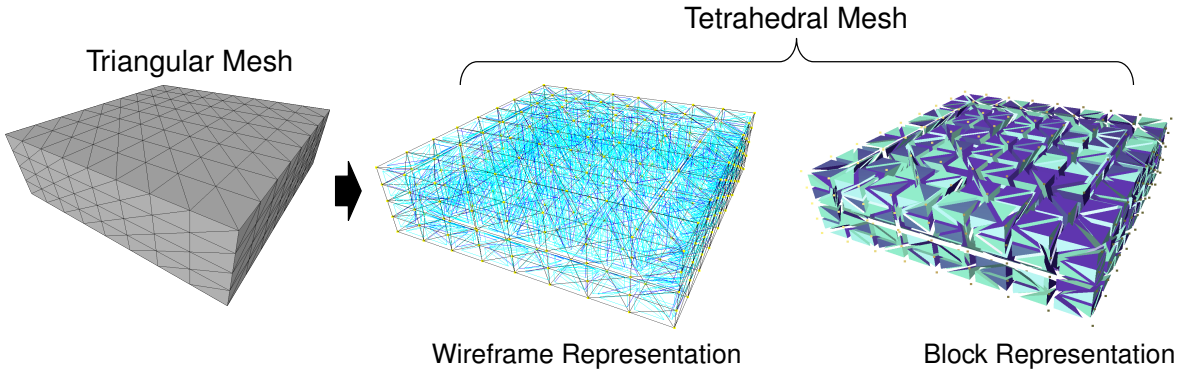


FIGURE 1.7 – The triangular, surface model coupled with a tetrahedral, volumetric model

This scheme for relating force with displacement can be generalized to arbitrarily oriented beam elements and 3D triangular, tetrahedral or hexahedral elements. Only tetrahedral elements are used to model the mechanics for the FEM used in this thesis. Tetrahedral meshes have the advantage of producing low dispersion error [Wu and Lee, 1997] when used in the context of FEM. This encourages us to use tetrahedral mesh for the mechanical model (Fig. 1.7), while using triangular mesh for the surface model.

Frame 2: Example of FEM in a 3D Tetrahedral Element

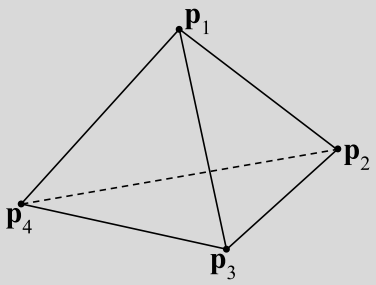


FIGURE 1.8 – A tetrahedral element of a FEM with four vertices

An example of FEM formulation for a single tetrahedral element and its assembly into a global mechanical mesh is discussed below. A typical tetrahedral element of FEM comprises of four vertices $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$ as shown in Fig. 1.8. Each point is represented by its 3D coordinates, e.g: $\mathbf{p}_1 = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^\top$ etc. Let force vectors $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{F}_3$ and $\mathbf{F}_4$ be applied at the four vertices of this tetrahedron, causing a deformation of

$\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ and $\mathbf{u}_4$ respectively. The relation between the forces applied and the displacement caused is governed by the equation of motion derived from Newton's second law, given by:

$$\underset{[12\times12]}{\mathbf{M}}\ddot{\mathbf{u}} + \underset{[12\times12]}{\mathbf{D}}\dot{\mathbf{u}} + \underset{[12\times12][12\times1]}{\mathbf{K}}\mathbf{u} = \underset{[12\times1]}{\mathbf{F}} = \left(\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \mathbf{F}_4\right) \tag{1.45}$$

Here, $\mathbf{M}$ is the mass matrix, $\mathbf{D}$ is the damping matrix and $\mathbf{K}$ is the stiffness matrix. $\mathbf{M}$ can be approximated by a diagonal matrix from the total mass of the system under the assumption of uniform mass distribution. There are multiple alternatives for constructing $\mathbf{D}$, the simplest being the Rayleigh damping formulation given by:

$$\mathbf{D} = \alpha_{\mathrm{e}}\mathbf{M} + \beta_{\mathrm{e}}\mathbf{K} \tag{1.46}$$

where α_{e} and β_{e} are proportionality constants (usually determined by modal analysis of the structure). The stiffness matrix is given by:

$$\mathbf{K} = \underbrace{\mathrm{V}_{\mathrm{K}}}_{\text{scalar}}\underset{[12\times6]}{\mathbf{B}_{\mathrm{K}}^{\top}}\underset{[6\times6]}{\mathscr{D}}\underset{[6\times12]}{\mathbf{B}_{\mathrm{K}}} \tag{1.47}$$

Where V_{K} is the volume of the tetrahedral element, $\mathbf{B}_{\mathrm{K}}$ is the strain matrix and $\mathscr{D}$ is the material matrix. For the rest of this thesis, we assume that all objects used are isotropic. For isotropic materials, $\mathscr{D}$ can be expressed as:

$$\mathscr{D} = \left(\frac{\mathrm{E}(1-\nu)}{(1+\nu)(1-2\nu)}\right)\begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{1.48}$$

where E and ν are the Young's modulus and Poisson ratio of the material respectively. Equivalent matrices can be derived for orthotropic [Li and Barbič, 2014] and anisotropic [Zhao et al., 2016] materials, if needed. The strain matrix can be expressed as:

$$\mathbf{B}_{\mathrm{K}} = \frac{1}{6\mathrm{V}_{\mathrm{K}}}\mathscr{G}(\mathbf{u}) \tag{1.49}$$

where $\mathscr{G}(\mathbf{u})$ can be expanded as:

$$\mathscr{G}(\mathbf{u}) = \begin{bmatrix} a_1 & 0 & 0 & a_2 & 0 & 0 & a_3 & 0 & 0 & a_4 & 0 & 0 \\ 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 & b_4 & 0 \\ 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 & 0 & c_4 \\ b_1 & a_1 & 0 & b_2 & a_2 & 0 & b_3 & a_3 & 0 & b_4 & a_4 & 0 \\ 0 & c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 & 0 & c_4 & b_4 \\ c_1 & 0 & a_1 & c_2 & 0 & a_2 & c_3 & 0 & a_3 & c_4 & 0 & a_4 \end{bmatrix} \tag{1.50}$$

where:

$$\begin{aligned} a_1 &= -(y_3z_4 - z_3y_4) + (y_2z_4 - y_4z_2) - (y_2z_3 - y_3z_2) \\ b_1 &= (x_3z_4 - x_4z_3) - (x_2z_4 - x_4z_2) + (x_2z_3 - x_3z_2) \\ c_1 &= -(x_3y_4 - y_3x_4) + (x_2y_4 - x_4y_2) - (x_2y_3 - x_3y_2) \\ a_2 &= (y_3z_4 - y_4z_3) - (y_1z_4 - y_4z_1) + (y_1z_3 - y_3z_1) \\ b_2 &= -(x_3z_4 - x_4z_3) + (x_1z_4 - x_4z_1) - (x_1z_3 - x_3z_1) \\ c_2 &= (x_3y_4 - x_4y_3) - (x_1y_4 - x_4y_1) + (x_1y_3 - x_3y_1) \\ a_3 &= -(y_2z_4 - y_4z_2) + (y_1z_4 - y_4z_1) - (y_1z_2 - y_2z_1) \\ b_3 &= (x_2z_4 - x_4z_2) - (x_1z_4 - x_4z_1) + (x_1z_2 - x_2z_1) \\ c_3 &= -(x_2y_4 - x_4y_2) + (x_1y_4 - x_4y_1) - (x_1y_2 - x_2y_1) \\ a_4 &= (y_2z_3 - y_3z_2) - (y_1z_3 - y_3z_1) + (y_1z_2 - y_2z_1) \\ b_4 &= -(x_2z_3 - x_3z_2) + (x_1z_3 - x_3z_1) - (x_1z_2 - x_2z_1) \\ c_4 &= (x_2y_3 - x_3y_2) - (x_1y_3 - x_3y_1) + (x_1y_2 - x_2y_1) \end{aligned} \tag{1.51}$$

The volume of the tetrahedron is available using elementary geometry, given by the expression $\mathrm{V_K} = \frac{1}{6}\left|\left(\mathbf{p}_1 - \mathbf{p}_4\right) \cdot \left((\mathbf{p}_2 - \mathbf{p}_4) \times (\mathbf{p}_3 - \mathbf{p}_4)\right)\right|$. Having summarized the FEM formulation for a single tetrahedral element, we now describe how to assemble these element-wise matrices into a global equation for the entire object.

Global Assembly

We denote the global assembled matrices with the suffix $(\cdot)_g$, thereby providing us with global stiffness matrix $\mathbf{K}_g$, global force vector $\mathbf{F}_g$ and so on. Let us consider the case of an object that has been modelled with N tetrahedral elements. Let $\mathbf{F}_d$ be the force vectors of the N elements stacked column-wise, i.e., $\underset{[12\mathrm{N}\times 1]}{\mathbf{F}_d} = \left(\underset{[12\times 1]}{\mathbf{F}^1}, \mathbf{F}^2, \ldots, \mathbf{F}^{\mathrm{N}}\right)$.

It is possible to construct an assembly matrix $\mathscr{A}$ such that:

$$\begin{aligned} \underset{[12\mathrm{N}\times 1]}{\mathbf{F}_d} &= \underset{[12\mathrm{N}\times \mathrm{N}_V]}{\mathscr{A}} \underset{[\mathrm{N}_V\times 1]}{\mathbf{F}_g} \\ \Rightarrow \underset{[\mathrm{N}_V\times 1]}{\mathbf{F}_g} &= \underset{[\mathrm{N}_V\times 12\mathrm{N}]}{\mathscr{A}^\top} \underset{[12\mathrm{N}\times 1]}{\mathbf{F}_d} \end{aligned} \tag{1.52}$$

where N_V is the number of vertices in the global mechanical model of the object. It is possible to use the assembly matrix to construct the global stiffness matrix, given by:

$$\underset{[\mathrm{N}_V\times \mathrm{N}_V]}{\mathbf{K}_g} = \underset{[\mathrm{N}_V\times 12\mathrm{N}]}{\mathscr{A}^\top} \underset{[12\mathrm{N}\times 12\mathrm{N}]}{\mathbf{K}_d} \underset{[12\mathrm{N}\times \mathrm{N}_V]}{\mathscr{A}} \tag{1.53}$$

where:

$$\mathbf{K}_d = \underbrace{\begin{bmatrix} \mathbf{K}^1 & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{K}^2 & \vdots & \\ \mathbf{0} & \dots & \ddots & \\ \vdots & & & \mathbf{K}^{\mathrm{N}} \end{bmatrix}}_{[12\mathrm{N}\times 12\mathrm{N}]} \tag{1.54}$$

Similarly, $\mathbf{u}_g$ can be constructed using the approach of Eqn. 1.52 and $\mathbf{M}_g$. $\mathbf{D}_g$ can be constructed using the approach of Eqn. 1.53.

Co-rotational FEM

We utilize the co-rotational formulation [Müller and Gross, 2004] for modelling deformation using FEM since the co-rotational model is more robust and computationally efficient. The main idea of the co-rotational method is to introduce a local coordinate system that continuously rotates and translates with the system. From the literature [Rankin, 1988, Felippa, 2000], some of the advantages of the co-rotational method can be summarized as:

— Co-rotational FEM is better suited to handle problems with large rotational motion but with small strains

— It decouples the non-linearity and possible anisotropic behavior of the material from geometric non-linearities of motion

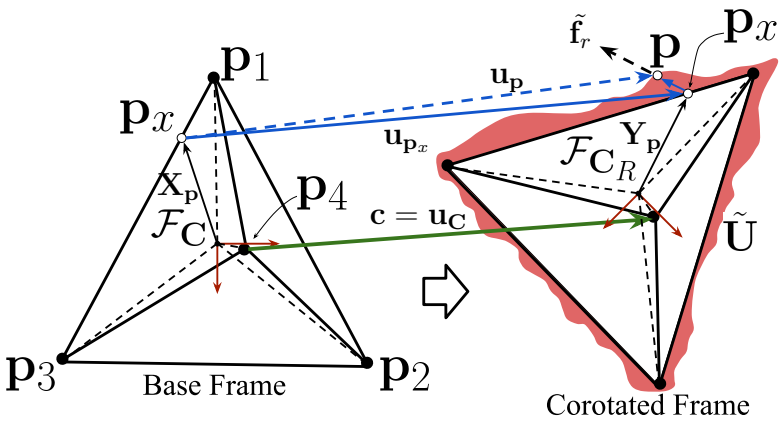


FIGURE 1.9 – Demonstration of the kinematics of a single element of a corotated, tetrahedral mesh. In the corotated frame, the red outline shows the deformed surface

We now present an introductory summary of co-rotational formulation for FEM. Let us assume that the model undergoes a rigid rotation as well as a deformation, as shown in Fig. 1.9. Let the rotated centroid at the new frame be given by $\mathcal{F}_{\mathbf{C}_R}$. An arbitrary point $\mathbf{p}_x$ in the first frame gets deformed to $\mathbf{p}$ in the next frame. The base/un-deformed frame is $\mathcal{F}_\mathbf{C}$ and the co-rotated/deformed frame is $\mathcal{F}_{\mathbf{C}_R}$ (all variables with $\tilde{\cdot}$ are expressed in the corotated frame $\mathcal{F}_{\mathbf{C}_R}$). $\mathbf{c}$ is the displacement of the centroid of this element from $\mathcal{F}_\mathbf{C}$ to $\mathcal{F}_{\mathbf{C}_R}$. $\mathbf{X}_\mathbf{p}$ and $\mathbf{Y}_\mathbf{p}$ are the position vectors of $\mathbf{p}_x$ at $\mathcal{F}_\mathbf{C}$ and $\mathcal{F}_{\mathbf{C}_R}$ respectively. Using elementary geometric transformations [Felippa, 2000], we get:

$$\tilde{\mathbf{d}}_\mathbf{p} = \mathbf{R}^\top \mathbf{d}_\mathbf{p} = \mathbf{R}^\top \big(\mathbf{u}_\mathbf{p} - \mathbf{c} + \mathbf{X}_\mathbf{p}\big) - \mathbf{X}_\mathbf{p} \tag{1.55}$$

where $\mathbf{R}$ is the rotation matrix between $\mathcal{F}_\mathbf{C}$ and $\mathcal{F}_{\mathbf{C}_R}$, $\mathbf{u}_\mathbf{p}$ is the total displacement of $\mathbf{p}_x$, expressed in the base frame $\mathcal{F}_\mathbf{C}$, and $\tilde{\mathbf{d}}_\mathbf{p}$ is the deformational displacement of $\mathbf{p}_x$, expressed in the corotated frame $\mathcal{F}_{\mathbf{C}_R}$ and $\mathbf{d}_\mathbf{p}$ is the same displacement in the base frame. Assuming a nodal force $\tilde{\mathbf{f}}_r$ acting on $\mathbf{p}_x$ and the associated deformational stiffness is $\tilde{\mathbf{K}}$, if the system is modelled to be *conservative*, we get:

$$\mathcal{U}(\tilde{\mathbf{d}}_\mathbf{p}, \lambda_r) = \mathrm{U}(\tilde{\mathbf{d}}_\mathbf{p}) - \mathrm{W}(\tilde{\mathbf{d}}_\mathbf{p}, \lambda_r) \tag{1.56}$$

where $\mathcal{U}(\tilde{\mathbf{d}}_\mathbf{p}, \lambda_r)$ is the element-level discrete potential energy, λ_r is the load parameter, while $\mathrm{U}(\tilde{\mathbf{d}}_\mathbf{p})$ and $\mathrm{W}(\tilde{\mathbf{d}}_\mathbf{p}, \lambda_r)$ are the internal strain energy and the external work function

respectively. Differentiating Eqn. 1.56 w.r.t $\tilde{\mathbf{d}}_{\mathbf{p}}$ yields:

$$\tilde{\mathbf{r}} = \frac{\partial \mathcal{U}(\tilde{\mathbf{d}}_{\mathbf{p}}, \lambda_r)}{\partial \tilde{\mathbf{d}}_{\mathbf{p}}} = \frac{\partial \mathrm{U}}{\partial \tilde{\mathbf{d}}_{\mathbf{p}}} - \frac{\partial \mathrm{W}}{\partial \tilde{\mathbf{d}}_{\mathbf{p}}} = \tilde{\mathbf{f}}_r^{\mathrm{I}} - \tilde{\mathbf{f}}_r = 0 \tag{1.57}$$

Clearly, $\tilde{\mathbf{f}}_r^{\mathrm{I}}$ is the vector of internal forces. This gives us the formulation for the residual force vector $\tilde{\mathbf{r}}$ in the co-rotated frame. This leads to the formulation:

$$\mathbf{r} = \boldsymbol{\tau}^\top \tilde{\mathbf{r}} = \mathbf{f}_r^{\mathrm{I}} - \mathbf{f}_r \tag{1.58}$$

$$\mathbf{f}_r = \boldsymbol{\tau}^\top \tilde{\mathbf{f}}_r \tag{1.59}$$

$$\mathbf{f}_r^{\mathrm{I}} = \boldsymbol{\tau}^\top \tilde{\mathbf{f}}_r^{\mathrm{I}} \tag{1.60}$$

$$\mathbf{K} = \boldsymbol{\tau}^\top \tilde{\mathbf{K}} \boldsymbol{\tau} + \tilde{\mathbf{r}} \cdot \mathbf{Q} \tag{1.61}$$

where the matrix $\boldsymbol{\tau}$ denotes the change of position of $\tilde{\mathbf{d}}_{\mathbf{p}}$ with $\tilde{\mathbf{u}}_{\mathbf{p}}$ and the variation of the i-th vertex w.r.t the j-th vertex's displacement is given by:

$$\boldsymbol{\tau}_{ij} = \frac{\partial \tilde{\mathbf{d}}_{\mathbf{p}_i}}{\partial \tilde{\mathbf{u}}_{\mathbf{p}_j}} \tag{1.62}$$

and:

$$\mathbf{Q}_{ijk} = \frac{\partial^2 \tilde{\mathbf{d}}_{\mathbf{p}_i}}{\partial \tilde{\mathbf{u}}_{\mathbf{p}_j} \partial \tilde{\mathbf{u}}_{\mathbf{p}_k}} \tag{1.63}$$

$\boldsymbol{\tau}$ and $\mathbf{Q}$ forms the transformation array for this co-rotated configuration. When $\tilde{\mathbf{r}} = 0$, we can write $\mathbf{K} = \boldsymbol{\tau}^\top \tilde{\mathbf{K}} \boldsymbol{\tau}$.

Now, applying similar motion to the nodes of the tetrahedron, we can represent the internal, elastic forces acting on the nodes by:

$$\mathbf{F}_e = \mathbf{R}_e \mathbf{K}_e \tilde{\mathbf{U}}^R \tag{1.64}$$

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

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

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}_g \mathbf{p} = \mathbf{F}_{ext} + \mathbf{F}_e \tag{1.65}$$

where $\mathbf{M}$ and $\mathbf{D}$ are the model's mass and damping matrices respectively. The elements of $\mathbf{M}$ are given as :

$$\mathbf{M}_{ij} = \int_{\Psi} \boldsymbol{\phi}_i \boldsymbol{\phi}_j \mathrm{d}V \tag{1.66}$$

where $\mathbf{M}_{ij}$ denotes the element of $\mathbf{M}$ at the i-th row and j-th column, $\boldsymbol{\phi}_i$, $\boldsymbol{\phi}_j$ is the shape function at the i-th and j-th vertex of $\mathcal{V}^{\mathrm{M}}$ respectively, V is the volume of the tetrahdron associated with i and j and Ψ is the domain of $\mathcal{V}^{\mathrm{M}}$. In this thesis, we use a Rayleigh style damping matrix $\mathbf{D}$ given by :

$$\mathbf{D} = \mathrm{r}_{\mathrm{M}}\mathbf{M} + \mathrm{r}_{\mathrm{K}}\mathbf{K}_g \tag{1.67}$$

where r_{M} and r_{K} are the Rayleigh mass and Rayleigh stiffness respectively. Non-Rayleigh style damping matrices can be used as well [Pilkey, 1998]. $\mathbf{K}_g$ is the global stiffness matrix. $\mathbf{F}_{ext}$ are the external forces acting on the vertices.

Interested readers may refer to [Hughes, 2012] for a detailed treatment of standard FEM formulation.

1.3.1.1 Tracking using Physics-based Models

Using a triangular finite element method, [Agudo et al., 2017] proposes a method to tackle deformation of planar objects using modal analysis using the shape basis of the modes of deformation. [Agudo et al., 2017] is based on the seminal work of [Bregler et al., 2000], which proposed a new method to recover 3D non-rigid shape models from image sequences by representing the 3D model as a combination of basis shapes. In [Agudo and M-Noguer, 2015] the elastic model of the object is estimated by approximating the full force-field from a low-rank basis. This approach estimates the entire force field acting on all points of the considered object while estimating the elastic model of that object and the pose of the camera using an Expectation Maximization (EM) algorithm. [Agudo et al., 2014] uses modal analysis from continuum mechanics to model the shape as discretized, linear elastic triangles and the solution to the force balance equations for an undamped free vibration model of the underlying FEM is used to recover the shape of the deforming object. Using a slightly different approach, [Malti and Herzet, 2017] proposes a new class of method, termed Sparse Linear Elastic - SfT (SLE-SfT), which replaces the ℓ_2-norm minimization of standard SfT approaches with ℓ_0-norm for minimizing the cardinal of the non-zero components of the deforming

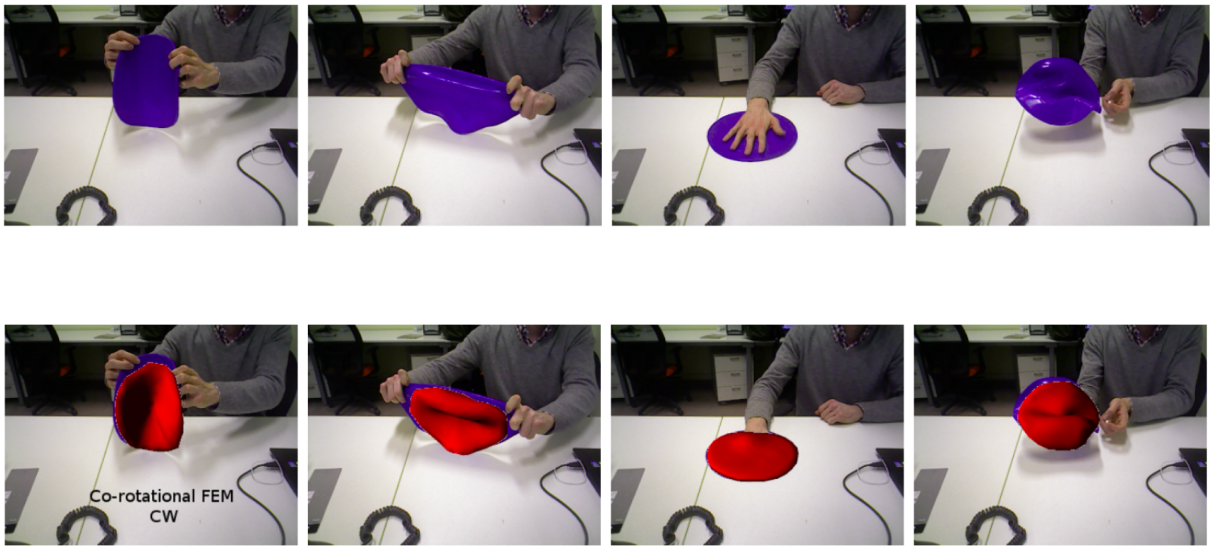


FIGURE 1.10 – Physical model based tracking of deforming objects with the approach of [Petit et al., 2015a]. *Top:* input images and *bottom:* tracked model. (Image Courtesy: [Petit et al., 2015a])

forces. An alternate formulation minimizing the ℓ_1-norm of the sum of absolute values of force component has also been proposed. None of these approaches uses the Solid Boundary Constraints (SBC) to rigidly position the shape in the object frame.

[Royer et al., 2017] proposed an elastic model based technique for tracking of deformable internal organs of human using ultrasound data. [Petit et al., 2015a] proposes a real-time approach for tracking 3D deformable object using a volumetric mesh in real-time, with the help of FEM simulations. In this approach, the images from the RGB-D sensor are segmented to isolate the object to be tracked using a *grabcut* [Rother et al., 2004] approach, which is in turn utilized to extract a segmented pointcloud. To track the deformation of this object, an external force is applied to the mechanical model (simulated with FEM) of the object. The per-vertex force exerted is given by:

$$\mathbf{F}_i^{\text{ext}} = \boldsymbol{\zeta}_i k(\mathbf{P}_i^{\mathcal{V}_m} - \mathbf{P}_i^{\psi}) \tag{1.68}$$

where $\mathbf{F}_i^{\text{ext}}$ is the external force vector for the i-th element of $\mathcal{V}_m$ which is given by $\mathbf{P}_i^{\mathcal{V}_m}$, $\boldsymbol{\zeta}_i = \sigma_1 \mathrm{e}^{-(\frac{d_i}{\sigma_2})}$ such that d_i is the closest distance of $\mathbf{P}_i^{\mathcal{V}_m}$ from the projected 2D contour of the object, σ_1 and σ_2 are two empirically chosen constants, k is the stiffness of the element corresponding to the i-th node, and the variable $\mathbf{P}_i^{\psi}$ is obtained from the

segmented pointcloud $\boldsymbol{\psi}$ by the formulation:

$$\mathbf{P}_i^{\boldsymbol{\psi}} = \begin{cases} \alpha \mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m}) + (1-\alpha)\frac{1}{\#\boldsymbol{\psi}_i} \sum\limits_{\mathbf{P}_j \in \boldsymbol{\psi}_i} \mathbf{P}_j & \text{if } \#\boldsymbol{\psi}_i > 0 \\ \alpha \mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m}) + (1-\alpha)\mathbf{P}_i^{\mathcal{V}_m} & \text{else} \end{cases} \tag{1.69}$$

where $\mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m})$ denotes the nearest neighbor of $\mathbf{P}_i^{\mathcal{V}_m}$ in $\boldsymbol{\psi}$, $\boldsymbol{\psi}_i$ is a subset of $\boldsymbol{\psi}$ such that $\forall(\boldsymbol{\psi}_i)_j \in \boldsymbol{\psi}_i$ there exists $\mathrm{NN}_{\mathcal{V}_m}((\boldsymbol{\psi}_i)_j) = \mathbf{P}_i^{\mathcal{V}_m}$ and α is a tunable parameter. An example of the tracking achieved by this approach is shown in Fig. 1.10. [Petit et al., 2015b] extends the same approach to track fractures in the deforming objects, while tracking their deformation at the same time. Both these approaches use co-rotational FEM as the mechanical model and simulates the full volumetric deformation of the object to track deformation. [Petit et al., 2018] also extends this method to capture collision among interacting objects while tracking their deformation simultaneously.

[Wang et al., 2015] proposes a method to capture deformation of soft objects using co-rotational FEM, where the tracking is handled with a probabilistic approach. Given a set of points $\boldsymbol{\psi}_i \in \boldsymbol{\psi}$ in a pointcloud, the problem statement is to fit a surface model $\mathcal{V}_s$ to the pointcloud to maximize the probability (represented by the function $\mathrm{p}(\cdot)$) of vertex position, given the measurement of the pointcloud, such that:

$$\mathcal{V}_s = \arg\max_{\mathcal{V}_s} \mathrm{p}(\mathcal{V}_s|\boldsymbol{\psi}) \tag{1.70}$$

This maximum-aposteriori-estimation (MAP) problem is solved by regulating the external force acting on the mechanical model $\mathcal{V}_m$ which is connected with the surface model $\mathcal{V}_s$ in a scheme similar to [Petit et al., 2015a], as shown in Fig. 1.5. This external, virtual force is expressed as:

$$\mathbf{f}_j = \boldsymbol{\eta} \sum_i \mathrm{p}(z_{ji}) \boldsymbol{\sigma}_j^{-1} (\mathbf{P}_i^{\boldsymbol{\psi}} - \mathbf{P}_j^{\mathcal{V}_s}) \tag{1.71}$$

where $\boldsymbol{\eta}$ is a scaling factor, $\boldsymbol{\sigma}_j$ is the diagonalized co-variance matrix of $\mathbf{P}_i^{\boldsymbol{\psi}}$ and z_{ji} is a latent variable that indicates if $\mathbf{P}_i^{\boldsymbol{\psi}}$ has any contribution towards $\mathbf{P}_j^{\mathcal{V}_s}$ or not. A graphical outline of this approach is shown in Fig. 1.11.

It must be noted that [Wang et al., 2015] is an improved variant of the approach proposed in [Schulman et al., 2013] and most of the fundamental concepts of tracking pointclouds using virtual force is based on Schulman et. al. However, unlike [Schulman

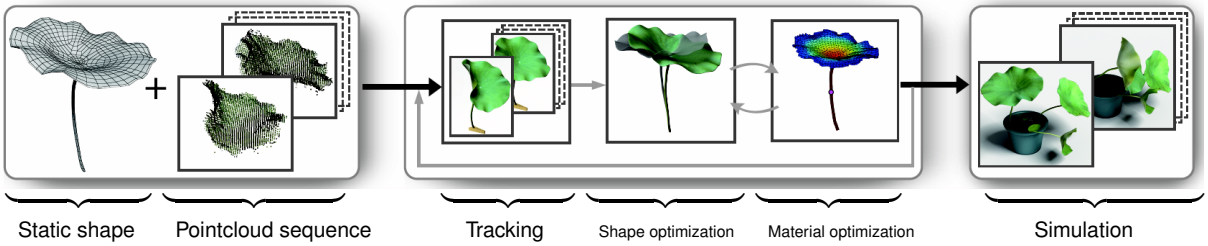


FIGURE 1.11 – Outline for deformation tracking by fitting a FEM based object model to a point-cloud by controlling the forces applied on the mesh vertices, as given in Eqn. 1.71 (Image Courtesy [Wang et al., 2015])

et al., 2013], Wang et. al. extends the deformation tracking approach to also estimate the material properties of the object being tracked by studying the oscillation of the object as it settles down to its original, undeformed configuration after a simple twisting or pulling interaction. A closely related problem statement was also considered by [Petit et al., 2017a] wherein the material properties of a deforming object that is being manipulated by a robotic arm was estimated, as well as the force on the contact points were tracked after the estimation of the material property has been done using the tracking mechanism of [Petit et al., 2015b].

1.3.2 Geometric Models

As delineated in Fig. 1.4, not all model based tracking methodology uses physically based models to explain the deformation. There are many approaches in the non-rigid object tracking literature that utilizes geometric models, often inspired from the computer graphics community, to track deformation. We now describe some of the fundamental concepts required to model and track deformations geometrically.

To control the deformation of a mesh model geometrically, we need a regularizing function or a method that can propagate the displacement of a single vertex, often called the *control handle*, to the adjacent mesh vertices. It is preferable to utilize a method that propagates the deformation without causing any un-realistic mesh deformation or decimation. Obviously, these methodologies have many different sub-categories, which are frequently combined with each other to derive numerous variations of geometric deformations models. However, it is still possible to cluster most of these methods into some frequently recurring themes, some of which are highlighted below:

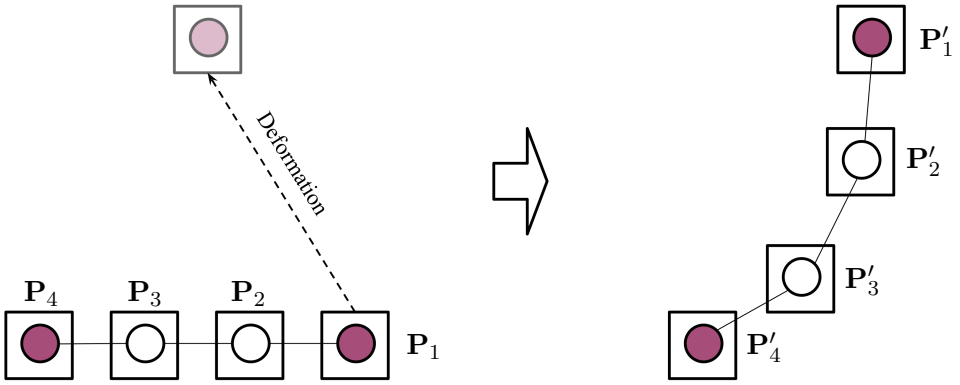


FIGURE 1.12 – Regularization of deformation using embedded deformation graphs

Embedded Deformation Graph

In this approach, the spatial deformation of an object is represented by a collection of affine transformations organized in a graph structure. Each node of the graph has a particular transformation associated with it and the deformation is propagated to the nearby space by a series of affine transforms, as explained below. Let a particular node $\mathbf{P}_i$ has the rotational matrix $\mathbf{R}_i$ and translation $\mathbf{t}_i$ associated with it. This transformation is propagated to a node $\mathbf{P}_j$ (which is connected to $\mathbf{P}_i$ through a graph of undirected edges) to produce its deformed configuration $\mathbf{P}'_j$ via the relationship:

$$\mathbf{P}'_j = \sum_{i=1}^{n} \mathrm{w}_i(\mathbf{P}_j)\Big(\mathbf{R}_i(\mathbf{P}_j - \mathbf{P_i}) + \mathbf{P}_i + \mathbf{t}_i\Big) \tag{1.72}$$

where $\mathrm{w}_i(\mathbf{P}_j)$ is a weighting parameter. As shown in Fig. 1.12, with this deformation propagation mechanism in place [Sumner et al., 2007], a set of rotational and regularizational constraints can be imposed on this graph structure to result in spatially coherent and smooth deformation when some controlled nodes are deformed arbitrarily (either by user specified commands, or by minimization of some secondary error terms).

Thin plate spline (TPS)

TPS warp presents an efficient shape interpolation and smoothing function for controlling or regularizing the deformation of non-rigid surfaces in 2D or 3D. In its sim-

plest form, the 3D interpolation is given by:

$$\mathbf{P} = a_0 + a_1 \mathrm{X}_{\mathbf{P}} + a_2 \mathrm{Y}_{\mathbf{P}} + a_3 \mathrm{Z}_{\mathbf{P}} + \sum_{i=1}^{\mathrm{N}} \lambda_i \mathrm{U}\Big(\mid\mid \mathbf{P} - \mathbf{P}_i \mid\mid \Big) \tag{1.73}$$

where

$$\Big\{ \mathbf{P}_i = (\mathrm{X}_{\mathbf{P}_i}, \mathrm{Y}_{\mathbf{P}_i}, \mathrm{Z}_{\mathbf{P}_i}) \mid i = 1, 2, \cdots, \mathrm{N} \Big\} \tag{1.74}$$

are the landmark points, $\Big\{ a_0, a_1, a_2, a_3, \lambda_1, \cdots, \lambda_{\mathrm{N}} \Big\}$ are weighting coefficients and $\mathbf{P} = (\mathrm{X}_{\mathbf{P}}, \mathrm{Y}_{\mathbf{P}}, \mathrm{Z}_{\mathbf{P}})$ are the interpolated points. $\mathrm{U}(x) = x^2 \log(x^2)$ is the radial basis function.

As-rigid-as-possible (ARAP) regularizer

Assuming that we have a triangular mesh $\mathcal{V}$ which gets deformed to a new configuration $\mathcal{V}'$, while two connected model vertices are denoted by $\mathbf{P}_i$ and $\mathbf{P}_j$, ARAP [Bregler et al., 2000] proposes to regularize deformation by minimizing the energy term:

$$\mathrm{E}_{\mathrm{ARAP}} = \sum_{i=1}^{n} w_i \sum_{j \in \mathbf{X}^{\mathcal{V}}} w_{ij} \mid\mid \Big(\mathbf{P}'_i - \mathbf{P}'_j \Big) - \mathbf{R} \Big(\mathbf{P}_i - \mathbf{P}_j \Big) \mid\mid^2 \tag{1.75}$$

where w_i and w_{ij} are suitable cell and edge weights, while $\mathbf{R}$ is the approximate rigid rotation matrix between $\mathcal{V}$ and $\mathcal{V}'$.

Coherent Point Drift (CPD)

In its simplest form, CPD [Myronenko and Song, 2010] offers a mechanism to align two pointclouds with each other, irrespective of the nature of the transformation between them, i.e., it can be used to estimate both rigid and non-rigid transformations. Assuming that a set of target point cloud is denoted by $\Big\{ \boldsymbol{\psi}_{\mathrm{X}} = \{\mathbf{p}_{x_i}\} \mid i = 1, \cdots, \mathrm{N} \Big\}$, while $\boldsymbol{\psi}^*_{\mathrm{Y}}$ denotes the centroids of the Gaussian Mixture Model (GMM) of another set of source pointcloud $\Big\{ \boldsymbol{\psi}_{\mathrm{Y}} = \{\mathbf{p}_{y_j}\} \mid j = 1, \cdots, \mathrm{M} \Big\}$, CPD based alignment involves the minimization of the following term:

$$\mathrm{E}_{\mathrm{CPD}} = -\sum_{n=1}^{\mathrm{N}} \log \sum_{m=1}^{\mathrm{M}+1} \mathrm{P}(m) \mathrm{p}(\mathbf{p}_x \mid m) \tag{1.76}$$

where

$$\mathrm{p}(\mathbf{p}_x \mid m) = \frac{1}{(2\pi\sigma^2)} \mathrm{e}^{-\frac{||\mathbf{p}_x - \mathbf{p}_{ym}||^2}{2\sigma^2}} \tag{1.77}$$

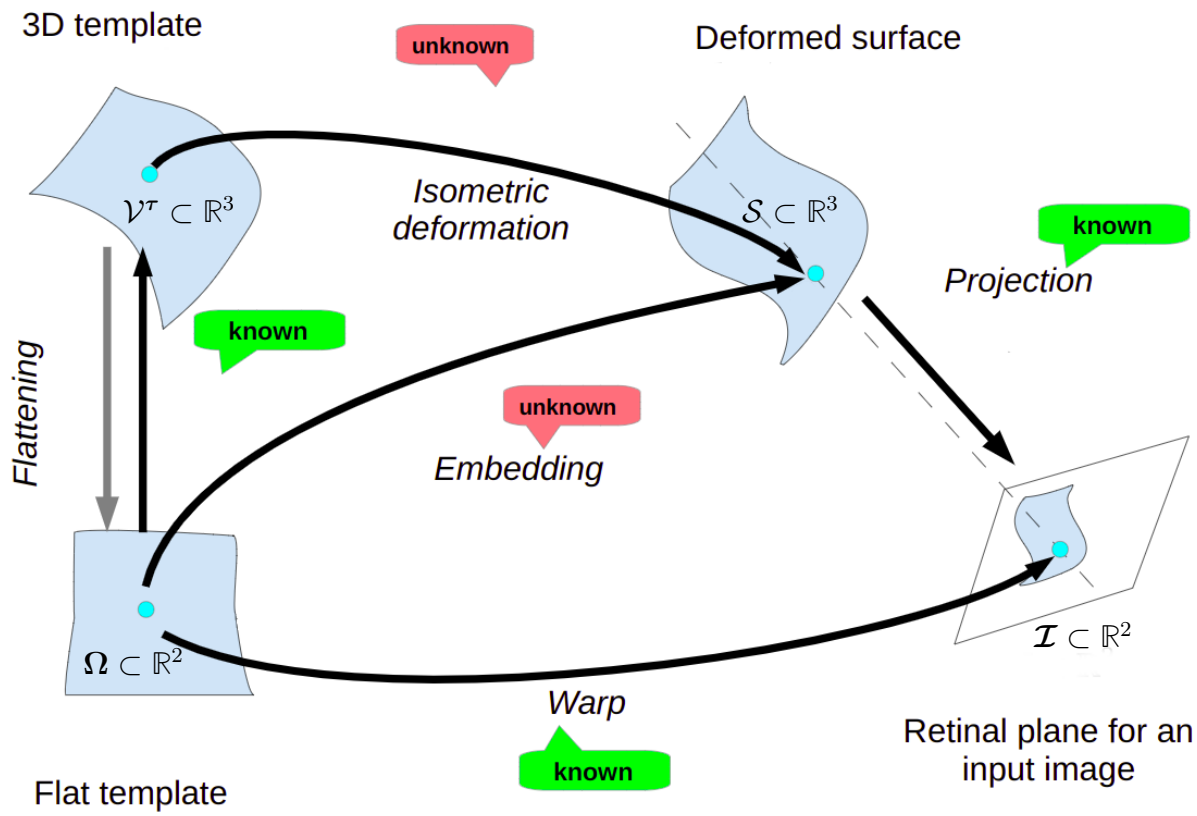


FIGURE 1.13 – The problem statement of SfT (Image courtesy [Chhatkuli, 2016])

Here, membership probability $\mathrm{P}(m) = \frac{1}{\mathrm{M}}$ while σ^2 is the isotropic covariance of the GMM components. The parameters of the GMM as well as σ^2 can be estimated using an Expectation Maximization (EM) [Dempster et al., 1977] strategy, thereby providing us with a method to register both rigidly transforming and/or deforming points.

Shape-from-Template (SfT)

As demonstrated in Fig. 1.13, the basic problem statement of SfT involves determining the function $\boldsymbol{\varphi} \in \mathbb{R}^3$ that embeds a flat template image $\boldsymbol{\Omega} \subset \mathbb{R}^2$ into a 3D deforming surface $\mathcal{S} \subset \mathbb{R}^3$, given that the 3D undeformed model $\mathcal{V}^\tau \subset \mathbb{R}^3$ is given and the warping function $\boldsymbol{\varrho} \subset \mathbb{R}^2$ between the 2D template and the image of the deformed surface can be determined externally. [Bartoli et al., 2015] provides the basic solution to this

problem. The embedding function can be represented as:

$$\varphi = \chi \circ \Delta \tag{1.78}$$

where χ is the isometric deformation of $\mathcal{V}^\tau$ and Δ is the function that embeds Ω into $\mathcal{V}^\tau$. On such a system, two different constraints are imposed. The first constraint aims to preserve the isometry, and is given by:

$$\mathbf{J}_\varphi^\top \mathbf{J}_\varphi = \mathbf{J}_\Delta^\top \mathbf{J}_\Delta \tag{1.79}$$

where $\mathbf{J}_\varphi$ and $\mathbf{J}_\Delta$ are the Jacobians obtained by differentiating φ and Δ. The second constraint ensures the consistency between the warp and the projection of the embedding in the image Π, and is given by:

$$\varrho = \Pi \circ \varphi \tag{1.80}$$

Eqn. 1.79 and eqn. 1.80 forms a system of partial differential equations and an analytic solution for the unknown parameters has been established in [Bartoli et al., 2015]. Many variants of the standard SfT formulation exists in the literature and is briefly reviewed in the following paragraph.

1.3.2.1 Tracking using Geometric Models

A method to detect and recover three dimensional shape from monocular video sequence was proposed in [Salzmann et al., 2007]. This approach creates a low dimensional model of deformable, planar 3D surfaces and based on the assumption that the deformation is strictly isometric, a dimensionality reduction technique is used to the deformation models needed for tracking and detection. [Pilet et al., 2008] provided a method to detect and register non-rigid, planar surfaces based on wide-baseline point matches between the undeformed and deformed image of an object. [Salzmann et al., 2008] offers a closed form solution to the problem of recovering 3D shape from non-rigid, elastic 3D-to-2D correspondences. The set of 3D-to-2D correspondences constitutes a linear system, which can also be expressed as a weighted sum of eigen vectors, the solution of which accounts for the inextensibility constraints between neighboring mesh vertices.

[Bartoli et al., 2015] proposes a *first-order* method using both image point loca-

FIGURE 1.14 – Tracking of a deforming object using the approach proposed in [Salzmann et al., 2008] (Image Courtesy [Salzmann et al., 2008])

tions and the first-order differential structure extracted from the warp from template of an object to an input, deformed image to reconstruct the deformed shape of the surface (the SfT problem). [Bartoli et al., 2015] dealt with planar meshes only. However, it was extended to handle volumetric meshes in [Parashar et al., 2015]. Two methods were proposed for interpolating the volumetric shape, one was based on a global smoothness prior, while the other is based on a locally rigid transform at every point of the visible surface, somewhat similar to ARAP. [Collins and Bartoli, 2015] describes a real-time SfT based approach. [Haouchine et al., 2014] combined SfT with elasticity priors with SfT to achieve realtime augmentation on planar deforming surfaces. [Willimon et al., 2012] proposes a joint energy minimization of sparse feature correspondences and depth based error terms along with smoothness priors to track deforming laminar objects from RGB-D data.

1.3.3 Non-rigid Tracking and Reconstruction

Model free tracking of deformation involves tracking the surface of the deforming object as well as iteratively generative the mesh model of the entire surface of the object or scene under consideration. Many of the fundamentals used for geometric model-based tracking are re-utilized for reconstruction and tracking as well. Tracking and reconstruction of non-rigid objects using RGB-D cameras have been developed in recent years. [Zollhöfer et al., 2014] tracked a deformable template in real-time, without having any limitations about apriori knowledge of the template. It must be noted that this research is entirely based on the output of a stereo rig and not from a depth camera. The approach works by minimizing the global rigid transformation as well as the per-vertex displacement between consecutive frames. This is done by minimizing four error terms: a point-to-point distance, a point-to-plane distance, direct color intensity difference and a geometric prior term acting as the ARAP regularizer.

Frame 3: Truncated Signed Distance Field (TSDF)

TSDF and Signed Distance Function (SDF) has been used extensively in rigid and non-rigid reconstruction from depth sensors in the recent years, including [Newcombe et al., 2011], [Izadi et al., 2011], [Newcombe et al., 2015], [Slavcheva et al., 2016], [Slavcheva et al., 2017a] and [Slavcheva et al., 2018], among many others. The TSDF algorithm provides a simple method to reconstruct rigid scene from multiple depth scans (using any depth sensor) and it is possible to modify the TSDF algorithm to effectively reconstruct non-rigid scenes too. Given its efficacy and popularity, it can be useful to take a closer look at the TSDF methodology for the sake of completeness. TSDF was introduced for rigid object reconstruction by [Newcombe et al., 2011] and [Izadi et al., 2011]. However, the principle of TSDF is heavily based on SDF, introduced by [Curless and Levoy, 1996]. SDF is a voxel based implicit representation of a scene, where each voxel is associated with a function measuring the distance of the voxel from the nearest surface.

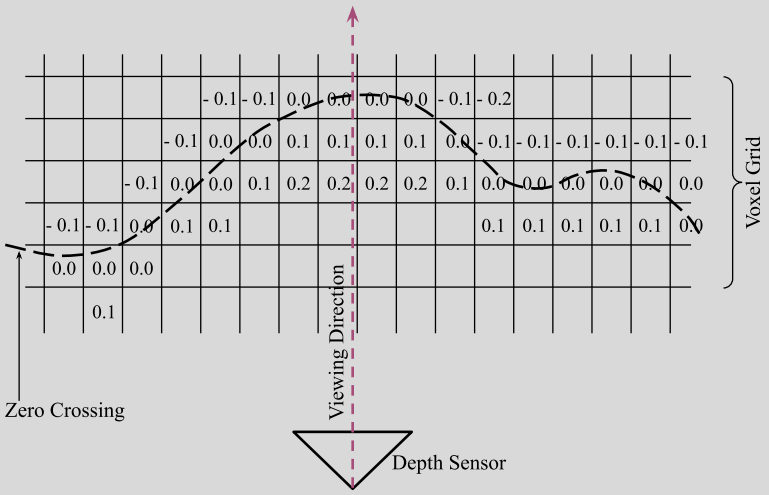


FIGURE 1.15 – SDF computation from depth scan

Given a depth sensor, the SDF algorithm proposes to maintain a set of voxels which contain the distance function $\mathrm{d}_{\mathrm{sdf}}$ that gives the distance of a voxel from the nearest surface $\mathbf{S}_{\mathrm{sdf}}$ and has a positive sign if it is closer to the sensor w.r.t the viewing direction and negative if it is farther, as shown in Fig. 1.15. For every new data frame from the depth sensor, the range surface ($\mathbf{S}^k_{\mathrm{sdf}}$ for the k-th frame) is a piecewise linear surface created by connecting nearest neighboring points with triangles in the depth scan. Every voxel is also associated with a weight function $\mathrm{w}_{\mathrm{sdf}}$, which maybe an elaborate function modeling the uncertainty in measurement at each updated voxel, or in the simplest case of [Curless and Levoy, 1996], proportional to the angle between the viewing direction of the sensor and the surface normal of $\mathbf{S}_{\mathrm{sdf}}$ at a point closest to the given voxel. The algorithm for updating the SDF for the i-th voxel in a new data frame ($(k+1)$-th range image) is given below :

Algorithm 1: Computing SDF from depth data

Construct $\mathbf{S}^{k+1}_{\mathrm{sdf}}$ from $(k+1)$-th range data ;
for $i \leftarrow 1$ **to** *no. of voxels in* $(k+1)$*-th range image* **do**
  Compute $(\hat{\mathrm{d}}^{k+1}_{\mathrm{sdf}})_i$ and $(\hat{\mathrm{w}}^{k+1}_{\mathrm{sdf}})_i$, which are i-th value of $\mathrm{d}_{\mathrm{sdf}}$ and $\mathrm{w}_{\mathrm{sdf}}$ respectively, computed just from $\mathbf{S}^{k+1}_{\mathrm{sdf}}$;
  $(\mathrm{d}^{k+1}_{\mathrm{sdf}})_i = \frac{(\hat{\mathrm{d}}^{k+1}_{\mathrm{sdf}})_i(\hat{\mathrm{w}}^{k+1}_{\mathrm{sdf}})_i + (\mathrm{d}^{k}_{\mathrm{sdf}})_i(\mathrm{w}^{k}_{\mathrm{sdf}})_i}{(\hat{\mathrm{w}}^{k+1}_{\mathrm{sdf}})_i + (\mathrm{w}^{k}_{\mathrm{sdf}})_i}$;
  $(\mathrm{w}^{k+1}_{\mathrm{sdf}})_i = (\hat{\mathrm{w}}^{k+1}_{\mathrm{sdf}})_i + (\mathrm{w}^{k}_{\mathrm{sdf}})_i$
end

[Izadi et al., 2011] modified the method of Algorithm 1 to store the SDF only for a truncated region around the actual surface, resulting in the TSDF value $\mathrm{d}_{\mathrm{tsdf}}$, given by :

$$(\mathrm{d}_{\mathrm{tsdf}}^{k+1})_i = \begin{cases} \min\Big(1, \frac{(\mathrm{d}_{\mathrm{sdf}}^{k+1})_i}{\sigma_{\max}}\Big) & \text{If } (\mathrm{d}_{\mathrm{sdf}}^{k+1})_i > 0 \\ \max\Big(-1, \frac{(\mathrm{d}_{\mathrm{sdf}}^{k+1})_i}{\sigma_{\min}}\Big) & \text{else} \end{cases} \tag{1.81}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum truncation depth required by the algorithm. An example of surface reconstruction from 3 consecutive RGB-D frames (captured at 30 fps) using this TSDF methodology is demonstrated in Fig. 1.16.

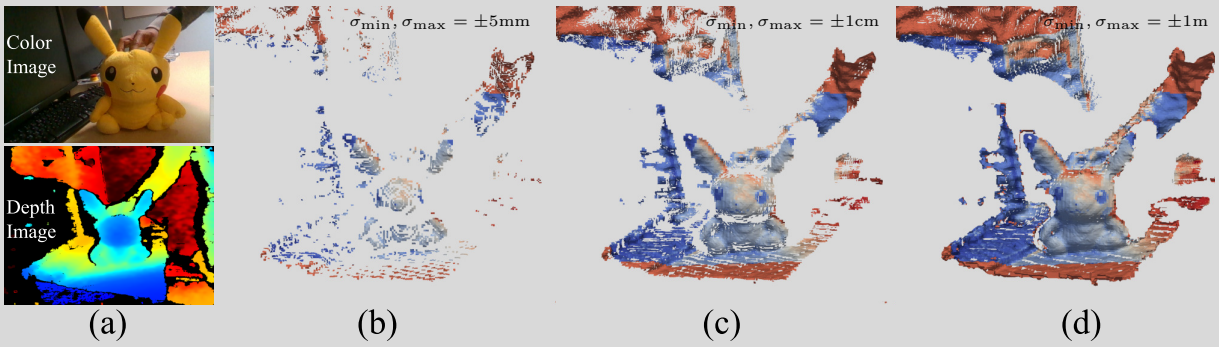


FIGURE 1.16 – Example of rigid reconstruction using TSDF methodology (similar to [Izadi et al., 2011]) for 3 consecutive frames coming from a RGB-D camera, as shown in (a). The TSDF volume is computed and the isosurface is extracted using the Marching Cubes method [Lorensen and Cline, 1987] with a truncation distance of 5mm, 1cm and 1m for (b), (c) and (d) respectively

DynamicFusion [Newcombe et al., 2015] extended the premise of *KinectFusion* into non-rigid objects. It enabled the tracking and modeling of deformable objects on the go, without using a previously known template. In *DynamicFusion*, the objective is split into three steps, estimating the warping function of the current depth frame from the canonical model, fusion of the live depth frame into the canonical model once the warping function has been determined and adaption of the warp field structure to track the camera movement using ICP. The non-rigid warp field is represented as a 6D transformation in $\mathrm{SE3}$, but to avoid memory overhead, this 6D transformation is kept sparse, while all the dense points in between are interpolated using a weighted blending of dual quaternion. The weight alters the radius of influence of each transformation node. Fusion between the current depth frame and the canonical model is done using the TSDF fusion technique. The warp field is estimated by minimizing an energy function comprising of a data term, which is the dense model-to-frame ICP cost, and a regula-

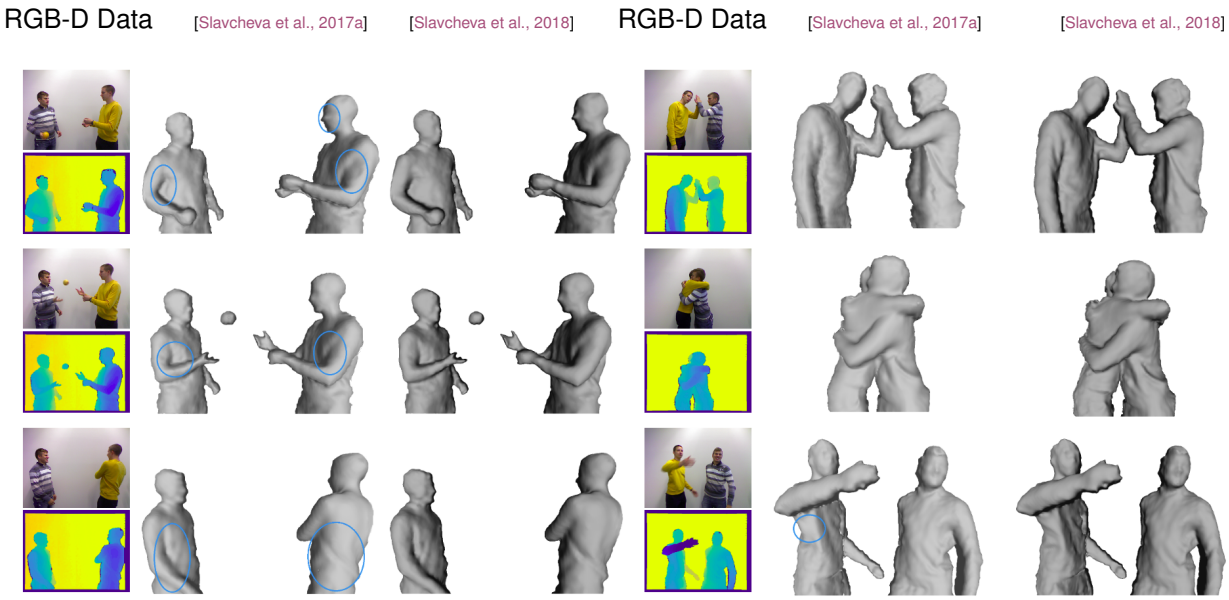


FIGURE 1.17 – Comparison between [Slavcheva et al., 2017a] and [Slavcheva et al., 2018]. The highlighted regions show the over-smoothing tendencies of [Slavcheva et al., 2017a] (Image Courtesy [Slavcheva et al., 2018])

rization parameter. The results of *DynamicFusion* has been demonstrated on a variety of dataset consisting of 3D toys and human body.

[Gao and Tedrake, 2018] replicates the approach of *DynamicFusion* using *surfels* instead of volumetric TSDF, thereby imporving computational time requirements. [Innmann et al., 2016] extended the work of *DynamicFusion* to produce even more robust and accurate results. *VolumeDeform* is quite similar to *DynamicFusion*. The notable differences between the two approach is that *VolumeDeform* does not use dual quaternion based interpolation to convert sparse correspondence to dense. Instead, it uses a novel point-to-plane alignment strategy as the dense term of the energy function. It also uses a SIFT feature based sparse color point correspondence to robustify the tracking of the model. A regularization term is maintained as before. [Slavcheva et al., 2017a] proposes a non-rigid reconstruction technique using a combination of three data terms: the first penalizing Euclidean distance between the voxel positions, the second ensuring that the non-rigid deformation follows the Killing property [Ben-Chen et al., 2010] and the third ensuring that the gradient magnitude of SDF always remains unity. Named after Wilhelm Killing [Coleman, 1989], Killing vector fields are used in the context of deformation tracking as an isometric or nearly-isometric regularizer. It is

surprisingly straightforward to adapt the *Killing conditions* to act as a geodetic distance preserving constraint on deforming, discrete manifolds. Assuming a warp field denoted by $\left\{\boldsymbol{\varphi} = (\alpha, \beta, \gamma) \mid \boldsymbol{\varphi} \in \mathbb{R}^3\right\}$ acting on a set of points which can be represented in the form $\mathbf{P} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})$, the Jacobian of the warp field is given by $\mathbf{J}_\varphi = \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{P}}$. On such a vector field, the *Killing condition* is denoted by $\mathbf{J}_\varphi + \mathbf{J}_\varphi^\top = 0$. However, to allow nearly isometric deformations, most authors prefer to penalize deformation by minimizng the *Killing energy*:

$$\mathrm{E}_{\mathrm{AKVF}} = \sum_{\mathrm{X,Y,Z}} || \mathbf{J}_\varphi + \mathbf{J}_\varphi^\top ||_{\mathrm{F}}^2 \tag{1.82}$$

where $|| \cdot ||_{\mathrm{F}}^2$ is the Frobenius norm.

Assuming that $\boldsymbol{\Upsilon}$ gives the current TSDF volume, a comparison between [Newcombe et al., 2015], [Gao and Tedrake, 2018], [Innmann et al., 2016] and [Slavcheva et al., 2017a] can be delineated as given in Table 1.1. [Slavcheva et al., 2018] alters the approach of [Slavcheva et al., 2017a] by proposing a Tokhonov regularization strategy of Sobolev gradient flow to replace the level-set data term and approximate Killing condition based regularization of [Slavcheva et al., 2017a].

	DynamicFusion [Newcombe et al., 2015]	SurfelWarp [Gao and Tedrake, 2018]	VolumeDeform [Innmann et al., 2016]	KillingFusion [Slavcheva et al., 2017a]
Cost Function	$\mathrm{E} = \mathrm{E}_{\mathrm{data}} + \lambda \mathrm{E}_{\mathrm{reg}}$	$\mathrm{E} = \mathrm{E}_{\mathrm{depth}} + \lambda \mathrm{E}_{\mathrm{reg}}$	$\mathrm{E} = w_d \mathrm{E}_{\mathrm{dense}} + w_r \mathrm{E}_{\mathrm{reg}} + w_s \mathrm{E}_{\mathrm{sparse}}$	$\mathrm{E} = \mathrm{E}_{\mathrm{data}} + w_r \mathrm{E}_{\mathrm{levelset}} + \mathrm{E}_{\mathrm{killing}}$
Data Term	$\mathrm{E}_{\mathrm{data}}$: pt.-to-plane distance between transformed model surface and live vertices in $\boldsymbol{\Upsilon}$ with a Tukey based penalty	$\mathrm{E}_{\mathrm{depth}}$: pt.-to-plane distance between transformed model surface and live vertices in $\boldsymbol{\Upsilon}$	$\mathrm{E}_{\mathrm{dense}}$: pt.-to-plane distance between rasterized iso-surface of canonical model and current pointcloud	$\mathrm{E}_{\mathrm{data}}$: squared norm of the displacement vector between global canonical frame and current SDF
Regularization	$\mathrm{E}_{\mathrm{reg}}$: displacement between current and previous vertex position	$\mathrm{E}_{\mathrm{reg}}$: norm of displacement between current and previous vertex position	$\mathrm{E}_{\mathrm{reg}}$: displacement in grid vertices between $\boldsymbol{\Upsilon}$ and the isosurface	$\mathrm{E}_{\mathrm{levelset}}$: difference between magnitude of SDF and unity
Additional Constraint			$\mathrm{E}_{\mathrm{sparse}}$: Euclidean distance between points matched by SIFT	$\mathrm{E}_{\mathrm{killing}}$: damped Killing condition enforced on the Jacobian of the displacement field

TABLE 1.1 – Comparison of the error terms that is minimized by some selected non-rigid tracking and reconstruction approaches. λ, w_d, w_r and w_s are weighting terms

1.3.4 Non-Rigid Structure from Motion (NR-SfM)

NR-SfM is an excellent technique to resolve the shape of a deforming object along with camera motion parameters. It is inherently a batch technique, requiring sequential observation of deforming keypoints on the object. This makes it somewhat unsuitable for sequential, fast tracking of deforming surfaces. However, for the sake of completeness, we take a brief look at some of the notable approaches developed from this technique.

Let the shape of a non-rigid object be described by a set of K keyframe basis, denoted by $\underset{3\times\mathrm{P}}{\mathbf{S}_i}$ denoting P points. The overall shape can be described using:

$$\mathbf{S} = \sum_{i=1}^{\mathrm{K}} l_i \cdot \mathbf{S}_i \mid l_i \in \mathbb{R} \tag{1.83}$$

Under scaled orthographic projection, the basis set can be used to express the image coordinates (u_i, v_i) of the keypoints as:

$$\begin{bmatrix} u_1 & \cdot & \cdot & \cdot & u_{\mathrm{P}} \\ v_1 & \cdot & \cdot & \cdot & v_{\mathrm{P}} \end{bmatrix} = \begin{bmatrix} l_1\mathbf{R}_{[2]} & \cdot & \cdot & \cdot & l_{\mathrm{K}}\mathbf{R}_{[2]} \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{S}_{\mathrm{K}} \end{bmatrix} \tag{1.84}$$

where $\mathbf{R}_{[2]}$ gives the first two rows of the camera rotation matrix. Assuming we have L data frames for the sequence, we can re-arrange Eqn. 1.3.4 as:

$$\underset{2\mathrm{L}\times\mathrm{P}}{\varsigma} = \begin{bmatrix} u_1^1 & \cdot & \cdot & \cdot & u_{\mathrm{P}}^1 \\ v_1^1 & \cdot & \cdot & \cdot & v_{\mathrm{P}}^1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ u_1^{\mathrm{L}} & \cdot & \cdot & \cdot & u_{\mathrm{P}}^{\mathrm{L}} \\ v_1^{\mathrm{L}} & \cdot & \cdot & \cdot & v_{\mathrm{P}}^{\mathrm{L}} \end{bmatrix} = \underbrace{\begin{bmatrix} l_1^1\mathbf{R}_{[2]}^1 & \cdot & \cdot & \cdot & l_{\mathrm{K}}^1\mathbf{R}_{[2]}^1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ l_1^{\mathrm{L}}\mathbf{R}_{[2]}^{\mathrm{L}} & \cdot & \cdot & \cdot & l_{\mathrm{K}}^{\mathrm{L}}\mathbf{R}_{[2]}^{\mathrm{L}} \end{bmatrix}}_{\underset{2\mathrm{L}\times3\mathrm{K}}{\mathbf{Q}}} \underbrace{\begin{bmatrix} \mathbf{S}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{S}_{\mathrm{K}} \end{bmatrix}}_{\underset{3\mathrm{K}\times\mathrm{P}}{\mathbf{S}_l}} \tag{1.85}$$

ς is termed as the *tracking matrix* and can be formed by observing a set of keypoints across multiple frames (e.g: using a feature detector/matching technique). Applying a

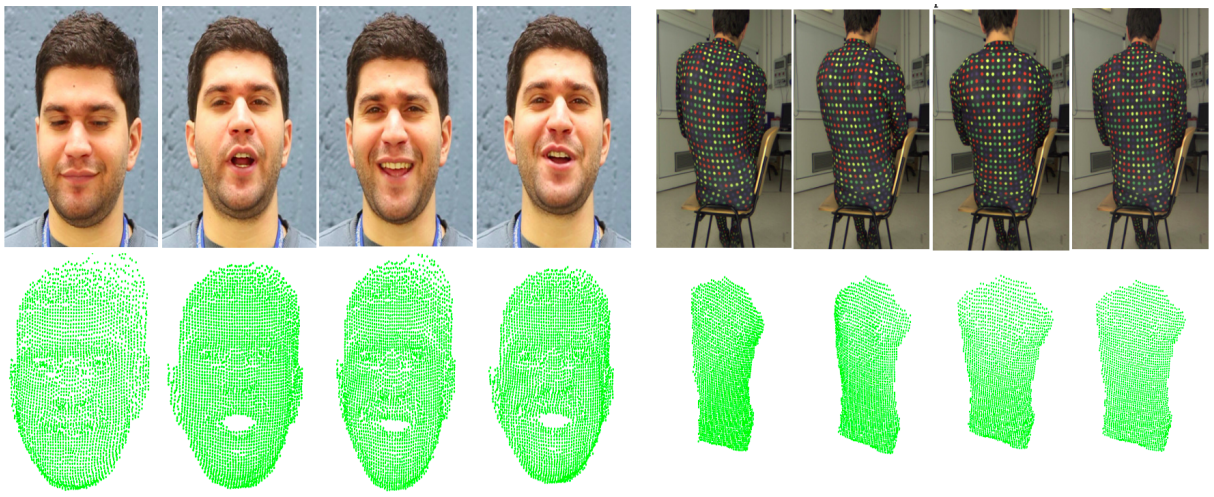


FIGURE 1.18 – Results from applying NR-SfM using the approach of [Kumar et al., 2016]. The top row shows the input images, while the row at the bottom shows the extracted meshes. (Image Courtesy [Kumar et al., 2016])

SVD decomposition on ς, we get SVD: $\varsigma = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{Q}\mathbf{S}_l$. Solving this equation, the rotational pose of camera $\mathbf{R}$ and shape configuration $\mathbf{S}$ can be factored simultaneously.

[Gotardo and Martinez, 2011a] proposed a kernel trick to better constraint the weakened low rank constraint, without requiring additional basis shapes to model keypoints moving along curved paths. [Gotardo and Martinez, 2011b] improved the standard NR-SfM representation by considering K complementary spaces of rank 3 as the 3D basis shapes for factorization. [Kumar et al., 2016] extends NR-SfM to multibody deformation tracking using a subspace clustering approach (Fig. 1.18).

1.4 Positioning this Thesis

Having provided an overview of the existing state-of-the-art for deformation tracking, we now discuss the aim of this thesis and compare our contribution with the available literature from an application perspective. Before we begin discussing the technical details in the subsequent chapters, it is important to justify the approach that we utilize to track deforming objects. It is important to weigh the relative advantages and disadvantages of the state-of-the-art approaches in the context of this thesis.

The prominent research approaches presented in Sec. 1.3.1 through Sec. 1.3.4 can be grouped on the basis of the input data, as shown in Table 1.2. We discard the

Monocular	*NR-SfM* [Gotardo and Martinez, 2011b] [Gotardo and Martinez, 2011a] [Kumar et al., 2016]	*SfT*: [Bartoli et al., 2015] [Malti et al., 2015] (planar); [Parashar et al., 2015] (volumetric)	*Planar model* [Salzmann et al., 2007] [Salzmann et al., 2008] [Pilet et al., 2008] [Agudo et al., 2014] [Agudo et al., 2017]
Depth-based	*Reconstruction & Tracking* [Newcombe et al., 2015] [Slavcheva et al., 2018]	*Volumetric Model* [Schulman et al., 2013] [Petit et al., 2015b]	

TABLE 1.2 – Grouping some of the prominent approaches from state-of-the-art on the basis of the input data

approaches that primarily confine themselves to monocular input images. Inexpensive depth cameras are readily available as consumer-grade sensors (e.g.: the ones mentioned in Sec. 1.1.4) and not using the depth information is unnecessary and counter-intuitive for robotic applications. We also want to deal with approaches that track objects based on volumetric information, i.e., laminar objects are not of much interest to this thesis.

Once we remove all the monocular camera based approaches and laminar object tracking methodologies, we are left with the geometric and physical model based approaches. We are also left with reconstruction and tracking algorithms. At this stage, it is important to recall the objective for the non-rigid tracking approaches proposed in this dissertation. This thesis has been primarily directed towards aiding robotic manipulation of soft objects. In the context of robotic manipulation, it can be noted that only with volumetric and physical-model based approaches do we get the following advantage:

— A well-developed deformation model that is suitable for small and large deformation, i.e. FEM, co-rotational FEM etc.

— The ability to aide and enable robotic grasping and manipulation of deforming objects

— Tracking deformation model using coarse model of arbitrary resolution (i.e., suitable for tracking using object model of very low polygonal density) and mechanical model of arbitrary physical parameters

— Deformation tracking using physically-based models can be readily extended to act as a contact force tracker and also utilized to estimate the physical parameters of a deforming object

Given these strong advantages, we dedicate the rest of this dissertation to the study of deformation tracking using physically-based models.

1.5 Conclusion

In Sec. 1.1 of this chapter, we presented a summary of classical techniques involving visual tracking of rigid objects and physical modelling of deforming objects using standard and co-rotational FEM. In this thesis, a combination of these two paradigms are going to be used to design a set of methodologies that enable the tracking of deforming objects.

In Sec. 1.3 of this chapter, we discuss the state-of-the-art for non-rigid object tracking using visual information. Only a small subset of these approaches involve physical models for deformation tracking. We consider volumetric physical model based deformation tracking as the most relevant framework, since it enables us to achieve the following objectives (some of which has already been implemented while some which remains a future work [marked with a *]):

— Accurate and fast 3D tracking of deforming surface, with and without the knowledge of the physical parameters (Young's modulus, Poisson's ratio etc.). Presence of advanced physical model enables us to track large deformations

— Extending the 3D tracking framework to either track external, applied forces on the object or to estimate the physical parameters of the object

— Tracking multi-object collision and non-isometric deformations (cutting, tearing, fracturing etc.) of the object *

— Providing a deformation model that can be used to predict the future state of a deforming object when force is applied on it - which can have significant utilities in the area of robotic manipulation of compliant objects

Chapter 2 provides a framework for tracking of rigid objects, which is an essential initial step for subsequent non-rigid tracking. Chapter 3 details the first methodology developed to track deforming object using depth data only. This is followed by Chapter 4, which improves the approach of Chapter 3 to utilize both depth and photometric data simultaneously, while the tracking methodology evolves to become faster and more robust. Chapter 5 describes a robotic application of the approach developed in Chapter 3 involving estimation of physical parameters of objects and visual force tracking.

CHAPTER 2

RIGID OBJECT TRACKING

In the previous chapter, we introduced some of the existing literature on non-rigid object tracking and discussed some of the fundamental concepts about deformation modelling and tracking. Multiple real-time non-rigid tracking approaches, e.g., [Petit et al., 2018], uses two separate methods for non-rigid object tracking. One that approximately tracks the object rigidly, and another that tracks the deformation independently from the rigid motion. This is especially useful for the physical model based deformation tracking that we intend to do for this thesis. To this end, in this chapter we present a method to track rigid motion in objects with complex shapes using approximate 3D models of those objects. This is entirely based on RGB-D sensors. The key contributions of this chapter are:

— Combining point-to-plane distance minimization and photometry for tracking of complex objects using coarse model ;

— Using the concept of 'keyframe' in object tracking for increased robustness ;

— Accurate tracking of objects with inaccurate and coarse object models.

In this thesis, a *coarse model* implies a 3D, surface based model of an object without high density of vertices (approximately < 6800 vertices$/m^2$ on the surface of the mesh) and with relatively higher registration error ($\sim \pm 2cm$ in the worst cases) with the object.

2.1 Background

Although a brief summary of research about rigid object tracking was already provided in Sec. 1.2.2, it is worthwhile to look into some more state-of-the-art research related to rigid object tracking in the context of this chapter specifically. Research related to localizing and/or tracking simple geometric shapes is extensive [Lepetit et al., 2005] and has been studied since a long time [Armstrong and Zisserman, 1995]. Some approaches use probabilistic models for tracking shapes. [Prisacariu and Reid, 2012]

aims to maximize the discrimination between background and foreground of a scene while simultaneously tracking the 6DoF pose of an object with known 3D model. This method used only RGB images as input. [Teichman et al., 2013] extended this method to depth cameras, adding more cues like local image appearance, depth discontinuities, optical flow, and surface normal to inform the segmentation decision in a conditional random field model. Similar approaches have been used to track articulated objects [Cifuentes et al., 2017] with remarkable precision, but the tracking of articulated joints remain beyond the scope of our discussion. Particle filtering based approaches have been parallelized using GPU and implemented towards tracking rigid objects [Choi and Christensen, 2013]. The *second* group of algorithms uses learning-based techniques to track objects. [Hinterstoisser et al., 2012] proposes a pose estimation technique for complex objects under heavy clutter, but the training model needs color gradient information. In our method, however, we assume that the object's accurate model and any model based on color / texture are not available. [Doumanoglou et al., 2016], [Michel et al., 2017] and [Xiang et al., 2018] propose increasingly accurate posing estimation techniques, but they can not be considered real-time. [McCormac et al., 2018] and [Runz et al., 2018] are both a dynamic SLAM algorithm that uses R-CNN Mask for segmentation and performs object tracking as a by-product. The accuracy of the per frame object tracking has not been evaluated explicitly. The *third* group of algorithms use well known minimization techniques to track objects using geometric and photometric constraints. [Pomerleau et al., 2015] has already been introduced in Sec. 1.2 as an excellent summary of classical ICP. Many variants of ICP have been proposed for probabilistic [Segal et al., 2009] implementation of point set registration. In this context, SLAM and object tracking have been tackled interchangeably by some authors [Salas-Moreno et al., 2013]. CoFusion [Rünz and Agapito, 2017] is a dynamic SLAM system that uses a combination of dense ICP and photometric error minimization between the model and the current frame to track objects. The approach we propose, in terms of the framework for object tracking, is somewhat closer to CoFusion [Rünz and Agapito, 2017]. However, we do not undertake explicit motion segmentation, we are not interested in reconstruction and we use the concept of keyframes. Moreover, the presence of coarse object model makes it a different problem statement altogether.

In this chapter we focus on the high-accuracy tracking of rigid, complex shapes with approximately known geometry using depth cameras. We consider the *coarse*

model to be a highly decimated, minimal representation of the object model (instead of a high-resolution and detailed CAD model), containing very few triangular faces at best. The model does not contain any color or textural information. This is easy to generate and can also be rendered without a depth scanner (e.g: using manual measurements), making it suitable for various industrial applications. Our approach is significantly robust to measurement errors in the *coarse model*. We are interested in tracking all the 6 degrees of freedom [Yilmaz et al., 2006] of the object. The proposed approach is validated both quantitatively and qualitatively in the subsequent sections.

2.2 Method

The approach proposed here to track rigid objects is a combination of point-to-plane distance minimization and photometric error minimization between frames. We work with two types of input data: 1) registered and calibrated depth and grayscale images from the depth camera, and 2) simulated data of 3D points along with their corresponding grayscale values.

We denote the depth data (obtained from the sensor in the RGB-D format) as:

$$\boldsymbol{\psi} = \Big((\mathbf{P}_1, c_1), \cdots, (\mathbf{P}_N, c_N) \Big) \tag{2.1}$$

where $\mathbf{P}_i = (\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i)$ and c_i is the intensity value of the point $\mathbf{P}_i$, expressed in the camera centered coordinate frame. A function $\mathbf{\Pi}(\cdot)$ acts upon a 3D point and projects it to an image plane using the pinhole camera model such that $\mathbf{p}_i = \mathbf{\Pi}(\mathbf{P}_i)$. The function $c_i = \boldsymbol{\mathcal{I}}(\mathbf{p}_i)$ provides the image intensity of the point $\mathbf{p}_i$. $\nabla\boldsymbol{\mathcal{I}}_{i,x}$ and $\nabla\boldsymbol{\mathcal{I}}_{i,y}$ gives the gradient of the image along x and y axis.

The model of the object is also presumed to be known for the purpose of tracking. We represent it as a surface 3D mesh composed of a set of planes given by ${}^{\mathrm{O}}\mathcal{V}, \mathbf{X}^{\mathcal{V}}$ and $\mathbf{n}_j = (n_j^X, n_j^Y, n_j^Z)$ is the normal to the j-th plane of the mesh which lies at a perpendicular distance of d_j from the origin of the camera-centered coordinate frame.

The n-th frame from the depth sensor (or from a simulated data) is denoted by C_n, while certain frames get tagged as keyframes $\mathrm{C}_n \Rightarrow \mathrm{C}_n^{\mathrm{K}}$. $\mathcal{F}_{\mathrm{o}}$ denotes the object centered coordinate frame. These keyframes serve as the reference for photometric tracking and is explained in details in Sec. 2.2.3.

2.2.1 Tracking

For each data frame, we propose to minimize two cost functions that depend on a geometric term based on point-to-plane alignment and a photometric term that minimizes the difference in intensity between the predicted image (based on an estimate of the interframe transformation) and the projected image (based on the pointcloud transformation).

2.2.1.1 Point-to-plane Distance Minimization

For the geometric term, we minimize the point-to-plane distance between the 3D points in the n-th frame ψ_n, with respect to the set of planes registered with the point cloud in the previous frame ψ_{n-1}. This is given by minimizing the distance error:

$$\mathrm{e}_i^{dist}({}^n\mathbf{q}_{n-1}) = \Big(\big({}^n\mathbf{R}_{n-1}\mathbf{P}_i + {}^n\mathbf{t}_{n-1}\big)\cdot\mathbf{n}_k\Big) - d_k \tag{2.2}$$

where i denotes a specific point in the pointcloud and k is the index of the plane in the object model, to which it corresponds. We address the topic of this correspondence in the next subsection.

This cost function is optimized using Gauss-Newton optimization and the Jacobian is given by partial differentiation of eq. (2.2) with respect to ${}^n\mathbf{q}_{n-1}$, given by:

$$\mathbf{J}_i^{dist} = \begin{bmatrix}\mathbf{n}_k^\top & \big[\mathbf{n}_k\big]_\times \mathbf{P}_i{}^\top\end{bmatrix} \tag{2.3}$$

2.2.1.2 Photometric Minimization

The estimate at the first iteration for the transformation between the last keyframe $\mathrm{C}_p^{\mathrm{K}}$ and the current data frame C_n used in photometry is given by:

$${}^n\mathbf{T}_p = {}^n\mathbf{T}_O\big({}^p\mathbf{T}_O\big)^{-1} \tag{2.4}$$

where p denotes the data frame number for the last keyframe. From this initial estimate of the transformation between the last keyframe and current frame, the image intensity error that we seek to minimize for every image point is given as:

$$\mathrm{e}_i^{img}\Big({}^n\mathbf{q}_p\Big) = \boldsymbol{\mathcal{I}}_p\Big(\boldsymbol{\Pi}\big(\mathbf{P}_i\big)\Big) - \boldsymbol{\mathcal{I}}_n\Big(\boldsymbol{\Pi}\big({}^n\mathbf{R}_p\mathbf{P}_i + {}^n\mathbf{t}_p\big)\Big) \tag{2.5}$$

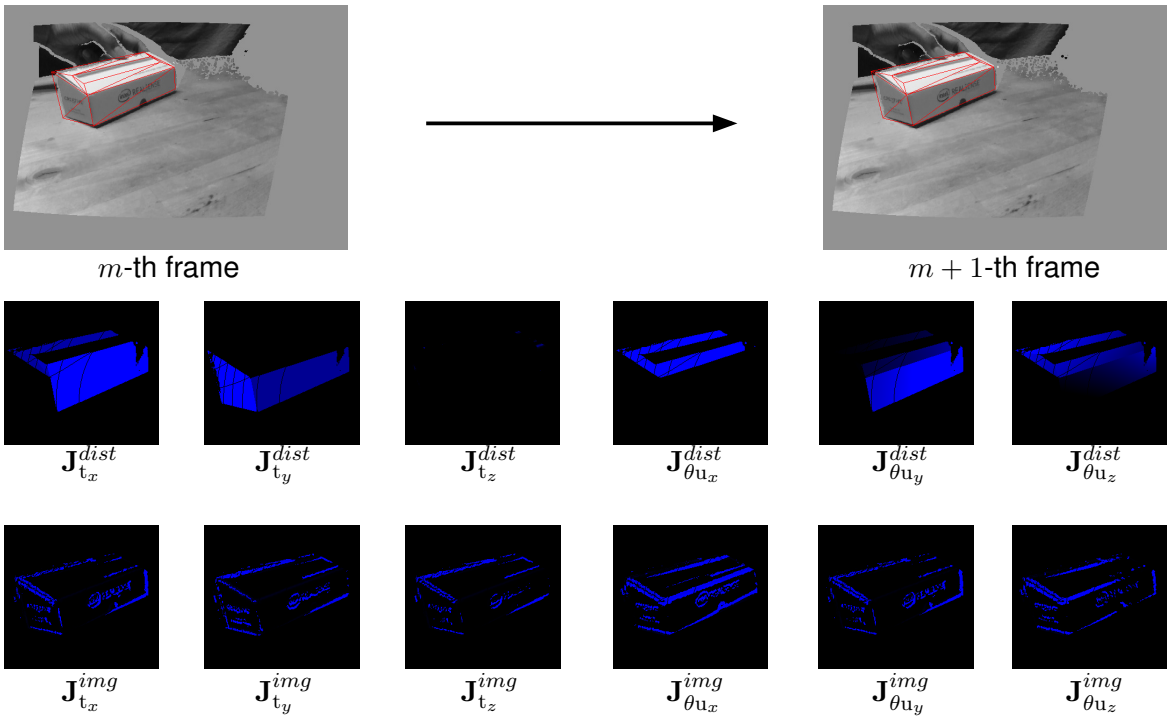


FIGURE 2.1 – Jacobian of depth and photometric error between two data frames m and $m+1$. $\mathbf{J}^{dist}$ and $\mathbf{J}^{img}$ are the gradient of the geometric and photometric error respectively, while the suffix $\mathrm{t}_x, \mathrm{t}_y, \mathrm{t}_z, \theta\mathrm{u}_x, \theta\mathrm{u}_y, \theta\mathrm{u}_z$ represents the component of the gradient along the 6 dimensions of the $(\mathbf{t}, \theta\mathbf{u})$ vector (the gradient intensities have been thresholded for visualization)

The Jacobian used for this minimization is:

$$\underset{[1\times 6]}{\mathbf{J}_i^{img}} = \underset{[1\times 2]}{\nabla \boldsymbol{\mathcal{I}}_i} \underbrace{\begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}}_{[2\times 2]} \underbrace{\begin{bmatrix} -\frac{1}{Z} & 0 & \frac{X}{Z^2} & \frac{XY}{Z^2} & -(1+\frac{X^2}{Z^2}) & \frac{Y}{Z} \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} & (1+\frac{Y^2}{Z^2}) & -\frac{XY}{Z^2} & -\frac{X}{Z} \end{bmatrix}}_{[2\times 6]} \tag{2.6}$$

where

$$\nabla \boldsymbol{\mathcal{I}}_i = \begin{bmatrix} \nabla \boldsymbol{\mathcal{I}}_{i,x} & \nabla \boldsymbol{\mathcal{I}}_{i,y} \end{bmatrix} \tag{2.7}$$

and f_x and f_y are the focal lengths. The value of the Jacobians expressed in eq. 2.3 and eq. 2.6 can be projected on the image plane and visualized, as shown in Fig. 2.1.

2.2.1.3 Optimization

Having computed the error and the Jacobian matrices for the two different types of error function, we now describe the method used for computing the pose update that

minimizes these two errors jointly. The optimization that we use is the Iteratively Re-weighted Least Squares (IRLS). The weighting function, as described below, allows the optimization to handle the presence of additional objects that occludes the object being trackes, as well as minor changes to illumination and the presence of sensor noise. The pose update is given by:

$$\boldsymbol{\xi} = -\lambda\Big(\mathbf{WJ}\Big)^{+}\mathbf{We} \tag{2.8}$$

where:

$$\mathbf{J} = \Big(\mathbf{J}_1^{dist}, \mathbf{J}_2^{dist}, \cdots, \mathbf{J}_1^{img}, \mathbf{J}_2^{img}, \cdots\Big) \tag{2.9}$$

and:

$$\mathrm{e} = \Big(\mathrm{e}_1^{dist}, \mathrm{e}_2^{dist}, \cdots, \gamma\mathrm{e}_1^{img}, \gamma\mathrm{e}_2^{img}, \cdots\Big) \tag{2.10}$$

are the stacked Jacobian matrices and error vectors respectively. $\boldsymbol{\xi}$ can be interpreted as the velocity screw acting on the camera that transforms it from data frame n-1 to n. The pose update is given as:

$${}^{n}\mathbf{T}_{n-1} = (\boldsymbol{\Delta}\mathbf{T})\,{}^{n}\hat{\mathbf{T}}_{n-1} \tag{2.11}$$

where $\boldsymbol{\Delta}\mathbf{T} = \exp(\boldsymbol{\xi})$ represents the transformation between the last estimate and the updated estimate of the pose of the pose of the current frame and ${}^{n}\hat{\mathbf{T}}_{n-1}$ represents the previous estimate of the transformation between the $(n-1)$-th frame and the current frame. Here :

$$\gamma = \frac{\overline{\mathrm{e}^{dist}}}{\overline{\mathrm{e}^{img}}} \tag{2.12}$$

is computed only once per frame, at the first iteration. γ serves as a scaling factor to ensure that the point-to-plane distance error and the photometric error are at a similar order of magnitude. $\mathbf{W}$ is a diagonal matrix of weights obtained from the m-estimator for being robust to outliers. We use Tukey biweight operator as the m-estimator [Malis and Marchand, 2006]. For any given residual $\mathbf{e}_i$, we define the Tukey operator as:

$$\rho(\mathbf{e}_i, \kappa) = \begin{cases} \left(\frac{\mathbf{e}_i^6}{6} + \frac{\kappa^2\mathbf{e}_i^4}{2} + \frac{\kappa^4\mathbf{e}_i^2}{2}\right) & \text{if } |\mathbf{e}_i| < \kappa \\ \frac{1}{6}\kappa^6 & \text{else} \end{cases} \tag{2.13}$$

where $\kappa = 4.7\hat{\sigma}$ and $\hat{\sigma} = \left\{1.48 \times \mathbf{Median}(|\mathbf{e}_i - \tilde{\mathbf{e}_j}|)\right\}$ and:

$$\mathbf{W} = \begin{bmatrix} \rho(\mathbf{e}_1^{dist}, \kappa) & & & & & & \\ & \rho(\mathbf{e}_2^{dist}, \kappa) & & & & & \\ & & \ddots & & & & \\ & & & \rho(\mathbf{e}_2^{img}, \kappa) & & & \\ & & & & \rho(\mathbf{e}_2^{img}, \kappa) & & \\ & & & & & \ddots & \end{bmatrix} \tag{2.14}$$

The error vectors, the Jacobian matrices and weight matrices are re-computed for every iteration of the optimization. At this point, it must be noted that the Jacobian of the two error vectors $\mathbf{J}^{dist}$ and $\mathbf{J}^{img}$ are both full rank, thereby making it unnecessary to prioritize one error minimization over the another.

2.2.2 Point Correspondences

As indicated in eq. (2.2), there is a need to associate every 3D point $\mathbf{P}_i$ with one of the planes of the model, which can be represented by the tuple $({}^{\mathrm{O}}\mathcal{V}_k, \mathbf{X}_k^{\mathcal{V}})$, the k-th plane of the model. At every frame, we know the value of ${}^{C_{n-1}}\mathbf{T}_O$. We project the 3D points ${}^{\mathrm{O}}\mathcal{V}_k$ for all visible planes into the image obtained in the current frame. This gives us a set of 2D polygons in the image, each representing one of the visible faces. All the points in the image (x_i, y_i) that intersects with a particular plane obtained from $\mathbf{\Pi}\left({}^{\mathrm{O}}\mathcal{V}_k\right)$ are considered to be associated with this plane. This method is demonstrated with a toy example in Fig. 2.2.

2.2.3 Selection of Keyframe

We use keyframes to reduce drift in frame-to-frame photometric tracking, as shown in Fig. 2.3. The estimate of the transformation between the last keyframe and the current frame is given by ${}^{C_n}\mathbf{T}_{C_k}$. We decompose this transformation matrix into the translational component (t_x, t_y, t_z) and the $\theta\mathbf{u}$ rotational component. We define a variable p_k, such that:

$$p_k = \begin{cases} 1 & \text{if } ||(t_x, t_y, t_z)|| > 0.05 \text{ or } \theta > 0.15 \\ 0 & \text{else} \end{cases} \tag{2.15}$$

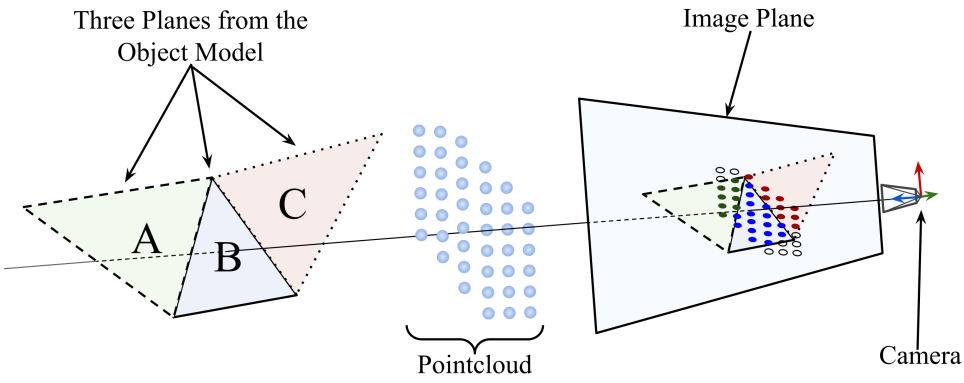


FIGURE 2.2 – A toy example demonstrating the point-plane correspondence methodology. The correspondence is decided entirely on the image plane, with the green, blue and red dots corresponding to the planes marked A, B and C

The distance threshold is expressed in millimetre while the angular threshold is given in radian. At the very first frame, we set $k = n = 0$, thereby providing an initialization for the keyframe.

2.3 Results

For benchmarking the performance of the tracker, we generated three short sequences of simulated RGB-D data, comprising of three objects: a *marmalade container*, a *coffee machine* and a *simulated car*. A very simple model is used to track these objects. The model is not required to be continuous or topologically closed. We end up with a coarse approximation of the shape of the object, comprising of 24, 48 and 62 faces for the three objects respectively. The tracker is initialized using the ViSP library [Marchand et al., 2005], by matching some keypoints detected in the very first image with those extracted in the training images using an approach similar to [Choi and Christensen, 2010].

Very few model-free tracking and reconstruction algorithms are publicly available along with their open source code (e.g: [Izadi et al., 2011], [Rünz and Agapito, 2017]), but it is not trivial and requires significant modifications to compare reconstruction and tracking methods with model-based object tracking using RGB-D data. Hence, we compare the proposed method with two approaches: **a)** 'edge + keypoint + depth

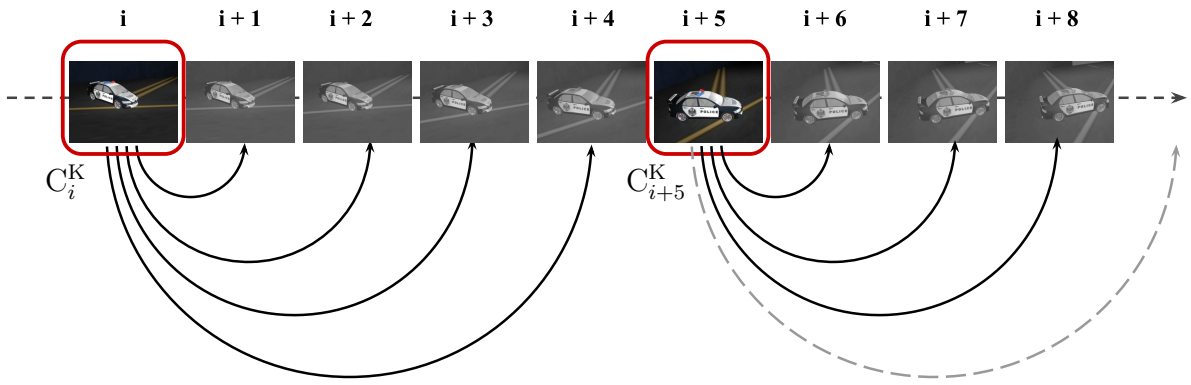


FIGURE 2.3 – The photometric tracking is based on a keyframe-to-frame error minimization. This figure shows an example of keyframes being generated after every 5-th frame. The exact criteria for generating a new keyframe is given in Eqn. 2.15

tracker'from [Trinh et al., 2018] (denoted in the figures as *ViSP*), and **b)** stacked error minimization of point-to-plane distance with photometric error, without using keyframes (denoted as *No KF*). The proposed approach is denoted as *PA* in the figures.

The quantitative comparison of the various tracking methods [1] with the ground-truth is summarized in Table 2.1. The *marmalade container* sequence (Fig. 2.4a) shows accurate tracking with all the approaches. The proposed approach manages to outperform the other two methods, although by a small margin. However, in the *coffee machine* sequence, the proposed approach outperforms *ViSP* significantly. As shown in frame 275 of Fig. 2.4b, *ViSP* shows a noticeable drift while tracking a set of coplanar faces with not enough image features in it. This is the only instance among all the sequences tested across all the available methods, where the tracking resulted in a visually noticeable drift. *No KF* alone, does not solve the issue of the drift completely, but the proposed approach eliminates the visible drift and the positional tracking is 83.09% better than that of *ViSP*. *Simulated car* is a bit more challenging sequence due to larger inter-frame motion towards the end. *No KF* shows very low accuracy in this sequence. In the rotation, *ViSP* outperforms the proposed approach by average of $0.51°$ over the entire sequence. This drift is not noticeable visually, and happens because a combination of a large number of robust feature points and edges makes it easier for *ViSP* to track the overall orientation of the car, while larger inter-frame motion disadvantages the photometric minimization. However, the proposed approach is more accurate than *ViSP* in terms of translation.

1. All results can be viewed at: youtu.be/_TPJkleBu3w

TABLE 2.1 – Summary of RMSE values

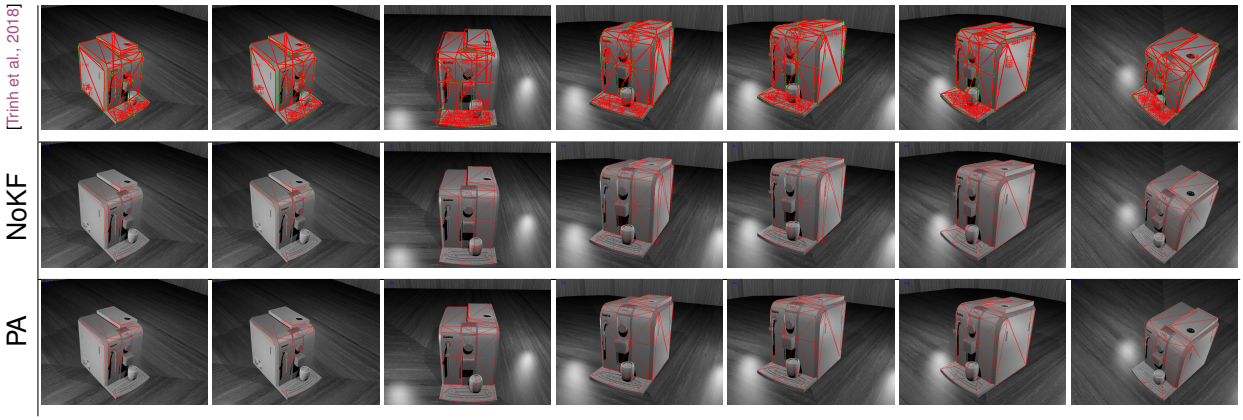
	Marmalade Container		Coffee Machine		Simulated Car	
	Translation	Rotation	Translation	Rotation	Translation	Rotation
No KF	0.045	0.139	0.284	0.032	0.265	0.072
[Trinh et al., 2018]	0.053	0.004	0.118	0.008	0.183	**0.027**
PA	**0.043**	**0.003**	**0.048**	**0.006**	**0.179**	0.036

Across the three sequences, it can be concluded that the proposed approach performs better than both *No KF* and *ViSP*. We show the tracking error plots of only *ViSP* and the proposed approach in Fig. 2.5a and Fig. 2.5b. Some additional results on real-world, complex shaped objects from the dataset of [Choi and Christensen, 2013] is shown in Fig. 2.7.

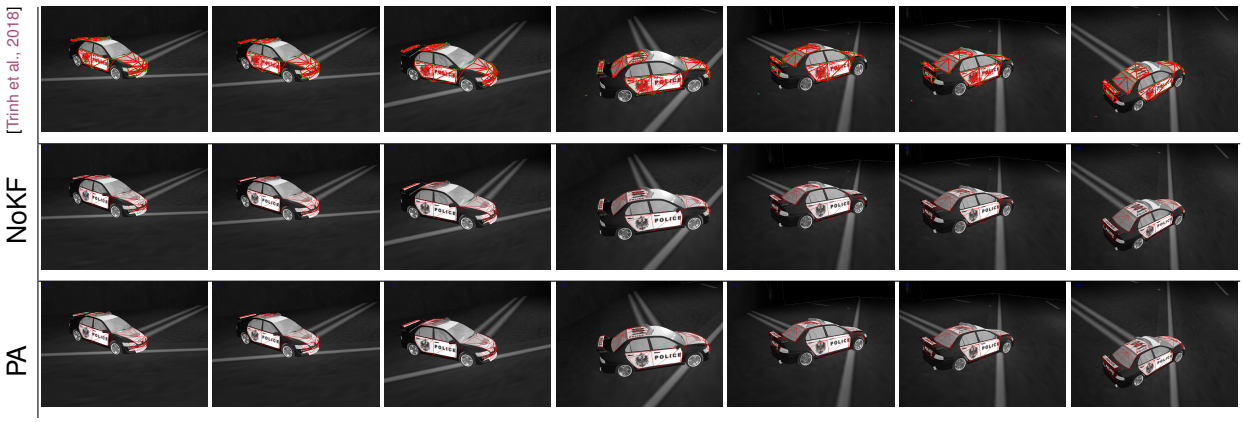
For the real objects, all the object models were constructed using manual measurements of the object. The data was captured using Intel RealSense SR300. The *banana* model is only a rough approximation of the shape of the real banana. Both the *box* and the *car* got accurately tracked (for visual validation, refer to Fig. 2.6), despite the obvious inaccuracies in the model. The proposed approach does not get affected by moderate occlusion of the objects by the hand. There were some slippage of the model from the actual object while tracking the *banana*. However, it never completely looses tracking.

The proposed approach has been implemented on C++ with occasional utilization of the PCL library [Rusu and Cousins, 2011] and the ViSP library for I/O and the IRLS implementation respectively. The simulated data has been generated using the Blender software [Blender, 2018].The proposed algorithm has been tested on an Intel Core i7-6600U CPU with 16 GB RAM. Running on a single core of the system, without SSE optimization and without using GPU in any format, the basic C++ code written for implementing the proposed approach achieves a runtime of 100 - 160 ms per frame, including data capture, tracking and display on a simple GUI. It can be envisaged that with either SSE optimization or with the use of GPU, the overall algorithm can run much faster, if required.

(a) *Marmalade Container*



(b) *Coffee Machine*



(c) *Simulated Car*

FIGURE 2.4 – Comparison of tracking results from 'ViSP Generic Tracker' [Trinh et al., 2018] (**ViSP** - first row), 'Point-to-plane + Photometry without Keyframes' (**No KF** - second row) and the 'Proposed Approach' (**PA** - third row)

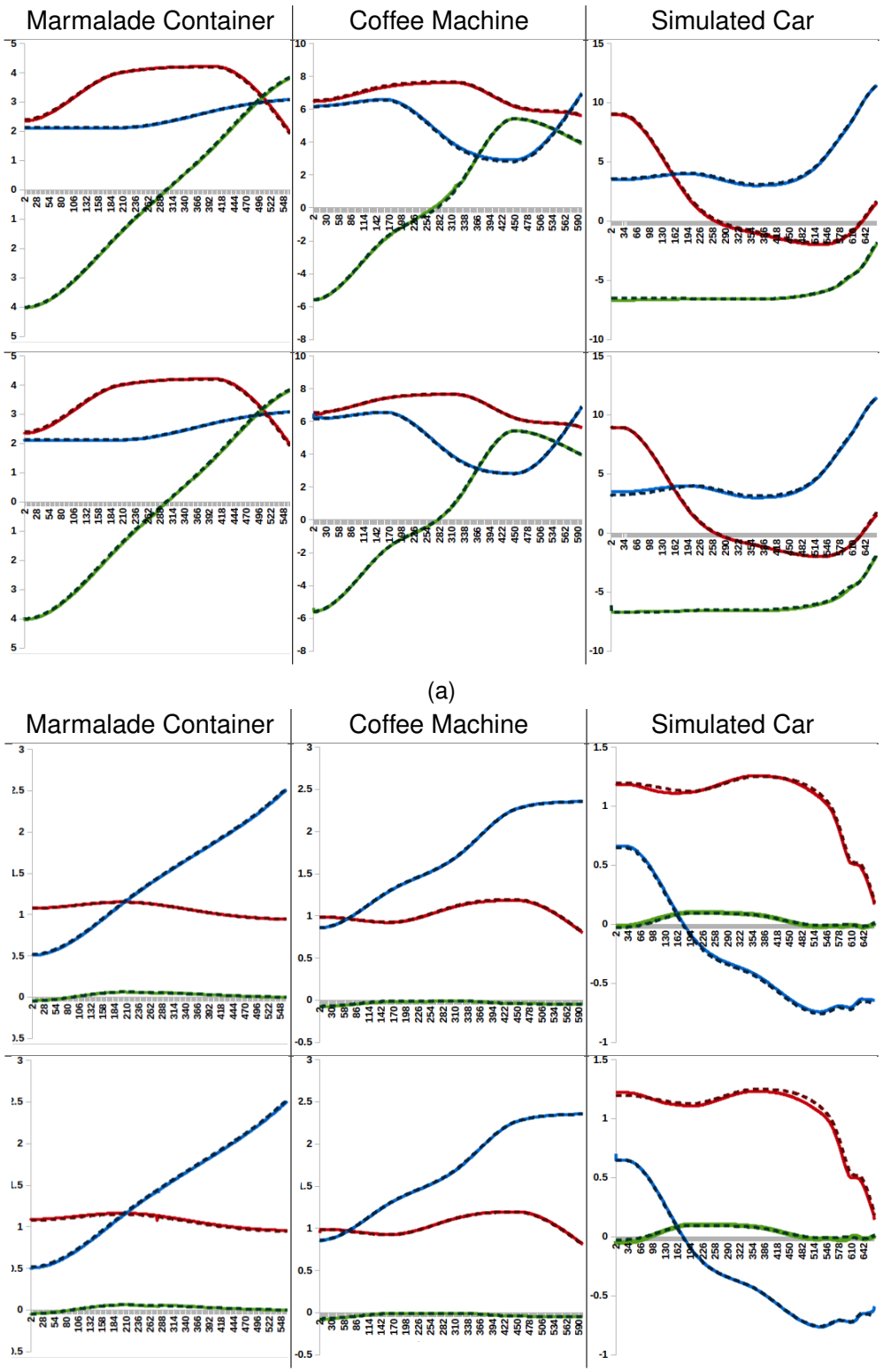


FIGURE 2.5 – Comparison of **a)** Translation along X, Y, Z axis, plotted against groundtruth (*GT*), and **b)** Rotation (in radian) along X, Y, Z axis, plotted against groundtruth (*GT*). X-axis shows the frame number, Y-axis represents the translation and rotation respectively

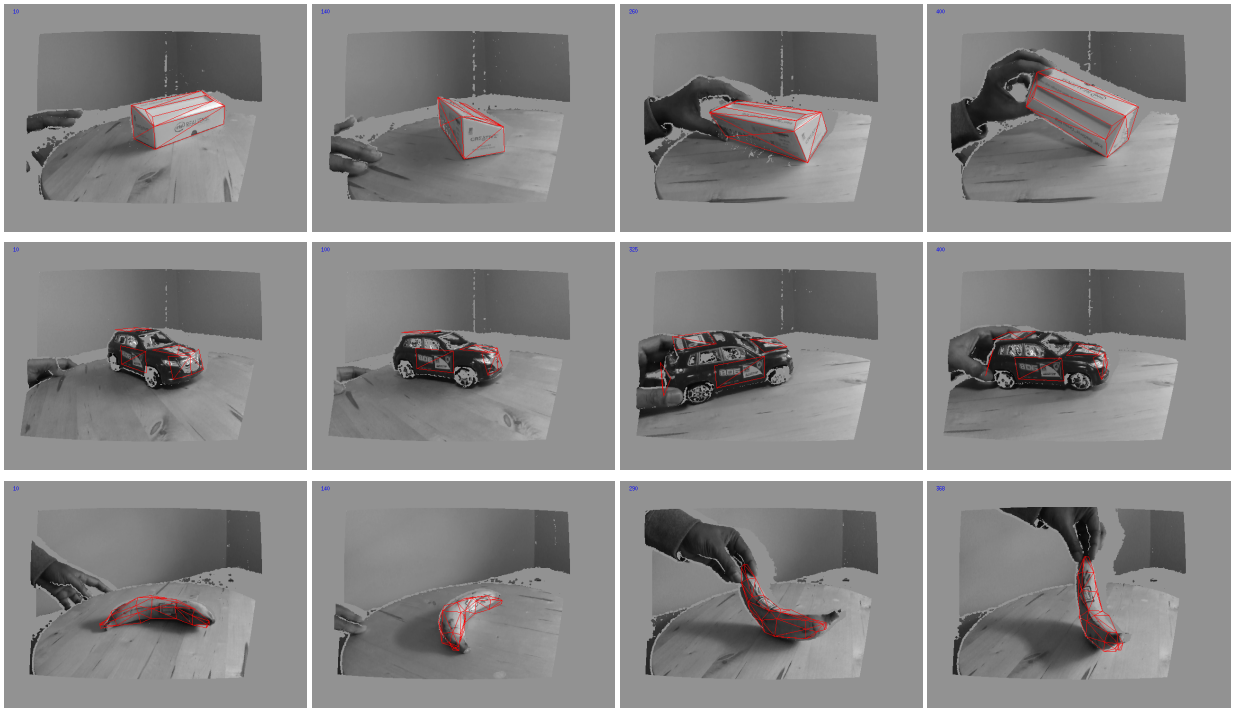


FIGURE 2.6 – Tracking results from real data captured using Intel RealSense for a) a box, b) a toy car, c) a banana

2.4 Conclusion

In this chapter, we present an algorithm to accurately track the pose of rigid and complex-shaped objects. The tracking is robust to occlusion and partial specularity of the scene. We provide validation on both simulated and real data. The proposed approach outperforms one of the best among the open-sourced, model-based, 6DoF object tracking methods. It also outperforms a partial re-implementation of a state-of-the-art tracking method from recent advances in the field of tracking and reconstruction. The proposed approach is an efficient method to track complex objects that a) does not require detailed object model (reducing the setup time in practical applications), b) tracks objects with better accuracy than comparable state-of-the-art approaches.

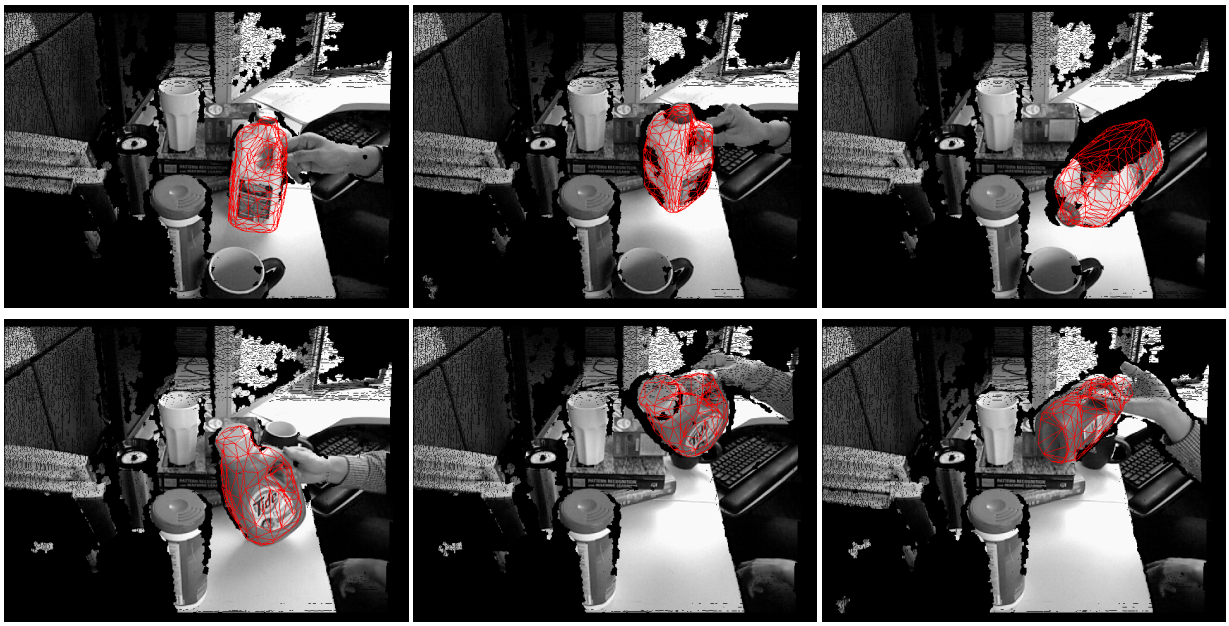
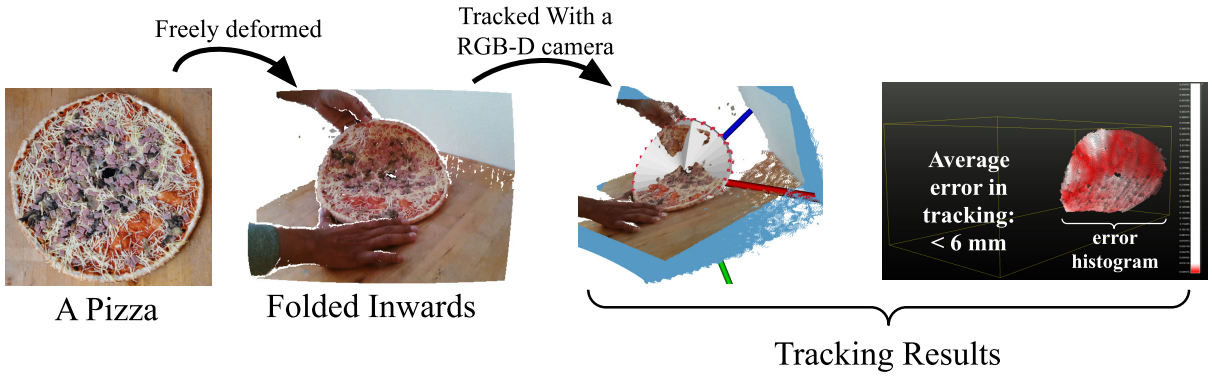


FIGURE 2.7 – Tracking results on the real data of the dataset from [Choi and Christensen, 2013]

CHAPTER 3

DEPTH BASED NON-RIGID OBJECT TRACKING



Demonstration of the method for deformation tracking proposed in this chapter, utilized for the tracking of the deforming surface of a *pizza*

In the previous chapter, we discussed a method to track rigid objects using a coarse 3D model. This is meant to serve as a prior to the non-rigid object tracking methodologies discussed in the rest of this thesis. In this chapter, we present a method to track deformable objects using co-rotational Finite Element Method (FEM), by controlling the force applied on the physical object model at certain nodes of the mesh.

We describe a method to handle the problem of accurate tracking of the surface of non-rigid objects undergoing deformation. A commodity-level RGB-D camera is used for sensing. It is assumed that we know the *visual-surface model* of the object (which can also be a CAD model), but this model does not have to be precise. The approach proposed in this chapter tracks the deforming objects by regulating virtual forces acting on the surface of the simulation of a physically-based mechanical mesh.

The primary contribution of this chapter is a method for tracking the entire visible surface of a deforming, non-rigid object in 3D, such that:

— it requires only a very coarse estimate of the physical properties (Young's modulus and Poisson ratio) of the object ;

— it performs accurate frame-to-frame tracking of deformation of the non-rigid object ;

— it has been validated on simulated data with ground truth, as well as real data.

3.1 Background

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

Having decided to utilize physically based models for deformation tracking, in this chapter we begin the study of tracking approaches based on depth information only. Since depth data is an important information available from the RGB-D sensors and can be conveniently utilized for tracking the temporal evolution of non-rigid objects, we design a methodology for tracking this depth data using the co-rotational FEM model. However, such approaches are not completely unprecedented.

As discussed in Sec. 1.3.1.1, physics based framework for deformation tracking has been demonstrated using mass-spring-damper systems [Royer et al., 2017], finite element model (FEM) [Petit et al., 2015a] and kinematic chains [Schmidt et al., 2014]. For model based tracking of non-rigid objects, the representation of the model is usually done by a set of triangular faces representing the surface of the object. However, tetrahedral, volumetric model is preferred [Paloc et al., 2002] for tracking using mass-spring-damper systems, while the FEM based systems available in the literature

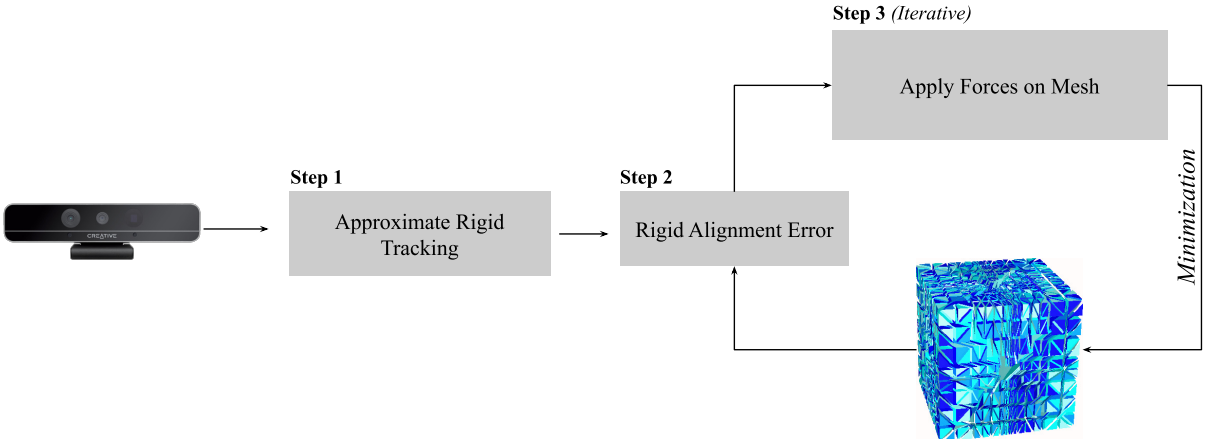


FIGURE 3.1 – Overview of the approach for deformation tracking proposed in this chapter

typically preferred to use a combination of surface model and volumetric model. Haouchine *et. al.* [Haouchine et al., 2015] proposed a linear tetrahedral co-rotational FEM based model for tracking large deformations. [Petit et al., 2015a] [Petit et al., 2018] use co-rotation FEM based model to track deformation by estimating the direction and magnitude of elastic force acting on the object using depth information. As summarized in [Nadon et al., 2018], many FEM based approaches in the available literature suffer from excessive dependency on the availability of the accurate physical properties of the object being tracked.

The method that we propose in this chapter removes this dependency using a novel, closed loop minimization technique. The same lack of dependency on the accuracy of the physical parameters of the model is also applicable to [Petit et al., 2015a]. However, as demonstrated in Sec. 3.4, the approach proposed here outperforms the underlying basic framework of [Petit et al., 2015a] under practical constraints. Moreover, a closed-loop error minimization improves robustness to occlusion and noise.

3.2 Method

The methodology that we use for tracking deformable objects will be described in this section. We begin by describing the preliminary notations and the elastic deformation model that we use for interpreting the physics behind the objects we track. This is followed by a brief description of the rigid registration process (Sec. 3.2.3). The description of matching the deformable object between consecutive frames is discussed

next (Sec. 3.2.4). We follow this up with the mechanism for computing the Jacobian that links the variation in the geometric error with the variation of force applied on a particular vertex. We use this Jacobian for minimizing the error using an Iteratively Re-weighted Least Square (IRLS) formulation. We finish this section by summarizing the steps for minimizing the non-rigid error.

In the approach proposed here, first the deforming object is tracked rigidly using its 3D object model. After the rigid tracking, the remaining residual error occurs purely due to the deformation of the model. In the second step, this non-rigid error is minimized by applying a deforming force on the physical object model. The method for obtaining this deforming force that minimizes the non-rigid error is the primary contribution of this chapter and is described in Sec. 3.2.4. The overview of this approach is described in Fig. 3.1.

3.2.1 Notations

We use two types of 3D model for the non-rigid tracking. The first is a *visual-surface model*, denoted by $\mathcal{V}^{\mathrm{S}}$, and the other one is a tetrahedral, volumetric, *internal model*, denoted by $\mathcal{V}^{\mathrm{M}}$. Continuing from the definition in Sec. 1.1.3, ${}^{C}\mathcal{V}^{\mathrm{S}}$ and ${}^{O}\mathcal{V}^{\mathrm{S}}$ denotes the *visual-surface model* in the camera ($\mathcal{F}_c$) and object frame ($\mathcal{F}_o$) respectively while ${}^{C}\mathcal{V}^{\mathrm{M}}$ and ${}^{O}\mathcal{V}^{\mathrm{M}}$ denotes the *internal model* in a similar manner. The *visual-surface model* is described using a set of planes which are represented using the nodes defining the boundary of the surface (usually a triangle) and the connectivity between these nodes. We can represent this surface as

$$\mathcal{V}^{\mathrm{S}} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{\mathrm{M}} \end{bmatrix} \tag{3.1}$$

where M is the number of vertices in $\mathcal{V}^{\mathrm{S}}$ and the i-th vertex is $\mathbf{p}_i = \begin{pmatrix} X_i, Y_i, Z_i \end{pmatrix}$, the 3D coordinate of a point in the reference frame of the base model. As given in Sec. 1.1.3, we have a corresponding, connectivity map, given by $\mathbf{X}^{\mathcal{V}^{\mathrm{S}}}$. M is the number of vertices and N is the number of triangular faces in $\mathcal{V}^{\mathrm{S}}$. The 3D coordinates of the three vertices of the triangle representing a given face can be derived from $\mathbf{X}^{\mathcal{V}^{\mathrm{S}}}$. Clearly, the normal vector $\mathbf{n}_i = (n_i^X, n_i^Y, n_i^Z)$ and distance to origin d_i for any given face can be derived from $\mathcal{V}^{\mathrm{S}}$ and $\mathbf{X}^{\mathcal{V}^{\mathrm{S}}}$ using elementary geometry. We similarly define a tetrahedral, volumetric, internal model $\mathcal{V}^{\mathrm{M}}$ along with its connectivity map $\mathbf{X}^{\mathcal{V}^{\mathrm{M}}}$, such that every element of $\mathbf{X}^{\mathcal{V}^{\mathrm{M}}}$ has four indices, instead of three. As shown in Fig. 3.2, $\mathcal{V}^{\mathrm{M}}$ represents

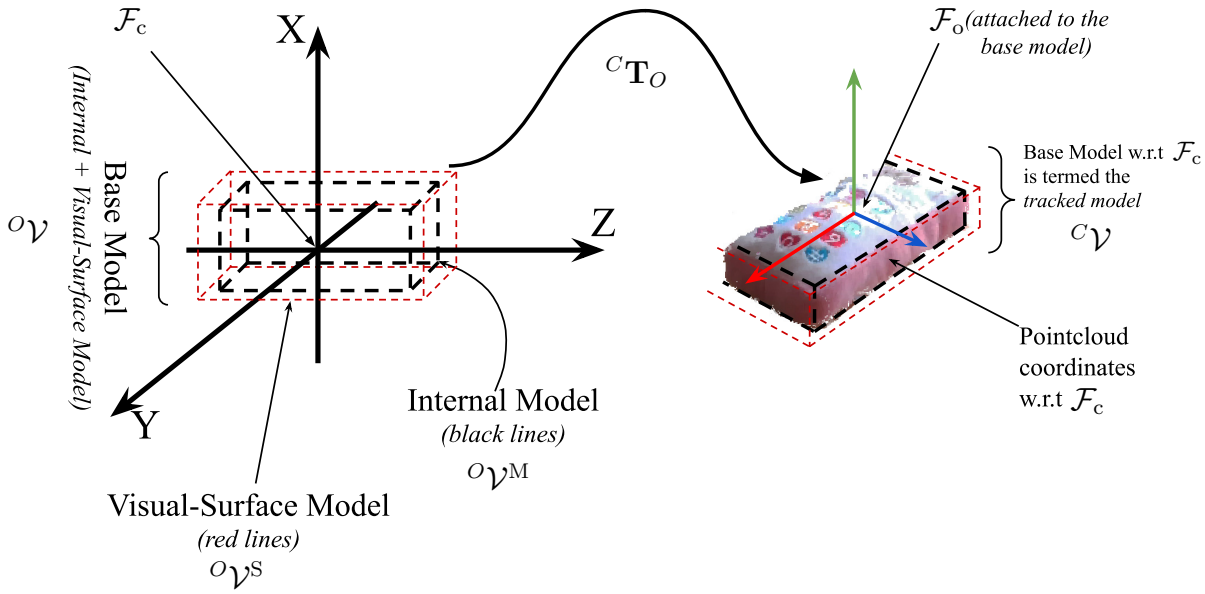


FIGURE 3.2 – The depth camera is centered at $\mathcal{F}_C$ while the *tracked model* is used for the rigid registration of the *base model* with the pointcloud. $\mathcal{F}_c = \mathcal{F}_o$ at the beginning of the sequence, when ${}^C\mathbf{T}_O$ has not been initialized

the internal, mechanical model (*internal model*) while $\mathcal{V}^S$ represents the surface model (*visual-surface model*) of the object. The *visual-surface model* and the *internal model* are maintained at the same reference frame and together and we refer to them as the *base model* and is denoted by the set $\{{}^O\mathcal{V}, \mathbf{X}^{\mathcal{V}}\}$, where ${}^O\mathcal{V} = \begin{pmatrix} {}^O\mathcal{V}^S & {}^O\mathcal{V}^M \end{pmatrix}$ and $\mathbf{X}^{\mathcal{V}} = \{\mathbf{X}^{\mathcal{V}^S}, \mathbf{X}^{\mathcal{V}^M}\}$.

The reference frame used for this setup of non-rigid tracking is given in Fig. 3.2. ${}^C\mathbf{T}_O$ denotes the transformation from the camera-centered coordinate frame $\mathcal{F}_C$ to the object in the pointcloud. For rigid tracking, we define a *tracked model* using the tuple $\{{}^C\mathcal{V}, \mathbf{X}^{\mathcal{V}}\}$, such that:

$$ {}^O\mathcal{V} = {}^O\mathbf{R}_C {}^C\mathcal{V} + \underset{3\times M}{\begin{pmatrix} {}^O\mathbf{t}_C & \cdots & {}^O\mathbf{t}_C \end{pmatrix}} \tag{3.2} $$

During non-rigid deformation of the *base model* (${}^O\mathcal{V}$) centered at $\mathcal{F}_C$, the transformation of (3.2) is repeated, so that the model at the object centered reference frame always stays updated according to the deformation.

3.2.2 Deformation Modelling

We use co-rotational FEM as the deformation model, for developing the deformation tracking method proposed in this chapter. As explained in Sec. 1.3.1, co-rotational FEM offers us a method to model large deformations efficiently.

We can now briefly recall some of the relevant concepts used for simulating the deformation using FEM. For a tetrahedral mesh, we can represent the internal elastic forces acting on the nodes by:

$$\mathbf{F}_e = \mathbf{R}_e \mathbf{K}_e \tilde{\mathbf{U}}^R \tag{3.3}$$

Here, $\tilde{\mathbf{U}}^R = (\tilde{\mathbf{p}_1}, \tilde{\mathbf{p}_2}, \tilde{\mathbf{p}_3}, \tilde{\mathbf{p}_4})$ is obtained using Eqn. 3.3, $\underset{12\times12}{\mathbf{K}_e}$ is the stiffness matrix and $\underset{12\times12}{\mathbf{R}_e}$ is the block diagonal matrix of four $\mathbf{R}$ rotation matrices stacked diagonally. $\underset{12\times1}{\mathbf{F}_e}$ gives the elastic forces acting on the nodes of this tetrahedral element.

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

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}_g\mathbf{p} = \mathbf{F}_{ext} + \mathbf{F}_e \tag{3.4}$$

where $\mathbf{M}$ and $\mathbf{D}$ are the model's mass and damping matrices respectively, derived using Eqn. 1.66 and Eqn. 1.67. $\mathbf{K}_g$ is the global stiffness matrix. $\mathbf{F}_{ext}$ are the external forces acting on the vertices. The method for determining $\mathbf{F}_{ext}$ is discussed in the subsequent sections. To solve this differential equation, we use a linear solver based on conjugate gradient descent [Faure et al., 2012b]. To impose additional constraint, Eqn. (3.4) can be multiplied using a projection matrix that sets the values of certain indices to zero. We use the projective constraint to eliminate the rigid motion of the object. This is discussed in the next section

It must be noted that the force $\mathbf{F}_{ext}$ is just the deforming force acting on the model, and does not include the forces causing the rigid transformation of the body. In fact, the deformation model proposed here is completely independent of rigid motion that results from the effects of gravity and interaction with other contact surfaces. As described in Sec. 3.2.3, the rigid motion is tracked separately. This separation of rigid and non-rigid tracking method, along with the minimization described Sec. 3.2.4, makes the overall system independent from the inaccuracies of the physical parameters.

3.2.3 Rigid Registration

The rigid registration is a joint minimization of two error terms: depth based geometric error and keypoint based feature tracking [Trinh et al., 2018]. The two error terms are explained below:

3.2.3.1 Depth based geometric error

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

$$\mathrm{e}^{\mathrm{D}}({}^{n}\mathbf{q}_{n-1}) = \left(\left({}^{n}\mathbf{R}_{n-1}\mathbf{P} + {}^{n}\mathbf{t}_{n-1}\right) \cdot \mathbf{n}_{k}\right) - d_{k} \tag{3.5}$$

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

$$\mathbf{J}_{i}^{\mathrm{D}} = \begin{bmatrix} \mathbf{n}_{k}^{\top} & \begin{bmatrix}\mathbf{n}_{k}\end{bmatrix}_{\times}\mathbf{P}_{i}{}^{\top} \end{bmatrix} \tag{3.6}$$

3.2.3.2 Feature based minimization

The Harris corner features are used for tracking with keypoints. This is based on the classical KLT algorithm [Baker and Matthews, 2004]. Let $\mathbf{u} = (x, y, 1)$ be the homogeneous 2D coordinate of a feature point in the $(n-1)$-th image and $\mathbf{u}^{*} = (x^{*}, y^{*}, 1)$ be the matched coordinate for the same point in the n-th frame. We define the error term as:

$$\mathrm{e}^{\mathrm{K}}({}^{n}\mathbf{q}_{n-1}) = \begin{pmatrix} \mathrm{x}({}^{n}\mathbf{q}_{n-1}) - x^{*} \\ \mathrm{y}({}^{n}\mathbf{q}_{n-1}) - y^{*} \end{pmatrix} \tag{3.7}$$

where

$$(\mathrm{x}({}^{n}\mathbf{q}_{n-1}), \mathrm{y}({}^{n}\mathbf{q}_{n-1}), 1) = {}^{n}\mathbf{H}_{n-1}\overline{\mathbf{u}} \tag{3.8}$$

Here, $\overline{\mathbf{u}} = (x, y, 1)$ and $\overline{\mathbf{u}}^{*} = (x^{*}, y^{*}, 1)$ are the homogeneous pixel coordinates of the tracked feature point in the $(n-1)$-th and n-th data frame respectively. Moreover:

$${}^{n}\mathbf{H}_{n-1} = {}^{n}\mathbf{R}_{n-1} + \frac{{}^{n}\mathbf{t}_{n-1}}{d}\mathbf{n}^{\top} \tag{3.9}$$

where $\mathbf{n}$ and d are interpreted in the same way as (3.5). The corresponding Jacobian that relates the variation of $\mathrm{e}^{\mathrm{K}}({}^{n}\mathbf{q}_{n-1})$ with the time variation of ${}^{n}\mathbf{q}_{n-1}$ is given by:

$$\mathbf{J}^{\mathrm{K}} = \begin{bmatrix} -\frac{1}{\mathrm{Z}} & 0 & \frac{x}{\mathrm{Z}} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{\mathrm{Z}} & \frac{y}{\mathrm{Z}} & (1+y^2) & -xy & -x \end{bmatrix} \tag{3.10}$$

where (x, y) are the pixel coordinates and Z is in meters. Given these two errors and their corresponding Jacobian, the combined error $\mathrm{e}({}^{n}\mathbf{q}_{n-1})$ is obtained by stacking the vectors $\mathrm{e}^{\mathrm{D}}({}^{n}\mathbf{q}_{n-1})$ and $\mathrm{e}^{\mathrm{K}}({}^{n}\mathbf{q}_{n-1})$, while the combined Jacobian $\mathbf{J}$ is given by $\mathbf{J} = (\mathbf{J}^{\mathrm{D}}, \mathbf{J}^{\mathrm{K}})$. The combined error is minimized with the update given by:

$$\boldsymbol{\xi} = -\lambda\Big(\mathbf{W}\mathbf{J}\Big)^{+}\mathbf{W}\mathbf{e} \tag{3.11}$$

where $\mathbf{W}$ is the weight matrix for outlier removal using the Tukey m-estimator [Beaton and Tukey, 1974], and $\boldsymbol{\xi} \in \mathfrak{se}(3)$. This becomes an iteratively re-weighted least squares problem.

3.2.4 Non Rigid Tracking

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

At the end of the rigid registration process, we can construct a map that links the points in the 3D pointcloud to their corresponding geometric error from (3.5). Given that $\mathbf{e}_i^{\mathrm{D}}$ denotes the error for the i-th point $\mathbf{P}_i$ in the entire pointcloud $\mathbf{P}$, it is possible to derive a new pointcloud $\mathbf{P}^*$ using a linear thresholding operation, such that:

$$\mathbf{P}^* = \{\mathbf{P}_i \in \mathbf{P} \Big| |\mathbf{e}_i^{\mathrm{D}}| \geq \theta^{\mathrm{D}}\} \tag{3.12}$$

where θ^{D} is a threshold that depends on the geometry of the object being tracked. We subject the pointcloud $\mathbf{P}^*$ to a clustering step using Eucledian distances [Hartigan

and Wong, 1979]. Very small clusters and clusters spanning more than half of the size of the entire pointcloud are discarded. Let us assume that we obtain j clusters from this operation, denoted by $\mathbf{K}_1, \mathbf{K}_2, \cdots, \mathbf{K}_j$, and their corresponding centroids are represented by $\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_j$, where $\mathbf{k} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1)$ is the homogeneous 3D coordinate of the point w.r.t $\mathcal{F}_C$. These points are inverse transformed with the last estimate of ${}^{C}\mathbf{T}_{O}$ obtained after minimization with (3.11), such that

$$\mathbf{k}_i^* = \left({}^{C}\mathbf{T}_{O}\right)^{-1} \mathbf{k}_i \tag{3.13}$$

For all these j centroids, we determine their nearest neighbors in ${}^{C}\mathcal{V}^{\mathrm{M}}$ using the k-nearest neighbor algorithm [Fukunage and Narendra, 1975]. Let the nearest neighbors of $\mathbf{k}_1^*, \mathbf{k}_2^*, \cdots, \mathbf{k}_j^*$ in ${}^{C}\mathcal{V}^{\mathrm{M}}$ be denoted by ${}^{C}\mathcal{V}^{\mathrm{M}}_{k1}, {}^{C}\mathcal{V}^{\mathrm{M}}_{k2}, \cdots, {}^{C}\mathcal{V}^{\mathrm{M}}_{kj}$.

For the following discussion, let us consider an arbitrary k_i-th cluster alone, which we refer to as ${}^{C}\mathcal{V}^{\mathrm{M}}_{i}$ for the sake of simplicity. At this stage, we must modify (3.5) to accommodate a slightly different variant of the same error function. We re-define it as $\mathrm{e}^{\mathrm{D}}\left({}^{C}\mathcal{V}^{\mathrm{S}}_{i}\right)$, assuming the r-th plane corresponds to the s-th point in the pointcloud, and the normal $\mathbf{n}_r$ and the distance to origin d_r is derived from ${}^{C}\mathcal{V}^{\mathrm{S}}_{i}$. This change in notation is necessary because the error term e^{D} is no longer a function of ${}^{n}\mathbf{q}_{n-1}$ at this point of non-rigid tracking, it is rather a function of ${}^{C}\mathcal{V}^{\mathrm{S}}_{i}$, i.e., the position of the surface model of the object w.r.t the s-th point in the pointcloud. The correspondence between the point and the plane is done using a strategy similar to Sec. 2.2.2. The propagation of the vertex displacement from ${}^{C}\mathcal{V}^{\mathrm{M}}_{i}$ to ${}^{C}\mathcal{V}^{\mathrm{S}}_{i}$ happens through the barycentric mapping $\mathscr{J}$ (defined in Sec. 1.3.1).

3.2.4.1 Jacobian Computation

The Jacobian that relates the variation of $\mathrm{e}^{\mathrm{D}}\left({}^{C}\mathcal{V}^{\mathrm{S}}_{i}\right)$ with the variation of the applied force is computed numerically by perturbing the node ${}^{O}\mathcal{V}^{\mathrm{M}}_{i}$ by a very small, constant force $\Delta\mathbf{F}_{\mathrm{J}}$ successively along the three axes, as shown in Fig. 3.3. After applying the force on the node, a simulation based on FEM is done using the minimization described in Sec. 3.2.2 - Eqn. 3.4. After the conjugate gradient solver optimizes (3.4), the system will attain a static condition. There will be six such configurations of the mesh, obtained by the perturbation along the positive and the negative direction of the three axes. We denote these new six configurations of ${}^{O}\mathcal{V}^{\mathrm{M}}_{i}$ by using the axis direction $\mathrm{X}+, \mathrm{X}-, \cdots, \mathrm{Z}-$

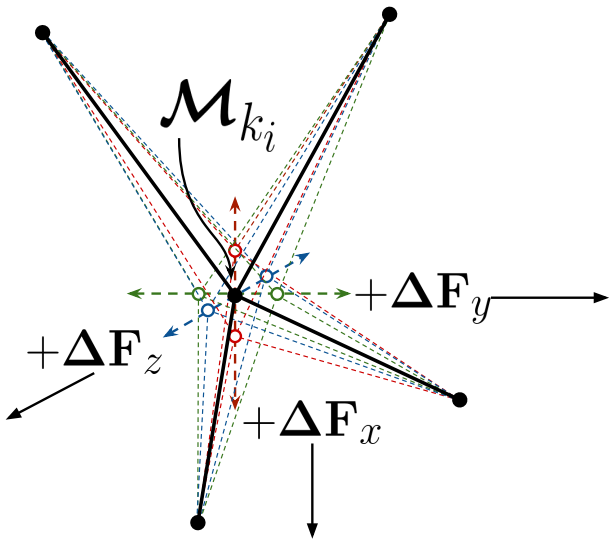


FIGURE 3.3 – The node closest to the centroid of a cluster is perturbed by a small force along the three axes in both positive and negative direction, producing six deformed configurations per node

as a superscript, given as:

$$({}^{O}\mathcal{V}_i^{\mathrm{M}})^{\mathrm{F}} = \left[({}^{O}\mathcal{V}_i^{\mathrm{M}})^{X+} \quad ({}^{O}\mathcal{V}_i^{\mathrm{M}})^{X-} \quad ({}^{O}\mathcal{V}_i^{\mathrm{M}})^{Y+} \quad ({}^{O}\mathcal{V}_i^{\mathrm{M}})^{Y-} \quad ({}^{O}\mathcal{V}_i^{\mathrm{M}})^{Z+} \quad ({}^{O}\mathcal{V}_i^{\mathrm{M}})^{Z-}\right] \tag{3.14}$$

Following the definition of Sec. 1.3.1, the deformed meshes from Eqn. 3.14 is propagated down to the *visual-surface model* via the barycentric map:

$$({}^{O}\mathcal{V}_i^{\mathrm{S}})^{\mathrm{F}} = \mathscr{J}\left(({}^{O}\mathcal{V}_i^{\mathrm{M}})^{\mathrm{F}}\right) \tag{3.15}$$

We take these modified, visual mesh and transform it back to the registered pointcloud using:

$$({}^{C}\mathcal{V}_i^{\mathrm{S}})^{\mathrm{F}} = {}^{C}\mathbf{R}_O\left(({}^{O}\mathcal{V}_i^{\mathrm{S}})^{\mathrm{F}}\right) + \underbrace{\left[{}^{C}\mathbf{t}_O \quad \cdots \quad {}^{C}\mathbf{t}_O\right]}_{3\times 6\mathrm{M}} \tag{3.16}$$

where M is the number of vertices of $\mathcal{V}^{\mathrm{S}}$. The relation of the variation of the external force with the variation in error can be expressed as:

$$\dot{\mathrm{e}}^{\mathrm{D}}\left({}^{C}\mathcal{V}_i^{\mathrm{S}}\right) = \mathbf{J}_i\dot{\mathbf{F}}_{ext} \tag{3.17}$$

The term $\mathbf{J}_i$ is obtained numerically by finite difference computation using central differences. The final Jacobian used is:

$$\mathbf{J}_i = \begin{pmatrix} \mathbf{J}_i^X & \mathbf{J}_i^Y & \mathbf{J}_i^Z \end{pmatrix} \tag{3.18}$$

$$\Rightarrow \mathbf{J}_i = \begin{bmatrix} \frac{({}^O\mathcal{V}_i^{\mathrm{M}})^{X+} - ({}^O\mathcal{V}_i^{\mathrm{M}})^{X-}}{2\mathbf{F}_{\mathrm{J}}} & \frac{({}^O\mathcal{V}_i^{\mathrm{M}})^{Y+} - ({}^O\mathcal{V}_i^{\mathrm{M}})^{Y-}}{2\mathbf{F}_{\mathrm{J}}} & \frac{({}^O\mathcal{V}_i^{\mathrm{M}})^{Z+} - ({}^O\mathcal{V}_i^{\mathrm{M}})^{Z-}}{2\mathbf{F}_{\mathrm{J}}} \end{bmatrix} \tag{3.19}$$

3.2.4.2 Minimization

So far, j nodes have been selected for application of external force to deform the model. We have p points in the pointcloud, and their correspondence with the model is known. If the iteration of minimization is denoted by n, the initial estimate of the vertically stacked force vector at the $(n-1)$-th iteration is denoted by $\underbrace{\mathbf{F}_{ext}^{n-1}}_{3j\times 1}$ and the values are set to zero. The Jacobian matrices are stacked up horizontally, such that:

$$\underbrace{\mathbf{J}}_{p\times 3j} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{J}_2 & \cdots & \mathbf{J}_j \end{pmatrix} \tag{3.20}$$

The update is computed as:

$$\Delta\mathbf{F} = -\lambda\Big(\underbrace{\mathbf{W}}_{p\times p}\mathbf{J}\Big)^{+}\mathbf{W}\underbrace{\mathrm{e}^{\mathrm{D}}\Big({}^C\mathcal{V}_i^{\mathrm{S}}\Big)}_{p\times 1} \tag{3.21}$$

$\mathbf{W}$ is a weighting matrix obtained from the Tukey based m-estimator, similar to Sec. 3.2.3. λ is a scaling factor. The force vector is updated by:

$$\mathbf{F}_{ext}^{n} = \mathbf{F}_{ext}^{n-1} + \Delta\mathbf{F} \tag{3.22}$$

This force is applied on the *base model* and the final node displacements are determined, once again, using (3.4).

3.3 Implementation

In order to compute the cost function, it is necessary to associate every 3D point $\mathbf{P}_i$ with one of the planar surfaces of the *visual-surface model* ${}^O\mathcal{V}^{\mathrm{S}}$. Given that we know the

value of ${}^{C}\mathbf{T}_{O}$ obtained at the previous frame, we project the 3D points ${}^{O}\mathcal{V}^{\mathrm{S}}$ for all visible planes into the image obtained in the current frame. The visibility is checked using the classical ray-casting algorithm [Roth, 1982]. The ray-casting is also needed for imposing the projective constraints. The nodes invisible to the camera are considered to be immobile under the influence of external force. Moreover, we do not consider the volumetric, internal mesh to be an input to the system. The mesh is rather generated apriori, using Dirichlet tessellation, followed by Delaunay tetrahedralization of the input mesh using the Bowyer-Watson algorithm [Si, 2015].

In contrast to [Petit et al., 2015a] and [Petit et al., 2018], the approach proposed here does not require a visual segmentation for separating the region of interest from the background. This is done by initializing the pose of the object at the first frame using pre-trained markers on the object. It is done using the ViSP library [Marchand et al., 2005], by matching the keypoints detected in the very first image with those extracted in the training images using an approach similar to [Dementhon and Davis, 1995].

The algorithm is implemented using a parallel framework, where two different processes are involved for the rigid and non-rigid tracking of the object respectively, as shown in Figure 3.4. The rigid tracking process is capable of processing each frame at < 100 ms, thereby ensuring real-time interaction. The un-optimized, non-rigid tracking code runs at 800ms to 1.3s (approximately) per frame, depending on the size of the object and its proximity to the camera. The results reported here have been achieved using processes running on a single core of a i7-6600U CPU with 16GB of RAM. An Intel Skylake GT2 GPU has been utilized, but only for the ray-casting. The FEM solvers are implemented using the SOFA library [Faure et al., 2012b], without using CUDA. The rigid tracking process runs at > 10 fps, while the non-rigid tracking process handles every 10-th/12-th frame coming from the sensor.

3.4 Results

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

1. Please visit the following URL for all results on real data: https://youtu.be/RFd-Ix9hcdg

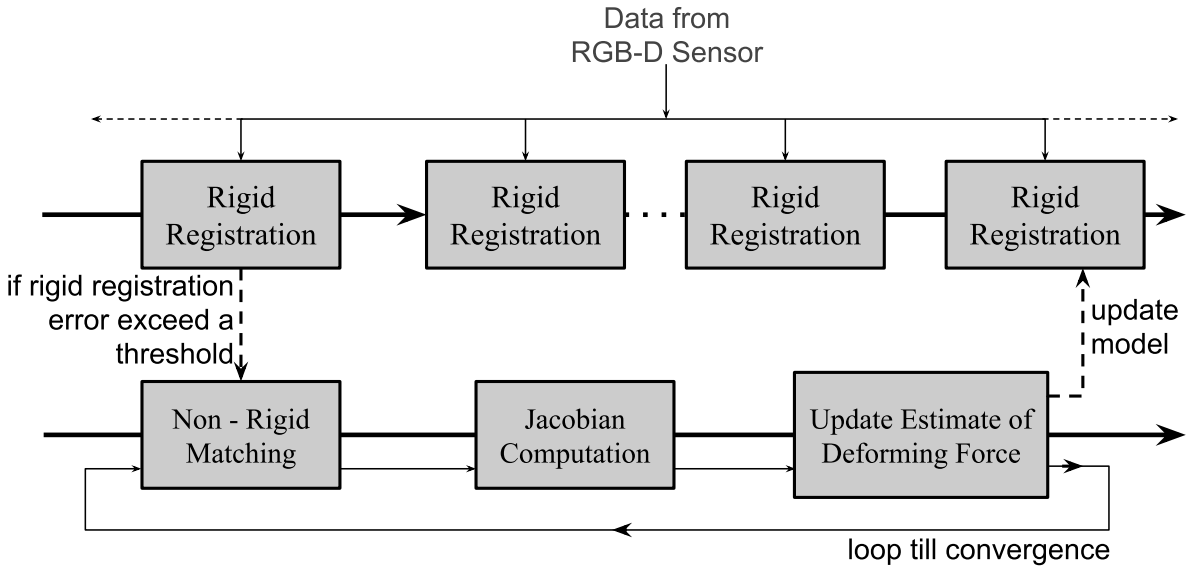


FIGURE 3.4 – The parallel implementation of the rigid and non-rigid tracking processes enable us to track the deforming surface at frame rate

3.4.1 Simulation

For the simulated data, the deformation of the objects were generated using simulation of co-rotational FEM. Two objects are considered, a *cube* and a rectangular *board*. They are made to undergo simple but large deformations, as shown in Fig. 3.5. We made the dataset available at: github.com/lagadic/nr-dataset. The visual models produced as an output of the simulation were subjected to texturing, shading and rendering using the Blender software, followed by the generation of the pointcloud using a RGB-D camera simulator. In the simulation, both the objects were modelled using Young's modulus (YM) of 50000 Pa and Poisson's ratio of 0.3. Rayleigh mass was assumed to be 0.1 and Rayleigh stiffness to be 0.3.

3.4.1.1 Comparison

We utilize the simulated dataset not only to validate our approach, but also to compare the proposed method to existing paradigm for deformation tracking from the state-of-the-art literature. As a comparison, it must be noted that merely idea of applying controlled forces to mesh vertices for deformation tracking is not entirely novel. As noted in Sec. 1.3.1.1, similar ideas have been proposed by [Schulman et al., 2013], [Wang et al., 2015] and [Petit et al., 2015a]. The novelty of the approach proposed by us lies in

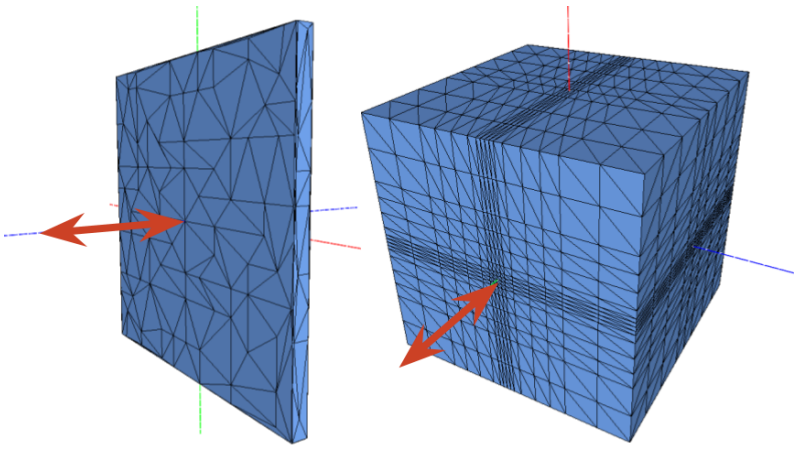


FIGURE 3.5 – The undeformed model of the *board* and the *cube*. The arrows in the image shows the approximate direction of application of force

the implementation of a geometric error minimization mechanism using a numeric optimization scheme. This introduces a novel paradigm in non-rigid object tracking using volumetric, physically-based meshes.

In this context, it becomes necessary to compare our approach with the fundamental deformation tracking mechanism of [Schulman et al., 2013], [Wang et al., 2015] and [Petit et al., 2015a]. The source code or results on a benchmarked dataset is not available from any of these authors. Hence, we design a minimalistic strategy to obtain a comparable tracking mechanism, as explained below.

[Schulman et al., 2013] uses a standard FEM as the deformation model while [Wang et al., 2015] and [Petit et al., 2015a] uses a co-rotational FEM. We restrict our comparison to the co-rotational model only, since it provides a more evolved formulation over the standard FEM. The external, elastic forces, $\mathbf{f}_{ext}$, acting on the object is given by:

$$\mathbf{f}_{ext,i} = k_{ext,i}(\mathbf{x}_i - \mathbf{y}_i) \tag{3.23}$$

where k_{ext} is the stiffness corresponding to the external elastic force and $\mathbf{y}_i$ is a point in the pointcloud which has been matched to $\mathbf{x}_i$, a vertex on the mesh. Eqn. 3.23 forms the default, per-vertex force estimation method for both [Wang et al., 2015] (Eqn. 1.71) and [Petit et al., 2015a] (Eqn. 4.29). We re-implement this force computation, whereas the per vertex displacement vector $(\mathbf{x}_i - \mathbf{y}_i)$ is obtained from the ground-truth in the simulation. The force is applied on the vertices selected by clustering and the FEM simulation is allowed to resolve the mesh displacements over the entire object model for

a fixed number of iterations. The number of iterations allowed to converge is constant for both this re-implementation and the approach proposed by us. This implies that in the re-implementation, the displacement vector associated with the deformation is provided exactly from the groundtruth, providing it with a significant advantage over the approach proposed by us. In general, the fundamental difference between the method proposed in this chapter and the approach of [Petit et al., 2015a] lies in the fact that we propose an optimization based method for deformation tracking while [Petit et al., 2015a] relies on force-estimation from pointcloud matching using a modified k-nearest neighbor search.

In the subsequent text, the results obtained from the approach proposed in this chapter are denoted by *Numeric-optimization*, while the results from this partial re-implementation scheme (as discussed above) are denoted by *Linear-force-estimate*.

To analyze the robustness of the two approaches, experiments were repeated by varying the Young's modulus of the model for determining the variance in error with change in physical parameters. The physical parameters used for simulation are considered unknown in the implementation of the proposed approach. The tracking was repeated over the values of 5 Pa, 500 Pa, 50000 Pa, 5×10^6 Pa and 5×10^8 Pa respectively. The results are summarized in Fig. 3.6 and Fig. 3.7. In the figure, the ground-truth of the model undergoing deformation is provided in the left-most column. The rest of the images show the 3D model (the tracked model) obtained from the tracking algorithm, placed into the pointcloud with the latest estimate of ${}^{C}\mathbf{T}_{O}$.

Since the ground-truth for the deformation is known, it is possible to compute the Hausdorff distance metric [Henrikson, 1999] between the output of the tracking and the ground-truth, to quantify the error between the ideal and the actual output. Fig. 3.8 delineates the Hausdorff distance of the output for various values of Young's modulus.

As given in Table 3.1, the mean error in terms of Hausdorff distance using *Numeric-optimization* and *Linear-force-estimate* are 0.305 and 1.622 units for the *cube* dataset and 0.834 and 1.100 units for the *board* dataset respectively. As a reference, the length of the largest diagonal is 69.28 and 55.19 units for the *cube* and the *board* models respectively (the data is simulated using geometrically defined values for the dimension of the objects and the FEM is simulated without applying the gravity vector, hence the exact unit of length cannot be expressed in SI units directly). Hence, the tracking accuracy of the proposed approach is 81.18% better than *Linear-force-estimate* in the *cube* data and 24.20% better than *Linear-force-estimate* in the *board* data. The tracking

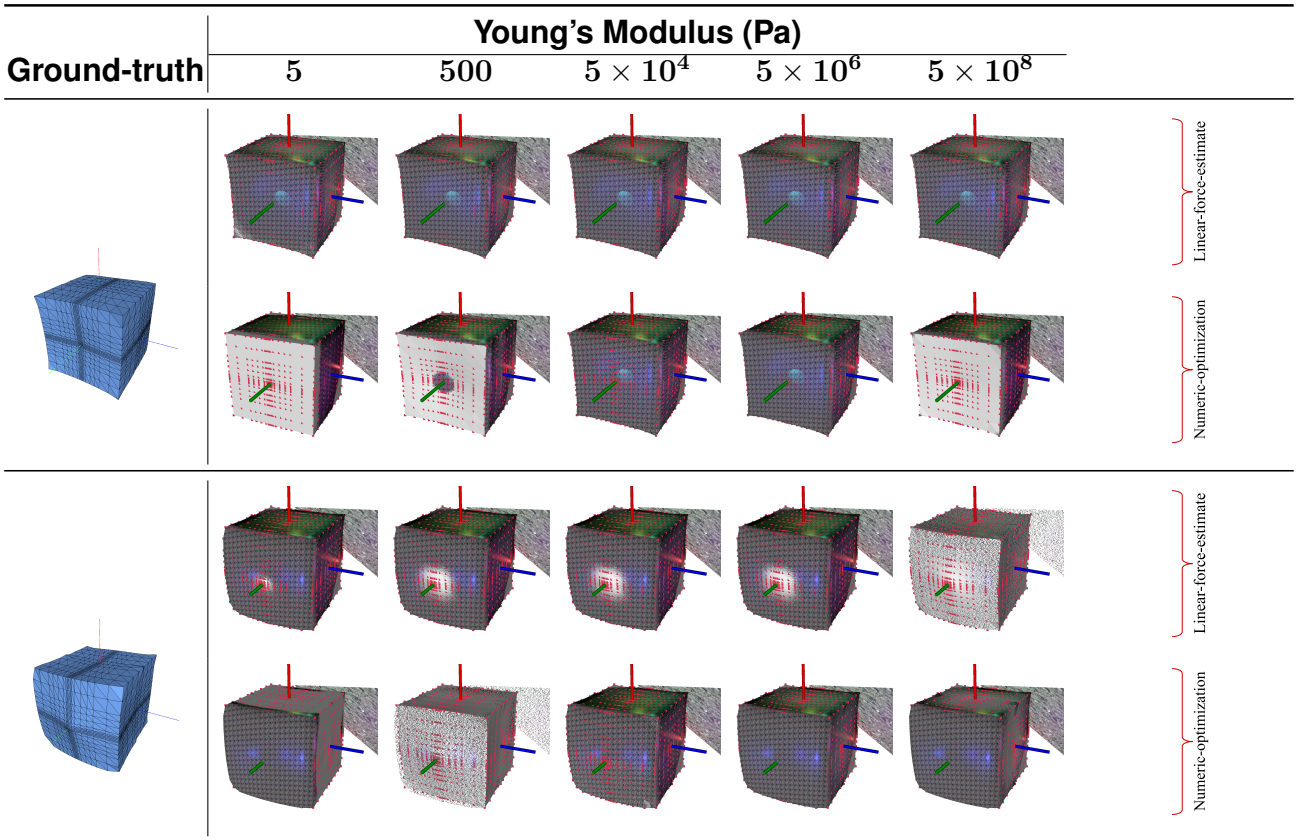


FIGURE 3.6 – Tracking deformation of the *cube* across different values for Young's modulus

accuracy of the proposed approach in terms of Hausdorff distance varies only by an average of 5.006% when the Young's modulus is increased by a factor of 10^8. It can therefore be claimed that the system is significantly robust to error in the estimation of the physical properties used to model the deforming object. It must be noted that the results shown for *Linear-force-estimate* contains only the error in estimating the magnitude of deformation of the objects. As stated before, the direction of application of force is provided to *Linear-force-estimate* from the ground-truth in our re-implementation (while *Numeric-optimization* makes no such exception), which explains the impressive accuracy of *Linear-force-estimate* when using Young's modulus equal to the ground-truth value for simulation (50000 Pa).

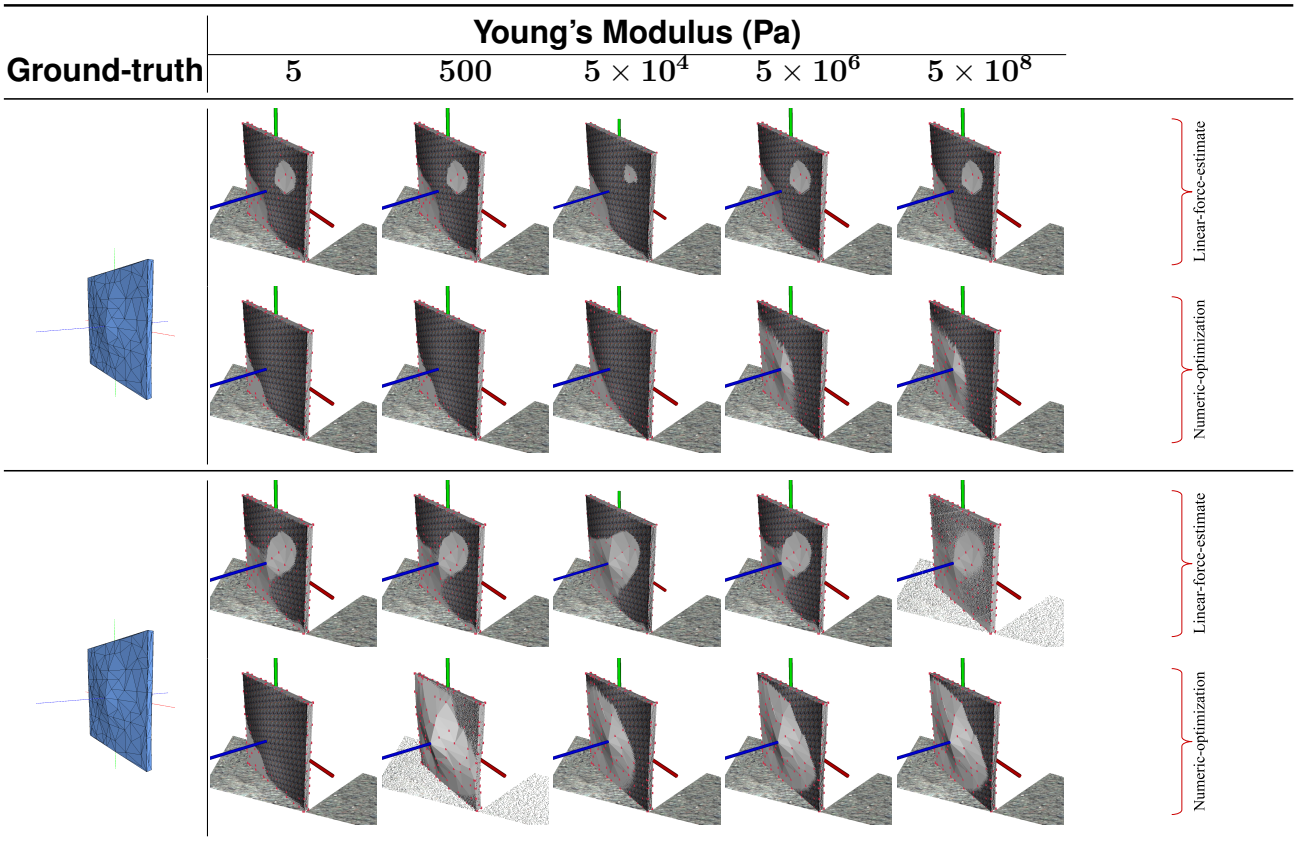


FIGURE 3.7 – Tracking deformation of the *board* across different values for Young's modulus

3.4.2 Real Data

The real data has been captured using the Intel RealSense SR300 RGB-D camera. It is a commodity level depth sensor that produces RGB-D images of reasonably good resolution between 20 to 150 cm depth. Three objects have been tracked for the validation: a) a *pizza*, b) a cuboidal soft *toy* and c) a rectangular *sponge*. A Poisson's ratio of 0.3 was assigned to all the objects uniformly. All the objects were tracked using multiple values of Young's modulus which were set coarsely and empirically. The *pizza* data was tracked with an Young's modulus of 5×10^3 Pa, 5×10^4 Pa and 5×10^5 Pa, the *toy* was tracked using 5×10^4 Pa, 5×10^5 Pa and 5×10^6 Pa, and the *sponge* was tracked using 8×10^5 Pa, 8×10^6 Pa and 8×10^7 Pa. The tracking results are shown in Fig. 3.9, Fig. 3.10 and Fig. 3.11.

Given these results, the average, per pixel point-to-plane error across the entire sequence for multiple values of Young's modulus are given in Fig. 3.12.

Young's Modulus:		**5 Pa**	**500 Pa**	5×10^4 **Pa**	5×10^6 **Pa**	5×10^8 **Pa**
Cube	Numeric-optimization	**0.3118**	**0.3118**	0.3009	**0.3009**	**0.3009**
	Linear-force-estimate	4.6448	2.3788	**0.0051**	0.3455	0.7383
Board	Numeric-optimization	**0.8239**	**0.8239**	0.8761	**0.8237**	**0.8246**
	Linear-force-estimate	2.0562	1.5674	**0.064**	0.9092	0.9081

TABLE 3.1 – Table for comparison of Hausdorff distance between output and ground-truth for *Numeric-optimization* and *Linear-force-estimate* for multiple values of Young's modulus. Lower Hausdorff distance indicates higher accuracy

3.5 Conclusion

We have presented an algorithm for tracking deforming objects using RGB-D camera with the help of physically-based models. The deformation tracking approach is not strongly dependant on the accuracy of the estimate of the physical parameters. A minimization of depth based error term enables us to track the deforming surface accurately. The algorithm can be enabled to track objects in real-time with the help of a parallel architecture. The approach has been qualitatively validated on simulated data, as well as tested on real objects with unknown physical parameters.

However, it is obvious that the depth data provided by the RGB-D sensor is only a fraction of the entire information available to us. It is not prudent to discard the photometric information entirely. Tracking based on additional information can increase the accuracy of the system. Moreover, it is also obvious that the computation of Jacobian by the numerical estimation scheme via repetitive simulation of deformation is extremely expensive in terms of time requirement. Any possible mechanism to compute the Jacobian analytically would greatly improve the per-frame time required for tracking. These two concerns are addressed in the following chapter, where we extend our deformation tracking with the addition of photometric information as well. We also propose a mechanism to approximate the Jacobian analytically, thereby resulting in significantly reduced time requirements.

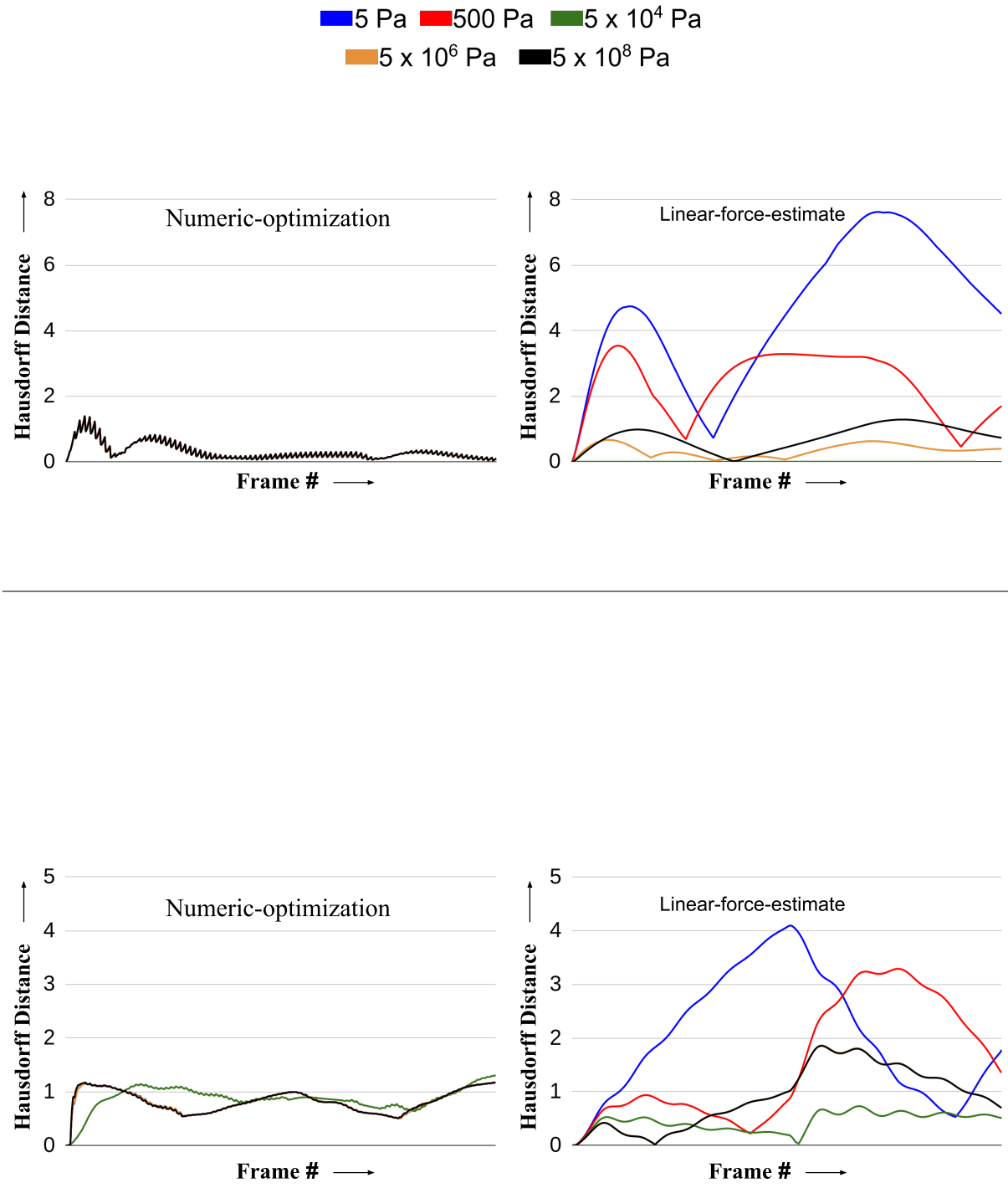


FIGURE 3.8 – Comparison of Hausdorff distance between output and ground-truth for *Numeric-optimization* and *Linear-force-estimate* for multiple values of Young's modulus

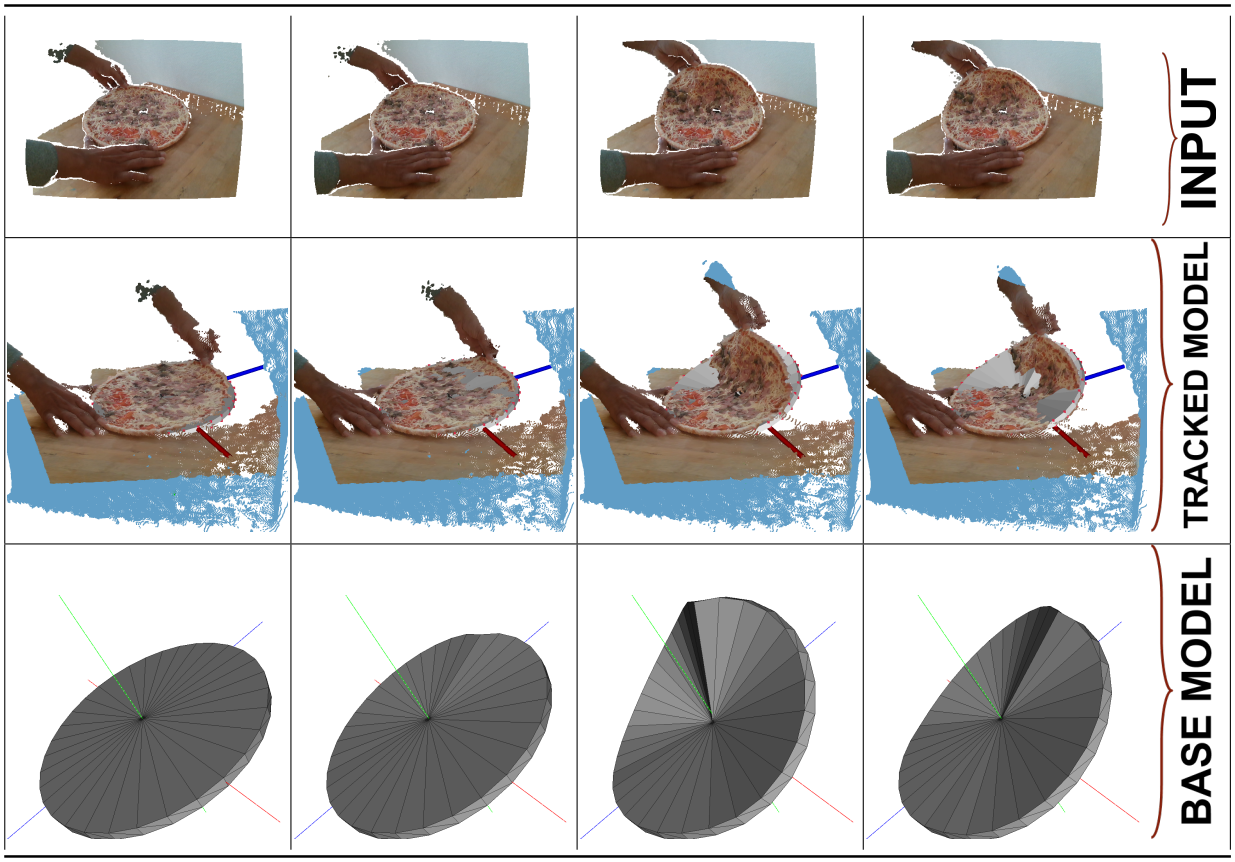


FIGURE 3.9 – Results from tracking of the *pizza* using the method proposed in this chapter. The *top row* shows the input data in the RGB-D pointcloud. The *bottom row* shows the deformed model of the pizza which is produced as a result of the tracking. The *middle row* shows the object model placed inside the 3D pointcloud and rendered with occlusion. The red vertices in the *middle row* corresponds to the vertices of the visual model

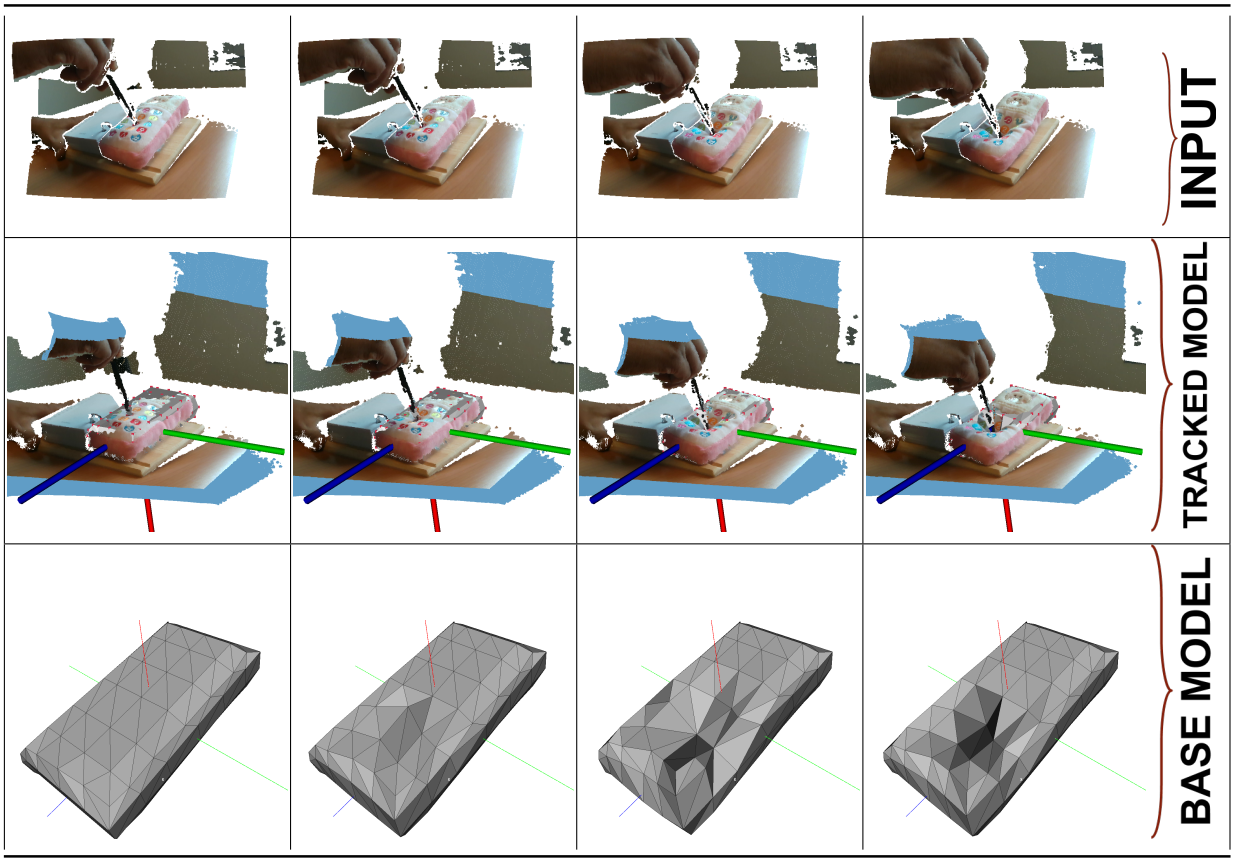


FIGURE 3.10 – Results from tracking of the *toy* using the method proposed in this chapter. The *toy* gets deformed by the application of pressure using a sharp probe. The *top row* shows the input data in the RGB-D pointcloud. The *bottom row* shows the deformed model of the *toy* which is produced as a result of the tracking. The *middle row* shows the object model placed inside the 3D pointcloud and rendered with occlusion. The red vertices in the *middle row* corresponds to the vertices of the visual model

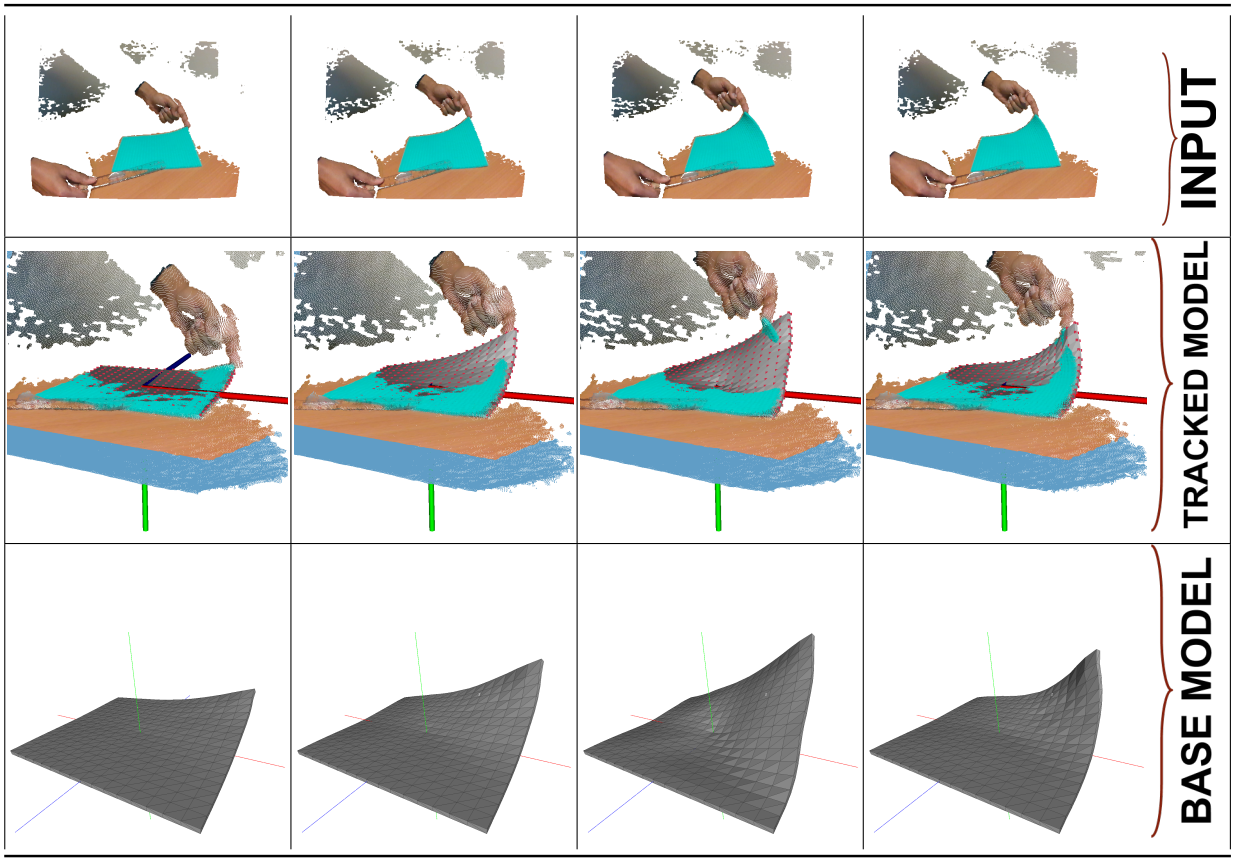
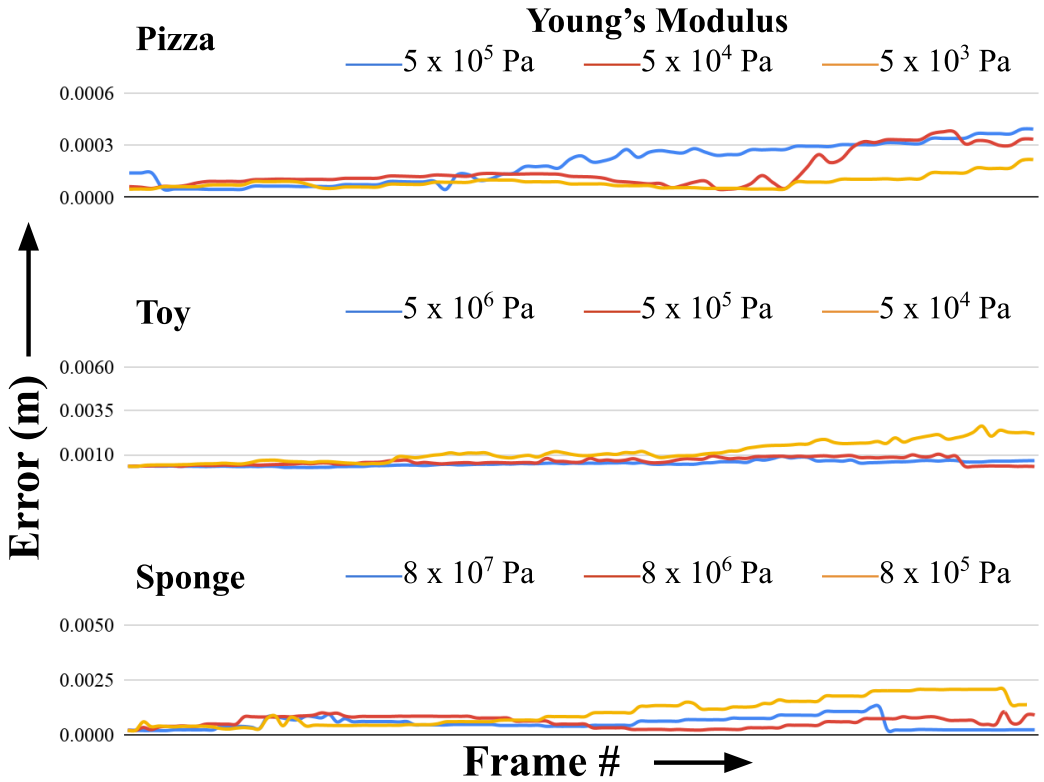
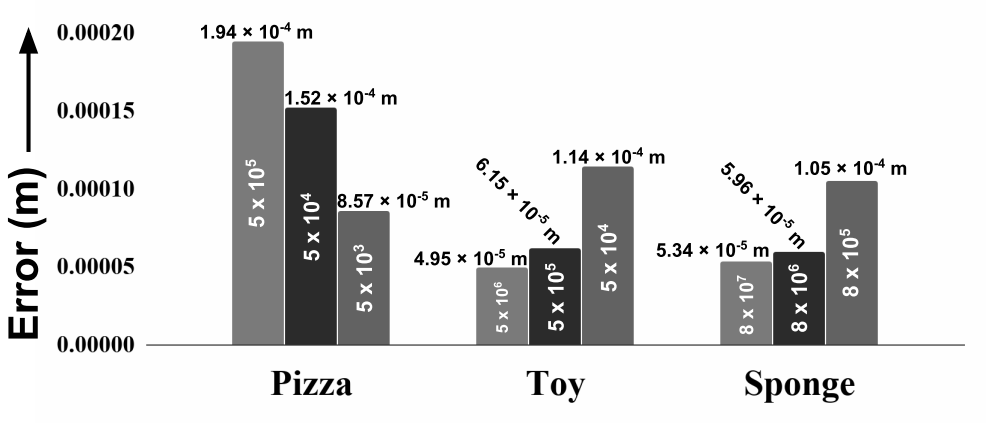


FIGURE 3.11 – Results from tracking of the *sponge* using the method proposed in this chapter. The *sponge* gets folded inwards by lifting one of its end while holding down the other end with a transparent ruler. The *top row* shows the input data in the RGB-D pointcloud. The *bottom row* shows the deformed model of the *sponge* which is produced as a result of the tracking. The *middle row* shows the object model placed inside the 3D pointcloud and rendered with occlusion. The red vertices in the *middle row* corresponds to the vertices of the visual model



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

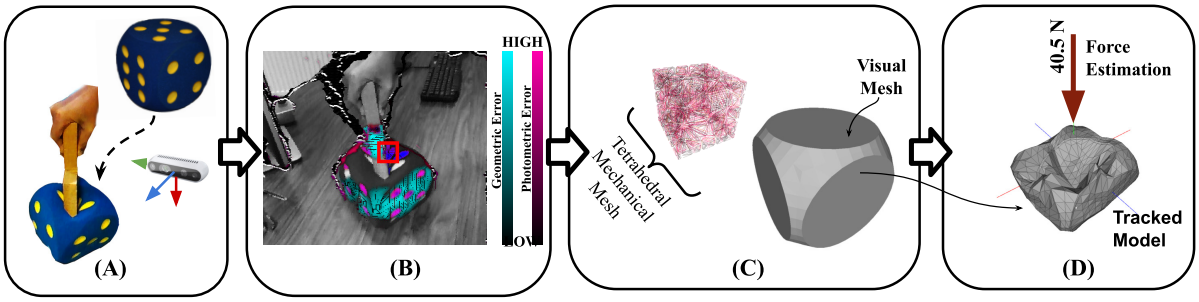


(b) The mean point-to-plane error across all the frames, plotted against the respective Young's Modulus (shown inside the bars, the values are in Pascal)

FIGURE 3.12 – Summary of point-to-plane error obtained from experiments on real data

CHAPTER 4

NON-RIGID OBJECT TRACKING WITH DEPTH AND PHOTOMETRY



A method for model based tracking of deformation and deforming forces is presented in this chapter: (A) the deformation is captured using a RGB-D sensor, (B) a combination of geometric and photometric errors are minimized, (C) the minimization is done using FEM, (D) the deformed mesh is the default output of the system, while the force applied can also be estimated if the material properties of the object are known

4.1 Introduction

In the previous chapter, we presented a method to iteratively minimize depth based geometric error using FEM. The method has been utilized to accurately track both simulated and real deforming objects. However, in the formulation presented in the previous chapter, there are two aspects which can be improved.

The first concern arises from computing the gradient of error using repeated simulation of mesh deformation with the FEM model. It requires a significant amount of time. We managed to extricate fast runtime from the system by utilizing a parallelization scheme, but this type of implementation tracks the deformation while skipping certain

frames, because the rigid tracking runs significantly faster than the non-rigid tracking method. It is not trivial to modify this algorithm into a sequentially real-time system. It is not possible to optimize $\sim 20 - 30$ iterations of complete FEM simulation, each containing $\sim 40 - 60$ iterations of the conjugate gradient solver into a real-time system. However, skipping intermediate frames is also not desirable.

The second aspect which merits closer inspection is the fact that purely depth based object tracking (especially using a point-to-plane distance based cost function) has scope for improvement. In Chapter 2, we already noticed that combining photometric error minimization with depth based error lead to better accuracy in the case of rigid tracking. It can be interpreted intuitively that addition of more error terms into the minimization loop of Chapter 3 should usually lead to better accuracy in the case of non-rigid object tracking too. The availability of images from the RGB sensor of the depth camera leads us to conclude that a direct photometric intensity based error term is an obvious choice for this additional error.

In this chapter, we address these concerns arising out of Chapter 3. A framework is proposed to track non-rigid objects using RGB-D cameras by utilizing physically-based models. The physics of the model is similar to that proposed in Chapter 3, i.e., co-rotational FEM. The methodology proposed here integrates classical computer vision based error minimization using photometric and geometric cost functions with the deformation simulation capabilities of co-rotational FEM.

The main contributions of the methodology proposed in this chapter are:

— Combined minimization of error terms based on depth and photometric information ;

— A new technique for approximating the gradient of these errors w.r.t displacement of vertices ;

— An improved methodology for determining the vertices which are to be controlled for tracking ;

— Improved runtime

4.2 Background

This chapter combines classical computer vision with geometric analysis of deforming manifolds and physics-based structural models from mechanical engineering.

Tracking deformation using RGB-D cameras is an advanced area of research in computer vision. However, purely computer vision based algorithm for deformation tracking does not provide any real world information about the nature of physical interactions that the object is undergoing. The objective of this chapter is to treat the deforming object from a physics based perspective, such that both the deformation and the interaction causing the deformation can be accounted for.

Deformation has been tracked in real-time using mass-spring-damper models, as well as linear Finite Element Models (FEM) by either computing a point-to-point correspondence [Petit et al., 2017b] for estimating the deformation of the surface, or by using a steepest gradient descent based approach [Royer et al., 2017] to minimize a SSD/SSE type cost function. FEM is being extensively used to solve mechanical engineering problems involving complex elasticity and structural analysis for the last eight decades [Hrennikoff, 1941] [Courant et al., 1943] or more. Consequently, FEM offers us a wide variety of highly evolved mathematical tools to model mechanical properties of real objects. A model based minimization strategy to track deformation has been proposed in in the previous chapter, but it cannot be considered real-time. Tracking of topological changes such as cutting and tearing has also been demonstrated using simple feature correspondences [Paulus et al., 2015]. An approach has been proposed to combine NRSfM with FEM by representing the deformation as a linear combination of modal shapes [Agudo et al., 2014], but the method has been validated only on laminar models.

Interesting use-cases, such as estimation of material properties, begin to appear when tracking vibrations of objects using computer vision [Davis et al., 2015], but these approaches have not been developed for deformation tracking of volumetric objects. External force sensing by vision has been proposed by Zhang et. al. [Zhang et al., 2019], but the deformation tracking method proposed here is a variant of feature matching, which is partially similar to [Petit et al., 2017b], and hence can be expected to suffer all the drawbacks of point matching based object tracking approaches. The approach proposed in this chapter has been compared against point matching based approaches for deformation tracking (see Section 4.4) and has been demonstrated to outperform the state-of-the-art for these types of physically based tracking algorithms.

It is evident that there is a clear and strong separation between the evolved computer vision techniques used in physical model free deformation tracking, and the relatively simpler techniques used while tracking using physics based models. In this

chapter, we propose to bridge this divide between vision based tracking and mechanical models by enabling an iterative minimization strategy for generic, vision-based cost functions using FEM. The method described here involves the minimization of a combination of a geometric and a photometric cost functions. A textureless, coarse model is required as the input of the system while the output consists of the tracked 3D model. Moreover, the availability of an advanced physically based model enables us to track the forces acting on the deformed object, whenever an approximate estimate of the material properties are available.

4.3 Proposed Approach

The input to the system consists of RGB-D data stream from a depth camera and a coarse 3D model of the surface of the object being tracked. Unlike similar methods in the literature, we do not require the material properties of the object being tracked or the textured model of the object. For each frame, the first step is to approximately track the rigid pose of the object w.r.t the camera (described in Section 4.3.3). This is followed by determining the approximate areas where the error in rigid registration has a higher magnitude (see Section 4.3.2). The mesh vertices lying close to these areas with high rigid registration error are termed as *control handles*. These *control handles* are utilized to minimize a set of error terms (explained in Section 4.3.1), which are defined w.r.t the surface model of the object.

4.3.1 Non-rigid Tracking

We briefly recap our motivation behind proposing the non-rigid object tracking method of this chapter before explaining the technical details.

4.3.1.1 Motivation

The procedure for real-time, RGB-D based tracking of non-rigid objects without using mechanical model usually involves the iterative minimization of one or multiple highly non-linear objective function, such as point-to-point distance [Wang et al., 2017], point-to-plane distance [Newcombe et al., 2015], combination of point-to-plane distance and sparse feature alignment error [Innmann et al., 2016], etc. However, in the

absence of a physics based framework for modelling the real-world properties of the deforming object, a regularizing or smoothing data term is used to prevent unnatural deformations while tracking the object. For tracking using physics based models, the problem statement becomes much more constrained due to the presence of a model that already mimics the physical properties of the actual object. The tracking methodology changes from an iterative minimization to displacing one or multiple vertex/vertices to follow the object deformation, since estimating the gradient of the non-linear error term with respect to the vertex displacement involves time-consuming deformation simulation, which is not suitable for online tracking. However, we have demonstrated in Chapter 3 that iterative optimization of depth based error term is possible using physically-based model and results in accurate tracking. Through the mechanism proposed in this chapter for tracking non-rigid objects, we aim to bridge this divide between the physical-model free and physical-model based approaches even further. This is done by the minimization of photometric error term along with depth based error while the gradient of error in approximated analytically. We first describe the overall framework for generic objective functions. The specific error term is described thereafter.

4.3.1.2 Methodology

We now describe our proposed methodology for tracking non-rigid objects. Let us denote the non-linear error term that we seek to minimize by E. We first discuss the procedure that we follow to minimize this error using the physically based model, i.e., co-rotational FEM. Throughout the rest of the chapter, the notation $\mathbf{P}^{\mathrm{M}}$ is used to denote the vertices of the mechanical tetrahedral mesh $\mathcal{V}^{\mathrm{M}}$, while $\mathbf{P} = (\mathrm{P_X}, \mathrm{P_Y}, \mathrm{P_Z})$ denotes the vertices of the surface model $\mathcal{V}^{\mathrm{S}}$ w.r.t the object centered reference frame $\mathcal{F}_{\mathrm{o}}$. The surface model consists of triangular faces and the j-th face is denoted by:

$$\underbrace{\mathbf{V}_j}_{3\times 3} = \begin{pmatrix} \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2} \end{pmatrix} \tag{4.1}$$

starting from the i-th vertex of $\mathcal{V}^{\mathrm{S}}$. The mapping $\mathcal{V}^{\mathrm{S}} \mapsto \mathcal{V}^{\mathrm{M}}$ is done via a standard master-slave configuration using barycentric mapping. The RGB-D camera is assumed to be at a reference frame $\mathcal{F}_{\mathrm{c}}$. For the sake of brevity, the baseline between the depth and the color camera is not mentioned explicitly and all data presented here are assumed to be aligned with the color camera.

Let us assume that the method chosen for the deformation tracking involves a physical model $\mathcal{P}$ that describes the interaction between the applied forces and displacement of the vertices of $\mathcal{V}^{\mathrm{M}}$. Recall that $\mathcal{V}^{\mathrm{M}}$ is coupled with the surface geometry $\mathcal{V}^{\mathrm{S}}$ of the object. We propose to track the deformation of this object using the minimization of one or multiple error term(s) defined strictly on the triangular elements of the faces of $\mathcal{V}^{\mathrm{S}}$. This is done by defining a set of *control handles* $\mathbf{C}$ on $\mathcal{V}^{\mathrm{M}}$, such that the entire deformation becomes optimizable by regulating the displacement of $\mathbf{C}$. Note that any displacement of $\mathbf{C}$ gets propagated to all vertices of $\mathcal{V}^{\mathrm{M}}$ using the properties of $\mathcal{P}$. With these constraints in place, let us assume that we are trying to optimize for a set of N objective functions $\mathrm{E}_1, \mathrm{E}_2, ..., \mathrm{E}_{\mathrm{N}}$, defined on $\mathcal{V}^{\mathrm{S}}$. We are interested in minimizing

$$\mathrm{E}_{\mathrm{S}} = (\mathrm{E}_1, \mathrm{E}_2, \cdots, \mathrm{E}_{\mathrm{N}}) \tag{4.2}$$

with the help of the displacement vector on $\mathbf{C}$, given by $\mathbf{u}_{\mathrm{C}}$. However, to optimize in a non-linear least squares fashion, we need to obtain the gradient of the stacked/combined cost function, which can be done numerically. But with any modern physical model $\mathcal{P}$, numeric estimation can be computationally expensive as determining the node displacements typically involves multiple iteration of conjugate gradient descent for solving an equation of the nature of (3.4). This is beyond the reasonable limits of performance optimization and, to the best of our knowledge, online method for deformation tracking using minimization of visual error term(s) w.r.t node displacements of a mechanical model is unavailable in the literature. We propose an assumption to solve this problem, as described below.

Let us analyze the case of an arbitrary point $\mathbf{p}$ of the RGB-D pointcloud, which lies on the triangle $\begin{pmatrix}\mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2}\end{pmatrix}$ of $\mathcal{V}^{\mathrm{S}}$. Let us assume that a small displacement:

$$\Delta\mathbf{u}_{\mathrm{C}} = \begin{bmatrix}\Delta\mathrm{u}_{\mathrm{C_X}} & \Delta\mathrm{u}_{\mathrm{C_Y}} & \Delta\mathrm{u}_{\mathrm{C_Z}}\end{bmatrix} \tag{4.3}$$

produces the displacement $\Delta\mathbf{x}, \Delta\mathbf{y}$ and $\Delta\mathbf{z}$ on the j-th surface plane of $\mathcal{V}^{\mathrm{S}}$ comprising of three vertices $\mathbf{P}_i, \mathbf{P}_{i+1}$ and $\mathbf{P}_{i+2}$ respectively. This vector of vertex positions is given by:

$$\boldsymbol{\vartheta}_j = \begin{bmatrix}\mathbf{P}_i^\top & \mathbf{P}_{i+1}^\top & \mathbf{P}_{i+2}^\top\end{bmatrix} \tag{4.4}$$

Subsequently, the gradient of the error $\mathrm{E}_{\mathbf{S}}(\mathbf{p})$ w.r.t $\mathbf{u}_{\mathrm{C}}$ is given by the Jacobian:

$$\mathbf{J}_{\mathbf{S}}(\mathbf{p}) = \frac{\partial \mathrm{E}_{\mathrm{S}}(\mathbf{p})}{\partial \mathbf{u}_{\mathrm{C}}} = \underset{1\times 9}{\frac{\partial \mathrm{E}_{\mathrm{S}}(\mathbf{p})}{\partial \boldsymbol{\vartheta}_j}} \underset{9\times 3}{\frac{\partial \boldsymbol{\vartheta}_j}{\partial \mathbf{u}_{\mathrm{C}}}} \tag{4.5}$$

Estimating $\frac{\partial \mathrm{E}_{\mathrm{S}}(\mathbf{p})}{\partial \boldsymbol{\vartheta}_j}$ is usually a straightforward modification of a series of well established techniques in computer vision [Hartley and Zisserman, 2003]. The details of how this can be done in our case is demonstrated in section 4.3.4. On the other hand, the term $\frac{\partial \boldsymbol{\vartheta}_j}{\partial \mathbf{u}_{\mathrm{C}}}$ is somewhat similar to the classical strain-displacement matrix, as expressed in conventional FEM literature. As stated before, determining the exact value of $\frac{\partial \boldsymbol{\vartheta}_j}{\partial \mathbf{u}_{\mathrm{C}}}$ for any sufficiently advanced choice of $\mathcal{P}$ will require computationally expensive simulation of model deformation at every iteration of the non-linear least squares solver.

We propose that this additional complexity can be avoided by estimating the deformation of the mechanical model $\mathcal{V}^{\mathrm{M}}$, for every new *control handle* explored by our proposed approach, only once. For the j-th *control handle* $\mathbf{C}_j$, this is done by displacing the vertex in $\mathcal{V}^{\mathrm{M}}$ corresponding to $\mathbf{C}_j$ by a small distance along the positive direction of X, Y and Z axis. The magnitude of this deformation $\Delta\mathbf{u}_C$ is empirically determined such that it slightly exceeds the RMSE of the sensor noise (e.g: we use Intel RealSense D435 for capturing the real data, and the magnitude of $\Delta\mathbf{u}_C$ was set to 2 mm, which successfully overwhelmed the sensor noise [Ahn et al., 2019]).

This estimation step produce three deformed meshes per *control handle*, which can be directly utilized for determining $\frac{\partial \boldsymbol{\vartheta}_j}{\partial \mathbf{u}_{\mathrm{C}}}$ using forward finite differences. Assuming that $\mathbf{u}_{\mathrm{C}}$ points to the c-th vertex of $\mathcal{V}^{\mathrm{M}}$, the values of $\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_{\mathrm{C}}}$ can be mapped to a matrix $\boldsymbol{\Gamma}_{\mathrm{O}}$ by a mapping function $\mathscr{C}$ such that:

$$\boldsymbol{\Gamma}_{\mathrm{O}} = \mathscr{C}\Big(\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_{\mathrm{C}}}\Big) \tag{4.6}$$

given that:

$$\frac{\partial \boldsymbol{\vartheta}_j}{\partial \mathbf{u}_{\mathrm{C}}} = \Big(\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_{\mathrm{C}}}, \frac{\partial \mathbf{P}_{i+1}}{\partial \mathbf{u}_{\mathrm{C}}}, \frac{\partial \mathbf{P}_{i+2}}{\partial \mathbf{u}_{\mathrm{C}}}\Big) \tag{4.7}$$

Here, $\mathscr{C}$ represents the mapping $\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_{\mathrm{C}}} \mapsto \boldsymbol{\Gamma}_{\mathrm{O}}$ and it is a one-to-one, bijective map as long as the number of vertices in $\mathcal{V}^{\mathrm{M}}$ and $\mathcal{V}^{\mathrm{S}}$ are equal. Assuming that M denotes the

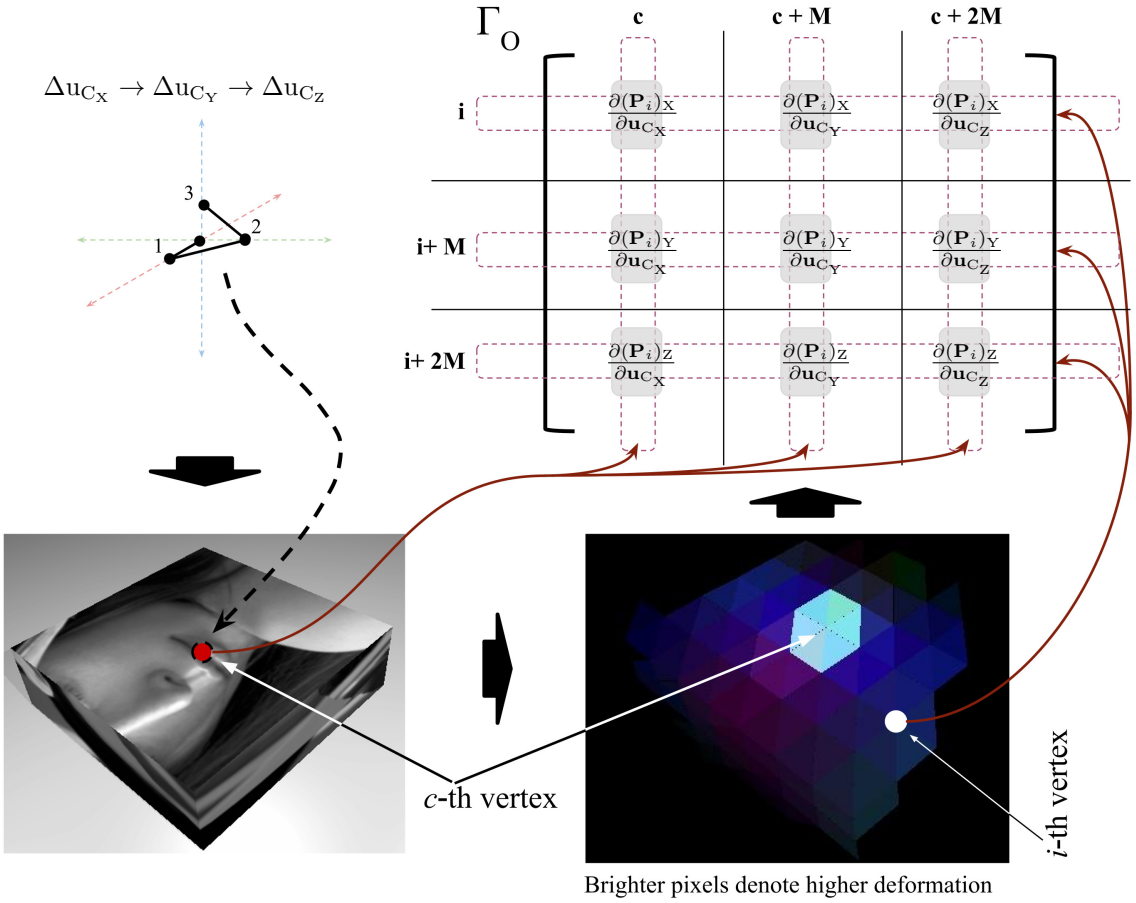


FIGURE 4.1 – This figure shows the deformational displacement $\Delta \mathbf{u}_C$ being applied to the c-th vertex of the mechanical mesh. The effect of this displacement on the i-th vertex gets stored in $\mathbf{\Gamma}_O$. This procedure is repeated for all $c, i \in M$

number of vertices in $\mathcal{V}^M$ and $\mathcal{V}^S$, $\mathscr{C}$ can be expressed as:

$$\mathbf{\Gamma}_O = \mathscr{C}\left(\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_C}\right) = \sum_{i=0}^{M}\sum_{c=0}^{M} \frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_C} \otimes \mathfrak{J}^{i,c}_{M\times M} \tag{4.8}$$

where $\otimes$ gives the Kronecker product and $\mathfrak{J}^{i,c}_{M\times M}$ denotes the conventional single-valued matrix of dimension $[M \times M]$, such that:

$$\left(\mathfrak{J}^{i,c}_{M\times M}\right)_{x,y} = \begin{cases} 1 & \text{if } x = i \wedge y = c, \forall x, y \in M \\ 0 & \text{else} \end{cases} \tag{4.9}$$

The construction of $\mathbf{\Gamma}_{\mathrm{O}}$ is demonstrated graphically in Fig. 4.1. Based on the matrix indices, $\mathbf{\Gamma}_{\mathrm{O}}$ can be represented as:

$$\big(\mathbf{\Gamma}_{\mathrm{O}}\big)_{a\mathrm{M}+i,b\mathrm{M}+c} = \big(\frac{\partial \mathbf{P}_i}{\partial \mathbf{u}_{\mathrm{C}}}\big)_{a,b} \mid \forall i, c \in \mathrm{M} \wedge a, b \in \{0,1,2\} \tag{4.10}$$

In the expanded form, $\mathbf{\Gamma}_{\mathrm{O}}$ for the i-th vertex $\mathbf{P}_i$ under the effect of the displacement vector $\mathbf{u}_{\mathrm{C}}$ on the c-th vertex can be expressed as:

$$\underset{3\mathrm{M}\times 3\mathrm{M}}{\mathbf{\Gamma}_{\mathrm{O}}} = \left[\begin{array}{ccccc|ccccc|ccccc|c}
\mathbf{P}_1 & \cdots & \mathbf{P}_c & \cdots & \mathbf{P}_{\mathrm{M}} & \mathbf{P}_1 & \cdots & \mathbf{P}_c & \cdots & \mathbf{P}_{\mathrm{M}} & \mathbf{P}_1 & \cdots & \mathbf{P}_c & \cdots & \mathbf{P}_{\mathrm{M}} & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_\cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{X}}}{\partial \mathbf{u}_{\mathrm{C_X}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{X}}}{\partial \mathbf{u}_{\mathrm{C_Y}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{X}}}{\partial \mathbf{u}_{\mathrm{C_Z}}} & \cdot & \cdot & \mathbf{P}_i \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_{\mathrm{M}} \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_\cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Y}}}{\partial \mathbf{u}_{\mathrm{C_X}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Y}}}{\partial \mathbf{u}_{\mathrm{C_Y}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Y}}}{\partial \mathbf{u}_{\mathrm{C_Z}}} & \cdot & \cdot & \mathbf{P}_i \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_{\mathrm{M}} \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_\cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Z}}}{\partial \mathbf{u}_{\mathrm{C_X}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Z}}}{\partial \mathbf{u}_{\mathrm{C_Y}}} & \cdot & \cdot & \cdot & \cdot & \frac{\partial(\mathbf{P}_i)_{\mathrm{Z}}}{\partial \mathbf{u}_{\mathrm{C_Z}}} & \cdot & \cdot & \mathbf{P}_i \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vdots \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{P}_{\mathrm{M}}
\end{array}\right] \tag{4.11}$$

$\mathbf{\Gamma}_{\mathrm{O}}$ can be conveniently computed offline once per every object model and stored. During tracking, $\mathbf{\Gamma}_{\mathrm{O}}$ is loaded from memory when required.

For the next step, let us recall that the transformation from the object to the camera reference frame is given by ${}^{O}\mathbf{T}_{C}$, following the standard notation defined in Sec. 1.1.1. We reconstruct the rotation matrix ${}^{C}\mathbf{R}_{O}$ using the element indices given in Eqn. 1.3 to

represent a modified matrix $\mathbf{R}^{\Gamma}$ representing the rotation such that:

$$\underset{3\mathrm{M}\times 3\mathrm{M}}{\mathbf{R}^{\Gamma}} = {}^{C}\mathbf{R}_{O} \otimes \mathbb{1}_{\mathrm{M}\times\mathrm{M}} = \begin{bmatrix} \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{11}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{12}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{13}) \\ \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{21}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{22}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{23}) \\ \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{31}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{32}) & \underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{33}) \end{bmatrix} \tag{4.12}$$

where:

$$\underset{\mathrm{M}}{\mathrm{diag}}(\mathrm{r}_{x}) = \underbrace{\begin{bmatrix} \mathrm{r}_{x} & & \\ & \ddots & \\ & & \mathrm{r}_{x} \end{bmatrix}}_{\mathrm{M}\times\mathrm{M}} \tag{4.13}$$

and $\mathbb{1}_{\mathrm{M}\times\mathrm{M}}$ is an identity matrix of dimension $[\mathrm{M}\times\mathrm{M}]$. This allows us to rotate $\mathbf{\Gamma}_{\mathrm{O}}$ (obtained from offline computation) to the current camera reference frame. This transformed matrix $\mathbf{\Gamma}_{\mathrm{C}}$ is given by:

$$\mathbf{\Gamma}_{\mathrm{C}} = \mathbf{R}^{\Gamma}\mathbf{\Gamma}_{\mathrm{O}} \tag{4.14}$$

We then derive the matrix $\mathbf{\Gamma}$ by thresholding $\mathbf{\Gamma}_{\mathrm{C}}$ such that its value at an arbitrary index m, n associated with the i-th vertex $\mathbf{P}_i$ is given by:

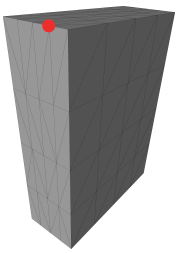
$$\mathbf{\Gamma}_{m,n} = \begin{cases} 1 & \text{if } \boldsymbol{\varepsilon} > \frac{3\alpha||\Delta\mathbf{u}_C||}{2} \\ 0 & \text{else} \end{cases} \tag{4.15}$$

given that:

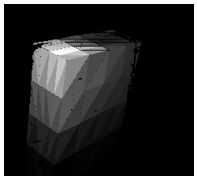
$$\boldsymbol{\varepsilon} = \sum_{a=0}^{2}\sum_{b=0}^{2}\left(\left(\mathbf{\Gamma}_{\mathrm{C}}\right)_{m+a,n+b}\right)^{2} \tag{4.16}$$

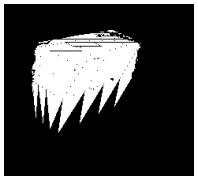
where $\Delta\mathbf{u}_C$ is the calibrating deformation (implying that $||\Delta\mathbf{u}_C||$ is a constant for a particular $\mathbf{\Gamma}_{\mathrm{O}}$). α is a tunable parameter which regulates the area for which the value of $\mathbf{\Gamma}$ will be 1.

At this point, it is natural to question the necessity of thresholding the value of $\mathbf{\Gamma}_{\mathrm{C}}$. using Eqn. (4.15). This thresholding is done purely because we intend to utilize the point-to-plane distance as an error term in the minimization, as explained in Sec. 4.3.4. Point-to-plane distance minimization using planar (or nearly planar) surface model re-



(a) Control Handle - marked in red





(b) Intensity map of $\mathbf{\Gamma}$ (c) Thresholded values of $\mathbf{\Gamma}$

FIGURE 4.2 – Given the *control handle* marked by red in (a), the numerically obtained values of $\mathbf{\Gamma}$ are plotted on the pointcloud registered to the object model, as shown in (b). The intensity of $\mathbf{\Gamma}$ linearly decreases as we go farther from the *cotrol point*. The values of $\mathbf{\Gamma}$ after applying the threshold (4.15) are shown in (c)

sults in the model '*sliding*' over the 3D data it aims to minimize [Chen and Medioni, 1992]. This makes point-to-plane distance minimization ineffective for tracking planar surfaces. However, this is exactly what happens if we directly use the value of $\mathbf{\Gamma}_{\mathrm{C}}$. The calibrating deformation $\Delta\mathbf{u}_C$ tends to exaggerate the deformation close to the vertex index c, i.e., $\mathbf{\Gamma}_{\mathrm{C}}$ highly prioritizes vertices close to c. This, in turn, converts the Jacobian in Eqn. (4.5) to effectively consider only a small patch of surface near the vicinity of the c-th vertex while neglecting the small magnitude of deformation observed in the vertices farther away from the c-th vertex. To overcome this problem, we employ this trick of binarizing $\mathbf{\Gamma}_{\mathrm{C}}$. The parameter α in Eqn. (4.15) is used to expand the area of influence of $\mathbf{\Gamma}_{\mathrm{C}}$ artificially, such that it covers more orthonormal (or nearly orthonormal) faces, thereby eliminating the problem of drift. As explained in Sec. 4.3.4.1, this manner of approximation for the value of Eqn. (4.5) does not affect the minimization of error in a significant manner.

We term this matrix $\mathbf{\Gamma}$, after the threshold has been applied, as the *influence matrix*, and the value is kept constant (for a given *control handle*), once it has been determined. It has been demonstrated experimentally that the assumption of keeping the *influence matrix* constant after the initial calibration results in negligible loss of accuracy in tracking (see Table 4.1) while providing significant improvement in the time required

to handle each data frame (see Table 4.2). Intuitively, these values after applying the threshold in the *influence matrix* merely determines the faces of $\mathcal{V}^{\mathrm{S}}$ for which the error term $\mathrm{E_S}$ will be minimized, as shown in figure 4.2. Replacing the matrix elements with 1 turns M into a pseudo-rigid object around the vicinity of $\mathbf{u}_C$. As long as the inter-frame deformation remains small, this assumption remains valid and does not affect the outcome of the minimization in a significant manner.

4.3.2 Determining the Control Handles

The determination of the *control handles* can also be handled in a generic way. Given the set of N objective functions of $\mathrm{E_S}$ acting on L different points of the pointcloud, such that $\mathbf{p} = \begin{pmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_{\mathrm{L}}\end{pmatrix}$. We define a function $\mathcal{I}_{\mathrm{E}_n}\big(\mathbf{\Pi}(\mathbf{p}_{\mathrm{L}})\big)$ which represents the error value using the n-th cost function $\mathrm{E}_n(\mathbf{p}_l)$ projected on the image. The combined error matrix $\mathcal{I}_{\mathrm{E_S}}\big(\mathbf{\Pi}(\mathbf{p})\big)$ on the image plane is obtained by:

$$\mathcal{I}_{\mathrm{E_S}}\big(\mathbf{\Pi}(\mathbf{p})\big) = \big\langle \mathcal{I}_{\mathrm{E_1}}\big(\mathbf{\Pi}(\mathbf{p})\big)\big\rangle \odot \cdots \odot \big\langle \mathcal{I}_{\mathrm{E_N}}\big(\mathbf{\Pi}(\mathbf{p})\big)\big\rangle \tag{4.17}$$

where $\odot$ denotes the Hadamard product and $\langle\cdot\rangle$ gives the normalized matrix. $\mathcal{I}_{\mathrm{E_S}}\big(\mathbf{\Pi}(\mathbf{p})\big)$ is clustered into multiple clusters, and the centroid of each cluster is associated with the nearest vertex $\mathbf{\Pi}(\mathbf{r})$, the projection of the visible vertices of $\mathcal{V}^{\mathrm{M}}$ on the image plane. These associated vertices of $\mathcal{V}^{\mathrm{M}}$ are identified as the *control handles* for that particular frame. This approach of determining the point of interest is highly effective, since it treats the determination of *control handles* as a function of the objective functions in use.

4.3.3 Approximate Rigid Tracking

Before commencing the minimization of the objective function(s) used for deformation tracking, the rigid pose of the object in the current frame needs to be determined approximately. This enables us to track the deforming motions acting on the surface of the object alone, without having to care about the interaction with the surrounding scene (forces causing rigid motion, friction, collision etc.). This is done in a manner exactly similar to Sec. 3.2.3.

4.3.4 Non-rigid Error Terms

Once an approximate estimate of ${}^{p}\mathbf{T}_{p-1}$ has been obtained, a combination of depth based geometric error and direct photometric error is minimized to achieve the tracking of the deformation using the method described in Section 4.3.1. We now define the error term that needs to be minimized for a single point $\mathbf{p}$ of the pointcloud at the p-th data frame. The point-to-plane distance based geometric error is given by:

$$\mathrm{E_N}(\mathbf{p}^p) = \mathbf{n}_j \cdot \mathbf{p}_l^p - d_j \tag{4.18}$$

assuming that the j-th surface of $\mathcal{V}^{\mathrm{S}}$ has been associated with the l-th point of pointcloud $\mathbf{p}_l$. We propose a photometric error term defined on the (p-1)-th frame by:

$$\mathrm{E_P}(\mathbf{p}^{p-1}) = \boldsymbol{\mathcal{I}}_p(\mathbf{p}_e^{p-1}) - \boldsymbol{\mathcal{I}}_{p-1}(\mathbf{p}^{p-1}) \tag{4.19}$$

The updated point position $\mathbf{p}_e^{p-1}$ is determined by a barycentric map

$$\mathbf{p}_e^{p-1} = \begin{bmatrix} \mathbf{P}'_i & \mathbf{P}'_{i+1} & \mathbf{P}'_{i+2} \end{bmatrix} \mathbf{B} \tag{4.20}$$

such that $\mathbf{p}^{p-1}$ corresponded with the triangle $\begin{pmatrix} \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2} \end{pmatrix}$ and $\mathbf{P}'_i$ gives the updated vertex position of $\mathbf{P}_i$ when subject to the update $\boldsymbol{\vartheta}$, given that $\mathbf{B}$ is a column vector denoting the barycentric coordinates of $\mathbf{p}^{p-1}$ w.r.t $\mathbf{P}_i, \mathbf{P}_{i+1}$ and $\mathbf{P}_{i+2}$.

The combined cost function is given by $\mathrm{E_S} = (\mathrm{E_N}, \mu \mathrm{E_P})$, where $\mu = \frac{\|\mathrm{E_N}\|}{\|\mathrm{E_P}\|}$ is used for bringing the geometric and photometric error terms to the same scale. Following the outline provided by (4.5), we can now develop the partial derivatives $\frac{\partial \mathrm{E_N}(\mathbf{p})}{\partial \boldsymbol{\vartheta}}$ and $\frac{\partial \mathrm{E_P}(\mathbf{p})}{\partial \boldsymbol{\vartheta}}$.

4.3.4.1 Jacobian

We describe the method to obtain the Jacobian for the geometric and photometric error term described in Eqn. 4.18 and Eqn. 4.19. The combined cost function that we seek to minimize is given as:

$$\mathrm{E_S} = (\mathrm{E_N}, \mu \mathrm{E_P}) \tag{4.21}$$

We aim to minimize (4.21) w.r.t $\mathbf{u}_C$, the displacement of the control handle. The

Jacobian relating the change of $\mathbf{u}_C$ to the change of $\mathrm{E_S}$ is given by:

$$\mathbf{J} = \frac{\partial \mathrm{E_S}}{\partial \mathbf{u}_C} = \begin{bmatrix} \frac{\partial \mathrm{E_N}}{\partial \boldsymbol{\xi}} \\ \mu \frac{\partial \mathrm{E_P}}{\partial \boldsymbol{\xi}} \end{bmatrix} \boldsymbol{\Gamma} \tag{4.22}$$

where:

$$\frac{\partial \mathrm{E_N}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} -\mathbf{n}_j^\top - \mathbf{n}_l^\top \Big(\mathbf{A} \Big[\mathbf{P}_{i+2} - \mathbf{P}_{i+1} \Big]_\times \Big) \\ -\mathbf{n}_l^\top \big(\mathbf{A} \Big[\mathbf{P}_i - \mathbf{P}_{i+2} \Big]_\times \Big) \\ -\mathbf{n}_l^\top \Big(\mathbf{A} \Big[\mathbf{P}_{i+1} - \mathbf{P}_i \Big]_\times \Big) \end{bmatrix}^\top \tag{4.23}$$

where

$$\mathbf{A} = \frac{1}{\|\overline{\mathbf{n}}_j\|} (\mathbb{1}_{3\times 3} - \mathbf{n}_j \mathbf{n}_j^\top) \tag{4.24}$$

given that:

$$\overline{\mathbf{n}}_j = (\mathbf{P}_{i+2} - \mathbf{P}_i) \times (\mathbf{P}_{i+1} - \mathbf{P}_i) \tag{4.25}$$

and $\mathbf{n}_l = \mathbf{p}_l^p - \mathbf{P}_i$.

On the other hand:

$$\frac{\partial \mathrm{E_P}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \nabla \boldsymbol{\mathcal{I}}_p^u & \nabla \boldsymbol{\mathcal{I}}_p^v \end{bmatrix} \begin{bmatrix} b_1 & 0 \\ 0 & b_1 \\ b_2 & 0 \\ 0 & b_2 \\ b_3 & 0 \\ 0 & b_3 \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_{\mathbf{P}_i} & & \\ & \mathbf{H}_{\mathbf{P}_{i+1}} & \\ & & \mathbf{H}_{\mathbf{P}_{i+2}} \end{bmatrix} \mathbb{1}_{9\times 9} \tag{4.26}$$

where:

$$\mathbf{H}_{\mathbf{P}_q} = \begin{bmatrix} \frac{f_x}{(\mathbf{P}_q)_Z} & 0 & -f_x \frac{(\mathbf{P}_q)_X}{(\mathbf{P}_q)_Z^2} \\ 0 & \frac{f_y}{(\mathbf{P}_q)_Z} & -f_y \frac{(\mathbf{P}_q)_Y}{(\mathbf{P}_q)_Z^2} \end{bmatrix} \forall q \in i, (i+1), (i+2) \tag{4.27}$$

and $\mathbf{B} = \Big(b_1, b_2, b_3\Big)$ are the barycentric coordinates.

The optimization method of our choice is Iteratively Re-weighted Least Squares (IRLS) solver, and the update is given by:

$$\Delta \mathbf{u}_\mathrm{C} = -\lambda \Big(\mathbf{W}\mathbf{J} \Big)^+ \mathbf{W} \mathrm{E_S} \tag{4.28}$$

where $\mathbf{W}$ is a weighting matrix using Tukey based M-estimator [Meer et al., 1991], and λ is a scaling factor. The new position $\mathbf{u}_\mathrm{C} + \Delta \mathbf{u}_\mathrm{C}$ of $\mathbf{C}_j$ is to be applied on the mechanical

model such that it deforms to minimize the errors from (4.18) and (4.19).

4.3.5 Implementing the Mechanical Model

The goal of this chapter is to formalize a method for dissociating the cost function $\mathbf{E_S}$ from the physical model $\mathcal{P}$ while using them together to solve a non-linear least squares problem. To this end, as indicated in Sec. 1.3.1, we intend to base our choice of $\mathcal{P}$ on FEM. In practice, we validate our approach using two mechanical models: a linear FEM and a co-rotational FEM using a tetrahedral mesh.

As indicated in section 4.3.3, the rigid tracking of the object is handled without using the mechanical model (i.e., only based on the surface model), thereby allowing the deformation estimation to tackle a better posed problem. The object is assumed to be attached to a fixed base via projective constraints on the vertices of $\mathbf{M}$ that lies on the ground plane, thereby eliminating the necessity of modelling friction or contact forces.

4.3.6 Force Tracking

With the deformation tracking methodology in place, it is possible to utilize this approach to online estimate the deforming forces acting on the object. To do this, two additional constraints are required: the material properties of the object being tracked, i.e., Young's modulus, Poisson's ratio, Rayleigh stiffness etc. and the approximate point of contacts on the object. The *control handles* derived in Section 4.3.2 are virtual points that may or may not have a physical meaning. However, to track forces, we need the actual points of application of force. All our force tracking examples have been performed by tracking the tool applying the force using external, fiducial markers (classical hand or finger tracking algorithms [Letessier and Bérard, 2004] [Hamer et al., 2009] can be used in unison with the approach proposed here to freely track applied forces on hand-manipulated objects).

Once a point $\mathbf{P}_C$ has been identified as the point of contact, its nearest mechanical mesh vertex $\mathbf{P}_C^{\mathrm{M}}$ is obtained using a nearest neighbor search. Thereafter, the force applied on $\mathbf{P}_C$ is given as $\left\| \sum_i \mathbf{F}_{\mathbf{P}_i^{\mathrm{M}}} \right\|$, for all indices i such that $\mathbf{P}_C^{\mathrm{M}}$ is a neighbor of $\mathbf{P}_i^{\mathrm{M}}$, wherein the force vectors $\mathbf{F}_{\mathbf{P}_i^{\mathrm{M}}}$ is derived using (3.3).

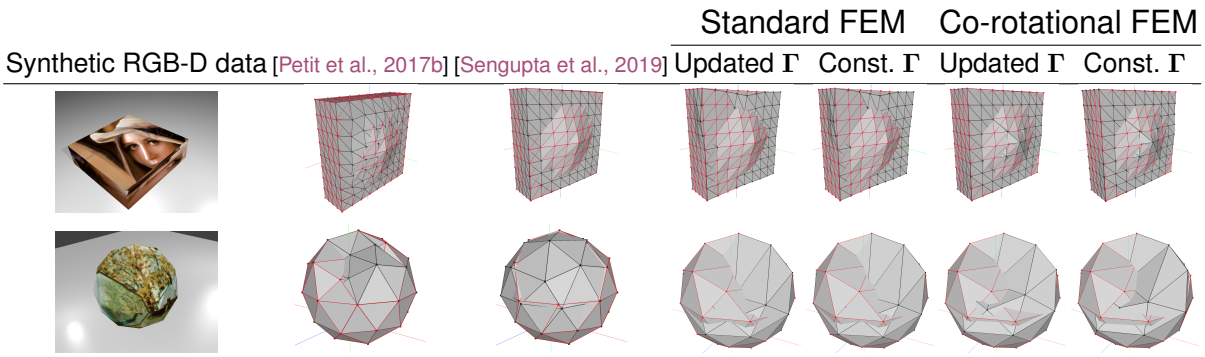


FIGURE 4.3 – Comparison between groundtruth (red edges) and tracking output (black edges) for the two synthetic sequences *cube* and *icosphere*. The red edges and vertices shows the object model from the groundtruth, the black ones are from the tracking output.

4.4 Results

To validate the deformation tracking methodology and to quantitatively compare with the state-of-the-art, we propose a synthetic RGB-D dataset of deforming objects. There are some dataset of non-rigid objects already available in the public domain. However, to quantitatively validate the approach proposed in this chapter, we require a dataset that has: *a)* color and depth data from the scene, *b)* a 3D CAD (or similar), textureless model or some 3D template of the object that needs to be tracked, and *c)* the object being tracked should be non-rigid and must undergo deformation without the presence of any articulated joints and without any change of topology (cutting, tearing etc.). Among the publicly available dataset, [Slavcheva et al., 2017b] is not applicable since the current approach does not handle modelling of articulated joints. [Jensen et al., 2018] is a 2D dataset, not suitable to our problem statement. The synthetic dataset from [Sengupta et al., 2019] does not have valid textural information.

Our proposed synthetic dataset consists of *two* objects undergoing non-rigid deformation, created using the Blender software [Blender, 2018]. This dataset will be made available publicly. For comparing the output of the algorithm with the groundtruth, the error metric of our choice is the Hausdorff distance between the object model from the 3D scene used for creating the data and the output of the algorithm.

In the implementation of our method, all the FEM models have been developed using the SOFA framework [Faure et al., 2012a]. For the physical model, the input to our system is a coarse 3D model of the surface of the object at the inital, un-deformed configuration. The volumetric, tetrahedral mesh is generated using Dirichlet tessella-

tion and by Delaunay tetrahedralization of the input mesh using the Bowyer-Watson algorithm [Si, 2015]. The initialization of ${}^{C}\mathbf{T}_{O}$ at the very first frame is done by matching pre-trained markers with those extracted from the initial images, similar to the approach of [Choi and Christensen, 2010]. Since we track the model frame-to-frame, no segmentation is required for this approach.

Frame 4: Re-implementation of [Petit et al., 2015a]

In [Petit et al., 2015a], the images from the RGB-D sensor are segmented to isolate the object to be tracked using a *grabcut* approach, which is in turn utilized to extract a segmented pointcloud. To track the deformation of this object, an external force is applied to the mechanical model (simulated with FEM) of the object. The per-vertex force exerted is given by:

$$\mathbf{F}_i^{\text{ext}} = \boldsymbol{\zeta}_i k(\mathbf{P}_i^{\mathcal{V}_m} - \mathbf{P}_i^{\boldsymbol{\psi}}) \tag{4.29}$$

where $\mathbf{F}_i^{\text{ext}}$ is the external force vector for the i-th element of $\mathcal{V}_m$ which is given by $\mathbf{P}_i^{\mathcal{V}_m}$, $\boldsymbol{\zeta}_i = \sigma_1 \mathrm{e}^{-(\frac{d_i}{\sigma_2})}$ such that d_i is the closest distance of $\mathbf{P}_i^{\mathcal{V}_m}$ from the projected 2D contour of the object, σ_1 and σ_2 are two empirically chosen constants, k is the stiffness of the element corresponding to the i-th node, and the variable $\mathbf{P}_i^{\boldsymbol{\psi}}$ is obtained from the segmented pointcloud $\boldsymbol{\psi}$ by the formulation:

$$\mathbf{P}_i^{\boldsymbol{\psi}} = \begin{cases} \alpha \mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m}) + (1-\alpha)\frac{1}{\#\boldsymbol{\psi}_i} \sum\limits_{\mathbf{P}_j \in \boldsymbol{\psi}_i} \mathbf{P}_j & \text{if } \#\boldsymbol{\psi}_i > 0 \\ \alpha \mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m}) + (1-\alpha)\mathbf{P}_i^{\mathcal{V}_m} & \text{else} \end{cases} \tag{4.30}$$

where $\mathrm{NN}_{\boldsymbol{\psi}}(\mathbf{P}_i^{\mathcal{V}_m})$ denotes the nearest neighbor of $\mathbf{P}_i^{\mathcal{V}_m}$ in $\boldsymbol{\psi}$, $\boldsymbol{\psi}_i$ is a subset of $\boldsymbol{\psi}$ such that $\forall (\boldsymbol{\psi}_i)_j \in \boldsymbol{\psi}_i$ there exists $\mathrm{NN}_{\mathcal{V}_m}((\boldsymbol{\psi}_i)_j) = \mathbf{P}_i^{\mathcal{V}_m}$ and α is a tunable parameter. Some additional weights are assigned to the vertices based on the distance to the nearest contour of the mesh, as visible from the projection of the model on the image plane. We re-implement this method from [Petit et al., 2015a], while implementing co-rotational FEM using SOFA library and repeat the experiments while fine-tuning the empirical parameters. We keep only the best results among all the experiments. Moreover, to avoid ambiguity, we restrict the comparison to dataset with small deformation only (deformation confined inside the contour of the projected model).

		Cube	Icosphere
Std. FEM	Constant $\mathbf{\Gamma}$	5.76 %	**4.83** %
	Updated $\mathbf{\Gamma}$	5.75 %	5.42 %
CR FEM	Constant $\mathbf{\Gamma}$	**2.14** %	5.25 %
	Updated $\mathbf{\Gamma}$	**2.14** %	5.34 %
[Petit et al., 2017b]		2.93 %	7.93 %
[Sengupta et al., 2019]		3.28 %	6.67 %

TABLE 4.1 – Comparison of tracking accuracy for the two synthetic sequences in terms of Hausdorff distance between tracking output and groundtruth. Standard (Std.) and co-rotational (CR) FEM has been tested while maintaining $\mathbf{\Gamma}$ constant, as well as while updating it numerically in every iteration. The same sequences have been compared with [Petit et al., 2017b] and [Sengupta et al., 2019] in the last two rows. The Hausdorff distances are expressed as percentage of the longest side of the 3D bounding box of the model of the object in undeformed configuration.

The assumption introduced in Eqn. 4.15 (i.e., $\mathbf{\Gamma}$ for a particular mesh vertex can be thresholded, binarized and kept constant throughout a sequence of deformation) forms the theoretical basis of the entire methodology proposed in this chapter. We demonstrate that the difference in accuracy between maintaining the *influence matrix* $\mathbf{\Gamma}$ constant (after it has been computed only once during the initial estimation step) and re-computing $\mathbf{\Gamma}$ at each frame numerically is not significant. To establish this proposition, we run tests on the synthetic data with and without holding $\mathbf{\Gamma}$ constant for both standard, linear FEM and co-rotational FEM. The results are summarized in Table 4.1. It can be clearly seen that the variation in accuracy of tracking is between 0 to 0.59 % for both the sequences tested here. This comes in exchange for a large improvement in runtime ($> 53\%$ improvement in per frame time requirement, see Table 4.2) of the entire algorithm, since multiple FEM simulations per iteration of the IRLS solver is highly expensive. Fig. 4.3 shows the visual comparison of all the approaches tested on the synthetic sequences.

For the synthetic dataset, the approach proposed in this chapter outperforms the accuracy of the state-of-the-art methods for physical model based tracking proposed in [Petit et al., 2017b] and [Sengupta et al., 2019]. Table 4.1 describes the variance in the accuracy when the two synthetic sequences were tested with [Petit et al., 2017b] and [Sengupta et al., 2019]. The proposed deformation tracking approach is highly tolerant to variance in the estimate of the Young's modulus (YM) and Poisson's ratio (PR) used to model the FEM. While varying the YM between 5000 Pa to 5 MPa, the

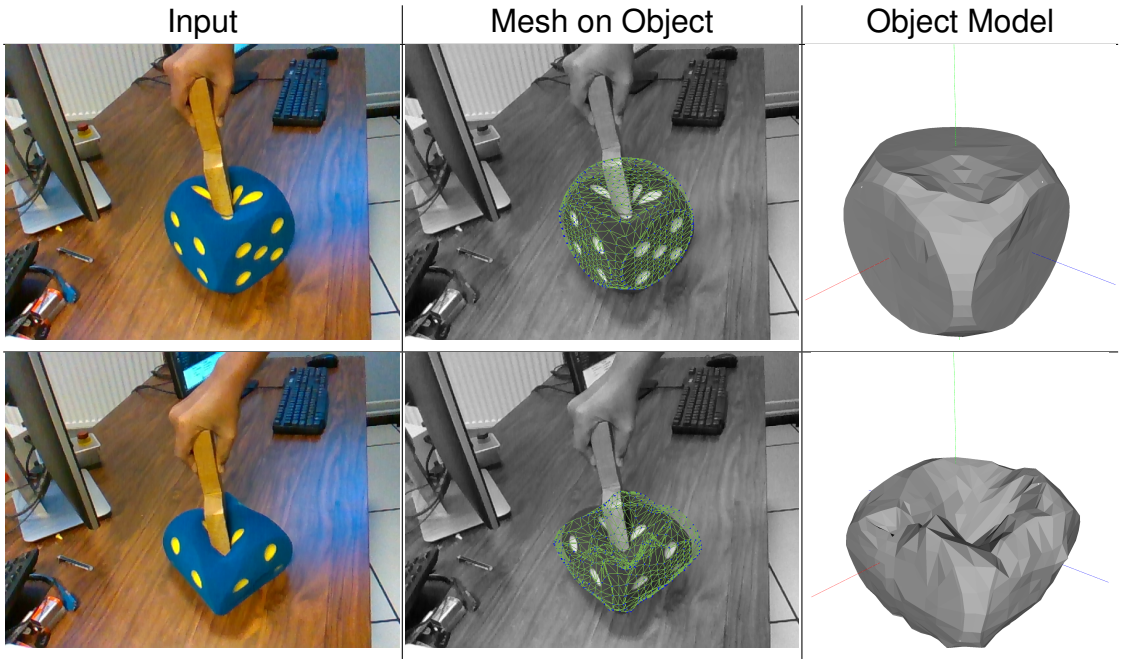
accuracy varied by less than 0.001%, which is practically negligible. While varying the Poisson's ratio between 0.2 and 0.5, the accuracy drifted by $< 0.129\%$.

To validate our approach on real objects, we utilized two objects: a deformable *dice* made of foam and a block of *sponge* in the shape of a cuboid. The results are visually demonstrated in Fig. 4.4. The tracking results are visually coherent while suffering occlusions/outliers in the range of ± 5 cm. The *dice* was compressed for more than 14 cm ($> 87\%$ of its own height), but our approach still tracked the entire deformation accurately.

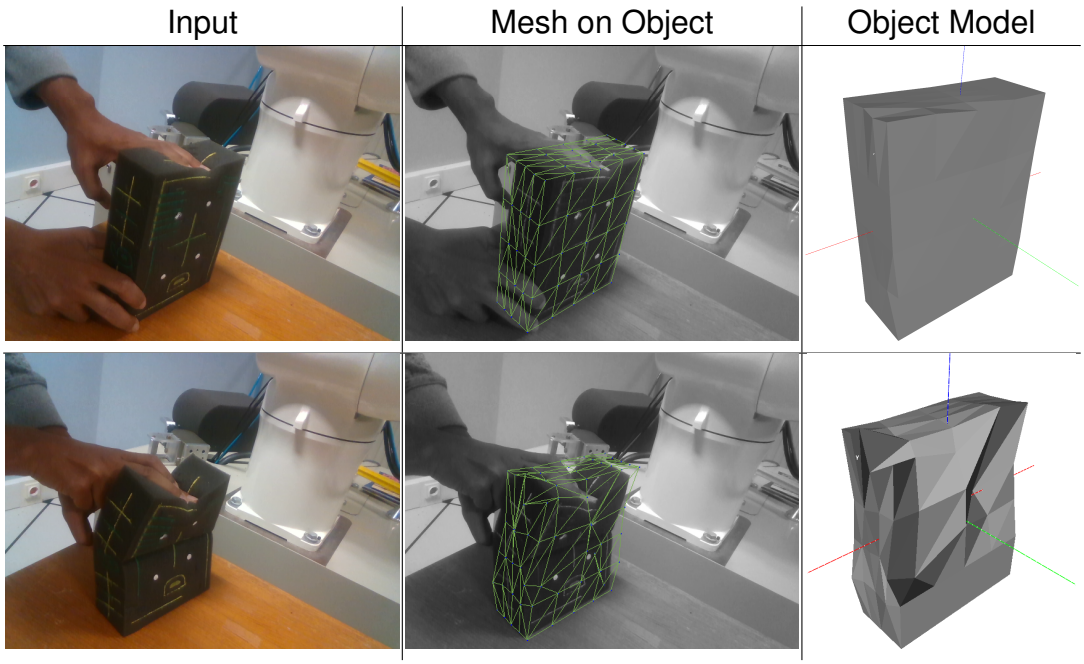
4.4.1 Validation of Force Tracking

To validate the force tracking methodology, we use a 6-DOF anthropomorphic robot arm (a Viper 850 from ADEPT) fitted with a ATI Gamma IP65 force/torque sensor and a 3D-printed stylus as an end-effector distal tool. The robot is used only to utilize its force sensor to obtain a groundtruth of the force that is being applied on the object. To get a proper estimate of the force using the proposed method, we need to know the material property (Young's modulus, Poisson's ratio, etc.) of the object, and the point of contact needs to be measured separately. To demonstrate this feature, we use the *sponge* and a *ball* made of foam. First, the Young's modulus of the *sponge* and the *ball* is determined by repeated indentation tests. It is determined that the *sponge* and the *ball* has a Young's modulus of 460 kPa and 160 kPa respectively. Next, these objects are subjected to a strong deformation using the robot's end-effector, while our proposed method is used to track the deformation and the force applied simultaneously. The 3D printed end-effector is tracked using pre-trained markers from the image data of the RGB-D camera. The mesh vertex nearest to the tip of the end-effector gives us the point of contact. The results of the force tracking approach are summarized in Fig. 4.5.

We observe a force tracking accuracy of $\sim 97\%$ for the *sponge* and of $\sim 90\%$ for the *ball*, when the deformation is optimally observable. For very small deformations (< 1 cm), the sensor noise causes small variances in the force measurement while at very large deformation, the tip of the end-effector occludes the deformed surface to a very large extent, causing the accuracy to decrease in the experiment with the *sponge*. However, with properly observable deformations (> 1 cm and free from large occlusions), the approach is very accurate. The measured forces show a consistent Pearson correlation coefficient of > 0.92 with the actually applied forces over a large



(a) Deformation tracking results from the *dice*



(b) Deformation tracking results from the *dice*

FIGURE 4.4 – Results from deformation tracking of a *dice* (a) and a *sponge* (b). Due to the presence of the physical model (co-rotational FEM), it is possible to track large deformations without producing any un-natural deformation in the object's mesh model

number of experiments.

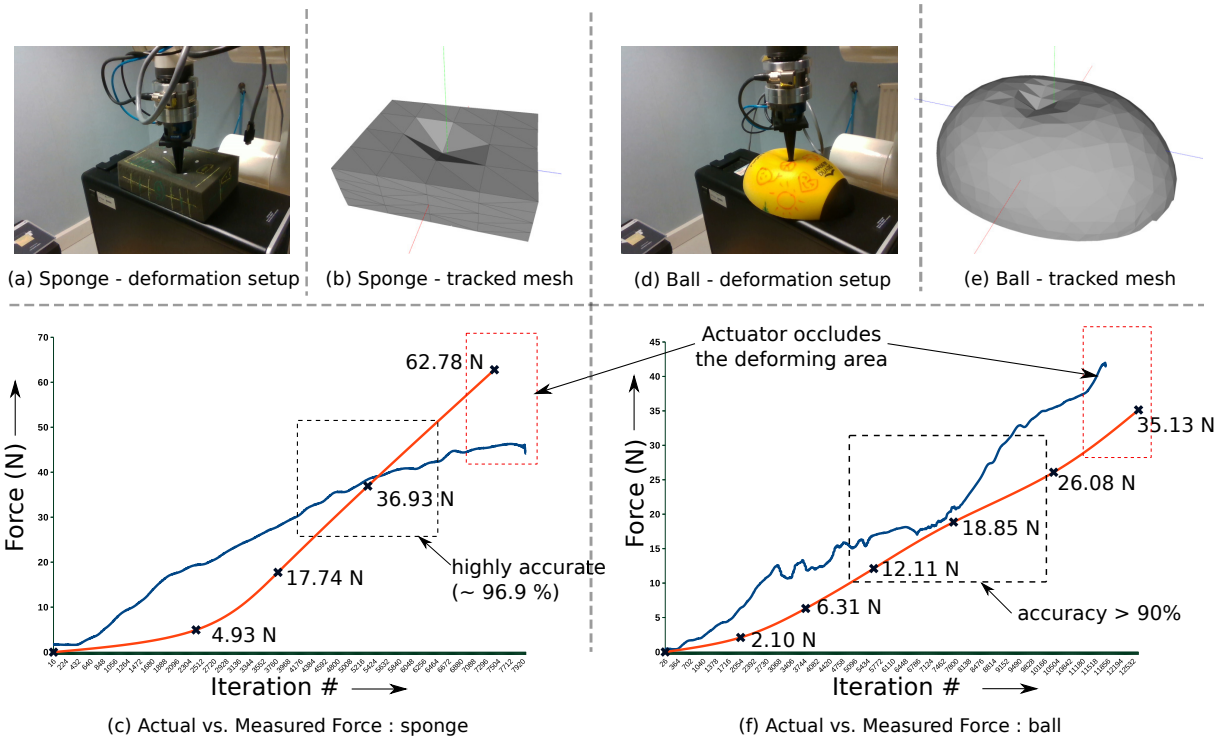


FIGURE 4.5 – Setup and results for force tracking on the *sponge*. The Viper 850 robot is being used deform the object while being tracked. The red plot is the force measurement from the robot's force sensor, the blue plot is the estimated force reported by the proposed approach

4.4.2 Runtime Evaluation

The experiments discussed in this chapter were performed on a Intel Xeon CPU working at 3.70GHz. Utilizing only a single core of the computer, the un-optimised code was able to achieve a runtime of 350 msec - 550 msec/frame while tracking deformations using the constant value of Γ, showing that it can achieve real-time performance at frame rate. The time taken to estimate the value of Γ for each node is approximately 0.5 sec - 0.6 sec per frame, which is a one time cost that can be either done offline or computed when needed, as per the requirements of the application. The runtime for the real data used in this chapter is summarized in Table 4.2.

		Constant $\mathbf{\Gamma}$	Updated $\mathbf{\Gamma}$
Std. FEM	Dice	0.505 sec	1.401 sec
	Sponge	0.492 sec	0.825 sec
CR FEM	Dice	0.393 sec	1.007 sec
	Sponge	0.465 sec	0.734 sec

TABLE 4.2 – Comparison of the per frame time requirement for deformation tracking with constant and updated values of $\mathbf{\Gamma}$ using Standard (Std.) and Co-rotational (CR) FEM

4.5 Conclusion

The chapter presented here describes a new method for combining visual error minimization with FEM to create an accurate and fast deformation tracking method. The algorithm has been tested on synthetic and real data and has been shown to outperform state-of-the-art methods in tracking accuracy for generic deforming objects. The algorithm proposed here can be extended to other, more complex physical models without loss of generalization. The method proposed here has been demonstrated as a reliable visual force tracking system, when the material properties of the object being tracked are available. We believe that there is a large potential for such applications in the field of robotics and augmented/mixed reality.

In this dissertation, we have so far presented two different approaches for deformation tracking. In the next chapter, we are going to present some applications of deformation tracking, especially highlighting the benefits of using a FEM based approach as the primary deformation model. We will present a robotic application involving estimation of elasticity parameters of a deforming object while it is being tracked by a RGB-D camera.

CHAPTER 5

ROBOTIC APPLICATIONS FOR NON-RIGID OBJECT TRACKING

The preceding chapters dealt with different mechanisms to track deforming object using mechanical model. As indicated in Chapter 1, the usage of mechanical models enable us to re-utilize these deformation tracking methodologies into some interesting use-cases, which would not have been straightforward using physical model-free deformation tracking approaches. In this chapter, we take a look into some of the practical applications that arises out of using physical models for deformation tracking. More specifically, we describe a method to estimate the physical properties of an object using the deformation tracking algorithm and an external force sensor mounted on a robotic arm. Later in this chapter, we extend that approach to track freely applied forces on the object, once the physical properties have been determined.

Robotic manipulation of rigid objects is a well-studied and well-developed area of research. However, non-rigid object manipulation using robotic end-effector is a more difficult problem to address. Part of the problem arises from the fact that real-time tracking of non-rigid objects had been an unsolved problem until the last few years. Even with the recent development of non-rigid object tracking methods using computer vision, there remains significant scope for improvement in tracking the entire surface model of the object under challenging conditions. Another important question arising while interacting with soft objects is the determination of the elasticity parameters of the object, especially the Young's modulus. Simultaneous manipulation and tracking of deformable objects could improve different use-cases such as humanoid robots interacting with soft objects, compliance testing in product manufacturing or automated, preventive healthcare [Stults, 1984]. The aim of this approach is to simultaneously track and estimate the elasticity parameters of a soft object that is being deformed by the end-effector of a robot. In this chapter, we propose a closed-loop method consisting of: **a)** a deformation tracking method and **b)** an elasticity estimation algorithm. This

closed-loop method allows to achieve two objectives: *STEPE* (Simultaneous Tracking and Estimation of Parameters of Elasticity) and *remote force estimation*. The required inputs of *STEPE* are a coarse 3D geometric model (mesh) of the object to track and the external measurements of the forces deforming the object. The deformation tracking method uses a depth sensor to capture the deformation and tracks it using a physics-based simulation of the object deformations. The elasticity estimation algorithm uses the result of the deformation tracking method and measurements of the deformation forces, obtained from an external force sensor, in order to estimate the elasticity parameters of the object. Once the elasticity estimation is achieved, the parameter can be used by the deformation tracking method, thereby closing the loop. The method will iteratively converge towards a correct estimation of the elasticity parameters of the object. The elasticity parameters obtained at the convergence of the method are thereafter used for *remote force estimation*. From this stage, the deformation tracking method can be used to estimate the deformation forces acting on the object without any external force sensor.

The main contributions of this chapter are summarized below:

— We propose a novel method to simultaneously track the deformation of soft objects and estimate their elasticity parameter. The tracking of the deformable object is performed by combining the visual information acquired with a RGB-D sensor with interactive simulation of the object deformation based on FEM ;

— In parallel, the elasticity parameter estimation minimizes the error between the tracked object and a simulated object deformed by the forces that are measured using a force sensor ;

— Once the elasticity parameters are estimated, our tracking algorithm can then also be used to estimate the deformation forces applied on an object without the use of a force sensor

5.1 Background

As discussed in Sec. 1.3, non-rigid object tracking can be done using either a physics-based model, or a geometry-based model for computing the deformations. Both physics-based model such as [Newcombe et al., 2015, Innmann et al., 2016] or geometric-based models such as [Bronte et al., 2017, Bronte et al., 2014] offer mecha-

nism to track complex 3D shapes without utilizing any kind of machine-learning based approach. However, iteratively refining the physical parameters of the object being tracked is not possible using any of these techniques. Zhang et al. use a RGB-D camera to track the external forces applied on an object, but they utilized very specific soft-robots for the purpose, instead of doing it for any arbitrary deformable objects [Zhang et al., 2019]. The approach of Petit et al. [Petit et al., 2018] is better suited for the purpose of deformation tracking using a physics-based model. However, since it relies solely on depth data and a point-to-plane correspondence matching scheme to determine the direction and magnitude of deformation, this approach is more susceptible to error induced from incorrect correspondences due to heavy occlusions. Among the machine learning based approaches for deformation tracking, Varol et al. use a constrained latent variable model for learning the deformational behavior of an object from its latent state to an output state [Varol et al., 2012]. The results from this approach has however only been demonstrated on planar objects. From the literature, the methodology proposed by Frank et al. [Frank et al., 2010] has some resemblance to the method proposed in this chapter for simultaneously tracking deformation and estimating parameters. The approach utilizes a combined tracking and elasticity estimation modules using a point-to-point ICP [Besl and McKay, 1992] for pointcloud registration along with a linear, tetrahedral FEM as the deformation model. The use of linear FEM makes this approach not suitable when large rotational deformations occur. Additionally, point-to-point ICP is not tolerant towards occlusion adequately, which compels the authors to utilize a narrow wooden probe (attached to the end-effector) to avoid occlusion. To the best of our knowledge, this approach has not been used as a force estimator.

Manipulation of soft objects becomes an easier problem statement when the elasticity parameters of the objects are known. Sedef *et al.* proposed a method using a haptic device and a force sensor to compute the linear viscoelastic parameters of a homogeneous soft biological tissue [Sedef et al., 2006]. Bickel *et al.* proposed a method for handling elastic objects [Bickel et al., 2009] of heterogeneous composition. A soft object, monitored by a marker-based stereovision system, is stimulated by a probe equipped with a force sensor to estimate the Lamé's coefficients of heterogeneous deformable objects. Data-driven methods are also available to determine elasticity parameters of heterogeneous elastic objects. Wang *et al.* proposed an approach that iteratively performs deformation tracking and parameter estimation using a physics-based model with the help of a two-step optimization method [Wang et al., 2015]. A method

based on static Euler-Bernouilli deformation model along with a tracking methodology using multi-section parameterization, was developed for simultaneous tracking and estimation of deformable objects in [Fugl et al., 2012]. This method has two assumptions: the first is that the object is homogeneous and linearly elastic, the second assumption is that the bending of the object is subject to gravity. However, this method has only been tested on planar object.

In this chapter, we propose the simultaneous tracking and elasticity parameter estimation of arbitrary soft objects using the same physical model, which can be refined iteratively, as opposed to [Frank et al., 2010]. The deformation tracking methodology used here is similar to Chapter 3. It does not use any fiducial markers on the deforming object. An approximate model of the object is enough for both deformation tracking and elasticity parameter estimation. As opposed to [Zhang et al., 2019], the *remote force estimator* does not require a prior knowledge of the elasticity parameters. The *STEPE* method can run on inexpensive hardware requiring only a consumer-grade RGB-D sensor and force measurements.

This chapter is divided into three sections. The next section details the proposed methodology. The results are presented next. The discussions and the concluding remarks are conducted in the last two sections.

5.2 Elasticity Estimation from Deformation

The objective of the methodology proposed in this chapter is to track a deformable object while estimating its elasticity parameters simultaneously. In this chapter, we achieve this objective using a method, that we named *STEPE* (Simultaneous Tracking and Estimation of Parameters of Elasticity), made of three different modules: one performing the deformation tracking, one estimating the elasticity parameters and one measuring the external deformation forces applied on the object. In *STEPE*, all the modules are involved. When doing *remote force estimation*, only the module using the deformation tracking method is considered. The overview of the principle of the method is depicted in Fig. 5.1.

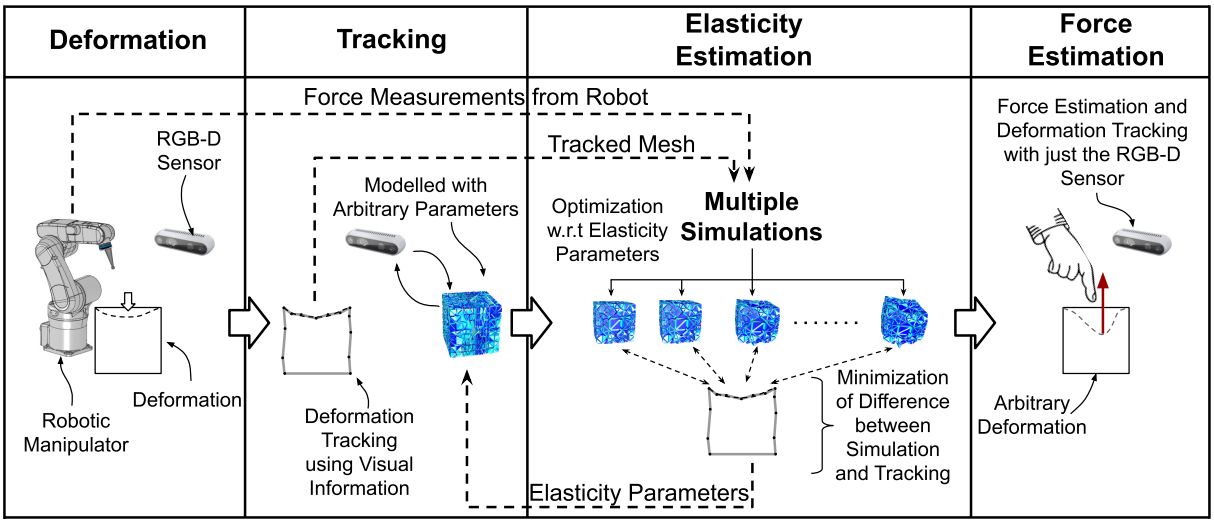


FIGURE 5.1 – Overview of the STEPE and Remote Force Estimation architecture.

5.2.1 Modeling

The deformation model proposed for this method is similar to Chapter 3. The co-rotational formulation of FEM [Müller and Gross, 2004] is used to avoid large deformational artifacts. The fundamental principle involves the *Hooke's law* that relates the stress and the strain applied on an object using a linear equation that depends on the elastic parameters of the object, i.e., the Young's modulus and the Poisson ratio.

Let the number of vertices of the mesh $\mathcal{V}^{\mathrm{M}}$ be n, $\mathbf{M}$ the mass matrix, $\mathbf{D} \in \mathbb{R}^{3n\times 3n}$ the damping matrix and $\mathbf{K} \in \mathbb{R}^{3n\times 3n}$ the stiffness matrix which depends on the Young's modulus and Poisson ratio of the entire object. Let $\mathbf{p}(t) \in \mathbb{R}^{3n}$ denote the position of each vertex of the mesh at time t, $\mathbf{f}^{ext} \in \mathbb{R}^{3n}$ the external forces applied onto the object and $\mathbf{f}^{int} \in \mathbb{R}^{3n}$ the internal forces resulting from the deformation. The linear algebraic equation of motion of an object is given by Eqn. 3.4. This equation is used to estimate the elasticity parameters from the deformation observations as well as to compute the external deformation forces from deformation observations when the elasticity parameters have converged.

5.2.2 STEPE

The *STEPE* methodology consists primarily of three sub-modules: the *first* one measures the external forces and controls the robotic end-effector to deform the object

gradually by applying a pre-determined displacement along a direction which minimizes the geometric error defined on the surface model of the object. *Second* module tracks the object using the deformation tracking methodology that will be described here. *Third* module estimates the elasticity parameters of the object using the output of deformation tracking and with the sensor data obtained from the measurement module. We now describe these modules in details.

5.2.2.1 External Force Measurements Module

The external measurements module is responsible for measuring the deforming forces that are applied on the tracked object. To ensure that the experiments are repeatable, we decided to use a robotic arm equipped with a force sensor in order to both manipulate the object and measure the resulting forces. The end-effector deforms the object by sequential, compressive motions approximately along the direction of the normal to the surface of contact while recording the forces that are exerted. The end-effector motion is stopped at a certain point in order to maintain a static deformed state. This module stores the external force measurements and timestamps at which they were acquired in order to transmit them to the estimation algorithm when a steady-state has been reached in the deformation tracking.

5.2.2.2 Deformation Tracking Method

In this chapter, we implemented a deformation tracking method similar to the one proposed in [Sengupta et al., 2019]. The initial pose of the object(O) w.r.t the camera (C) is obtained in the first image frame using pre-trained markers with an approach similar to [Marchand et al., 2016]. The pose of the camera w.r.t the object, denoted by the homogeneous transformation matrix ${}^{C}\mathbf{T}_{O}$ and is updated at the beginning of each frame, using an approximate rigid object tracking method with the assumption that the object is rigid. The rigid tracking is done with the technique described in Section 3.2.3 of Chapter 3.

The geometric residual error is computed according to Eqn. 3.5. As given in Sec. 3.2.4, the minimization of this residual error by regulating the virtual forces acting on the surface of the mechanical object model results in the deformation tracking. The Jacobian ($\mathbf{J}$) is computed using the axial perturbation method, which is somewhat similar to the approach demonstrated in shown in Fig. 3.3. Every node is subjected to a small

force given by $\Delta\mathbf{F}_x$, $\Delta\mathbf{F}_y$ and $\Delta\mathbf{F}_z$, acting along the three axes. The displacement of all vertices of the mesh are computed by solving Eqn. 3.4 using an Euler-Implicit mechanism of a conjugate gradient based linear solver [Faure et al., 2012a]. The Jacobian $\mathbf{J}$ is computed using finite differences in the value of e^{N} obtained from the three deformed configurations. The only difference between the Jacobian computation in this chapter w.r.t the method described in Chapter 3.2.4 is the modification of the axial perturbation from central-differences to forward-finite differences, which results in performance optimization.

Once the Jacobian $\mathbf{J}$ relating the variation of e^{N} w.r.t the variation of the force vector $\mathbf{F}$ is obtained, the force update is given by:

$$\Delta\mathbf{F} = -\boldsymbol{\lambda}\Big(\mathbf{W}\mathbf{J}\Big)^{+}\mathbf{W}\mathrm{e}^{\mathrm{N}}({}^{k-1}\mathbf{P}) \tag{5.1}$$

where $\mathbf{W}$ is a weighting matrix from Tukey based M-estimator [Meer et al., 1991], and

$$\boldsymbol{\lambda} = \begin{pmatrix}\boldsymbol{\lambda}_X & \boldsymbol{\lambda}_Y & \boldsymbol{\lambda}_Z\end{pmatrix} \tag{5.2}$$

is a scaling factor. The deformation caused in the mesh due to the Jacobian computation is discarded and the update:

$${}^{k-1}\mathbf{F} = {}^{k-1}\mathbf{F} + \Delta\mathbf{F} \tag{5.3}$$

is applied on the node, where ${}^{k-1}\mathbf{F}$ is the external force already existing on the node at the previous image frame. This Iteratively Re-weighted Least Squares scheme is allowed to repeat for a fixed number of iteration at every image frame.

These estimated forces using Eqn. 5.3 result in deformation of the mechanical mesh using the co-rotational FEM formulation, as described in Sec. 1.3.1. The vertices of the mesh $\mathcal{V}^{\mathrm{M}}$ obtained by the procedure given above, i.e., tracking of the observed RGB-D data, is denoted by $\mathbf{p}^*$, and will be used for elasticity parameter estimation, as described in the following section.

5.2.2.3 Estimation algorithm

In this chapter, we chose to use the Hooke's law evoked in 5.2.1 to represent the deformable objects. Consequently the elasticity parameters that can be estimated are the Young's modulus and the Poisson ratio. In order to estimate the elasticity parameters

at a contact point of a deformable object, the algorithm relies on the tracking results $\mathbf{p}^*$ and the external measurements of the deformation forces $\mathbf{f}^*$ taken at the same point. Let $\mathcal{V}_v^{\mathrm{S}}$ (the visible mesh and vertices are denoted by the suffix $(\cdot)_v$) be the vertices that the camera can see at any given frame (determined by a raytracing algorithm) and $\mathbf{p}_v$ denotes every vertex of the visible mesh. A cost-function $\mathrm{e}(\mathcal{V})$ is minimized using the Levenberg-Marquardt algorithm [Moré, 1978] for updating the value of $\boldsymbol{v}$, the Young's modulus of the object, and is given by:

$$\mathrm{e}(\mathcal{V}) = \sum_{\mathbf{p}_v \in \mathcal{V}_v^{\mathrm{S}}} \mathrm{e}_v(\boldsymbol{v})^2 \tag{5.4}$$

In this expression, e_v is the error function defined such as:

$$\mathrm{e}_v(\boldsymbol{v}) = \|\mathbf{p}_v^* - \mathbf{p}_v^s(\mathbf{f}^*, \boldsymbol{v})\| \tag{5.5}$$

and is equal to the Euclidean distance between $\mathbf{p}_v^*$ and $\mathbf{p}_v^s$, which are respectively the tracked and simulated 3D positions of $\mathcal{V}_v^{\mathrm{S}}$.

$\mathbf{p}_v^*$ is taken in at the final deformed state at the end of tracking and $\mathbf{p}_v^s$ at the end of the simulation. The estimated parameters are the ones for which the cost-function $\mathrm{e}(\mathcal{V})$ is minimized.

The simulation applies forces on the simulated object, which replicate the ones that were recorded during the experiment. The Jacobian of the cost-function is needed to compute the update of the elasticity parameters between two steps k and $k+1$ of the minimization algorithm. This Jacobian is numerically estimated by running several corotational FEM simulations using different values of elasticity parameters and computing the error between the experimental data and the simulation result. Let $\Delta\boldsymbol{v}(k)$ be the update of the elasticity parameters at the k-th iteration step, $\bigtriangledown\mathbf{e}$ be the Jacobian at this step of the function $\mathbf{p}^s(\mathbf{f}^*, \boldsymbol{v})$ w.r.t. the elasticity parameters and μ be a damping factor that is adjusted at each Levenberg-Marquardt step. The update of the elasticity parameters can be computed by solving for $\Delta\boldsymbol{v}(k)$ the linear equation :

$$\Delta\boldsymbol{v}(k) = \Big(\bigtriangledown\mathbf{e}^\top \bigtriangledown\mathbf{e} + \mu\ \mathrm{diag}\Big(\bigtriangledown\mathbf{e}^\top \bigtriangledown\mathbf{e}\Big)\Big)^{-1}\Big(\bigtriangledown\mathbf{e}^\top\mathbf{e}\Big) \tag{5.6}$$

$$\boldsymbol{v}(k+1) = \boldsymbol{v}(k) + \Delta\boldsymbol{v}(k) \tag{5.7}$$

In this chapter, we focus on estimating the Young's modulus . The Poisson ratio is assumed to be known a priori. Thus, the elasticity parameters vector is given by $\boldsymbol{\upsilon} = (\mathrm{E}) \in \mathbb{R}$. At this stage, the estimated value of Young's modulus obtained from the *STEPE* method described above is defined by $\hat{\mathrm{E}}$.

5.2.3 Remote force estimation

When doing *remote force estimation*, the deformation tracking module can be used without external force sensor. Let A be the set of vertices representing the active region of the object. The active region is an approximation of the surface of the model where the deformation is happening. This region is determined using some practical heuristics e.g., neither the vertices lying on the ground plane nor the vertices not visible from the camera are taken into account. Active vertices, denoted by a, are the vertices belonging to the active region. The method is able to estimate the external deformation forces, given by:

$$\widehat{\mathbf{f}^{ext}} = \sum_{a} \widehat{\mathbf{f}_a^{ext}} \tag{5.8}$$

which are applied on the object in the active region, as shown in Fig. 5.2. The estimated forces are projected onto the normal of the active surface:

$$\mathbf{n}^A = \frac{\sum_a \mathbf{n}_a}{\| \sum_a \mathbf{n}_a \|} \tag{5.9}$$

because the tracking method cannot have information about tangential forces, since (4.18) is a point-to-plane distance. This produces the following estimate of the magnitude of the deformation forces:

$$\|\widehat{\mathbf{f}^{ext}}\| \approx \widehat{\mathbf{f}^{ext}} \cdot \mathbf{n}^A \tag{5.10}$$

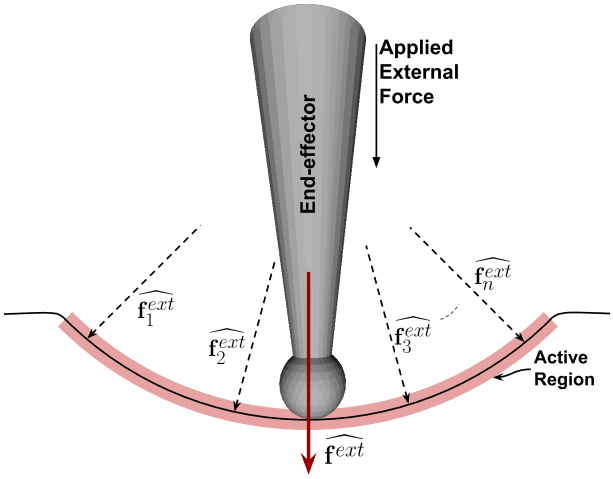


FIGURE 5.2 – Force estimation in the active region.

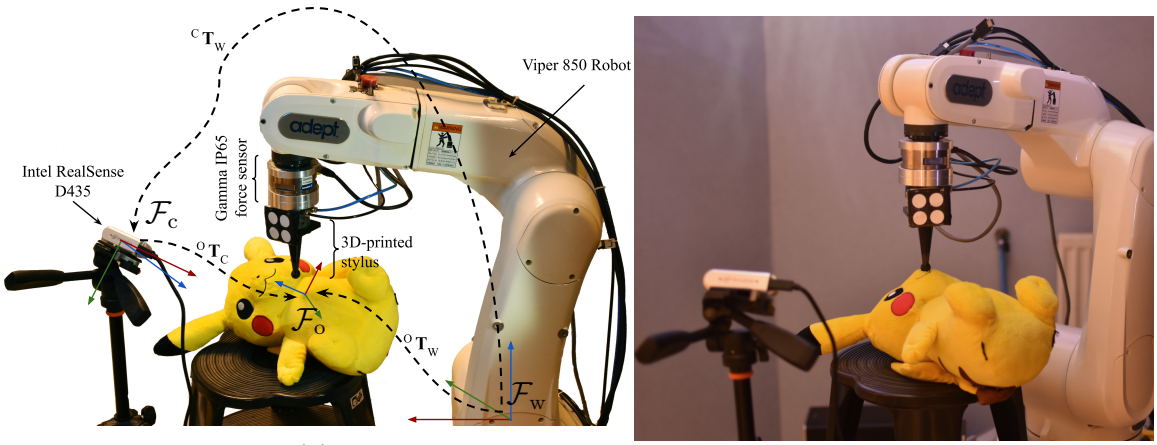
5.3 Experiments

5.3.1 Setup

The experimental setup shown in Fig. 5.3 consists of a 6-DOF anthropomorphic robot arm (a Viper 850 from ADEPT) equipped with a ATI's Gamma IP65 force/torque sensor and a 3D-printed stylus used as an end-effector distal tool. A RGB-D Intel Realsense D435 camera is used for the tracking. The implementation has been done on a computer that has an Intel Xeon CPU working at 3.70GHz with 16 logical cores. We tested our method with different soft objects: a foam block, a soft ball and a complex-shaped plush toy. The transformation between the robot frame and the object frame is computed only once, i.e., at the initialization of the method. Thereafter, it is utilized to express measured forces into the coordinate frame of the object. The mesh of the plush toy has been generated by photogrammetry using the Meshroom software [Moulon et al., 2012, Jancosek and Pajdla, 2011]. The corotational FEM simulations are performed using SOFA framework [Faure et al., 2012a].

5.3.2 Results

We now describe a series of experimental results and the necessary implementation details associated with these experiments. The experiments are categorized into certain *conditions*. Each of these *condition* represents a particular position for a speci-



(a) (b)

FIGURE 5.3 – The experimental setup shown with and without annotation in Fig. 5.3a and Fig. 5.3b. The robot (at reference frame $\mathcal{F}_{\mathrm{w}}$) is deforming a plush toy (at reference frame $\mathcal{F}_{\mathrm{o}}$) while the data is being captured from an Intel RealSense D435 RGB-D Camera (at reference frame $\mathcal{F}_{\mathrm{c}}$). The transformation between these reference frames are shown in Fig. 5.3a, following the convention established in Sec. 1.1.1

fic object. E.g., the first *condition* described below denotes the rectangular foam being deformed at the center of its largest face, while being placed horizontally on a table. To evaluate the accuracy of the estimation process, the ground truth of Young's modulus E_{GT} has been determined through indentation tests for each condition of experiments [McKee et al., 2011]. The indentation tests can be described as: repeated, slow and incremental vertical displacements applied on the objects using the robot while measuring the forces and the displacements. Forces are measured from the force sensor fitted in the robot's arm while displacements are obtained using the odometry of

TABLE 5.1 – Evaluation of tracking and estimation times and Young's modulus (in kPa) estimation accuracy w.r.t. different quality of visual and mechanical meshes.

Objects	No. of vertices in Visual Mesh	No. of vertices in Mechanical Mesh	$\bar{t}_{tr}(\mathrm{s})$	$\mathrm{std}(t_{tr})(\mathrm{s})$	$t_{st}(\mathrm{s})$	$t_{\mathrm{est}}(\mathrm{s})$	$\bar{\mathrm{E}}_{\mathrm{GT}}$	$\mathrm{E}_{\mathrm{est}}$	Error(%)
foam	35	1049	1.94	0.14	62	310	454	431	5.08
foam	415	1049	2.37	0.15	122	180	454	438	3.62
foam	178	554	1.21	0.02	60	180	454	497	9.47
ball	404	627	1.38	0.03	126	70	156	136	12.8
ball	404	1060	1.61	0.02	118	96	156	148	5.0
ball	404	1954	3.09	0.07	130	330	156	148	5.1

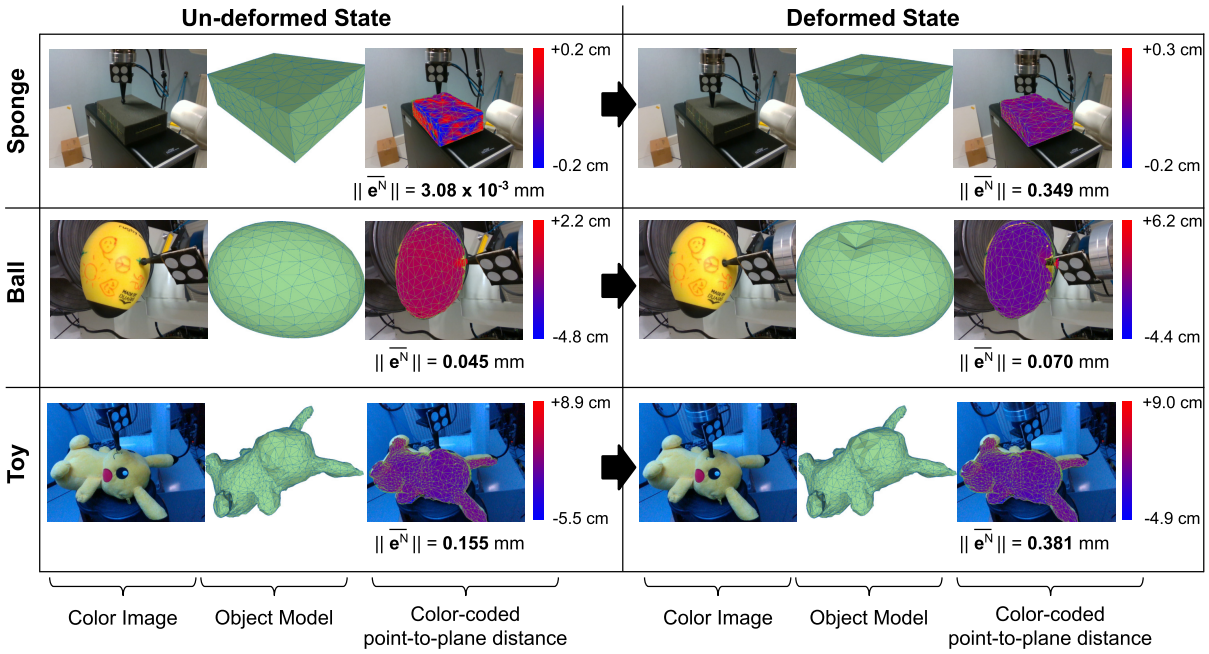


FIGURE 5.4 – Visual results from the deformation tracking methodology. The first and the fourth column shows the color image, the second and the fifth column shows the tracked object model and the third and the sixth column shows the value of $\mathbf{e}^{\mathrm{N}}$ (the geometric error computed using Eqn. 3.5), color-coded and augmented on the object

the robot. For each *condition* of experiment, the average of several indentation tests is considered to be the ground truth. We denote that as: $\overline{\mathrm{E}_{GT}}$.

Let $\hat{\mathrm{E}}(i)$ be the estimation of the Young's modulus obtained from the tracking results for the i^{th} deformation. The Young's modulus that is thereafter used for the external force estimation, denoted by E_{est}, is the one for which the following convergence criterion is respected:

$$\frac{\hat{\mathrm{E}}(i) - \hat{\mathrm{E}}(i-1)}{\hat{\mathrm{E}}(i-1)} < 0.05 \tag{5.11}$$

Some examples of the output of the deformation tracking method are shown in Fig. 5.4. The mean of the norm of the error ($\overline{\|\mathrm{e}^{\mathrm{N}}\|}$) was found to be 1.53 mm, 0.51 mm and 0.21 mm for the foam block, soft ball and plush toy, while the standard deviations were 7.2 mm, 0.7 mm and 1.4 mm respectively, across all the experiments reported in this article. In the three example sequences shown in Fig. 5.4, the mean of the norm of error varied between a maximum of 0.381 mm (for the plush toy, deformed by its nose) to as low as 3 μm (for the undeformed sponge block), despite having outlying correspondence or noise in the range of -5.5 cm to +9.0 cm (which gets rejected by

the M-estimator). In the experiments shown in Table 5.1, the time required for tracking varies between 1.21 to 3.1 sec/frame. However, it was possible to run the same algorithm on the foam block at ~ 800 ms/frame without using any GPU based computation, with negligible loss of accuracy ($< 10\%$).

Our method can be evaluated on different criteria. The first set of experiments was conducted to evaluate the time-performance of the tracking and the quality of the Young's modulus estimation w.r.t. the number of vertices of both the visual mesh and the mechanical mesh. The results of this set of experiments are grouped in Table 5.1. In this table, #Visual designates the number of vertices of the visual mesh while #Mechanical designates the number of vertices of the mechanical mesh. $\overline{t_{tr}}$ is the average deformation tracking time, std(t_{tr}) is its standard deviation, t_{st} is the time that was required to deform the object and reach a steady-state and t_{est} is the time to estimate the Young's modulus. All the times are expressed in seconds. $\overline{\mathrm{E}_{GT}}$ and E_{est} correspond to the average ground truth of Young's modulus and estimated Young's modulus respectively, both expressed in kilo Pascals. Finally, Error designates the percentage of error of the estimation and is given by $Error = 100 * abs(\overline{\mathrm{E}_{GT}} - \mathrm{E}_{est})/\overline{\mathrm{E}_{GT}}$ where abs designates the absolute value.

TABLE 5.2 – Evaluation of the estimation of the Young's modulus (in kPa).

Objects	E_0	$\overline{\mathrm{E}_{\mathrm{GT}}}$	$\mathrm{E}_{\mathrm{est}}$	Error
foam middle	200	454	480	5.7%
	1500	454	451	0.7%
	15000	454	488	7.5%
foam comer	200	247	224	9.3%
ball	1000	156	154	1.3%
toy nose	200	51	53	3.790
toy leg	50	42	43.1	2.6%

The second set of experiments was conducted to evaluate the consistency of the Young's modulus estimation w.r.t. the initial estimate E_0. Low initial estimate are usually of the order of $0.1 \times \overline{\mathrm{E}_{GT}}$ and high initial estimate of the order of $10 \times \overline{\mathrm{E}_{GT}}$. The results of this set of experiments are grouped in Table 5.2.

The last set of experiments was conducted to evaluate the estimation of the defor-

TABLE 5.3 – Comparison of the average norm of the measured and estimated forces.

Objects	Orientation	Indentation Depth (cm)	$\overline{f_{\mathrm{GT}}}(\mathrm{N})$	$f_{\mathrm{est}}(\mathrm{N})$	$\Delta f(\mathrm{N})$
foam	horizontal	2	18.8	20.6	1.8
foam	horizontal	2	21.36	23.11	1.75
foam	horizontal	4	45.46	42.67	−2.79
foam	tilted	4	38.94	35.59	−3.35
ball	horizontal	2.5	9.49	8.57	−0.92
toy	horizontal	2	5.32	5.81	0.49

mation forces in different conditions of experiments. For these experiments, the estimated Young's modulus values summarized in Table 5.2 were used by the deformation tracking method. The results of this set of experiments are grouped in Table 5.3. $\overline{f_{GT}}$ and f_{est} designate the average norm of the ground truth forces measured by the force sensor and the norm of the forces estimated by the deformation tracking method respectively, expressed in Newtons. $\Delta f = f_{est} - \overline{f_{GT}}$ represents the error between the estimated and ground truth forces. The estimated forces are projected onto the measured forces when compared with the ground truth. The force experiments have been conducted with different orientations of the objects (horizontal and tilted by approximately 25 degrees w.r.t. the z-axis of the object frame). Different depths of stimulation have also been tested and are indicated in the depth column of Table 5.3.

Finally, Fig. 5.5 depicts a possible use case of our method to estimate deformation forces exerted on an object in an environment where no force measurements are available.

5.4 Discussion

The first set of experiments whose results are summarized in Table 5.1 shows that the accuracy of the Young's modulus estimation is improved as the resolution of the mechanical mesh increases. This results from the fact that the deformation can be represented with finer details and is thus closer to the reality. Table 5.2 shows that our method is able to estimate the Young's modulus of an arbitrary object even when the initial estimate is much different from the ground truth. A future work would consist in

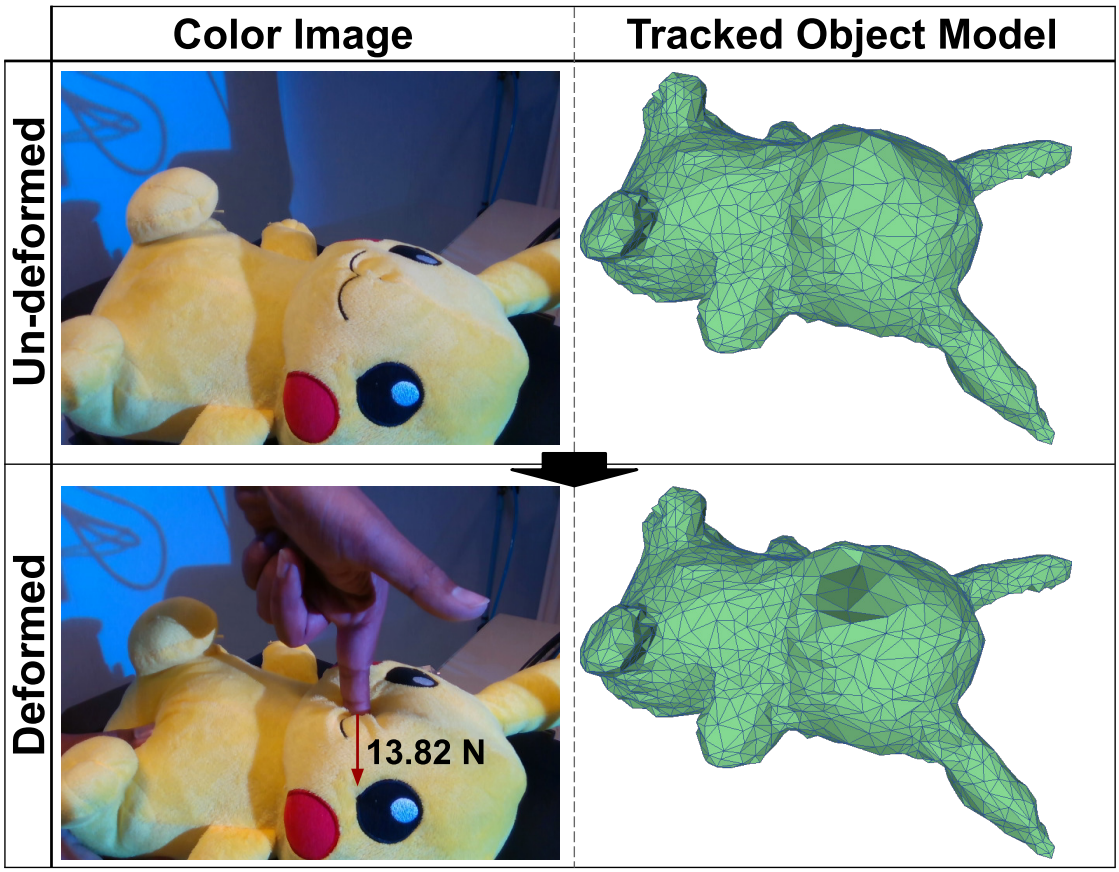


FIGURE 5.5 – Demonstration of force estimation while the plush toy gets freely deformed by hand. The deformation forces (13.8 N) are in the expected range.

using our method to estimate both Poisson ratio and Young's modulus within an iterative process. Table 5.3 shows that the estimated Young's modulus could be used by our method to accurately estimate the deformation forces from visual information only. The use case scenario depicted in Fig.5.5 shows one possible application of our method in environments where no force sensor could be used, for instance in human-performed manipulation tasks. This application opens novel perspectives for better estimating the forces applied on a deformable object by different operators, as well as on a cooperative task on such objects.

5.5 Conclusion

In this chapter, we described a method, that we named STEPE, to simultaneously estimate the elasticity parameters of a deforming object while tracking their surface using a RGB-D camera. The fundamental technique for tracking the surface was already developed in Chapter 3, while this chapter produced a more optimized method of deformation tracking, enabling the estimation of elasticity parameters simultaneously. Moreover, we utilize the estimated parameters to track the deforming forces acting on the object remotely, without using any external force sensor. These results pave the way for better control of deformable objects manipulated by a robot.

CHAPTER 6

CONCLUSION AND PERSPECTIVES

6.1 Conclusion

Here, we first briefly recall the concepts presented throughout this thesis. This is followed by a concluding summary of the individual chapters. We then proceed to end this thesis by discussing some of the short and long term perspective research work that can be spawned by the approaches proposed in this thesis.

6.1.1 Brief Summary of the Thesis

In Chapter 2, we present a framework for precisely tracking objects of complex shapes with joint optimization of depth-based geometric and photometric error terms using a low-accuracy 3D object model with the help of RGB-D cameras. Object tracking with coarse 3D model is useful for industrial applications, where scanning and reconstructing exact 3D model is often prohibitively difficult. In this chapter, a method is proposed that uses a combination of point-to-plane distance minimization and photometric error minimization to track objects accurately. 'Keyframes' are used in this mechanism of object tracking for minimizing drift. The approach proposed in this chapter has been validated on both simulated and real data. It has been shown that our approach is more accurate than existing state-of-the-art approaches on benchmarked, simulated dataset, especially while dealing with low-textured objects with co-planar faces.

In Chapter 3, a method to track deformable objects with a RGB-D camera using a coarse 3D model is presented. The model of deformation is based on corotational FEM. The physical model of the deforming object does not need to be precise and we do not require the precise physical properties of the object to track it accurately. The vertex positions of the surface mesh of the tracked object is regulated using virtual forces which are obtained by minimizing a point-to-plane distance based geometric error between the pointcloud and the mesh. The point of application of force is obtained

by thresholding and clustering of the error obtained from rigid tracking, which is run in parallel with the non-rigid tracking algorithm. This parallel implementation also enables the overall system nearly real-time at frame-rate. The proposed approach has been evaluated on a synthetic dataset with ground-truth, as well as on real data.

In Chapter 4, we propose a methodology for tracking non-rigid objects with RGB-D cameras by minimizing a combination of depth and photometric error. The approach proposed in this chapter does not depend on the accuracy of the material properties provided as input for deformation tracking. However, if an accurate estimate of the material properties are known, it can be used to measure the external, deforming forces acting on the object with just the visual data obtained from the RGB-D camera. The mechanism proposed in this chapter combines computer-vision based methodologies with physical model based deformation representation without computing expensive numerical optimization for minimizing the non-linear error terms. The method proposed here has been validated on synthetic and real data and has been shown to outperform some of the state-of-the-art techniques in physical model based deformation tracking, while opening up new applications, such as visual force estimation on deforming objects.

In Chapter 5, a new approach to simultaneously track the deformation of soft objects and estimate their elasticity parameters has been suggested. This approach also allows the estimation of deforming forces applied on an object from the visual information only. This is done by combining corotational FEM simulations and minimization algorithms to track the deformation and estimate the elasticity parameters of a soft object jointly. A commercial RGB-D sensor is used for obtaining the visual information needed for tracking. This information is utilized to deform the object model according to the virtual forces computed using the deformation tracking algorithm. The module involved in the elasticity parameter estimation minimizes the error between the tracked object and a simulated object deformed by the forces that are measured using a force sensor mounted on a robotic arm. Once the elasticity parameter is estimated, the deformation tracking can then estimate the deforming forces applied on the object without the use of a force sensor. The accuracy of the elasticity parameter estimation and of the force estimation has been validated on several soft objects with various complexities of shape.

6.2 Perspectives

There are some important work which can be done to follow up the methodologies proposed in this dissertation. We categorize them based on the expected duration it would take to complete this perspective research work. The perspectives can also be differentiated from the possible application areas of deformation tracking, which can be numerous.

6.2.1 Short-term Perspectives

The approaches proposed here have been demonstrated to work with depth and photometry based error minimization. However, in Chapter 4, we have already set the groundwork for easily incorporating multiple new cost functions e.g., sparse-feature based, edge-based or wavelet based error terms. This can be a relatively straightforward extension of our approach in Chapter 4 that could lead to further improvement in tracking accuracy.

All of the methods implemented in this thesis have been implemented on a single core of a CPU (OpenMP has been used sparingly for the *STEPE* method presented in Chapter 5). Therefore, all run-time performances reported in this thesis can be subjected to extensive performance optimization, thereby decreasing the runtime. There is a strong potential for improving the runtime significantly, given that some of the computations involved are highly parallelizable.

This thesis makes the use of *control handles* for tracking deformation. In Chapter 3 and Chapter 4, two different methods for determining these *control handles* from the input data has been proposed. However, determining the exact sequence of *control handles* that maximizes the accuracy of the tracking requires some heuristics and can be sensitive to noise and occlusion. Investigating a novel and robust mechanism for selecting these *control handles* to maximize the non-rigid tracking accuracy can be an interesting research area. This can possibly use the help of deep learning based approaches.

6.2.2 Long-term Perspectives

The approaches proposed in Chapter 3 and Chapter 4 can be extended to perform deformation tracking using monocular data. Given that the initial object to camera

transformation at the first frame is known, a frame-to-frame tracking of the deforming surface can be done using a similar method, but with suitably modified cost functions to suit the monocular data.

Iterative mechanical mesh refinement using adaptive remeshing [Narain et al., 2013] could be combined with the approaches proposed in this thesis for tracking the tearing or fracturing of volumetric objects without the prior knowledge of the physical properties of the object being tracked. On the same note, adaptive remeshing can also be used to reconstruct a physically-based model from RGB-D based voxel data. This online reconstructed physical mesh can be used to do the non-rigid object tracking.

A challenging direction for future research could possibly involve developing an unified, physics-based framework that is capable of tracking multiple mutually-interacting rigid objects, non-rigid objects and fluid surfaces using visual information only.

BIBLIOGRAPHIE

[Acton, 1990] Acton, F. S. (1990). *Numerical methods that work*. The Mathematical Association of America (MAA).

[Agudo et al., 2014] Agudo, A., Agapito, L., Calvo, B., and Montiel, J. M. (2014). Good vibrations : A modal analysis approach for sequential non-rigid structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1558–1565.

[Agudo et al., 2011] Agudo, A., Calvo, B., and Montiel, J. (2011). Fem models to code non-rigid ekf monocular slam. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1586–1593.

[Agudo and M-Noguer, 2015] Agudo, A. and M-Noguer, F. (2015). Learning shape, motion and elastic models in force space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 756–764.

[Agudo et al., 2017] Agudo, A., Montiel, J. M. M., Agapito, L., and Calvo, B. (2017). Modal space : A physics-based model for sequential estimation of time-varying shape from monocular video. *Journal of mathematical imaging and vision*, 57(1) :75–98.

[Ahn et al., 2019] Ahn, M. S., Chae, H., Noh, D., Nam, H., and Hong, D. (2019). Analysis and noise modeling of the intel realsense d435 for mobile robots. In *2019 16th International Conference on Ubiquitous Robots (UR)*, pages 707–711.

[Armstrong and Zisserman, 1995] Armstrong, M. and Zisserman, A. (1995). Robust object tracking. In *Asian Conference on Computer Vision*, volume 1, pages 58–61.

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5) :698–700.

[Baker and Matthews, 2001] Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–1090.

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

[Banchoff et al., 2014] Banchoff, T. et al. (2014). Salvador dalí and the fourth dimension. *Proceedings of the Bridges*.

[Bartoli et al., 2015] Bartoli, A., Gerard, Y., Chadebecq, F., Collins, T., and Pizarro, D. (2015). Shape-from-template. *IEEE transactions on pattern analysis and machine intelligence*, 37(10) :2099–2118.

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

[Ben-Chen et al., 2010] Ben-Chen, M., Butscher, A., Solomon, J., and Guibas, L. (2010). On discrete killing vector fields and patterns on surfaces. In *Computer Graphics Forum*, volume 29, pages 1701–1711.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

[Bickel et al., 2009] Bickel, B., Bächer, M., Otaduy, M. A., Matusik, W., Pfister, H., and Gross, M. (2009). Capture and modeling of non-linear heterogeneous soft tissue. In *ACM Transactions on Graphics (TOG)*, volume 28, page 89.

[Blender, 2018] Blender, O. C. (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

[Bregler et al., 2000] Bregler, C., Hertzmann, A., and Biermann, H. (2000). Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 2690.

[Bronte et al., 2017] Bronte, S., Bergasa, L., Pizarro, D., and Barea, R. (2017). Model-based real-time non-rigid tracking. *Sensors*.

[Bronte et al., 2014] Bronte, S., Paladini, M., Bergasa, L. M., Agapito, L., and Arroyo, R. (2014). Real-time sequential model-based non-rigid sfm. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1026–1031.

[Brown, 1971] Brown, D. C. (1971). Close-range camera calibration, photogrammetric engineering. *Engineering and Remote Sensing*, 37(8) :855–866.

[Chan and Zhu, 2005] Chan, T. and Zhu, W. (2005). Level set based shape prior segmentation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1164–1170.

[Chen and Medioni, 1992] Chen, Y. and Medioni, G. G. (1992). Object modeling by registration of multiple range images. *Image Vision Comput.*, 10(3) :145–155.

[Chhatkuli, 2016] Chhatkuli, A. (2016). *Local Analytic and Global Convex Methods for the 3D Reconstruction of Isometric Deformable Surfaces*. PhD thesis.

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

[Choi and Christensen, 2013] Choi, C. and Christensen, H. I. (2013). RGB-d object tracking : A particle filter approach on GPU. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1084–1091.

[Cholesky, 2005] Cholesky, A.-L. (2005). Sur la résolution numérique des systèmes d'équations linéaires. *Bulletin de la Sabix. Société des amis de la Bibliothèque et de l'Histoire de l'École polytechnique*, (39) :81–95.

[Cifuentes et al., 2017] Cifuentes, C. G., Issac, J., Wüthrich, M., Schaal, S., and Bohg, J. (2017). Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters*, 2(2) :577–584.

[Coleman, 1989] Coleman, A. (1989). The greatest mathematical paper of all time. *The Mathematical Intelligencer*, 11(3) :29–38.

[Collins and Bartoli, 2015] Collins, T. and Bartoli, A. (2015). Realtime shape-from-template : System and applications. In *ISMAR*, pages 116–119.

[Courant et al., 1943] Courant, R. et al. (1943). *Variational methods for the solution of problems of equilibrium and vibrations*. Verlag nicht ermittelbar.

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312.

[Davis et al., 2015] Davis, A., Bouman, K. L., Chen, J. G., Rubinstein, M., Durand, F., and Freeman, W. T. (2015). Visual vibrometry : Estimating material properties from

small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5335–5343.

[Dementhon and Davis, 1995] Dementhon, D. F. and Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International journal of computer vision*, 15(1-2) :123–141.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society : Series B (Methodological)*, 39(1) :1–22.

[Dhome et al., 1989] Dhome, M., Richetin, M., Lapreste, J.-T., and Rives, G. (1989). Determination of the attitude of 3d objects from a single perspective view. *IEEE transactions on pattern analysis and machine intelligence*, 11(12) :1265–1278.

[Doumanoglou et al., 2016] Doumanoglou, A., Kouskouridas, R., Malassiotis, S., and Kim, T.-K. (2016). Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3583–3592.

[Eggert et al., 1997] Eggert, D. W., Lorusso, A., and Fisher, R. B. (1997). Estimating 3-d rigid body transformations : a comparison of four major algorithms. *Machine vision and applications*, 9(5-6) :272–290.

[Faugeras, 1993] Faugeras, O. (1993). *Three-dimensional computer vision : a geometric viewpoint*. MIT press.

[Faure et al., 2012a] Faure, F., Duriez, C., Delingette, H., Allard, J., Gilles, B., Marchesseau, S., Talbot, H., Courtecuisse, H., Bousquet, G., Peterlik, I., and Cotin, S. (2012a). SOFA : A Multi-Model Framework for Interactive Physical Simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321.

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

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

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395.

[Frank et al., 2010] Frank, B., Schmedding, R., Stachniss, C., Teschner, M., and Burgard, W. (2010). Learning the elasticity parameters of deformable objects with a manipulation robot. In *Proceedings of the IEEE Conference on Intelligent Robots and Systems*, pages 1877–1883.

[Fugl et al., 2012] Fugl, A. R., Jordt, A., Petersen, H. G., Willatzen, M., and Koch, R. (2012). Simultaneous estimation of material properties and pose for deformable objects from depth and color images. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 165–174.

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

[Gao and Tedrake, 2018] Gao, W. and Tedrake, R. (2018). Surfelwarp : Efficient non-volumetric single view dynamic reconstruction. In *Robotics : Science and System (RSS 2018)*.

[Gao et al., 2003] Gao, X.-S., Hou, X.-R., Tang, J., and Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8) :930–943.

[Garon and Lalonde, 2017] Garon, M. and Lalonde, J.-F. (2017). Deep 6-dof tracking. *IEEE transactions on visualization and computer graphics*, 23(11) :2410–2418.

[Gotardo and Martinez, 2011a] Gotardo, P. F. and Martinez, A. M. (2011a). Kernel non-rigid structure from motion. In *2011 International Conference on Computer Vision*, pages 802–809.

[Gotardo and Martinez, 2011b] Gotardo, P. F. and Martinez, A. M. (2011b). Non-rigid structure from motion with complementary rank-3 spaces. In *CVPR 2011*, pages 3065–3072.

[Grunnet-Jepsen et al.,] Grunnet-Jepsen, A., Sweetser, J. N., Winer, P., Takagi, A., and Woodfill, J. Projectors for intel® realsense™ depth cameras d4xx. Intel® Corporation.

[Hager and Belhumeur, 1998] Hager, G. D. and Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10) :1025–1039.

[Hamer et al., 2009] Hamer, H., Schindler, K., Koller-Meier, E., and Van Gool, L. (2009). Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 1475–1482.

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

[Haouchine et al., 2014] Haouchine, N., Dequidt, J., Berger, M.-O., and Cotin, S. (2014). Single view augmentation of 3d elastic objects. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 229–236.

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

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

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[Helten et al., 2013] Helten, T., Baak, A., Müller, M., and Theobalt, C. (2013). Full-body human motion capture from monocular depth images. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 188–206.

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

[Henry et al., 2013] Henry, P., Fox, D., Bhowmik, A., and Mongia, R. (2013). Patch volumes : Segmentation-based consistent mapping with rgb-d cameras. In *2013 International Conference on 3D Vision-3DV 2013*, pages 398–405.

[Hinterstoisser et al., 2012] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose

estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conf. on computer vision*, pages 548–562.

[Horn, 1987] Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4) :629–642.

[Horn et al., 1988] Horn, B. K., Hilden, H. M., and Negahdaripour, S. (1988). Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7) :1127–1135.

[Hrennikoff, 1941] Hrennikoff, A. (1941). Solution of problems of elasticity by the framework method. *Journal of applied mechanics*, 8(4) :169–175.

[Huang et al., 2015] Huang, J., Ong, S.-K., and Nee, A. Y. (2015). Real-time finite element structural analysis in augmented reality. *Advances in Engineering Software*, 87 :43–56.

[Hughes, 2012] Hughes, T. J. (2012). *The finite element method : linear static and dynamic finite element analysis*. Courier Corporation.

[Innmann et al., 2016] Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., and Stamminger, M. (2016). Volumedeform : Real-time volumetric non-rigid reconstruction. *arXiv preprint arXiv :1603.08161*.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion : real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568.

[Jancosek and Pajdla, 2011] Jancosek, M. and Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Jensen et al., 2018] Jensen, S. H. N., Del Bue, A., Doest, M. E. B., and Aanæs, H. (2018). A benchmark and evaluation of non-rigid structure from motion. *arXiv preprint arXiv :1801.08388*.

[Kehl et al., 2016] Kehl, W., Milletari, F., Tombari, F., Ilic, S., and Navab, N. (2016). Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220.

[Kneip et al., 2011] Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*, pages 2969–2976.

[Kumar et al., 2016] Kumar, S., Dai, Y., and Li, H. (2016). Multi-body non-rigid structure-from-motion. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 148–156.

[Kümmerle et al., 2011] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o : A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613.

[Lepetit et al., 2005] Lepetit, V., Fua, P., et al. (2005). Monocular model-based 3d tracking of rigid objects : A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1) :1–89.

[Letessier and Bérard, 2004] Letessier, J. and Bérard, F. (2004). Visual tracking of bare fingers for interactive surfaces. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122.

[Li and Barbič, 2014] Li, Y. and Barbič, J. (2014). Stable orthotropic materials. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 41–46. Eurographics Association.

[Logan, 2011] Logan, D. L. (2011). *A first course in the finite element method*. Cengage Learning.

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes : A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4) :163–169.

[Lowe, 1991] Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5) :441–450.

[Lu et al., 2000] Lu, C.-P., Hager, G. D., and Mjolsness, E. (2000). Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6) :610–622.

[Luenberger et al., 1984] Luenberger, D. G., Ye, Y., et al. (1984). *Linear and nonlinear programming*, volume 2. Springer.

[Malis and Marchand, 2006] Malis, E. and Marchand, E. (2006). Experiments with robust estimation techniques in real-time robot vision. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 223–228. IEEE.

[Malti et al., 2015] Malti, A., Bartoli, A., and Hartley, R. (2015). A linear least-squares solution to elastic shape-from-template. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1629–1637.

[Malti and Herzet, 2017] Malti, A. and Herzet, C. (2017). Elastic shape-from-template with spatially sparse deforming forces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3337–3345.

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

[Marchand et al., 2016] Marchand, E., Uchiyama, H., and Spindler, F. (2016). Pose estimation for augmented reality : a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12) :2633–2651.

[Marquardt, 1963] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2) :431–441.

[McCormac et al., 2018] McCormac, J., Clark, R., Bloesch, M., Davison, A., and Leutenegger, S. (2018). Fusion++ : Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41.

[McKee et al., 2011] McKee, C. T., Last, J. A., Russell, P., and Murphy, C. J. (2011). Indentation versus tensile measurements of young's modulus for soft biological tissues. *Tissue Engineering Part B : Reviews*, 17(3) :155–164.

[Meer et al., 1991] Meer, P., Mintz, D., Rosenfeld, A., and Kim, D. Y. (1991). Robust regression methods for computer vision : A review. *International Journal of Computer Vision (IJCV)*, 6(1) :59–70.

[Michel et al., 2017] Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., and Rother, C. (2017). Global hypothesis generation for 6d object pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Moerman et al., 2009] Moerman, K. M., Holt, C. A., Evans, S. L., and Simms, C. K. (2009). Digital image correlation and finite element modelling as a method to determine mechanical properties of human soft tissue in vivo. *Journal of biomechanics*, 42(8) :1150–1153.

[Moré, 1978] Moré, J. J. (1978). The levenberg-marquardt algorithm : implementation and theory. In *Numerical analysis*, pages 105–116. Springer.

[Moulon et al., 2012] Moulon, P., Monasse, P., and Marlet, R. (2012). Adaptive structure from motion with a contrario model estimation. In *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*, pages 257–270. Springer Berlin Heidelberg.

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

[Myronenko and Song, 2010] Myronenko, A. and Song, X. (2010). Point set registration : Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12) :2262–2275.

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

[Narain et al., 2013] Narain, R., Pfaff, T., and O'Brien, J. F. (2013). Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG)*, 32(4) :1–8.

[Newcombe et al., 2015] Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion : Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352.

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion : Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136.

[Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6) :169.

[Paloc et al., 2002] Paloc, C., Bello, F., Kitney, R. I., and Darzi, A. (2002). Online multiresolution volumetric mass spring model for real time soft tissue deformation. In

International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 219–226.

[Parashar et al., 2015] Parashar, S., Pizarro, D., Bartoli, A., and Collins, T. (2015). As-rigid-as-possible volumetric shape-from-template. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 891–899.

[Paulus et al., 2015] Paulus, C. J., Haouchine, N., Cazier, D., and Cotin, S. (2015). Augmented reality during cutting and tearing of deformable objects. In *2015 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 54–59.

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

[Petit et al., 2017a] Petit, A., Ficuciello, F., Fontanelli, G. A., Villani, L., and Siciliano, B. (2017a). Using physical modeling and rgb-d registration for contact force sensing on deformable objects. In *2017 International Conference on Informatics in Control, Automation and Robotics*.

[Petit et al., 2017b] Petit, A., Lippiello, V., Fontanelli, G. A., and Siciliano, B. (2017b). Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot. *Robotics and Autonomous Systems*, 88 :187–201.

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

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

[Pilet et al., 2008] Pilet, J., Lepetit, V., and Fua, P. (2008). Fast non-rigid surface detection, registration and realistic augmentation. *International Journal of Computer Vision*, 76(2) :109–122.

[Pilkey, 1998] Pilkey, D. F. (1998). *Computation of a damping matrix for finite element model updating*. PhD thesis, Virginia Tech.

[Pomerleau et al., 2015] Pomerleau, F., Colas, F., Siegwart, R., et al. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1) :1–104.

[Pottmann et al., 2004] Pottmann, H., Leopoldseder, S., and Hofer, M. (2004). Registration without icp. *Computer Vision and Image Understanding*, 95(1) :54–71.

[Prisacariu and Reid, 2012] Prisacariu, V. A. and Reid, I. D. (2012). Pwp3d : Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, 98(3) :335–354.

[Ramalingam and Taguchi, 2013] Ramalingam, S. and Taguchi, Y. (2013). A theory of minimal 3d point to 3d plane registration and its generalization. *International journal of computer vision*, 102(1-3) :73–90.

[Rankin, 1988] Rankin, C. (1988). Consistent linearization of the element-independent corotational formulation for the structural analysis of general shells. *NASA Contractor Report 278428, Lockheed Palo Alto Res. Lab., CA, 1988*.

[Rao, 1995] Rao, S. S. (1995). *Mechanical Vibrations Laboratory Manual*. Year, Edition Addison-Wesley Publishing Company.

[Ren et al., 2017] Ren, C., Prisacariu, V., Kähler, O., Reid, I., and Murray, D. (2017). Real-time tracking of single and multiple objects from depth-colour imagery using 3d signed distance functions. *International Journal of Computer Vision*, pages 1–16.

[Roth and Vona, 2012] Roth, H. and Vona, M. (2012). Moving volume kinectfusion. In *BMVC*, pages 1–11.

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

[Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut : Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM.

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

[Rünz and Agapito, 2017] Rünz, M. and Agapito, L. (2017). Co-fusion : Real-time segmentation, tracking and fusion of multiple objects. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4471–4478.

[Runz et al., 2018] Runz, M., Buffier, M., and Agapito, L. (2018). Maskfusion : Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018*

IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 10–20.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3D is here : Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

[Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). Slam++ : Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359.

[Salzmann et al., 2008] Salzmann, M., Moreno-Noguer, F., Lepetit, V., and Fua, P. (2008). Closed-form solution to non-rigid 3d surface registration. In *European conference on computer vision*, pages 581–594.

[Salzmann et al., 2007] Salzmann, M., Pilet, J., Ilic, S., and Fua, P. (2007). Surface deformation models for nonrigid 3d shape recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8) :1481–1487.

[Salzo and Villa, 2012] Salzo, S. and Villa, S. (2012). Convergence analysis of a proximal gauss-newton method. *Computational Optimization and Applications*, 53(2) :557–589.

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

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

[Sedef et al., 2006] Sedef, M., Samur, E., and Basdogan, C. (2006). Real-time finite-element simulation of linear viscoelastic tissue behavior based on experimental data. *IEEE Computer Graphics and Applications*, 26(6).

[Segal et al., 2009] Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics : science and systems*, volume 2, page 435.

[Sengupta et al., 2019] Sengupta, A., Krupa, A., and Marchand, E. (2019). Tracking of non-rigid objects using rgb-d camera. In *2019 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Bari, Italy*.

[Shum and Szeliski, 2001] Shum, H.-Y. and Szeliski, R. (2001). Construction of panoramic image mosaics with global and local alignment. In *Panoramic vision*, pages 227–268. Springer.

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

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

[Slavcheva et al., 2017b] Slavcheva, M., Baust, M., Cremers, D., and Ilic, S. (2017b). KillingFusion : Non-rigid 3D Reconstruction without Correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Slavcheva et al., 2018] Slavcheva, M., Baust, M., and Ilic, S. (2018). Sobolevfusion : 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2646–2655.

[Slavcheva et al., 2016] Slavcheva, M., Kehl, W., Navab, N., and Ilic, S. (2016). Sdf-2-sdf : Highly accurate 3d object reconstruction. In *European Conference on Computer Vision*, pages 680–696.

[Solomon, 2015] Solomon, J. (2015). *Numerical algorithms : methods for computer vision, machine learning, and graphics*. AK Peters/CRC Press.

[Stein, 1997] Stein, G. P. (1997). Lens distortion calibration using point correspondences. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 602–608.

[Stults, 1984] Stults, B. M. (1984). Preventive health care for the elderly. *Western Journal of Medicine*, 141(6) :832.

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

[Sun et al., 2018] Sun, L., Aragon-Camarasa, G., Rogers, S., and Siebert, J. P. (2018). Autonomous clothes manipulation using a hierarchical vision architecture. *IEEE Access*, 6 :76646–76662.

[Teichman et al., 2013] Teichman, A., Lussier, J. T., and Thrun, S. (2013). Learning to segment and track in rgbd. *IEEE Transactions on Automation Science and Engineering*, 10(4) :841–852.

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

[Varol et al., 2012] Varol, A., Salzmann, M., Fua, P., and Urtasun, R. (2012). A constrained latent variable model. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2248–2255.

[Walker et al., 1991] Walker, M. W., Shao, L., and Volz, R. A. (1991). Estimating 3-d location parameters using dual number quaternions. *CVGIP : image understanding*, 54(3) :358–367.

[Wang et al., 2015] Wang, B., Wu, L., Yin, K., Ascher, U., Liu, L., and Huang, H. (2015). Deformation Capture and Modeling of Soft Objects. *ACM Trans. Graph.*, 34(4) :94 :1–94 :12.

[Wang et al., 2015] Wang, B., Wu, L., Yin, K., Ascher, U. M., Liu, L., and Huang, H. (2015). Deformation capture and modeling of soft objects. *ACM Trans. Graph.*, 34(4) :94–1.

[Wang et al., 2017] Wang, K., Zhang, G., and Xia, S. (2017). Templateless non-rigid reconstruction and motion tracking with a single rgb-d camera. *IEEE Transactions on Image Processing*, 26(12) :5966–5979.

[Wei et al., 2009] Wei, Y., Cotin, S., Fang, L., Allard, J., Pan, C., and Ma, S. (2009). Toward real-time simulation of blood-coil interaction during aneurysm embolization. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 198–205.

[Whelan et al., 2013] Whelan, T., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Deformation-based loop closure for large scale dense rgb-d slam. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555.

[Wikipedia, 2016] Wikipedia (2016). Iterative closest point. [Online ; accessed 04-January-2016].

[Willimon et al., 2012] Willimon, B., Hickson, S., Walker, I., and Birchfield, S. (2012). An energy minimization approach to 3d non-rigid deformable surface estimation using rgbd data. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2711–2717.

[Wood and Zienkiewicz, 1977] Wood, R. D. and Zienkiewicz, O. (1977). Geometrically nonlinear finite element analysis of beams, frames, arches and axisymmetric shells. *Computers & Structures*, 7(6) :725–735.

[Wu et al., 2018] Wu, J., Zhou, B., Russell, R., Kee, V., Wagner, S., Hebert, M., Torralba, A., and Johnson, D. M. (2018). Real-time object pose estimation with pose interpreter networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6798–6805.

[Wu and Lee, 1997] Wu, J.-Y. and Lee, R. (1997). The advantages of triangular and tetrahedral edge elements for electromagnetic modeling with the finite-element method. *IEEE Transactions on Antennas and Propagation*, 45(9) :1431–1437.

[Xiang et al., 2018] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn : A convolutional neural network for 6d object pose estimation in cluttered scenes.

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking : A survey. *Acm computing surveys (CSUR)*, 38(4) :13.

[Zabatani et al., 2019] Zabatani, A., Surazhsky, V., Sperling, E., Moshe, S. B., Menashe, O., Silver, D. H., Karni, T., Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2019). Intel® realsense™ sr300 coded light depth camera. *IEEE transactions on pattern analysis and machine intelligence*.

[Zeng et al., 2012] Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2012). A memory-efficient kinectfusion using octree. In *Computational Visual Media*, pages 234–241. Springer.

[Zhang et al., 2019] Zhang, Z., Petit, A., Dequidt, J., and Duriez, C. (2019). Calibration and external force sensing for soft robots using an rgb-d camera. *IEEE Robotics and Automation Letters*, 4(3) :2356–2363.

[Zhao et al., 2016] Zhao, J.-M., Song, X.-X., and Liu, B. (2016). Standardized compliance matrices for general anisotropic materials and a simple measure of anisotropy degree based on shear-extension coupling coefficient. *International Journal of Applied Mechanics*, 8(06) :1650076.

[Zollhöfer et al., 2014] Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., et al. (2014). Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4) :156.

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre :Suivi visuel d'objets déformables avec une caméra RGB-D

Mot clés : Suivi visuel, suivi des déformations, FEM

Resumé : Le suivi d'objets déformable à partir d'informations visuelles à de nombreuses applications dans le domaine de la robotique, de l'animation ou de la simulation. Dans cette thèse, nous proposons de nouvelles approches pour le suivi d'objets rigides et non rigides à l'aide d'une caméra RGB-D. Cette thèse comporte quatre contributions principales. La première contribution est une nouvelle approche de suivi d'objets dans des images RGB-D qui utilise des erreurs basées sur la profondeur et la photométrie pour suivre et localiser des formes complexes en utilisant leur modèle 3D grossier. La seconde contribution porte sur une méthode de suivi d'objets non rigides reposant sur une approche par éléments finis (FEM) pour suivre et caractériser les déformations. La troisième contribution est une approche de suivi de la déformation qui minimise une combinaison d'erreurs géométriques et photométriques tout en utilisant la FEM comme modèle de déformation. Finalement, la quatrième contribution consiste à estimer les propriétés d'élasticité d'un objet analysant ses déformations toujours à l'aide d'une caméra RGB-D. Une fois les paramètres d'élasticité estimés, la même méthodologie peut être réutilisée pour caractériser les forces de contact.

Title : Visual Tracking of Deformable Objects with RGB-D Camera

Keywords : Visual Tracking, Deformation Tracking, FEM

Abstract : Tracking soft objects using visual information has immense applications in the field of robotics, computer graphics and automation. In this thesis, we propose multiple new approaches for tracking both rigid and non-rigid objects using a RGB-D camera. There are four main contributions of this thesis. The first contribution is a rigid object tracking method which utilizes depth and photometry based errors for tracking complex shapes using their coarse, 3D template. The second contribution is a non-rigid object tracking method which uses co-rotational FEM to track deforming objects by regulating the virtual forces acting on the surface of a physics based model of the object. The third contribution is a deformation tracking approach which minimizes a combination of geometric and photometric error while utilizing co-rotation FEM as the deformation model. The fourth contribution involves estimating the elasticity properties of a deforming object while tracking their deformation using RGB-D camera. Once the elasticity parameters have been estimated, the same methodology can be re-utilized for tracking contact forces on the surface of deforming objects.