TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2021

128

Alberto Badías Herbera

Simulated Reality: Physics-Based Mixed and Augmented Reality for Intelligence Augmentation

Director/es

González Ibáñez, David
Alfaro Ruiz, Icíar

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Tesis Doctoral

SIMULATED REALITY: PHYSICS-BASED MIXED AND AUGMENTED REALITY FOR INTELLIGENCE AUGMENTATION

Autor

Alberto Badías Herbera

Director/es

González Ibáñez, David
Alfaro Ruiz, Icíar

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería Mecánica

2020

Simulated Reality: Physics-Based Mixed and Augmented Reality for Intelligence Augmentation

PhD thesis by

Alberto Badías Herbera

Doctoral advisors

Dr. David González Ibáñez

Dr. Icíar Alfaro Ruiz

PhD Programme in **Mechanical Engineering**

December 2019

Universidad
Zaragoza
1542

Simulated Reality: Physics-Based Mixed and Augmented Reality for Intelligence Augmentation

PhD thesis by

Alberto Badías Herbera

Doctoral advisors

Dr. David González Ibáñez
Dr. Icíar Alfaro Ruiz

PhD Programme in **Mechanical Engineering**
December 2019

Universidad
Zaragoza
1542

A mis padres y mi hermana.

A Maribel.

Agradecimientos

Quiero dar las gracias, en primer lugar, al equipo que me ha ayudado en este arduo camino que es la tesis doctoral: Icíar Alfaro, Elías Cueto y David González. Los tres empezaron como directores de mi tesis, pronto se convirtieron en compañeros y finalmente en grandes amigos, que además de dirigir mi trabajo con gran sabiduría, han contribuído en formarme como persona y en animarme cuando más lo he necesitado, haciéndome ver la importancia real de las cosas. Quiero incluir también aquí a Paco Chinesta, por acogerme durante mi estancia en Nantes y convertirse también en un cuarto director de mi trabajo desde la lejanía, cuyo conocimiento y humor son tan extensos, que no sabría definir cuál es mayor.

Quiero agradecer a todos los compañeros que me han aguantado a lo largo de mi tesis, con especial atención a cuatro personas que se han convertido en mi familia académica. En primer lugar, Javier Ortún, mi hermano académico durante la tesis, con quien he compartido mucho aprendizaje y experiencias, y las que nos quedan por vivir. A Beatriz Palacios, con quien cerrábamos ese triángulo de amistad y recordamos con cariño, aunque haya optado por un futuro mejor. A Carlos Quesada, quien me descubrió todas las bondades de la PGD y pronto se convirtió en mi confesor, siendo otro de mis hermanos académicos. Y finalmente a Silvia Hervás, cuya complicidad es tal que con solo verme ya sabe lo que estoy pensando, algo que creo que es mutuo. No me quiero olvidar del grupo del café, donde cada día aprendo lecciones sobre ciencia, arte o reflexiones de la vida en general desde una perspectiva desenfadada, por parte de gente que llegará, sin duda alguna, muy lejos en la vida.

Quiero dar las gracias también a la familia de Nantes, destacando a Adrien Scheuer, Ángel León, Carlos Sandino, Clara Argerich, Elena López, Giacomo Quaranta, Laura Oter, Rubén Ibáñez y Santiago Montagud, los que me acogieron en su entorno más íntimo y me permitieron disfrutar de una estancia en Francia sintiéndome como en casa, compartiendo grandes momentos y cerveza belga.

Debo agradecer también toda la ayuda técnica recibida por parte de Sarah Curtit con las simulaciones aerodinámicas, de José María Martínez Montiel por su inestimable consejo con los problemas de visión por computador y a Ashis Banerjee por permitirme descubrir el mundo del *deep learning* durante mi estancia en Seattle.

Quiero dedicar esta tesis a mi familia, en particular a mis padres y mi hermana, que siempre han velado por mi futuro de manera incesante. Todo lo que pueda ser en esta

vida se deberá a ellos. Y a Maribel, con la que espero compartir un largo futuro. Quiero agradecer el apoyo de todos mis amigos. Agradezco también a los futuros lectores por tomarse el interés de leer este documento. Y si me dejo a alguien, también se lo agradezco.

No quiero olvidarme de la Universidad de Zaragoza y el Ministerio de Economía y Competitividad, que me han dado un lugar de trabajo y han financiado mi tesis, así como a la empresa ESI Group por apostar por nuestras ideas mediante un proyecto de colaboración.

A todos, gracias de corazón.

Abstract

Simulated reality is a new paradigm that allows computers to understand and reproduce the physical phenomena that occur in their environment. By physical phenomena we mean the static and dynamic description of the changes that happen around us: rigid body movements, solid objects that deform, liquids that flow or change their state, gases that disperse, electromagnetic waves that propagate or even the joint of all the previous effects. Any phenomenon that can be simulated by means of a computer has a place within simulated reality, which consists of reproducing these interactions in a virtual way (inside a computer) to be subsequently shown in real time to a user. Our work consists in creating a human-centered tool that allows the user to observe the behavior of all these physical phenomena from a scientific point of view, that is, perceiving information about stresses, strains, velocities, flow rates or intensities with their specific values at the same time they are happening. It is about providing information that the user cannot perceive directly with his senses (intelligence augmentation), putting to his service a series of tools that allow him to make decisions with a greater capacity of knowledge, or simply to observe the dynamic phenomena promoting higher comprehension.

In general, the proposed tool requires a great capacity of computation to describe all these phenomena. The simulation of physical problems usually requires large computational resources that are far from solving the equations in real time. Complex models with non-linearities and coupled phenomena usually appear in this type of problems. This is why we use model order reduction techniques, to reduce the complexity of the models and evaluate them at the same speed as real phenomena using data assimilation procedures, to later show the results to the user. The most common methods of dimensionality reduction require to preprocess the solutions, carried out off-line, but allowing an on-line evaluation that meets time requirements. This type of off-line processing usually include multiparametric solutions that cover a wide range of solutions in order to create physically consistent estimations in the on-line evaluation. In addition, dimensionality reduction methods project the data to a new space that is, usually, more efficient, which translates into the compression of the data to reduce the storage space, at the same time very fast evaluations for specific parametric values are assured.

In order to favour the assimilation of all this amount of information by the user, we use mixed reality tools and visual devices. We show the information by creating augmented video sequences with physical information coming from the simulations, allowing an interactive and simple visualization of the results. Since the communication is done by visual

path, we are forced to work at a frequency around 30-60 Hz, since these are the standard refresh frequencies on the common devices.

In this thesis a complete framework has been developed covering all the necessary pieces to create a simulated reality system, including new methodologies in model order reduction methods using non-linear projections, the application of existing order reduction methods on new physical problems, the use of artificial intelligence, the improvement in the interaction and visualization of data and new implementations to improve the visual acquisition process using standard cameras in dynamic environments.

Resumen

La realidad simulada es un nuevo paradigma que pretende que los ordenadores comprendan y reproduzcan los fenómenos físicos que sufren los objetos del entorno. Con fenómenos físicos nos referimos a la descripción estática y dinámica de los cambios que se producen a nuestro alrededor: objetos sólidos que se mueven, que se deforman, líquidos que fluyen o cambian de estado, gases que se dispersan, ondas electromagnéticas que se propagan o incluso la interacción conjunta de todos los anteriores. Todo fenómeno que pueda simularse mediante computador tiene cabida dentro de la realidad simulada, que consiste en reproducir esas interacciones de manera virtual en el interior de un ordenador para ser posteriormente mostradas en tiempo real sobre el entorno. Este trabajo consiste en crear una herramienta centrada en la persona que permita a cualquier usuario observar el comportamiento de todos esos fenoménos físicos desde un punto de vista científico, es decir, aportando información acerca de deformaciones, tensiones, velocidades, caudales o intensidades mostrando sus valores concretos al mismo tiempo que están sucediendo. Se trata de brindar una información que el usuario no puede percibir directamente con sus sentidos (inteligencia aumentada), poniendo a su servicio una serie de herramientas que le permitan tomar decisiones con una mayor capacidad de conocimiento o simplemente observar los fenómenos dinámicos favoreciendo su comprensión.

La herramienta propuesta requiere, en general, gran capacidad de cómputo para poder describir todos estos fenómenos. La simulación de los problemas físicos planteados suele demandar grandes recursos computacionales que están muy lejos de resolver dichos cálculos en tiempo real. Modelos complejos con no linealidades y fenómenos acoplados suelen aparecer en este tipo de problemas. Es por ello que empleamos técnicas de reducción de la dimensionalidad para poder reducir la complejidad de estos modelos y poder evaluarlos a la misma velocidad que suceden los fenómenos reales, lo que se conoce como asimilación de datos, para inmediatamente mostrar los resultados al usuario. Los métodos más comunes de reducción de la dimensionalidad requieren de un preprocesado de los modelos, llevado a cabo de manera *off-line*, pero que permite una evaluación *on-line* que cumple con los requisitos temporales fijados. Este tipo de cálculos *off-line* suelen incluir soluciones multiparamétricas que abarcan un rango de valores amplio para poder emitir estimaciones físicamente consistentes en su posterior evaluación *on-line*. Además, los métodos de reducción de la dimensionalidad suelen buscar un nuevo espacio para representar dichos datos de manera optimizada, lo que se traduce en la compresión de los mismos para que ocupen menos espacio de almacenamiento, al mismo tiempo que

permiten evaluaciones muy rápidas para unos valores paramétricos concretos.

Para favorecer la absorción de toda esta cantidad de información por parte del usuario empleamos herramientas de realidad mixta y dispositivos visuales. Mostramos la información creando secuencias de video aumentadas con la física que proviene de las simulaciones, permitiendo una visualización de los resultados interactiva y sencilla. Puesto que la comunicación se realiza mediante vía visual, tenemos una frecuencia de actualización de los datos impuesta entorno a 30-60 Hz, puesto que estas son las frecuencias de refresco estándar en las pantallas de los dispositivos.

En esta tesis se ha desarrollado un entorno completo que engloba todas las partes necesarias para crear un sistema de realidad simulada, incluyendo desde nuevas metodologías en técnicas de reducción empleando proyecciones no lineales, la aplicación de métodos de reducción ya existentes sobre nuevos problemas físicos, la mejora en la interacción y visualización de datos o nuevas implementaciones para mejorar la captación de información mediante cámaras en ambientes dinámicos.

Index

Contents

List of Figures

List of Algorithms

Thesis

Chapter 1

Introduction

Engineering simulation has undergone a great revolution in recent years. Thanks to digital technologies, current society has almost unlimited and instant access to a large amount of information stored on the cloud. No matter the question, it is common to get an answer through our favorite search engine. This ease and speed of access to information has also been transferred to the industry, which is increasingly demanding shorter response times. Standard solvers of differential equations applied to computational mechanics have been strongly developed during the last decades, reaching a high level of maturity where few improvements can be made. This level of saturation contrasts with the demands of industry, where it seems that data science can provide that little push on the computational side to satisfy these demands. However, data techniques must be strongly backed by physics, fulfilling equilibrium equations, so not any approach is valid.

According to the artificial intelligence director of Tesla, Andrej Karpathy[1], there is a clear difference between *Software 1.0*, understood as *"classical programming written in languages such as Phyton, C++, etc."* and *Software 2.0*, that *"can be written in much more abstract, human unfriendly language, such as the weights of a neural network"*. This second software version is based on the direct use of data, where the relevant information is extracted through machine learning techniques. According to Karpathy, *"a large portion of real-world problems have the property that it is significantly easier to collect the data (or more generally, identify a desirable behavior) than to explicitly write the program"*, where *"they represent the beginning of a fundamental shift in how we write software"*.

This way of approaching problems is very useful, as Karpathy says, for certain applications where data acquisition is simple but modeling is not. In other words, problems that are hard to explain by means of simple models in closed form, like image classification [Deng et al., 2009], image inpainting or denoising [Xie et al., 2012], natural language processing [Collobert and Weston, 2008], music composition [Carr and Zukowski, 2018] or even teaching a machine how to properly play in a 3D role-playing game [OpenAI, 2019]. However, for the moment, it seems that the computation of physical problems in engineering still needs to rely on models, although some works appeared where deep learn-

[1]`https://medium.com/@karpathy/software-2-0-a64152b37c35`

ing techniques are used as solvers for differential equations [Raissi et al., 2017]. It seems that data science can have a place in engineering simulation, but for the moment it is understood as a mixture between classical solvers and modern learning tools.

The term *intelligence augmentation*[2] or *intelligence amplification* (IA, not to be confused with AI) has recently become popular, understood as the process by which a machine provides knowledge to a user in order to increase his human capabilities. With respect to industry, this concept is applied to improve decision-making in real time, where humans can perceive information that is not directly available through their senses. This information can be treated in multiple ways, but a natural perception is desired, so visual interaction seems to be the fastest way since images suppose an enormous flow of information for the human being (30 percent of the cortex is employed in visual processing, as compared with 8 percent for touch and just 3 percent for hearing [Grady, 1993]). Following the same line, the concept of *Digital Twin* arises, which consists in a virtual replica of real objects or processes with the aim of modelling their interaction with the environment. The introduction of all these new concepts demonstrates the speed at which the interaction between user and technology is evolving, and it is precisely in this aspect where this thesis is focused.

1.1 General Overview

Many of the terms presented above may not sound familiar, so probably some clarifications are needed to better understand the scope of the thesis. Fig. 1.1 shows some of these concepts in a hierarchical fashion.

AI can be defined as *"the study of agents that receive percepts from the environment and perform actions."* [Russell and Norvig, 2016]. In other words, AI is understood as an *agent* that shows abilities of deduction (*symbolic processing*) and learning (*subsymbolic processing*) [Marsland, 2014]. Symbolic processing refers to behaviors such as computer reasoning and deduction of facts, which was part of the early AI (trying to emulate human intelligence). That principle of artificial intelligence was more related to philosophy, fiction, and imagination [Buchanan, 2005]. On the other hand, the term subsymbolic processing, also known as *machine learning* (ML) can be summarized as the set of tasks of *remembering*, *adapting* and *generalising* [Marsland, 2014]. For this, it is necessary to make use of the experience, extracting the relevant information from training data. It finally translates into learning a behaviour that produces an output to certain stimulus.

In turn, ML can also be divided into smaller groups, which perhaps some of the most representative are *classification*, *regression*, *clustering* and *dimensionality reduction*. The purpose of classification is to tag observed data to group them into a set where certain properties are shared. Regression aims to discover the relationships that exist inside the data and how dependent variables change. The clustering process aims to group together data that share similarities, but unlike classification, discriminant functions are not known

[2] http://iadesignkit.com/

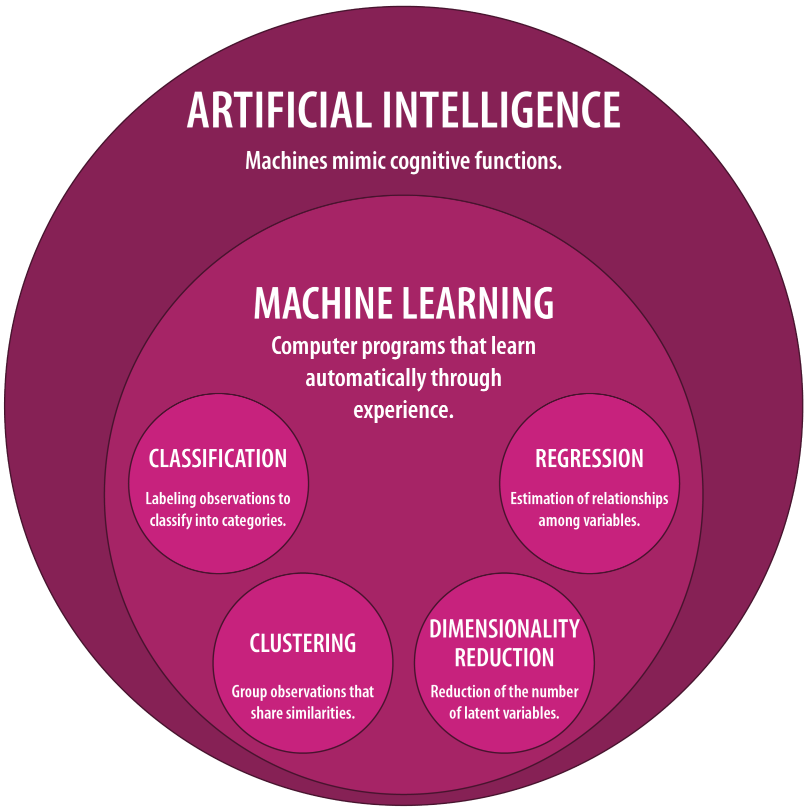

ARTIFICIAL INTELLIGENCE
Machines mimic cognitive functions.
MACHINE LEARNING
Computer programs that learn automatically through experience.
CLASSIFICATION
Labeling observations to classify into categories.
REGRESSION
Estimation of relationships among variables.
CLUSTERING
Group observations that share similarities.
DIMENSIONALITY REDUCTION
Reduction of the number of latent variables.

Figure 1.1: Hierarchical explanation of Artificial Intelligence (AI).

a priori. Finally, dimensionality reduction tries to find the latent variables that truly define the data. It is this part where this thesis is focused in: the development of dimensionality reduction techniques to reduce the complexity in physical simulations.

Regarding the models or techniques that can be used in ML, we can talk about *classical methods* and *modern methods*. The latter include artificial neural networks and all the recent development of *deep learning*. In the classical methods, and with focus on order reduction, we include techniques such as *Principal Component Analysis* (PCA) [Wold et al., 1987], *Linear Discriminant Analysis* (LDA) [Welling, 2005] and *Proper Generalized Decomposition* (PGD) [Ammar et al., 2006]. These methods are discussed in greater detail in next sections.

Another classification with respect to the type of learning can be made in ML: supervised, unsupervised and semi-supervised. A brief description of each type is given below.

- Supervised learning, in general, requires some previous work on the data, which must be tagged before training. It is usually the user who must label the data, being one of the most important and tedious steps in machine learning. Well labeled and structured data is the basis for success in any supervised learning method. The data is usually structured in two parts: the *training set* and the *test set*. The training dataset is used to train the machine learner, while the test part is used to evaluate the prediction of the learned model. *Classification* and *Regression* are two examples of supervised learning.

- It is not always easy to get a clean and well labelled dataset, so in such cases, unsupervised learning can be applied, without that prior work being necessary. Unsupervised techniques are based in algorithms that are capable of identifying notable variations in the data autonomously and without explicit instructions. *Clustering* and *Dimensionality Reduction* are examples of unsupervised learning, where tools such as *k-means clustering* [Hartigan and Wong, 1979], *PCA* or *autoencoders* [Baldi, 2012] are used.
- There is also an intermediate option known as *semi-supervised* learning, where a set of tagged data is needed, but learning is improved with unlabeled data. *Generative Adversarial Nets* [Goodfellow et al., 2014] are an example, where two opposing neural networks are used to increase their learning.

1.1.1 Data vs Models

The confrontation between data and models is still an open question, and there are large groups of people positioned on one side or the other. However, data is not incompatible with the models, they are precisely complementary. If we go back to the ancient thinkers, we find models like the very popular *Newton's law of universal gravitation* (17th century) [Newton, 1833], or more complex models such as the *Navier-Stokes equations* (19th century), where currently only an analytical solution is known for very particular cases [Fefferman, 2006]. Some of those models or laws are contrasted and tested through proofs, while others like the laws of material behavior are based on experimentation, since we are not able to prove them otherwise, and we adjust experimental models with test results. That is why there are some problems that can be solved in a very simple way through models, while others seem to suggest a direct analysis from the data.

Classical models were based on observation, so initially they needed small datasets to postulate laws. But this approach largely contrasts with the idea of forgetting the models and working only with the available data, creating the *Data-Driven* approach. Those who defend this approach argue that machine learning techniques must define those models, which may not be understandable by humans, but can identify variations of information that escape the human mind. However, human-created models generally require less computational capacity making it possible to compute simpler simulations. They are simplifications that allow us to describe and understand the phenomena that surround us, where it is not necessary to deal with problems such as data filtering. Due to the above, we consider that a distinction should be made between model-based problems and data-driven problems, where both have a place in current engineering. For example, even if a human being is able to identify a cat in an image, he is not able to define a general law to identify that cat in any image. We do not know how many, or which parameters are needed to distinguish a cat from other objects in an image. But we do know how to train a classifier with a database as large as *Imagenet* [Deng et al., 2009] that is capable of identifying and locating a cat in an image.

On the other hand, the application of Model Order Reduction (MOR) methods can be interpreted as a data-based analysis technique. Although data may come from model-based computational simulations, we are trying to identify latent variations in information. In addition, it is possible that this variation can be represented in a vector space of variables without physical sense, but more efficient. And that is precisely where the advantage of MOR methods lies. Knowing these principal variations of information, and if we are also able to move along that reduced space, we can estimate new observations without recomputing the models.

To conclude, we have to highlight that the use of data is not as novel as it seems; statistical science has been developing for centuries. However, new developments in parallel processing hardware, together with increased storage capacity and large data production, have made us capable of handling large amounts of information (*Big Data*). This massiveness of data is what opens the new era of data-driven techniques.

1.1.2 Spatial Artificial Intelligence

Nowadays, artificial intelligence is used in many fields. Medicine, education, transport, design, manufacturing, building, retail, advertising or entertainment are some of the examples where it is applied. Anything that can produce data is approachable by these new technologies, although in the vast majority of cases we are really talking about machine learning (it is usually erroneously generalized as AI). There seems to be no limit to the exploitation of data, and many applications are being developed that only a few decades ago sounded like science fiction.

An AI agent should be able to interact with the world and make decisions autonomously, which requires processing and understanding of the surrounding environment. The system must learn continuously, having the ability to extract relevant information. It should interact with an environment that is dynamically changing, while making decisions in real time. For this, Andrew Davison suggests the use of the term *Spatial AI* [Davison, 2018], where he defines a 3-layer hierarchical system for the spatial understanding of the environment [Davison, 2019]: *Robust Localization*, *Dense Mapping* and *Semantic Understanding*. We trust it would be necessary to add a last layer able to understand the physical phenomena that occur around the system (see Fig. 1.2). Probably, according to Davison's vision, this physical understanding is gathered within the *Semantic Understanding*, but we consider them so important fields that they should be separated.

The first level of the Spatial AI approach is related to the ability of the agent to locate itself in space and navigate through it in a robust manner. The great development in the last decades in computer vision techniques has brought new algorithms of *Simultaneous Localization and Mapping* (SLAM) performing localization tasks very efficiently and

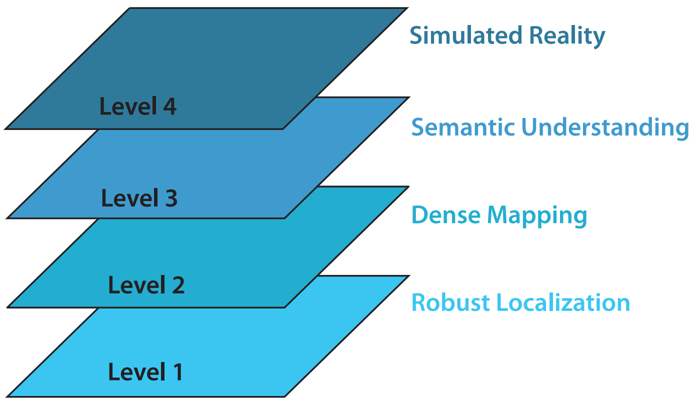


Figure 1.2: Levels in the Spatial AI approach. Image inspired from the CVPR presentation of A. Davison [Davison, 2019].

robustly [Mur-Artal et al., 2015]. Level 2 refers to the dense mapping that reconstructs the environment with a high level of detail [Newcombe et al., 2011]. Level 3 requires the system to identify and recognize the objects around it and to be able to establish semantic relationships [Chen et al., 2017]. And finally, at level 4, we place the term defined as *Simulated Reality* (SR) [Stoica et al., 2017], where the system needs to know the dynamics of the environment around it in order to have a deeper understanding of the world. SR is based on simulating (to understand) the physical phenomena of the environment with which the system interacts. The world is dynamic and evolves continuously and very quickly, so artificial intelligence systems must be prepared to understand these changes. Learning this kind of concepts is a long and complex process, since we must teach a machine how to interpret the physics.

1.1.3 Intelligence Augmentation (IA)

The common assumption today is that AI will replace human labor. However, this is not always true. The tools developed by AI can be used to help, rather than replace, the human being. Once an AI system can understand those physical phenomena that surround it using Simulated Reality techniques, it can give valuable information to a user. This is known as *Intelligence Augmentation* where, as we said in the Introduction section, it allows to increase the human capacities to make decisions with greater knowledge. According to this approach, the ultimate goal is not to teach a machine, the goal is that the machine provides us with relevant information that we could not perceive by ourselves. IA is human-focused, where a human assistance is desired rather than replacement. Finally this translates into having the right information at the right time, a trend we are heading towards with the great development of the connected world.

It is not easy for a machine to interpret or understand the physical phenomena that occur around us. But it is even less simple to the machine to describe us how these phenomena happen. This communication between the system and the user is of great importance in the AI environment, so we have chosen the *Augmented Reality* (AR) framework to transmit all that information.

1.1.4 Mixed Reality (MR)

Augmented reality (AR) is a phenomenon already introduced into our lives. There are numerous examples of current applications in the smartphone markets, and this is because it has passed the valley of research. Right now AR is in a mature stage of development where large companies are making big efforts to make use of its advantages. Although it is a well known term, let us introduce the environment in which augmented reality is framed.

There is a domain called the *Reality-Virtuality Continuum* [Milgram et al., 1994] that includes both the real world (*Real Reality*) and the virtual world (*Virtual Reality*). It is a continuous fusion of both worlds where real and virtual objects coexist in the same environment: Mixed Reality (MR).

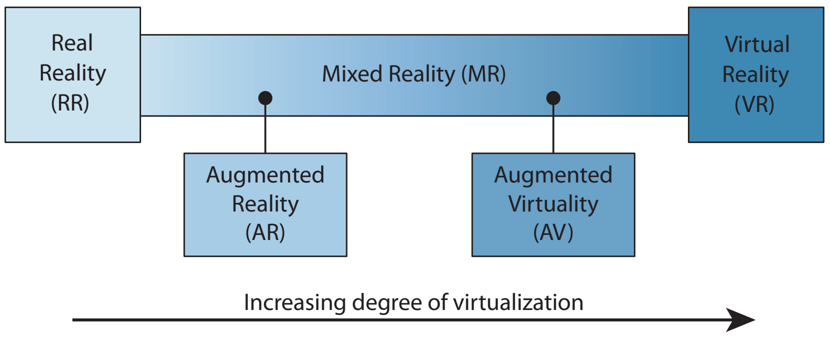


Figure 1.3: Reality-Virtuality continuum that explains how mixed reality consists of a mixture between real and virtual environments. Adapted from [Milgram and Kishino, 1994].

The introduction of virtual objects into our real world, where most objects are real, is known as Augmented Reality [Azuma, 1997] (see figure 1.3). While the introduction of real objects into a virtual environment is known as *Augmented Virtuality* (AV). Since it is a continuum, it is not always easy to separate between AR and AV, and perhaps in a close future we will not be able to distinguish between virtual and real objects, so the term MR begins to be more common than AR.

The degree of interaction between real and virtual worlds is also undergoing great development. At the beginning of AR, the positioning of static virtual objects in the real world in a robust way involved great complexity. With the development of better portable hardware it was possible to add three-dimensional animations with some motion. Improvements in software techniques made it possible to subsequently create better interactions such as occlusions between real and virtual objects. A more natural interaction including collisions was the next step. And finally, the application of the physical laws of our real world on virtual objects closes the set of tools to merge virtual and real worlds, so that the interaction is so natural that virtual objects behave as if they were really in front of the user.

1.1.5 Real-Time Simulations

To develop a system capable of understanding the surrounding physical phenomena, we must give it the ability to work in real time. But since we also want the agent to be portable, two options arise: a remote connection to a computer cluster, or compressing the computation needed to work with a portable system. It is believed that the first option could begin to be contemplated with the arrival of new technologies like 5G and low latencies. For the time being, in this thesis we opt for the second option, as long as it is possible to apply techniques that are reliable enough to represent a physically consistent world and work in real time. Since we use MR to visualize and interact with virtual objects, the video frequencies are fixed in a range of 30 to 60 frames per second (fps). Therefore, data update requirements are imposed by that display frequency, which for the human eye is highly suitable.

Model order reduction methods are based on the compression of the data to reduce the storage memory. But at the same time they must recover the original high-fidelity data in a fast way, fulfilling visualization frequencies. One of the reasons why these methods are so employed is because they *certificate* that the physical equations are accomplished inside a defined range. It's not a matter of simplifying the models as much as possible in order to be able to work in real time (as happens, for example, with certain physical engines in video games), but rather MOR are based in the real physical equations. The power of these techniques allows us to simulate realistic behaviors that describe and adapt perfectly to the changes that occur in the environment. But, of course, the bigger the problem we want to solve, the greater the complexity is required by the method.

1.1.6 On-the-Loop Data Assimilation

Data assimilation is another key concept in this thesis. It is necessary to feed the models (they can also be data-driven) with real-time information from the environment, which can also evolve over time. We need to dynamically inform the system, what is known as *On-the-Loop Data Assimilation*. It is about having a virtual model representative of the real environment, where the virtual model is updated in real time with the changes that happen in reality. This methodology is also called *Digital Twin* [Boschert and Rosen, 2016], and consists of designing a digital copy that is living in the virtual environment but evolves in the same way as its real twin. This has great utility in industry with the new introduction of Big Data and the Internet-of-Things (IoT) where a connected world is foreseen capable of simulating using that digital copy the complete process of a company. The Digital Twin can be applied to processes, management, manufacturing or any concept that could be simulated.

However, going one step further, *ESI Group*[3] defined the Hybrid Twin™ [Chinesta et al., 2018], where the system learns corrections with respect to the models. In case these models were erroneously defined, or the real system suddenly diverges by unforeseen events, it is necessary to provide the system with the ability to learn these variations and adapt the models to evolving environments.

1.2 Motivations and Objectives

The main objective of this thesis is to develop an intelligence augmentation framework to understand the physical phenomena of the environment and provide a user with valuable information about the dynamic changes suffered by the surrounding objects. An illustrative scheme can be seen in Fig. 1.4. The physical understanding is carried out by solv-

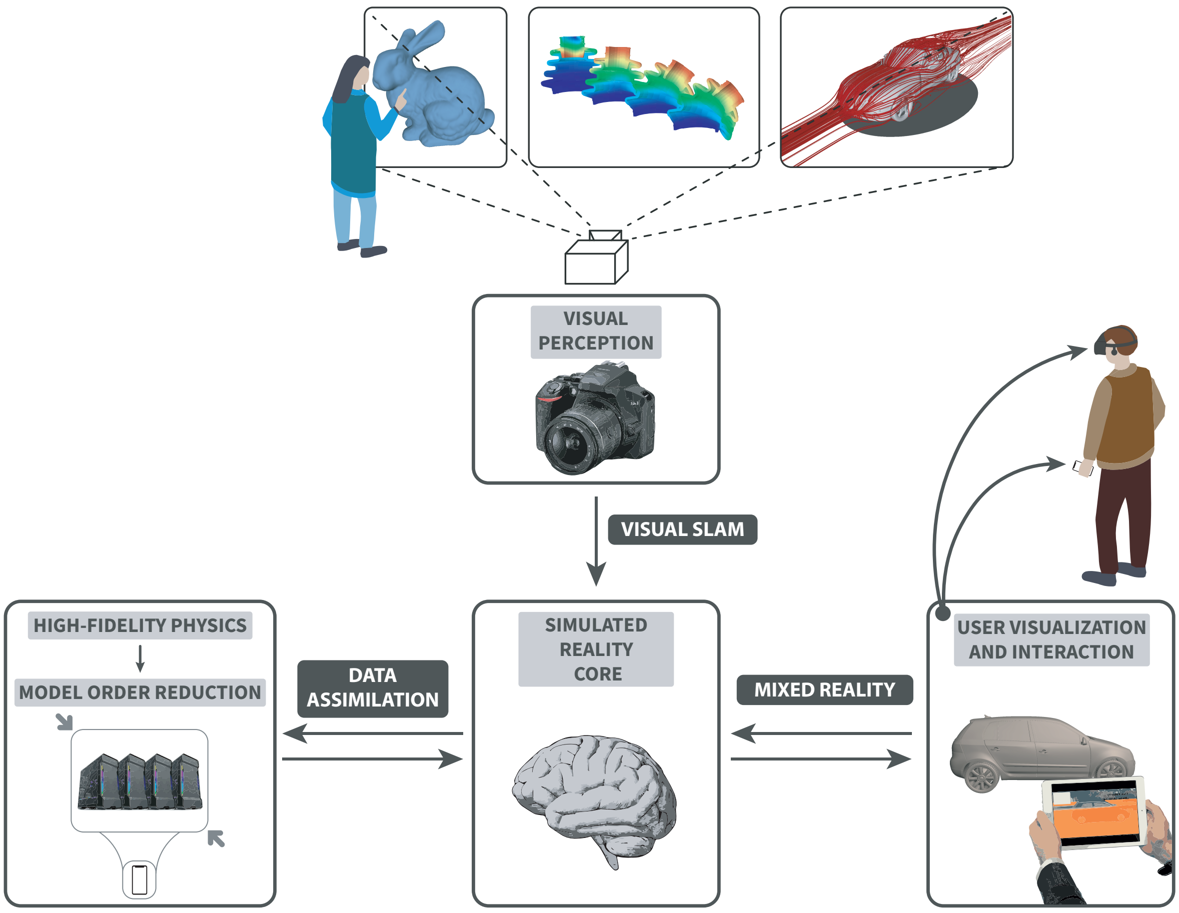


Figure 1.4: Simulated-Reality system overview.

ing the real equations that model the behavior of those objects using simulation based engineering solvers, but usually they need long periods of computation, far from video frequencies. This is why model order reduction are used in this context to reduce the

[3]https://www.esi-group.com/

complexity of the computations and save time and memory resources. MOR methods precompute large multiparametric problems and project them into a more efficient basis to allow real-time evaluations. On the other hand, the interaction with the environment is carried out by video sequences, where mixed reality techniques are used to capture and plot the information, allowing a user interaction and data information of the physical phenomena.

Simulation based engineering science still has great potential, but sometimes the use of simulation solvers is not enough and we need that summation of knowledge that data science can provide through techniques such as model order reduction methods. The estimation of latent variations in the data seems to have outstanding importance, but we must remember that everything is working because we deal with physical models that meet the equations of equilibrium. The sum of these two parts is what makes MOR methods so powerful for real time simulation, that together with a natural interaction based on Mixed Reality creates the full scope of this thesis.

The lack of an environment that allows to simulate the reality in real time is what has motivated our work, which mixes developments on numerical algorithms for solving partial derivative equations (PDE), model order reduction methods, data-based learning, measurement techniques using computer vision and visualization tools based on computer graphics.

Some tasks were defined to accomplish the objective of this thesis:

- Development of model order reduction methods with local approaches applied to non-separable problems.
- Application of non-linear model reduction techniques for complex problems with focus in aerodynamics.
- Deterministic Data-driven approach to update parametric solutions.
- Creation of a framework that allows the fluid interaction between image acquisition, solving partial derivative equations, data-driven updating of parametric equations and the visualization of the result, all in real time.
- Estimation of errors made in measuring and updating models by means of computer vision tools.
- Development of image capture techniques to estimate displacements in deformable objects.
- Visualization of results to improve the representation of information and favor a natural interaction with the user.

We think the scope of this work can have multiple and very varied applications. Some examples can be Industry 4.0, education, surgical intervention aid, entertainment and product design, where in some of them we have already implemented example applications.

1.3 Thesis Outline

This document aims to collect all the framework that is necessary to create a Mixed Reality environment to understand and explain the physical phenomena of the objects. For this purpose, the structure of the document is organized in 7 chapters and 2 appendices, where a self-contained document that includes all the necessary parts has been created.

- Chapter 1: this is the current chapter and includes the introduction, motivation and objectives of the thesis. It aims to give a global idea of the scope of this work while introducing some of the key concepts that make up the work developed.
- Chapter 2: to create a self-contained document, this chapter contains the state of the art concerning the thesis and a brief explanation on the visual spatial localization, to introduce some basic concepts to readers not related to computer vision techniques. This section aims to explain the process of measurement using images, covering the camera model used and its calibration, an introduction to projective geometry, the problem of 3D reconstruction and the extraction of relevant information from images.
- Chapter 3: this chapter aims to describe the basis of model order reduction methods, from their linear aspect to non-linear implementations. It shows a local development to achieve a global non-linear behavior depending on the complexity of the solution, where we have focused on non-separable problems. In addition to the generation of non-linear behavior due to calculation in local domains, other non-linear information extraction techniques such as *kernel-Principal Component Analysis* (k-PCA) are used.
- Chapter 4: this chapter is dedicated to the implementation of a data-assimilation system taking information from a video sequence, from a purely deterministic point of view. The data is used to feed a reduced model that stores the solutions to a given parametric problem. Several examples of deformable solids with non-linear material laws are included, working in real time.
- Chapter 5: in this chapter, we implemented a method of interaction between real objects and deformable virtual objects, where it is intended to simulate the contact between objects of both worlds. Again, the system works with images creating an immersive Mixed Reality environment where the user can interact naturally with virtual objects.
- Chapter 6: this chapter contains a learning method of the macroscopic variation of vehicle aerodynamics to produce a tool capable of estimating the behavior of air flow over a car. It is about using machine learning techniques to extrapolate aerodynamic behavior on new car models.
- Chapter 7: this chapter gathers the thesis conclusions to finally discuss the results obtained.

Chapter 2

Background

Simulated-Reality describes the physical phenomena that objects undergo in real time, something that in general requires a high computational cost. In simple cases like rigid-body motions the system does not require such a complex computation, but to describe the stresses and strains of deformable objects with hyperelastic behaviors interacting in real time with a user implies a strong computing effort. This drawback is even greater if we jump to more complex problems such as fluid mechanics or electromagnetism, where non-linear terms and coupled equations may appear, in addition to requiring very fine meshes to ensure the convergence of the solution. The complexity in these problems becomes intractable and it would be necessary to work with computer clusters to achieve video frequencies. As we can imagine, standard users have no access to this type of resources and in addition the users increasingly prefer the use of portable devices. This is the main reason why we use model reduction. In Sec. 2.1 we introduce the general background of Model Order Reduction methods, as well as a summarized state of the art.

The development of techniques that allow the user to fluently interact with objects is another important ingredient in Simulated-Reality. Experience and interaction of the user with virtual objects must take place in a very natural way. Some devices and a brief introduction to current methods is included in Sec. 2.2.

Finally, we add a fast introduction to computer vision techniques (Sec. 2.3). Although we could integrate this part inside the interaction section (Mixed-Reality), we consider that the acquisition of visual information has such an important role in the Simulated-Reality framework that we must describe it separately. We introduce a quick history of the camera by reviewing some significant findings from the first *camera obscura* descriptions to current devices. To finish, we include some basic concepts about camera models and multi-view reconstruction, showing the problems of rigid and non-rigid *Structure from Motion*.

In the previous chapter we presented some of the concepts and placed the thesis in its technological framework, but it is in this chapter where we collect the scientific environment on which all our work is based. This chapter shows the general background where this thesis is encompassed as well as a summarized state of art. Some of the concepts explained in current chapter were previously published in the following papers:

- Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, vol. 112, no. 12, pp. 1715-1732.
- González, D., Badías, A., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Model order reduction for real-time data assimilation through Extended Kalman Filters. *Computer Methods in Applied Mechanics and Engineering*, vol. 326, no. Supplement C, pp. 679 – 693.
- Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Under revision. Real-time interaction of virtual and physical objects in Mixed Reality applications. *Computational Mechanics*.

2.1 Model Order Reduction Methods

Computer-based modeling and simulation in science has become one of the most valuable and helpful tool in design, development and prediction in the world. The chance of simulate the behavior of any system allows us to understand its response in a previous way. Some big fields like economics, biology, sociology or engineering are some examples of application of computer simulation. Today researches in new techniques of simulation will surely be the basic tools used in next years due the fast evolution of the field and the high requirements of the users. Fast adaptation and continuous development of new models and techniques are some known features in simulation, among other things, by the high quantity of people involved.

The joint between human thirst for knowledge and the increased computing power of today computers has triggered a new revolution in simulation that allows us to obtain the behavior of any system involving high quantities of parameters. But researchers desire to manipulate large data sets to obtain models with high-degree of similarity with the real behavior, something that not always can be satisfied with computers of today.

Model Order Reduction (MOR) [Ryckelynck et al., 2006, Chinesta and Ladevèze, 2014] methods is a group of techniques to reduce the complexity of the high-fidelity models that can be adapted to the field of computational mechanics. From an algebraical side, they transform the solution of a given problem from the original basis to a new and reduced basis preserving the most quantity of information. Traditionally, this is known as transforming the solution from the high-dimensionality problem to the reduced basis problem extracting only the relevant information, where the compressed solution can have non-physical dimensions, but it is possible to build the real solution by performing algebraic operations.

2.1.1 Dimensionality Reduction

Let us assume a governing equation that depends on a vector of parameters $\boldsymbol{\mu} \in \mathcal{P}$ where $\mathcal{P}$ is the set of all possible values of the parameters, being a compact subset of $\mathbb{R}^{\text{n}_{\text{param}}}$. The manifold or solution set is $\mathcal{M}$, where all solutions $\boldsymbol{u}(\boldsymbol{\mu})$ remain [Quarteroni et al., 2015].

$$\mathcal{M} = \varphi(\mathcal{P}) = \{\boldsymbol{u}(\boldsymbol{\mu}) \in V : \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{\text{n}_{\text{param}}}\},$$

where φ is the solution map and V is a suitable Hilbert space. The solution map φ is defined as

$$\varphi : \mathcal{P} \to V, \qquad \boldsymbol{\mu} \mapsto \boldsymbol{u}(\boldsymbol{\mu}).$$

Usually it is not possible to work with the original solution set due to the finite memory of computers, so we have to make a simplification by choosing a suitable discretization technique to obtain the high-fidelity solution set $\mathcal{M}_h$, also known as the discrete manifold, obtained for example using finite element techniques. We assume that $\mathcal{M}_h$ is so close to $\mathcal{M}$ that no differences appear in discretization points. The discretized solution of the equation is $\boldsymbol{u}_h(\boldsymbol{\mu})$ and it belongs to a finite-dimensional subspace V_h of dimension N_h,

$$\mathcal{M}_h = \varphi_h(\mathcal{P}) = \{\boldsymbol{u}_h(\boldsymbol{\mu}) \in V_h : \boldsymbol{\mu} \in \mathcal{P}\} \subset V_h.$$

After applying the approximation to $\boldsymbol{u}(\boldsymbol{\mu})$, we obtain also the discrete solution map

$$\varphi_h : \mathcal{P} \to V_h, \qquad \boldsymbol{\mu} \mapsto \boldsymbol{u}_h(\boldsymbol{\mu}).$$

There are several MOR methods to reduce the dimensionality of the problem with the goal of obtaining a reduced basis with an optimal (hopefully) number of components. The reduced solution thus belongs to a low-dimensional subspace $V_N \subset V_h$ of dimension $N \ll N_h$ where the construction of the basis of V_N is generated from a set of N functions. The particular form of constructing these functions gives rise to the family of model order reduction techniques available in the literature. A classical division can be maid between *a posteri* and *a priori* methods.

- The *a posteriori* methods (see Fig. 2.1) are built after computing some solutions of our system or the whole set of solutions, and are especially useful when the model has a relatively small number of parameters but many observations. The most famous example is *Proper Orthogonal Decomposition* (POD) [Liang et al., 2002] that is based on a statistical procedure called *Principal Component Analysis* (PCA) [Hotelling, 1933]. Other examples are *Reduced Basis* (RB) method [Rozza et al., 2008] or *Empirical Interpolation Method* (EIM) [Barrault et al., 2004]. In general, *a posteriori* methods do not need to know the equation that governs the behavior of the problem, they only need a suitable set of representative snapshots. Non-linear implementations can be also applied using previous transformations to the data, like *kernel-Principal Component Analysis* (k-PCA) [Schölkopf et al., 1998] or *Locally Linear Embeding* (LLE) [Roweis and Saul, 2000], among others.

- The *a priori* methods (see Fig. 2.1) are built before computing any solution of the problem and are based in the knowledge of the equation that governs the behavior of the problem. The solution is not known a priori, but is computed directly in the reduced space: it is projected at the same time it is solved. A representative example of this kind of methods is *Proper Generalized Decomposition* (PGD) [Chinesta et al., 2013a], and there exist also non-linear implementations [Badías et al., 2017].

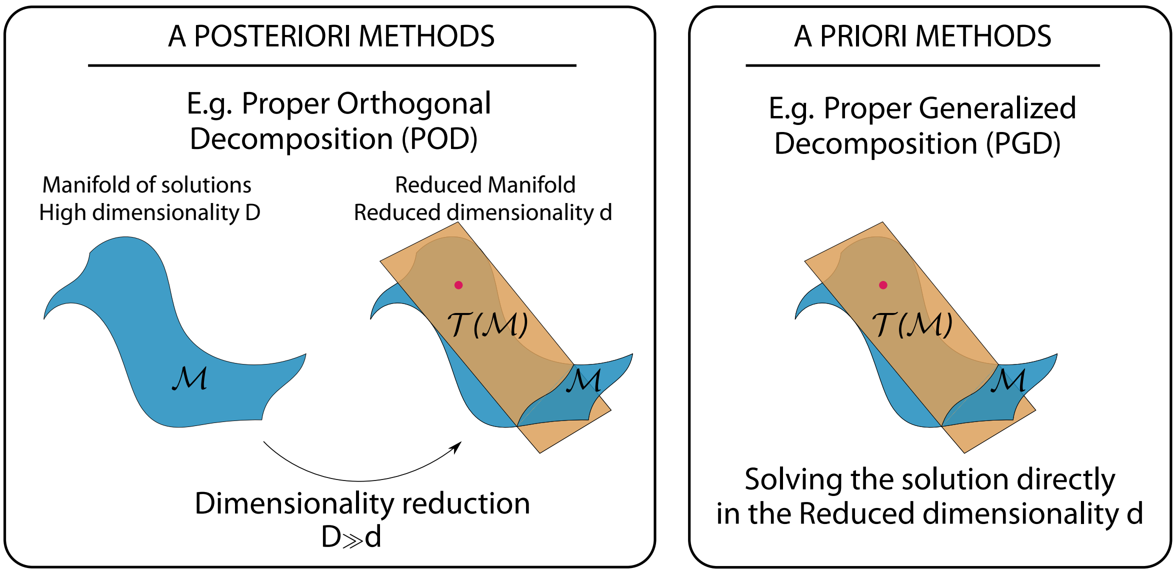

A POSTERIORI METHODS
E.g. Proper Orthogonal Decomposition (POD)
Manifold of solutions
High dimensionality D
Reduced Manifold
Reduced dimensionality d
$\mathcal{M}$
$\mathcal{T}(\mathcal{M})$
$\mathcal{M}$
Dimensionality reduction
D≫d
A PRIORI METHODS
E.g. Proper Generalized Decomposition (PGD)
$\mathcal{T}(\mathcal{M})$
$\mathcal{M}$
Solving the solution directly
in the Reduced dimensionality d

Figure 2.1: Classification of the model order reduction methods in *a posteriori* and *a priori*. $\mathcal{M}$ is the manifold or solution set with dimension N and $\mathcal{T}(\mathcal{M})$ is the tangent plane (or hyperplane) where the solution is projected when a reduced basis is computed. The red circle represents the tangent point.

A tensor example

We assume the precomputed solution of a parametric problem $\boldsymbol{u}(\boldsymbol{\mu})$ is, for example, an N-th order tensor $\mathcal{A}$ depending on a vector of parameters $\boldsymbol{\mu}$ of size N. To particularize the solution of the problem for a set of values of the parameters $\boldsymbol{\mu}$, we need to extract the solution from the tensor structure coordinates $i, j, k, ..., n$ of size N. In a 3D tensor problem (Fig. 2.2) we need to access the component i, j, k of the $\mathcal{A}$ tensor to obtain the solution with parameter values $\mu_{i,j,k}$, being $N = 3$ and $[I_1, I_2, I_3]$ the size of the discretized vectors of $\mu_{i,j,k}$ parameters, respectively.

A tensor structure is a geometric object that describes, in this case, the solution of a problem in an organized way. An N-th order tensor, being $N > 2$, can be decomposed, for example, using CANDECOMP/PARAFAC decomposition [Kolda and Bader, 2009]. This kind of decompositions factorizes a tensor $\mathcal{A}$ into a sum of component rank-one tensors (Fig 2.2) as:

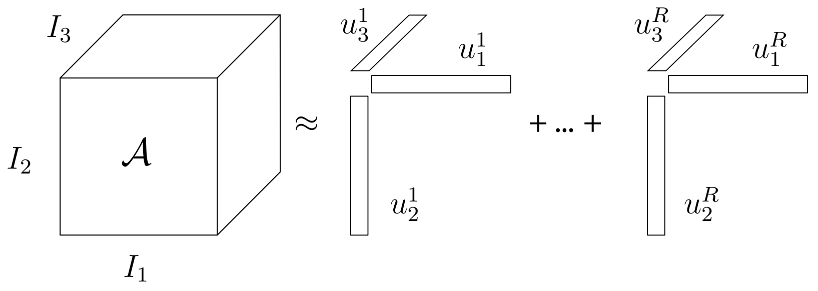


Figure 2.2: Greedy projection of a tensor expressed as the sum of products in separate variables (CANDECOMP/PARAFAC decomposition). Note that at least one of the vectors must not be normalized, so that it contains both the base vectors and the projection coefficients.

$$\mathcal{A} \approx \sum_{j=1}^{R} \prod_{i=1}^{N} \alpha^j u_i^j = \sum_{j=1}^{R} \alpha^j u_1^j \otimes u_2^j \otimes u_3^j \tag{2.1}$$

being R the number of *modes* or sums to approximate the solution and N the order of the tensor or dimensionality of the problem (3 in Fig. 2.2). If (u_1, u_2, u_3) are normalized, α^j has to be estimated to store the coefficients of the tensor $\mathcal{A}$ projected in the separate vectors. There exist also other implementations like high-order extensions of the singular value decomposition method (HOSVD, see the survey [Kolda and Bader, 2009] for more details).

Assuming our goal is to solve a parametric problem with many parameters, we probably fall into a problem called the *curse of dimensionality* [Bellman, 2013], where the amount of data grows exponentially with the dimensionality (and tends to be sparse). One way to avoid this problem and store the data in a compact structure is applying rank-one decompositions. However, instead of pre-computing all the possible solutions to later apply this type of decompositions, we can use *a priori* methods. Although it is necessary to know the differential equation that governs the problem, the advantage is that the solution is projected in separate variables at the same time it is solved.

2.1.2 A posteriori MOR: Proper Orthogonal Decomposition

For completeness, we begin with the most classical model order reduction method, from which all the rest emerge in some sense. Proper Orthogonal Decomposition (POD) method [Berkooz et al., 1993] is a model order reduction method that applies an orthogonal decomposition to a set of observed snapshots to obtain a reduced basis whose vectors are linearly uncorrelated. In other words, it is a method to project the solution in a new basis space with, hopefully, reduced dimensionality using an algebraic decomposition. It comes from the well-known method Principal Component Analysis (PCA) [Karhunen, 1946, Loève, 1963], and it basically consists in a linear transformation of the coordinate axes trying to maximize the variance of the data.

Let's assume that the mechanical behavior of an object is modeled using a linear elastic material under quasi-static conditions (negligible inertia terms). The governing equation for linear elasticity in its strong form tries to find $\boldsymbol{u}(\boldsymbol{x})$ such that

$$\begin{cases} \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}+\boldsymbol{b} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{\sigma}\,\boldsymbol{n} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_t, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_u. \end{cases} \tag{2.2}$$

where $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{b}$ is the body force per unit volume, $\bar{\boldsymbol{t}}$ are the traction forces applied to the solid and $\bar{\boldsymbol{u}}$ are the prescribed displacements (see [Fish and Belytschko, 2007] for a complete description). Using the finite element method and after interpolation with shape functions, integration in the elementary domains and the assembly of the stiffnes global matrix, we can express in a simplified way the equation of the system as

$$\boldsymbol{K}\cdot\boldsymbol{U}=\boldsymbol{F} \tag{2.3}$$

where $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{U}$ is the vector of global displacements in the undeformed configuration and $\boldsymbol{F}$ the vector of loads. Assume also that a set of observations is obtained and stored in matrix form as

$$\boldsymbol{P}=\begin{pmatrix} u_{1,x}^1 & u_{1,x}^2 & \dots & u_{1,x}^M \\ u_{1,y}^1 & u_{1,y}^2 & \dots & u_{1,y}^M \\ u_{1,z}^1 & u_{1,z}^2 & \dots & u_{1,z}^M \\ u_{2,x}^1 & u_{2,x}^2 & \dots & u_{2,x}^M \\ u_{2,y}^1 & u_{2,y}^2 & \dots & u_{2,y}^M \\ u_{2,z}^1 & u_{2,z}^2 & \dots & u_{2,z}^M \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,x}^1 & u_{n,x}^2 & \dots & u_{n,x}^M \\ u_{n,y}^1 & u_{n,y}^2 & \dots & u_{n,y}^M \\ u_{n,z}^1 & u_{n,z}^2 & \dots & u_{n,z}^M \end{pmatrix},$$

with $\boldsymbol{P}$ a matrix with size $D\times M$, where D is the number of spatial coordinates and M is the number of observations of our problem (different deformed configurations of our problem, for example). Assuming this is a three-dimensional problem, $D=3n$ being n the number of nodes in the mesh of the model. Let $\boldsymbol{p}\in\mathbb{R}^D$ be a vector of the $\boldsymbol{P}_{D\times M}$ set of observations of the solution, there exists a linear transformation $\boldsymbol{W}$ that converts each $\boldsymbol{p}$ vector in the original basis to the $\boldsymbol{l}\in\mathbb{R}^d$ vector of the $\boldsymbol{L}_{d\times M}$ reduced basis matrix, where $d\leq D$,

$$\boldsymbol{p}=\boldsymbol{W}\cdot\boldsymbol{l}.$$

The covariance matrix $\boldsymbol{C_{PP}}$ of the observations can be estimated as

$$\boldsymbol{C_{PP}}=E\{\boldsymbol{P}\cdot\boldsymbol{P}^T\}\approx\boldsymbol{P}\cdot\boldsymbol{P}^T,$$

and it is related with the covariance matrix of the reduced basis as

$$\boldsymbol{C_{LL}} = \boldsymbol{W}^T \cdot \boldsymbol{C_{PP}} \cdot \boldsymbol{W}.$$

We can decompose the matrix $\boldsymbol{C_{PP}}$, by employing the singular value decomposition method, in the product of three matrices: $\boldsymbol{U_P}$, $\boldsymbol{\Sigma_P}$ and $\boldsymbol{V_P}$ in the way:

$$\boldsymbol{C_{PP}} = \boldsymbol{U_P} \cdot \boldsymbol{\Sigma_P} \cdot \boldsymbol{V_P^T},$$

where $\boldsymbol{U_P}$ contains the left orthonormal eigenvectors, $\boldsymbol{\Sigma_P}$ contains the eigenvalues and $\boldsymbol{V_P}$ contains the right orthonormal eigenvectors. Since $\boldsymbol{C_{PP}}$ is, by definition, a symmetric matrix, $\boldsymbol{U_P} = \boldsymbol{V_P}$, and therefore

$$\boldsymbol{C_{LL}} = \boldsymbol{W}^T \cdot \boldsymbol{U_P} \cdot \boldsymbol{\Sigma_P} \cdot \boldsymbol{U_P^T} \cdot \boldsymbol{W},$$

$$\boldsymbol{W} = \boldsymbol{U_P} \cdot \boldsymbol{I}_{D\times d}.$$

being $\boldsymbol{I}_{D\times d}$ the identity matrix of size $D \times d$. And finally:

$$\boldsymbol{C_{LL}} = \boldsymbol{I}_{d\times D} \cdot \boldsymbol{\Sigma_P} \cdot \boldsymbol{I}_{D\times d},$$

where we can prove that matrix $\boldsymbol{C_{LL}}$ contains the new basis values of the vectors ordered by variance, where we can extract the latent variables. Coordinates of the new basis are stored in matrix $\boldsymbol{U_P}$, and to estimate the reducibility of the original basis we observe how the values of the diagonal matrix $\boldsymbol{\Sigma_P}$ are changing. Selecting the number of N values from matrix $\boldsymbol{\Sigma_P}$, we create the matrix $\boldsymbol{B}$ as

$$\boldsymbol{B} = \boldsymbol{U}_{\boldsymbol{P}_{[1,\ldots,N]}}.$$

The vector containing the displacement solution $\boldsymbol{u}$ for a determined configuration is thus approximated from the new basis as

$$\hat{\boldsymbol{u}} = \boldsymbol{B} \cdot \boldsymbol{\xi}, \tag{2.4}$$

where $\boldsymbol{\xi}$ are the coordinates of the solution in the reduced new basis. It means we are projecting our original solution in the reduced space using $\boldsymbol{B}$. Going back to Eq. (2.3), we apply the reduction of Eq. (2.4) to obtain

$$\boldsymbol{K} \cdot \boldsymbol{B} \cdot \boldsymbol{\xi} = \boldsymbol{F}.$$

The use of the Galerkin projection method results in

$$\boldsymbol{B^T} \cdot \boldsymbol{K} \cdot \boldsymbol{B} \cdot \boldsymbol{\xi} = \boldsymbol{B^T} \cdot \boldsymbol{F}.$$

To finally solve the system we only compute the values of $\boldsymbol{\xi}$, which are the coefficients of our projected solution that solves the system

$$\boldsymbol{\xi} = (\boldsymbol{B^T} \cdot \boldsymbol{K} \cdot \boldsymbol{B})^{-1} \cdot \boldsymbol{B^T} \cdot \boldsymbol{F}. \tag{2.5}$$

It is important to highlight that we are solving the reduced system, not the high-dimensional one, so Eq. (2.5) can be expressed

$$\boldsymbol{\xi} = \boldsymbol{K}_{\text{red}}^{-1} \cdot \boldsymbol{F}_{\text{red}}, \tag{2.6}$$

where $\boldsymbol{K}_{\text{red}}$ is the reduced stiffness matrix with size $d \times d$ instead of the original matrix with size $D \times D$. The reduction is also applied to the source term. To finally build the solution in the original space, we need to reproject the data by applying Eq. (2.4).

2.1.3 A priori method: Proper Generalized Decompositions

The proper generalized decomposition (PGD) is an *a priori* model order reduction method. This means that no results or observations of the system are necessary to construct the low-rank approximation to the unknown field. The method is based in two stages: an off-line step where the computation of the low-rank approximation is performed; and an on-line phase where the parametric solution is evaluated (rather than computed) under real-time constraints. Essentially, PGD assumes a separate (affine) representation of the unknown displacement field. The precise form of the functional, low-rank approximation is determined by means of a greedy algorithm.

Basic ingredients

We assume that we are looking for a suitable approximation (usually by finite elements) of a space and time dependent field $\boldsymbol{u}(\boldsymbol{x}, t)$ that can be expressed as

$$\boldsymbol{u}(\boldsymbol{x}, t) \approx \sum_{i=1}^{N_{\text{MOR}}} \boldsymbol{F}_i(\boldsymbol{x}) \circ \boldsymbol{G}_i(t), \tag{2.7}$$

where $\boldsymbol{F}_i(\boldsymbol{x})$ and $\boldsymbol{G}_i(t)$ represent the so called *modes* of the approximation, i.e., N_{MOR} vector-valued functions approximated in a finite element sense that best approximate the unknown field $\boldsymbol{u}$. The symbol "$\circ$" represents the Hadamard or Schur (component-wise) product of vectors.

The simplest way to determine an optimal approximation of the type given by Eq. (2.7), sometimes referred to as *affine* or *separate* decomposition, is

$$\boldsymbol{u}(\boldsymbol{x}) \approx \sum_{i=1}^{N_{\text{MOR}}} \alpha_i(\boldsymbol{x}) \boldsymbol{F}_i(\boldsymbol{x}). \tag{2.8}$$

We try to estimate a suitable set of coefficients $\alpha_i(\boldsymbol{x})$ that weight some space-dependent functions $\boldsymbol{F}_i(\boldsymbol{x})$. These spatial functions can be obtained by Principal Component Analysis (PCA) (as exaplined in the previous section) from a set of snapshots of the evolution of the system. Once the modes $\boldsymbol{F}_i(\boldsymbol{x})$ have been identified, by injecting the approximation given by Eq. (2.8) into the weak form of the differential equation governing the problem, weighting coefficients α_i can be determined. Other methods like Reduced Basis (RB) employ directly a set of snapshots of the system, whose parameter value is chosen under well-defined error criterion [Quarteroni et al., 2015]. Applying the same approach to space and time spaces we may obtain the Eq. (2.7), where the α_i coefficients are included inside $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$ functions. And also, we can expand the same approach to higher dimensional problems

$$\boldsymbol{u}(x_1, x_2, \ldots, x_D) \approx \sum_{i=1}^{N_{\text{MOR}}} \boldsymbol{F}_i^1(x_1) \circ \boldsymbol{F}_i^2(x_2) \circ \ldots \circ \boldsymbol{F}_i^D(x_D) \tag{2.9}$$

where $\boldsymbol{u}(x_1, x_2, \ldots, x_D)$ is the solution of the equation depending on D independent variables. These D variables can be physical space variables like cartesian coordinates or parameters that modify the behavior of our system. Functions $\boldsymbol{F}_i(x_j)$ store the *energy* of the solution of the equation projected on variable j. A problem with dimension $D > 2$ implies that we need to employ a different strategy (since there is no equivalent to the PCA in three or more dimensions, as there is no unique—nor optimal—rank-1 decomposition of a tensor of order three).

Proper Generalized Decompositions [Chinesta et al., 2011] [Chinesta et al., 2010] [Cueto et al., 2016] [Badías et al., 2017] do not employ any snapshot of the system. They use a greedy algorithm to determine each of the sums of Eq. (2.9). Within each sum, each mode (its finite element approximation) is determined by solving the resulting non-linear problem that appears when we introduce a solution as product of functions. Different methods can be used, where an alternating direction strategie seems to be the simplest minimization method.

The employ of appropriate error estimates [Ammar et al., 2010] [Moitinho de Almeida, 2013] [Ladeveze and Chamoin, 2011] [Rozza et al., 2008] [Alfaro et al., 2015] allows to fix the number of modes n_{MOR} to employ in the approximation, by controlling the error in the reduced-order approximation. The reducibility of the solution (i.e., the number of necessary terms N) depends on the smoothness of the solution map, the form of the affine expansion of the parameter functions and the analyticity of the problem [Quarteroni et al., 2015].

PGD approach presents several advantages. First of all, the general approximation can be computed off-line and stored advantageously in memory in the form of a set of vectors. These vectors include only the nodal values of the functions $\boldsymbol{F}_i^j$. At any other point of the model, the value of the field $\boldsymbol{u}$ is obtained by finite element interpolation. So model order reduction methods allow to post-process, rather than simulate, under real-time constraints. Second, this separate representation of the unknown field presents important advantages for the solution of inverse problems, as will be explained below. Sensitivities can be computed straightfowardly thanks to this precise form of the approximation.

To better explain the application of PGD method, we present an introductory example. As a toy model we consider another time a linear elasticity problem, the case of a cantilever beam in which the position of the applied load is considered as an input parameter to the system.

An introductory example: linear elasticity

Let us assume a vertical and constant force F is applied over the upper part of a cantilever beam. The unknown field of the problem is the displacement field $\boldsymbol{u}(\boldsymbol{x}, s)$, which depends on the spatial position $\boldsymbol{x}$ and, of course, on the position on the load F, s (see Fig. 2.3). The load can be applied at any point of the upper part $s \in \bar{\Gamma} \subset \partial\Omega$, with $\bar{\Gamma}$ the portion of the boundary Γ_t with non-vanishing Neumann conditions.

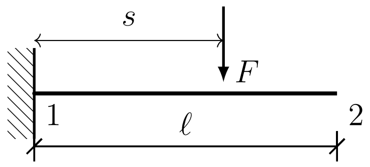


Figure 2.3: Cantilever beam problem. A moving load F is parameterized by its position coordinate s.

The strong form of the governing equation under infinitesimal strain theory is the same as Eq. (2.2), but adding s as a parameter

$$\boldsymbol{u}(x, y, s) = \begin{cases} \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{b} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{\sigma}\, \boldsymbol{n} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_t, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_u. \end{cases} \tag{2.10}$$

The constitutive equation represents the material behavior and relates stress and strain fields,

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon},$$

with $\mathbf{C}$ the fourth-order constitutive tensor, $\boldsymbol{\varepsilon}$ the strain tensor and ":" the double contraction tensor product. Assuming a hyperelastic framework, this constitutive tensor can be obtained by differentiating the strain energy density function Ψ, whose precise form is usually determined from physical experiments (constitutive model). Function Ψ is in our case

$$\Psi(\boldsymbol{\varepsilon}) = \frac{1}{2} E \boldsymbol{\varepsilon}^2,$$

where parameter E is the Young's or elastic modulus. Finally, the strain-displacement (kinematic) equations are

$$\boldsymbol{\varepsilon} = \frac{1}{2}[\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T].$$

Keeping in mind that $\boldsymbol{u}$ depends on $\boldsymbol{x} = (x, y)$ and s, the weak form of the governing equation under the PGD formalism can be described as

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\boldsymbol{s}} \boldsymbol{u}^* : \boldsymbol{\sigma} \, d\Omega \, d\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_t} \boldsymbol{u}^* \, \boldsymbol{t} \, d\Gamma_t \, d\bar{\Gamma} \tag{2.11}$$

where $\bar{\Gamma}$ represents the accesible object boundary, Γ_t the boundary where traction is applied, $\boldsymbol{u}^* \in H_0^1(\Omega)$ is an arbitrary test function in the appropriate Sobolev space of functions vanishing along the Dirichlet boundary of the solid and $\boldsymbol{\nabla}_{\boldsymbol{s}} = \frac{1}{2}[\boldsymbol{\nabla} + (\boldsymbol{\nabla})^T]$ is the symmetric gradient operator. The solution $\boldsymbol{u}(\boldsymbol{x}, s)$ is assumed to be expressed in separate form,

$$\boldsymbol{u}(\boldsymbol{x}, s) \approx \sum_{i=1}^{N_{\mathsf{MOR}}} \boldsymbol{F}_i(\boldsymbol{x}) \circ \boldsymbol{G}_i(s),$$

An alternating direction scheme is used to obtain the N_{MOR} functions $\boldsymbol{F}_i(\boldsymbol{x})$ and $\boldsymbol{G}_i(s)$ so that, at iteration p we search the $p+1$ functions $\boldsymbol{R}(\boldsymbol{x})$ and $\boldsymbol{S}(s)$ that enrich the solution

$$\boldsymbol{u}_{p+1}(\boldsymbol{x}, s) = \boldsymbol{u}_p(\boldsymbol{x}, s) + \boldsymbol{R}(\boldsymbol{x}) \circ \boldsymbol{S}(s).$$

Therefore, the test function is

$$\boldsymbol{u}^*(\boldsymbol{x}, s) = \boldsymbol{R}^*(\boldsymbol{x}) \circ \boldsymbol{S}(s) + \boldsymbol{R}(\boldsymbol{x}) \circ \boldsymbol{S}^*(s).$$

The load F has been designed as a unitary force acting along the vertical axis y ,

$$\boldsymbol{t} = F \, \delta(x - s) \boldsymbol{e_y}, \quad x \in \Gamma_t,$$

where δ is the Dirac delta function. Function $\boldsymbol{t}$ needs also to be expressed in separate form to comply with the PGD formalism, i.e.,

$$t_y \approx \sum_{j=1}^{m} h_j(\boldsymbol{x}) \, k_j(s); \qquad t_x = 0,$$

where m is the number of sums to approximate function t_y and $h_j(\boldsymbol{x})$, $k_j(s)$ are the functions in space and load position, respectively. Remember that PGD is an *a priori* method, so to solve the non-linear product of functions $\boldsymbol{R}(\boldsymbol{x})$ and $\boldsymbol{S}(s)$, the algorithm uses an alternating direction scheme based on the fixed point method (described in Algorithm 2.1) at every enrichment iteration. Any other linearization method, such as Newton iterations, can be used. For a detailed description of the problem and a Matlab implementation, the interested reader can consult [Cueto et al., 2016].

The first modes of $\boldsymbol{F}(\boldsymbol{x})$ (only its vertical component, since it is a vector-valued displacement field) and $\boldsymbol{G}(s)$ are shown in Fig. 2.4.(a) and 2.4.(b), respectively. Finally, the sum of the products of pairs of modes builds the approximate solution, see Fig. 2.4.(c). The particularization of this solution at a singular value $s = s_t$ gives the vertical displacement along the beam, as can be noticed in the bottom part of Fig. 2.5. Of course, the whole solution stores the vertical and horizontal displacements $\boldsymbol{u}(x, y)$ for all points of the beam, not shown in Figs. 2.4 and 2.5 for clarity.

Algorithm 2.1: Pseudo-code of the greedy algorithm used by the PGD method.

```
Initialization;
While (NMode $<$ maxNModes) & (Tol $>$ Error) do
{
    EnrichmentInitialization;
    While (Iters $<$ maxIters) & (enrTol $>$ enrError) do
    {
        $R \leftarrow$ EnrichFromS;
        $S \leftarrow$ EnrichFromR;
        enrError $\leftarrow$ enrErrorEstimation;
        Iters++;
    }
    $F$(NMode) $\leftarrow R$;
    $G$(NMode) $\leftarrow S$;
    Error $\leftarrow$ ErrorEstimation;
    NMode++;
}
```

2.1.4 Parametric Solution

Model Order Reduction techniques tend to be more profitable the more dimensions the problem has. For this reason, MOR methods are usually applied to problems for which the exploration of the parametric space becomes prohibitive (curse of dimensionality). A simple example to understand this concept is shown here, taking the Young's modulus E as an additional parameter inside the solution. This means that we can solve Eq. (2.10) for a range of values of E. The approximate solution in separate variables is now

$$\boldsymbol{u}(\boldsymbol{x}, s, E) \approx \sum_{i=1}^{N_{\text{MOR}}} \boldsymbol{F}_i(\boldsymbol{x}) \circ \boldsymbol{G}_i(s) \circ \boldsymbol{H}_i(E),$$

where $\boldsymbol{H}_i(E)$ are the parametric modes depending on E, and the structure of the solution can be seen in Fig. 2.5. To obtain the displacement field of the beam we must particularize the solution for a value of $s = s_t$ and $E = E_t$,

$$\boldsymbol{u}(\boldsymbol{x}, s_t, E_t) \approx \sum_{i=1}^{N_{\text{MOR}}} \boldsymbol{F}_i(\boldsymbol{x}) \circ \boldsymbol{G}_i(s_t) \circ \boldsymbol{H}_i(E_t)$$

Note that the last expression involves the multiplication of N_{MOR} vectors $F_i(\boldsymbol{x})$ times N_{MOR} scalars $\boldsymbol{G}_i(s_t)$ and $\boldsymbol{H}_i(E_t)$, involving a high efficiency to build the solution in the original space. The solution is stored in memory in terms of vectors comprising the nodal values of the modes at the discretization nodes.

After analyzing some MOR methods and their application to multiparametric problems, it is time to move on to the part of user interaction with the spatial environment.

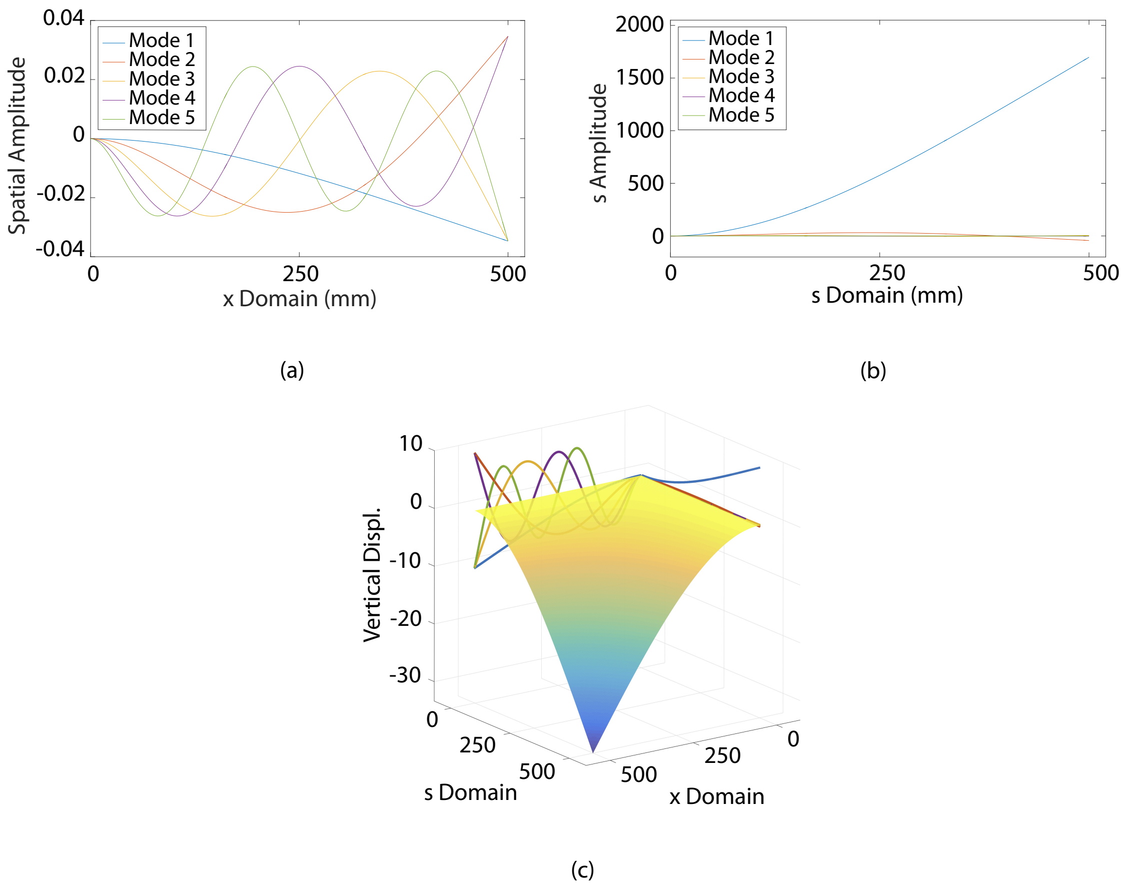


Figure 2.4: Separated solution of the vertical displacement. (a) first five modes depending on space. (b) first five modes depending on the position of the load. (c) Built solution using the sum of the product of modes. Notice how the result of the PGD approximation provides with a sort of response surface, computed entirely in one single simulation, without the need of any parameter space sampling strategy.

2.2 Spatial Interaction

New technologies are bringing better tools to improve the mixed reality experience. These tools are in the form of both hardware and software. On the hardware side we have the development of new devices to show information to the user, such as smartphones, or high resolution virtual reality glasses[1]; we also find great efforts on the development of devices to capture information around us (RGB-D systems[2], stereo cameras); and even systems that include both the capture and visualization of information, such as *Microsoft Hololens 2*[3] or *Magic Leap One*[4], which allow to capture the environment, the visualization of virtual objects and the interaction using built-in controls.

[1]StereoLabs ZED Mini on Oculus Rift. https://www.stereolabs.com/zed-mini/setup/rift/
[2]Microsoft Kinect for Windows. https://developer.microsoft.com/en-us/windows/kinect
[3]Microsoft Hololens 2. https://www.microsoft.com/en-us/hololens
[4]Magic Leap One Headset. https://www.magicleap.com/magic-leap-one

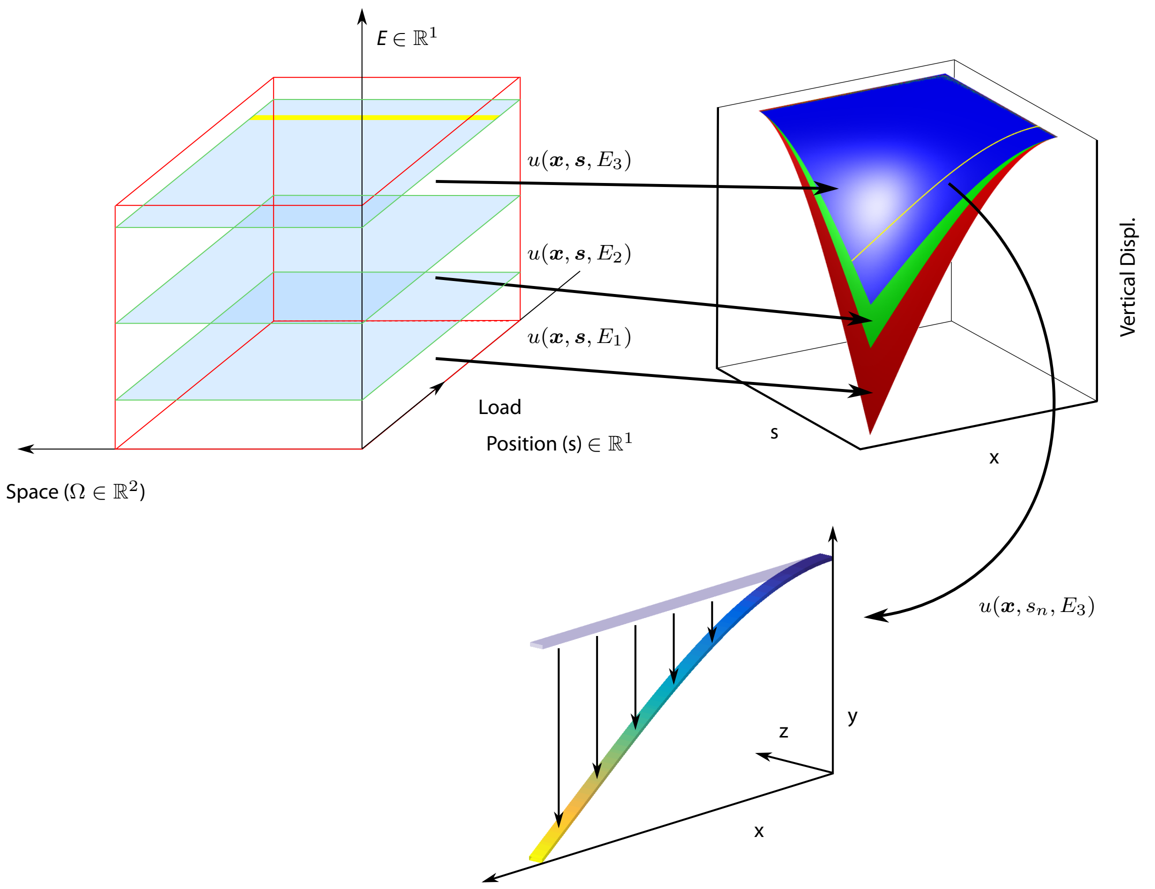

$E \in \mathbb{R}^1$
$u(\boldsymbol{x}, \boldsymbol{s}, E_3)$
$u(\boldsymbol{x}, \boldsymbol{s}, E_2)$
$u(\boldsymbol{x}, \boldsymbol{s}, E_1)$
Load
Position (s) $\in \mathbb{R}^1$
Space ($\Omega \in \mathbb{R}^2$)
Vertical Displ.
s
x
$u(\boldsymbol{x}, s_n, E_3)$
y
z
x

Figure 2.5: Three-field parametric solution depending on space x, position of the load s and elastic modulus E. The parametric response of the system has been here represented as a stack of response surface-like solutions.

From the software point of view, a great work has been done to generate new environments to simplify the process of capturing data, such as the new development kits *Apple ARKit 3*[5] or *Google ARCore*[6]; other software development kits are more focused and integrated into helmets such as Hololens; there are also great advances in visualization libraries such as *OpenGL*[7] or other propietary libraries like *Nvidia CUDA*[8] or *Apple Metal*[9]; and also there has been a big development in new techniques to take robust measures from a scene, such as ORB-SLAM [Mur-Artal et al., 2015] or LSD-SLAM [Engel et al., 2014], among others.

Many AR applications have been developed in different fields such as consumer applications [Yovcheva et al., 2012], industrial environments [Nee et al., 2012], shopping applications (fitting room) [Pachoulakis and Kapetanakis, 2012] or education [Wu et al., 2013],

[5]Apple ARKit 3. `https://developer.apple.com/augmented-reality/`
[6]Google ARCore. `https://developers.google.com/ar/`
[7]OpenGL Library. `https://www.opengl.org/`
[8]NVidia CUDA. `https://developer.nvidia.com/cuda-zone`
[9]Apple Metal. `https://developer.apple.com/metal/`

among others. Video games have been adapted to smartphones, having acquired great popularity and caused a milestone so relevant that has led to the study of the behavior of the users [Howe et al., 2016] and the changes on the habits of these users [LeBlanc and Chaput, 2016]. We believe that augmented reality is understood as the positioning of static information in space [Fedorov et al., 2016] or animated sequences [Paavilainen et al., 2017], but in very few cases deformable solids are taken into account [Haouchine et al., 2012]. We are looking for a simple and natural interaction with virtual objects, which can deform as if they were really in front of the user. The union between machine learning, computer graphics and computer vision can bring great results, where each one provides the necessary tools to obtain the desired result.

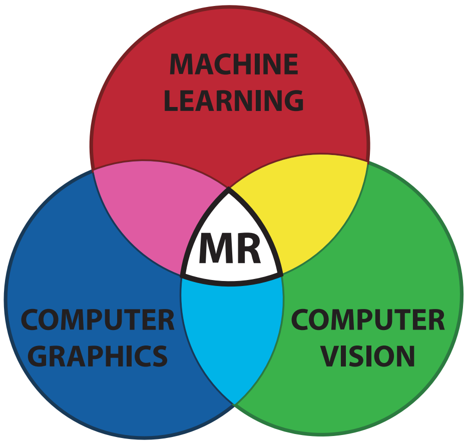


Figure 2.6: Mixed Reality (MR) as the interaction of three sciences: machine learning, computer graphics and computer vision.

2.2.1 User Interaction

Regarding the user interaction experience, there are many published works. Some papers are focused in hand detection using appearance detectors over monocular cameras [Kölsch and Turk, 2004]. Other works use stereo cameras or RGB-D systems [Suarez and Murphy, 2012] for hand localization and gesture classification. There was also a breakthrough in the creation of devices for tracing hand movements, such as the *Leap Motion* sensor [Potter et al., 2013] or the *Kinect* system [Ren et al., 2013]. In recent years, some headsets such as the *Microsoft Hololens* also developed a good tracking of manual gestures [Chaconas and Höllerer, 2018]. There are also haptic devices such as robotic arms [Quesada et al., 2018] and even gloves that allow the user to feel a touch experience [Perret and Vander Poorten, 2018]. However, our work aims to go further, and not only perform the interaction between virtual objects and the hands of the user, but allow an interaction with any real object. There are some implementations [Lundgren, 2017] that also pretend to simulate the collision and occlusions between any type of object, but we remember that in this work we also solve the real physics of virtual deformable objects.

Human visual sense is very intrusive. An enormous quantity of information can be extracted and processed in our brain coming from visual source, simply by observing a few images. The same behavior appears in camera acquisition, where an image can provide a high quantity of information. A strong development in augmented reality in the fields of computer graphics and computer vision has been carried out for these reasons. Visual augmented reality is based in the addition of graphical information to interact with reality, so it requires great efforts in graphical modeling. On the other side, to display the augmented information emulating the real behavior, we need to fix virtual objects inside the real world. This means we need to know the map of the real world to locate the objects that we are seeing, and also we need the camera position and orientation to project virtual objects in the camera reference. This process is known as Structure-from-Motion (SfM), and it consists in the estimation of rigid 3D structures from planar images using different camera view points (see Sec. 2.3.3 for more details).

2.2.2 Data Assimilation Process

If the information to augment the video is extracted from numerical simulations, the difficulties are mainly two-fold. On one hand, we need to couple the computer model to the physical environment, through the lens of the camera. This process is known as *registration*. On the other hand, we have to update in real time the solution by feeding the model with the camera measurements. If we consider, for instance, deformable solids with general non-linear constitutive laws under finite strain settings, the problem of solving such a model 30 times per second becomes an evident bottleneck. That is why we use MOR methods. But we have to introduce new information at any frame from the real environment. So the problem is cast in the form of a classical data assimilation procedure, for which the minimization of a functional is needed. This minimization procedure is greatly beneficed from the separated parametric form of the MOR methods.

This last point can be seen, indeed, as an inverse problem in which some parameters, or even the whole state of the system, governed by a set of Partial Differential Equations (PDEs) must be identified from a set of experimental measurements. This inverse problem can be solved from a deterministic point of view, if we postulate that the sought values are perfectly well matched by a numerical discretization of the governing PDEs [Gonzalez et al., 2012] [Ghnatios et al., 2012]. The minimization process is

$$\boldsymbol{\mu} = \underset{\boldsymbol{\mu}^* \in \mathcal{P} \subset \mathbb{R}^p}{\operatorname{argmin}} \, \mathcal{J}(\boldsymbol{\mu}) = \sum_{j=1}^{n_{\mathtt{meas}}} \left(\boldsymbol{u}^{\mathtt{meas}}(\boldsymbol{x}_j) - \boldsymbol{u}^h(\boldsymbol{x}_j, \boldsymbol{\mu}) \right)^2 .$$

The minimization process provides an approximation to the true values of the set of parameters $\boldsymbol{\mu}$. Here, $\boldsymbol{u}^{\mathtt{meas}}$ represents the vector of measured displacements on the object boundary, and $n_{\mathtt{meas}}$ is the number of measurements. The superscript h indicates a finite element discretization of the displacement field.

Even if this type of procedure can provide very good results [Ghnatios et al., 2011] [Ghnatios et al., 2012] [Nadal et al., 2015] [Gonzalez et al., 2012], other approaches can be applied if we assume that there is some degree of uncertainty associated to both the environment (we usually do not know the value of applied forces, for instance) and the measurement process (unavoidable noise associate to experimental devices). This leads to a formulation of the problem under a Bayesian framework [Prudencio et al., 2013] [Prudencio et al., 2014]. In the Bayesian uncertainty quantification (UQ) framework, both the set of parameters $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$ and the observations are considered as random variables subjected to some probability density functions (PDF). Under this rationale, one of the most popular strategies is that of Kalman filtering [Kalman, 1960].

In order to continue with the interaction of the Simulated Reality system with the scene, we now explain some basic concepts of computer vision applied to Mixed Reality.

2.3 Fast Introduction to Computer Vision

As we have already seen, computer vision has great weight in mixed reality applications. This is the reason why we want to show the most relevant aspects in the current section. To introduce the process of photography, we made a brief review of the history of the camera, showing only some of the most relevant milestones that have been developed throughout the history. Later, we talk about the camera models used in computer vision, multi-view reconstructions and relevant information extraction from images. In case the reader wants to go deeper we recommend reading the well-known book [Hartley and Zisserman, 2003], which explains the basic photogrammetric aspects in computer vision.

Quick history of the camera

Although there is a diversity of opinions, it seems clear that the first camera was the *camera obscura*. It consisted of a *"darkened room with a white wall or screen on one side, and on the other a small opening facing some object or scene that could be brightly illuminated."* [Gage, 1914]. The first documented evidence of the *camera obscura* is published in *Problemata*, a book attributed to Artistotle published in the 4th century B.C. [Hammond, 1986]. Aristotle rejected the vision theory of Plato that assumed eyes to emmit light rays, and he set the principles of the diaphragm in today's cameras by observing that the smaller the hole, the sharper the image was. Euclid's *Optics*, published in 300 B.C., assumed that light travels in straight lines, being the *camera obscura* a demostration of that[10].

Afterwards, during the 9th and 10th centuries, there was a parallel development in the Chinese and Arab communities. The Greeks had previously identified the basic concepts of the *camera obscura*, but its application in image formation had not yet taken place. It is in China, in the 9th century, when the first record of an image formed in a true *camera*

[10]Ancient Greece: Optics. Michael Lahanas. `https://bit.ly/2Y8oafn`

obscura occured [Hammond, 1986]. Tuan Chheng Shih, in the early 9th century, recovers the study of the pinhole and *camera obscura* [Needham et al., 1971], observing the inverted image but confusing its conclusions. Subsequently, Shen Kua correctly explained the inversion of the pinhole image and Yu Chao-Lung, in the 10th century, studied the aperture, and direction and divergence of the light rays [Hammond, 1986]. However, the Arabs developed a deeper theory of theoretical concepts in optics and vision, retrieving and translating the scientific documents previously written by Greek civilization. Again in the 9th century, Al-Kindi proves the rectilinear propagation of light rays, where a luminous body (a candle) is projected inside the dark chamber. Al-Kindi defends and employs Euclid's knowledge of optics, publishing his well-known *De Aspectibus*, a series of geometrical demonstrations based on a set of axioms [Adamson, 2006, Szulakowska, 2000]. He did not agree with Euclid's idea that the eyes emit rays that allow us to see, but confirms Aristotle's theories of vision by explaining the formation of shadows. Also relevant is the work of Alhazen, around 11th century, where the *camera obscura* is used to observe eclipses (in his essay "On the Form of the Eclipse") and demonstrated the human vision and the structure of the eye and image formation [Hammond, 1986], which later had a great influence on the theory of Kepler [Lindberg and Lindberg, 1981].

Later, deeper developments took place in the Latin West, carried out by Bacon, Witelo or Pecham, among others, in the 13rd century [Doble, 2012, Peckham and Lindberg, 1972]. However, it is in the 15th century when a remarkable interest in the *camera obscura* arose by the Italian Leonardo da Vinci, who demonstrated some optical principles such as inversion of images, non-interference, intensity of light or shade and contrary motion. He also extracted some analogies with the eye and explored different shapes and number of apertures [Veltman, 1986].

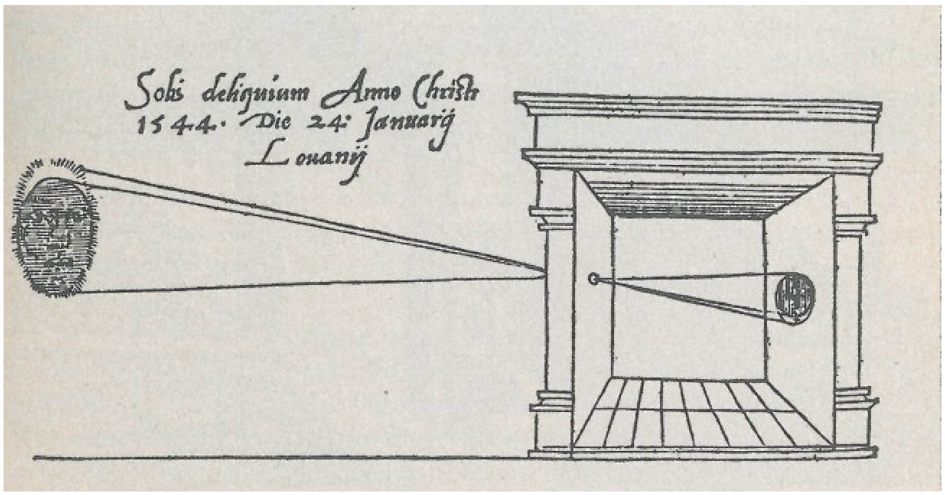


Figure 2.7: *Camera obscura*, drawn by Reinerus Frisius Gemma, published in *De Radio Astronomica et Geometrico*, 1545, considered the first illustration of a *camera obscura*. It shows how Reinerus observed an eclipse of the Sun on 24 January 1544, being the first published picture of the *camera obscura*.

In 1551 Girolamo Cardano apparently introduced the use of convex lenses to permit a larger aperture and project more deatiled images [Ilardi, 2007]. He discussed the use of lenses, mirrors and prisms to improve images and correct the inversion. At that time, the *camera obscura* was also used as a means to help artists draw city views and landscapes, where its use is precisely described in the work of Daniele Barbaro [Ilardi, 2007]. But like da Vinci, the image had not yet been inverted, which means extra work for the painter who must invert the image in his paintings. This problem was solved in 1585 when Giovanni Battista Benedetti incorporated a 45-degree angle mirror for invert and reverse the images. Also portable *camera obscura* systems are proposed, such as the one described by Athanasius Kircher, to be able to help in painting anywhere. As we can see in Fig. 2.8, these were not yet very portable proposals.

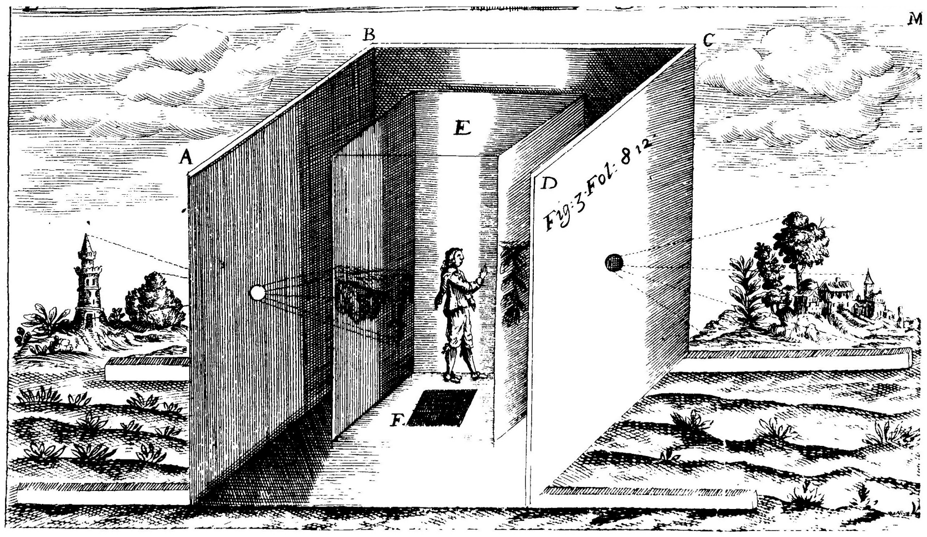


Figure 2.8: Portable *camera obscura* drawn by Athanasius Kircher in Ars Magna Lucis Et Umbra, 1646.

It is curious that, although it had been in use for centuries, the first time the term of *camera obscura* is used is in 1604, by Johannes Kepler. He developed and justified a new theory of optical imagery, described the inverse-square law governing the intensity of light and also attached an analogy between the eye and the *camera obscura* [Dupré, 2008].

Although important developments continued to take place, we jump into the 19th century, where the first photographs are taken. It is Joseph Niécephore Nièpce who takes the oldest image that exists today, known as *View from the Window at Gras* (1826–7), taken with a *camera obscura* from the upper-story of Nièpce's summer house in Saint-Loup-de-Varennes, France [Emerling, 2013]. Initially, it was believed that about 8 hours of exposure were needed (taking into account how sunlight is reflected on buildings) but later it was discovered that even days were needed [Niépce, 2017].

In 1835 Henry Fox Talbot developed a process for creating relatively quick photographs, which he also makes public. This method uses negatives from which the images are ob-

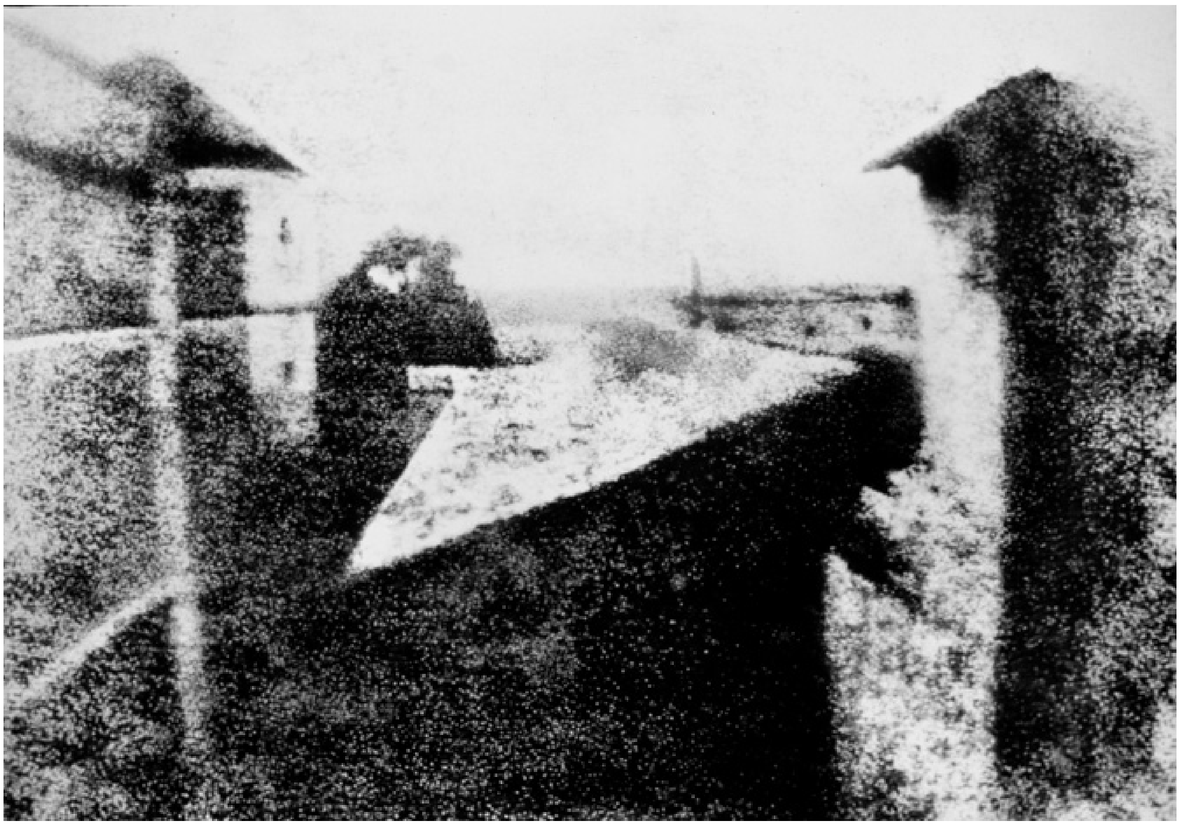
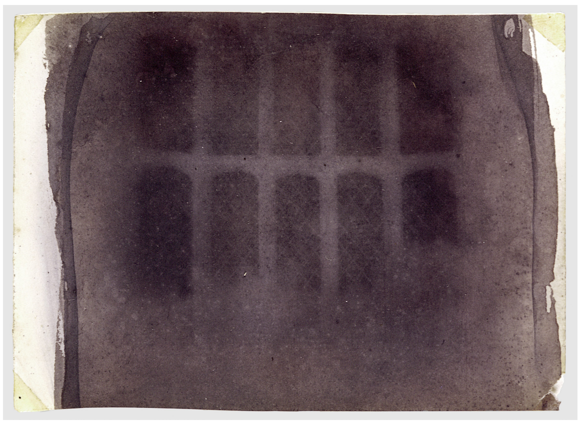


Figure 2.9: *Point de vue du Gras* by Joseph Nicéphore Niépce, the earliest surviving photograph of a real scene, 1826-7.

tained (positive) [Schaaf, 2000]. This makes it possible to create a large number of duplicates of the positive print. Only a few years later, in 1839, Louis Daguerre introduced worlwide the *daguerreotype*, a process that using a *camera obscura* produced photographs in a reduced period of time (minutes) [Curley, 2010].

From the 19th century onwards the device began to be called a photographic camera, corrections were added to the lenses, it was possible to vary the aperture and it began to use the photographic film, pioneered by George Eastman with his *Kodak* camera in 1888 [Rines, 1920]. Great developments on the analog device were carried out at this time. It was in 1975 when Steven Sasson, working at Eastman Kodak Company, invented and built a self-contained electronic camera [Präkel, 2010]. And later, in October 2000, J-Phone introduced the world's first digital camera phone, the J-SH04, in Japan [Okabe, 2004].

Of course, a great development in devices has occurred and continues nowadays, but the purpose of this section is not to achieve a detailed search, but to introduce superficially some of the most significant advances in the history of photography.


(a) Paper negative, irregularly trimmed. `https://www.metmuseum.org/toah/works-of-art/1997.382.1/`

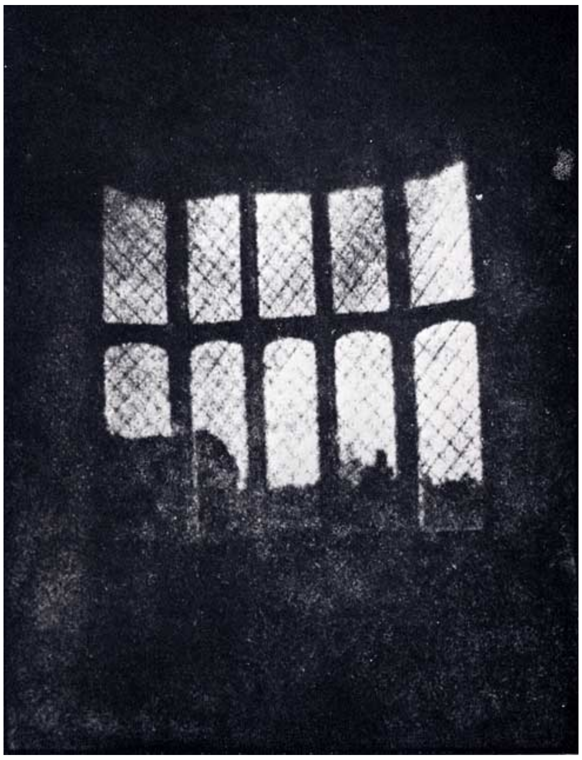

(b) Positive image. `http://www.edinphoto.org.uk/1_P1_photographers_talbot_smm_latticed_window.htm`

Figure 2.10: Window in the South Gallery of Lacock Abbey made from the oldest photographic negative in existence, William Henry Fox Talbot, 1835.

2.3.1 Projective Geometry and Camera Model

As we have already seen, throughout history great advances have been made in the technological development of image capture devices. However, although there is a variety of mathematical models that describe the process of image generation, the most commonly used model is the *pinhole camera*. It is a basic model of projection of straight rays where the effect of the lens is not taken into account. However, it is common to previously *rectify* the image before applying the pinhole model to take into account the distortion effects applied by the lens. Below, the projective geometry is presented and the camera model used in this thesis is shown.

In the computer vision environment it is common to use homogeneous coordinates, where a point in the image space (2D) with coordinates $(x, y) \in \mathbb{R}^2$ is represented as (kx, ky, k). Thus, the basic way to represent that point would be $(x, y, 1)$, although it is equivalent, for example, to $(3x, 3y, 3)$. The same happens with a point in the three-dimensional space, which is usually represented as $(X, Y, Z, 1)$. The use of homogeneous coordinates allows a much simpler handling of the projective geometry, where 3D points found in infinity are easily representable taking its fourth component to zero. The treatment and operations on points, lines and planes also becomes simpler with this homogeneous representation, where multiple operations can be carried out, carrying and bringing geometric elements from or to infinity. Although the use of infinity in geometry may sound bizarre, it is not so strange when dealing with projections (such as, for example, the conical projection). Parallel lines converge at a vanishing point, which we can even observe and locate in an image, although we know that they really cut only at infinity. Projective geometry allows us to describe all these phenomena which, although we are not used to working in a mathematical way, we are used to perceiving with our own eyes, where straight lines may appear curved due to our spherical visual system.

The pinhole camera model allows us to project a point from the 3D world to the 2D plane of the image on the camera. The mapping that projects these points $\Pi\colon \mathbb{R}^3 \mapsto \mathbb{R}^2$ is modeled by means of simple ray tracing and using the similarity of triangles the coordinates in the image plane can be obtained (see Fig. 2.11) as

$$m_{1_x} = f\frac{M_{1_X}}{M_{1_Z}}, \quad m_{1_y} = f\frac{M_{1_Y}}{M_{1_Z}}.$$

being f the focal distance, $(M_{1_X}, M_{1_Y}, M_{1_Z})$ the (X, Y, Z) components of point $\boldsymbol{M_1}$ and $(m_{1_x}, m_{1_y}, m_{1_s})$ the (x, y, s) components of point $\boldsymbol{m_1}$. The projecting beam perpendicular to the image plane is called the *principal axis*, it passes through the point $\boldsymbol{C}$ (*camera centre*) and intersects with the image plane at the point $\boldsymbol{c}$ (*principal point*). Assuming we

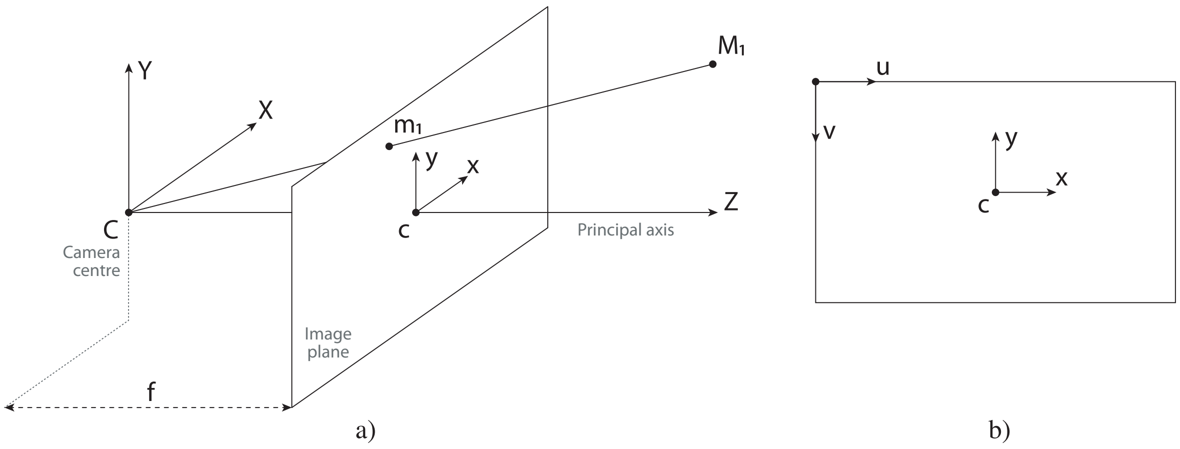


Figure 2.11: Scheme showing the projection of a point M to the image plane in the pinhole camera.

are using digital photography, the coordinates of the 2D image are usually measured in pixels, the smallest unit of measure of a digital image. It is usual in image processing to transfer and invert the origin of coordinates of the image so that it is placed in the upper left corner (Fig. 2.11.b), being more intuitive to treat images as pixel matrices. Thus, the general projection equation Π can be written in matrix form as $P = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$. The part $\boldsymbol{K}$ corresponds to the camera intrinsic matrix, which contains the focal distance (f), the pixel size (d_x,d_y) and the coordinates of the principal point (c_x,c_y), used to transform the coordinates to pixels with origin in the upper left margin. On the other hand, $[\boldsymbol{R}|\boldsymbol{t}]$ represents the camera extrinsic matrix. It transforms the original 3D scene points to the camera reference. The extrinsic matrix can be decomposed in an $SO(3)$ rotation matrix ($\boldsymbol{R}$) and translation vector ($\boldsymbol{t}$), and it is used to reference the points observed in the image with respect to a global coordinate system for all frames. The third component of $\boldsymbol{m_1}$ (m_{1_s}) refers to the scaling factor according to the homogeneous coordinate system. In its expanded view, the projection can be expressed as

$$\begin{bmatrix} m_{1_u} \\ m_{1_v} \\ m_{1_s} \end{bmatrix} = \begin{bmatrix} {}^{f}/_{d_x} & 0 & c_x & 0 \\ 0 & {}^{f}/_{d_y} & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} M_{1_x} \\ M_{1_y} \\ M_{1_z} \\ 1 \end{bmatrix}.$$

Intrinsic parameters are unchanged, generally[11], for each camera and must be determined by an estimation process known as *calibration*. By photographing an image pattern and knowing its real dimensions, it is possible to estimate the intrinsic parameters of the camera as well as the distortion produced by the lens. For more information about this calibration process, please consult the book [Hartley and Zisserman, 2003]. There are different models to estimate the deformations produced by the lens, but for conventional cameras such as those used in this thesis, a simple correction model taking into account both radial and tangential distortions can be used [Bradski and Kaehler, 2008]. The radial correction used is governed by the expansion

$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6),$$
$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6),$$

being (k_1, k_2, k_3) the parameters to estimate and r the radial coordinate with origin in the camera centre. The tangential distortion is corrected with the equation

$$x_{\text{corrected}} = x + [2p_1 xy + p_2(r^2 + 2x^2)],$$
$$y_{\text{corrected}} = y + [2p_2 xy + p_1(r^2 + 2y^2)].$$

being (p_1, p_2) the tangential parameters. This set of distortions usually have their origin in the manufacturing process of the lenses, being more profitable to correct these effects in a computational way than trying to create a perfect lens. The sum of both corrections makes it possible to recover the original image without geometric aberrations.

2.3.2 Multi-view reconstruction

A standard monocular camera (like the one we can have in our *smartphone*) requires, in general[12], the application of the triangulation process to estimate the 3D position of

[11] Some intrinsic parameters depend on the variation of camera settings, such as the zoom of the lens, which can modify the focal distance.

[12] 3D reconstructions are also possible from one image if some information about the scene is known (*Single view reconstruction* [Hartley and Zisserman, 2003]), and even some systems based on artificial intelligence are able to estimate the depth of the scene from a single 2D image [Liu et al., 2015, Facil et al., 2019], but we do not use this type of systems in our work since in this case we bet more on a system based on measurements instead of learning to estimate the 3D position of objects in space.

points in space.

Due to the type of projection of the process, the inverse function $\Pi^{-1}: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ that relates the points in the image to the objects in the real scene is not so easy to get. The projection of the camera can be seen as the collapse of the three-dimensional world on a two-dimensional plane, where that collapsed dimension cannot be recovered again. Therefore, more than one 2D image taken from different positions of the same objects are required (and with certain restrictions[13]) in order to accurately estimate the 3D position of objects in space using triangulation techniques, also known as *Structure from Motion* [Ullman, 1979] (Fig. 2.12).

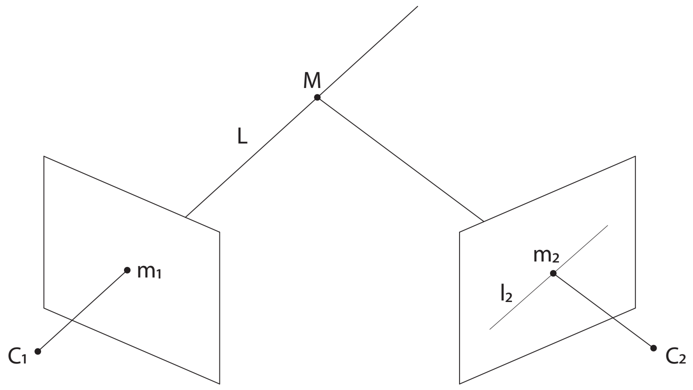


Figure 2.12: Monocular triangulation method to estimate the 3D position of point M, observed from two images with different camera locations. m_1 and m_2 are the image projections of world point M, L is the projecting ray from C_1 to M and l_2 is the projection of this ray on the second image.

Although it may seem that the positions of both cameras in Fig. 2.12 are not related, they really are. Epipolar geometry establishes the set of relationships between cameras that observe the same points in 3D space. The projecting ray coming from camera 1 ($\boldsymbol{L}$) is seen from camera 2 ($\boldsymbol{l_2}$) and is called *epipolar line*. The optical centre of camera 1 ($\boldsymbol{C_1}$) observed from camera 2 is called *epipole* or *epipolar point*. The same geometric properties are observed inversely, from camera 1. These geometric relations are contained in the *fundamental matrix* $\boldsymbol{F}$, that allows to transform a point in image 1 to its counterpart point in image 2, according to the relationship

$$\boldsymbol{m}_2^T \, \boldsymbol{F} \, \boldsymbol{m}_1 = 0. \tag{2.12}$$

Points $(\boldsymbol{m}_1, \boldsymbol{m}_2)$ are expressed in homogenenous coordinates and measured in the 2D image plane. To estimate the fundamental matrix it is necessary to have at least 7 correspondences (points observed in both images and labeled as the same point) and they do not have to be part of the well-defined critical configurations[14]. With these correspondences and the algorithm of the 7 points [Hartley and Zisserman, 2003] it is possible to

[13]Some problems may appear with the known *degenerated configurations* [Hartley and Zisserman, 2003].
[14]Some point configurations do not allow a correct estimation of the fundamental matrix (e.g., points that are linearly dependent, as for example if the 7 points are collinear, located in a straight line).

estimate the matrix $\boldsymbol{F}$ (3×3) that relates the relative positions between the two cameras and the 3D space position of each point, up to a scale factor.

In the case of three cameras, the fundamental matrix is not used anymore, but the trifocal tensor $(3 \times 3 \times 3)$ is used. It stores the relative projective geometry of the three cameras. Again, at least 7 corresponding points are needed to solve the problem linearly, although by other non-linear resolution methods they can be reduced to 6 correspondences. The advantage is that correspondences between points and lines can be used, and the reconstruction acquires more *stability* avoiding unstable configurations, which are always undesirable.

It is possible to switch to a quadrifocal tensor for the use of 4 cameras, although the level of complexity begins to be elevated. Again, the quadrifocal tensor obeys internal constraints, so not all components are independent. It is, as in the previous cases, a non-iterative method, but its implementation is becoming impractical.

Let's imagine that we want to place more camera positions (a video sequence where the camera moves may involve 30 different positions per second). The complexity that last methods require can be impracticable, and although some methods propose to solve by using 2 by 2 (or 3 by 3) camera positions, it seems that the algorithm that best fits is the *bundle adjustment* (BA). It is an iterative generic method that can be applied to any type of scene geometry, trying to minimize the reprojection error between observed objects and camera projected objects (observed in more than one image), allowing the estimation of the 3D points and all camera locations at the same time [Triggs et al., 1999]. The drawback is that no solution is guaranteed in the initialization step, so other methods are often used to initialize the system, and then bundle adjustment is applied. In short, to estimate the position in space of a point $\boldsymbol{M}$ we apply the triangulation process by adjusting the projective rays, which is usually done by minimizing the reprojection error $d(\boldsymbol{m}, \hat{\boldsymbol{m}})$ on the 2D image having a set of corresponding points in (at least) two images (although, of course, other techniques exist [Lee and Civera, 2019]). Being d the euclidean distance between the pixel coordinates $\boldsymbol{m}$ of a point observed in the image and the pixel coordinates of the point reprojected on the same image $\hat{\boldsymbol{m}}$ (from the 3D point $\boldsymbol{M}$ and the camera matrix $\boldsymbol{P}$, parameters that are also optimized). Using, for simplicity, the minimum number of two camera positions, the minimization of the following reprojection functional is accomplished

$$\sum_{j=1}^{N} d(\boldsymbol{m}_1^j, \hat{\boldsymbol{m}}_1^j)^2 + d(\boldsymbol{m}_2^j, \hat{\boldsymbol{m}}_2^j)^2$$

over the pre-estimated fundamental matrix $\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{m}}_i^j$, $j = 1, \ldots, N$, with N the number of identified (and matched) points in image i. This distance functional is minimized using the Levenberg-Marquardt [Press et al., 1988] algorithm over $3N$ (3D points) + 12 (2 camera source locations) variables. The problem can easily be extrapolated to an M number of images (camera positions).

The position and orientation (pose) of the camera and 3D points are estimated by using known fixed points (that can be fiducial markers or image features, depending on the particular application) along with consensus techniques, such as RANSAC [Fischler and Bolles, 1981], to identify the subset of fixed points in the scene that best fit the estimated transformations. The success of RANSAC techniques depends on the ratio of inliers (fixed points) to outliers (deformable points).

2.3.3 Structure from Motion

As we have seen in section 2.2.1, the estimation process of the rigid 3D structures from planar images using different camera view points is also known as Structure-from-Motion (SfM). The standard approach to compute rigid SfM is based in bundle adjustment, being normally a post-processing technique giving (in general words) more accurate results as much information is extracted from images. But some applications, such as automatic navigation, need an on-line mapping in real time. This means we need to compute SfM method in real time to be able to navigate in an unkown world, receiving the name of *Simultaneous Localization And Mapping* (SLAM).

Many works have been developed in the SLAM environment, allowing to use different sensors (monocular [Davison et al., 2007], stereo [Paz et al., 2008], RGB-D [Gutierrez-Gomez et al., 2016] or laser technology [Cole and Newman, 2006], among others). We can also make a division in visual SLAM between dense [Engel et al., 2014] and feature-based [Mur-Artal et al., 2015]. Dense approaches (also called direct) use the whole image pixels minimizing the photometric error, while feature-based systems extract features from the images and use them as keypoints to apply bundle adjustment techniques. Another classification can be maid between filtered [Smith et al., 1990] or keyframe-based [Klein and Murray, 2007] techniques, resulting the first idea in a filtering technique based in probability functions and the second idea in a sparse implementation requiring graph optimizations. The work carried out for the last decades has allowed these techniques to move from only research to the development stage, appearing many companies working in this field and offering tools based in SLAM techniques.

Another classification, with special interest for us, can be maid regarding the movement of the scene objects. Standard implementations of SfM (and SLAM) suppose a rigid world, but sometimes this assumption cannot be maid. The term non-rigid structure from motion (NRSfM) appeared to take into account this kind of movements, and it assumes the camera can move inside the world but also scene objects can move (or deform), as we can see in the right side of Fig. 2.13. This is an ill-posed problem as we are not able to build the 3D object from 2D points triangulation. In fact, a different configuration (non-rigid object deformation) can be observed in each image, impeding the use of standard triangulation techniques. It also means that different configurations may result in a very similar image projection, making difficult a direct estimation of the objects geometry.

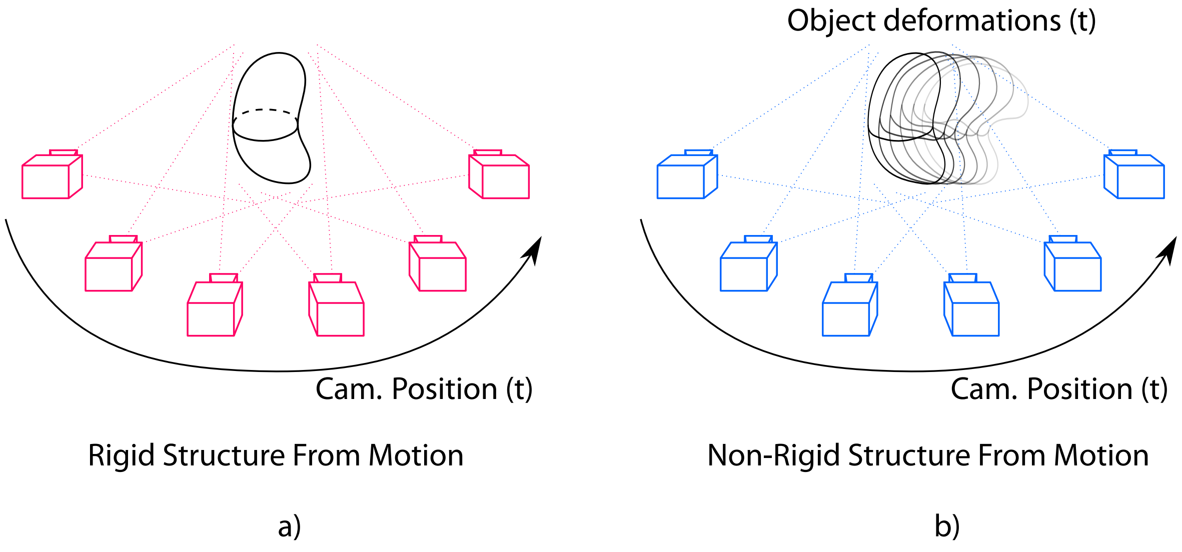


Figure 2.13: Rigid and Deformable SfM. a) Rigid SfM allows a reconstruction of the world from different views of the rigid objects. The camera is taking images while changing the pose. b) Non-rigid SfM implies that both camera and scene may take different positions and shapes (time-dependent).

During last years, many works regarding NRSfM have been developed trying to solve this problem. Some of them used factorization approaches to obtain low-rank representations of the objects from image streams [Bregler et al., 2000], where the 3D shape in each frame is a linear combination of a set of basis shapes (previously applied to rigid shapes by [Tomasi and Kanade, 1992]). Also the non-rigid movements have been computed as a union of subspaces to model complex motions with clustering local subspaces [Zhu et al., 2014]. Some other works computed the trajectory space by a linear combination of basis trajectories instead of basis shapes using *generic* bases [Akhter et al., 2009]. Non-linear dimensionality reduction methods have also been used to model the 3D shape as a non-linear combination of basis shapes, using kernel functions [Gotardo and Martinez, 2011], allowing a reduced number of bases if the problem is non-linear and the kernel adjusts better that non-linearity than the linear adjustment. Also bayesian implementations have been used [Agudo and Moreno-Noguer, 2018].

Priors were introduced to narrow down the search for an optimal solution [Torresani et al., 2008], but recently some works allow a prior less estimation of the structure from motion assuming compressible 3D objects as a block sparse dictionary learning problem [Kong and Lucey, 2016], and also applying dense variational reconstructions requiring GPU acceleration [Garg et al., 2013].

Recent works are starting to reconstruct the shape of deformable objects using templates previously created [Bartoli et al., 2015], obtaining accurate results, but requiring rare 2D parameterizations as they work only with the external surface. It means some energy constraints need to be imposed to assure boundary continuity and elastic deformations.

Also simple physic-based implementations have tried to solve the motion of the external surface using physical priors and Kalman filter formulations [Agudo et al., 2016].

2.4 Chapter Conclusions

In this chapter we have tried to describe some of the basic concepts included in this thesis, as well as the state of the art where it is encompassed. It is of vital importance to highlight the relevance of spatial localization in mixed reality applications. We need a system capable of *understand* the space around the user, which as we know is dynamic, so the system must be able to navigate around the scene as well as to describe the deformations of those non-static objects. It would be nice to have a semantic knowledge of objects, as some deep learning applications are proposing nowadays [Guerrero-Viu et al., 2019], but in this work we want to go a little further. We are not trying to identify and locate all static objects of a scene, we are trying to create a system that is capable of simulating interactions with some objects. By interactions we mean tracking of deformations of a given object, interaction with virtual objects or addition of external information such as the aerodynamic behavior of a vehicle. We therefore need to know:

- The static environment containing real-world objects.
- The position of the camera inside the scene at any moment.
- Localization, movements and deformations of objects that can be deformed.

It is in this last part where we introduce data science in combination with realistic physical simulations to describe the behavior of deformable objects.

After presenting the basis and some theoretical concepts on which this thesis is based, we can now move on to the chapters that describe the work developed in greater depth.

Chapter 3

Non-linear Model Order Reduction

The need for reduced models to work in real time has already been demonstrated in the previous chapter. However, inside MOR methods there are multiple variants and families. This chapter focuses on the *a priori* methods, more specifically on a local variant of the Proper Generalized Decomposition (PGD) method. Although variable separation introduces an intrinsic non-linearity into the resolution of the differential equation governing the problem, the PGD method itself can be considered as a linear projection, since the solution is projected onto separate vector spaces by linear combinations. In other words, we are projecting parts of the solution onto spaces of smaller dimensionality, in the style of the PCA, although without fulfilling orthogonality properties. However, these multiple projections are simply assembled by simple algebraic operations such as sums and multiplications, which makes the method robust and fast in the on-line phase.

In cases where the variation of the solution is complex, or the solution is not easily separable, we may need too many terms in order to approximate the solution. Non-linear approaches can be used in these cases to force higher compressions that, although use non-linear functions, translate into lower number of modes (reduction of the storage cost).

The structure of the chapter has been divided in four sections. There is an introduction to show the problems that standard PGD method has in non-separable problems. Section 3.2 introduces some well-known cases where PGD encounters difficulties in the resolution and shows an analysis of the proposed local PGD (hereafter, ℓ-PGD) method. Three different approaches to ℓ-PGD are included, where two of them take advantage of the known information at each moment. Section 3.3 introduces three examples to validate the method: a first example of 2D linear elasticity with a moving load, a second example solving the *Idelsohn's Benchmark*, a transient heat equation with moving source [Neron and Ladeveze, 2013], and a third example solving the parametric Idelsohn's Benchmark, adding the conductivity as a parameter in the PGD formulation. Finally an analysis with the conclusions of the chapter is covered in Section 3.4.

Some content of this chapter has previously been published in the paper:

- Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, vol. 112, no. 12, pp. 1715-1732.

3.1 Introduction

PGD has demonstrated to be specially efficient in the solution of high-dimensional parametric problems [Chinesta et al., 2011]. But it has also some limitations that we have to keep in mind:

- On one hand, some problems have, by definition, a solution structure which is intrinsically non-separable. According to Fourier's decomposition [Dirichlet, 1829] "*any periodic function can be decomposed in a finite (or infinite) sum of a set of simple functions*". We can extrapolate this approximation to a non-periodic function, and also to the PGD method with negligible loss of accuracy. So we can approximate a non-separable solution as a sum of the product of separable functions, although the number of required terms can be really high.

- On the other hand, PGD is optimal for solving elliptic differential equations up to two dimensions. In that aspect, PGD demonstrated to be equivalent to POD [Falcó et al., 2013]. For parabolic and hyperbolic differential equations, or elliptic equations with more than 2 parameters, PGD method is in general able to find a solution but there is not evidence enough to assure that the solution is optimal. But this is not only a PGD problem. Methods based upon POD are in a similar situation, since high order singular value decomposition (HOSVD [De Lathauwer et al., 2000]), or similar methods like the *Tucker3* decomposition [Tucker, 1966], are not optimal for 3^{rd} or more order tensors [Nouy, 2010].

The origin of the poor separability of the solution relies in the manifold structure of the solution. Many MOR methods look for a suitable basis set in order to project the solution. In general, this basis, even if it is globally optimal, can lead to very poor results if the manifold structure of the solution is very intricate. To avoid these problems, the idea of local basis arises naturally. Initial works were done in the framework of POD-based methods [Shepard, 1962a, Shepard, 1962b, Kruskal, 1964a, Kruskal, 1964b, Hernández et al., 2014]. After that, PCA was also used in its local basis approximation with data analysis [Fukunaga and Olsen, 1971], pattern recognition [Hinton et al., 1995] and image interpolation [Bregler and Omohundro, 1995] purposes. Kambahtla *et al.* worked also with local PCA with clustering techniques to determine the local regions, and compared the results with neural network implementations of nonlinear PCA [Kambhatla and Leen, 1997].

Other works are based in hyper reduction methods [Ryckelynck, 2005] on local reduced order bases [Amsallem et al., 2012]. They make use of an *offline-online* scheme to precompute the local bases on the *offline* stage using POD and obtaining the particular solution in the *online* stage by exploiting selectively the local bases. Other model order reduction methods project over a restricted subset of the spatial domains [Astrid et al., 2008] or use weighted functions to determine the most relevant *snapshots* [Peng and Mohseni, 2016]. This last work also established an interesting classification of the POD method into global POD, local POD and adaptive POD. Global POD projects the original system onto a subspace considering all the snapshots of the domain, local POD projects a local region of the system onto a subspace considering only the snapshots in the subdomain, and adaptive POD uses the global domain to create adaptive reduced bases making use of interpolation methods [Amsallem and Farhat, 2008]. Of course, in addition to POD, this classification can be used with any other MOR techniques.

The difficulty of using PGD in a local context relies precisely in its *a priori* character, i.e., in the fact that we have absolutely no information on the manifold structure of the solution. In this chapter we present a method in which we adaptively unveil this manifold structure on the fly, without the need for any sampling of the solution.

3.2 Local PGD (ℓ-PGD)

As we have seen, PGD assumes a separated variable structure of the solution that may confront with some problems called *non-separable*. Some physical phenomena have a solution structure that does not fit with a separate variable expression, and PGD may need too many modes to obtain an accurate enough solution with acceptable error rates.

3.2.1 Difficulties in the reduction process

The sources of non-separability of the solution may come from different sides. Following [Quarteroni et al., 2015], we could mention

- The global dimension of the problem. In a given space-time (or parameter) point of the solution manifold we can approximate the solution by the tangent hyperplane, which is in turn spanned by a basis vector of dimension equal to that of the space plus that of the parameter vector and time. However, to approximate well *all* the solution manifold, the number of vectors in the basis may be higher.
- Related to last item, the separability also depends on the approximability of the solution manifold by n-dimensional subspaces, the so-called Kolmogorov n-width. It is specially high in the case of the already mentioned Idelsohn's problem (see Sec. 3.3.2).
- Differentiability and regularity of the solution map.
- Parametric complexity, i.e., to what extent the solution is affine in the parametric space, thus admitting a separated expression.

Several methods have been tried so far. One solution would be to employ space-time (thus, non-separated) basis functions. An example is the work of Gerbeau [Gerbeau and Lombardi, 2012], that employed the concept of Lax Pairs [Lax, 1968] to obtain a reduced model with a special application to wave and soliton problems.

In general, we have different methods to minimize the effect of the poor regularity of the solution map or the parametric complexity. However, some of them are focused in the two first sources of poor separability mentioned before, namely the global dimension of the problem and the high Kolmogorov n-width. If a local implementation where the local dimension is much lower than the global one, one could reasonably expect that the employ of local PGD approximations could improve the results.

In what follows we discuss three different strategies for the construction of local PGD (ℓ-PGD) approximations to the problem at hand.

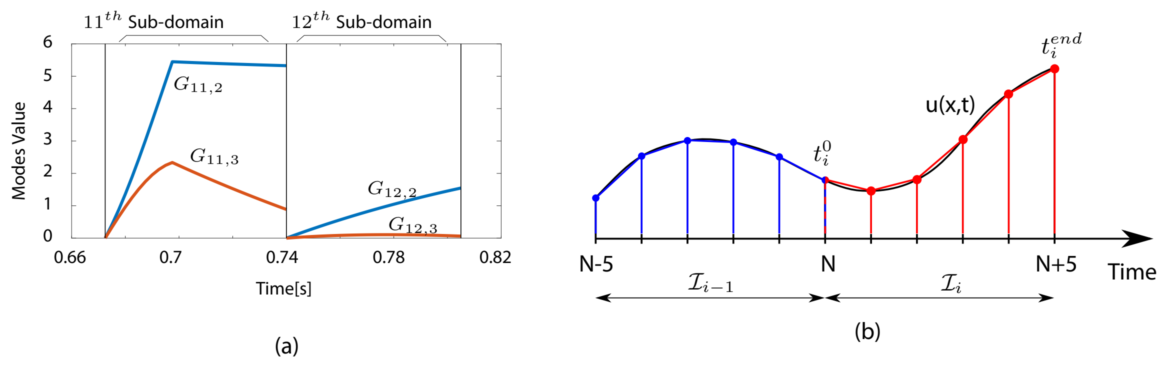


Figure 3.1: (a) Time modes $G_i(t)$ may be discontinuous after juxtaposition along time. However, the resulting solution, (b) is made continuous by only imposing appropriate initial conditions at each sub-domain boundary.

3.2.2 Constant local partitions (Cℓ-PGD)

Consider, to begin with, a space-time problem without any parametric dependence. We can apply the most naive implementation, i.e., by partitioning uniformly the time domain only. Each resulting two-dimensional sub-region is therefore formed by the same number of nodes both in spatial and temporal axes. In this way, we obtain a local basis for every reduced sub-domain. A solution in separate variables is obtained, where the spatial modes are supported in the entire domain Ω and time modes, on the contrary, span only a local sub-domain $\mathcal{I}_i := [t_i^0,\, t_i^{\text{end}}]$.

No special procedure is needed to guarantee the continuity of the solution, even if time modes could result discontinuous in boundaries (see Fig. 3.1). The only requirement is to impose the final values at interval $\mathcal{I}_i$ as initial conditions of the $\mathcal{I}_{i+1}$ interval. This method still allows for parallel implementations of ℓ-PGD although non-homogeneous boundary and initial conditions can be imposed, as it is implemented in [Gonzalez et al., 2010].

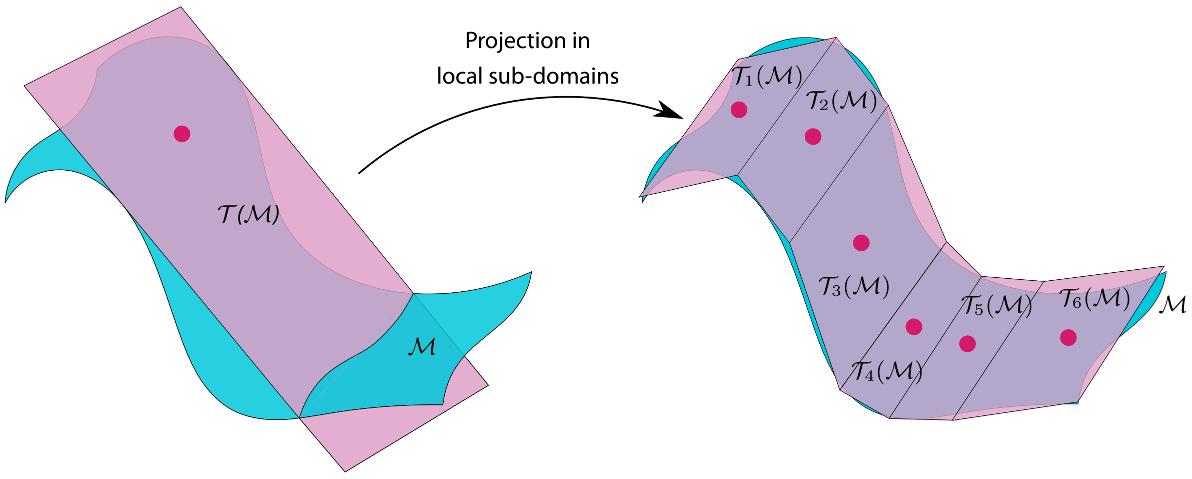

Projection in
local sub-domains
$\mathcal{T}(\mathcal{M})$
$\mathcal{M}$
$\mathcal{T}_1(\mathcal{M})$
$\mathcal{T}_2(\mathcal{M})$
$\mathcal{T}_3(\mathcal{M})$
$\mathcal{T}_4(\mathcal{M})$
$\mathcal{T}_5(\mathcal{M})$
$\mathcal{T}_6(\mathcal{M})$
$\mathcal{M}$

Figure 3.2: Manifold $\mathcal{M}$ where the solution lies and tangent hyper-planes $\mathcal{T}_i(\mathcal{M})$ corresponding to global (left) and local (right) basis approximations. Red circles indicate the tangent points.

The partition of the domain into constant size sub-regions allows PGD to capture the behavior of the solution in a faster and accurate way in some problems —as can be noticed in Section 3.3—, but we are not fully exploiting the potential of ℓ-PGD. The number of necessary modes may be highly variable for each subdomain. So we could perform a more efficient domain partitioning by employing variable subdomain sizes. Solution manifolds with highly intricate geometries can be described in a better way by using local domains with varying sizes. For example, solution regions with small variation in the original manifold can be approximated with bigger sub-domains ($\mathcal{T}_3(\mathcal{M})$ in Fig. 3.2) than regions with higher variation in the original manifold that may need a higher number of modes ($\mathcal{T}_4(\mathcal{M})$ and $\mathcal{T}_5(\mathcal{M})$ in Fig. 3.2).

In order to overcome the already mentioned limitations of constant partitioning, we propose the use of two different alternatives producing an adaptive partitioning of the domain in a smart way. The first one is an iterative method similar to the one proposed in [Dihlmann et al., 2011] for time domain partitioning. This method computes the size of the subdomain by imposing equal number of modes within each subdomain and a specific error tolerance. The second method uses Kernel Principal Component Analysis (k-PCA) projections [Schölkopf et al., 1997] of the already computed solution to unveil the manifold structure of the solution, and to suggest local divisions depending on the degree of complexity of the manifold.

3.2.3 Mode number optimization adaptive partitioning (MNOℓ-PGD)

Let us assume that the solution of a multidimensional problem remains on a manifold $\mathcal{M}$ as it is shown in Fig. 3.2. An adaptive partitioning allows us to better describe the manifold $\mathcal{M}$ by projecting the solution onto local bases $\mathcal{T}_i(\mathcal{M})$ that may be thought of as hyperplanes tangent to the manifold. Tangent hyperplanes may have non-constant size since the solution manifold may have an irregular variability.

For space-time problems, we will work on local regions formed by the full spatial domain and a target number of M local partitions in time. Local time-partitioning will be modified iteratively and PGD solved in each step. We start with a seed number p of time nodes, optimizing p to allow the number of local modes to be as close as possible to the sought amount M, which has been fixed previously, and an error tolerance E. Given a number of local modes N_i required to build the solution on a sub-domain i, and assuming a space and time discretization Δx and Δt both changeless, since we can only modify the number of local time nodes T_i, it is possible that the number of modes to obtain M is not achievable, $N_i \neq M$. We decided to choose the stopping number of modes N_i to be below and closest to the optimal, being $N_i \leq M \, \forall \, i \in [1, k]$ to ensure the maximum number of local modes is at most M,

$$\max_i N_i \leq M.$$

Algorithm 3.1: Pseudo-code to implement the adaptive partitioning MNOℓ-PGD.

```
AllowedModes = M;
while LastTimeInstantOfLastPartition < GlobalTimeEnd do
    InitialLocalPartitioning(i);
    while (LocalModesObtained(i) $\neq$ AllowedModes) & (LoopOut == 0) do
        [SpaceModes(i), TimeModes(i), LocalModesObtained(i)] =
        LocalPGDImplementation(i);
        if LocalModesObtained(i) == AllowedModes then
            break;
        else if LocalModesObtained(i) $\neq$ AllowedModes then
            UpdateLocalPartitioning(i);
        end
        if AllowedModesCanNeverBeReached then
            UpdateLocalPartitioning(i);
            LoopOut = 1;
        end
    end
    Solution(i) = SpaceModes(i) * TimeModes(i)';
    SetInitialConditions(i+1);
end
```

The implementation of the adaptive time-partitioning method with mode number optimization can be shown in Alg. 3.1. We have employed unitary increments and decrements of the number of local modes T_i in each algorithm iteration to assure the simplicity and robustness, but other techniques may be used to accelerate the convergence.

In the case of partitioning both the space and time axis, local domains depend on two parameters: the number of nodes in space Z_i and time N_i. In this case, the explained iterative method of Alg. 3.1 is no longer optimal. The difficulty increases if we also add parameters to the PGD separated formulation. This disadvantage led us to think in a third method able to extract information on the manifold structure of the solution, so as to allow us to perform adaptive partitions on the fly.

3.2.4 *kernel*-PCA-based ℓ-PGD ($k\ell$-PGD)

The application of PGD locally can be carried out in multiple ways. There could be eventually many other methods to find the number of local modes depending on the available information. But our goal remains the same, estimating the size of each local sub-domain to obtain a separated representation of the solution with a minimal amount of terms and, therefore, computational time and storage costs.

With this third method we intend to use manifold learning tools to unveil the manifold structure of the solution. The difficulty, however, remains to be the *a priori* character of PGD. Since PGD starts the approximation to the solution from scratch, without any sampling of the parametric space, we have no information on the manifold structure of the solution. Instead, we start by performing an arbitrary partitioning. We will employ k-PCA methods [Schölkopf et al., 1998] to check the appropriateness of this partitioning and, eventually, correct it iteratively.

k-PCA methods work, surprisingly, by projecting data onto a higher dimensional (eventually, infinite dimensional) space where data is exactly separable. In this space, standard PCA techniques can be applied. Then, data can be projected back onto the principal components and the first few, more relevant projections, retained. To obtain a better understanding of the manifold structure of the solution, we will measure the distance between different parameter-time-dependent solutions by resorting to the concept of Hausdorff distance [Hausdorf and Aumann, 1914].

Kernel Principal Component Analysis

As stated before, the idea behind k-PCA methods may seem surprising, but the elegance of the method comes from its conceptual simplicity: data that is not linearly separable in D dimensions, could be linearly separated if it is previously projected to a space in $Q > D$ dimensions [Schölkopf et al., 1998, Schölkopf et al., 1997]. Thus, k-PCA begins by projecting the data to an even higher dimensional space. To do it, a mapping is necessary:

$$\phi : \mathcal{M} \subset \mathbb{R}^D \to \mathbb{R}^Q, \ \boldsymbol{y} \to \boldsymbol{z} = \phi(\boldsymbol{y}),$$

where Q may be any dimension. One of the biggest advantages of this tool is that there is no need to explicitly determine the analytical expression of the mapping ϕ. This is known as the *kernel trick* in the machine learning community.

The resulting symmetric matrix $\mathbf{\Phi} = \boldsymbol{Z}^T \boldsymbol{Z}$ needs to be decomposed in eigenvalues and eigenvectors. In addition, the mapped data $\boldsymbol{z}_i$ involved in $\mathbf{\Phi}$ must be previously centered. The centering can be performed even if the mapping is unknown, in an implicit way by performing the so-called double centering.

Let us denote the mean of the j-th column of $\mathbf{\Phi}$ as $\mu_i(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$, and the mean of its i-th row as $\mu_j(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$. The mean of all entries of $\mathbf{\Phi}$ reads $\mu_{i,j}(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$. The double centering process gives

$$\boldsymbol{z}_i \cdot \boldsymbol{z}_j - \mu_i(\boldsymbol{z}_i \cdot \boldsymbol{z}_j) - \mu_j(\boldsymbol{z}_i \cdot \boldsymbol{z}_j) + \mu_{i,j}(\boldsymbol{z}_i \cdot \boldsymbol{z}_j).$$

The mentioned eigenvalue-eigenvector decomposition can be performed on the doubly centered matrix, leading to

$$\mathbf{\Phi} = \boldsymbol{U}\mathbf{\Lambda}\boldsymbol{U}^T.$$

It is important to realize that the mapping ϕ needs to be evaluated in the computation of the matrix $\mathbf{\Phi}$. If the number of dimensions of the target space, Q, is high (even infinite), the explicit computation of its entries may be prohibitive. The kernel trick allows us to overcome this difficulty by founding a kernel function κ that directly gives the value of the scalar product $\kappa(\boldsymbol{y}_i, \boldsymbol{y}_j) = \boldsymbol{z}_i \cdot \boldsymbol{z}_j$. This property follows from Mercer's theorem [Schölkopf et al., 1998, Schölkopf et al., 1997].

Many different kernels fulfilling Mercer's condition exist. Among them, the most popular ones are:

- Polynomial kernels: $\kappa(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} \cdot \boldsymbol{v} + 1)^p$, with p an arbitrary integer;
- Gaussian kernels: $\kappa(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2}\right)$ for a real σ;
- Sigmoid kernels: $\kappa(\boldsymbol{u}, \boldsymbol{v}) = \tanh(\boldsymbol{u} \cdot \boldsymbol{v} + b)$ for a real b.

In this work, and without loss of generality, we have employed Gaussian kernels. These provided excellent results in the determination of the manifold structure of the solution. Actually, what we pursue with the application of k-PCA is to determine the true *distance* between different parameter-specific solutions in the manifold. In other words, to unveil how intricate the manifold is.

Hausdorff distance

Our goal is to determine how far each space-time (or parameter) particularization of the solution is from its neighboring particularizations. In other words, to unveil the regularity of the solution manifold. To this end, once the k-PCA structure has been obtained, we determine the Hausdorff distance between particularized solution. The Hausdorff distance between two subsets X, Y of a metric space, is defined as

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\}.$$

The Hausdorff distance between particular realizations of the solution reveals the intrinsic variability of the solution. Once the distance between realizations has been determined, an algorithm must be designed to estimate the most appropriate size of the *patches* on which $k\ell$-PGD is applied. To this end, we have chosen the ε-neighborhood tool to cluster the information and extract the limits of local patches.

ε-neighborhood clustering

The neighborhood of a point p in a metric space $M(X,d)$ is a set V if there exists an open ball with center p and radius $r > 0$, such that $B_r(p)$ is contained in V

$$B_r(p) = B(p;r) = \{x \in X \mid d(x,p) < r\}.$$

We consider the ε-neighborhood ε-N_{D_i} of a point p as the set of all points in the real space $\mathbb{R}^D$ that are located at a distance less than ε from p measured in dimension D_i,

$$B(a;\varepsilon) = \{x \in \mathbb{R}^D : |x-p|_{D_i} < \varepsilon\}.$$

Algorithm 3.2: Pseudo-code to implement the *kernel*-PCA-based ℓ-PGD ($k\ell$-PGD).

```
VariablesInitialization;
ConstantLocalSolution = ComputeConstantLocalPGD;
while $j <$ NumberOf-kPCACurves do
    kPCACurves(j) = ComputekPCA(ConstantLocalSolution);
    $j \leftarrow j+1$;
end
ComputeHausdorffDistance(AllkPCACurves);
LocalkPCADivisions = Clustering$\varepsilon$-neighborhood;
while $i <$ NumberOfLocalkPCADivisions do
    [SpaceModes(i), TimeModes(i)] = ComputeLocalPGD;
    Solution(i) = SpaceModes(i) * TimeModes(i)';
    SetInitialConditions(i+1);
    $i \leftarrow i+1$;
end
```

In sum, an iterative algorithm has been designed that proceeds by

1. Choosing an arbitrary partition of the space-time(-parameter) space,

2. unveiling its manifold structure by applying k-PGD,
3. measuring the distance between particular time-parameter solutions by means of its Hausdorff distance, and
4. eventually modify the original partition and adapting it to the distance distribution.

In the next Section we present some examples of application of the three different methods and compare their relative performance.

3.3 Examples

We introduce here three benchmark examples. ℓ-PGD is applied and results compared with the classical, global PGD, and even the standard, also global, POD method.

The first example considers displacement of a bi-dimensional cantilever beam with a moving vertical load throughout its upper boundary. The second example considers the Idelsohn's benchmark. And, finally, the third example is the parametric version of the Idelsohn's benchmark, by adding the conductivity k as a parameter in the PGD formulation. Problems in up to three dimensions have been considered for visualization purposes, but the presented technique is completely general and can be applied to problems in arbitrary dimensions.

3.3.1 Cantilever beam with a moving load

This problem considers the horizontal and vertical displacement $\boldsymbol{u}(x, y)$ of any point of a bi-dimensional cantilever beam Ω, Fig. 3.3, with a moving vertical load F applied at a variable location on the upper boundary $\boldsymbol{s} \in \bar{\Gamma} \subset \partial\Omega$ where $\bar{\Gamma}$ represents the portion of the Neumann boundary Γ_t with non-vanishing conditions.

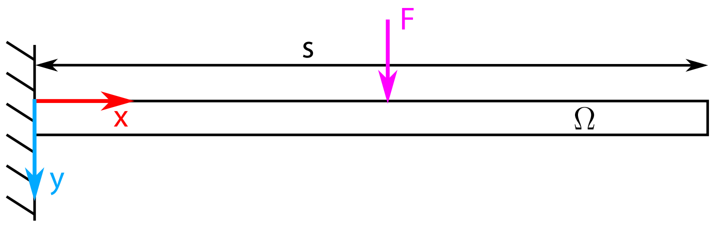


Figure 3.3: Cantilever beam with a moving vertical load F.

The governing equation for linear elasticity in its strong form is well known [Fish and Belytschko, 2007].

Find $\boldsymbol{u}(\boldsymbol{x})$ *such that*

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{\sigma}\, \boldsymbol{n} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_t, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_u. \end{cases}$$

Since we are interested in finding the parametric solution $\boldsymbol{u}(\boldsymbol{x}, s)$, i.e., for any possible load position along $\bar{\Gamma}$, we develop a doubly weak form by integrating not only in Ω, but also on $\bar{\Gamma}$ (details on the Matlab implementation of this problem can be found in [Cueto et al., 2016], see also [Chinesta et al., 2013a, Chinesta and Cueto, 2014]):

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla_s} \boldsymbol{u}^* : \boldsymbol{\sigma} \, d\Omega \, d\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma_t} \boldsymbol{u}^* \, \boldsymbol{t} \, d\Gamma_t \, d\bar{\Gamma},$$

where $\boldsymbol{u}^* \in H_0^1(\Omega)$ is an arbitrary test function and $\boldsymbol{\nabla_s} = \frac{1}{2}[\boldsymbol{\nabla} + (\boldsymbol{\nabla})^T]$ is the symmetric gradient operator. Following the standard PGD assumption on the separability of the solution, we postulate the solution $\boldsymbol{u}(\boldsymbol{x}, s)$ separated in the following way,

$$\boldsymbol{u}(\boldsymbol{x}, s) \approx \sum_{i=1}^{n} F_i(\boldsymbol{x}) G_i(s),$$

where n is the number of terms, $F_i(\boldsymbol{x})$ are the space modes in separate variables and $G_i(s)$ are the modes referred to the position of the load. We consider here s as a scalar, since the boundary of the two-dimensional beam is actually a one-dimensional region.

Standard PGD methods use a greedy algorithm to obtain the n functions $F_i(\boldsymbol{x})$ and $G_i(s)$. Assume that, at iteration p, we look for the $p+1$-th functions $R(\boldsymbol{x})$ and $S(s)$ that produce the enrichment of the solution:

$$u_{p+1}(\boldsymbol{x}, s) = u_p(\boldsymbol{x}, s) + R(\boldsymbol{x}) S(s).$$

Therefore, the test function is

$$u^*(\boldsymbol{x}, s) = R^*(\boldsymbol{x}) S(s) + R(\boldsymbol{x}) S^*(s).$$

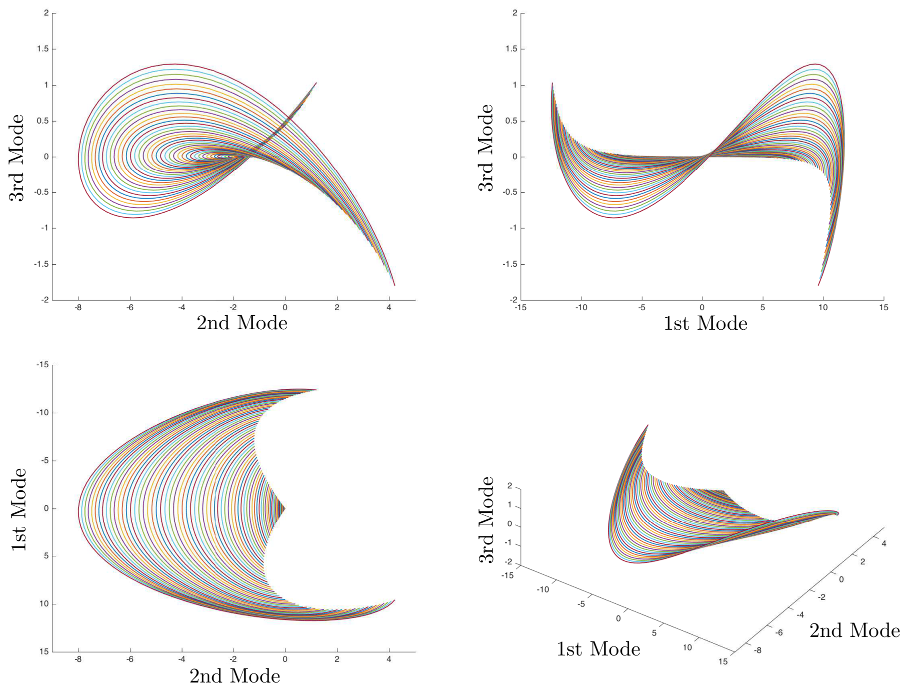
The load F has been designed as a unitary force acting along the vertical axis y at a variable position s along the upper boundary of the beam, see Fig. 3.3,

$$\boldsymbol{t} = F \, \delta(x - s) \boldsymbol{e}_y,$$

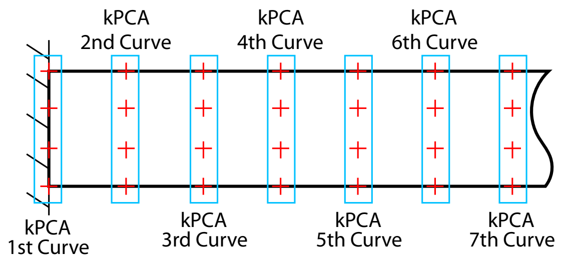
where δ represents the Dirac delta function. Function $\boldsymbol{t}$ needs to be expressed in separate form to comply with the PGD method, so

$$t_x = 0; \quad t_y \approx \sum_{j=1}^{m} h_j(\boldsymbol{x}) \, k_j(s),$$

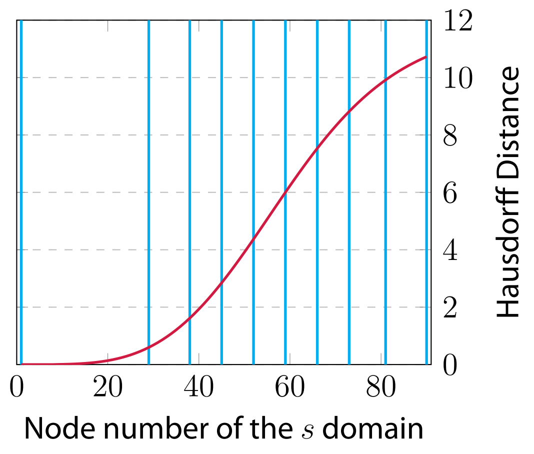
where m is the number of sums to approximate function t_y and $h_j(\boldsymbol{x})$, $k_j(s)$ are the functions in space and load position, respectively. We have employed a fixed point algorithm to solve the resulting non-linear problem of finding the pair functions $R(\boldsymbol{x}) S(s)$ at every enrichment iteration, but any other method (noteworthy, Newton) can be used. See [Cueto et al., 2016] for a more precise description of the matrix form of this problem and the implementation code in Matlab language.



(a)



(b)



(c)

Figure 3.4: a) Set of curves obtained after applying k-PCA. b) Sketch of the employed strategy. Red crosses (nodes) are grouped in columns and k-PCA is applied to the resulting vector of nodal displacements. Each column will represent a curve in a). c) Hausdorff distance between k-PCA curves and the origin of the coordinate system. Blue lines indicate the partitions in the s domain.

In the $C\ell$-PGD framework, uniformly-spaced local partitions have been made in the s domain maintaining the whole space $\boldsymbol{x}$ for every partition. Assuming M partitions in the parameter-space domain, the solution will be at each one of these partitions $u_i(\boldsymbol{x}, s) \in H_1(\Omega_x) \times H_1(\Omega_y) \times L_2(\bar{\Gamma}_i)\ \forall\ i \in [1, M]$, with $\Omega_x := [0,9][m]$, $\Omega_y := [0,1][m]$ and $\bar{\Gamma}_i := [s_i^0, s_i^{end}]$. We employed a mesh with size $\Delta_x = \Delta_y = \Delta_s = 0.1[m]$.

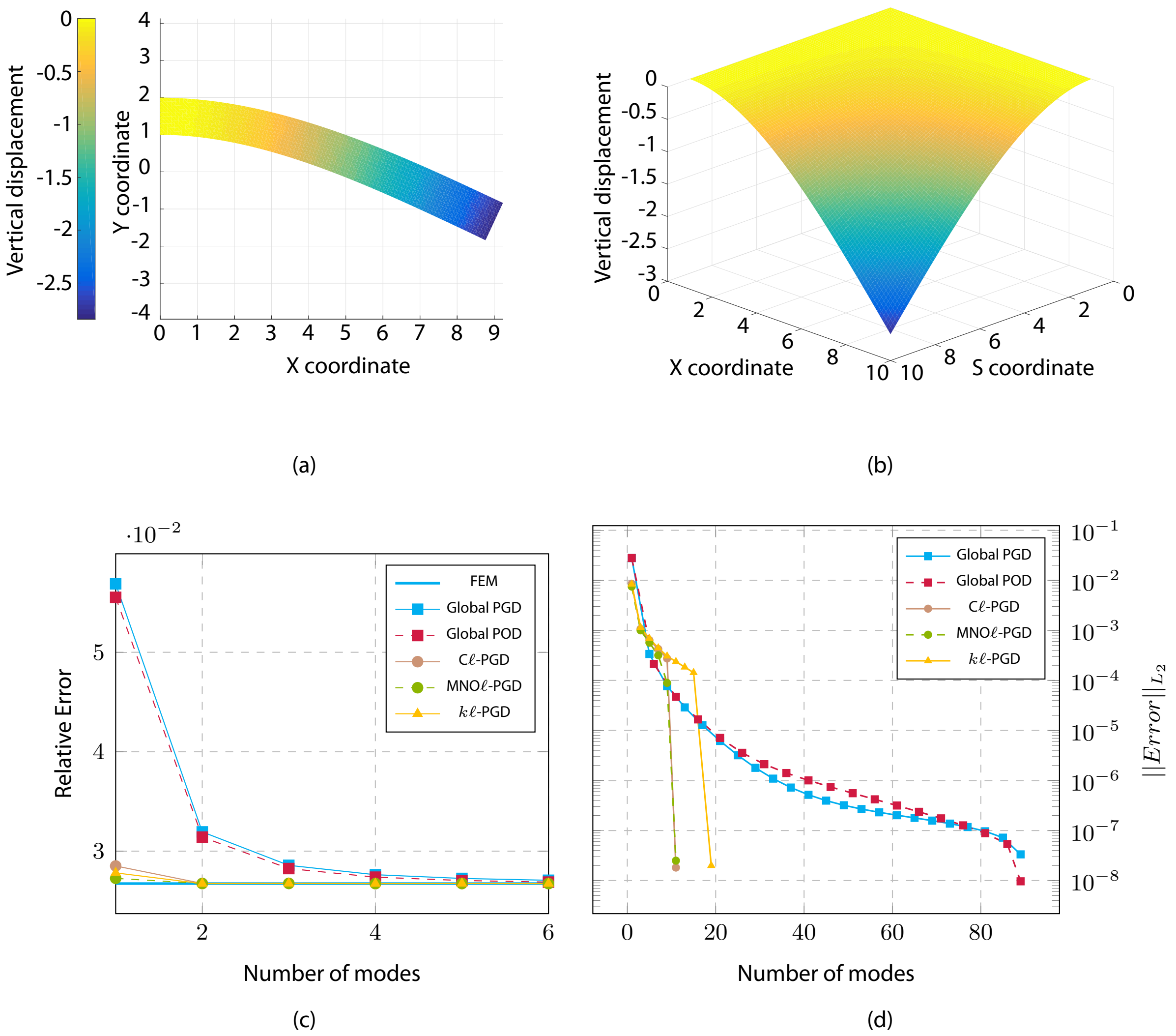


Figure 3.5: a) Vertical displacement of the cantilever beam solved with FEM with $F = -1[N], E = 1000[N/m^2], \nu = 0.3, s = 9[m]$ (this plot corresponds to the case of the load at the end tip). b) Vertical displacement of the cantilever beam solved for every position of the load in the s domain. c) Relative error of the vertical displacement of the end point (load at end tip) for some methods respect to the analytic Euler-Bernoulli-Navier solution. d) L_2-norm error of the vertical displacement for every position of the load in the s domain between each technique and the finite element solution with the same discretization, up to an error rate of $e = 10^{-8}$.

In the $k\ell$-PGD approach, on the contrary, we have applied k-PCA to vectors containing nodal values for the same s position (see Fig. 3.4b) using the vertical displacement as input. We obtained a set of curves (Fig. 3.4a) and making use of the Hausdorff distance concept we measured the distance between curves and the origin of the coordinate system so as to estimate the manifold structure of the problem (Fig. 3.4c). From here, it is possible to divide the s domain in partitions using clustering tools as described in Sec. 3.2.4.

The resulting vertical displacement field for the load in the end position of the beam is shown in Fig. 3.5a and Fig. 3.5b shows the surface obtained for the load applied in any available position.

To determine the accuracy of each proposed local implementation of the PGD, we compare our results with the well-known analytic solution of the 1D Euler-Bernoulli-Navier model of a cantilever beam [Truesdell, 1960]. Fig. 3.5c shows the relative error between the analytic solution of the beam with a load applied in the end and the next techniques: standard finite element method (FEM), PGD in its global implementation, POD also in its global version, $C\ell$-PGD with constant divisions of domain s, MNOℓ-PGD with optimal-modes divisions and $k\ell$-PGD with k-PCA informed divisions. Finally, Fig. 3.5d shows the L_2-norm error computed by comparing the result obtained with each technique against the finite element solution with the same discretization.

It is worth noting that both ℓ-PGD versions provide very similar results, showing that merely two load modes in nine divisions provide with the same error as the finite element solution for that particular location of the load. On the contrary, both global implementations of POD and PGD needed some five to six (global) modes to obtain the same accuracy.

3.3.2 Idelsohn's benchmark

An already classical example of non separable problems was established by Sergio Idelsohn in a joint Spanish-French workshop held at Jaca, Spain, in 2013. It is based in the transient diffusion equation in one dimension with a moving source. The governing equation and boundary conditions were described in a subsequent technical report by D. Neron and P. Ladevèze [Neron and Ladeveze, 2013]:

$$\begin{aligned} \rho c_p \partial_t u - \nabla \cdot k \nabla u &= f(x,t) \quad \text{in } \Omega \times \mathcal{I}, \\ u &= u_D \qquad \text{on } \Gamma_D \times \mathcal{I}, \\ u &= u_0 \qquad \text{on } \Omega \times \{0\}, \end{aligned} \tag{3.1}$$

with $\Omega := [0,\pi][m]$, $\mathcal{I} := [0,1][s]$, $u(0,t) = u(\pi,t) = u_D = 0$ and $u(x,0) = u_0 = 0$. The value of the thermal conductivity is $k = 0.05\left[\frac{W}{m\,C}\right]$, density is $\rho = 1\left[\frac{kg}{m^3}\right]$ and the specific heat capacity $c_p = 1\left[\frac{J}{kg\,C}\right]$. The source term is described by the equation:

$$f(x,t) = \begin{cases} 0 & t < t_i \text{ or } t > t_f \\ A\cos\left(\frac{\pi}{L}(x - x_0(t))\right) & t_i < t < t_f \text{ and } -\frac{L}{2} < x - x_o(t) < \frac{L}{2} \end{cases},$$

where $x_0(t) = x_i + \frac{x_f - x_i}{t_f - t_i}(t - t_i)$, $A = 100$, $L = 0.15$, $t_i = 0.2$, $t_f = 0.7$, $x_i = 2\pi/7$ and $x_f = 5\pi/7$. The source term $f(x,t)$ is the heat added per unit volume. Fig. 3.6 sketches the problem. In what follows, we consider also the possibility of a parametric problem, where the conductivity k acts itself as a parameter of the problem. This problem has also been analyzed, among others, by Allier *et al.* [Allier et al., 2015].

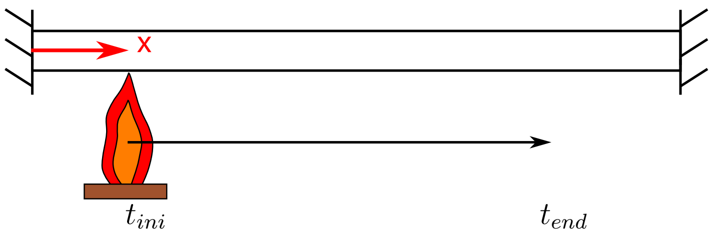


Figure 3.6: Idelsohn's benchmark in 1D.

Idelsohn's benchmark problem has been solved by applying POD and PGD in their global, classical version. These will serve as a reference measure of the degree of reduction attainable by ℓ-PGD.

The temperature distribution of the 1D solid can be plotted in a 3D surface where the first axis is the time coordinate, the second axis comprises physical coordinates of the solid (assumed to be one-dimensional) and the third, vertical, axis shows the temperature distribution in the space-time domain (Fig. 3.7a). For completeness, we also plot a 2D surface to better appreciate the sharp discontinuity in the solution (Fig. 3.7b).

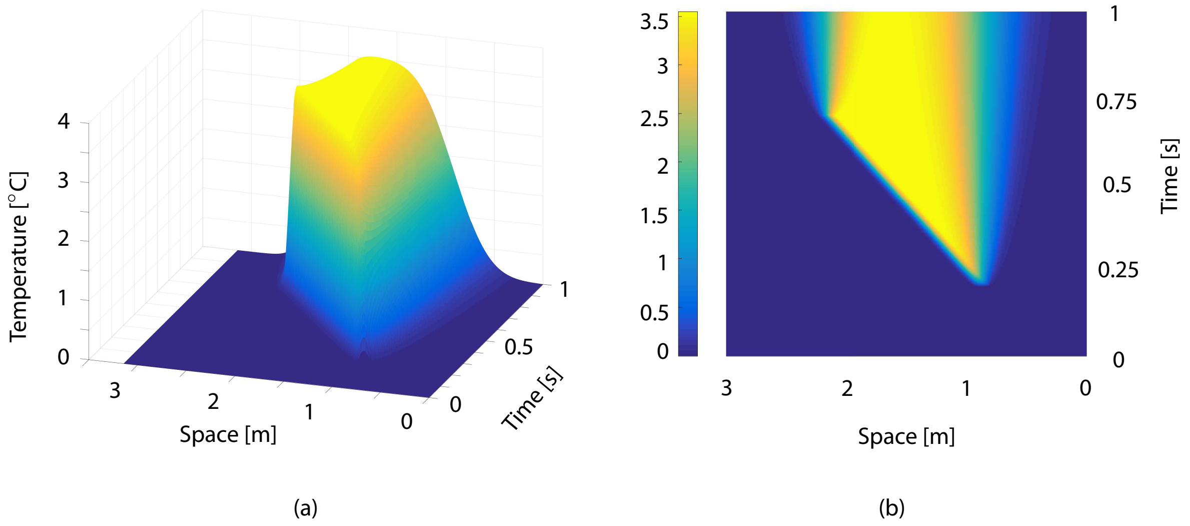


Figure 3.7: a) 3D surface of the solution of the heat equation with moving source with $x_f = 5\pi/7$[m], $k = 0.05$ [W/(m K)]. b) 2D view of the same problem to better appreciate the difficulty of separating a sharp discontinuity running across the diagonal of space-time.

The L_2-norm error is computed by comparing the result obtained against the reference finite element solution with the same discretization (i.e., the full-order model). The k-PCA projection of the solution onto the three first eigenvectors is shown in Fig. 3.8.

Fig. 3.9 shows the result of the L^2-norm error of the temperature distribution in the space-time surface (Fig. 3.7). The size of the mesh is N_x=301 space nodes and N_t=401 time nodes. It should be pointed out that local time modes can be arranged (and stored

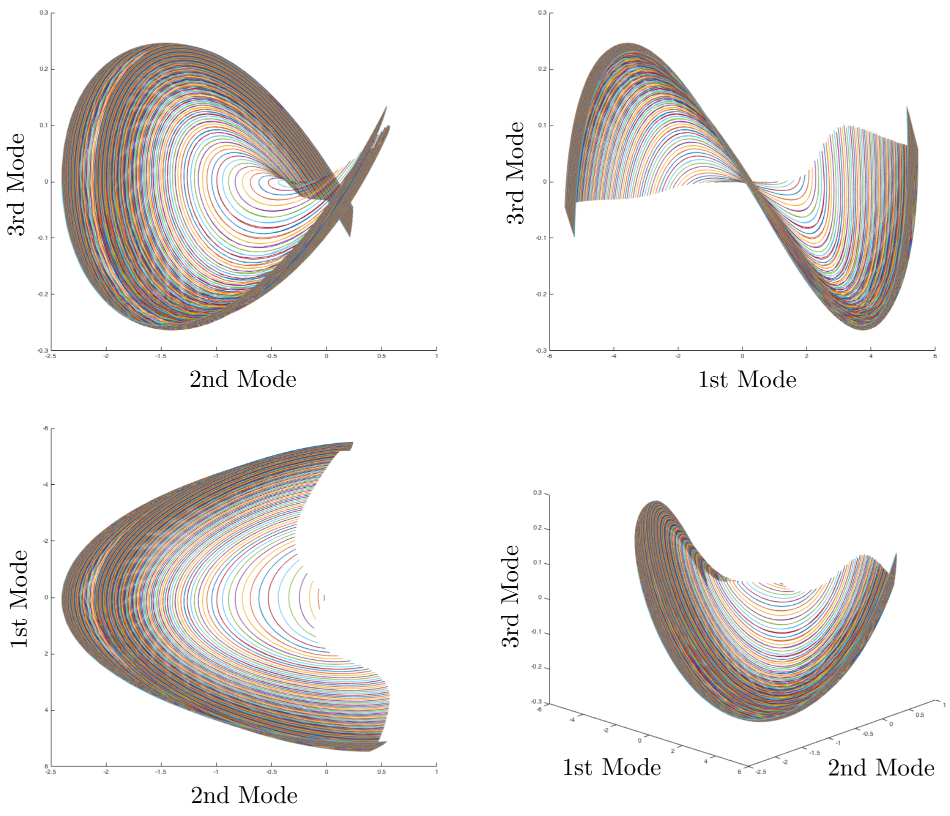


Figure 3.8: Example of application of k-PCA to the reference solution of the Idelsohn's benchmark. Each curve represents a space-time particularization of the solution to Eq. (3.1).

in memory) as global, though discontinuous, time modes. However, since we are taking the full spatial domain to form each sub-domain, spatial modes can not be organized in the same way. This requires more storage space when compared with the same number of global modes, but still allows us to obtain a much smaller computation time, as can be shown in Appendix 3.5.

Fig. 3.9 shows that ℓ-PGD provides with substantial savings in the number of necessary modes for a prescribed error in the approximation. It is also important, however, to determine the true savings in terms of stored memory or how the number of modes is distributed for each sub-domain. A constant number of modes in all sub-domains denotes that we are using the memory in the best way, and that the size of the sub-domains is balanced in terms of variability of the manifold structure of the solution. Fig. 3.10 shows the number of modes per sub-domain for the three ℓ-PGD methods with a value of x_f=$2\pi/7$ (Fig. 3.10a) and x_f=$5\pi/7$ (Fig. 3.10b). The Hausdorff curve that measures the distance between every k-PCA curve and the origin, used with $k\ell$-PGD, is shown in Fig. 3.11. In this figure we also plot the divisions employed to form each sub-domain in the time axis.

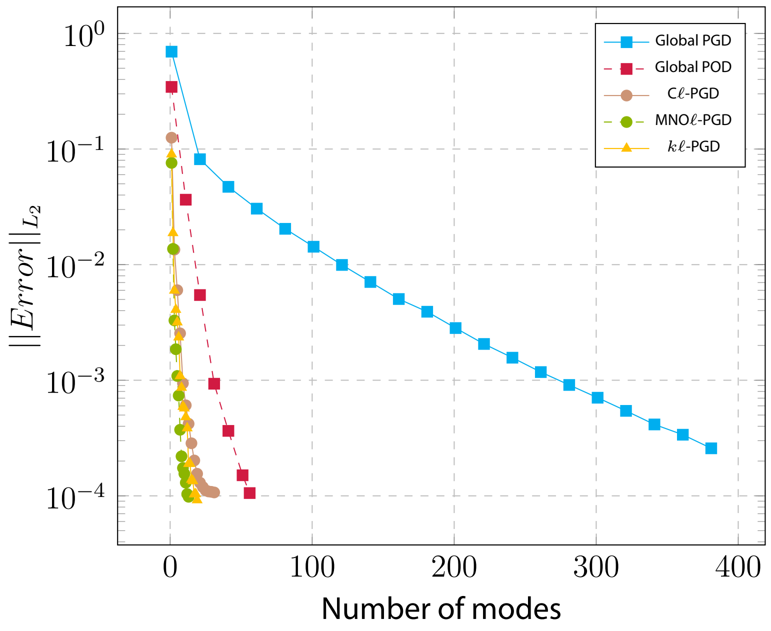


Figure 3.9: L^2-norm error of the temperature distribution versus the number of modes for global PGD, global POD and ℓ-PGD in its three proposed variants, computed respect FEM solution with the same discretization and up to an error rate of $e = 10^{-4}$.

3.3.3 Parametric transient heat transfer equation

This example shows the addition of the thermal conductivity k into the PGD formulation of the transient heat transfer equation (already considered in Section 3.3.2) as a parameter. It modifies the problem by moving it to a three-dimensional space, where the solution takes now the form $u(x,t,k) \in H_1(\Omega) \times L_2(\mathcal{I}) \times L_2(\mathcal{K}_k)$ where $\mathcal{I},\ \mathcal{K}_k \subset \mathbb{R}$ represent the considered intervals for time and conductivity, respectively. In the spirit of PGD, we assume a separate form for the unknown,

$$u(x,t,k) \approx \sum_{i=1}^{n} F_i(x) \cdot G_i(t) \cdot H_i(k).$$

At iteration $p+1$ we look for an enrichment of the already known approximation at order p (i.e., we employ a greedy algorithm),

$$u_{p+1}(x,t,k) \approx u_p(x,t,k) + R(x) \cdot S(t) \cdot P(k).$$

again, the admissible variation of the solution u^* is

$$u^*(x,t,k) = R^*(x) \cdot S(t) \cdot P(k) + R(x) \cdot S^*(t) \cdot P(k) + R(x) \cdot S(t) \cdot P^*(k).$$

The resulting nonlinear problem is solved by employing a fixed point algorithm.

We present a comparison of the L_2-norm error of the solution for one particular conductivity value (k=$0.05\ \frac{W}{mK}$) in Fig. 3.12.

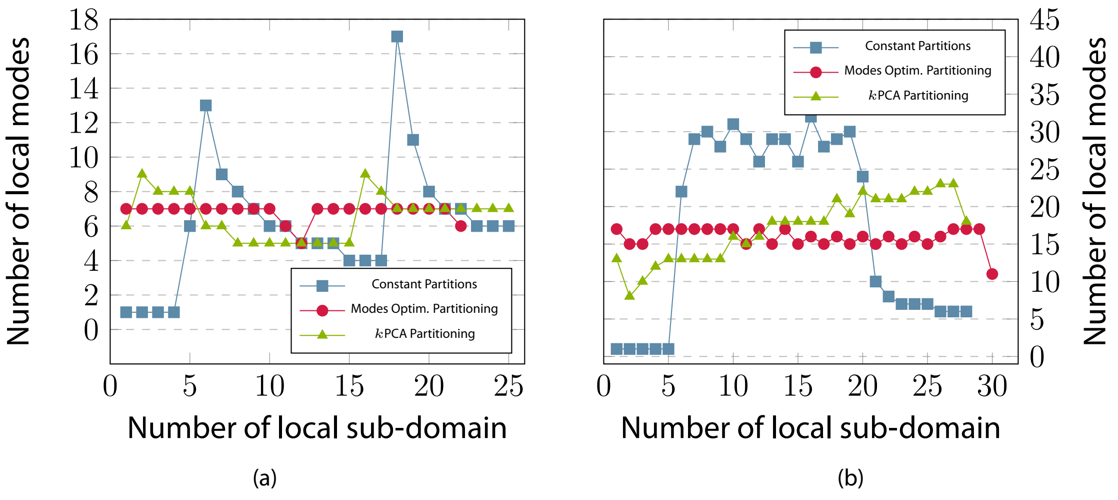


Figure 3.10: Number of modes per sub-domain for the three ℓ-PGD methods with a) x_f=$2\pi/7$ (global PGD needs 211 modes) and b) x_f=$5\pi/7$ (global PGD needs 382 modes)

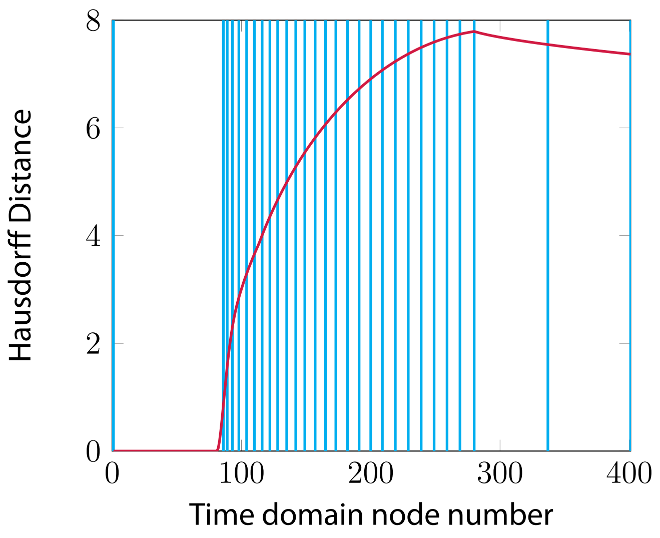


Figure 3.11: Hausdorff curve measured with respect the coordinate origin in red with sub-domain divisions in blue.

Note that in this example the Hausdorff distance plot becomes a surface because of the 3-dimensional nature of the problem. One of the advantages of $k\ell$-PGD method is that we can work in a general number of dimensions easily since k-PCA, Hausdorff distances and ε-neighborhoods are well-defined entities in any number of dimensions. The Hausdorff distance plot in this case is shown in Fig. 3.13. Note that, in general, the distance between results grows monotonically up to $t \approx 0.7$, and then remains constant (for $k = 0.0$) or decreases.

In general, as in previous examples, ℓ-PGD shows much better convergence rates with respect to the global versions of High-Order Singular Value Decomposition (HOSVD) and

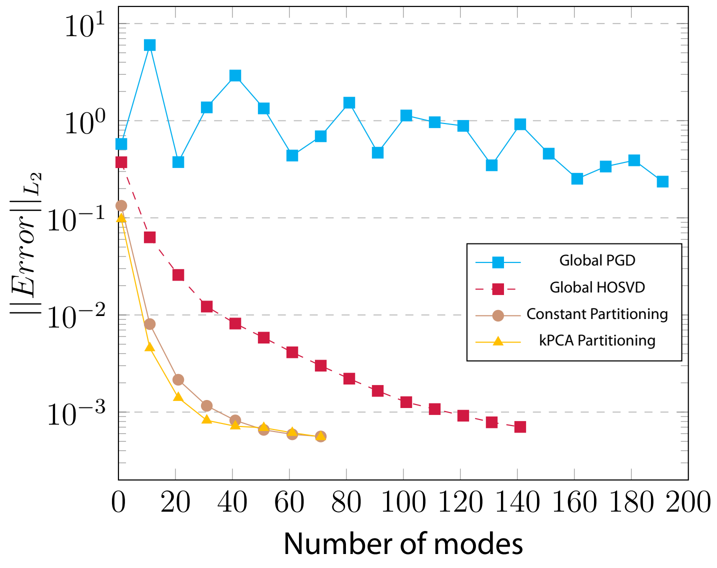


Figure 3.12: L^2-norm error of the temperature distribution versus the number of modes for global PGD, global High-Order SVD and ℓ-PGD in its three variants, computed respect FEM solution with the same discretization, in the parametric heat transient example.

PGD.

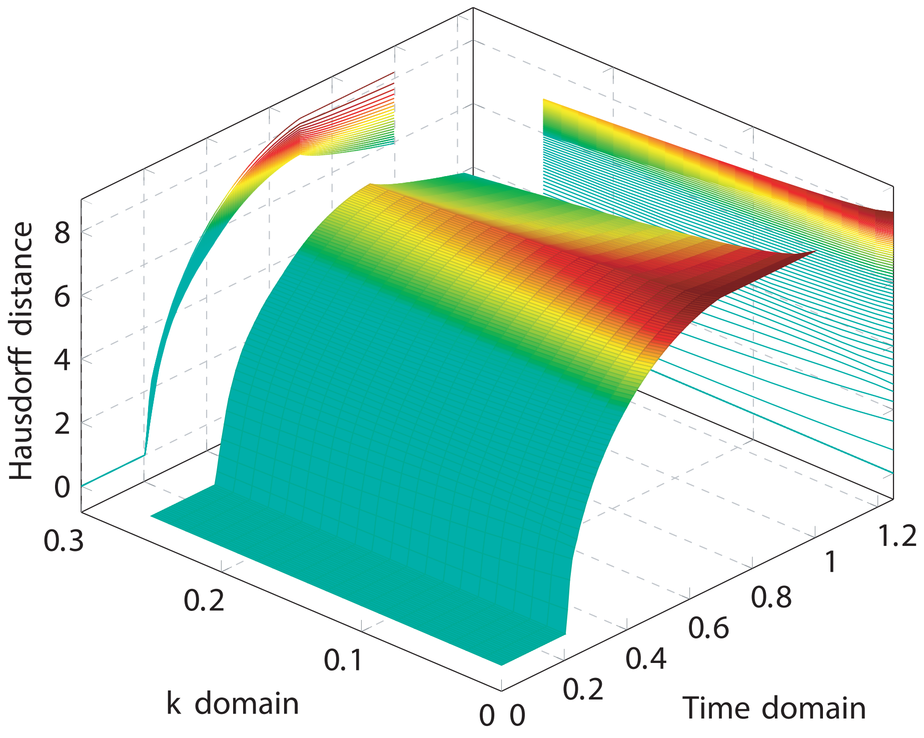


Figure 3.13: Hausdorff surface obtained from every time-kPCA Hausdorff curve moved along the k domain. Time domain is $[0, 1]$ and k domain is $[0, 0.25]$.

3.4 Chapter Conclusions

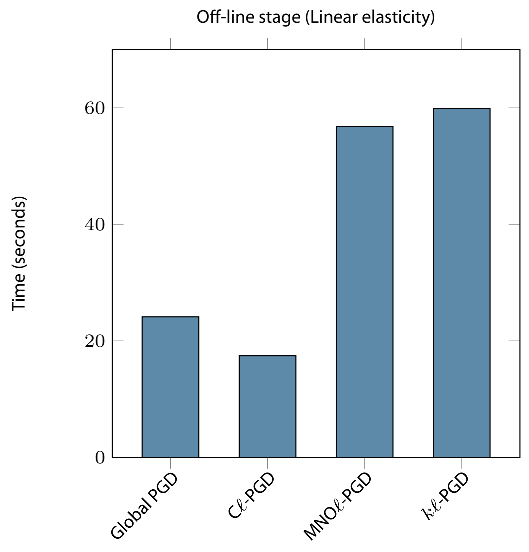
It is well known that for problems whose solution presents an intricate manifold structure, local model order reduction strategies provide with better results than global ones. However, very little has been done for *a priori* methods. The main reason for that is precisely the difficulty of determining the manifold structure of the solution *a priori*, with no snapshot of the solution available.

We have proposed here three possible methods for estimating the most appropriate size of the local sub-domains. Of course, many other methods in addition to those proposed may be envisaged. But what we know for sure is that adaptive strategies allow a better capture of the behavior of the solution manifold and look for an as constant as possible size of the reduced bases.

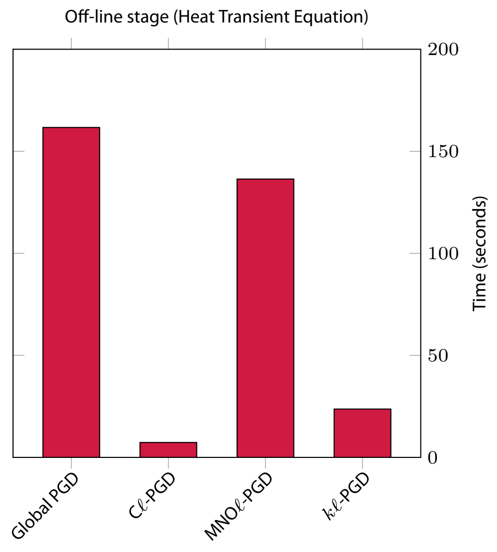
The method of constant division ($C\ell$-PGD) may be a valid option with respect to global implementations, but when compared with the other two adaptive methods shows its *brute force* nature. The adaptive method with mode number optimization (MNOℓ-PGD) is the optimal method when only one dimension of the whole domain needs to be adjusted to choose the size of the sub-domains. But when the number of dimensions increases we propose the $k\ell$-PGD as the best method to adaptive partitioning the domain with, as we have seen, excellent results with respect to global *a priori* and *a posteriori* strategies.

3.5 Appendix: Time and memory cost

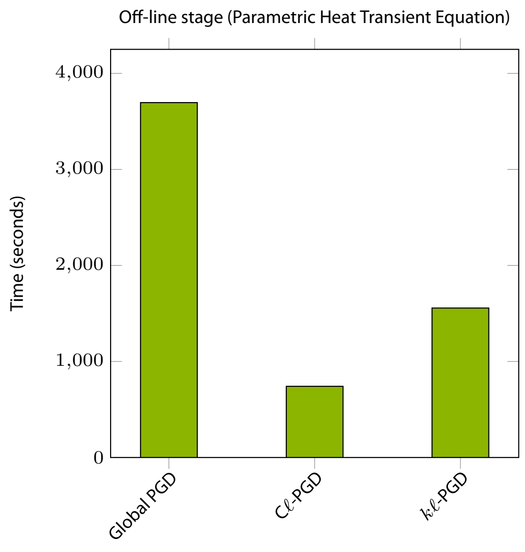
This appendix shows a graphical comparison between off-line computation time, on-line evaluation time and memory storage cost for all examples and methods. Note that the parametric heat transient example does not plot the MNOℓ-PGD method data, as it is not possible to estimate an optimal subdomain size when need more than one dimension may be modified.



(a)

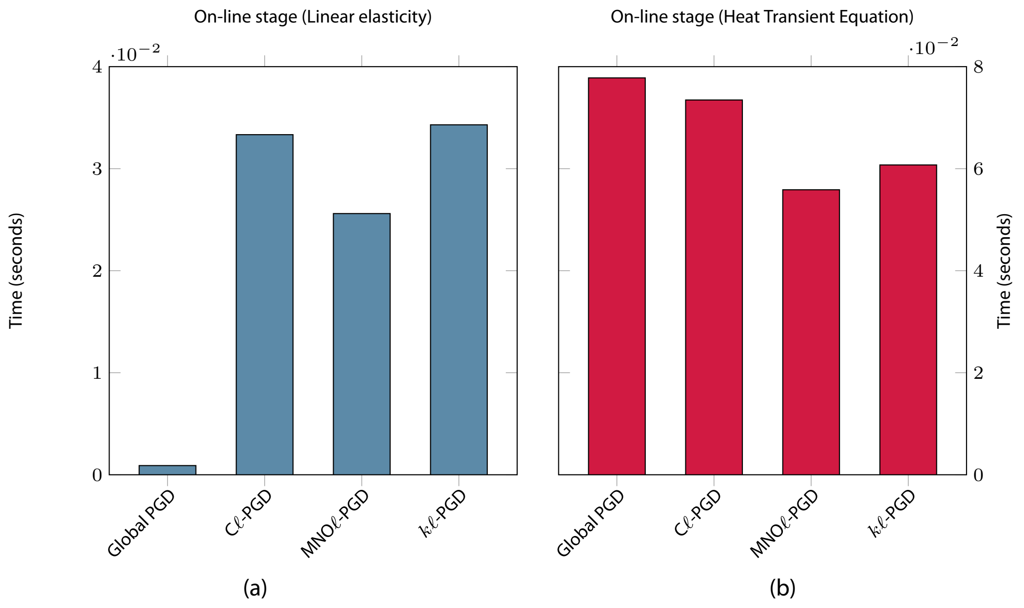


(b)



(c)

Figure 3.14: Time needed to compute the off-line stage for the a) Linear Elasticity example, b) Heat Transient Equation and c) Parametric Heat Transient Equation.



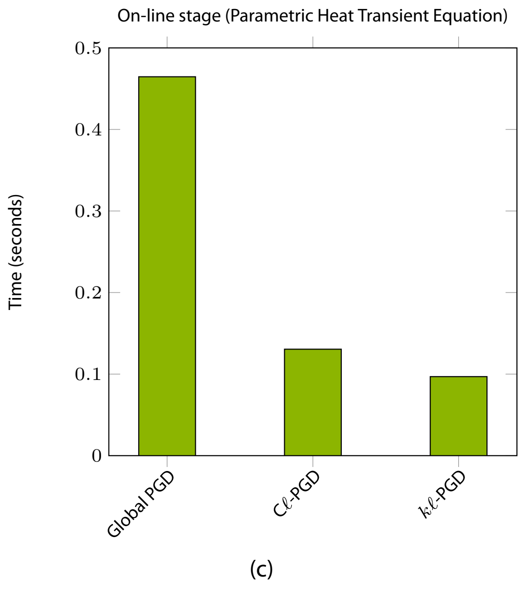


Figure 3.15: Time needed to compute the on-line stage for the a) Linear Elasticity example, b) Heat Transient Equation and c) Parametric Heat Transient Equation.

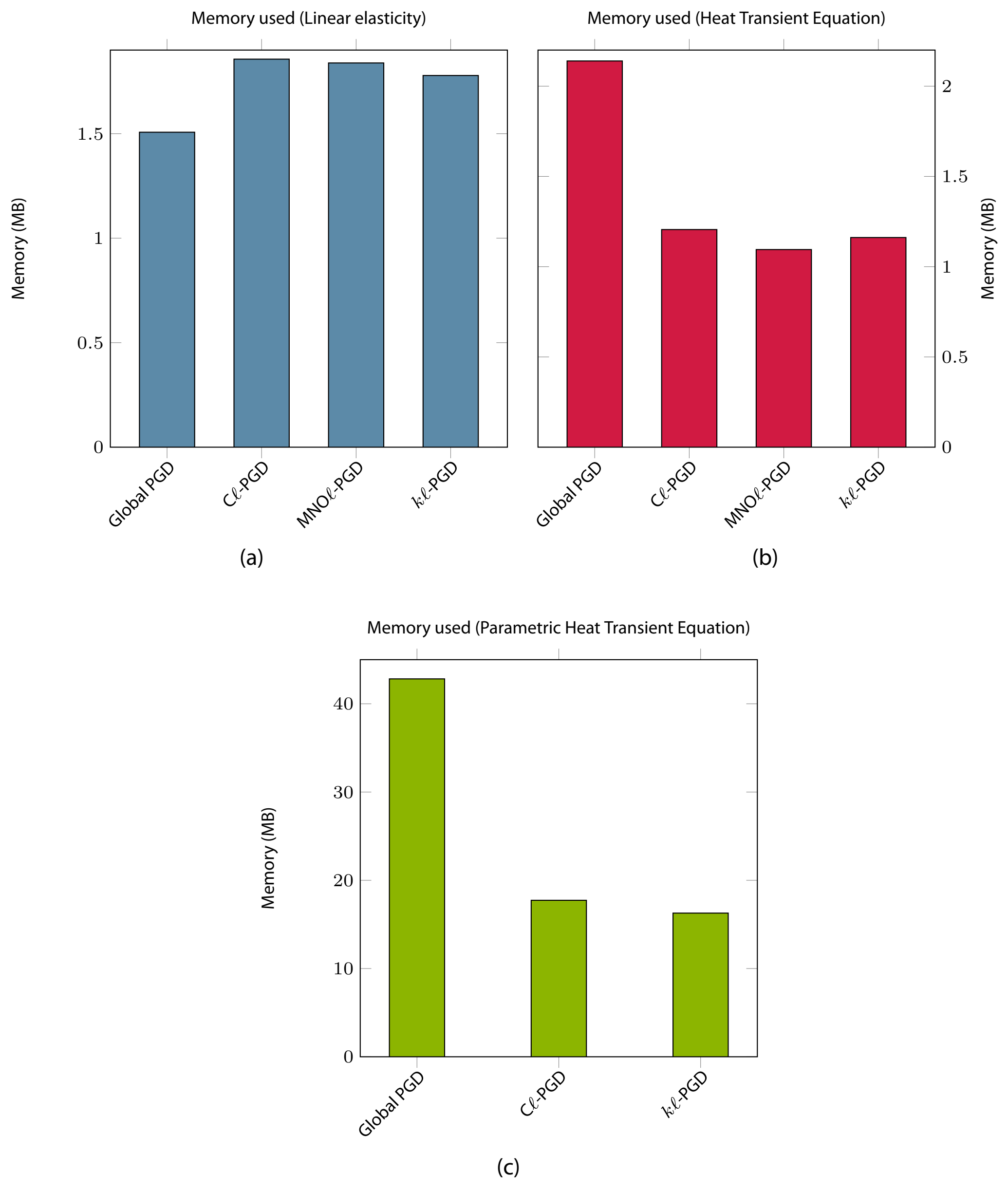


Figure 3.16: Memory cost for the a) Linear Elasticity example, b) Heat Transient Equation and c) Parametric Heat Transient Equation.

Chapter 4

Simulated Reality as a Data Assimilation Process

A first approximation to Simulated Reality can be made as a data assimilation problem of data coming from images, where the most relevant information is extracted using computer vision techniques. To do this, we assume the existence of a known parametric model whose parameter values are precisely what we want to obtain.

Data assimilation is the process of updating the state of a parametric model to match the data that is introduced into the system. Since the systems we work with are often complex and cannot be solved in real time with a standard solver, it is necessary to pre-compute the parametric solution. This implies that the different states on which the system can evolve must be delimited –since we are pre-calculating all potential possibilities– and are dependent on a set of parameters μ. Our parametric model must be able to perfectly estimate any possible solution, so we can affirm that we are applying a pure deterministic approach in this chapter. We trust there are no deviation errors in the data with respect to the model. However, other approaches could be taken into account, as it is stated in the conclusions chapter (Sec. 4.6).

The purpose of this chapter is to assimilate data to simulate the behavior of deformable objects. The covered problems are, therefore, the image acquisition in deformable objects, computational mechanics and model order reduction. We also present the framework to mix all these tools in a single application that works in real time. Finally, several examples are presented to justify the viability of our method. Some of the content of this chapter can be found in the following journal paper

- Badías, A., Alfaro, I., González, D., Chinesta, F. and Cueto, E. (2018). Reduced order modeling for physically-based augmented reality. *Computer Methods in Applied Mechanics and Engineering*, 341, 53-70.

4.1 Introduction

Embedding virtual objects in a real environment using video sequences and making them interact is still a challenging task. The challenge comes precisely from the fact that current video cameras record at a minimum of 30 frames per second (fps). Some modern smartphones may record at 60 fps, and even some models record at 120 or 240 fps in slow motion mode. Therefore, to embed a virtual object in a video stream involves generating synthetic images —compliant with the laws of physics— at almost those 30 fps.

In this chapter we produce physically-based synthetic images at such rates. We propose the augmentation of non-rigid real solids captured in a video sequence with virtual information related to their deformations (e.g. stress or strain fields, deformation of internal parts or other hidden details). This could have a vast range of possible applications, from monitoring and decision making in the framework of Industry 4.0, to assistance for laparoscopic surgery [Haouchine et al., 2015b], to name some examples. For instance, augmented reality could allow us to project in different formats (such as tablets, smartphones or AR glasses) hidden information within a manufacturing process (strains, stresses or defects, among others). In biomedical applications like laparoscopic surgery, some critical information such as the precise location of blood vessels is often hidden for the surgeon [Haouchine et al., 2015a], and the techniques proposed in this chapter may help to bring information to the specialist. We consider the possibility of enriching existing objects with hidden and valuable information to the user, as well as incorporating synthetic objects to the scene. Projecting all this information on top of video streams could be of utmost interest for these and many other applications.

Our model should be able to generate results at the prescribed rate. If we consider, for instance, deformable solids with general non-linear constitutive laws under finite strain settings, the problem of solving such a model some thirty times per second becomes an evident bottleneck. To avoid such problems, we make use of reduced-order mechanical models that govern the actual behavior of the solids and we solve the inverse problem related to the estimation of the kinematic (strain) and dynamic (stress) states.

The precise type of reduced order model is not an essential ingredient of the technique here presented. For instance, Reduced Basis techniques [Quarteroni et al., 2011] [Maday and Ronquist, 2004] [Maday et al., 2002] have been employed successfully for data assimilation problems [Manzoni et al., 2016] [Manzoni et al., 2014] [Manzoni et al., 2012a]. Here, Proper Generalized Decompositions (PGD) have been employed [Ladeveze, 1999] [Chinesta and Cueto, 2014] [Ladeveze et al., 2010] [Cueto et al., 2016] [Chinesta et al., 2010] [Ladeveze and Chamoin, 2011] [Croft and Phillips, 2017] [Badías et al., 2017]. As we have seen in Chapter 2, this technique constructs the solution of the problem in the form of a finite sum of separate functions. This particular form of the solution, which has a lot in common with Reduced Basis techniques, greatly simplifies the task of inverse identification that will be needed in this type of problems, as will become clear hereafter [Gonzalez et al., 2012] [Ammar et al., 2014] [Aguado et al., 2015] [Ghnatios et al., 2012].

We do not impose artificial spatiotemporal restrictions or priors, nor *ad hoc* energy impositions [Bartoli et al., 2015], since we are solving the real physics under a finite element framework guaranteeing energy conservation and elastic (or hyperelastic) constitutive laws. Of course, initial and boundary conditions need to be applied in the computation step, but they can also be parameterized in fact. The proposed method can be applied both to articulated and continuous solids, beating any other existing method in terms of frequency rates (since it only requires an *evaluation* of the parametric solution, rather than a true simulation), accuracy and robustness (noisy observations are minimized without the need of bayesian implementations). Our method has no dependency on the acquisition system used (monocular, stereo or RGB-D systems). In fact, we think that one of the biggest advantages appears with the use of monocular sensors, since usually it results in an ill-posed problem but the use of parametric minimizations reduces the resulting complexity, obtaining an optimal solution.

Being purist, we can say that our procedure is not a structure-from-motion method as we are not building a 3D structure from the motion data. The proposed method is related to the *Shape from Template* family of methods [Bartoli et al., 2015], in which we register an off-line (reduced-order) parametric model to the camera observations. To better understand the procedure, we detail next the type of model order reduction methods commonly applied to continuum mechanics and how they can be of great help in this application.

4.2 Problem Formulation

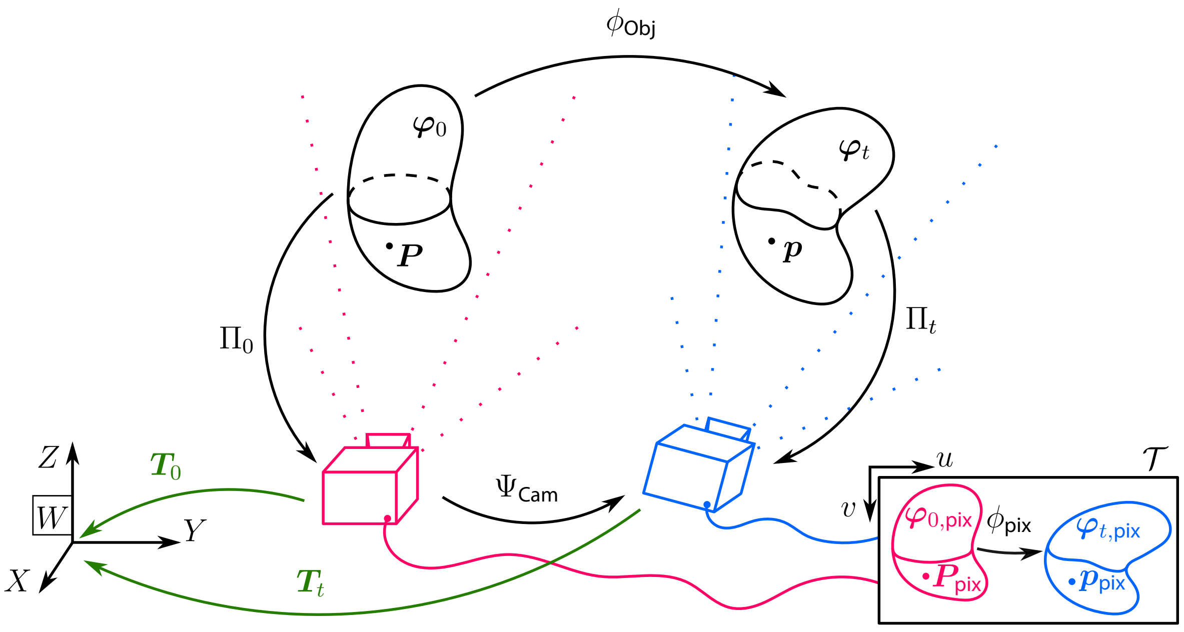


Figure 4.1: Kinematic modeling of the proposed formulation.

Very often the so-called Non-Rigid Structure from Motion (NRSfM) problem has been formulated related to the motion of the visible portion of the deforming solid (the domain for which information captured by the camera is available). However, it is well-known that the motion of the visible part of the solid is heavily influenced by internal details, as described by the Navier equations of solid mechanics. Therefore, in this section we formulate the problem as arising from a complete, three-dimensional description of the solid [Bonet and Wood, 2008], thus avoiding previous formulations focused only in the visible part of the solid whose motion was modeled by shell finite elements [Agudo et al., 2016], [Bartoli et al., 2015]. The type of problems we are interested in (augmented reality for mechanical design, surgery guidance, ...) allows us to assume that a full three-dimensional, albeit reduced-order model is available off-line.

4.2.1 Kinematic description of deforming solids

Neglecting inertial effects and assuming that there is no change in temperature, we define a continuous solid Ω where we identify a point $\boldsymbol{P}$, with coordinates (X, Y, Z) respect to the *World* (*W* in Fig. 4.1). These coordinates, the *undeformed configuration*, are given by a map $\varphi_0 : \Omega \rightarrow \mathbb{R}^3$. This object undergoes a deformation so that point $\boldsymbol{P}$ moves to the new position $\boldsymbol{p}$ with coordinates (x, y, z) (also with respect to *W*) in the deformed configuration, $\varphi_t : \Omega \rightarrow \mathbb{R}^3$. The function that explains the movement of the object is $\phi_{\text{Obj}} : \Omega \times \mathcal{I} \rightarrow \mathbb{R}^3$, being $\mathcal{I} \in \mathbb{R}$ the time interval where the movement is produced. Therefore, making use of a Lagrangian description (material, always following the same infinitesimal particle) of the amount of movement, we can express the deformed configuration of the point $\boldsymbol{P}$ as

$$\boldsymbol{x} = \phi_{\text{Obj}}(\boldsymbol{X}, t),$$

being $\boldsymbol{x}$ the coordinates of point $\boldsymbol{p}$ in the deformed configuration, $\boldsymbol{X}$ the coordinates of point $\boldsymbol{P}$ in the initial configuration and t the instant of time elapsed between both states. We call deformation gradient tensor (a two-point tensor) to the fundamental quantity that explains the deformation produced between two neighboring particles between the initial and the deformed state (assuming continuity in the mapping function). It is defined, with respect to the initial configuration, as

$$\boldsymbol{F} = \frac{\partial \phi_{\text{Obj}}(\boldsymbol{X}, t)}{\partial \boldsymbol{X}} = \boldsymbol{\nabla} \phi_{\text{Obj}},$$

or, alternatively,

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix}.$$

It is convenient to define a measurement system independent of the type of movement carried out on the solid (rigid-solid translation or rotation, relative deformations and the combination of all of them). For independence in rotation, we use the *right Cauchy-Green deformation tensor*, defined as

$$\boldsymbol{C} = \boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{F}.$$

And finally, using the *Green-Lagrangian strain tensor* (material reference) we can measure the deformations independently of rigid-body motions (pure translation)

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}),$$

or also,

$$\boldsymbol{E} = \frac{1}{2}\Big((\boldsymbol{\nabla u})^T + \boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T\boldsymbol{\nabla u}\Big), \tag{4.1}$$

with $\boldsymbol{u}(\boldsymbol{X}) = \boldsymbol{x} - \boldsymbol{X}$ the material displacement. We are obtaining the concept of *strain* used to evaluate the deformations that appear locally in a solid body. The particularization of this tensor applied to linear elasticity (linear behavior of the material) and using hypotheses of small deformations (assuming infinitesimal deformations, which are very small with respect to the dimensions of the object) allows us to obtain the well-known *Cauchy strain tensor*,

$$\boldsymbol{\varepsilon} = \frac{1}{2}\Big((\boldsymbol{\nabla u})^T + \boldsymbol{\nabla u}\Big).$$

4.2.2 Camera-centered kinematic description

Assume now that we obtain an image of the boundary of the solid at the undeformed configuration, $\partial\Omega_0$ (we consider, for simplicity, opaque solids occluding any interior detail). We consider a perspective projection camera [Hartley and Zisserman, 2003] (after intrinsic calibration and lens distortion correction [Bradski and Kaehler, 2008]) applying a transformation $\Pi_0 : \partial\Omega_0 \to \mathcal{T}$, being $\mathcal{T} \in \mathbb{R}^2$ the input image space. This camera operator Π_0 maps a point $\boldsymbol{P}$ from *World* coordinates to $\boldsymbol{P}_{\text{pix}}$ in *pixel* coordinates (u, v) making use of the camera intrinsic and extrinsic parameters

$$\boldsymbol{P}_{\text{pix}} = \Pi_0(\boldsymbol{P}).$$

The camera projection can be also expressed as $\Pi_0 = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$ where $\boldsymbol{K}$ corresponds to the camera intrinsic matrix and $[\boldsymbol{R}|\boldsymbol{t}]$ represents the camera extrinsic matrix ($\boldsymbol{T}_0$ in Fig. 4.1). In its expanded view, the projection can be expressed as

$$\begin{bmatrix} u \\ v \\ s \end{bmatrix} = \lambda \begin{bmatrix} {}^{f}/_{d_x} & 0 & c_x \\ 0 & {}^{f}/_{d_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

The camera extrinsic matrix transforms 3D scene points into 3D points in the camera reference. It establishes the rotation ($\boldsymbol{R}$) and translation ($\boldsymbol{t}$) between the corresponding points. The camera intrinsic matrix $\boldsymbol{K}$ contains information about the projection centre (c_x,c_y), the pixel size (d_x,d_y) and the focal distance (f), which is used to map 3D camera points into 2D pixel coordinates. The whole equation makes possible the transformation of real 3D points into image pixels.

The same process occurs with the $\Pi_t : \partial\Omega_t \to \mathcal{T}$ projection, where points in the deformed configuration φ_t are projected to the $\varphi_{t,\text{pix}}(\Omega_{\text{pix}}) \subset \mathbb{R}^2$ image of the deformed configuration.

4.2.3 Solving the unknown kinematics

Using a monocular implementation, the mapping functions unknown in this problem for any time instant $t \in \mathcal{I}$ are

- The displacement of the current configuration in a material description (i.e., with respect to the reference configuration) $\phi_{\text{Obj}}(\boldsymbol{X}, t)$.
- The camera projection Π_t, but only the extrinsic component, since it is the part that can change with the camera movement (and also the redundant function $\Psi_{\text{Cam}} : \mathbb{R}^3 \to \mathbb{R}^3$, that maps the movement of the camera with respect its initial position).

The estimation of the mapping Π_t for each frame capturing rigid scenes is known as the Structure-from-Motion (SfM) problem and it involves the estimation of the extrinsic parameters T_t (and the three-dimensional position of the points of the scene $\boldsymbol{P}_t$), by minimizing a reprojection error in L_2-norm. If we also add the estimation of ϕ_{Obj}, where objects can deform at any time instant, the problem becomes more difficult (ill-posed). It is possible that a different deformed configuration appears at every instant, with different camera positions. In other words, the camera captures both static points (rigid in the scene) and points that undergo displacements. The static points (the blue ones in Fig. 4.2) allow to estimate the pose of the camera, making it possible to determine the deformations of the non-rigid objects in the scene (red points in Fig. 4.2).

The position and orientation (pose) of the camera T_t is estimated by using known fixed points (that can be fiducial markers or image features, depending on the particular application) along with consensus techniques, such as RANSAC [Fischler and Bolles, 1981], to identify the subset of fixed points in the scene. The success of RANSAC techniques depends on the ratio of fixed points to deformable points. To estimate the position of the cameras, the reprojection error $d(\boldsymbol{x}_{\text{pix}}, \hat{\boldsymbol{x}}_{\text{pix}})$ is minimized, with d the euclidean distance between the pixel coordinates of a point observed in an image and the pixel coordinates of the point reprojected on the same image (from the 3D point $\boldsymbol{x}$ and the camera matrix Π_t, parameters that are also optimized). Using, for simplicity, the minimum number of positions of the camera, two—say 0 and t in Eq. (4.2)—, the minimization of the following

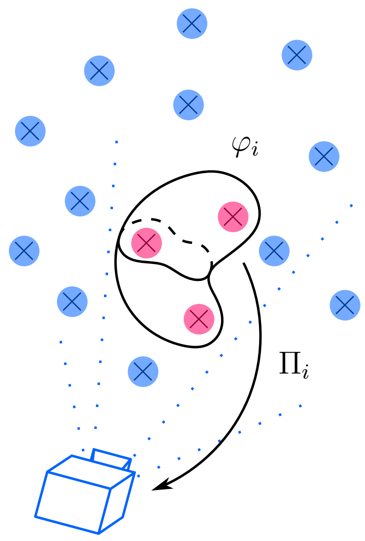


Figure 4.2: In general, we consider applications where an important part of the scene (blue points) are static, while some region (red points) is non-rigid or deformable, and thus may change its position.

reprojection functional is accomplished

$$\sum_{j=1}^{N} d(\boldsymbol{x}^j_{\text{pix},0}, \hat{\boldsymbol{x}}^j_{\text{pix},0})^2 + d(\boldsymbol{x}^j_{\text{pix},t}, \hat{\boldsymbol{x}}^j_{\text{pix},t})^2 \tag{4.2}$$

over the pre-estimated fundamental matrix $\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{x}}^j_{\text{pix},t}$, $j = 1, \ldots, N$, with N the number of identified (and matched) points. This distance functional is minimized using the Levenberg-Marquardt [Press et al., 1988] algorithm over $3N$ (3D points) + 12 (2 camera source locations) variables. The problem can easily be extrapolated to an M number of images (camera positions).

The next task is to estimate the deformations produced in the 3D object from its projection in the images. We use the red points in Fig. 4.2, or in other words, the *outliers*. RANSAC techniques applied to SLAM usually neglect those points that do not remain fixed in the scene, using only the inliers. But our true interest relies in the information associated with the deformation of the objects. It is located precisely there, in the outliers. Therefore, these points are the ones that provide the most valuable information about the displacements suffered by the deformable objects in the scene, which we have to identify after carrying out a preliminary registration of the objects.

4.2.4 Equilibrium in the deformable solid

In the previous section we have included a brief summary of the kinematic analysis of the deformable solid, but to fully resolve the mechanical problem it is necessary to take into account the stress analysis so as to guarantee equilibrium. If we split the solid by an arbitrary plane of unit normal $\boldsymbol{n}$, a force per unit area will appear at every point of the newly created surface,

$$\boldsymbol{t}(\boldsymbol{n}) = \boldsymbol{\sigma}\boldsymbol{n}; \qquad \boldsymbol{\sigma} = \sum_{i,j=1}^{3} \sigma_{ij}\boldsymbol{e}_i \otimes \boldsymbol{e}_j$$

where $\boldsymbol{t}$ is the stress vector, $\boldsymbol{n}$ is the vector normal to the plane and $\boldsymbol{\sigma}$ is the *Cauchy stress tensor*.

The stress magnitude that is *conjugate* to the gradient of deformation tensor $\boldsymbol{F}$—their product gives an energy measure—is, however, the so-called *first Piola-Kirchhoff stress tensor $\boldsymbol{P}$*, defined as

$$\boldsymbol{P} = J\boldsymbol{\sigma}\boldsymbol{F}^{-T}$$

being $J = \det(\boldsymbol{F})$ the Jacobian of the transformation φ. The stress measure conjugate to tensor $\boldsymbol{E}$ is the *second Piola-Kirchhoff stress tensor $\boldsymbol{S}$* defined as

$$\boldsymbol{S} = J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T},$$

so that we can write equivalently

$$\boldsymbol{P} = \boldsymbol{F}\boldsymbol{S}.$$

In order to guarantee equilibrium in the solid, we must satisfy the following equilibrium equation, in the absence of inertia terms—assuming a quasi-static process:

$$\boldsymbol{\nabla}\cdot\boldsymbol{P} + \boldsymbol{B} = 0 \text{ in } \Omega, \tag{4.3}$$

being $\boldsymbol{B}$ the applied body forces. These equations need to be supplemented with appropriate boundary conditions,

$$\begin{cases} \boldsymbol{\sigma}\,\boldsymbol{n} & = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_t, \\ \boldsymbol{u} & = \bar{\boldsymbol{u}} \quad \text{on } \Gamma_u \end{cases}$$

where $\bar{\boldsymbol{t}}$ are the traction forces acting along the boundary region Γ_t in terms of the unit normal to the boundary, $\boldsymbol{n}$, and finally $\bar{\boldsymbol{u}}$ are the prescribed displacements defined in Γ_u. Displacement and traction boundary conditions cannot be defined in the same of portion of the boundary, so that

$$\Gamma_u \cap \Gamma_t = \emptyset.$$

The *weak form* of Eq. (4.3) (also called *boundary value problem*) is created to relax continuity and derivability restrictions on the functions that approximate the final solution, and is obtained after multiplying by an arbitrary function (admissible variation) and integration by parts,

$$\int_\Omega \boldsymbol{S} : \delta\boldsymbol{E}\, d\Omega = \int_{\Gamma_t} \bar{\boldsymbol{t}}\, \delta u\, d\Gamma_t. \tag{4.4}$$

where $\boldsymbol{E}$ is the *Gree-Lagrange strain tensor* (Eq.4.1).

4.2.5 Constitutive equations

The last ingredient in describing the process underwent by the deformable solid is to describe its constitutive nature. In this work we consider hyperelastic materials. Hyperelastic materials are those whose stress-strain relationship derives from a strain energy density function W, whose precise form is usually determined from physical experiments (constitutive model). For these materials, the second Piola-Kirchhoff tensor $\boldsymbol{S}$ can be obtained from the strain energy density function W as

$$\boldsymbol{S} = \frac{\partial W(\boldsymbol{E})}{\partial \boldsymbol{E}}. \tag{4.5}$$

Substituting Eq. (4.5) onto Eq. (4.4) provides the problem to solve by using finite element methods [Fish and Belytschko, 2007],

$$\int_{\Omega} \delta \boldsymbol{E} : \mathbf{C} : \boldsymbol{E} \, d\Omega = \int_{\Gamma_t} \bar{\boldsymbol{t}} \, \delta u \, d\Gamma_t, \tag{4.6}$$

where **C** is the fourth-order constitutive tensor. Sec. 4.5 shows some practical examples of hyperelastic materials.

4.3 Dimensionality Reduction

The computational complexity of the problem just presented is such that it prevents its solution under real-time requirements. Note the nonlinear character imposed by the strain measures introduced in Eq. (4.1). To guarantee that we comply with video requirements, a reduction of the complexity of the problem is mandatory. In addition, in this section we re-formulate the problem as being parametric. These parameters will be, in general, any magnitude of interest from the augmented reality point of view. In other words, intrinsic magnitudes, in general hidden to the end user, that are going to be displayed in the screen or employed to determine the true magnitudes of interest. Their precise meaning will be clear hereafter.

4.4 Data Assimilation from Video Sequences

Once we have detailed the kinematic description of the solid movement, its capturing with a moving camera and a very efficient means of storing the solid's response in separate form, the final step of the proposed method is to formulate the augmentation of the video sequence as a data assimilation problem. In other words, the estimation, in any frame of the sequence, of the parameters that minimize the error between the measured data and the parametric solution of the problem, projected onto the image plane. The goal, therefore, is to minimize the following error functional

$$\mathcal{J}(\boldsymbol{\mu}) = \sum_{j=1}^{\mathrm{n_{meas}}} \left(\boldsymbol{u}_{\mathsf{pix}}^{\mathsf{meas}}(\boldsymbol{x}_j) - \Pi_t \left(\boldsymbol{u}^{\mathsf{PGD}}(\boldsymbol{x}_j, \boldsymbol{\mu}) \right) \right)^2, \tag{4.7}$$

being $\mathrm{n_{meas}}$ the number of measurements, $\boldsymbol{u}_{\mathsf{pix}}^{\mathsf{meas}}$ the pixel coordinates measured in the image, $\boldsymbol{u}^{\mathsf{PGD}}(\boldsymbol{x}, \boldsymbol{\mu})$ the 3D displacement predicted by the reduced-order parametric solution and Π_t the projection to the image plane at any instant t (frame).

Using Eq. (4.7) we can estimate the value of the $\boldsymbol{\mu}$ parameters that minimize the projection error

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}}{\mathsf{argmin}}\, \mathcal{J}(\boldsymbol{\mu}).$$

This minimization can be advantageously accomplished by employing Levenberg-Marquardt algorithms [Levenberg, 1944] [Marquardt, 1963], which take the form

$$[\boldsymbol{J}^T \boldsymbol{J} + \lambda\, \mathsf{diag}(\boldsymbol{J}^T \boldsymbol{J})]\, \delta = \boldsymbol{J}^T \left[\boldsymbol{u}_{\mathsf{pix}}^{\mathsf{meas}}(\boldsymbol{x}_j) - \Pi_t \left(\boldsymbol{u}^{\mathsf{PGD}}(\boldsymbol{x}_j, \boldsymbol{\mu}) \right) \right], \tag{4.8}$$

where δ is the perturbation parameter that moves the parameters in the direction of steepest descent and λ an algorithmic parameter that adaptively varies the behavior between the Gradient Descent Method and the Gauss-Newton Method. Finally, $\boldsymbol{J}$ is the Jacobian matrix, defined as

$$\boldsymbol{J} = \frac{\partial \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}}. \tag{4.9}$$

Here lies precisely the key point that makes the separate representation of the solution so powerful in the parameter optimization process. Recall the general form of PGD, Eq. (2.9), where we can notice that due to its separate form also a separate differentiation can also be performed. We can precompute the derivative function of each parameter offline, Eq. (4.9), since the finite element approximation computes the derivative as a multiplication of matrices. These derivative functions are usually known as *sensitivities* of the solution with respect to each parameter μ [Chinesta et al., 2013b]. We can interpret them as the local variation of the function $u(\boldsymbol{x}, \boldsymbol{\mu})$ with respect to changes in a parameter μ_k, where $k = 1, \ldots, \mathrm{n_{param}}$,

$$\boldsymbol{J}_k = \frac{\partial \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\mu})}{\partial \mu_k} = \sum_{i=1}^{N} \boldsymbol{F}_i^1(\mu_1) \circ \ldots \circ \frac{\partial \boldsymbol{F}_i^k(\mu_k)}{\mu_k} \circ \ldots \circ \boldsymbol{F}_i^{\mathrm{n_{param}}}(\mu_{\mathrm{n_{param}}}).$$

This method allows us to avoid the exploration of all of the parametric space in search of the derivatives, and directly apply the differentiation on the separated variable vectors (Fig. 4.3), involving a great reduction in the computational task.

The general procedure has been explained, and some of our examples have been developed using the described proceeding. But in the next section, we particularize the whole strategy to the ORB-SLAM2 method, where some specific treatments need to be applied.

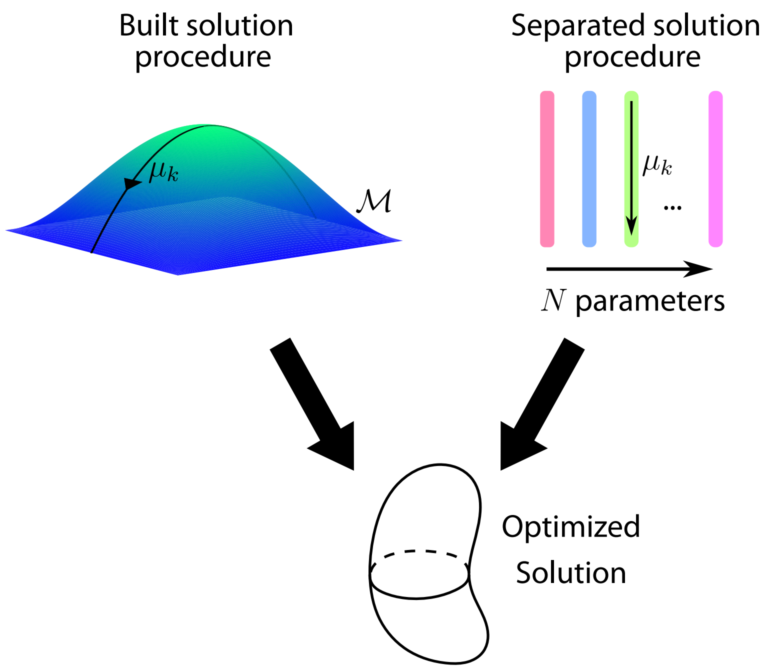


Figure 4.3: A *built solution procedure* implies the iterative evaluation of the derivatives within the multiparametric space. The *separated solution procedure* allows to perform the optimization with respect to k parameters of interest, so the search is carried out only in one direction instead of the whole parametric hypersurface.

4.4.1 A deformable implementation of ORB-SLAM2

We propose in this section a deformable implementation of the ORB-SLAM2 system [Mur-Artal and Tardós, 2017]. ORB-SLAM2 is a sparse and feature-based SLAM system with outstanding results of accuracy [Mur-Artal et al., 2015], and "as far as we know, [...] the most complete feature-based monocular vSLAM system" [Taketomi et al., 2017]. The unprecedented results made us to select ORB-SLAM2 as the basis system to achieve the degree of accuracy needed in augmented reality problems.

ORB-SLAM2 system is used to build the rigid scene and locate the camera in any frame. After a few seconds of video capturing the static scene, we obtain the 3D point cloud and apply an automatic and on-line registration against a CAD model of the object, based in RANSAC methods and the ICP (Iterative Closest Point) algorithm —we used the Point Cloud Library Implementation [Rusu and Cousins, 2011], a variation of the original ICP registration algorithm [Besl and McKay, 1992] [Chen and Medioni, 1992], but many other implementations are also available [Pomerleau et al., 2013].

The registration step allows us to estimate the displacements of the 3D SLAM points ($\boldsymbol{U}_{\text{SLAM}}(\boldsymbol{\mu})$) using the precomputed data, Eq. (4.10). And also a mapping function between our undeformed 3D model and the 2D image features can be established. It means we can afterwards deform the real object, extract image features in any frame and estimate the parameter values $\boldsymbol{\mu}$ by minimizing the functional of Eq. (4.7) against the precomputed set of solutions

$$\boldsymbol{U}_{\text{SLAM}}(\boldsymbol{\mu}) = f\Big(\boldsymbol{U}_{\text{Neighs,CAD}}(\boldsymbol{\mu})\Big). \tag{4.10}$$

The goal is to obtain the displacements $\boldsymbol{U}_{\text{SLAM}}(\boldsymbol{\mu})$ to add to the static set $\boldsymbol{V_P}$ of deformable 3D points initially estimated using the SLAM technique. This solution must maintain the separate multiparametric structure to obtain a specific solution for each possible value of our set of parameters. Therefore, the displacements of the SLAM points are expressed as a function of the closest registered CAD points. For this, computationally speaking, we establish a pointer that links each *MapPoint* of the SLAM with 3 neighbors of the CAD model to interpolate the solution using weighted values depending on the euclidean distance. In other words, we do not calculate $d\boldsymbol{P}$ for each SLAM point as such; it is estimated based on the nearest neighbors, being also that registration process capable to be updated in real time to optimize some of the parameters (as it happens in the α_Z parameter estimation in example 4.5.4). The procedure can be seen in the general overview schema (Fig. 4.4).

Blue camera locations are local key-frames and the green camera location shows the current frame. In the visual scanned point cloud there are 3D static points (used to fix camera position and orientation) and 3D deformable points, that may or may not be deformed in the current observation. At first, the whole set of 3D points ($\boldsymbol{V}_{\text{P}}$) is considered as formed by static points, but after the registration process, those points belonging to the deformable object are considered as possibly deformable points, belonging to the subset $\boldsymbol{V}_{\text{DP}}$, and the rest of points as static ($\boldsymbol{V}_{\text{SP}}$), being $\boldsymbol{V}_{\text{P}} = \boldsymbol{V}_{\text{DP}} \cup \boldsymbol{V}_{\text{SP}}$ and $\boldsymbol{V}_{\text{DP}} \cap \boldsymbol{V}_{\text{SP}} = \emptyset$. The camera captures and projects on the image plane a subset of points ($\boldsymbol{V}_{\text{p}}^{\text{obs}} \subset \boldsymbol{V}_{\text{p}}$), where $\boldsymbol{V}_{\text{p}}$ is the set of all points projected on the image plane and $\boldsymbol{V}_{\text{p}}^{\text{obs}}$ is the subset of points projected on the image plane and observed by the camera. The static points observed on the image create the subset $\boldsymbol{V}_{\text{sp}}^{\text{obs}}$ and the set of deformable points form the subset $\boldsymbol{V}_{\text{dp}}^{\text{obs}}$. Let's focus on the deformable 3D point $\boldsymbol{P} \in \boldsymbol{V}_{\text{DP}}$, with coordinates (X, Y, Z). The initial scanning estimates the static position of point $\boldsymbol{P}$ and when registration with respect to the CAD object occurs, that point is linked to the 3 nearest points of the CAD model. The position of point $\boldsymbol{P}$ in its deformed configuration is $\boldsymbol{P'}$, being $\boldsymbol{dP} = (\boldsymbol{P'} - \boldsymbol{P}) = (X', Y', Z') - (X, Y, Z)$, but this deformation $\boldsymbol{dP}$ is really estimated from the deformations of the neighbors of the CAD object, which depend on the multiparametric solution projected in separate variables. The point in its deformed configuration ($\boldsymbol{P'}$) is observed from the image as $\boldsymbol{p'}$ with pixel coordinates (u', v'), which must match the projection $\boldsymbol{p}'_e = \Pi_t(\boldsymbol{P} + \boldsymbol{dP}(\boldsymbol{\mu}))$ for the optimal set of parameters $\boldsymbol{\mu}$. The reprojection error in the image is therefore defined as $d(\boldsymbol{p'}, \boldsymbol{p}'_e)$ and the optimal parameter values of our parametric solution that minimize this error for the whole set of observed points are obtained from the next functional (similar to Eq. (4.7))

$$\mathcal{J}(\boldsymbol{\mu}) = \sum_{j=1}^{n_{\text{meas}}} \rho\Big(\boldsymbol{U}_{\text{pix}}^{\text{meas}}(\boldsymbol{x}'_j) - \Pi_t\left(\boldsymbol{U}^{\text{MOR}}(\boldsymbol{X}'_j, \boldsymbol{\mu})\right)\Big)^2 \tag{4.11}$$

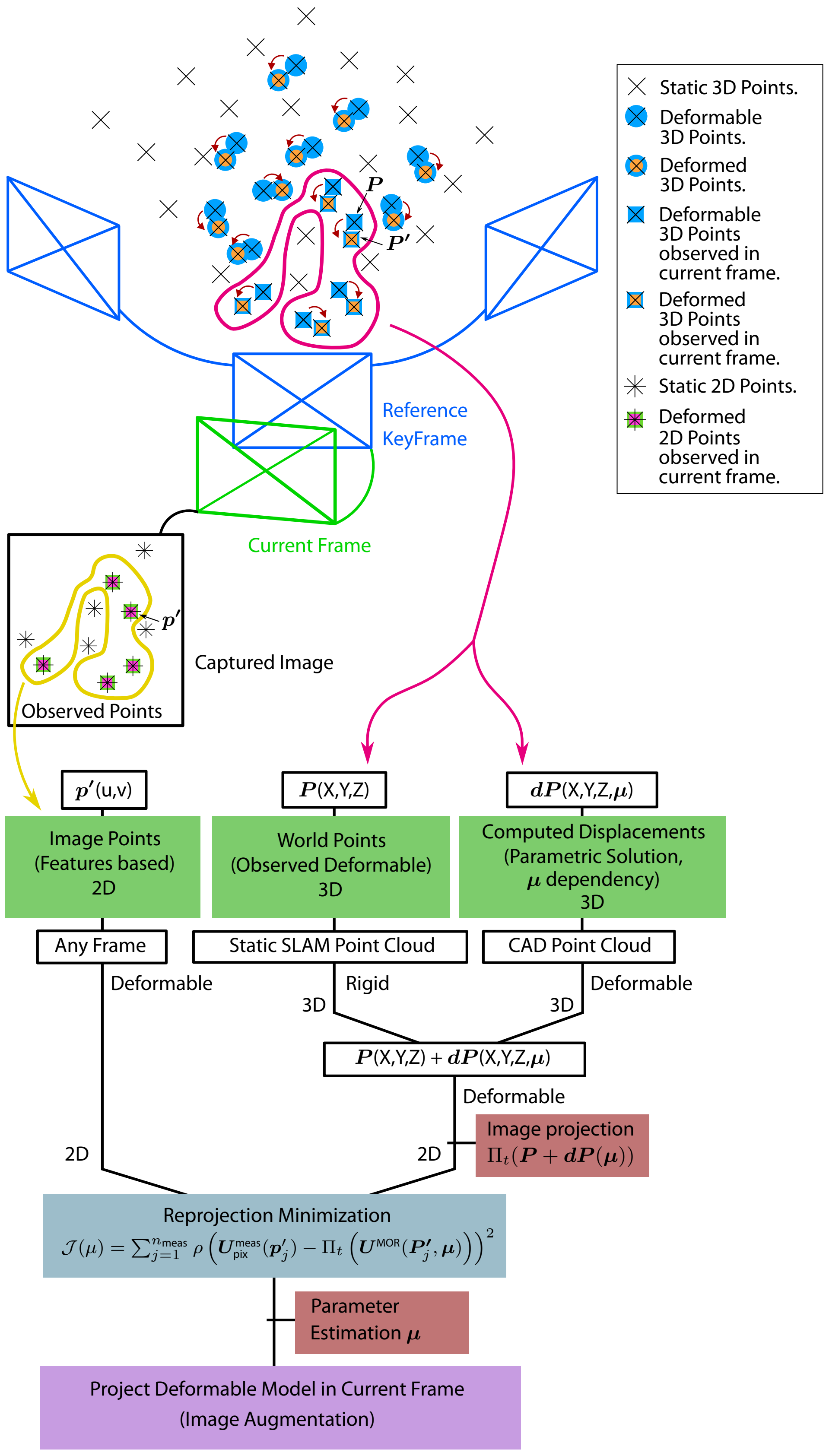

Static 3D Points.
Deformable 3D Points.
Deformed 3D Points.
Deformable 3D Points observed in current frame.
Deformed 3D Points observed in current frame.
Static 2D Points.
Deformed 2D Points observed in current frame.
P
P'
Reference KeyFrame
Current Frame
p'
Observed Points
Captured Image
p'(u,v)
P(X,Y,Z)
dP(X,Y,Z,μ)
Image Points (Features based) 2D
World Points (Observed Deformable) 3D
Computed Displacements (Parametric Solution, μ dependency) 3D
Any Frame
Static SLAM Point Cloud
CAD Point Cloud
Deformable
Rigid
3D
Deformable
3D
P(X,Y,Z) + dP(X,Y,Z,μ)
Deformable
Image projection $\Pi_t(P + dP(\mu))$
2D
2D
Reprojection Minimization
$\mathcal{J}(\mu) = \sum_{j=1}^{n_{\text{meas}}} \rho \left(U_{\text{pix}}^{\text{meas}}(p'_j) - \Pi_t \left(U^{\text{MOR}}(P'_j, \mu) \right) \right)^2$
Parameter Estimation μ
Project Deformable Model in Current Frame (Image Augmentation)

Figure 4.4: Deformable object tracking and augmentation procedure based in ORB features matching and MOR methods to estimate and restrict the 3D displacements.

for any time instant t where each frame is captured, and using the Huber robust function ρ. It is important to highlight that $\boldsymbol{U}^{\text{MOR}}(\boldsymbol{X}'_j, \boldsymbol{\mu})$ represents the whole solution of our hyperelastic problem in question by storing the position, deformations and stresses of each point, with parametric dependence and represented in the reduced space (MOR). However, to simplify the process, $\boldsymbol{U}$ in the functional defined in Eq. (4.11) only represents the deformed position of each point.

Applying the Levenberg-Marquardt algorithm to estimate the values of the optimal parameters we find that the partial components of the derivative terms of the multiparametric solution projected on the image need to be computed, since we are minimizing the reprojection error (2D). This means that, although we have precomputed the sensitiv-

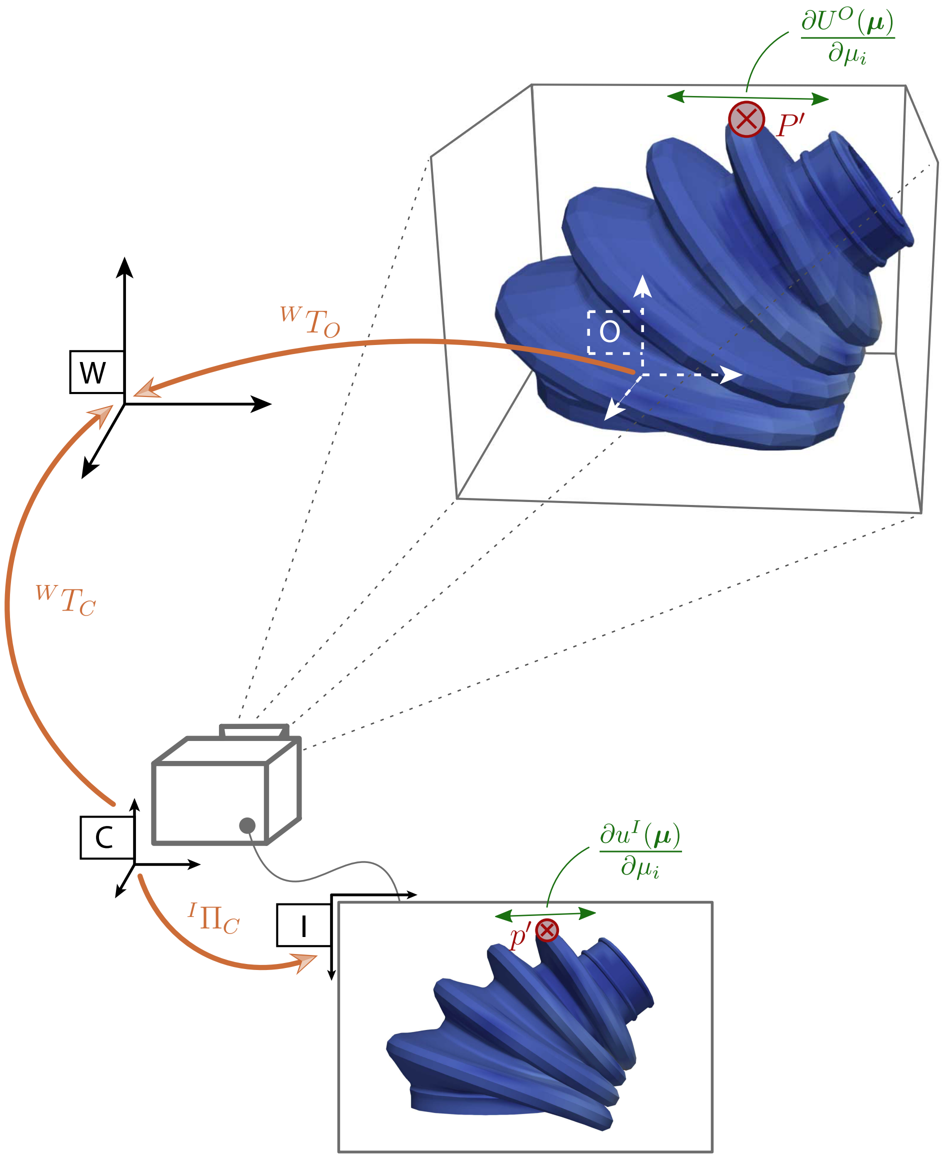

$\frac{\partial U^O(\boldsymbol{\mu})}{\partial \mu_i}$
P'
O
W
${}^W T_O$
${}^W T_C$
C
${}^I \Pi_C$
I
$\frac{\partial u^I(\boldsymbol{\mu})}{\partial \mu_i}$
p'

Figure 4.5: Transformations and projection of the precomputed sensitivities from the original system of coordinates to the image reference.

ities (parametric derivative terms) from the solution in separate variables in the original 3D space, we have to project these sensitivities onto the image plane. A graphic example to understand this process appears in Fig. 4.5, where

$$\frac{\partial \boldsymbol{U}^O(\boldsymbol{\mu})}{\partial \mu_i}$$

refers to the partial derivative term of the multiparametric 3D solution of the CAD object with respect to the parameter μ_i, evaluated at point $\boldsymbol{P'}$ and with the origin of coordinates defined on that object (O), and

$$\frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu})}{\partial \mu_i}$$

represents the same partial derivative projected on the image plane and evaluated on the point $\boldsymbol{p'}$. To obtain the value of this sensitivity in the image plane we use the chain rule

$$\frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu})}{\partial \mu_i} = \frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu})}{\partial \boldsymbol{U}^C(\boldsymbol{\mu})} \cdot \frac{\partial \boldsymbol{U}^C(\boldsymbol{\mu})}{\partial \boldsymbol{U}^W(\boldsymbol{\mu})} \cdot \frac{\partial \boldsymbol{U}^W(\boldsymbol{\mu})}{\partial \boldsymbol{U}^O(\boldsymbol{\mu})} \cdot \frac{\partial \boldsymbol{U}^O(\boldsymbol{\mu})}{\partial \mu_i},$$

being $\boldsymbol{u}^I(\boldsymbol{\mu})$ the position of point $\boldsymbol{p'}$ with respect to the image plane, $\boldsymbol{U}^C(\boldsymbol{\mu})$ with respect to the camera coordinates, $\boldsymbol{U}^W(\boldsymbol{\mu})$ with respect to the global coordinate axis (World) and $\boldsymbol{U}^O(\boldsymbol{\mu})$ with respect to the coordinate axis defined in the object. The solution derivative terms are known in the object reference

$$\frac{\partial \boldsymbol{U}^O(\boldsymbol{\mu})}{\partial \mu_i},$$

and their consecutive transformations in the Euclidean space $\mathbb{R}^3$ are carried out by the matrices ${}^W\boldsymbol{T}_O$ and ${}^W\boldsymbol{T}_C$, which are assumed to be known since they are estimated by the static part of ORB-SLAM2. However, the projection on the image plane ${}^I\boldsymbol{\Pi}_C$ does not allow such a simple treatment, since the camera conical projection used depends on the location of the 3D point with respect to the image plane. This means that the projection of the derivative terms $\mathbb{R}^3 \mapsto \mathbb{R}^2$ must take into account the position of the point where the gradients are evaluated. This projection is known as the image Jacobian, and is expressed as

$${}^I\boldsymbol{J\Pi}_C = \frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu})}{\partial \boldsymbol{U}^C(\boldsymbol{\mu})} = \begin{bmatrix} \frac{\partial p_u}{\partial X_C} & \frac{\partial p_u}{\partial Y_C} & \frac{\partial p_u}{\partial Z_C} \\ \frac{\partial p_v}{\partial X_C} & \frac{\partial p_v}{\partial Y_C} & \frac{\partial p_v}{\partial Z_C} \end{bmatrix} = \begin{bmatrix} \frac{1}{Z_C} f_x & 0 & -\frac{X_C}{Z_C^2} f_x \\ 0 & \frac{1}{Z_C} f_y & -\frac{Y_C}{Z_C^2} f_y \end{bmatrix},$$

where (X_C, Y_C, Z_C) are the 3D coordinates of point $\boldsymbol{P}$ with respect to the camera axes and (p_u, p_v) are the 2D coordinates of the point projected on the image, remembering that

$$p_u = f_x \frac{X_C}{Z_C} + p_{u_0}, \qquad p_v = f_y \frac{Y_C}{Z_C} + p_{v_0}.$$

Finally, the projection of the multiparametric derivative terms with respect to parameter μ_i on the image plane is expressed as

$$\frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu})}{\partial \mu_i} = {}^I\boldsymbol{J}\boldsymbol{\Pi}_C \cdot {}^W\boldsymbol{T}_C{}^{-1} \cdot {}^W\boldsymbol{T}_O \cdot \frac{\partial \boldsymbol{U}^O(\boldsymbol{\mu})}{\partial \mu_i}, \tag{4.12}$$

where we remember that the sensitivities defined with respect to the object axes are interpolated according to the neighbors in the separate variables precomputed solution, being

$$\boldsymbol{U}^O(\boldsymbol{\mu}) = f\Big(\boldsymbol{U}^O_{\text{Neighs,CAD}}(\boldsymbol{\mu})\Big), \qquad \boldsymbol{U}^O_{\text{Neighs,CAD}}(\boldsymbol{\mu}) = \sum_{i=1}^{N} \boldsymbol{F}_{i,1}(\mu_1) \circ \cdots \circ \boldsymbol{F}_{i,\mu_D}(\mu_D).$$

Regarding to the technical aspects, we use the feature search method developed in the ORB-SLAM2 code to track the deformations. As we said, in the minimization of the Eq. (4.11) we use the Huber robust cost function to penalize those outlier observations. Time requirements for the minimization algorithm are much smaller than the computation time of the SLAM tasks, so a simple parallelization of the whole code in a few threads allows us to work in a frequency of 30 fps using the CPU. We stop the *Mapping* tasks and work in *Localization* mode in ORB-SLAM2 once deformations are appearing.

4.5 Examples

To support the work presented in this chapter, we have applied the theoretical concepts on four examples of deformable objects:

- The first example is an aluminium beam with a linear elastic behaviour which is deformed by the application of a load on its upper face. The deformations have been measured using homographies and fiducial markers.
- The second example consists of a beam made of foam, so a hyperelastic material law has been applied. Homographies have again been used, although the use of markers has been reduced, making use of the own texture of the beam for the tracking task.
- In the third example we simulate the deformations of a constant velocity joint boot, made of rubber material, so again we use a hyperelastic behavior. The contact of the folds of the piece itself has also been modelled. However, the objective of this example is the addition of the virtual piece on our real world, without the real piece being present. Our purpose is the creation of a framework that allows the design and visualization of the behavior of any piece through mixed reality. Thus, we have used ORB-SLAM2 to locate the static environment on which to add the virtual piece, and by tracking real time tracking the position of a bar that contacts with the rubber piece we are able to deform the virtual piece.

- In the fourth example the same piece made in rubber has been used, with the same material law, but in this case we do have the real piece. This last example has served to demonstrate the deformable implementation that we developed on ORB-SLAM2, where we track each point of the deformable object to show some interesting physical information of the process, such as the stress field in any point of the piece.

4.5.1 Aluminum beam linear elastic employing fiducial markers

The first example is a linear elastic cantilever beam with a vertical load applied in the upper face at a variable position s, as Fig. 4.6 shows.

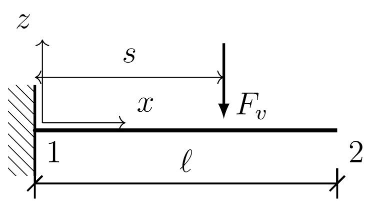


Figure 4.6: Cantilever beam with a moving load F_v that is parameterized through its position coordinate s.

The beam is made of aluminum with Young's modulus $E = 69$ GPa and Poisson ratio $\nu = 0.334$. The experiments were conducted in the linear elastic regime and the displacement field was assumed to have a form

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, s),$$

where $\boldsymbol{x} \in \Omega$ represents the coordinates of the considered point and s the position of the applied load along the upper face of the beam, see Fig. (4.6). Following the standard PGD method, an approximation of the parametric displacement field of the type

$$\boldsymbol{u}(\boldsymbol{x}, s) \approx \sum_{i=1}^{n} \boldsymbol{F}_i(\boldsymbol{x})\boldsymbol{G_i}(s)$$

was computed. In this case the classical elasticity problem is assumed to govern the physics. Its weak form consists in finding the displacement $\boldsymbol{u} \in \mathcal{H}^1(\Omega) \times L_2(\Gamma_{t2})$ such that for all $\boldsymbol{u}^* \in \mathcal{H}_0^1 \times L_2(\Gamma_{t2})$:

$$\int_\Gamma \int_\Omega \boldsymbol{\nabla}_s \boldsymbol{u}^* : \boldsymbol{\sigma} d\Omega d\Gamma = \int_\Gamma \int_{\Gamma_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} d\Gamma d\Gamma$$

where $\Gamma = \Gamma_u \cup \Gamma_t$ represents the boundary of the beam, divided into essential and natural regions, and where $\Gamma_t = \Gamma_{t1} \cup \Gamma_{t2}$, i.e., regions of homogeneous and non-homogeneous, respectively, natural boundary conditions. Here, $\boldsymbol{t} = P \cdot \delta(x - s)\boldsymbol{e}_k$, where δ represents the Dirac-delta function and $\boldsymbol{e}_k$ the unit vector along the z-coordinate axis (we consider here, for the ease of exposition, a load P directed towards the negative z axis of reference).

Once regularized, the Dirac-delta term is approximated by a truncated series of separable functions in the spirit of the PGD method, i.e.,

$$t_j \approx \sum_{i=1}^{m} f_j^i(\boldsymbol{x}) g_j^i(\boldsymbol{s}),$$

where m represents the order of truncation and f_j^i, g_j^i represent the j-th component of vectorial functions in space and boundary position, respectively. For this particular example, only $j = k$ (i.e., the third component of the vector) is assumed to exist.

In this example, fiducial markers were employed to help locate the deformed configuration of the beam. See Fig. 4.7 for a picture of the experimental device.

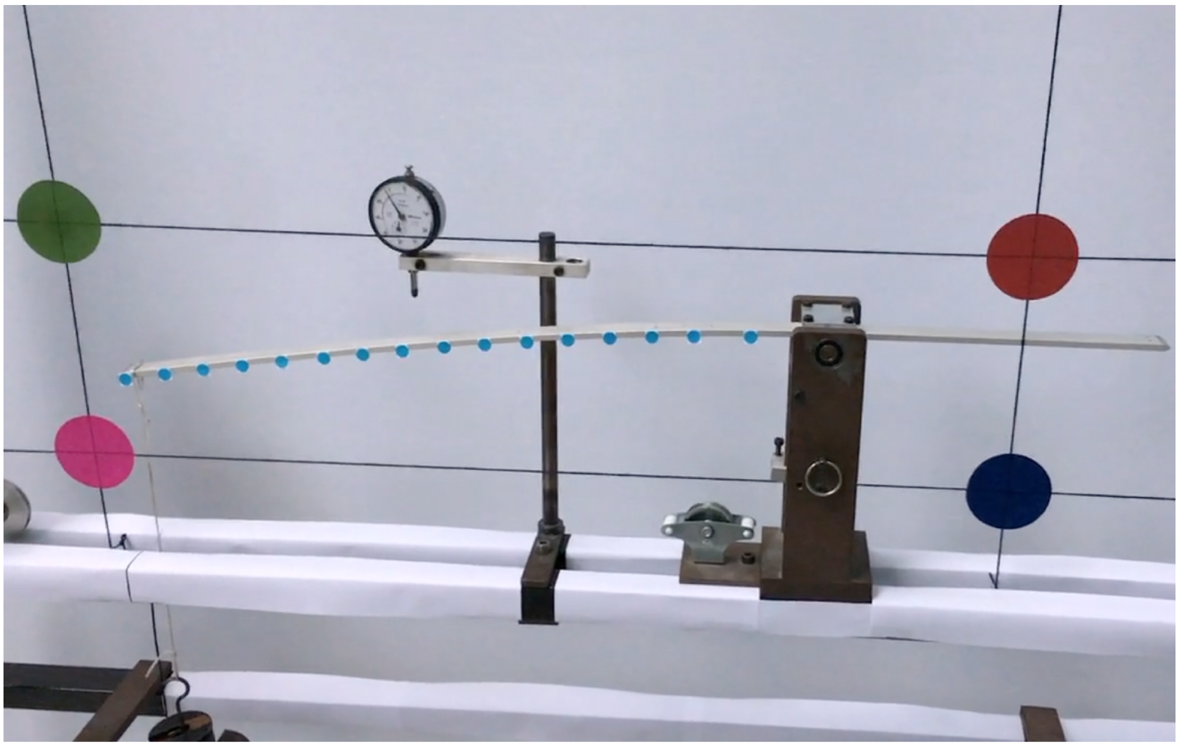


Figure 4.7: Experimental device for the aluminum beam example. Note the fiducial markers for the location of the beam and camera pose determination (big four circles) and the markers (small blue dots adhered to the beam) for the determination of the beam deflection.

The use of fiducial markers is particularly advantageous for some industrial applications (structural health monitoring, for instance), for which there is no limitation to the presence of small adhered paper or plastic pieces, for instance. If this is not possible —for aesthetic reasons, for instance, or just because we have no a priori access to the studied solid— feature-based approaches must be preferred. These will be analyzed in Sec. 4.5.3 below.

The number of modes n employed in this example was 19, even if for this problem much less are enough —see the big relative decay in modulus of the modes—. The first four spatial modes are represented in Fig. 4.8, while modes on s are depicted in Fig. 4.9.

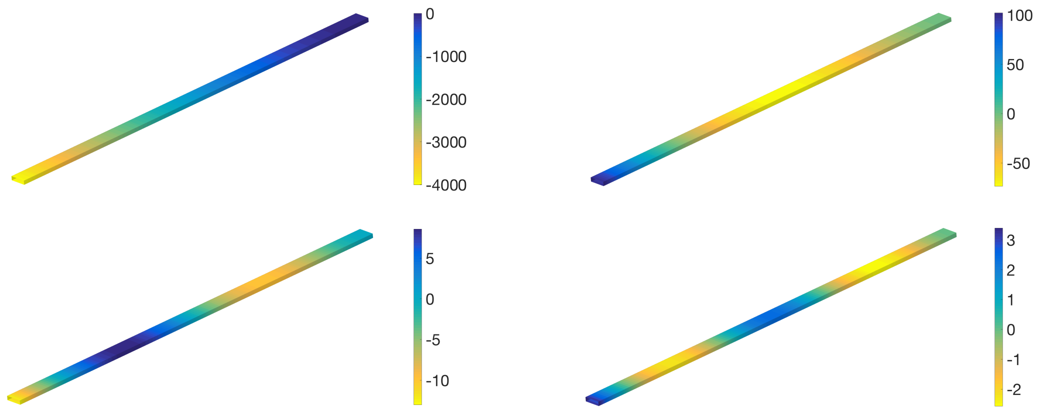


Figure 4.8: Modes $\boldsymbol{F}_i(\boldsymbol{x}), i = 1, 2, 3, 4$ for the aluminum beam example. These are vectorial modes. The colour map corresponds to the vertical displacement, the most relevant one.

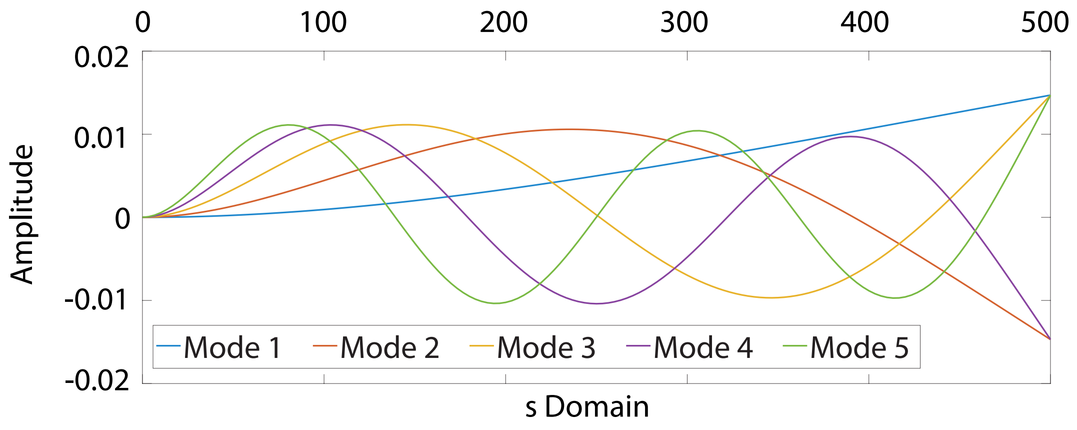


Figure 4.9: Modes $G_i(s), i = 1, \ldots, 5$, for the aluminum beam problem.

Results of the real-time assimilation procedure are reported in Fig. 4.10, where different snapshots of the raw video, assimilated load position and augmented video are reported. Quantitative results are included in Table 4.1. Note that the reported errors are absolute and represent the difference between the assimilated load position and the actual one. Except for load positions very close to the clamped support, that produce almost no deflection in the beam, and therefore a large error in the assimilation procedure, errors are less than 1% for most positions (for instance, 0.3% error for the load at $s = 460$ mm, or 0.1% for $s = 420$ mm). These values can be improved by employing more PGD modes, on one side, but are strongly limited by the camera's resolution and the proximity of the beam to the objective. In other words, it is the portion of the solid covered by a single pixel that usually determines the level of precision.

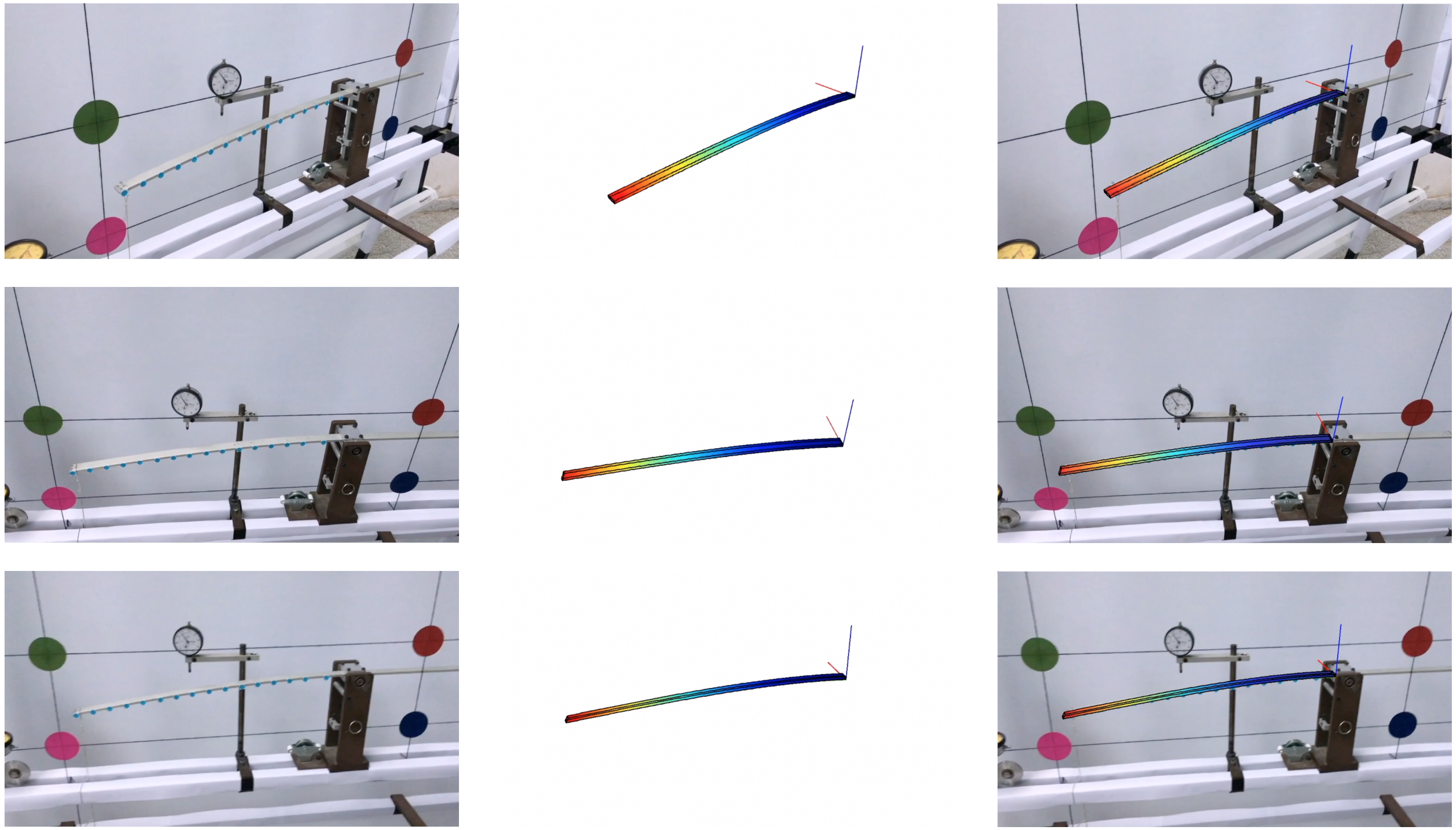


Figure 4.10: Three different snapshots of the raw video (left column), simulation (center) and augmented video (right). The colour map corresponds to the modulus of the displacement field.

Actual Position s (mm)	Mean Estimated Pos. (mm)	Difference abs. value s (mm)	Actual Position (mm)	Mean Estimated Pos. (mm)	Difference abs. value (mm)
0	0.14	0.14	260	262.21	2.21
20	7.98	12.02	280	281.10	1.10
40	26.64	13.36	300	301.30	1.30
60	57.66	2.34	320	320.44	0.44
80	82.09	2.09	340	338.33	1.67
100	102.54	2.54	360	358.57	1.43
120	122.67	2.67	380	383.49	3.49
140	143.89	3.89	400	401.35	1.35
160	164.55	4.55	420	420.70	0.70
180	182.25	2.25	440	440.27	0.27
200	204.27	4.27	460	458.40	1.60
220	222.75	2.75	480	477.69	2.31
240	242.22	2.22			

Table 4.1: Load positioning absolute error in the cantilever beam problem.

The resulting video can be consulted at `https://youtu.be/BK0sHfvjITo`. In it, the robustness of the proposed technique to the partial occlusion of some of the fiducial markers or even a total occlusion of the camera are tested. The system resulted to be very robust and recovered immediately the load position without any problem.

4.5.2 Hyperelastic beam tracking the texture

A $797.5 \times 200 \times 105\ mm^3$ foam beam was considered in this example. The beam was meshed from a picture of a deformed configuration under self-weight load, see Fig. 4.11. An experimental campaign was accomplished so as to determine that its (homogenized) constitutive law was assimilated to a Kirchhoff-Saint Venant law with $E = 0.11\ N/mm^2$ and Poisson coefficient $\nu = 0.2$. The Kirchhoff-Saint Venant model is characterized by the energy density functional given by

$$\Psi = \frac{\lambda}{2}(\text{tr}(\boldsymbol{E}))^2 + \mu \boldsymbol{E} : \boldsymbol{E},$$

where λ and μ are Lame's constants and $\boldsymbol{E}$ the Green-Lagrange strain tensor. In this case, the weak form of the problem takes the form

$$\int_{\Gamma}\int_{\Omega(t)} \boldsymbol{E}^* : \mathbf{C} : \boldsymbol{E} d\Omega d\Gamma = \int_{\Gamma}\int_{\Gamma_{t2}} \boldsymbol{u}^* \cdot \boldsymbol{t} d\Gamma d\Gamma,$$

where $\mathbf{C}$ represents the fourth-order constitutive tensor. Given the non-linear strain measure $\boldsymbol{E}$, the resulting problem is inherently non-linear.

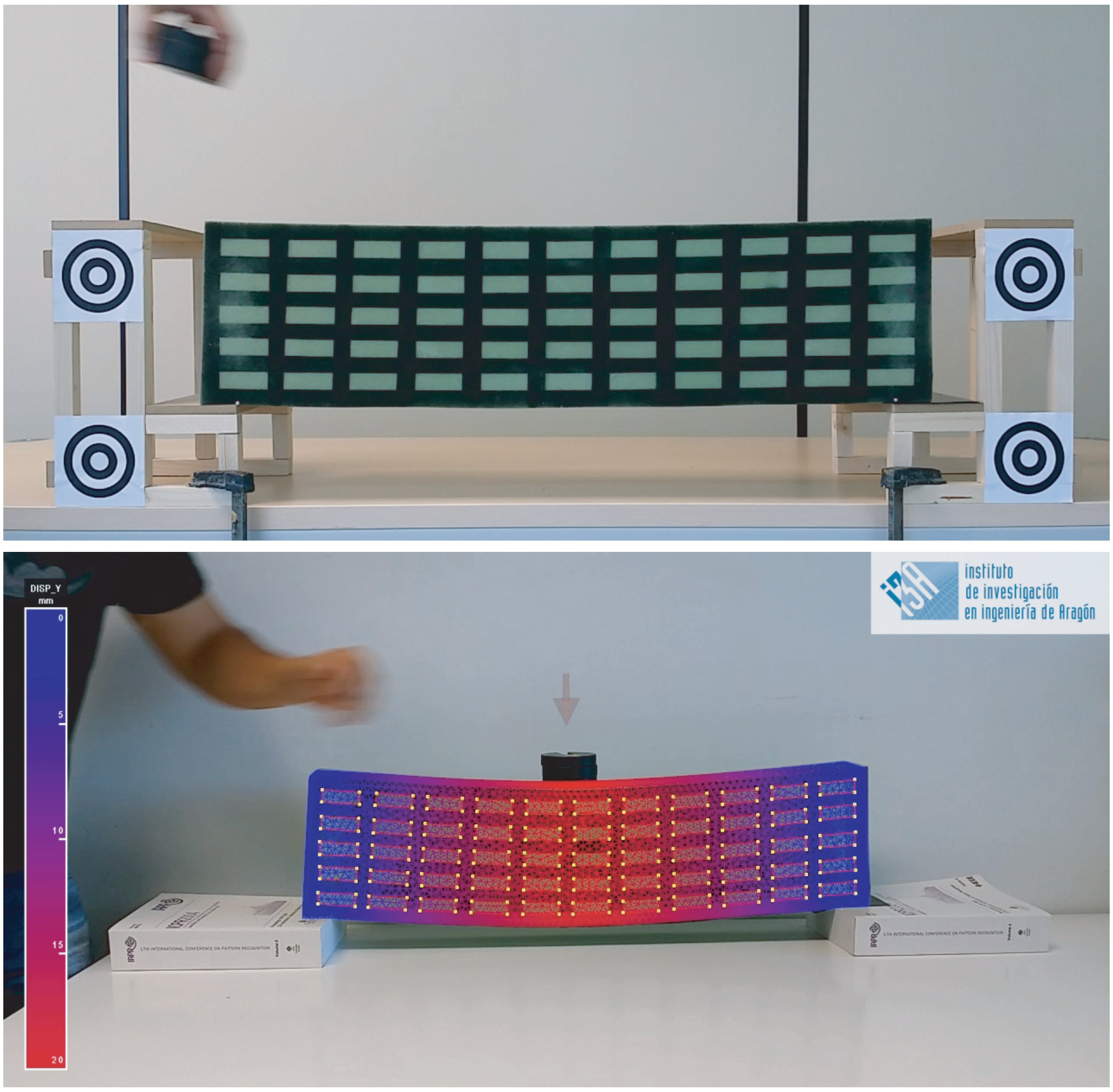


Figure 4.11: Hyperelastic beam experiment. Top image was taken as the reference configuration of the beam. Bottom image shows a frame from the resulting video (second 55) in which we appreciate the assimilated position of the load (see the arrow). Superimposed, the displacement field of the beam. Note that no fiducial marker is employed at this stage.

Despite the well-known limitations of the Kirchoff-Saint Venant law under compressive stresses, this law showed to fit much better than usual hyperelastic laws (neo-Hookean, Mooney-Rivlin) to the experiments. In fact, some mild viscoelastic effects were observed, that are responsible of some oscillations in the results —see the accompanying video—, but these were nevertheless neglected. For short-term periods, results showed to be in good accordance to this law, as will be detailed hereafter.

A mesh of 18938 nodes and 98863 linear tetrahedral elements was created. Boundary conditions were assimilated to a pinned support (foam-wood contact surface showed to be nearly an adhesion) on the leftmost $40\,mm$. On the right, a sliding support was accomplished by means of plastic laminates. $15\,N$ loads were applied by means of lead disks with $80\,mm$ diameter, whose size was considered in detail in the simulation (due to the weak stiffness of the foam, punctual loads provoked very poor results by greatly deforming the upper surface of the foam).

The goal of this experiment was to develop augmented reality videos under real-time constraints (no batch processing was allowed) in which the position of the load was determined and the displacement field was plotted on top of the actual geometry of the beam, so as to indicate the user its true magnitude. Therefore, the displacement field was —as in Section 4.5.1— assumed to have a parametric form

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, s),$$

where $\boldsymbol{x} \in \Omega$ represents the coordinates of the considered point and s the position of the applied load along the upper face of the beam. A PGD approximation of this parametric displacement field of the type

$$\boldsymbol{u}(\boldsymbol{x}, s) \approx \sum_{i=1}^{n} \boldsymbol{F}_i(\boldsymbol{x}) G_i(s)$$

was constructed off-line. $n = 19$ modes were considered in this approximation, although some experiments with only 11 modes provided results with very similar levels of accuracy. The load position coordinate, s was assumed to be one-dimensional, since the load was placed centered on the upper surface of the beam. The mesh along this dimension comprised 60 nodes in the central part of the beam (a portion of $100\,mm$ of the beam at the right and left ends was not considered, since the displacements provoked by the load in these regions was not perceptible, due to the absence of bending). A detailed derivation of the necessary linearization of the Kirchhoff-Saint Venant law under the PGD framework was accomplished in [Niroomandi et al., 2013]. The reader is referred to this reference or to [Cueto et al., 2016] for a complete Matlab implementation.

Fig. 4.12 shows some modes $G_i(s)$ of the model, in particular, $i = 1, 2, 3$ and 11 and Fig. 4.13 shows the corresponding modes in space $\boldsymbol{F}_i(\boldsymbol{x})$. On the other hand, Fig. 4.11 shows a frame from a real sequence in which we appreciate the accuracy of the assimilated displacement field and, notably, the location of the load position, marked with a red arrow.

The video was recorded at a resolution of 1280×720 p. (which is the resolution of an iPhone 6s). Under these circumstances, the proposed method is able to provide a response at 60 Hz. The mean error is on the order of 1.5 mm. This value is also dependent on the relative proximity of the deformable object (the foam beam) to the camera, of course. In other words, to the actual, physical, displacement registered by each pixel in the image. This error could therefore be improved by augmenting the resolution of the camera, by taking a closer view or, once we have tried any of the previous possibilities, by increasing the number of modes in the approximation of the displacement field.

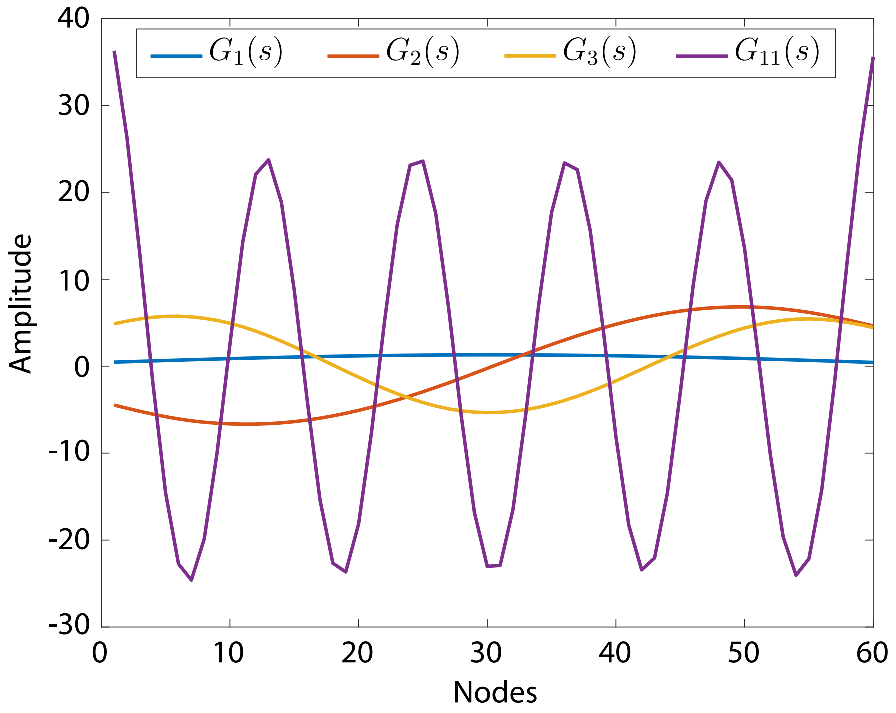


Figure 4.12: Modes 1, 2, 3 and 11 on s, the position of the load. Note that, as it is the case very often with PGD, the modes increase in frequency, the first one having the lowest frequency content.

The resulting video has been made public at `https://www.youtube.com/watch?v=byhXy8aJ1J8`.

4.5.3 Addition of a Virtual Joint Boot

We consider in this example the process of design and analysis of an automotive boot sealing made of a neo-Hookean rubber. The geometry of the seal is shown in Fig. 4.14. The model is composed by a mesh of 8640 nodes and 5940 linear hexahedral elements. The rubber is assumed to follow an incompressible law

$$W = C_1(I_1 - 3),$$

where W represents the strain energy density function, $C_1 = 1166$ MPa for this particular example, and I_1 represents the first invariant of the right Cauchy-Green strain tensor. The material is thus considered incompressible. The lever is assumed to be rigid, and rotates around a pinned support in the plane where the boot seal lies.

In this particular example the boot seal does not exist physically in the video. Instead, this example opens the possibility of employing augmented reality for engineering design purposes, by showing the performance and appearance that the virtual object could have in the physical world.

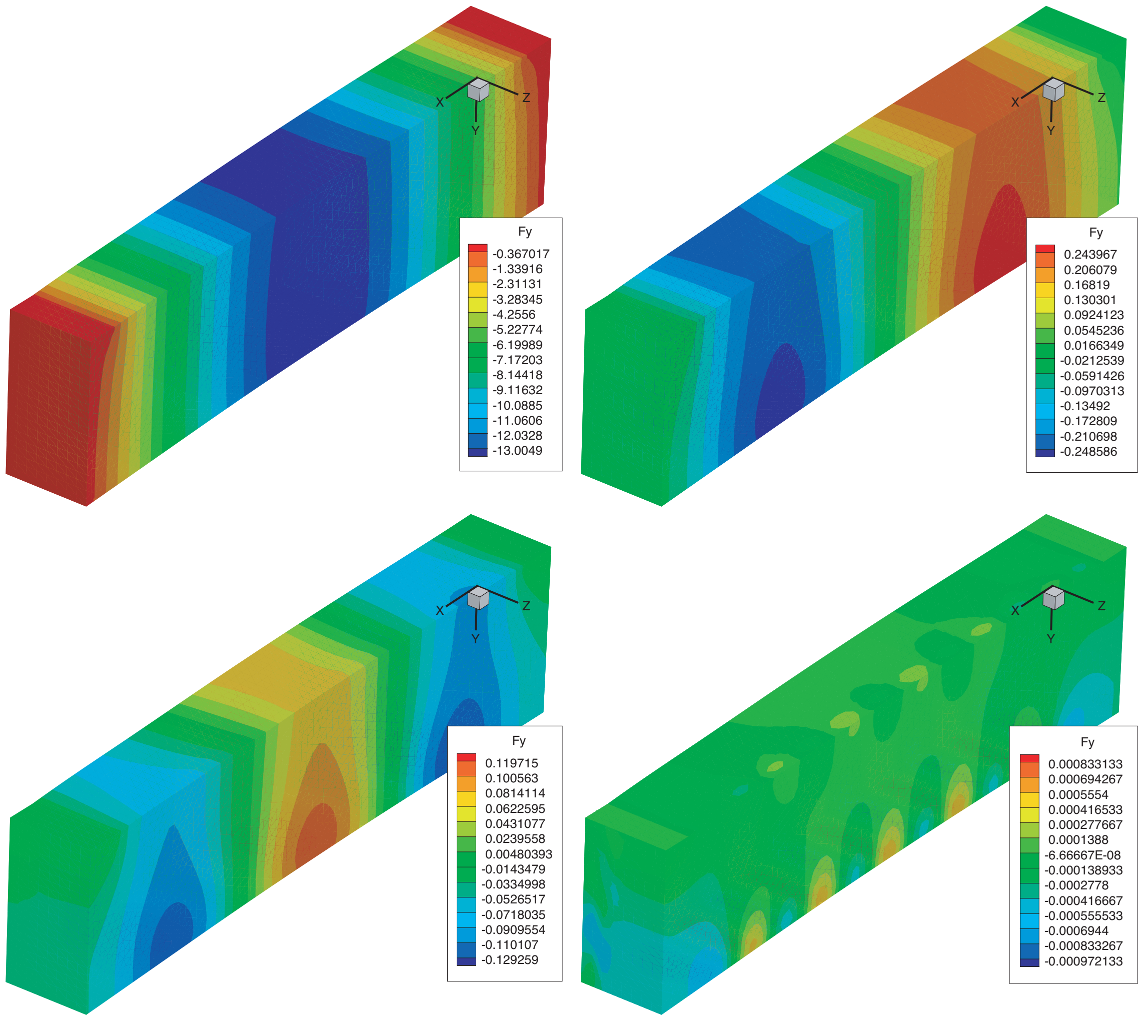


Figure 4.13: Modes $\boldsymbol{F}_i(\boldsymbol{x})$, $i = 1, 2, 3$ and 11 for the foam beam example. Again, modes increase in frequency as i increases. This a typical PGD result, although in some, very rare cases, modes show a somewhat more irregular pattern, see [Niroomandi et al., 2010], for instance.

In this particular example, the displacement field of the boot seal is parameterized by the angle of rotation of the lever, θ. As in the previous examples, we hypothesize that this dependence can be affinely approximated as

$$\boldsymbol{u}(\boldsymbol{x}, \theta) \approx \sum_{i=1}^{\mathrm{n}} \boldsymbol{F}_i(\boldsymbol{x}) G_i(\theta). \tag{4.13}$$

Construction of the reduced order model

For this particular example, the so-called non-intrusive PGD method has been employed [Zou et al., 2018]. In particular, we have employed the formulation introduced in [Borzacchiello et al., 2017], that employs adaptive samplings of the parametric domain and col-

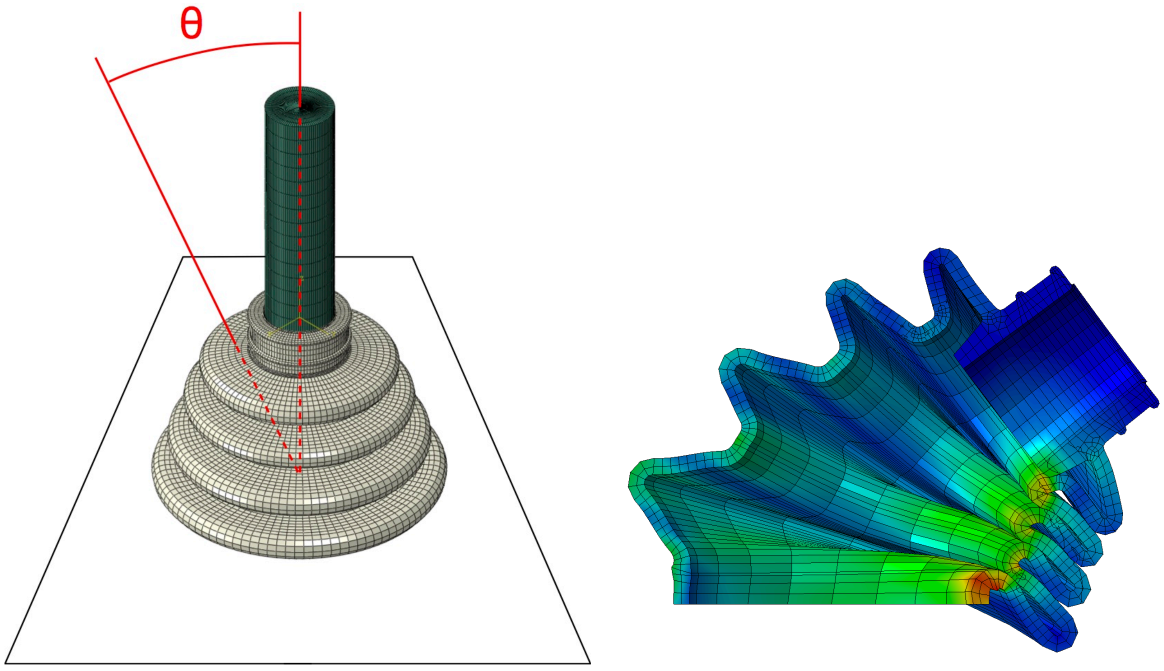


Figure 4.14: Geometry of the (rubbery) boot seal (left) and a snapshot during the process (right). The angle θ is the fundamental magnitude to be determined by the computer vision system, since the boot seal is entirely virtual. It does not exist in the physical reality.

location approaches. We thus avoid complex linearization procedures while employing off-the-shelf commercial softwares to construct the sought PGD approximation, Eq. (4.13).

The parametric space $\theta \in [0^\circ, 52^\circ]$ —here, symmetry has been employed— has been discretized into 193 finite elements. Probably much less degrees of freedom could be necessary but, given the inherent non-linearity of the problem due to the presence of contact phenomena, a very fine discretization was employed that nevertheless did not prevent the method to reach the real-time constraints.

Segmentation and registration

In this example, as already mentioned repeatedly, there is no physical boot seal. Therefore, the objective of the computer vision implementation is to determine the degree of rotation θ of the lever. To that end, no fiducial markers have been employed. In the absence of that markers, computer vision systems employ *feature matching* algorithms, capable of detecting and tracking local frame features. In this work we employ ORB feature detectors [Rublee et al., 2011]. These are the green markers in Fig. 4.15 (right). Within SLAM algorithms, ORB detectors provide with a very efficient means of unveiling the environment in which we move [Mur-Artal et al., 2015].

SLAM methods were primarily designed for robot navigation in rigid environments. But in this particular example the objective is to detect the lever tilt angle θ. In other words:

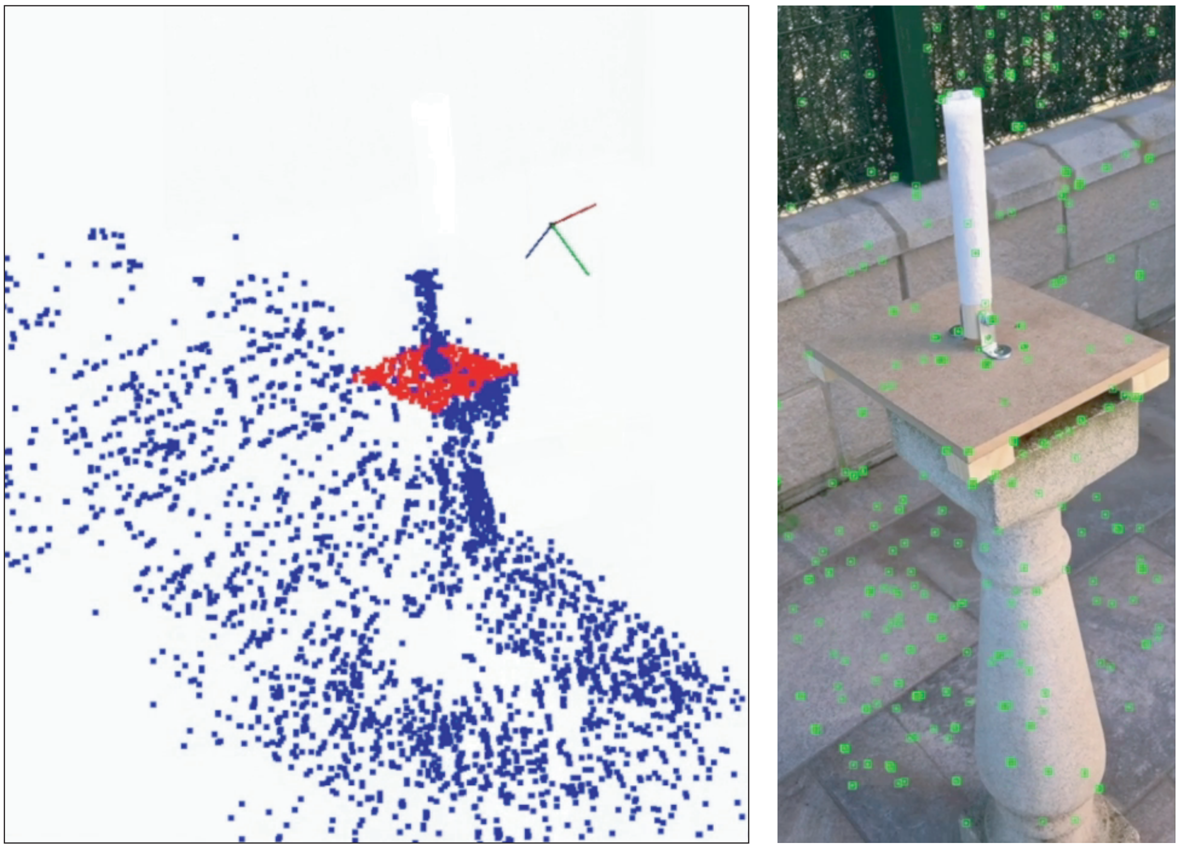


Figure 4.15: Left: 3D pointcloud from the real sequence in the rubber piece example with RANSAC plane detection (in red). Right: Raw video with the detected features in green. Note that there is no boot seal in the physical reality.

there is some part of the scene that is not rigid. To detect θ an algorithm is proposed in which we proceed by parts. First of all, it is necessary to detect the plate in which the lever is mounted. To that end, we employ Random Sample Consensus (RANSAC) methods [Fischler and Bolles, 1981]. RANSAC methods allow to fit a model from experimental data in the presence of a big number of outliers. This is precisely our case: a significant portion of the ORB features lie on a plane (precisely the plate in which the lever is mounted), while the rest of the detected features in the environment are considered outliers. The resulting detection is shown in Fig. 4.15.left, where the detected plate is depicted by means of red features, while the environment is depicted in blue.

Once the plate has been detected, the tilting angle of the lever is found by filtering the white color of the lever from the images. This allows for a very precise determination of the angle θ, see Fig. 4.16.

Performance

The resulting video from which frames in Fig. 4.16 have been extracted can be seen at `https://youtu.be/0whA92vFk1g`. In it, a robustness test is made for the proposed me-

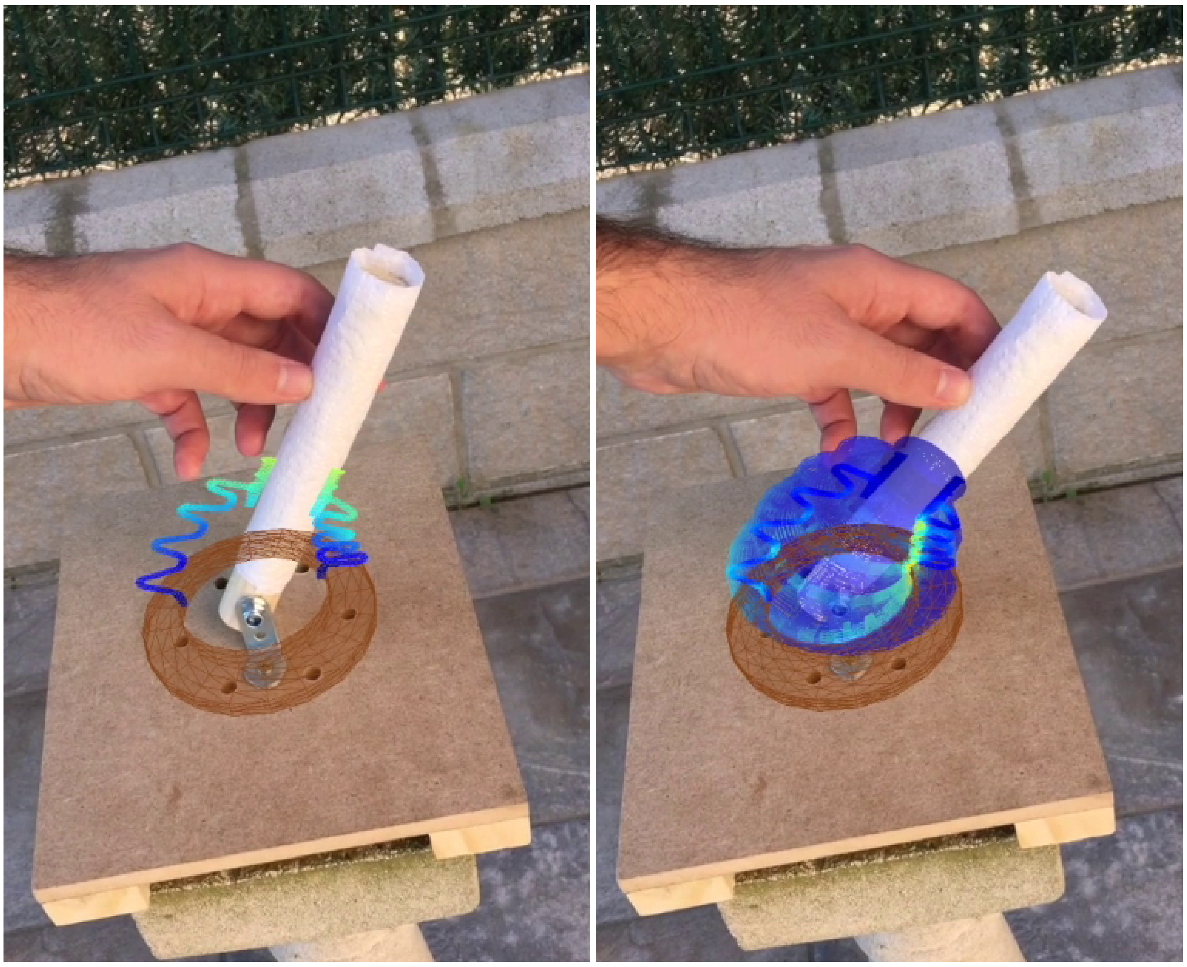


Figure 4.16: Two different augmented frames of the video. Color map corresponds to the modulus of the displacement field at the plane of symmetry (left) and von Mises stresses (right).

thod in which the camera is moved away of the lever position, so that it disappears from the image. The localization capabilities of ORB-SLAM methods are able, nevertheless, to find the lever back with great precision, thus closing the loop and restarting the augmentation with no problems.

4.5.4 Tracking of a Real Joint Boot

The last example shows the behavior of the rubber boot seal, the part that protects the gearshift of a car (Fig. 4.17). The object is deformed forced by the rotation of the gearshift in one direction, from $-52°$ to $52°$, but the simulation has been only applied inside the parametric range $\theta \in [0°, 52°]$, taking advantage of revolution symmetry. In last example, only one parameter was minimized (θ), as the platform under the object was assumed known and so the rotation axis orientation (α_Z). But in the current example, we assume the object is placed in any point of the space and we have no information about the rotation axis orientation, so both parameters $\boldsymbol{\mu} = (\theta, \alpha_Z)$ need to be estimated.

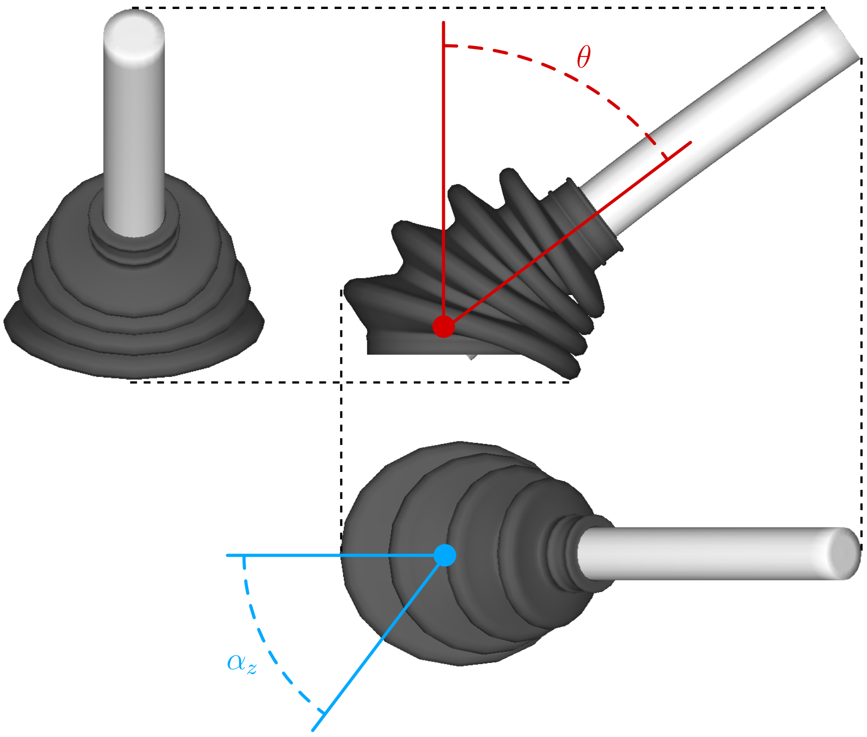


Figure 4.17: Different views of the rubber boot seal with a simplified gear shift lever. θ and α_z are the parameters to minimize based in the estimated deformation of the object.

The material behavior applied to the rubber is again a non-linear law, called Neo--Hookean model, with the strain energy density

$$W = \frac{\mu}{2}(I_1 - 3) - \mu \ln(J) + \frac{\lambda}{2}(\ln(J))^2$$

being I_1 the first invariant of the right Cauchy-Green deformation tensor

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$$

where λ_i are the eigenvalues. The gear shift lever has been modeled as a linear elastic material. The complexity of this problem, as it happened in the previous example, does not allow us to compute the object deformations in real time. But in the current example we have to impose more restrictions, as we need to take into account the self-contact of the walls of the rubber and the contact of the rubber and lever. Contact imposition is a very slow task, as the solver needs to check if two surfaces contact each other, and then go back to preserve non-insertion restrictions. But, another time, the use of MOR methods allows us to work in real time.

We consider two types of parameters: those that can vary in each frame, and others that remain constant throughout the entire sequence. An example of the first can be the

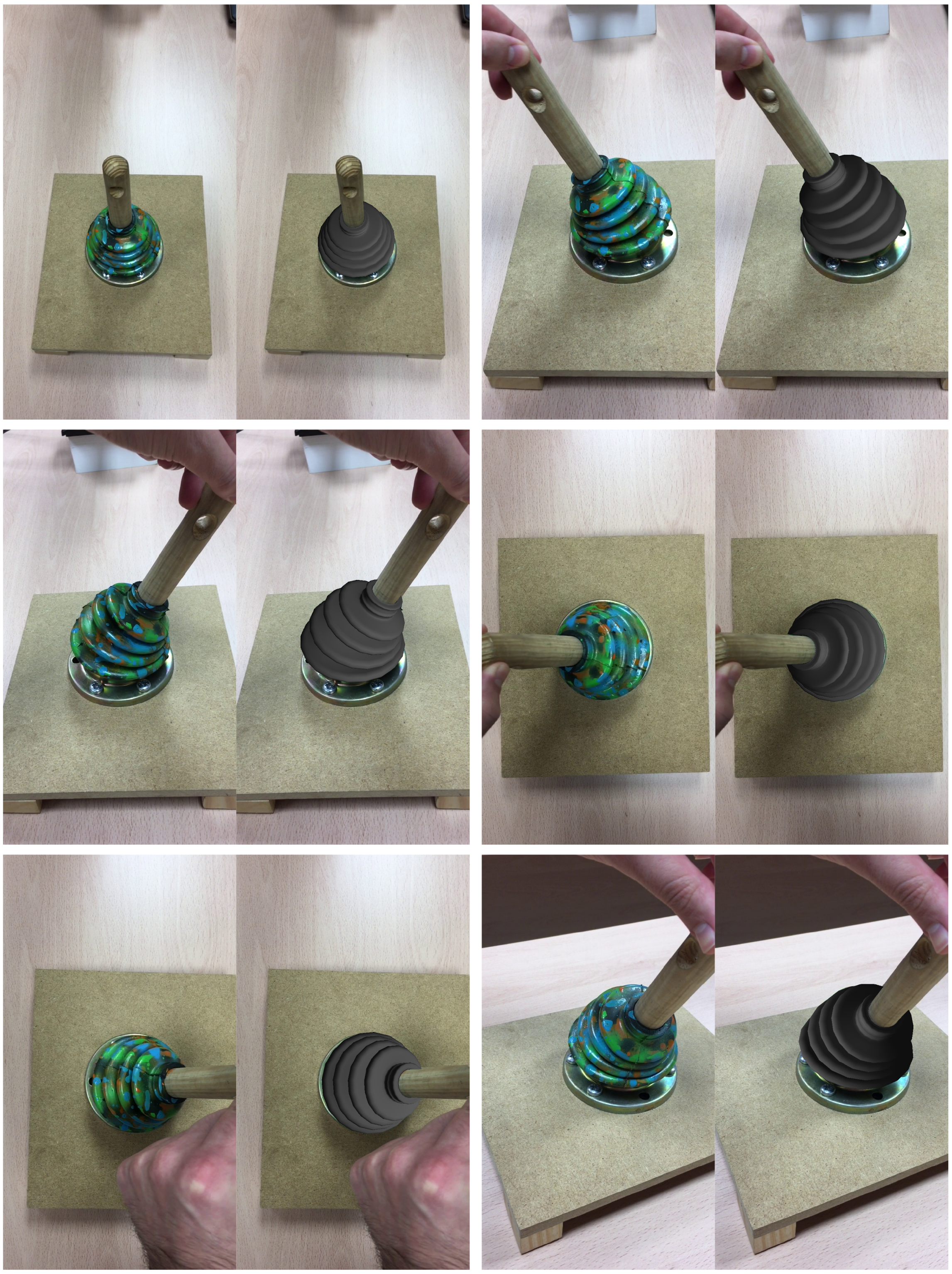


Figure 4.18: Some snapshots showing the original and augmented frames.

position, module or direction of the applied loads, that in our case is parameter θ. It can take a different value in any frame, depending on the interaction with the user. On the other hand, the parameters that define the material behavior, or the boundary conditions are examples of the second type, which are generally considered constant throughout the whole simulation. An example of the second type of parameters in this case is α_Z, and once it is estimated, it remains constant for the complete sequence.

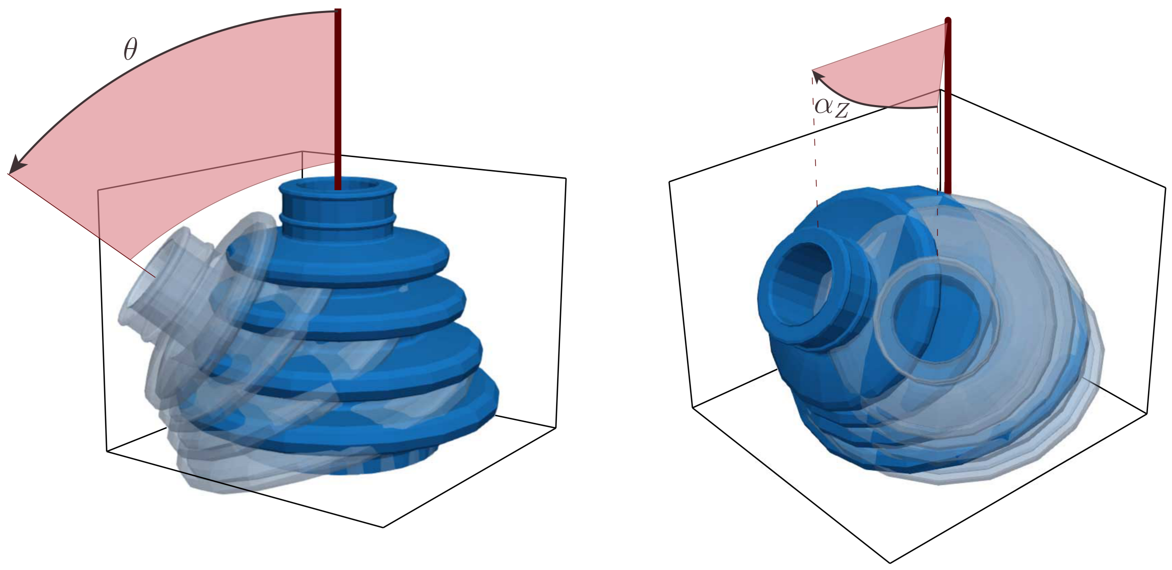


Figure 4.19: Example of variation of the parameter values of the model. The minimization algorithm must find the optimal values in each frame (left) and along the whole sequence (right).

We created a system that can be applied to any object regardless of its texture. In other words, we assume that we have an object with a defined geometry, behavior and conditions, but whose exterior texture has not been previously scanned. We identify the objects using a geometric registration, independently of their external aspect, although later we use their textures to track the deformations. This allows us to simulate the behavior of multiple equal objects, but provides the disadvantage that we do not know the boundary conditions, such as parameter α_Z in this case, and that is why we must include it in the minimization. Since we assume that parameter α_Z is constant throughout the simulation, we only need a few frames where the object is deformed to get the optimal value of the parameter.

The estimation of the parameter values is carried out using the Levenberg-Marquardt algorithm, applied to the functional defined in Eq. (4.11). The transformation of the solution derivative terms with respect to parameter θ, from the object coordinate system to the image, is carried out as indicated in Eq. (4.12). In the case of parameter θ_Z, we can define the partial derivative term of the solution as

$$\frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu}, \alpha_Z)}{\partial \alpha_Z} = \frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu}, \alpha_Z)}{\partial \boldsymbol{U}^C(\boldsymbol{\mu}, \alpha_Z)} \cdot \frac{\partial \boldsymbol{U}^C(\boldsymbol{\mu}, \alpha_Z)}{\partial \boldsymbol{U}^W(\boldsymbol{\mu}, \alpha_Z)} \cdot \frac{\partial \boldsymbol{U}^W(\boldsymbol{\mu}, \alpha_Z)}{\partial \boldsymbol{U}^O(\boldsymbol{\mu}, \alpha_Z)} \cdot \frac{\partial \boldsymbol{U}^O(\boldsymbol{\mu}, \alpha_Z)}{\partial \alpha_Z}.$$

As it is a geometric transformation, independent to the object deformations, so it does not depend on the separate variables solution of our problem, we can take this term out from the projected solution, largely facilitating the whole process. In addition, since it is a geometric transformation defined in the rotation group $SO(3)$, its derivative term can be quickly estimated. Finally, the solution sensitivity with respect to parameter α_Z can be expressed as

$$\frac{\partial \boldsymbol{u}^I(\boldsymbol{\mu}, \alpha_Z)}{\partial \alpha_Z} = {}^C\boldsymbol{J\Pi}_I \cdot {}^W\boldsymbol{T}_C \cdot {}^W\boldsymbol{T}_O{}^{-1} \cdot \frac{\partial \boldsymbol{T}_{\alpha_Z}}{\partial \alpha_Z} \cdot \boldsymbol{U}^O(\boldsymbol{\mu}).$$

Some images extracted from a video sequence, showing the original frame and the augmented frame, can be observed in Fig. 4.18. The virtual object deforms as the real object in real time. But, as we said, the goal is to provide information that the user can not perceive, so in Fig. 4.20 we show more images where the virtual object plots the stresses, that are obviously changing as the object deforms.

4.6 Chapter Conclusions

In this chapter a new method for the physically realistic augmentation of video sequences is proposed using numerical simulations. The method, given the stringent constraints imposed by actual video rates (usually 30 or 60 frames per second) is based on the employ of reduced order modeling and simulation. In particular, the affine parametric approximation of most reduced order techniques is exploited for a very efficient data assimilation procedure, that enables us to obtain a very accurate description of the objects.

The proposed method is valid for any augmented reality application in which there is interest about hidden information in the video, where the behavior is governed by a PDE (such as stress, for instance), but also to create and augment with non present objects the physical reality. In any case, the state-of-the-art computer vision techniques have been employed to detect the camera pose and to map the static environment. Four different examples have been analyzed in which different approaches to the SLAM problem have been applied. In the first of them we employed fiducial markers, a very versatile possibility in industrial settings, where we can identify the object of interest with such markers. In the second example we employed some existing pattern in the solid so as to track deformations in it. And finally in the third and fourth examples, feature-based approaches were applied (in particular, ORB-SLAM) to locate the solid, track its deformation and estimate the camera pose.

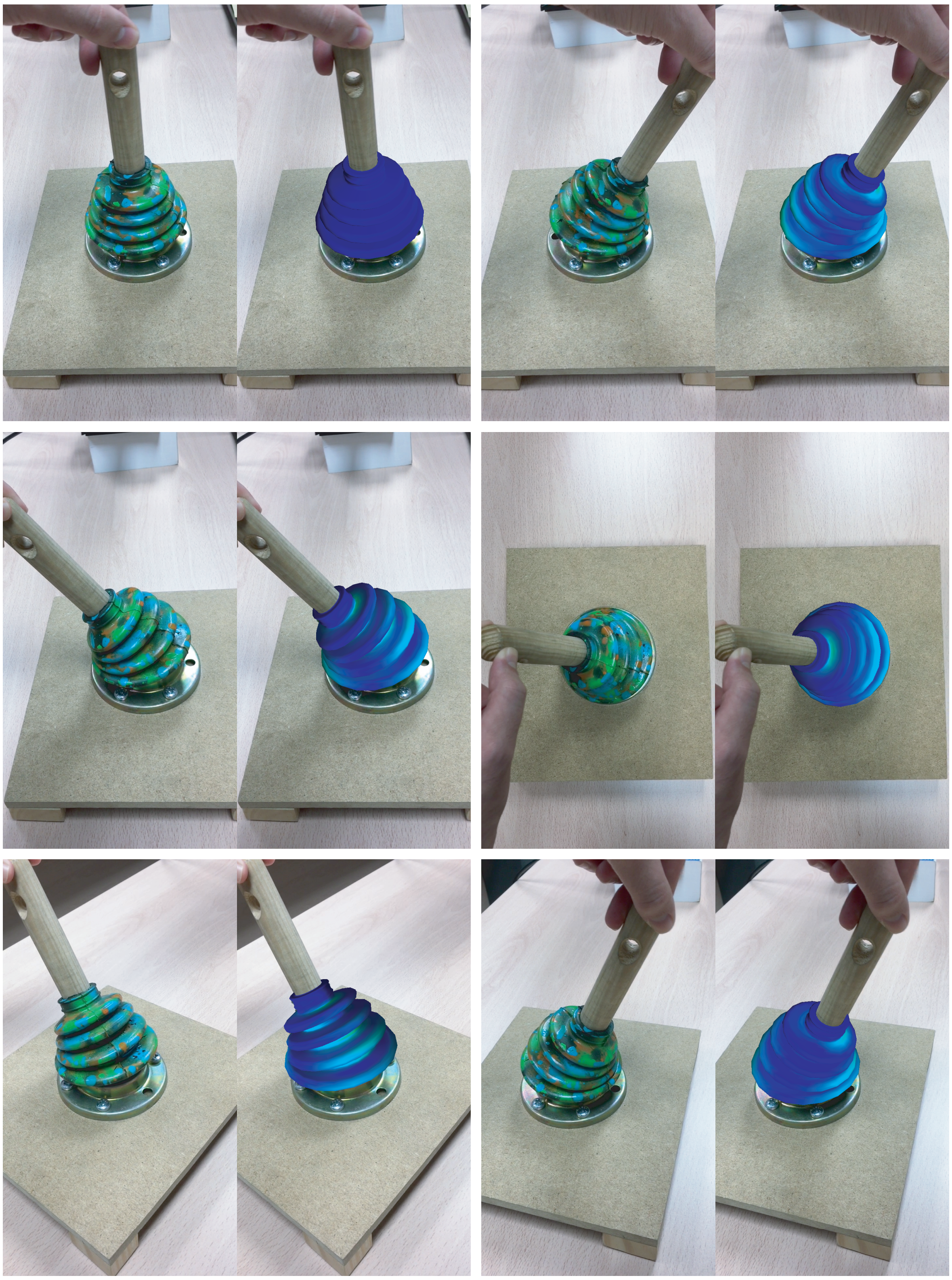


Figure 4.20: Some snapshots showing the original and augmented frames with stress information.

Our method has a great computational efficiency, being CPU-based, and it is quite robust with noisy observations. It is not necessary to precompute the sequences to extract the trajectories, nor learning shapes, as we obtain the information directly from our pre-computated models. But the solution must be obtained for a parametric set of loads, states and deformations enough to work with any sequence.

In all of the examples the usage of reduced order methods allowed for a very efficient and accurate determination of the variables of interest at video frequency. A standard iPhone 6s camera was employed in all examples.

The presented method opens the door for more complex augmented reality applications that truly enable a linkage of simulation techniques with video formats, thus widening the possibilities of modern CAE techniques.

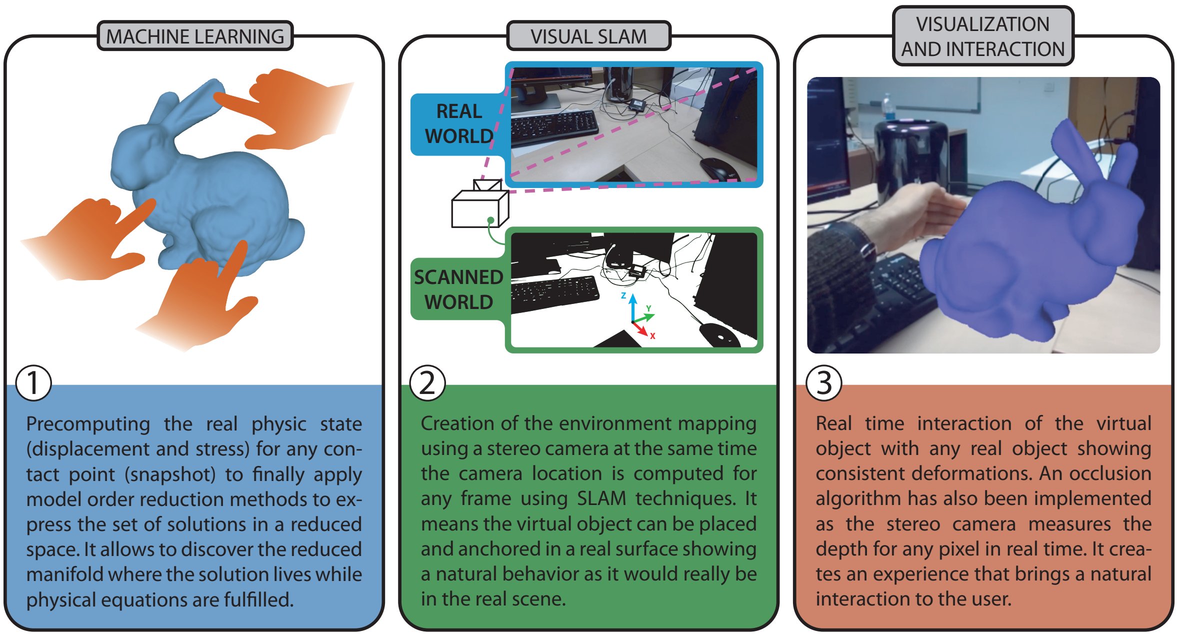
Chapter 5

Virtual-Physical Interaction in Simulated Reality Environments

New technologies and devices are joined with computational mechanics in the current chapter to create a mixed reality system able to interact with virtual objects using data extracted from images. We believe that this fusion is novel in the field of augmented reality, understood as the positioning of static information in space [Fedorov et al., 2016] or animated sequences [Paavilainen et al., 2017], but in very few cases deformable solids are taken into account, involving strong simplifications [Haouchine et al., 2012]. We are looking for a simple and natural interaction with virtual objects, to deform them as if they were really in front of the user.

In general words, this work aims to resolve in real time the contact problem between a virtual object and any real object. To do this, we first anchor the deformable virtual object in our real world, so we use methods of *Simultaneous Localization and Mapping* (SLAM) to locate the camera at any time while scanning the environment and creating a static 3D map of it. For this task, we used a *Zed Mini* stereo camera from *Stereo Labs*[1] that directly estimates the scene depth in each frame, at the same time it merges the data in time with the camera movements to build the global map. Once the environment surrounding the user has been scanned, we anchor the virtual object on a real surface of the scene. Since the stereo camera recovers the correct scale, the size of the virtual object is consequent with the scale environment. Finally, it is possible to interact with the virtual object in order to deform it by contact with any real object. To do that we previously defined a set of points on which the contact is evaluated, since solving the real physics of objects with complex geometries and non-linear material laws does not allow real time rates. When any real object approaches one of these points, deformation begins, which increases as contact grows. Figure 5.1 shows a summary of the process.

[1]`https://www.stereolabs.com/zed-mini/`



Precomputing the real physic state (displacement and stress) for any contact point (snapshot) to finally apply model order reduction methods to express the set of solutions in a reduced space. It allows to discover the reduced manifold where the solution lives while physical equations are fulfilled.

Creation of the environment mapping using a stereo camera at the same time the camera location is computed for any frame using SLAM techniques. It means the virtual object can be placed and anchored in a real surface showing a natural behavior as it would really be in the real scene.

Real time interaction of the virtual object with any real object showing consistent deformations. An occlusion algorithm has also been implemented as the stereo camera measures the depth for any pixel in real time. It creates an experience that brings a natural interaction to the user.

Figure 5.1: Summary of the whole process, starting from the application of reduced models to the precomputed solution (Machine Learning), going through the scanning process (SLAM) and finally the visualization and interaction with the virtual object in real time.

The chapter is divided in 5 sections. The first section covers the related work. In the second section the general problem is formulated, covering the physical models, model order reduction techniques applied, a stereo SLAM introduction, contact estimation and the visualization process. The section three covers the data assimilation process. Next section shows some experimental results, and finally, in section five the chapter conclusions are discussed.

The content of this chapter has resulted in the publication of the following journal paper:

- Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Under revision. Real-time interaction of virtual and physical objects in Mixed Reality applications. *Computational Mechanics*.

5.1 Related Work

The related work shows some implementations already published about contact on deformable solids.

5.1.1 Contact on Deformable Solids

General contact between solids is a classic problem in the mechanics of deformable solids. There are many published works that solve this problem from a numerical point of view by changing the boundary conditions. Two basic known implementations are the *Lagrange multipliers* [Chaudhary and Bathe, 1986] and the *penalty method* [Perić and Owen, 1992], together with multiple variants to adapt the fomulation to each type of problem (such as, for example, the contact dynamics problem [Brunßen et al., 2007]). Other works define the contact domain as independent, avoiding having to project certain restrictions between the slave and master surface [Oliver et al., 2009]. However, all this complexity cannot be dealt with in real time by a standard finite element solver. These solvers are based, in general, on a looping algorithm that must iterate until a solution that complies with the physics is found, avoiding the existence of penetrations from one object to another. We therefore need to use *Model Order Reduction* (MOR) methods to reduce the problem complexity, that consists in estimating the deformations due to contact, through a series of *snapshots* or previous observations of these deformations. In addition, since we want the contact to take place between the virtual object and any real object where we do not know the geometry of the latter, we cannot apply classical methods. We therefore apply the idea of *collision* (well known in computer graphics [Breen et al., 1995]) instead of the standard contact between two objects from classical mechanics. This idea of collision applied to deformable objects allows us to estimate how the virtual object is deformed when interacting with any object through the application of a system of loads located at a point, without being necessary to define that second object for the contact (see Fig. 5.2).

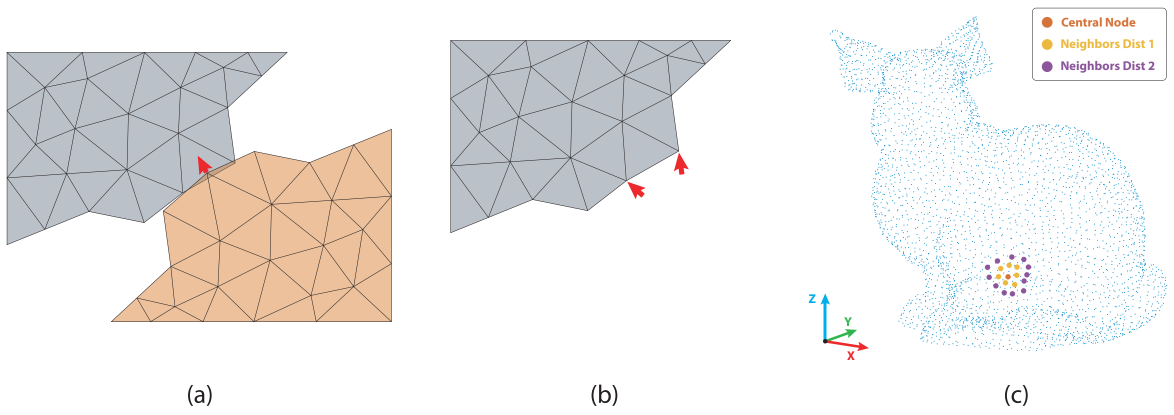


Figure 5.2: Collision problem showing the load application. (a) Standard contact problem, where the boundary forces applied (red arrow) depend, among other factors, on the stiffness of both solids. (b) Collision problem implemented in our method using incremental loads applied to the contour, as the rigidity of the second object is not known. (c) Example of load application in the bunny object showing the nodes where the load is applied to obtain smooth and more realistic deformations.

There are published some works that estimate the collision of objects in real time, such as [Takeuchi and Koike, 2017, Montero et al., 2019]. But, according to our knowledge, there is no work that allows the interaction in real time of deformable virtual objects where the real physics are solved, giving the correct values of the stress state of the object in any time instant.

5.2 Problem Formulation

The problem as a whole can be divided into four main tasks: precomputing some displacements (snapshots), applying model order reduction methods, scanning the static scene and contact detection in real time.

5.2.1 Solving the physics

In this case we are focused on deformable solids, but the method can be applied to any other physical problem, such as fluid mechanics (Chapter 6) or even electromagnetic problems [Chinesta et al., 2016].

In general, the mechanical problem of static deformable solids with hyperelastic behaviour is intended to solve the equilibrium equation:

$$\boldsymbol{\nabla P} + \boldsymbol{B} = \boldsymbol{0} \text{ in } \Omega_0, \tag{5.1}$$

where $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{B}$ collects the set of applied forces and Ω_0 represents the volume of the solid. The above equation is subject to boundary conditions

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{X}) = & \ \bar{\boldsymbol{u}} \text{ on } \Gamma_u, \\ \boldsymbol{PN} = & \ \bar{\boldsymbol{T}} \text{ on } \Gamma_t. \end{aligned}$$

where $\boldsymbol{u}(\boldsymbol{X})$ is the displacement field, $\bar{\boldsymbol{u}}$ are the imposed displacements, Γ_u the portion of the boundary where essential conditions are imposed, $\boldsymbol{N}$ the normal vector and $\bar{\boldsymbol{T}}$ the traction forces imposed in Γ_t, the portion of the boundary where natural conditions are imposed. The behavior of hyperelastic materials is defined by the strain-energy function Ψ (Helmholtz free-energy potential) [Reddy, 2013], which is related to the second Piola-Kirchhoff stress tensor by

$$\boldsymbol{S} = \frac{\partial \Psi}{\partial \boldsymbol{E}}, \tag{5.2}$$

being $\boldsymbol{E}$ the Green-Lagrange strain tensor. Eqs. (5.1) and (5.2) are related by $\boldsymbol{P} = \boldsymbol{FS}$ where $\boldsymbol{F}$ is the deformation gradient tensor. Finally, the weak form of the problem can be obtained combining Eq. (5.1) with a test function $\boldsymbol{u}^*$ (Galerkin standard projection) and integrating over Ω_0

$$\int_{\Omega_0} (\nabla \cdot \boldsymbol{FS} + \boldsymbol{B}) \cdot \boldsymbol{u}^* d\Omega_0. \tag{5.3}$$

The above equation will ultimately depend on the constitutive law chosen (through $\boldsymbol{S}$) to describe the behavior of the material. In this chapter we show some examples applying Saint-Venant–Kirchhoff behavior and Neo-Hookean behavior (see Sec. 5.4 for more information).

The computation of the weak equation also depends on the model order reduction method used, since some methods are intrusive and require some modifications in the original equation, as we have seen in Chapter 2. Therefore, the following section shows the application of MOR methods in both the intrusive and non-intrusive branches.

5.2.2 Model Order Reduction

In order to create a general methodology, we have applied both *a priori* (equation-based) and *a posteriori* (data-based) MOR methods. In both cases we have used the Proper Generalized Decomposition (PGD) methodology, in its intrusive version [Ammar et al., 2006] and in its non-intrusive version [Ibáñez et al., 2018]. There is a powerful non-intrusive implementation of PGD [Zou et al., 2018] where a commercial solver is used to perform the computation tasks, but the original PGD algorithm does not require modification of the weak form equation. It means continuous calls to the solver that computes the problem equation and feedbacks the PGD algorithm. In our work, we apply a different approach, where we only use the non-intrussive PGD algorithm to project the high-dimension data that comes from an external solver.

The intrusive version of the PGD method requires accessing and modifying the operator $\mathcal{L}(\boldsymbol{u})$ of the differential equation, which in the case of deformable solids refers to the Eq. (5.3). As we know, PGD is based on the expression of the solution $\boldsymbol{u}(\boldsymbol{\mu})$ in separate variables

$$\boldsymbol{u}(\boldsymbol{\mu}) = \sum_{i=1}^{N} \boldsymbol{F_i}(\mu_1) \circ \boldsymbol{G_i}(\mu_2) \circ \cdots \circ \boldsymbol{H_i}(\mu_p)$$

where N is the number of summands or *modes* needed to approximate the real solution, which is a finite number and hopefully small. Terms $\boldsymbol{F}, \boldsymbol{G}, ...\boldsymbol{H}$ are the functions expressed in separate variables depending on parameters $\boldsymbol{\mu}$, being p the number of those parameters. These number of independent parameters on which we want to separate the solution is our choice, giving the user the freedom to project the solution in any number of variables at pleasure. It is worth to mention that the higher the number of parameters p is, the greater the gain in terms of data compression is, since the undesirable effect of the curse of dimensionality is reduced. This is because the size of the problem takes a value of $\mathcal{N}_{\text{reduced}} = N_{\mu_1} \cdot N + N_{\mu_2} \cdot N + \ldots + N_{\mu_p} \cdot N$, being N a relatively small integer number, instead of $\mathcal{N}_{\text{original}} = N_{\mu_1} \cdot N_{\mu_2} \ldots \cdot N_{\mu_p}$, where N_{μ_i} refers to the discrete size of the vector μ_i. However, it is also relevant to mention that the larger p is, the more complicated the convergence of the method is, in general. Convergence depends on the differential operator of the equation to be solved and the appropriateness to linearly project the solution. Some works try to improve convergence rates working with local subdomains (nonlinear in global domain) as shown in Chapter 3, and other works adapted nonlinear operators to the PGD algorithm by cross approximations [Aguado et al., 2019] or asymptotic expansions [Niroomandi et al., 2010].

The non-intrusive version of the PGD method (NI-PGD) method tries to apply a greedy algorithm to obtain the decomposition. For this, let us suppose that the data lives in a space of dimensionality $D = 3$ and is stored in a tensor $\mathcal{F}$, wich depends on parameters (μ_1, μ_2, μ_3). The NI-PGD method approximates such data using the function $\boldsymbol{g}(\mu_1, \mu_2, \mu_3)$, defined as

$$\boldsymbol{g}(\mu_1, \mu_2, \mu_3) \approx \sum_{i=1}^{N} \boldsymbol{F_i}(\mu_1) \circ \boldsymbol{G_i}(\mu_2) \circ \boldsymbol{H_i}(\mu_3) \tag{5.4}$$

forced to

$$\left|\left|\mathcal{F} - \boldsymbol{g}(\mu_1, \mu_2, \mu_3)\right|\right|^2 = 0. \tag{5.5}$$

Like methods based on algebraic decompositions, it is also a non-optimal method in high dimensions, but the greedy algorithm of PGD is very simple to implement and obtains the modes very quickly, which in the end translates into a very simple tool to implement the projection in separate variables. This non-intrusive method makes sense when we do not know the differential equation from which the data come.

5.2.3 Stereo Visual Simultaneos Localization and Mapping

Although there are very precise monocular systems, we decided to use a stereo system to reduce the complexity of the spatial scanning process. The general theory of monocular SLAM has previously been introduced in Sec. 2.3 (Chapter 2), so here we jump directly to the differences that appear when a stereo camera is used.

Since a stereo vision system consists of two monocular cameras rigidly joined together, the triangulation process can be carried out on each frame (which are actually two images taken at the same time, see Fig. 5.3). In addition, the fact that both cameras are rigidly fixed implies that a series of assumptions can be applied to simplify the process, which can be summarized step-by-step below [Bradski and Kaehler, 2008]:

- Image undistortion: lenses used in conventional cameras apply radial and/or tangential distortions that must be corrected in order to obtain the real image.
- Image rectification: adjustment of the two images captured with the stereo camera to produce alignment and rectification.
- Correspondences: relate the points observed in the left image with their counterparts in the right image, producing the disparity map.
- Reprojection: obtaining the depth of each point from the disparity map.

As the images from both cameras have been previously undistorted and rectified, the triangulation process is simplified with the known baseline separation between cameras, where the cameras are assumed perfectly coplanar (Fig. 5.3).

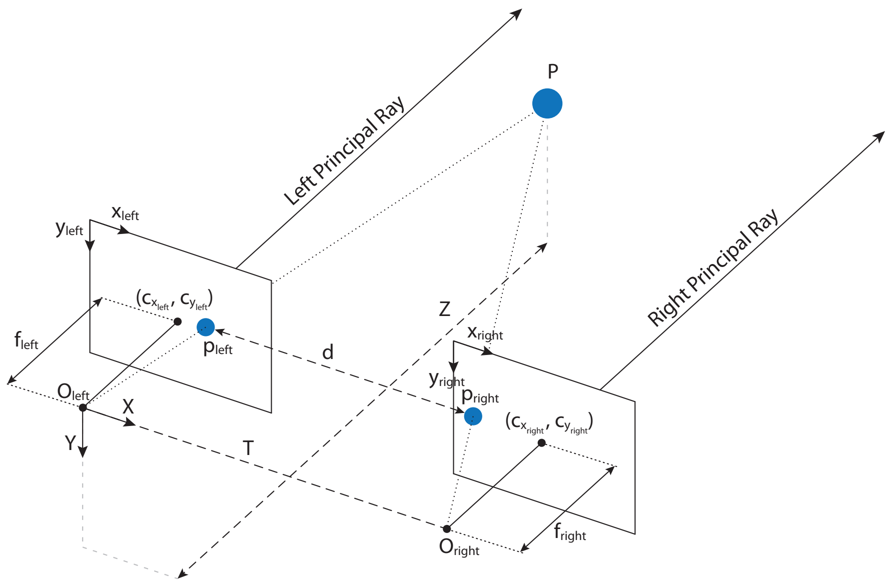


Figure 5.3: Stereo camera system, where the triangulation process is simplified assuming coplanarity. f_{left} is the left focal distance, $(c_{x_{left}}, c_{y_{left}})$ are the points where the left principal ray is intersecting the left image plane, O_{left} is the origin of the left principal ray, T is the displacement between left and right camera centers and d is the disparity (difference) of the horizontal coordinates of the left and right projections of point P.

Assuming point P of the scene is captured by both cameras in p_{left} and p_{right}, respectively, and that the calibration and rectification processes have been successful, for a fronto parallel camera both points appear on the same vertical coordinate, being $y_{p_{left}} = y_{p_{right}}$ (row aligned). The difference between the horizontal coordinates defines the disparity $d = x_{p_{left}} - x_{p_{right}}$, which can be related to depth by the equation

$$Z = \frac{f\,T}{x_{p_{left}} - x_{p_{right}}}$$

where f is the focal length (assuming same value for both cameras) and T is the displacement between camera centers of projection.

Once the camera geometry is defined, only the points observed in left and right images should be matched to recover the depth. This can be done by searching for relevant points extracting *features* [Fua, 1993] or in a *dense* way by comparing left and right image by applying *energy* based techniques [Alvarez et al., 2002]. The second technique is more desirable since it produces a dense map of the scene, estimating depth values for each pixel. As can be expected, this process is computationally expensive, so modern computers equipped with GPU are used to perform this massive computation. In our expermients, we obtain frequencies of 30 frames per second (fps) for 1080p video resolution or 60 fps for 720p[2].

[2]`https://www.stereolabs.com/zed-mini/`

In addition to the online stereo depth perception, the system must be self-localized at any time, so it is necessary to merge the depth captured in each frame for all time instants (SLAM). The result is a 3D map of the static objects that surround the camera while the camera trajectory is also estimated (Fig. 5.4).

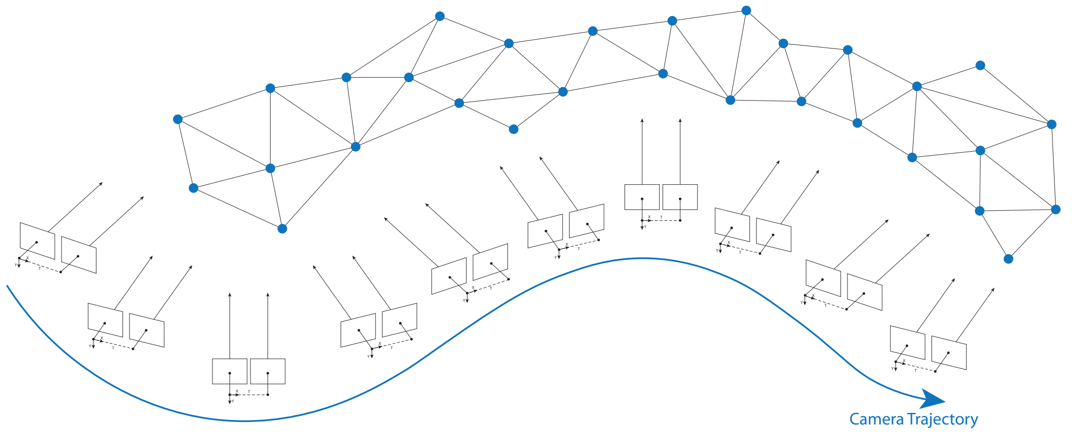


Figure 5.4: Trajectory of a stereo system, where the path travelled over time is shown by the blue line. The stereo camera estimates the depth in each frame, so it is necessary to merge the information of all frames to create a complete map of the scene. In addition, it is possible to apply a mesh algorithm to estimate the external surface of the objects and get more information about the scene [Lin et al., 2004].

5.2.4 Collision Estimation

Any real object that penetrates over the virtual object produces deformations. Collision estimation is evaluated thanks to the fast rates of the stereo camera where the graphic acceleration ensures real time frequencies to estimate, in any frame, the depth of each pixel. Therefore, the camera is capable to estimate also the intra-frame position of the *dynamic* objects, although they are excluded from the SLAM algorithm. It translates into a natural and robust feeling in the interaction, where static objects are used to locate the camera in space while real dynamic objects are used in the collision estimation.

We use a simple algorithm to measure 3D distances between virtual and real objects to estimate contact in the collision process, which is simplified in Alg. 5.1. The minimum distance of any real object with respect to the virtual object (evaluated in *nodeVirtualObject*) is $\mathrm{d}_{\mathrm{min}}$, and when $\mathrm{d}_{\mathrm{min}} < \delta$ contact is appearing, being δ a predefined value. The variable *deformationPseudoTime* indicates the deformation with respect to the maximum value (*maxDeformationPseudoTime*), understood as a pseudo-time whose range is $0 \leq$ *deformationPseudoTime* $\leq$ *maxDeformationPseudoTime*. It does not imply that the problem to be solved is linear elastic, in fact all the examples that appear in Sec. 5.4 are defined by hyperelastic material laws. It is simply a way of handling the solutions projected in the reduced space.

A decimation was applied on the depth image to reduce the computational complexity of estimating distances with the virtual object. Very small objects do not appear, in general, inside the normal distance range from the camera, so we consider this simplification has no drawbacks. In addition, there is an intrinsic error associated with estimating depth from the disparity map, which means that the dense reconstruction may involve noise in the estimation of very small objects. The decimation is only applied to the distance estimation algorithm, so the visualization process is carried out in high resolution.

As can be seen in Alg. 5.1, contact with the virtual object is only allowed at one point of the contour per instant, since deformations do not comply with the principle of superposition when the material does not follow a linear behavior. In case more than one contact point at a time is desired, new simulations need to be precomputed including these new loading states. Thus, the manifold $\mathcal{M}$ obtained after applying the MOR method to project the solution would store all that new information.

Algorithm 5.1: Pseudo-code of the collision-deformation algorithm.

```
matricesMemoryAllocation(3DCoordsVirtObject, Raw3DCoordsStereo);
while (True):
{
    (3DCoordsStereo) = decimate3DStereoData(Raw3DCoordsStereo, $N_{step}$);
    ($d_{min}$, nodeVirtualObject) = parallelDist(3DCoordsVirtObject, 3DCoordsStereo);
    if ($d_{min} < \delta$) and (deformationPseudoTime < maxDeformationPseudoTime):
    {
        deformationPseudoTime++;
        update3DNodePositions(nodeVirtualObject, deformationPseudoTime);
    }
    else if (deformationPseudoTime > minDeformationPseudoTime):
    {
        deformationPseudoTime–;
        update3DNodePositions(nodeVirtualObject, deformationPseudoTime);
    }
}
```

5.2.5 Visualization

To visualize all the content in a fluid way we use OpenGL, that allows to render primitives efficiently getting real-time rates (the minimum frequency of visualization is imposed by the capture rates of the stereo camera). Thanks to the simplicity and optimization of OpenGL we added *lighting* effects that translate into a more natural visualization of the virtual objects, fitting better into the real scene. It is not our goal to create a hyper-realistic rendering, so we employ basic visualization techniques.

Occlusions are a characteristic that gives a natural behavior to the augmented reality interaction, but its computation is one of the most complex tasks, since it is not easy to perfectly estimate the depth of all objects to *draw in front* those that are closest to the camera (*Z-culling*, term used in the computer graphics community for this process [Greene et al., 1993]). Since the stereo camera estimates the depth of all the pixels in the image, we implemented a simple algorithm capable of applying occlusions to the video sequence in real time. For this task we use *shaders* (written in *OpenGL Shading Language* or GLSL[3]), which are small programs that run directly on the GPU and are able to perform simple tasks in parallel and very efficiently. Occlusions are estimated between real and virtual objects for each pixel of the image as a function of the distance to the camera (see Fig. 5.5). It is important to remember that we are not using object recognition techniques, but only the depth information from the stereo camera.

5.3 Data Assimilation Process

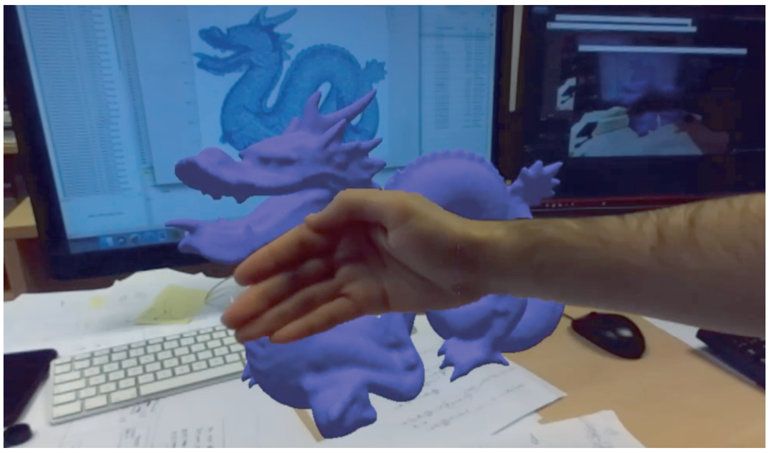
The fusion of all parts described in previous sections build the whole method. It is the interaction between all of them where the advantage lies, and we understand this process as *Data Assimilation*: collecting the information that comes from the images to feed our physical models and visualize the updated result of those models.

The information comes directly from the camera, where thanks to the contact algorithm, the point n^i where the load is applied and the pseudo-time deformation d^i in node i are estimated, being $d \in D = [0, 1]$ where $D_0 = 0$ and D_1 represents the maximum deformation (*maxDeformationPseudoTime*). Displacements are related to the module of this pseudo-time deformation d, so it is straightforward to reconstruct the particular solution for that set of parameters $\mu^i = (n^i, d^i)$ from the reduced dimensional space to the original space where the high-fidelity solution lies. The displacements and the stress field can then be plotted on the object at video frequency (30 frames per second).

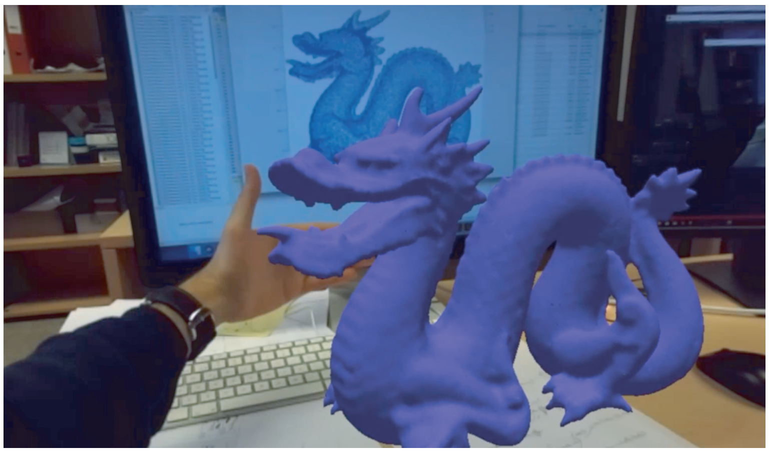
5.3.1 Implementation details

We used a stereo camera from *StereoLabs*, model *ZED Mini*. To carry out all computational tasks we used a workstation with an *Intel Core i7-8700K* CPU, where the graphics part was the most critical since the stereo camera needs graphic acceleration, for which we used a *Nvidia GeForce RTX 2070*. We used and integrated the SDK of the stereo camera with our own code, where the functions to obtain the depth of the camera and the SLAM fusion have been provided by the manufacturer.

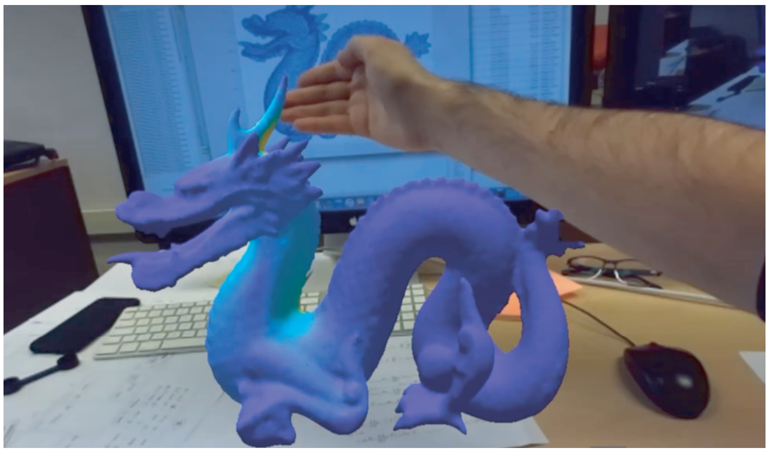
[3]https://www.khronos.org/opengl/wiki/Core_Language_(GLSL)



(a)



(b)



(c)

Figure 5.5: Occlusions implementation to visualize in a natural way the objects that are closer to the camera. Frames extracted from one of our videos (`https://www.youtube.com/watch?v=jtMe47mg82k`), working in real time and with no post-production modifications. (a) Hand of the user (real object) closer to the camera than the virtual object. (b) Virtual object closer to the camera than the hand of the user (real object). (c) Interaction between both objects, virtual and real, with the occlusion system working properly.

5.4 Experimental Results

In this section we show three examples to confirm that our method works robustly. The first example shows the contact between two virtual objects and any real object. Next examples only use one virtual object, but the interaction is allowed with any object of the real environment.

5.4.1 Cantilever Beams

The first example consists of two cantilever beams placed perpendicularly, where the free ends of both beams overlap, being that part where virtual contact occurs (see Fig. 5.6).

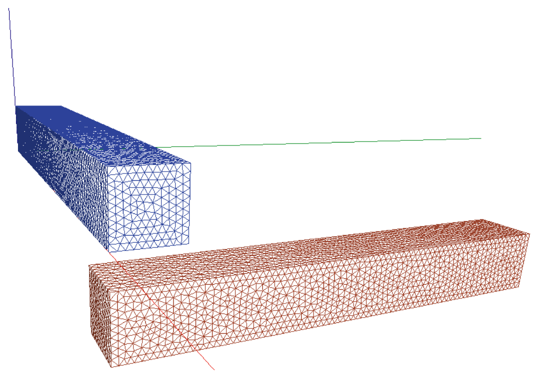


Figure 5.6: Beams distribution in space in the contact problem. The contact betweem real and virtual object is applied in the upper face of the blue beam and when it deforms enough, contact between both beams arises.

The beams are simulated as a hyperelastic material (Saint-Venant–Kirchhoff), where the energy density function can be expressed as

$$\Psi(\boldsymbol{E}) = \frac{1}{2}\lambda[tr(\boldsymbol{E})]^2 + \mu\boldsymbol{E} : \boldsymbol{E}.$$

We applied the PGD method in its intrusive standard version with one parameter $\boldsymbol{S}$ that serves to parameterize the position of load P. The solution in separate variables takes the form

$$\boldsymbol{u}(\boldsymbol{X}, \boldsymbol{S}) \approx \sum_{i=1}^{N} \boldsymbol{F}_i(\boldsymbol{X})\, G_i(\boldsymbol{S}), \tag{5.6}$$

where $\boldsymbol{X} = (x, y, z)$ are the spatial coordinates of each beam node, $\boldsymbol{S} = (s_x, s_y)$ are the application positions of the load on the top face of the blue beam (Fig. 5.6) and N is the number of modes needed to approximate the solution. Thus, functions $\boldsymbol{F}_i(\boldsymbol{X}) \in \mathbb{R}^3$ have a dimensionality of 3 and functions $G_i(\boldsymbol{S}) \in \mathbb{R}^1$ are 1-dimensional (vertical load).

To obtain the evaluated solution for a specific position of the load in the original high-dimensional space it is necessary to particularize the solution for a value of G_i, which is computed by simple products.

The weak form of the equation to be solved, therefore, is expressed as

$$\int_{\bar{\Gamma}} \int_{\Omega} \boldsymbol{\nabla}_{\boldsymbol{s}} \boldsymbol{u}^* : \boldsymbol{\sigma} \, d\Omega d\bar{\Gamma} = \int_{\bar{\Gamma}} \int_{\Gamma} \boldsymbol{u}^* \, \boldsymbol{t} \, d\Gamma d\bar{\Gamma} \tag{5.7}$$

where Ω represents the volumetric solid, $\bar{\Gamma}$ is the boundary region discretized by S where the load can be applied, $\boldsymbol{u}^*$ is the test function, $\boldsymbol{\nabla}_{\boldsymbol{s}}$ the symmetric operator, ":" the double contraction operator, $\boldsymbol{\sigma}$ the stress tensor and $\boldsymbol{t}$ the contour load applied. The load function can be separated as

$$\boldsymbol{t} \approx \sum_{j=1}^{M} \boldsymbol{f}_j(\boldsymbol{X}) \, g_j(\boldsymbol{S}). \tag{5.8}$$

By entering Eqs. (5.6) and (5.8) into (5.7), it is possible to solve the parametric problem by projecting the solution into separate spaces (intrusively). If more information is desired about the PGD application to the nonlinear equation of Saint-Venant–Kirchhoff, see [Niroomandi et al., 2013].

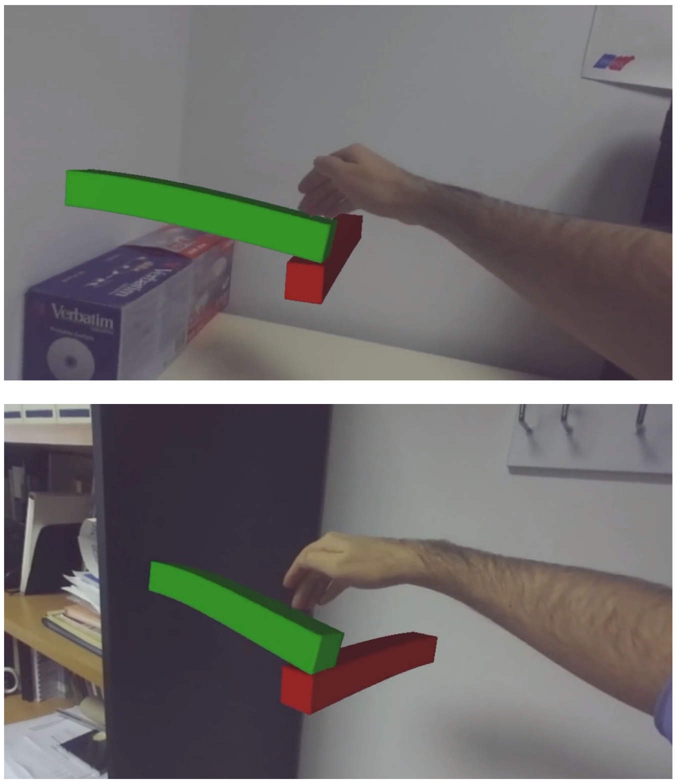


Figure 5.7: Frames extracted from the beam contact video sequence, where the real-virtual contact is applied in the upper part of the green beam and virtual-virtual contact is computed between virtual beams.

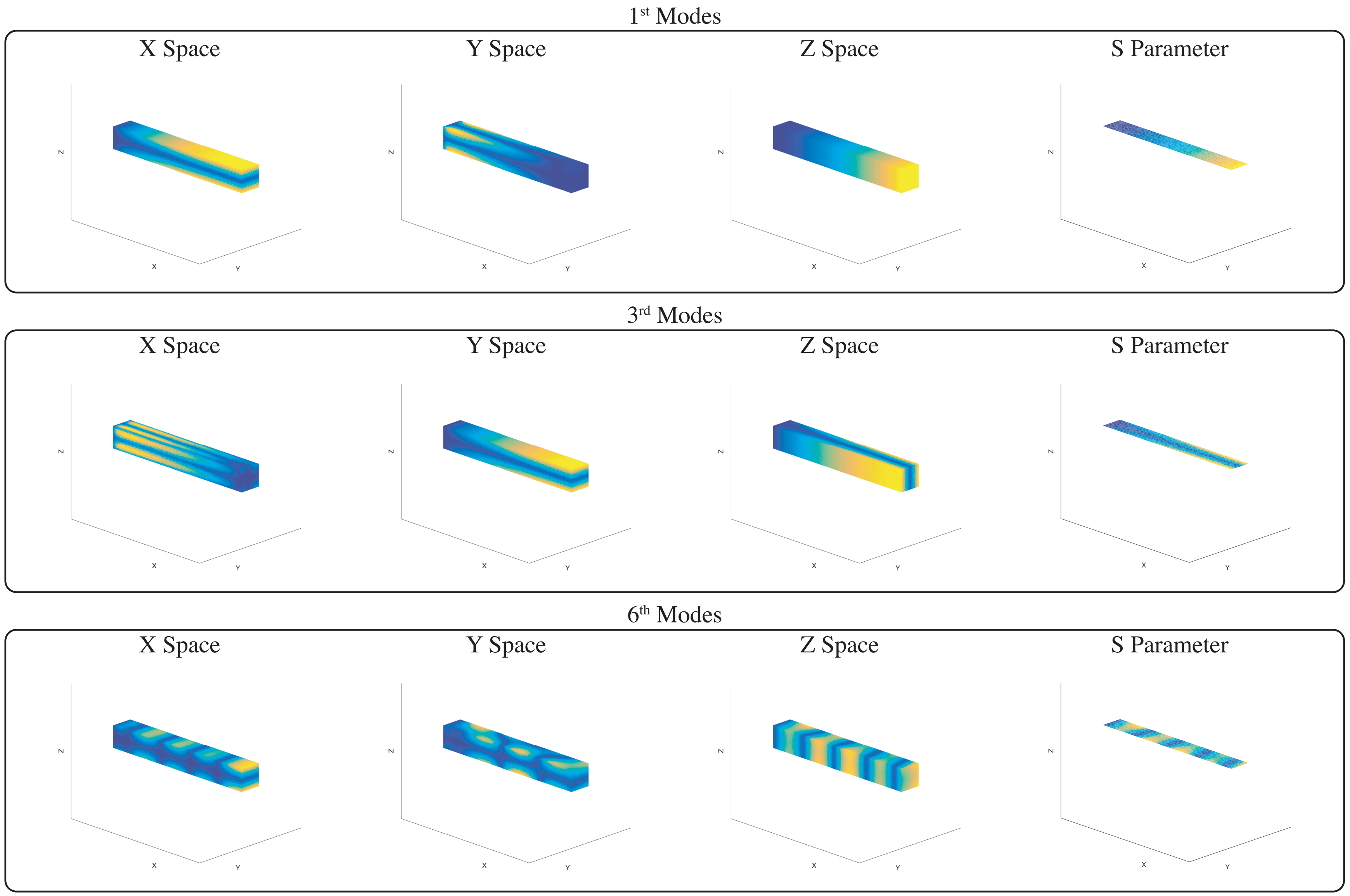


Figure 5.8: 1^{st}, 3^{rd} and 6^{th} modes of the cantilever beam example, showing the *energy* of the solution projected in separated spaces.

The beam finite element mesh has a total of 3381 nodes with 6758 elements in space, while the number of nodes for the S parameter is 1749 (the total number of load positions). The result of the $1^{st}, 3^{rd}$ and 6^{th} spatial and S parameter modes can be seen in Fig. 5.8, where the *energy* of the solution projected on the separate spaces is represented. In the case of S modes, we plot only the upper surface, since the load can only be applied in that domain.

The parametric solution obtained for the load applied in one beam is actually applied on both beams, since the problem to be solved is the same, although the procedence of the load is different. For the upper beam, contact is made with any real object captured by the stereo camera. For the lower beam, contact is produced by the deformed upper beam (which in turn has been deformed with the contact of the real object). Therefore, we have solved the problem only once, although the solution is evaluated online in two beams. The resulting sequence can be seen in `https://youtu.be/PaK7STfWGfs`, and some frames from that video are shown in Fig. 5.7.

5.4.2 Stanford Bunny

The second example uses the Stanford bunny [Stanford, 1993] as virtual object to deform. The material law applied is homogeneous and isotropic throughout the solid, with a Neo-Hookean compressible hyperelastic model. The energy density function can be expressed as

$$\Psi = C_1(\bar{\boldsymbol{I}}_1 - 3 - 2\ln J) + D_1(\ln J)^2 \tag{5.9}$$

being C_1 and D_1 material constants, $J = \det(\boldsymbol{F})$ being $\boldsymbol{F}$ the deformation gradient and $\bar{\boldsymbol{I}}_1$ the first deviatoric strain invariant defined as

$$\bar{\boldsymbol{I}}_1 = \bar{\lambda}_1^2 + \bar{\lambda}_2^2 + \bar{\lambda}_3^2 \tag{5.10}$$

where $\bar{\lambda}_i = J^{-1/3}\lambda_i$. Once the solution has been precomputed with a commercial solver (Abaqus CAE), we used the Non-Intrussive PGD method to project the solution obtained onto a space of reduced dimensionality. After the projection, we obtain the modes $\boldsymbol{F}(\boldsymbol{X})$, $\boldsymbol{G}(\boldsymbol{\sigma})$, $\boldsymbol{H}(\boldsymbol{S})$ where $\boldsymbol{F}$ stores the spatial information, modes $\boldsymbol{G}$ depend on the stress field and modes $\boldsymbol{H}$ store the *energy* of the solution projected on the space of parameter $\boldsymbol{S}$, that defines the position and module of the load. To simplify the problem, all loads are applied in the direction of the center of mass of the object. The discretization of the geometry is composed by a mesh of 16519 nodes and 87916 elements, where we defined 34 possible contact points (see Fig. 5.9, left) where the force is gradually applied during 26 pseudotime increments. However, to reduce the cost of visualization we only show the contour nodes (although calculations have been made with all the nodes), being reduced to 3733 nodes and 7462 elements. The result of the modes in space and stress can be seen in Fig. 5.10.

Fig. 5.11 plots the errors made by the approximation in the projection. The first column shows the normalized error (L_2-norm), while second and third columns show the errors

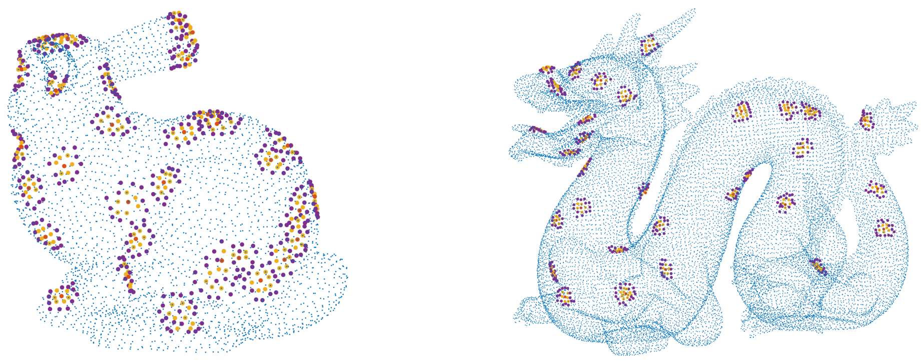


Figure 5.9: Contact points for the Stanford bunny and dragon. The applied loads simulating the contact are centered on the orange dots and have an application range of 2 neighbours (graph distance), marked in the figure with yellow and violet colors).

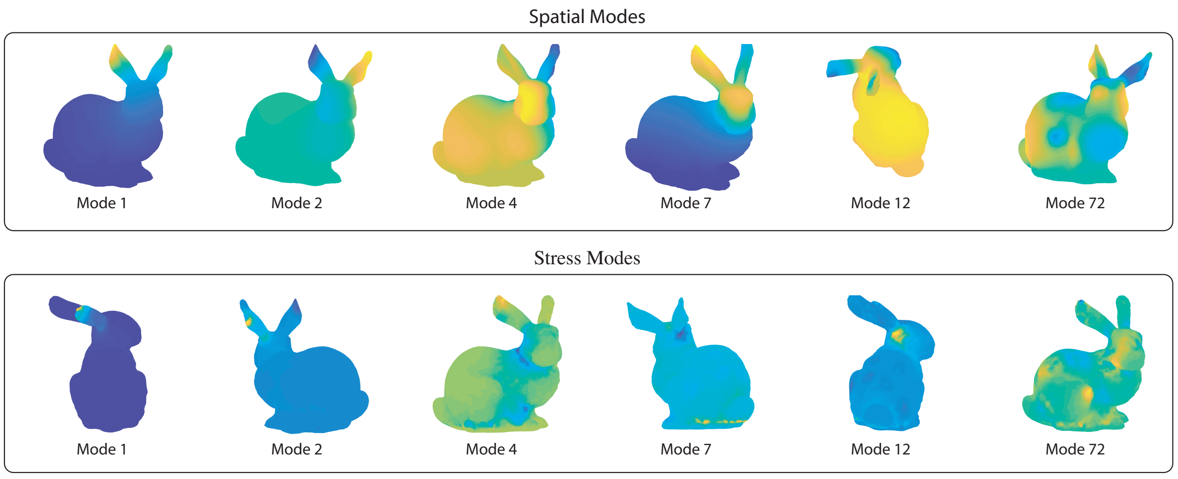


Figure 5.10: Spatial and stress modes, respectively, showing the *energy* of the solution projected in the reduced spaces.

with units (without normalizing) to facilitate a better understanding of the magnitude. For the spatial case we work with meters, and for the stress with megapascals. Since these errors are calculated for the 34 loading states and the 26 load increments, the average error is the most representative. But to be honest, we also show the maximum displacement and stress error for all loading states and for all instants (right column of Fig. 5.11).

Finally, Fig. 5.12 shows some frames extracted from a video sequence (`https://www.youtube.com/watch?v=lmApbJA6gH4`) recorded in a desktop. They show a person interacting with the virtual object touching it in different points, using his own hands but any other object could be used.

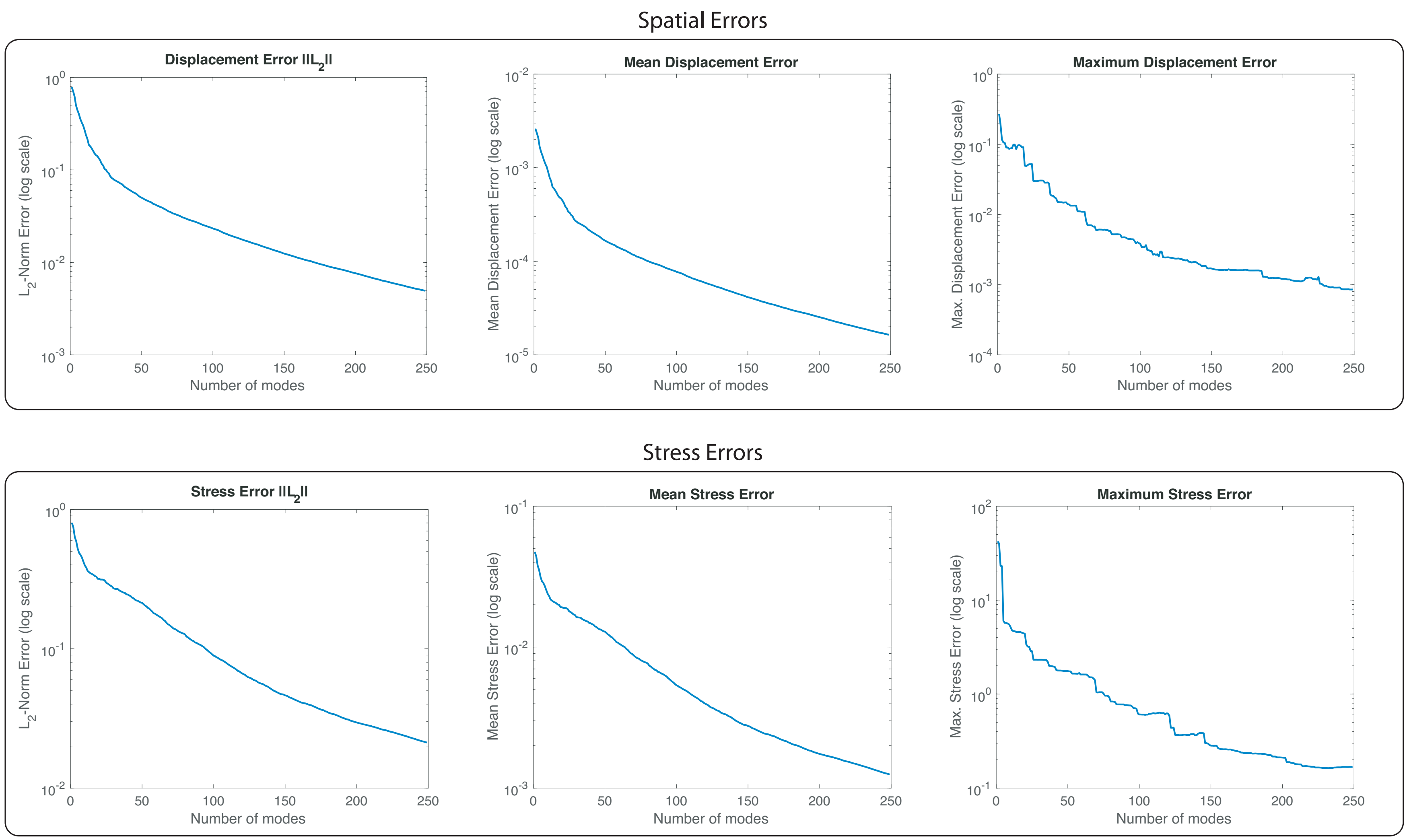


Figure 5.11: Reconstruction errors due to the projection process for the Stanford bunny, showing both the displacements and the stress field. Left column shows L_2-norm errors, center column shows median errors and right column shows maximum error for any loading point and any increment.

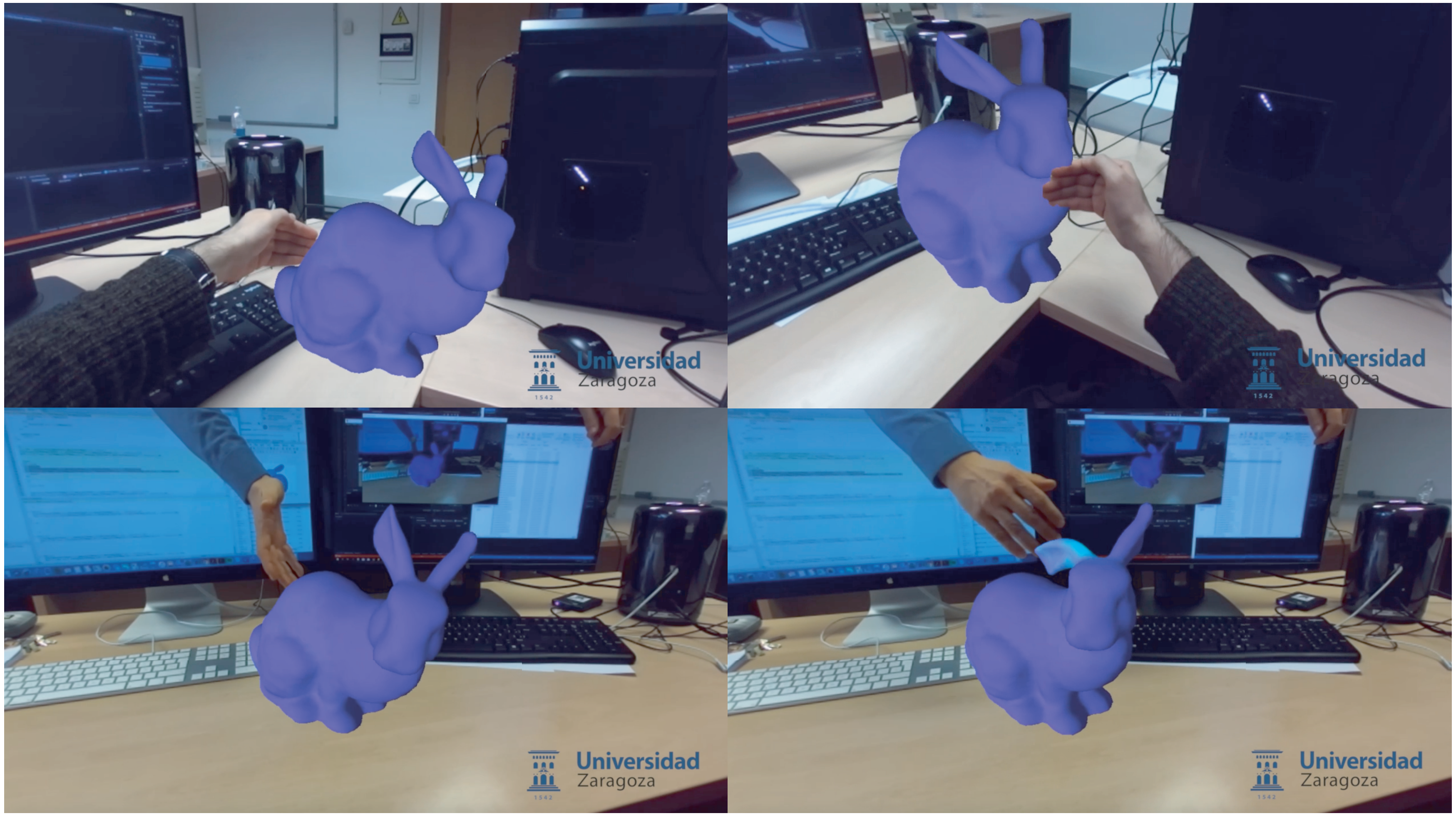


Figure 5.12: Four frames extracted from the bunny sequence. Colors show the stress field associated with the deformations imposed by contact with real objects.

5.4.3 Stanford Dragon

The third and last example uses the Stanford dragon [Stanford, 1993] as a deformable virtual object. The material law is the same as in the Stanford bunny. The number of possible contact points in this case is 32 (Fig. 5.9, right) with 16 pseudo-time increments to apply the loads progressively. Since the geometry of the dragon is more intricate than the bunny one, the directions of application of the loads follow the normal directions to the planes that form each set of load application points. The dragon mesh is discretized in 22982 nodes and 46540 elements, and Fig. 5.13 shows the result of the spatial and stress modes.

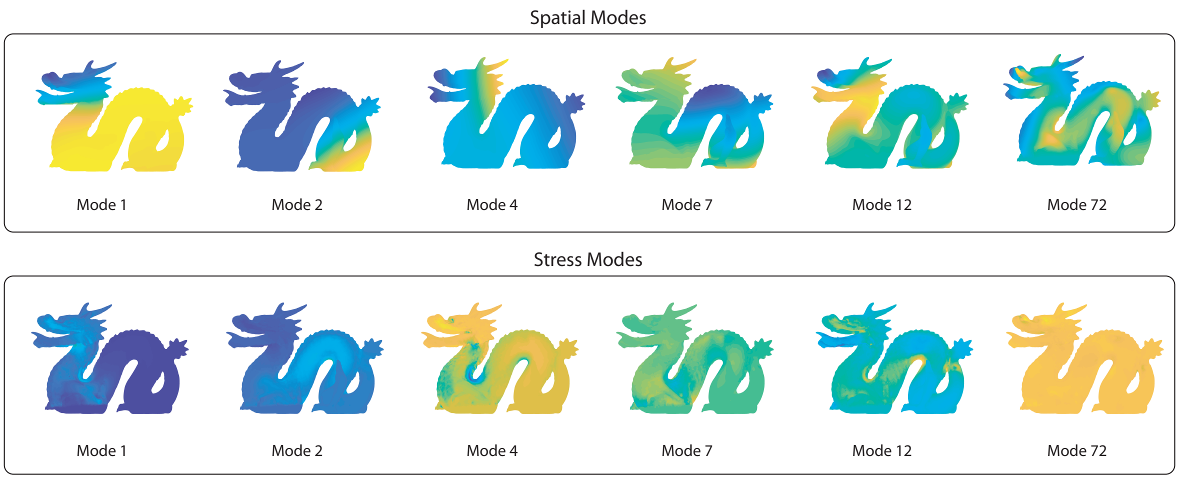


Figure 5.13: Spatial and stress modes, respectively, showing the *energy* of the dragon solution projected in the reduced space.

As in the bunny example, we plot the error reconstruction due to projection in Fig. 5.14. Left column shows the L_2-norm error for displacements and stress, central column shows the median error and right column shows the maximum error.

Fig. 5.15 shows four frames extracted from the same video sequence than the bunny. Here the user is touching two points of the dragon surface, producing displacements and showing in color the stress map that the contact forces generate.

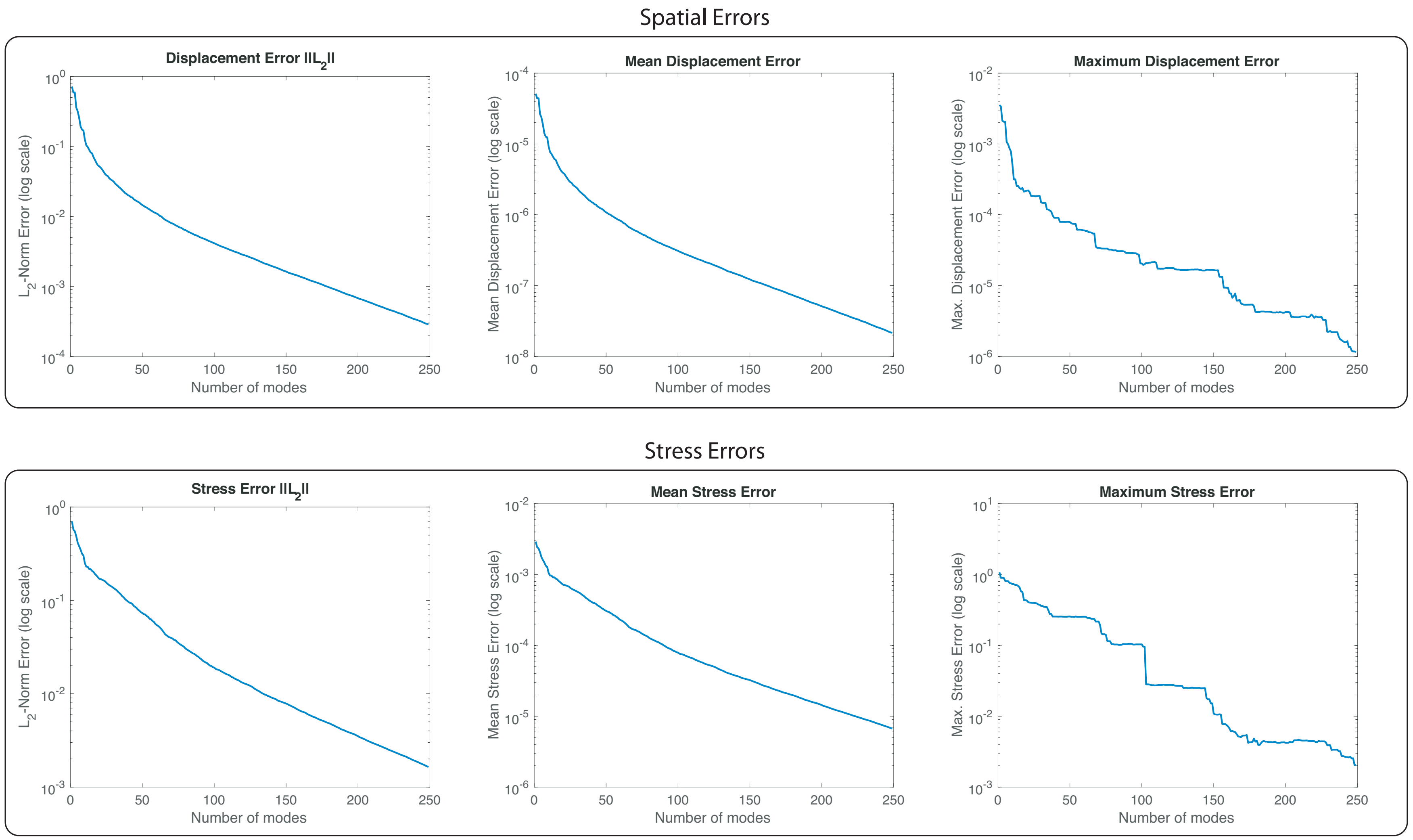


Figure 5.14: Reconstruction errors due to the projection process for the Stanford dragon, showing both the displacements and the stress field. Left column shows L_2-norm errors, center column shows median errors and right column shows maximum error for any loading point and any increment.

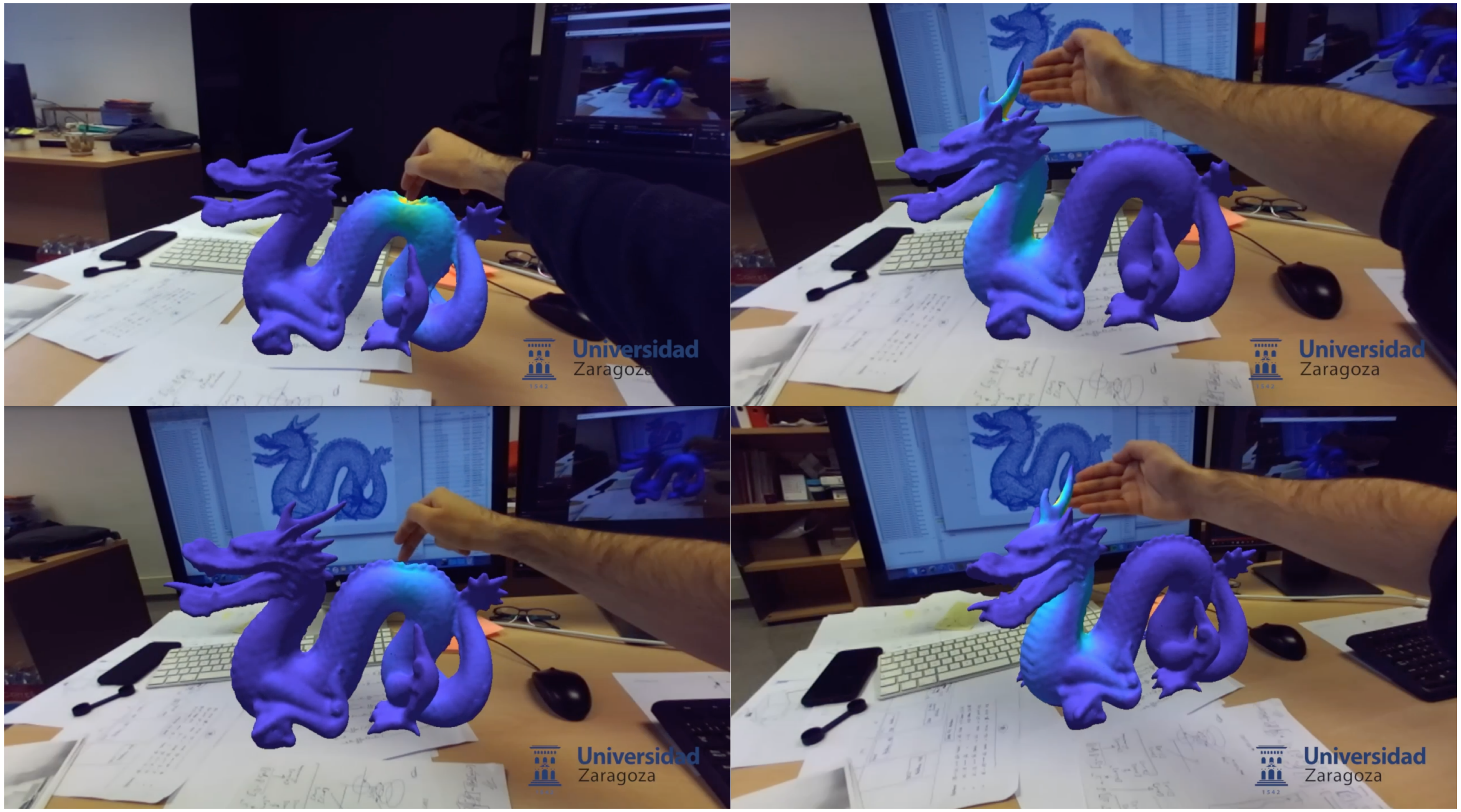


Figure 5.15: Four frames extracted from the dragon sequence. Colors show the stress field associated with the deformations imposed by contact with real objects.

5.5 Chapter Conclusions

In this chapter we present a system of interaction with virtual objects where three examples working in real time are included. Although the examples may seem related to the entertainment industry we would like to note the relevance that this type of work have in other areas such as medical surgery or industry. The visualization of relevant data in real time has an enormous importance in decision making, and by relevant data we mean visual information that our eyes cannot perceive directly. In the mechanical case we are talking about stress distribution in a solid, but we could talk about any other physical phenomenon that provides information with physical sense involving a simple and natural user interaction.

We believe that the introduction of model order reduction methods has much to say in this field. We used different methods, but of course many other MOR methods could be applied. The work proposed here represents a revolution in the estimation of physical phenomena, which serves to better describe the changes that occur in our environment. Physical engines (such as those used in video games) serve to greatly simplify physical equations working at video frequencies, but the results are far from what we really want to solve: high-fidelity models.

Visualization tasks and measurements with the camera are possible here thanks to the graphic acceleration. In our case we used a workstation, but nowadays it is also possible to use mobile devices with graphical acceleration, although of course the resolution is drastically reduced to maintain video frequencies around 30 frames per second. The problem of occlusions has also been treated in this work, analyzing the depth of each pixel by graphical acceleration and GLSL shaders, but it is still an open problem, where other works use dense segmentation of objects in images[4]. Using the stereo camera, small errors may appear in the measurement of the depth, which are usually due to not visible areas from both cameras at the same time, or surfaces that are oriented in the direction of the camera projecting rays. However, there are some works to correct these effects in geometry and texture by applying neural networks [Martin-Brualla et al., 2018].

[4] https://developer.apple.com/augmented-reality/arkit/

Chapter 6

Simulated Reality as a Machine Learning Problem

Simulation in engineering is living profound changes. It is not infrequent to hear about *democratization* –more than deployment– of simulation, or even about *appification*. In general, by *democratization* we mean to make simulation accesible to users that previously did not feel the need for it, but now consider it interesting. However, many undoubtedly interesting applications of simulation in engineering practice come at the price of many hours of user interaction followed by many CPU hours to obtain results. And this results in a barrier for many potential users.

In industry, particularly in automotive manufacturing, there is a claim for interactive ways to connect designers (artists, in sum) and engineers [Bertram et al., 2018]. For this to be possible, developing real-time, interactive simulation apps that could provide immediate responses on the physical effects of purely artistic decisions on mockups, for instance, could be of great help. This immediacy and interactivity should include the pre-processing stage in which engineers construct the CAE model of the product, the meshing operation for the development of a FEA model and a reasonable time to response.

In parallel, with the global deployment of mobile devices (smartphones, tablets), there is a tendency towards simulation *appification*, i.e., the implementation on such platforms of the usual simulation tools. While the performance of these platforms is nowadays impressive, they are undoubtedly less powerful than usual workstations. This imposes additional requirements to any realistic attempt to simulation democratization.

In this chapter we present our first attempt to democratize simulation by developing a platform that copes with the aforementioned requirements. We try to directly suppress the user time employed in the construction of the model. Instead, it is required that the model comes directly from physical reality, through computer vision techniques. As a proof of concept, we focus on car aerodynamics, and thus the geometry of the car (or prototype) will be captured by a commodity camera. No stereo or RGBD cameras are needed, in principle, even if their use is obviously possible and could eventually ease and improve the accuracy of the process. Previous approaches to this problem employ this type of cameras (Microsoft Kinect cameras, for instance, in [Harwood et al., 2018], that capture depth measures by means of an infrared laser).

Geometry of the objects to analyze will thus be acquired by computer vision, thus bypassing the need for time-consuming CAE pre-processing steps. Usually, the same device that captures geometry will serve to depict the results in an AR framework, but many different possibilities exist. Our approach makes use of feature-based Simultaneous Localization and Mapping (SLAM) techniques to acquire the geometry of the object with a single monocular camera, even though many modern smartphones are already equipped with stereoscopic cameras. In particular, we employ ORB-SLAM techniques [Mur-Artal et al., 2015]. These techniques allow us to capture selected points on the surface of the object, whose geometry will need nevertheless to be reconstructed from these points. In other words, a suitable parametrization of the shape of the object needs to be constructed. Previous works include the use of free-form deformation techniques [Lassila and Rozza, 2010] or the closely-related approach in [Umeta ni and Bickel, 2018].

Our approach to the problem is data-driven [Ghnatios et al., 2012] [Kirchdoerfer and Ortiz, 2016] [Ibañez et al., 2017] [Lam et al., 2017] [Peherstorfer and Willcox, 2015] [Latorre and Montáns, 2014]. This means that we employ manifold learning techniques [Lee and Verleysen, 2007] over a data base of previously computed CFD results. While there is a vast corps of literature on (linear) model reduction of flow problems [Díez et al., 2017] [Manzoni et al., 2012b] [Rozza et al., 2008] [Manzoni et al., 2012a] [Manzoni et al., 2012c] [Amsallem et al., 2010] [Amsallem and Farhat, 2008], these techniques rarely achieve real-time performance. For this particular application, by real-time we mean a code able to provide feedback response at some 30–60 Hz (i.e., the usual number of frames per second in modern smartphone cameras).

Other approaches, however, achieve interactivity by means of machine learning [Umeta ni and Bickel, 2018] or by model reduction [Bertram et al., 2018], but do not consider augmented reality output. On the other side of the spectrum, some previous studies have considered the *brute force* approach [Harwood et al., 2018]. This type of approach makes intensive use of GPUs capabilities in a Lattice Boltzmann framework and are not well suited, therefore, for their implementation on a mobile platform.

The employ of machine learning towards learning physical dynamics has gained considerable interest in recent times [Chang et al., 2016] [Bright et al., 2013] [Jeong et al., 2015]. In particular, deep learning architectures are of course tempting [Oliehoek, 2018]. In general, while model order reduction methods [Chinesta and Cueto, 2014] [Gonzalez et al., 2012] achieve important speedup by means of a considerable reduction in the number of degrees of freedom and also provide with important insight on the physics of the problem, deep learning architectures perform in nearly the opposite way [Kutz, 2017]. They are able to capture multiscale phenomena or transient features and, at the same time, are able to capture invariants of the system, but need for a tremendous effort to train the neural networks and, conversely, provide little or no interpretation of the results.

Some scientists have warned about the possible implications that the employ of machine learning on top of biased data could have [Ghosh, 2019].

Closely related, there is a strong research activity in the development of physics-aware artificial intelligence—in other words, machine learning approaches that satisfy first principles by construction [Swischuk et al., 2018] [Raissi et al., 2017] [González et al., 2018]. Our approach here is to employ, so to speak, the best of both worlds. Non-linear dimensionality reduction techniques—in particular, manifold learning—are able to substantially reduce the number of degrees of freedom of the problem, very much like classical, linear model order reduction. But, at the same time, they provide very useful insight on the non-linearity of the problem. As will be noticed from the results here presented, manifold learning techniques provide clear physical explanations on the structure of data, reduce the number of degrees of freedom and their dimensionality, and allow for very efficient computational approaches.

With all these ingredients—computer vision, machine (manifold) learning and augmented reality—we construct a reliable simulation platform that, even if we admit errors higher than those typical of state-of-the-art high-fidelity simulations, provide the user with a unique degree of interactivity and immediacy.

The chapter is structured as follows. In Section 6.1 we provide an overall description of the method we have developed. In Section 6.2 we give detailed insight on the construction of the database and its subsequent embedding in a lower dimensional space. Then, in Section 6.3 we briefly describe the state-of-the-art computer vision techniques employed to capture cars not being in the database. In Section 6.4 we describe how to interactively depict CFD results on top of the video stream of the particular car of interest. Finally, in Section 6.5 we analyze the accuracy of the results, computational cost, etc. The chapter is completed with some final comments about the interest of the proposed technique.

Some content of this chapter has been previously published in the following journal paper

- Badías, A., Curtit, S., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2019). An augmented reality platform for interactive aerodynamic design and analysis. *International Journal for Numerical Methods in Engineering*, vol. 120, no. 1, pp. 125-138.

6.1 Overall description of the method

Figure 6.1 depicts a sketch of the general pipeline of the method. The key ingredient is the employ of manifold learning techniques to *learn* car aerodynamics. This will allow us to avoid simulating in real time. Instead, we *interpolate*—in the right manifold space—existing CFD results (represented in the left column of Fig. 6.1). By obtaining a low-dimensional embedding of these results we will not only ensure to interpolate in the right space (instead of the Euclidean one) but also take profit of the non-linear reduced

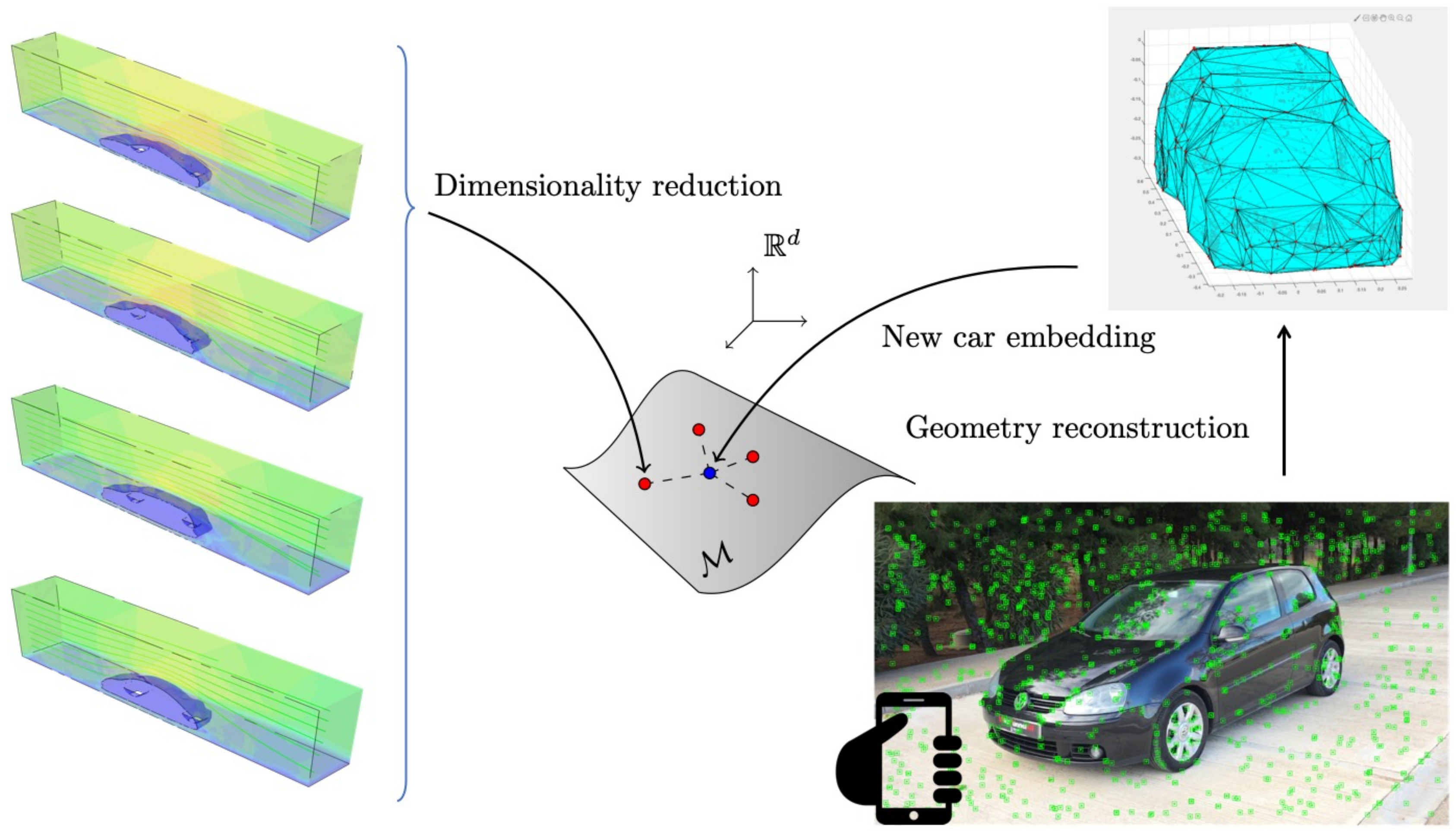


Figure 6.1: Schematic description of the method. We first construct a database of CFD results for known car geometries (left column). These results are embedded onto a low-dimensional manifold $\mathcal{M}$ by means of LLE techniques. Once a new car—not in the database—is captured with the camera, the features are first processed so as to obtain an approximate envelope of the new car geometry. This new geometry is then embedded onto the manifold (blue point) and its CFD results interpolated among its neighbors in the manifold (red points).

order modeling capabilities of Locally Linear Embedding techniques [Roweis and Saul, 2000], which will be thoroughly described in Section 6.2.

These results are obtained by usual CFD methodologies, starting from CAD descriptions of car bodies. Once unveiled, the manifold structure of these data, $\mathcal{M}$, will allow us to easily incorporate new designs, not previously analyzed (right column of Fig. 6.1). With the help of a standard smartphone we will capture the bodywork of a new car or prototype. To this end, it is necessary to film around the car, so as to capture its entire geometry. SLAM techniques [Mur-Artal et al., 2015] will allow us to detect special points in the body of the car called ORB features (green points in the bottom-right corner of Fig. 6.1) [Rublee et al., 2011].

After a reconstruction of the body of the car from these ORB points (top right corner of Fig. 6.1) we are in the position of embedding this new design onto the already available manifold of shapes $\mathcal{M}$. This will also provide us with the necessary weights so as to interpolate from surrounding CFD results in the database. This simple approach will allow us to obtain velocity maps, streamlines, etc. without the burden of meshing procedures, long CPU computations and subsequent post-processing.

Once the new results have been obtained, they are ready to be represented on the screen, on top of the video stream. For this to be possible, a *registration* process will be needed. By image registration we mean the process of finding the common system of reference between the video frames and the just obtained CFD results. So to speak, we need to *align* both so as to construct an augmented reality post-processing of these results.

These steps are detailed in the following sections. We begin by describing the steps towards the obtention of the car database.

6.2 Construction of a suitable database of CFD results

6.2.1 Parameterization of the shape

A crucial step in the machine learning of car aerodynamics is that of parameterizing shape. This issue has been deeply analyzed in [Umeta ni and Bickel, 2018], for instance. In the reduced order modeling literature there is another vast corps of literature, see, for instance, [Manzoni et al., 2012c] [Lassila and Rozza, 2010] [Ballarin et al., 2014] [Manzoni et al., 2012b] [Quarteroni and Rozza, 2003] for works employing freeform deformation techniques. Of course, usual CAD techniques based on the employ of NURBS or related approaches cannot be employed here, since the capture of shapes by computer vision is obliged in the tool we are developing. This problem shares many similarities, for instance, with the construction of patient avatars in the medical community [González et al., 2015] [Lassila et al., 2013] [Zimmer et al., 2015].

The original database was composed by object (`*.obj`) files. Prior to converting them to a manageable format, compatible with the computer vision system, a de-refinement of the bodywork of every analyzed model is mandatory, see Fig. 6.2. We employed the same de-refinement method as in [Umeta ni and Bickel, 2018]. In the examples analyzed herein, the intermediate level of refinement was chosen, that corresponds with the center figure in Fig. 6.2. Of course, the final accuracy of the method strongly depends on the level of detail to which the body of the car is represented. Further refinement leads of course to higher levels of accuracy.

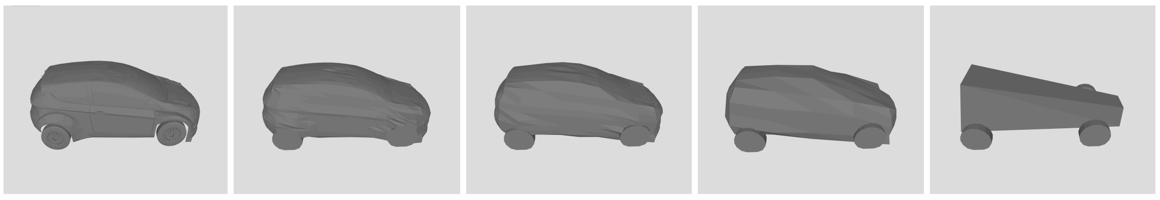


Figure 6.2: Model de-refinement for the Ford Fiesta example.

This de-refined geometry is then embedded into a grid of $140\times40\times20$ points uniformly distributed on a volume of $14\times5\times4$ cubic meters around each half car (we exploit here the symmetry of cars so as to include only one half of the bodywork in the database). While in some of our previous works [González et al., 2015] we employed a signed distance field to characterize the shape, in this work we have found that the employ of a presence function

$$\phi(\boldsymbol{x}) = \begin{cases} 1 \text{ if } \boldsymbol{x} \in \Omega, \\ 0 \text{ if } \boldsymbol{x} \notin \Omega, \end{cases} \tag{6.1}$$

where Ω represents the boundary of the car, rendered better results for the manifold learning procedure to be described next. In other words, we characterized each particular shape by a vector $\boldsymbol{z} \in \mathbb{R}^D$, where $D = 140 \times 40 \times 20 = 112000$, whose entries are the values of $\phi(\boldsymbol{x}_j)$ at each node of the grid, $\boldsymbol{x}_j$, $j = 1, \dots, 112000$.

6.2.2 Manifold learning

The key ingredient in our method is to hypothesize that car shapes describe a manifold structure whose precise form is to be found. To this end, we employ Locally Linear Embedding (LLE) techniques [Roweis and Saul, 2000]. In essence, LLE assumes that, given the fact that a manifold is homeomorphic to a flat space in a sufficiently small neighborhood of each point, each datum can be reasonably well approximated through a linear combination of its nearest neighbors—whose precise number M is to be fixed by the user—. Therefore, LLE assumes that for each $\boldsymbol{z}_m$ a linear reconstruction exists of the form

$$\boldsymbol{z}_m = \sum_{i\in\mathcal{S}_m} W_{mi}\boldsymbol{z}_i, \tag{6.2}$$

where W_{mi} are the unknown weights and $\mathcal{S}_m$ the set of the M-nearest neighbors of $\boldsymbol{z}_m$.

The search for the weights is done by minimizing the functional

$$\mathcal{F}(\boldsymbol{W}) = \sum_{m=1}^{M} \left\| \boldsymbol{z}_m - \sum_{i=1}^{M} W_{mi}\boldsymbol{z}_i \right\|^2 ,$$

where W_{mi} is zero if $\boldsymbol{z}_i \notin \mathcal{S}_m$, i.e., if cars i and m are not neighbors in the manifold of shapes.

LLE also hypothesizes that an embedding exists onto a lower-dimensional space $\mathbb{R}^d$, with $d \ll D$, such that the manifold structure is conserved. In other words, a new set of coordinates $\boldsymbol{\xi} \in \mathbb{R}^d$ exists for each car $\boldsymbol{z} \in \mathbb{R}^D$ and they can be found by minimizing a new functional

$$\mathcal{G}(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M) = \sum_{m=1}^{M} \left\| \boldsymbol{\xi}_m - \sum_{i=1}^{M} W_{mi}\boldsymbol{\xi}_i \right\|^2 ,$$

where the neighboring relation between points (i.e., the weights W_{mi}) are maintained. See Fig. 6.3 for a description of LLE.

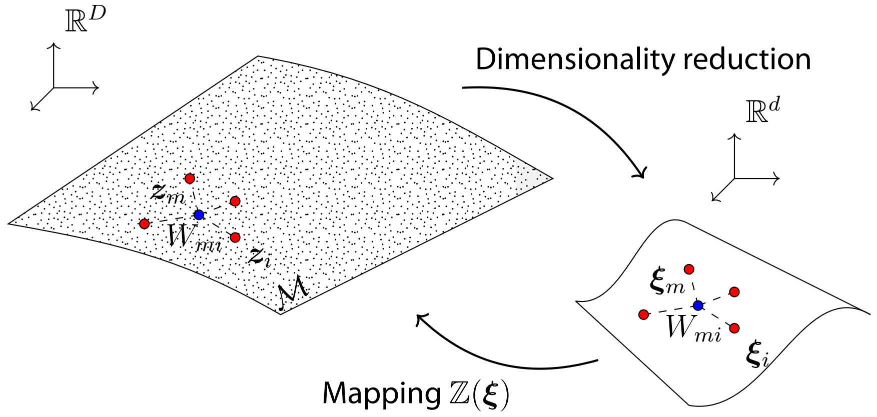


Figure 6.3: Hypothesis abut the existence of a manifold $\mathcal{M}$ on which the car shapes live. The small black dots represent the car database in a high-dimensional space $\mathbb{R}^D$. These data are then embedded onto a lower-dimensional manifold $\mathbb{R}^d$, with $d \ll D$. A new car not in the database (blue dot) will thus be interpolated from its neighboring cars (red points) through the weights W_{mi} provided by LLE, that define a mapping $\mathbb{Z}(\boldsymbol{\xi})$. By hypothesis, LLE assumes that the same neighborhood relationships hold in the high-dimensional manifold $\mathcal{M}$.

6.2.3 Our database

We analyzed a set of 80 different car bodies. All of them are subjected to a CFD analysis using commercial software. Fig. 6.4 shows a typical mesh of the environment of one of

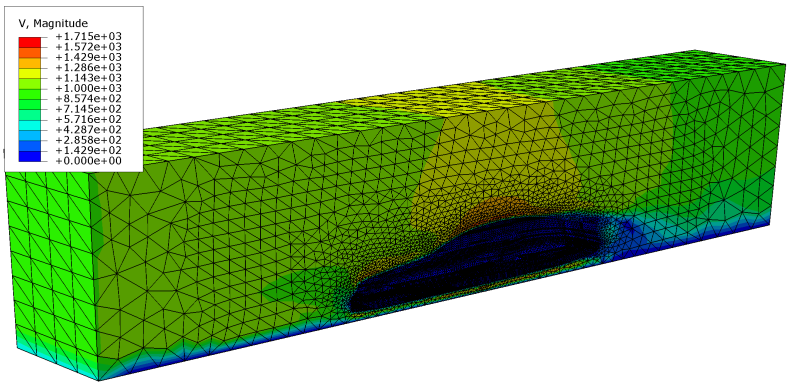


Figure 6.4: Contour plot of the velocity module in the mesh employed in the CFD calculations.

the models. All of them are subjected to a uniform velocity of 10 m/s at the leftmost part of the meshed volume. This gives us a Reynolds number on the order of 9000.

After the CFD computations, a projection of the velocity field onto the same grid mentioned in the Section 6.2.1 is made for subsequent interpolation purposes. Our cars are then manually classified into four different groups: large cars (Audi Q7, Hummer, Jeep Grand Cherokee, VW Touareg, among others), sport cars (Mercedes AMG GTS, BMW Z4, Audi R8, BMW i8, Porsche 911, ...), old cars (Pontiac firebird, Dodge Charger, Chevrolet Camaro, ...) and a fourth set of “various”, which comprises not easily classifiable models.

Our experiments showed that a range between $M = 6$ to 15 neighbors provided a good embedding onto $\mathbb{R}^2$. Figure 6.5 shows the embedding obtained with $M = 9$ neighbors, where it can be noticed how LLE clearly clusterizes sport cars apart from large, familiar cars, for instance. Old cars, whose shape does not follow any specific pattern, and those cars in the “various” group are somewhat mixed in the embedding procedure, as expected.

6.3 Computer vision for new car models

While the previous sections described the off-line work to construct the car database, in the following section we describe the on-line work done for the characterization of the new geometry and the subsequent interpolation of the results on the manifold of shapes.

After the database has been completed, and its dimensionality conveniently reduced, the pipeline of our platform faces the problem of reconstructing the geometry of a new car. In general, a new car or a new design will not be in the database, so we assume that no CAD description is available. Instead, the user will make use of a commodity camera (through a smartphone, for instance) to capture the new geometry.

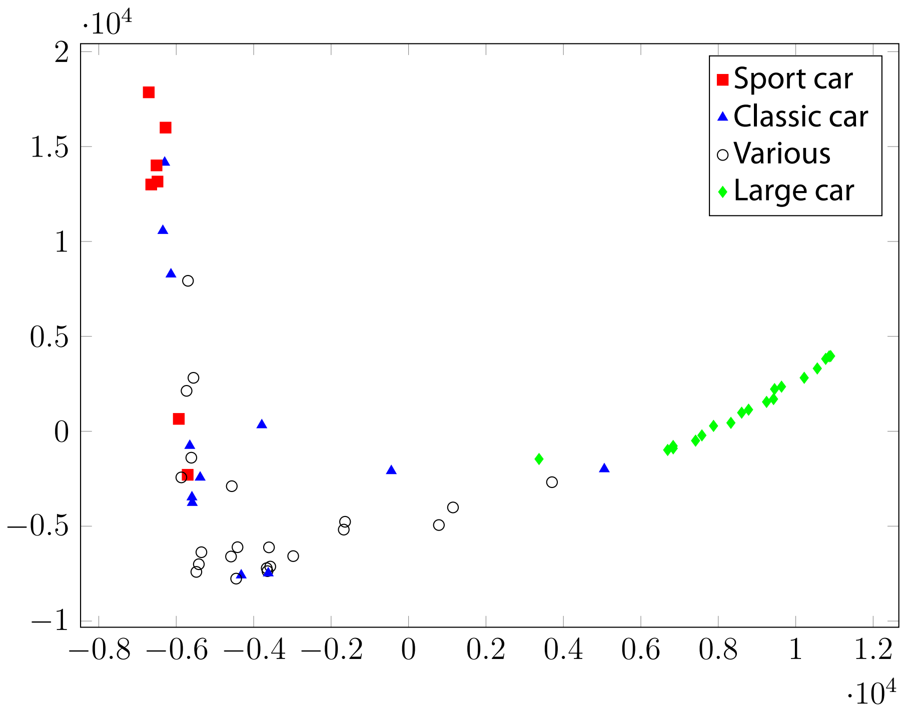


Figure 6.5: Results of the embedding of car geometries onto $\mathbb{R}^2$ by employing nine neighbors for each car. It can be noticed how LLE clusterizes sport cars far apart familiar (or "large") cars, while old cars and other models without a distinctive characteristic are somehow mixed.

6.3.1 Capturing the geometry from a video sequence

The problem of reconstructing a three dimensional geometry from a set of two-dimensional images is known in the robotics field as the *Structure from Motion*, SfM, problem [Szeliski, 2010] [Agudo et al., 2016] [Civera et al., 2009]. In essence, if we assume that the object to reconstruct is rigid—for the deformable case refer to [Badías et al., 2018] and references therein—, the problem is sketched in Fig. 6.6. The object is assumed to be attached to a fixed position in the frame of reference. The problem thus begins by determining the *pose*—location and orientation—of the camera.

The input image space is denoted by $\mathcal{I}_k \subset \mathbb{R}^2$, with $k = 1, \ldots, \mathrm{n}_{\text{images}}$, the number of available frames in the just captured video. We assume no batch approach in this platform but we do not consider the possibility of a video stream. In other words, the video has a finite number of frames.

Cameras perform a projection from three-dimensional, physical space to a two-dimensional pixel space. This projection is characterized by an *a priori* unknown camera operator denoted by $\Pi : \partial^*\Omega_t \to \mathcal{I}$ whose determination is known as the *calibration* of the camera. This projection can be expressed as $\Pi = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$, where $\boldsymbol{K}$ corresponds to the camera intrinsic matrix and $[\boldsymbol{R}|\boldsymbol{t}]$ represents the camera extrinsic matrix. We denote $\boldsymbol{x} = (x, y, z)^\top$

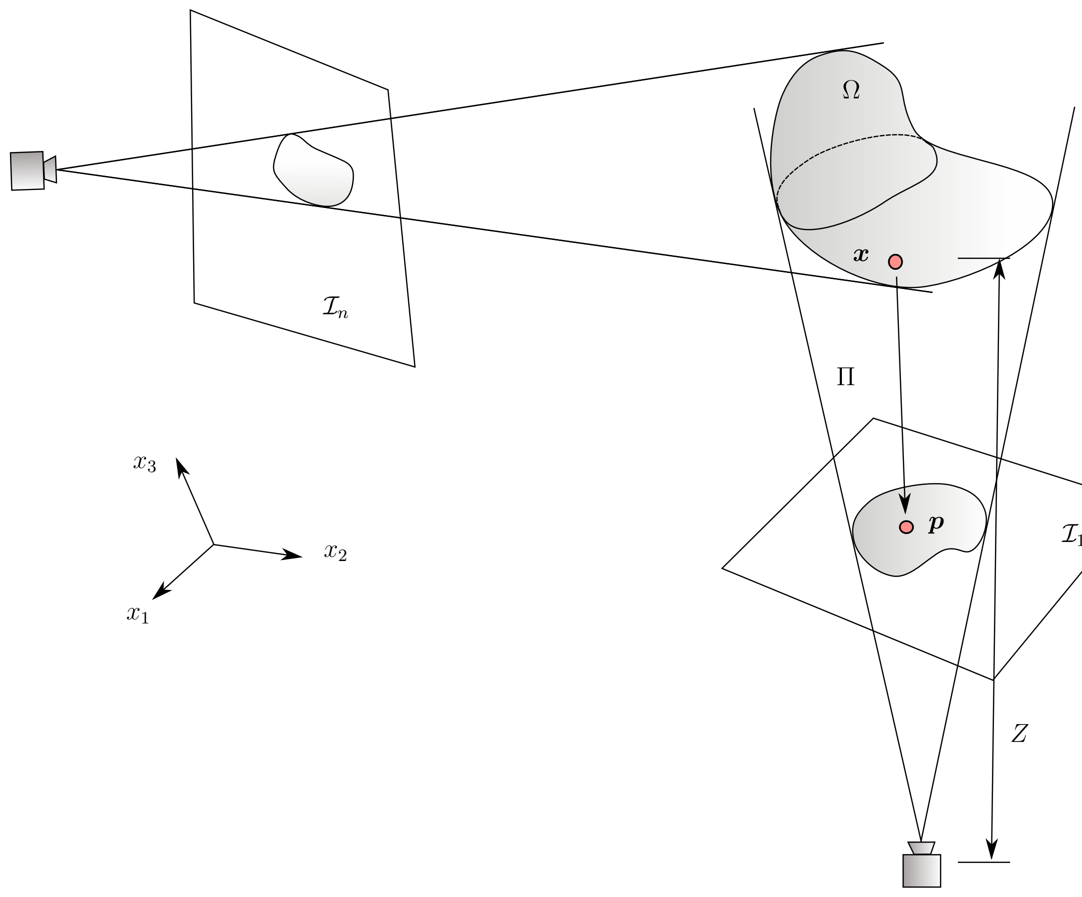


Figure 6.6: Sketch of the Structure from Motion problem. While the object is assumed to be rigid and attached to a fixed position in the frame of reference, the camera needs to move so as to reconstruct its three-dimensional geometry. Here, two video frames, namely $\mathcal{I}_1$ and $\mathcal{I}_n$ are shown, together with the camera projections of the solid. The projection Π of a particular point $\boldsymbol{x}$ on the visible surface of the solid gives rise to point $\boldsymbol{p}$ in the pixel space.

and $\boldsymbol{p} = (p_1, p_2)^\top$ and a scale factor s, so that,

$$\begin{bmatrix} p_1 \\ p_2 \\ s \end{bmatrix} = \lambda \begin{bmatrix} {}^{f}/_{d_x} & 0 & c_x \\ 0 & {}^{f}/_{d_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \boldsymbol{R} & | & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\boldsymbol{x}^c}, \tag{6.3}$$

where $\boldsymbol{x}^c$ represents the coordinates of $\boldsymbol{x}$ in the camera system (a moving frame of reference attached to the camera, not the world reference frame). $\partial^*\Omega$ denotes the visible part of the boundary of the object of interest, Ω.

Two transformations exist: first, the camera extrinsic matrix transforms three-dimensional physical points in the world system of reference into three-dimensional points in the camera moving frame of reference. This transformation is equivalent to the composition of a rotation ($\boldsymbol{R}$) and translation ($\boldsymbol{t}$) between the corresponding points. On the other hand, the camera intrinsic matrix $\boldsymbol{K}$ contains information about the projection centre (c_x, c_y), the pixel size (d_x, d_y) and the focal distance, f, that maps three-dimensional camera points into two-dimensional pixel coordinates. Finally, λ represents a scale factor that takes into account the fact that, with a single picture, an indeterminacy exists that makes it impossible to discern between the projection of a big object positioned far from the objective or a small one at a closer position.

Therefore, to determine the camera pose and the location of a rigid object we need more than one single image. To solve this indeterminacy, SfM algorithms minimize the *reprojection error*. In other words, SfM minimize the geometric error corresponding to the image distance between a projected point and a measured one [Hartley and Zisserman, 2003]. As sketched in Fig. 6.6, the camera performs a projection of the form

$$\boldsymbol{p} = \Pi(\boldsymbol{x}),$$

with $\boldsymbol{p} \in \mathbb{R}^2$ and $\boldsymbol{x} \in \mathbb{R}^3$. Note that we do not know the exact value of $\boldsymbol{x}$. Well established geometrical analysis will provide an estimation $\hat{\boldsymbol{x}}$ of the true position of the point [Bradski and Kaehler, 2008] [Hartley and Zisserman, 2003]. If we re-project this estimate for each frame by

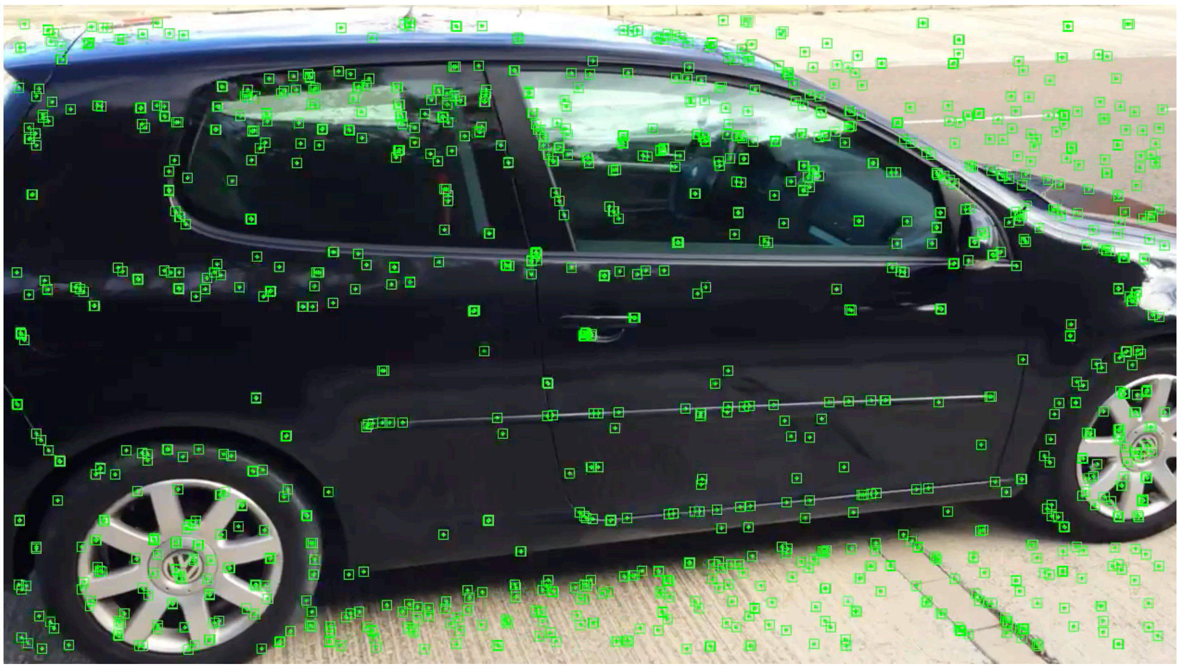
$$\hat{\boldsymbol{p}} = \Pi(\hat{\boldsymbol{x}}),$$

the sought reprojection error takes the form $d(\boldsymbol{p}, \hat{\boldsymbol{p}})$.

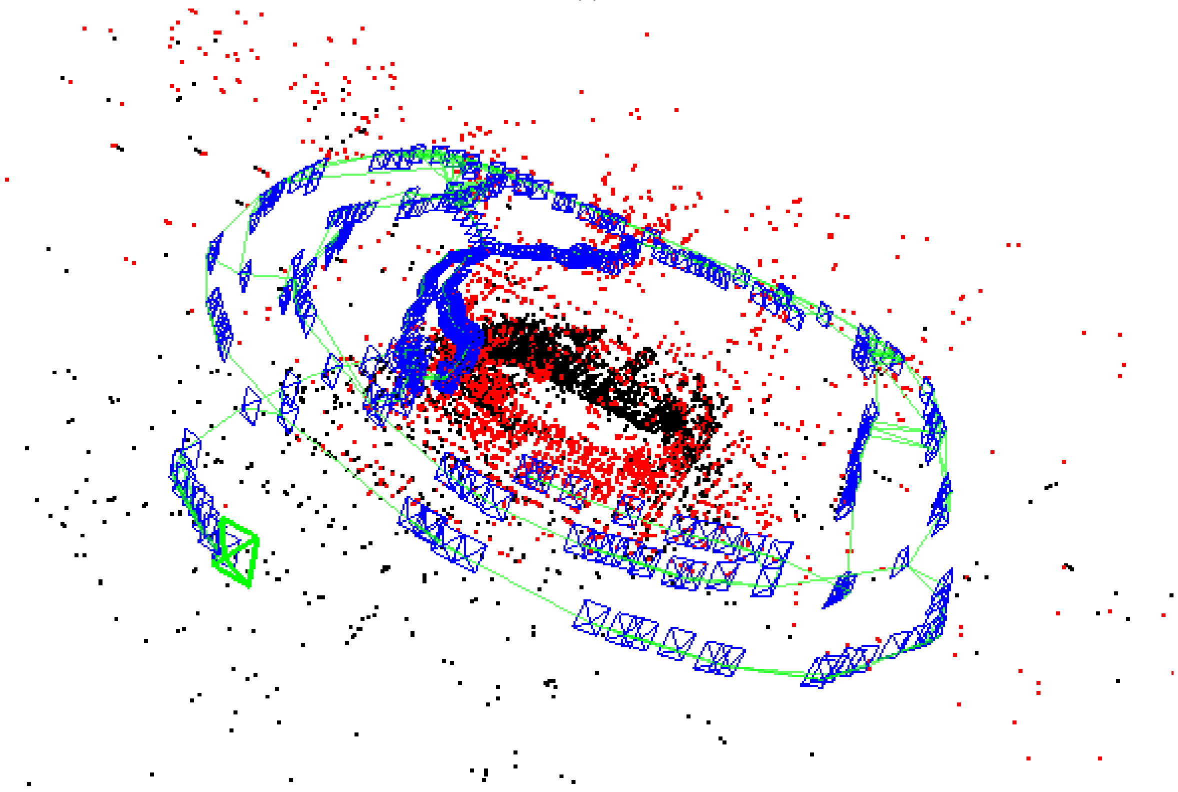
This analysis is often made with respect to some marked points usually known as *fiducial markers*. In general, this analysis is greatly simplified if the object to reconstruct can be labelled at some points with the help of some visible sticker, for instance. However, for our purpose this is not feasible. The tracked points $\boldsymbol{x}$ must be selected by the algorithm itself. To this end, we employ a method based on *features*, the so-called Oriented FAST and rotated BRIEF (ORB) method [Rublee et al., 2011]. In fact, we employ the so-called ORB-based Simultaneous Localization and Mapping (ORB-SLAM) method, able to solve the SfM problem while at the same time it constructs a map of the environment [Mur-Artal et al., 2015]. These ORB features are the green points shown in Fig. 6.7.

6.3.2 Projection onto the database

The ORB features just captured constitute a cloud of points in the boundary of the sought geometry (see Fig. 6.8). To determine the best possible approximation to the sought geometry we employ α-shapes [Edelsbrunner and Mücke, 1994]. Essentially, α-shapes reconstruct a volume given a cloud of points on its boundary with the help of one user-defined parameter, α, that represents the coarsest level of detail in the reconstructed geometry. Basically, α is on the order of the distance between recognized ORB features.



(a)



(b)

Figure 6.7: (a) An example of the captured ORB features for the VW Golf model. (b) A reconstruction of the camera movements in the video. Blue pyramids indicate the position and orientation (pose) of the camera. Dots represent the ORB's detected by the SLAM methodology, that belong both to the car and its surroundings.

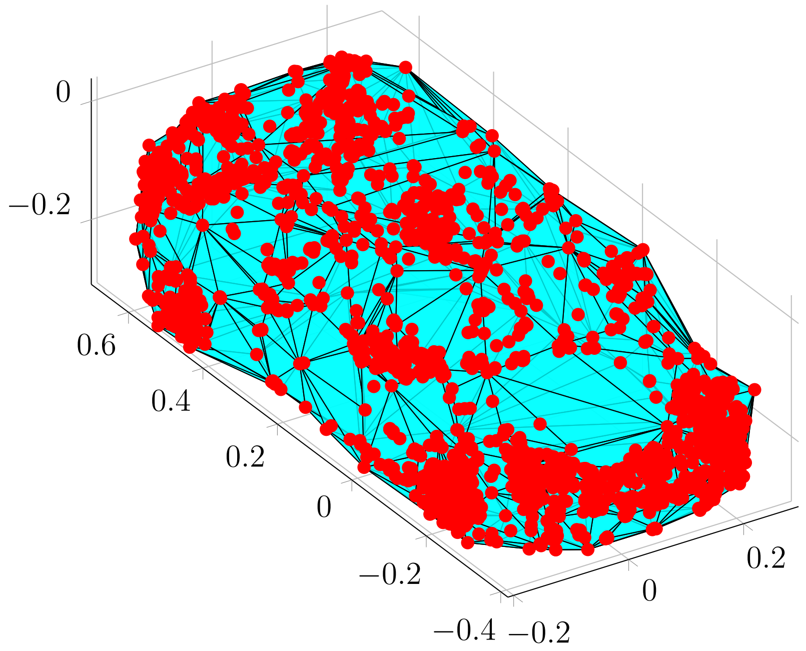


Figure 6.8: Reconstruction of the VW Golf model. Red points represent the detected ORB features. The blue volume is reconstructed by an α-shape algorithm.

Of course, a stereo or RGBD camera would provide a much more detailed geometry representation. We analyze, however, the case of deployed hardware, and consider that a simple smartphone or tablet is enough to obtain a reasonable representation of each car bodywork. With the just reconstructed geometry, we embed it in the same grid employed to construct the car database and evaluate the presence function $\phi(\boldsymbol{x})$ at each of the nodes of the grid, $\boldsymbol{x}_j$, $j = 1, \ldots, D$, see Fig. 6.9. This will provide a new vector $\boldsymbol{z}^{\text{new}} = [\phi(\boldsymbol{x}_1) \cdots \phi(\boldsymbol{x}_D)]^\top$ for the geometry at hand.

This new vector $\boldsymbol{z}^{\text{new}}$ can easily be embedded on the manifold without the need of re-computing every weight, by invoking incremental LLE algorithms, see [Schuon et al., 2008], for instance. This will provide the new weights W_i, $i = 1, \ldots, M$.

The key ingredient in our method is the assumption that the velocity field (for sufficiently low Re numbers) of a new car can be obtained in an accuracte enough manner by linear interpolation among the neighboring models in the manifold. In other words, that the new velocity field is approximated by

$$\boldsymbol{v}^{\text{new}}(\boldsymbol{x}_j) \approx \sum_{i=1}^{M} W_i \boldsymbol{v}_i(\boldsymbol{x}_j), \tag{6.4}$$

with $\boldsymbol{v}_i(\boldsymbol{x}_j)$ the velocity field for the i-th neighboring car model at the j-th node of the grid. This assumption, which may seem too restrictive, has demonstrated already to provide reasonably accurate results in a completely different context, but in a similar procedure, see [González et al., 2015].

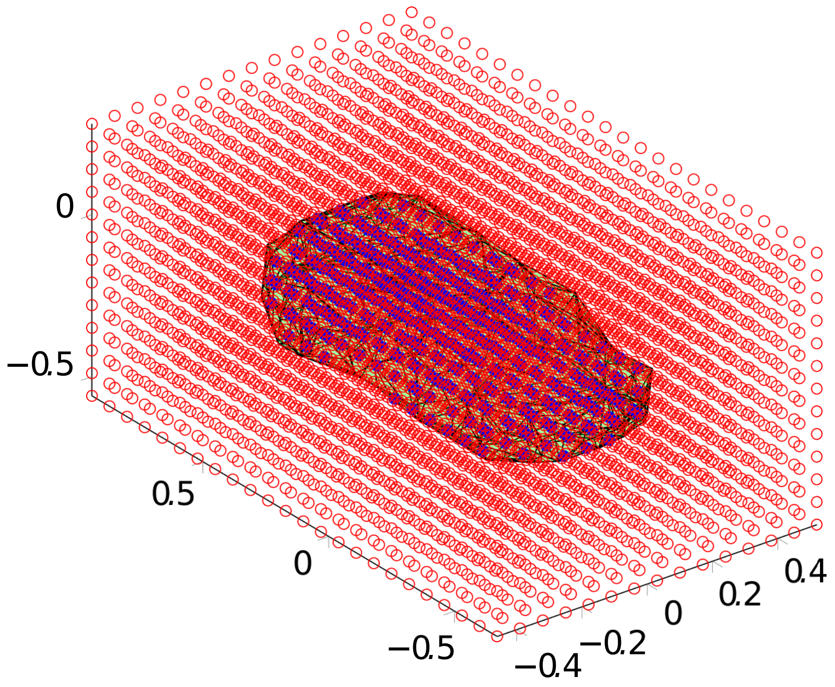


Figure 6.9: The car's geometry is then embedded on a grid and the presence function, Eq. (6.1), evaluated at each of its nodes. Red nodes are evaluated to zero (they fall outside of the car) while blue ones are evaluated to one (they fall inside). Note that the grid's resolution is lower than that actually employed $(140 \times 40 \times 20)$ just to make the figure visible.

6.4 Augmented Reality output

6.4.1 Car identification

ORB-SLAM is probably the best technique to reconstruct static 3D scenes in a sparse way, if we think of employing a commodity camera. We can estimate the 3D position $\boldsymbol{x}(x, y, z)$ of the most relevant points of a scene, extracted by using the feature descriptor ORB. This means that we need a relatively low computational power to work, it does not require the use of graphical acceleration by GPU, at the cost of reducing the number of observed points. With this system we do not expect to obtain a dense cloud of points, as it happens with other acquisition systems (such as RGB-D devices or stereo systems), but to be able to take very precise measurements in real time with little computational cost. It is also important to point out that densification techniques can be applied a posteriori to obtain a denser 3D point cloud, such as [Mahmoud et al., 2019].

Since we are using a monocular system (like the camera we can have in our smartphone) we can not recover the scale of the model without knowing any of the real dimensions of the objects (an RGB-D system or stereo camera could give us this information, but a standard user does not usually have this type of technology). Nor can we identify the direction of gravity (it could be done with an *Inertial Measurement Unit* or IMU), which complicates the estimation of the location or *pose* of the car in space. Our objective is to depend only on the visual system, obtaining the information of the scene from only a monocular camera.

We start, therefore, from a set of 3D points estimated in the scene, where we need to identify the geometry of a car. For this we assume that the user has surrounded a real car with its monocular system, where the car appears relatively centered in the cloud of points. Alg. 6.1 collects the algorithm used to identify, segment and locate the position of the car from the point cloud.

Algorithm 6.1: Pseudo-code of the car detection process.

```
function CarDetection (input SLAMCloud, output cleanCloud)
{
    densityPointCulling(SLAMCloud);
    centroid = geometryAnalysis(SLAMCloud);
    centroidPointCulling(SLAMCloud);
    floor = planeEstimationRANSAC(SLAMCloud);
    floorRemoving(SLAMCloud);
    PrincipalDirections = PrincipalComponentAnalysis(SLAMCloud);
    noiseRemoving(SLAMCloud);
    boxAdjustment(SLAMCloud);
    [S,R,T] = GeometricalTransformation(SLAMCloud, unitaryCarCloud);
    cleanCloud = linearTransformation(S,R,T,SLAMCloud);
}
```

The first task of the algorithm is a point culling process depending on the local density surrounding any point of the cloud. We use this type of cleaning methods because sparse SLAM methods can add noise in the triangulation step. The use of a pyramid of image scales may also encourage this type of defects.

Next we estimate the centroid of the cleared point cloud, which allows us to locate where the most relevant object in the scene is located. After this, we eliminate those points that are further away from 95% of the distances of all points from the centroid. This allows us to work without the need to add adjustment parameters.

We then estimate the plane of the ground, because we assume that the car is located on a large plane with texture. So we estimate the main floor plane, and remove the points contained in that plane.

A principal direction analysis is then applied to estimate the predominant direction of the vehicle, that with the centroid and the gravity vector (estimated from the floor plane) allow us to fix the coordinates origin of the car.

The last step is the scaling. A noise removing process is again applied based in the histogram of positions to remove areas of isolated points. Then the car is inserted within a box to facilitate the estimation of scale and geometric transformations, which are then applied to move the car to the origin of coordinates and scale it with a unitary scale. We can do this because although we do not recover the scale, if the camera is well calibrated, the system is able to obtain a very precise *3D aspect ratio* between the 3 dimensions. In other words, even if the object is in another scale, moved and oriented in another direction, it is not necessary to apply different scales in the X, Y or Z axes, but a common scale of type $\boldsymbol{x}_{\text{SC}}(x, y, z) = \lambda(\boldsymbol{x}_{\text{ORB}}(x, y, z))$, where $\boldsymbol{x}_{\text{SC}}$ are the 3D positions of the geometry with *scale correction* and $\boldsymbol{x}_{\text{ORB}}(x, y, z)$ are the initial 3D coordinates estimated by the SLAM system. This allows us to compare the estimated car with the database of the rest of the cars, where we have already precomputed the aerodynamic solution.

It is important to note that we could also have used modern techniques of recognition of car pose through images only, employing learning techniques [Mousavian et al., 2017], [Hödlmoser et al., 2012]. However, we think that it is possible that they do not reach the level of precision we expect for the geometry of a car. In addition, with our method we avoid the labeling and training phase. Working with image masks and then estimating the position of the 3D points by triangulation may involve similar noise problems, and some points could be located outside the real geometry of the car. Therefore, we should apply some filtering techniques again, as we have done in our proposed method. The tests we have done show that, although we use classic techniques in our method, they work in a robust way.

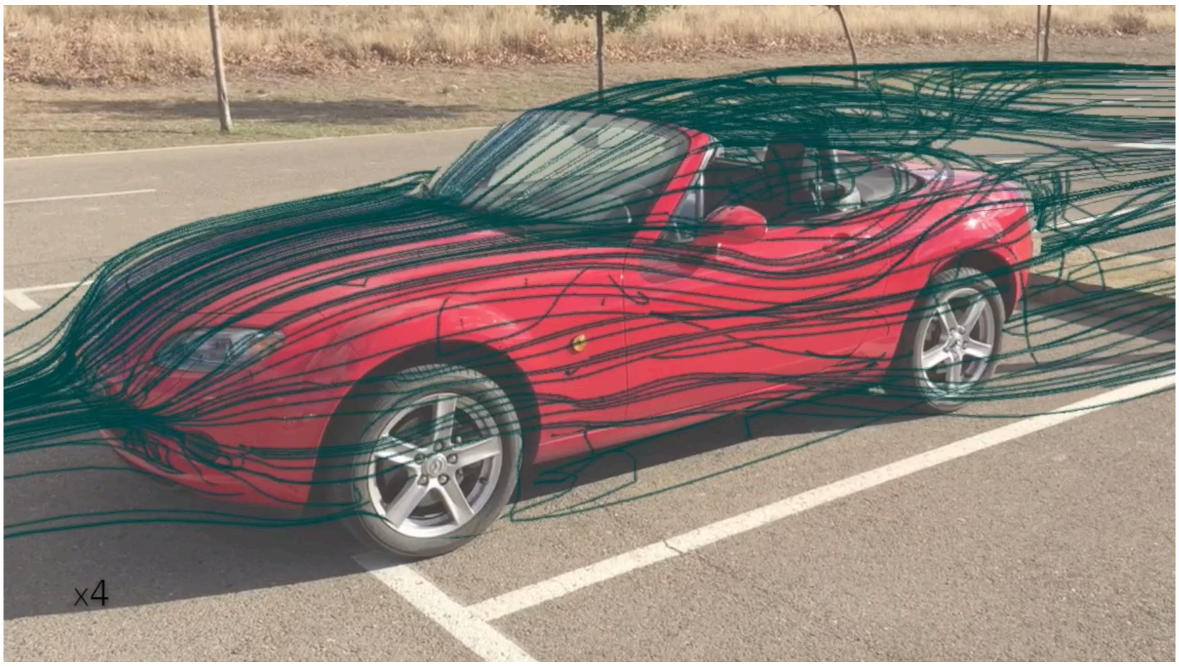
6.4.2 Final appearance

The previous identification algorithm provides very valuable information for the final augmentation of the video. In particular, having the reconstructed three-dimensional geometry is a necessary step for the computation of occlusion culling in the results, i.e., disabling objects not seen by the camera to appear in the image. All the steps in the augmentation process are implemented in OpenGL[1]. The final appearance of the augmented results is shown in Fig. 6.10.

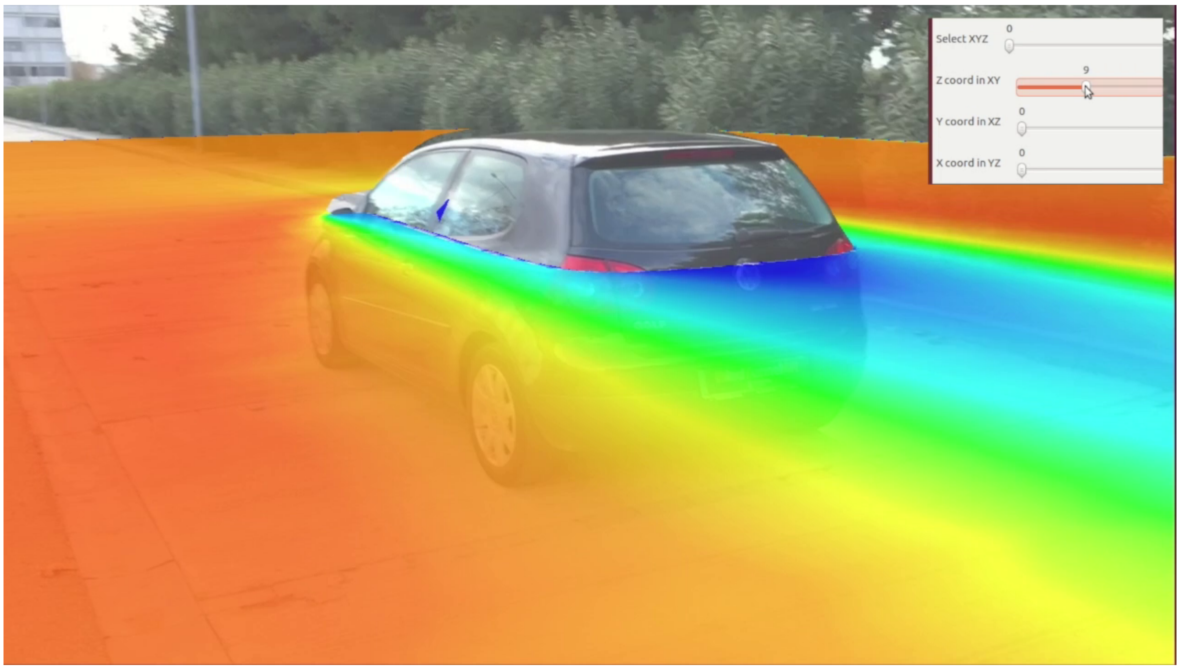
6.5 Analysis of the results

In this section we analyze the performance of the just presented methodology. Particular emphasis is made to the computational cost and real-time compliance of the on-line computations. A supplemental video showing both the process and the result of the proposed methodology can be downloaded from `https://youtu.be/Azm4E3Y7jS4`.

[1] OpenGL Library, `https://www.opengl.org/`



(a)



Select XYZ 0

Z coord in XY 9

Y coord in XZ 0

X coord in YZ 0

(b)

Figure 6.10: Final appearance of the augmented video (a) Streamtraces can be plotted on top of the reconstructed geometry. (b) Other possibilities, such as pressure or velocity contour plots, also exist.

6.5.1 Computer vision

The first part of the on-line procedure is the capture of the bodywork geometry through ORB-SLAM techniques. This algorithm runs in real time (i.e., it attains 30–60 frames per second) without any problem on a standard laptop. We employ the standard implementation in [Mur-Artal and Tardós, 2017], whose code is freely available.

6.5.2 Reconstruction of the geometry

Given the set of points captured in the process described in the previous section, the reconstruction of the geometry of the bodywork (i.e., the determination of the approximate geometry shown in Fig. 6.8) takes on average 0.30 seconds in a MacBook Pro 2016 computer equipped with 16 Gb RAM running an Intel Core i7 processor at 3.3 GHz.

6.5.3 Embedding of the new model

After the reconstruction of the geometry, the new model or prototype should be embedded onto the manifold of shapes. This allows us to obtain the interpolation weights $\boldsymbol{W}_i$, see Eq. (6.4). This process takes, on average, less than a second. Previous works have also reported how fast the LLE algorithm is, with some 500 different geometries running in about one second [González et al., 2015].

6.5.4 Car identification

The procedure sketched in Alg. 6.1 runs in real time on a standard laptop.

6.5.5 Error analysis

To test the accuracy of the LLE embedding onto the shape manifold, we deliberately kept five models out of the database. They were employed as ground truth, and their direct numerical simulation was compared to the result obtained by the interpolation of their velocity field, see Eq. (6.4). L_2-norm errors ranged from 1.02% for the Hummer model to 0.56% for a classic car labelled as *sedan 4*, see Fig. 6.11.

These results indicate that, for the scale of the grid employed in the characterization of geometry, many models provide very similar results. These velocity fields are, therefore, easy to interpolate. A highly refined mesh—which is not necessary for the type of applications we are envisaging here—would maybe report more important differences in the velocity field among different models.

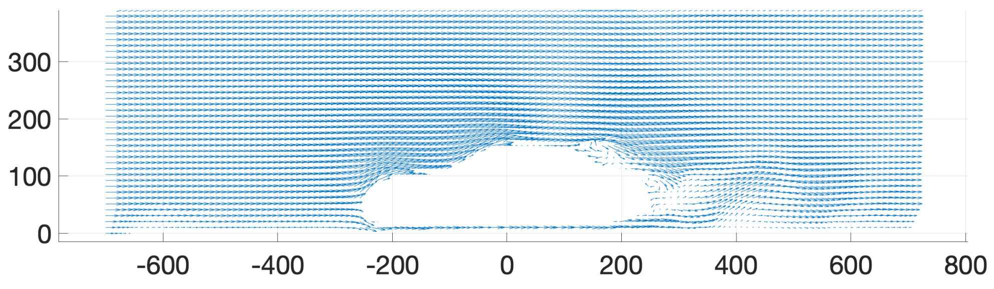


Figure 6.11: Interpolated velocity field for the *sedan 4* model. View along the symmetry plane.

6.6 Chapter Conclusions

We have presented a tool able to capture and reconstruct, with the help of a commodity camera, the geometry of a three-dimensional object—we have studied mainly cars—and to perform simple but reasonably accurate CFD predictions of the flow around the object for interactive design. This technique employs state-of-the-art structure from motion techniques for the geometrical reconstruction and manifold learning techniques to construct inexpensive predictions of the velocity and pressure fields around the object.

The advantage of such a platform is that what we obtain is actually an *appification* of CFD software. This tendency in our community is having tremendous commercial implications and is changing the way we see simulation. Here, the user needs only a smartphone to capture the geometry of the object to analyze, so as to obtain timely information about the physical consequences of his or her decisions. Artistic designers are thus invited to incorporate CFD to their workflow in a seamless way and with a minimum of knowledge in engineering—something that could be risky, nevertheless.

The combination of computer vision techniques, artificial intelligence (here, manifold learning) and augmented reality gives rise to a platform that changes the way we interact with simulation software. At the price of a little loss in accuracy with respect to existing, high-fidelity software, we obtain real-time responses to our query in a very short period of time, thus enabling decision making and an easy interplay between designers and engineers.

Chapter 7

Conclusions

The present chapter contains the general conclusions of the thesis, the proposed future work, the specific contributions and the publications resulting from the research work.

7.1 Concluding Remarks

This thesis lays the foundations of simulated reality, understood as an evolution of the augmented reality systems where information of the physical behaviour of objects is added. This information comes from simulation based engineering science, so these physical interactions are described with a high degree of evidence.

In this work, all the necessary parts of a simulated reality system have been covered: we solved the different equations of the mechanical models of each problem in order to obtain the information to be visualised (engineering simulations), we have worked on the acquisition medium (data extraction from image sequences) and on the complete visualization of the results (computer graphics). It is true that different implementations along the three areas were done to achieve a complete fusion, but we must remember that this doctoral thesis is included in the field of mechanical engineering, and it is in this field where the contributions have been made.

Throughout the whole document we have tried to show the complexity of computing simulations based on real physics. For example, we have applied differential equations to model the behavior of deformable solids and to estimate the aerodynamics of vehicles. As we have already seen, these complex tasks cannot be computed in real time, but thanks to the help of data science using MOR methods, we can estimate the variability of these data to compress the solution and work more efficiently.

In this work we are focused on the machine learning techniques that are based on the controlled projection of data on spaces of lower dimensionality. We have used both linear and non-linear projections, but they are not very complex mappings as those that deep neural networks can create. We use expansions where a space of smaller dimensionality to represent the data in a more optimal way has to be found. We also expect to move through these dimensions, which may not have physical meaning, but improve efficiency since they represent the true variability of the solution. The techniques used in this thesis are mainly based on the statistical tool PCA (Principal Component Analysis), although some non-linear versions have been also used: k-PCA (Kernel Principal Component Analysis) and LLE (Locally Linear Embedding). Another technique that has great importance in this thesis is the *a priori* version of PCA, known as PGD (Proper Generalized Decomposition), although with notable differences such as the non-imposition of orthogonality in the vectors of the reduced basis.

Sometimes, multiparametric solutions may have strong variabilities in local areas. It means that some approaches like PGD may have problems in estimating the solution with reasonably small error rates. To deal with this obstacle, we have decided to apply local approaches by reducing the domains where PGD is applied, so that small domains are used for areas with very variable and complex solutions, and larger domains are used to approximate those parts where the solution has smoother variabilities. We have proposed three methods to define the size of these local domains. The first of them (Cℓ-PGD) consists of making constant size partitions, the second method (MNOℓ-PGD) selects the local size to obtain a constant number of modes and the third method ($k\ell$-PGD) uses the information coming from the non-linear transformation that k-PCA provides to define the size of the subdomains. The results show that in solutions with abrupt localized variabilities up to 2 dimensions, the MNOℓ-PGD (modes number optimization) method is the most favorable, since it maintains a constant number of modes across all domains, which translates into lower storage costs. However, in problems of more than 2 dimensions, where there are also regions of abrupt variability combined with other softer areas, we have seen that the $k\ell$-PGD method that estimates the complexity from the k-PCA transformation is the method that provides the best results. According to the examples, the technique k-PCA is able to capture the multidimensional variability after the application of a non-linear mapping, and project it on the chosen number of dimensions, where tools such as the Hausdorff metric are very useful to estimate distances between different data snapshots. It is important to note that the kernel parameters require a slight adjustment to adapt the transformation applied to the optimal range of values. In our experiments we have used a Gaussian kernel.

Usually, our desire is to reduce the dimensionality of the multiparametric problem and estimate the reduced *manifold* in which the solution certainly lives. This implies that modifying one of the reduced parameter coordinates $\boldsymbol{\xi}$ (with probably non-physical dimensions) produces changes in the original solution, depending on the vector of parameters $\boldsymbol{\mu}$. In other words, the reduced dimensions are a function of the original dimensions (in case linear projection is used, the reduced parameters are linear combinations of the original parameters). Having said that, sometimes the variability of the system is so complex that a linear projection does not bring great advantages as we have seen in last paragraph. This opens the door to non-linear projections, but as we might expect, there are endless possibilities if we want to apply non-linear mappings to project the problem. A first example is the local implementation of PGD, but here we want to explore non-linear methods of projection. Some works [Brunton et al., 2016] aim to select the projection bases from non-linear combinations of the original solutions, building a library of functions that are weighted to approximate the original solution by minimizing a defined functional. Another option is, instead of reducing the dimensionality with complex non-linear projections, to increase the dimensionality of the solution in order to apply linear projections on multidimensional problems that are expected to be more easily separable. This is the principle on which k-PCA or other kernel-based methods such as SVM (Support Vector Machines) are based to improve separability in the classification of patterns.

Moving on with nonlinear projection techniques, LLE has been used in this thesis to estimate the aerodynamic behaviour of vehicles based on a previously computed set of simulations. First, LLE has been used to cluster the geometry of a vehicle based on other similar geometries, obtaining a matrix of weights that score the similarity of each vehicle with its neighbors. After this, this set of weights has been used to estimate the aerodynamic behaviour of a car travelling at constant speed. The result indicates that it is possible to estimate these results with a high degree of accuracy, although future approximations can be carried out by performing a more exhaustive analysis with more detailed geometric meshes.

From the point of view of capturing information from images, two types of devices have been used throughout the thesis: monocular and stereo cameras. As already seen in Chapter 2, monocular cameras add more complexity to the method because the position of the objects needs to be triangulated from different camera locations in the scene. A first approach to the problem has been carried out using fiducial markers and homographies (beam examples in Chapter 4), where the deformations are limited in a plane. It allowed us to estimate in a simple way the position of the load knowing the size of the beams. Subsequently, to estimate the position of the camera at any instant of time while scanning a less controlled scene we used ORB-SLAM2 [Mur-Artal and Tardós, 2017], a valuable tool that has been used in this thesis to simplify the complexity of measuring data from images. In static problems, such as the visualization of the aerodynamics on the vehicle, the standard implementation of ORB-SLAM2 has been directly used. We developed a registration and alignment method for each problem, where ad-hoc solutions had to be implemented, but the core of ORB-SLAM2 remained intact. However, in deformable problems, it has been necessary to modify the original code to track the deformable objects according to the deformations coming from the parametric models. This implementation can be considered as a contribution of this thesis, where all the variational formulation of the problem has been developed, taking into account that those derivative terms of each parameter must be projected on the image plane to obtain a functional that minimizes the reprojection error of the image. In the case of the stereo system, the SDK provided by the manufacturer solves the SLAM problem, producing a robust system including an inertial sensor that allows a direct use.

Regarding the graphic computing, state-of-the-art techniques have been used to visualize the results without developing any novel algorithm. But it has been necessary a learning period to manage the visualization pipeline, which has given us a valuable knowledge that we did not have previously. It helped us to understand some basic concepts such as *buffers* and *shaders*. These concepts allowed us to properly estimate the occlusions between real and virtual objects, providing a more realistic behavior and interaciont with the user.

The fusion of all of the techniques described above has allowed us to estimate the displacements of deformable objects and plot the stress states that objects suffered. Several examples have been implemented such as the cantilever and double-sided beams, the rubber piece deformation problem, the direct contact with the rabbit and the dragon and the visualization of the aerodynamics on vehicles.

From the point of view of data acquisition, we have assumed that measurement errors are negligible and have already been filtered by the visual algorithms using RANSAC [Fischler and Bolles, 1981] techniques. This purely deterministic approach has given good results in all of the examples shown since the models have been created specifically for each example. However, deviations of two types may appear: those due to measurement errors and deviations due to errors in the proposed model. Although in this thesis this type of corrections do not appear, we have worked in this type of implementations. In order to correct measurement errors, probabilistic methods can be applied to take into account data distribution and noise. A work applied to mechanical systems has already been published [González et al., 2017], although it has not been implemented to work with video sequences. On the other hand, in case model corrections were necessary (i.e., on the behavior law of deformable objects), the implementation of *Hybrid Twin* [Chinesta et al., 2018] can be added to correct the estimations. A first approach applied to images has also been implemented in [Moya et al., 2019], although more in-depth developments are foreseen.

7.2 Thesis Contributions

The fundamental contribution of this thesis is the development, for the first time, of a mixed reality methodology that simulates the interaction with solids of any type of behavior, with special interest in hyperelasticity. The developed methodology operates in real time (30-60 fps) which facilitates a very fluid interaction with the user.

In addition, in an equally novel way, a method to simulate contact mechanics has been developed to visualize, with a high degree of realism, the interaction between real and virtual deformable solids. The system allows the user to observe the values of the deformations and stresses suffered by the objects in real time. This same methodology is also applied to the fluid-solid interaction.

In addition to the above, we must add other technical contributions that have made it possible to achieve the aforementioned:

- Creation of a mixed reality framework that uses reduced models for real time applications, using both *a priori* and *a posteriori* methods, depending on the suitability of each method to the type of problem.
- Implementation of a non-linear method to improve the separability of the solution and reduce the cost of memory to store the data. This method consisted in a local approach of the PGD method, imposing the fulfillment of the boundary conditions between subdomains.
- Development of 3 different techniques to define the size of each local subdomain to subsequently apply the PGD method, highlighting the suitability of each technique according to the problem.
- Estimation of the deformation and load application point in deformable objects using homographies and monocular systems through the inverse problem on solutions projected in spaces of reduced dimensionality.
- Ad-hoc implementations of the registration process between CAD models and SLAM point clouds with high noise density.
- Estimation of the deformation and load application point in deformable objects using ORB-SLAM and monocular systems by solving the inverse problem on solutions projected in spaces of reduced dimensionality, from their interaction with rigid objects that move around the scene.
- Implementation of a deformable solution of the ORB-SLAM2 algorithm tracking any particular object point using image features taking into account the information projected in a space of reduced dimensionality.
- Estimation of the contact with occlusions between real and virtual objects in a mixed reality environment using a stereo system, where the deformations are based on true physics so that we can visualize the stress field of deformed objects with a high degree of fidelity.
- Visualizer of the aerodynamic behaviour of a vehicle using ORB-SLAM and a monocular camera allowing its application on a real vehicle from precomputed data.
- Approximation of a car aerodynamics by means of non-linear projection techniques from a previously calculated database of several vehicles, since it is not possible to carry out interpolations in high-dimensional space.

7.3 Future Work

Regarding the future work, there are many ideas that have emerged after the development of the work. In order to delimit and arrange these ideas, we are including them according to the order of chapters of the present document (without taking into account Chapters 1, 2 and 7).

- Chapter 3: we proposed three methods for defining the boundaries of local subdomains, but of course, this can be deepened by developing new methods for setting such boundaries, something that is no longer easy when dealing with large-scale problems. Data tend to be sparse as the number of dimensions increases, so the tools developed may require different treatment when dealing with many dimensions.

  On the other hand, as we have already seen with the application of k-PCA, some complex problems can improve the separability of the solution if they are projected onto a larger dimensional space, where data can be linearly separable [Hofmann, 2006]. That is the principle of kernel-based methods, and perhaps it could be applied directly to the PGD method to obtain its homonym *k-PGD* (kernel-Proper Generalized Decomposition), where the differential equation is solved in a space of greater dimensionality and the separability of the solution is, hopefully, improved.

- Chapter 4: we propose as future work the use of dense reconstruction techniques based on monocular systems, which could improve and simplify the tracking of objects. In this thesis we have implemented a sparse proof of concept, but the densification of the estimated point cloud can be very useful for the possible correction of the models in case of geometric deviations or erroneous behavior laws.

  Another improvement could be the use of massive data in multiparametric models, creating larger databases with more degrees of freedom to extract a greater amount of relevant information from the data. As an example, more load states could be computed where the direction, module and point of application of the forces would be considered as independent parameters. The analysis of deformations and stresses with larger databases could make it possible to search for new, reduced, and more optimal spaces on which to project the information. This would allow the data to be extrapolated to non-simulated load cases, but implies greater prior work, more typical of development phases than research.

  Although the data coming from images have already been filtered by robust adjustment, probabilistic data assimilation techniques could be used to take into account possible deviations in the models. We are thinking of models or laws that do not only take into account specific values of the parameters, but also their distribution by estimating the probability density. The analysis of the inverse problem varies slightly in these cases, but it can improve the adjustment in some conditions.

It is also planned to use future corrections on the models themselves. Sometimes we use material laws to define how objects behave, but it is possible that these laws does not faithfully approximate the real behavior; either by mistake in the definition or because the system has evolved and changed. This type of corrections are those that must be automatically extracted from the data in order to correct any deviations that may arise. In the mechanics of deformable solids, these types of effects may be more evident in viscous materials or plastic phenomena, where the solid history is highly relevant.

- Chapter 5: in the contact problem with virtual objects we have defined and simulated a series of points on which the user can press in real time. The development of an interpolator to estimate the deformations and stresses when the load is applied in any other point of the mesh is still pending. Although it may seem like a simple task, standard interpolations produce incorrect results in these cases because the local phenomena that appear when contact occurs are not easily propagated. We cannot interpolate linearly in the Euclidean space, but we must do it in the reduced space and project the results later on the original space. Another approach to face this problem is the use of neural networks based on graphs, where all the geometric complexity could be reduced by convolutions [Kipf and Welling, 2016].

- Chapter 6: the creation of a larger database would undoubtedly result in an improvement in the estimation of vehicle aerodynamics. The use of more parameters such as vehicle speed or the wind direction could also be proposed. A finer mesh to better capture variations could also be a notable improvement. But we consider that the application developed in this thesis to visualize aerodynamics in real vehicles is in itself a step forward for the democratization of simulations in engineering, which opens the door to great number of ideas that can be developed, such as the inclusion of turbulent phenomena.

Also as future work, although it is a slightly different idea to the work presented in any of the chapters, is the possibility of modifying the real world. We talk about being able to deform real objects from the virtual environment, so that although in reality there are no such deformations, the user perceives them through his visual immersion. Very few applications consider actually this possibility of manipulating reality. A known exception seems to be [Davis et al., 2015], where each video frame is considered as a two-dimensional elastic continuum that can be deformed according to the presence of a virtual object. Reality appears thus modified by the presence of virtual objects.

7.4 Publications

7.4.1 Journal Publications

1. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, vol. 112, no. 12, pp. 1715-1732.
2. González, D., Badías, A., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Model order reduction for real-time data assimilation through Extended Kalman Filters. *Computer Methods in Applied Mechanics and Engineering*, vol. 326, no. Supplement C, pp. 679 – 693.
3. Badías, A., Alfaro, I., González, D., Chinesta, F. and Cueto, E. (2018). Reduced order modeling for physically-based augmented reality. *Computer Methods in Applied Mechanics and Engineering*, vol. 341, pp. 53 – 70.
4. Badías, A., Curtit, S., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2019). An augmented reality platform for interactive aerodynamic design and analysis. *International Journal for Numerical Methods in Engineering*, vol. 120, no. 1, pp. 125-138.
5. Moya, B., Badías, A., Alfaro, I., Chinesta, F. and Cueto, E. Under revision. Digital twins that learn and correct themselves. *International Journal for Numerical Methods in Engineering*.
6. Badías, A., Alfaro, I., González, D., Chinesta, F. and Cueto, E. Under revision. Real-time interaction of virtual and physical objects in Mixed Reality applications. *Computational Mechanics*.
7. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. To be submitted. Reduced Order, Parametric Templates for Augmented Reality.

7.4.2 Oral communications in Conferences

8. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local PGD. *III workshop CSMA-SEMNI Numerical techniques for nowadays highly computationally demanding challenges: meshless, MOR and beyond.*, Jaca, Spain, February 2017.
9. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition. *20th International ESAFORM Conference on Material Forming*, Dublin, Ireland, 26-28 Abril, 2017.
10. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition (ℓ-PGD). *Congreso de Métodos Numéricos en Ingeniería*, Valencia, Spain, 3-5 July, 2017.

11. Badías, A., González, D., Alfaro, I. and Cueto, E. A reduced-order approach to augmented reality in biomedical applications. *VI ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing*, Oporto, Portugal, 18-20 October, 2017.
12. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition (ℓ-PGD). *4th International Workshop on Reduced Basis; POD and PGD Model Reduction Techniques*, Seville, Spain, 8-10 November, 2017.
13. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Mixing Model Order Reduction Methods with Augmented Reality Techniques: A New Paradigm to (Re)Discover. *IUTAM Symposium on Model Order Reduction of Coupled Systems*, Stuttgart, Germany, 22-25 May, 2018.
14. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Model Order Reduction for Physically-Based Augmented Reality. *6th European Conference on Computational Mechanics (ECCM 6), 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, Glasgow, UK, 11-15 June, 2018.
15. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Deformable Augmented Reality via Model Order Reduction Methods. *Virtual Physiological Human Conference*, Zaragoza, Spain, 5-7 September, 2018.
16. Badías, A., González, D., Alfaro, I. and Cueto, E. Reduced order modelling applied to augmented reality. *Jornada de Jóvenes Investigadores del I3A*, Zaragoza, Spain, 8 June, 2018.

7.4.3 Poster Communications

17. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Reduced-order modelling for augmented reality. *The 1st Workshop on data-based engineering, science and technology*, Nantes, France. March 27-29, 2017.
18. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Improving the Realism of Mixed Reality through Physical Simulation. *SIGGRAPH 2018*, 12-16 August, Vancouver, 2018.
19. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Towards Simulated Reality. *Third Workshop on Computer Vision for AR/VR, CVPR*, Long Beach, CA, United States, 16-20 June, 2019.

7.4.4 Complementary Dissemination

Several video sequences have been created for each example, which can be visualized in the following links:

- Aluminum cantilever beam with fiducial markers.
  https://www.youtube.com/watch?v=BK0sHfvjITo
- Foam beam with no markers.
  https://www.youtube.com/watch?v=byhXy8aJ1J8
- Simulation of the behaviour of the rubber piece.
  https://youtu.be/0whA92vFk1g
- Contact between real and virtual objects.
  https://youtu.be/jtMe47mg82k
- Virtual representation of aerodynamics on real vehicles.
  https://www.youtube.com/watch?v=Azm4E3Y7jS4

7.5 Research Stays

Throughout the doctoral period two research stays were made in foreign centers. A quick summary of each of these stays is given below.

7.5.1 Research stay at École Centrale de Nantes

- SUPERVISOR: Prof. Francisco Chinesta.
- DATES: March 23 to June 29, 2017 (99 days).
- SUMMARY: The work was based on the adaptation of new learning techniques on the basis of classical mechanical engineering. Two clearly differentiated works were carried out. The first of them consisted on the estimation of the physical condition of the driver of a car, a work with direct application to the industry. The second work consisted in the use of reduced order models for the calculation of relative displacements between objects subjected to cyclic loads, such as the union between the stem of a hip prosthesis and the femur. Cyclic calculations are always a challenge for engineers as they require a large amount of computing time. However, using model order reduction techniques this computational effort can be greatly reduced. More information is provided below:

  1. **Extraction of behavior patterns from car users data** This work has been carried out in collaboration with Professor H.Y. Choi of Hongik University, Seoul. Professor Choi carried out several experiments on people to characterize the state of the driver of a car. To do this, he applied movement to the car seat with a sinusoidal chirp signal (with increasing frequency) and measured the movements of the head with accelerometers and gyroscopes. The aim was to separate between drivers and

co-drivers, tense and relaxed users, and rigid and soft seats. The great variability of the data and the few signals taken played against us and we were not able to differentiate between all classes. However, by extracting second order system models (in Laplace domain) from each patient and making use of supervised learning techniques such as Support Vector Machines (SVM) we were able to extract consistent results to predict the state (tense or relaxed) of new users that were not taken into account in the learning phase with a success rate of 79.2%. The work was written as a document and sent to Professor Choi.

2. **Reduced Models in the Cyclic Analysis of Total Hip Replacement** The second work consisted in the application of reduced models to cyclical problems. It is of great scientific interest since cyclic loads are very expensive in standard solvers. In many cases, an equivalent static load is applied instead of the cyclic dynamic load to emulate the evolution of the mechanical system at the end of the load cycle, but these are statistical approximations. The computation of the real system for the whole time domain is usually considered too costly and is rarely carried out, although it allows us to obtain a much more reliable result than the statistical analysis of equivalent loads. For this reason, we developed a model order reduction method to project our solution on a reduced basis and considerably reduce both the complexity of the problem and its computation time, maintaining the advantages of solving for the entire time domain.

We applied this method to the study of relative displacements between the stem of a non-cemented hip prosthesis and the bone of the femur on which the prosthesis is inserted. The relative displacements between the two objects make it possible to evaluate whether the surgery has been successful, since the high values of these displacements mean that osseointegration between the bone and the prosthesis cannot occur and that the known effect of rejection of the prosthesis by the human body appears. To this end, we applied the system of loads to which objects are subjected when a person performs a daily task: the walking cycle. The success of the surgical intervention is evaluated according to the relative displacements between two objects subjected to cyclic loads. So, advances in this area can serve both to make predictions about the useful life of a prosthesis and to be able to select the best prosthesis to suit a given patient. This work has been carried out in collaboration with the ESI Group company.

7.5.2 Research stay at University of Washington

- SUPERVISOR: Prof. Ashis G. Banerjee.
- DATES: March 16 to June 30, 2019 (107 days).
- SUMMARY: The work done in Seattle served to propose two complementary metrics to estimate the complexity of a neural network architecture, with focus in convo-

lutional nets, but that can be extrapolated to any architecture. These metrics are focused in the structure of the net, being independent of the data, to detect the network global *expressivity*. Our first proposed metric was called *Intrinsic Power* and serves to meausure the *data power changes* or *compression* along the net. The second metric was *Layer Complexity* and tries to approximate the architecture complexity of the net. They were based in the *Layer Algebra* concept that was also developed in my stay, where any layer can be approximated as a local transfer function allowing a very simple computation of the metrics. We provided a comparison of the state-of-the art architectures complexity using our metrics.

Appendix A

Conclusiones

Este capítulo recoge las conclusiones generales en castellano, el trabajo futuro propuesto, las contribuciones concretas y las publicaciones fruto del trabajo realizado.

A.1 Conclusiones de la tesis

En esta tesis se sientan las bases de la realidad simulada, entendida como una evolución de los sistemas de realidad aumentada donde se añade información del comportamiento físico de los objetos. Dicha información proviene del resultado de simulaciones ingenieriles, por lo que dichas interacciones físicas son descritas con un alto grado de evidencia.

En el trabajo realizado se han cubierto todas las partes de un sistema de realidad simulada: hemos resuelto las ecuaciones de los modelos mecánicos de cada problema para así obtener la información a visualizar (simulaciones ingenieriles), hemos trabajado en el medio de adquisición (captura de datos mediante imágenes) y en la visualización completa de los resultados (computación gráfica). Se han desarrollado trabajos en cada una de estas áreas para conseguir una fusión completa, pero hay que recordar que esta tesis doctoral se engloba en el ámbito de la ingeniería mecánica, y es en este campo donde se ha producido la mayor cantidad de aportaciones.

A lo largo de todo el documento hemos pretendido mostrar la complejidad que existe en el cómputo de las simulaciones basadas en la física real. Hemos aplicado ecuaciones diferenciales para modelar el comportamiento de sólidos deformables y para estimar la aerodinámica de vehículos. Como ya hemos visto, estos complejos cálculos no pueden ser resueltos en tiempo real, pero gracias a la ciencia de datos empleando técnicas de reducción de modelos, podemos estimar la variabilidad de dichos datos para comprimir la solución y trabajar de manera más eficiente.

En este trabajo nos hemos centrado en las técnicas de *machine learning* basadas en la proyección controlada de los datos sobre espacios de dimensionalidad menor. Hemos empleado tanto proyecciones lineales como no lineales basadas en expansiones donde se pretende hallar un nuevo espacio para representar los datos de manera óptima. Incluso pretendemos ser capaces de movernos a lo largo de dicho espacio, que puede no tener significado físico, pero mejora la eficiencia puesto que representa la verdadera variabilidad de la solución. Las técnicas utilizadas en esta tesis están basadas principalmente en la herramienta estadística PCA (*Principal Component Analysis*) y en sus versiones no lineales: k-PCA (*Kernel-Principal Component Analysis*) y LLE (*Locally Linear Embedding*). Otra de las técnicas que tiene gran peso en este trabajo es la versión *a priori* de PCA, conocida como PGD (*Proper Generalized Decomposition*), aunque con diferencias notables como la no imposición de ortogonalidad en los vectores de la nueva base.

En ocasiones, las soluciones multiparamétricas pueden tener variabilidades muy concentradas en áreas locales, lo que supone que aproximaciones como las realizadas con PGD presenten problemas en la estimación de la solución requiriendo un número elevado de términos. Para ello, hemos decidido aplicar aproximaciones locales reduciendo los dominios donde se aplica PGD, de manera que se emplean dominios de pequeño tamaño para áreas con soluciones muy variables y complejas, y dominios más grandes para aproximar aquellas partes donde la solución presenta una variabilidad más suave. Hemos propuesto tres métodos para definir el tamaño de los dominios locales. El primero de ellos (Cℓ-PGD) consiste en realizar particiones de tamaño constante, el segundo método (MNOℓ-PGD) selecciona el tamaño local para obtener un número de modos constante y el tercer método ($k\ell$-PGD) emplea la información proveniente de la transformación no lineal que aporta k-PCA para definir el tamaño de los subdominios. Los resultados obtenidos demuestran que para soluciones con variabilidades abruptas localizadas de hasta 2 dimensiones, el método MNOℓ-PGD (optimizando el número de modos) es el más deseado, puesto que mantiene un número de modos constante a lo largo de todos los dominios, lo que se traduce finalmente en un menor coste de almacenamiento. Sin embargo, para problemas de más de 2 dimensiones, donde también aparecen zonas de variabilidad abrupta combinadas con otras áreas más suaves, hemos visto que el método $k\ell$-PGD que estima la complejidad a partir de la transformación de k-PCA es el método que mejor resultados aporta. Según los experimentos, la técnica k-PCA es capaz de capturar la variabilidad multidimensional tras la aplicación de un *mapping* no lineal, y proyectarla sobre el número de dimensiones elegido, donde herramientas como la distancia de *Hausdorff* son muy útiles para medir diferencias entre los datos. Es importante destacar que los parámetros del *kernel* requieren de un ajuste para adaptar la transformación aplicada al rango de valores óptimo. En nuestros experimentos hemos empleado un *kernel* gaussiano.

Normalmente se desea reducir la dimensionalidad del problema multiparamétrico y estimar el *manifold* reducido en el que verdaderamente vive la solución. Esto implicia que, modificando las coordenadas reducidas $\boldsymbol{\xi}$ (con dimensiones probablemente no físicas), se producen cambios en la solución original, con dependencia sobre el vector de parámetros $\boldsymbol{\mu}$. Dicho de otro modo, las dimensiones reducidas son función de las dimensiones originales (en caso de una proyección lineal, los parámetros reducidos son combinaciones lineales de los parámetros originales). Dicho esto, en ocasiones la variabilidad del sistema es tan compleja que una proyección lineal no aporta grandes ventajas. Esto abre la puerta hacia las proyecciones no lineales, pero como cabe esperar, existen infinidad de posibilidades si deseamos aplicar un *mapping* no lineal sobre nuestra solución. Algunos trabajos [Brunton et al., 2016] pretenden seleccionar las bases de proyección a partir de combinaciones no lineales de las soluciones originales, construyendo una librería de funciones que se ponderan para aproximar la solución original mediante la minimización de un funcional definido. Otra opción es, en vez de reducir la dimensionalidad con proyecciones no lineales complejas, incrementarla con el objetivo de aplicar *mappings* lineales sobre problemas multidimensionales que se espera que sean más fácilmente separables. Este es el principio sobre el que se sustenta k-PCA u otros métodos basados en *kernel*, como SVM (*Support Vector Machines*), para mejorar la separabilidad en la clasificación de patrones.

Continuando con técnicas no lineales de proyección, LLE ha sido empleada en esta tesis para estimar el comportamiento aerodinámico de vehículos en función de un conjunto de simulaciones previamente calculado. Primero, se ha empleado para *clusterizar* la geometría de un vehículo en función de geometrías similares, obteniendo una matriz de pesos que puntúan la similaridad de cada geometría con sus vecinos. Tras esto, dicho conjunto de pesos se ha utilizado para estimar el comportamiento aerodinámico de un vehículo circulando a velocidad constante. El resultado nos indica que es posible estimar dichos resultados con un alto grado de certeza, aunque futuras aproximaciones se pueden llevar a cabo realizando un análisis más exhaustivo con mallas geométricas más finas.

Desde el punto de vista de captura de información a partir de imágenes, se han empleado dos tipos de dispositivos a lo largo de la tesis: cámaras monoculares y sistemas estéreo. Como ya se ha visto en el Capítulo 2, las cámaras monoculares añaden mayor complejidad al método ya que se necesita triangular la posición de los objetos desde diferentes puntos de la escena. Una primera aproximación al problema se ha llevado a cabo empleando marcadores y homografías (ejemplos de las vigas en voladizo y biapoyada en el Capítulo 4), donde las deformaciones se efectúan en un plano del espacio, lo que nos ha permitido estimar de manera sencilla la posición de la carga conociendo el tamaño y posición de las vigas. Posteriormente, para estimar la posición de la cámara en todo instante de tiempo a la vez que se escanea una escena menos controlada hemos empleado ORB-SLAM2 [Mur-Artal and Tardós, 2017], una herramienta de gran valor que ha sido empleada en esta tesis para simplificar el proceso de medición a partir de imágenes. Para problemas estáticos, como la visualización de la aerodinámica sobre el vehículo, se ha empleado directamente la implementación básica del ORB-SLAM2. Ha sido necesario desarrollar técnicas de registro y alineamiento para cada problema, donde se han tenido que programar soluciones *ad-hoc*, pero el núcleo de ORB-SLAM2 se ha mantenido intacto. Sin embargo, para los problemas deformables ha sido necesario modificar el código original para *trackear* los objetos deformables en función de las deformaciones provenientes de los modelos paramétricos. Dicha implementación sí puede considerarse como una contribución propia de esta tesis, donde se ha desarrollado toda la formulación variacional del problema teniendo en cuenta que las derivadas de cada parámetro deben ser proyectadas sobre el plano de imagen para obtener un funcional que minimice el error de reproyección en la imagen. En el caso del sistema estéreo, las librerías del propio dispositivo se han encargado del problema de SLAM, que junto con el sensor inercial integrado producen un sistema robusto que permite su utilización directa.

Desde el punto de vista de la informática gráfica, se han empleado técnicas del estado del arte para visualizar los resultados sin haber desarrollado ningún algoritmo excesivamente novedoso. Aunque ha sido necesario un proceso de aprendizaje para manejar correctamente el proceso de visualización, lo que nos ha proporcionado conocimientos que no disponíamos previamente y nos ha permitido comprender el manejo de algunos conceptos básicos como los *buffers* o *shaders*. Estos conceptos nos han permitido implementar un sistema para visualizar las oclusiones entre objetos reales y virtuales, de modo que el comportamiento ante el usuario es mucho más realista.

La fusión entre todas las técnicas descritas anteriormente nos ha permitido estimar las deformaciones de objetos deformables y plotear los estados tensionales que se generan. Se han implementado varios ejemplos como las vigas empotrada y biapoyada, el guardapolvos (en sus dos vertientes), el contacto directo con el conejo y el dragón y la visualización de la aerodinámica sobre vehículos.

Desde el punto de vista de la adquisición de datos, hemos asumido que los errores de medida son despreciables y ya han sido filtrados por el algoritmo de SLAM empleando técnicas RANSAC [Fischler and Bolles, 1981]. Esta aproximación puramente determinista ha dado buenos resultados en todos los ejemplos mostrados ya que los modelos han sido creados específicamente para cada ejemplo. Sin embargo, pueden aparecer desviaciones de dos tipos: aquellas debidas a errores de medida y desviaciones debidas a errores en el modelo. Aunque en esta tesis no aparecen este tipo de correcciones, sí hemos trabajado en este tipo de implementaciones. Para corregir los errores de medida se pueden aplicar métodos probabilistas que tienen en cuenta la distribución de los datos y el ruido. Un trabajo aplicado a sistemas mecánicos ya ha sido publicado [González et al., 2017], aunque no se ha llegado a implementar para trabajar con una secuencia de video. Por otro lado, en caso de que sean necesarias correcciones sobre el modelo de comportamiento de los objetos deformables, se puede añadir la implementación del *Hybrid Twin* [Chinesta et al., 2018] para corregir las estimaciones. Una primera aproximación aplicada sobre imagen también ha sido implementada en [Moya et al., 2019], aunque se prevén desarrollos más profundos sobre dicho tema.

A.2 Contribuciones

La contribución fundamental de esta tesis ha sido el desarrollo, por primera vez, de una metodología de realidad mixta que permite simular la interacción con sólidos de cualquier tipo de comportamiento, con especial interés en los hiperelásticos. La metodología desarrollada opera en tiempo real (30-60 fps) lo que facilita una interacción muy fluida con el usuario.

Además, de manera igualmente novedosa, se ha desarrollado un método de simulación de fenómenos de contacto que permite visualizar con un alto grado de realismo la interacción entre sólidos deformables reales y virtuales, pudiendo observar los valores de las deformaciones o tensiones sufridas por los objetos en el mismo instante en que se deforman. Esta misma metodología se aplica también a la interacción con fluidos.

A todo esto, hay que sumar otras contribuciones de carácter más técnico que han permitido alcanzar las anteriormente mencionadas:

- Creación de un *framework* de realidad mixta que trabaja con modelos reducidos para aplicaciones en tiempo real, empleando tanto métodos *a priori* como *a posteriori*, dependiendo de la adecuación de cada método al tipo de problema en cuestión.

- Implementación de un método no lineal para mejorar la separabilidad de la solución y reducir el coste de memoria para almacenar los datos. Dicho método ha consistido en una aproximación local del método PGD, imponiendo el cumplimiento de las condiciones de contorno entre subdominios.

- Desarrollo de 3 técnicas diferentes para definir el tamaño de cada subdominio local y aplicar posteriormente el método PGD, destacando la idoneidad de cada técnica en función del problema.

- Estimación de la deformación y lugar de aplicación de la carga para objetos deformables empleando homografías y sistemas monoculares mediante la resolución del problema inverso sobre soluciones proyectadas en espacios de dimensionalidad reducida.

- Implementación *ad-hoc* de un algoritmo de registro entre modelos CAD y nubes de puntos con alta densidad de ruido como la que se obtiene a partir de un método SLAM basado en características.

- Estimación de la deformación y lugar de aplicación de la carga para objetos deformables empleando ORB-SLAM y sistemas monoculares mediante la resolución del problema inverso sobre soluciones proyectadas en espacios de dimensionalidad reducida a partir de su interacción con objetos rígidos que se trasladan en el entorno.

- Implementación de una solución deformable del algoritmo ORB-SLAM2 mediante el *tracking* particular de cada punto característico a partir de la solución proyectada en un espacio de dimensionalidad reducida.

- Estimación del contacto con oclusiones entre objetos reales y virtuales en un entorno de realidad mixta empleando un sistema estéreo, donde las deformaciones están basadas en la verdadera física, de modo que podemos visualizar las tensiones de los objetos deformados con un alto grado de fidelidad.

- Implementación de un visualizador del comportamiento aerodinámico de un vehículo basado en ORB-SLAM y un sistema monocular permitiendo su aplicación sobre un vehículo real a partir de datos precomputados.
- Aproximación de la aerodinámica mediante técnicas de proyección no lineales a partir de una base de datos de varios vehículos previamente calculada, puesto que no es posible realizar interpolaciones en el espacio de alta dimensión.

A.3 Trabajo Futuro

En cuanto al trabajo futuro, son múltiples las ideas que han ido surgiendo tras el desarrollo del trabajo. Para acotarlas y ordenarlas, vamos a englobar cada una de ellas según el orden de los capítulos del presente documento (sin tener en cuenta los capítulos 1, 2 y 7).

- Capítulo 3: hemos propuesto tres métodos para definir los límites de los subdominios locales, pero por supuesto, se puede profundizar en este aspecto desarrollando nuevos métodos para fijar dichas fronteras, algo que deja de ser sencillo cuando tratamos con problemas de alta dimensión. Los datos tienden a dispersarse conforme el número de dimensiones aumenta, por lo que las herramientas desarrolladas pueden requerir un tratamiento diferente cuando tratamos con muchas dimensiones.

  Por otro lado, como ya hemos visto con la aplicación de la herramienta k-PCA, algunos problemas complejos pueden mejorar la separabilidad de su solución si se proyectan sobre un espacio dimensional mayor, donde los datos puede hasta ser linealmente separables [Hofmann, 2006]. Ese es el principio de los métodos basados en *kernel*, y quizá podría aplicarse directamente sobre el método PGD para obtener su homónimo *k-PGD* (kernel Principal Component Analysis), donde se resuelva la ecuación diferencial en un espacio de dimensionalidad mayor pero a cambio se consiga mejorar la separabilidad de la solución.

- Capítulo 4: dejamos para un futuro la utilización de técnicas de reconstrucción densas basadas en sistemas monoculares, lo que podría mejorar y simplificar el *tracking* de los objetos. En esta tesis hemos implementado una prueba de concepto, pero la densificación de la nube de puntos estimada puede ser muy útil para la posible corrección de los modelos en caso de que existan desviaciones geométricas o en la ley de comportamiento.

  Otra vía de mejora podría ser la explotación a mayor escala de los modelos multiparamétricos, creando bases de datos mayores y con más grados de libertad para poder sacar una cantidad de información mayor a partir de los datos. Como ejemplo, podrían crearse más estados de carga donde la dirección, el módulo y punto de aplicación de las fuerzas fuesen considerados como parámetros independientes. El análisis de las deformaciones y tensiones con una base de datos de gran tamaño podría permitir buscar nuevas bases reducidas más óptimas sobre las que proyectar la información. Esto permitiría extrapolar los datos a nuevos casos de carga no simulados, pero implica un trabajo previo mayor, más enfocado hacia el desarrollo que hacia la investigación.

Aunque los datos que provienen de las imágenes ya han sido filtrados mediante ajuste robusto, se podrían emplear técnicas de asimilación de datos probabilistas que tengan en cuenta las posibles desviaciones en los modelos. Estamos pensando en modelos o leyes que no tengan únicamente en cuenta valores concretos de los parámetros, sino la distribución de los mismos mediante la estimación de la densidad de probabilidad de los mismos. El análisis del problema inverso varía ligeramente en estos casos, pero podría mejorar el ajuste.

También se prevé emplear correcciones sobre los propios modelos que describen la física de los objetos. En ocasiones, empleamos una ley de comportamiento para definir cómo se comportan los objetos, pero es posible que nuestra ley no aproxime de manera fiel el comportamiento real; ya sea por error en la definición de la misma o porque el sistema ha evolucionado y cambiado de ley de comportamiento. Este tipo de correcciones son las que se deben extraer automáticamente de los datos para poder corregir las desviaciones que puedan surgir. En cuanto a la mecánica de sólido deformable, este tipo de efectos pueden ser más evidentes en materiales viscosos o fenómenos plásticos, donde la historia por la que ha pasado el sólido es altamente importante.

- Capítulo 5: en el contacto con objetos virtuales hemos definido y simulado una serie de puntos de contacto sobre los que el usuario puede presionar en tiempo real. Queda pendiente el desarrollo de un interpolador para estimar las deformaciones y tensiones cuando la carga se aplique en cualquier otro punto de la malla. Aunque pueda parecer una tarea sencilla, las interpolaciones estándar no sirven en estos casos puesto que los fenómenos locales que aparecen cuando se produce el contacto no son fácilmente propagables. No podemos interpolar linealmente en el espacio euclídeo, sino que debemos hacerlo en el espacio reducido y proyectar los resultados posteriormente sobre la base original. Otra aproximación para afrontar este problema es la utilización de redes neuronales basadas en grafos, donde toda la complejidad geométrica se podría ir reduciendo mediante convoluciones [Kipf and Welling, 2016].

- Capítulo 6: la creación de una base de datos mayor, sin duda, resultaría en una mejora en la estimación de la aerodinámica de vehículos. También se podría proponer el uso de más parámetros como la velocidad del vehículo o la introducción de viento racheado en la conducción. Un mallado más fino que permita capturar mejor las variaciones podría ser también una mejora notable. Pero consideramos que la aplicación desarrollada en esta tesis para visualizar la aerodinámica en vehículos reales ya es en sí un paso hacia delante para la democratización de las simulaciones en ingeniería, por lo que se abre la puerta hacia multitud de ideas que pueden desarrollarse y mejorarse.

También como trabajo futuro, aunque se trata de una idea ligeramente diferente al trabajo realizado en cualquiera de los capítulos, nos planteamos la posibilidad de modificar el entorno real. Hablamos de ser capaces de deformar los objetos reales desde el entorno virtual, de modo que aunque en la realidad no se produzcan dichas deformaciones, el usuario pueda percibirlas desde su ambiente inmersivo. Parece ser que, por el momento, pocas aplicaciones consideran la posibilidad de manipular la realidad. Una excepción es [Davis et al., 2015], donde cada fotograma de una secuencia de video es considerado como un continuo elástico bidimensional que puede ser deformado ante la presencia de un objeto virtual. Por tanto, podemos decir que la realidad percibida por el usuario es virtualmente modificada.

A.4 Publicaciones

A.4.1 Revistas

1. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, vol. 112, no. 12, pp. 1715-1732.

2. González, D., Badías, A., Alfaro, I., Chinesta, F. and Cueto, E. (2017). Model order reduction for real-time data assimilation through Extended Kalman Filters. *Computer Methods in Applied Mechanics and Engineering*, vol. 326, no. Supplement C, pp. 679 – 693.

3. Badías, A., Alfaro, I., González, D., Chinesta, F. and Cueto, E. (2018). Reduced order modeling for physically-based augmented reality. *Computer Methods in Applied Mechanics and Engineering*, vol. 341, pp. 53 – 70.

4. Badías, A., Curtit, S., González, D., Alfaro, I., Chinesta, F. and Cueto, E. (2019). An augmented reality platform for interactive aerodynamic design and analysis. *International Journal for Numerical Methods in Engineering*, vol. 120, no. 1, pp. 125-138.

5. Moya, B., Badías, A., Alfaro, I., Chinesta, F. and Cueto, E. En revisión. Digital twins that learn and correct themselves. *International Journal for Numerical Methods in Engineering*.

6. Badías, A., Alfaro, I., González, D., Chinesta, F. and Cueto, E. En revisión. Real-time interaction of virtual and physical objects in Mixed Reality applications. *Computational Mechanics*.

7. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Pendiente de envío. Reduced Order, Parametric Templates for Augmented Reality.

A.4.2 Ponencias Orales en Congresos

8. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local PGD. *III workshop CSMA-SEMNI Numerical techniques for nowadays highly computationally demanding challenges: meshless, MOR and beyond.*, Jaca, España, febrero 2017.

9. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition. *20th International ESAFORM Conference on Material Forming*, Dublín, Irlanda, 26-28 de abril, 2017.

10. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition (ℓ-PGD). *Congreso de Métodos Numéricos en Ingeniería*, Valencia, España, 3-5 de julio, 2017.

11. Badías, A., González, D., Alfaro, I. and Cueto, E. A reduced-order approach to augmented reality in biomedical applications. *VI ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing*, Oporto, Portugal, 18-20 de octubre, 2017.

12. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Local Proper Generalized Decomposition (ℓ-PGD). *4th International Workshop on Reduced Basis; POD and PGD Model Reduction Techniques*, Sevilla, España, 8-10 de noviembre, 2017.

13. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Mixing Model Order Reduction Methods with Augmented Reality Techniques: A New Paradigm to (Re)Discover. *IUTAM Symposium on Model Order Reduction of Coupled Systems*, Stuttgart, Alemania, 22-25 de mayo, 2018.

14. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Model Order Reduction for Physically-Based Augmented Reality. *6th European Conference on Computational Mechanics (ECCM 6), 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, Glasgow, Reino Unido, 11-15 de junio, 2018.

15. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Deformable Augmented Reality via Model Order Reduction Methods. *Virtual Physiological Human Conference*, Zaragoza, España, 5-7 de septiembre, 2018.

16. Badías, A., González, D., Alfaro, I. and Cueto, E. Reduced order modelling applied to augmented reality. *Jornada de Jóvenes Investigadores del I3A*, Zaragoza, España, 8 de junio, 2018.

A.4.3 Comunicaciones Póster

17. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Reduced-order modelling for augmented reality. *The 1st Workshop on data-based engineering, science and technology*, Nantes, Francia. 27-29 de marzo, 2017.
18. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Improving the Realism of Mixed Reality through Physical Simulation. *SIGGRAPH 2018*, Vancouver, Canadá, 12-16 de agosto, 2018.
19. Badías, A., González, D., Alfaro, I., Chinesta, F. and Cueto, E. Towards Simulated Reality. *Third Workshop on Computer Vision for AR/VR, CVPR*, Long Beach, Estados Unidos, 16-20 de junio, 2019. Aceptado.

A.4.4 Comunicaciones Complementarias

Se han creado varias secuencias de video demostrativas para cada uno de los ejemplos, que pueden ser consultados en los siguientes enlaces:

- Viga empotrada de aluminio con marcadores.

  `https://www.youtube.com/watch?v=BK0sHfvjITo`
- Viga biapoyada de espuma sin marcadores.

  `https://www.youtube.com/watch?v=byhXy8aJ1J8`
- Simulación del comportamiento del fuelle guardapolvos.

  `https://youtu.be/0whA92vFk1g`
- Contacto entre objetos reales y virtuales.

  `https://youtu.be/jtMe47mg82k`
- Representación virtual de la aerodinámica sobre vehículos reales.

  `https://www.youtube.com/watch?v=Azm4E3Y7jS4`

A.5 Estancias de Investigación

Durante el transcurso de la tesis doctoral se han realizado dos estancias en centros de investigación extranjeros. Un rápido resumen de cada una de las estancias aparece a continuación.

A.5.1 Estancia en École Centrale de Nantes

- SUPERVISOR: Prof. Francisco Chinesta.
- FECHAS: 23 de marzo a 29 de junio de 2017 (99 días).
- RESUMEN: El trabajo realizado estuvo basado en la adaptación de nuevas técnicas de aprendizaje y reducción de modelos sobre las bases de la ingeniería mecánica clásica. Se realizaron dos trabajos claramente diferenciados a lo largo de los 3 meses. El primero de ellos, consistió en una aplicación para la estimación del estado físico del conductor de un automóvil, un trabajo con aplicación directa dados los avances en investigación del automóvil. El segundo trabajo consistió en el empleo de modelos reducidos para el cálculo de desplazamientos relativos entre objetos sometidos a cargas cíclicas, como puede ser la unión entre el vástago de una prótesis de cadera y el fémur. Los cálculos cíclicos siempre suponen un reto para los ingenieros al requerir gran cantidad de tiempo de cómputo. Sin embargo, empleando técnicas de reducción de modelos, ese trabajo de cómputo puede ser reducido enormemente gracias a las técnicas de proyección que empleamos.

  A continuación, se incluye más información acerca de los trabajos llevados a cabo:

  1. **Extracción de patrones de comportamiento a partir de datos de usuarios de automóviles.**

  Este trabajo se ha realizado en colaboración con el profesor H.Y. Choi de la universidad Hongik University, Seoul. El profesor Choi llevó a cabo varios ensayos sobre personas para caracterizar el estado del conductor de un automóvil. Para ello, estimuló con una señal sinusoidal de tipo chirp (con incremento de frecuencia) y midió mediante acelerómetros y giróscopos los movimientos de la cabeza ante esas señales. El objetivo era ser capaces de separar entre conductores y copilotos, usuarios tensos y relajados, y asientos rígidos y blandos. La gran variabilidad y las pocas señales tomadas hizo que no fuéramos capaces de diferenciar entre todas las clases. Sin embargo, mediante la extracción de modelos de cada paciente y haciendo uso de técnicas de aprendizaje supervisado como Support Vector Machines (SVM) fuimos capaces de extraer resultados coherentes pudiendo realizar predicciones del estado (tenso o relajado) de nuevos usuarios no tenidos en cuenta en la fase de aprendizaje, con un porcentaje de acierto del 79,2%. El trabajo fue redactado en forma de documento y enviado al profesor Choi.

  2. **Modelos reducidos en el análisis cíclico de artroplastia de cadera.**

  El segundo trabajo llevado a cabo consistió en la aplicación de modelos reducidos a un problema cíclico. Tiene gran interés científico puesto que los modelos cíclicos son muy costosos de calcular si se desea resolver el problema íntegro. En muchas ocasiones, se suele tender a resolver un problema de cargas equivalentes que emulan la evolución del sistema mecánico al finalizar el ciclo de cargas, pero se trata de aproximaciones estadísticas. La resolución para todo el dominio de tiempo suele considerarse un problema demasiado costoso y pocas veces se realiza, aunque nos permite obtener un resultado mucho más fiable que el análisis estadístico de cargas equivalentes. Por ello, hemos desarrollado un método de reducción de modelos que nos permite proyectar nuestra solución sobre una base reducida y disminuir considerablemente tanto la complejidad del problema, como el tiempo de cómputo del mismo, manteniendo las ventajas de resolver para todo el dominio temporal.

Hemos aplicado este método al estudio de los desplazamientos relativos entre el vástago de una prótesis de cadera no cementada y el hueso del fémur sobre el que se introduce dicha prótesis. Los desplazamientos relativos entre ambos objetos permiten evaluar si la cirugía ha sido exitosa, puesto que unos valores elevados de estos desplazamientos hacen que no se pueda dar la osteointegración entre hueso-prótesis y que aparezca el conocido efecto de rechazo de la prótesis por parte del cuerpo humano. Para ello, hemos tenido en cuenta el sistema de cargas al que se ven sometidos los objetos cuando una persona realiza una tarea cotidiana: el ciclo de caminar. Algo que puede parecer tan simple como evaluar los desplazamientos relativos entre dos objetos sometidos a cargas cíclicas, puede llegar a ser tan crítico como para dar por válida o no una intervención quirúrgica. Los avances en este área pueden llegar a servir tanto para realizar predicciones acerca de la vida útil de una prótesis, como para ser capaces de seleccionar la mejor prótesis que se adapte a un paciente determinado. Este trabajo se ha realizado en colaboración con la empresa ESI Group.

A.5.2 Estancia en University of Washington

- SUPERVISOR: Prof. Ashis G. Banerjee.
- FECHAS: 16 de marzo a 30 de junio de 2019 (107 días).
- RESUMEN: El trabajo realizado ha servido para proponer dos métricas complementarias para estimar la complejidad de la arquitectura de redes neuronales, con foco en redes convolucionales, pero que pueden ser extrapoladas a cualquier arquitectura. Estas métricas se centran en la estructura de la red, siendo independientes de los datos de entrenamiento, para detectar la *expresividad* global de la red. Nuestra primera métrica propuesta fue definida como *potencia intrínseca* y sirve para medir los cambios en la potencia de los datos, o dicho de otro modo, la compresión a lo largo de la red. La segunda métrica fue definida como *complejidad de las capas* e intenta aproximar la propia complejidad de la arquitectura de la red. Ambas métricas están basadas en el concepto de *álgebra de capas*, que también fue propuesto en la estancia, donde cualquier capa puede ser aproximada como una función de transferencia local permitiendo un cálculo muy simple de las métricas. Para termi-

nar el trabajo, se creó una comparación de la complejidad de las arquitecturas más relevantes en la actualidad empleando las métricas propuestas.

Appendix B

Abbreviations

- AI: Artificial Intelligence
- AR: Augmented Reality
- AV: Augmented Virtuality
- CAD: Computer-Aided Design
- CAE: Computer-Aided Engineering
- DEIM: Discrete Empirical Interpolation Method
- EIM: Empirical Interpolation Method
- FEA: Finite Element Analysis
- FEM: Finite Element Method
- fps: frames per second
- GPU: Graphics Processing Unit
- IA: Intelligence Augmentation
- ICP: Iterative Closest Point
- IoT: Internet Of Things
- k-PCA: kernel Principal Component Analysis
- k-PGD: kernel Principal Generalized Decomposition
- LDA: Linear Discriminant Analysis
- ML: Machine Learning
- MOR: Model Order Reduction
- MR: Mixed Reality

- NRSfM: Non-rigid Structure from Motion
- PCA: Principal Component Analysis
- PCL: Point Cloud Library
- PDE: Partial Derivative Equation
- PGD: Proper Generalized Decomposition
- RANSAC: Random Sample Consensus
- RB: Reduced Basis
- RMSE: Root Mean Square Error
- RR: Real Reality
- SDK: Software Development Kit
- SfM: Structure from Motion
- SLAM: Simultaneous Localization And Mapping
- SVD: Singular Value Decomposition
- UQ: Uncertainty Quantification
- VR: Virtual Reality

Bibliography

Bibliography

[Adamson, 2006] Adamson, P. (2006). Vision, light and color in al-kindī, ptolemy and the ancient commentators. *Arabic sciences and philosophy*, 16(2):207–236.

[Aguado et al., 2019] Aguado, J., Borzacchiello, D., Kollepara, K. S., Chinesta, F., and Huerta, A. (2019). Tensor representation of non-linear models using cross approximations. *Journal of Scientific Computing*, pages 1–26.

[Aguado et al., 2015] Aguado, J., Huerta, A., Chinesta, F., and Cueto, E. (2015). Real-time monitoring of thermal processes by reduced order modelling. *International Journal for Numerical Methods in Engineering*, 102(5):991–1017.

[Agudo and Moreno-Noguer, 2018] Agudo, A. and Moreno-Noguer, F. (2018). A scalable, efficient, and accurate solution to non-rigid structure from motion. *Computer Vision and Image Understanding (CVIU)*.

[Agudo et al., 2016] Agudo, A., Moreno-Noguer, F., Calvo, B., and Montiel, J. M. (2016). Sequential non-rigid structure from motion using physical priors. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):979–994.

[Akhter et al., 2009] Akhter, I., Sheikh, Y., Khan, S., and Kanade, T. (2009). Nonrigid structure from motion in trajectory space. *Advances in neural information processing systems*, pages 41–48.

[Alfaro et al., 2015] Alfaro, I., González, D., Zlotnik, S., Díez, P., Cueto, E., and Chinesta, F. (2015). An error estimator for real-time simulators based on model order reduction. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):30.

[Allier et al., 2015] Allier, P.-E., Chamoin, L., and Ladevèze, P. (2015). Proper generalized decomposition computational methods on a benchmark problem: introducing a new strategy based on constitutive relation error minimization. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):1.

[Alvarez et al., 2002] Alvarez, L., Deriche, R., Sanchez, J., and Weickert, J. (2002). Dense disparity map estimation respecting image discontinuities: A pde and scale-space based approach. *Journal of Visual Communication and Image Representation*, 13(1-2):3–21.

[Ammar et al., 2010] Ammar, A., Chinesta, F., Diez, P., and Huerta, A. (2010). An error estimator for separated representations of highly multidimensional models. *Computer Methods in Applied Mechanics and Engineering*, 199(25-28):1872 – 1880.

[Ammar et al., 2014] Ammar, A., Huerta, A., Chinesta, F., Cueto, E., and Leygue, A. (2014). Parametric solutions involving geometry: A step towards efficient shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268(0):178 – 193.

[Ammar et al., 2006] Ammar, A., Mokdad, B., Chinesta, F., and Keunings, R. (2006). A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3):153–176.

[Amsallem et al., 2010] Amsallem, D., Cortial, J., and Farhat., C. (2010). Towards real-time cfd-based aeroelastic computations using a database of reduced-order information. *AIAA Journal*, 48:2029–2037.

[Amsallem and Farhat, 2008] Amsallem, D. and Farhat, C. (2008). An Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity. *AIAA Journal*, 46:1803–1813.

[Amsallem et al., 2012] Amsallem, D., Zahr, M. J., and Farhat, C. (2012). Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10):891–916.

[Astrid et al., 2008] Astrid, P., Weiland, S., Willcox, K., and Backx, T. (2008). Missing point estimation in models described by proper orthogonal decomposition. *IEEE Transactions on Automatic Control*, 53(10):2237–2251.

[Azuma, 1997] Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385.

[Badías et al., 2018] Badías, A., Alfaro, I., González, D., Chinesta, F., and Cueto, E. (2018). Reduced order modeling for physically-based augmented reality. *Computer Methods in Applied Mechanics and Engineering*, 341:53 – 70.

[Badías et al., 2017] Badías, A., González, D., Alfaro, I., Chinesta, F., and Cueto, E. (2017). Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, 112(12):1715–1732.

[Baldi, 2012] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49.

[Ballarin et al., 2014] Ballarin, F., Manzoni, A., Rozza, G., and Salsa, S. (2014). Shape optimization by free-form deformation: existence results and numerical solution for stokes flows. *Journal of Scientific Computing*, 60(3):537–563.

[Barrault et al., 2004] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672.

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

[Bellman, 2013] Bellman, R. (2013). *Dynamic programming*. Courier Corporation.

[Berkooz et al., 1993] Berkooz, G., Holmes, P., and Lumley, J. L. (1993). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575.

[Bertram et al., 2018] Bertram, A., Othmer, C., and Zimmermann, R. (2018). Towards real-time vehicle aerodynamic design via multi-fidelity data-driven reduced order modeling. *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. *Sensor Fusion IV: Control Paradigms and Data Structures*, 1611:586–607.

[Bonet and Wood, 2008] Bonet, J. and Wood, R. D. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

[Borzacchiello et al., 2017] Borzacchiello, D., Aguado, J. V., and Chinesta, F. (2017). Non-intrusive sparse subspace learning for parametrized problems. *Archives of Computational Methods in Engineering*.

[Boschert and Rosen, 2016] Boschert, S. and Rosen, R. (2016). *Digital twin—the simulation aspect*. Springer.

[Bradski and Kaehler, 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

[Breen et al., 1995] Breen, D. E., Rose, E., and Whitaker, R. T. (1995). Interactive occlusion and collision of real and virtual objects in augmented reality. *European Computer Industry Research Center*.

[Bregler et al., 2000] Bregler, C., Hertzmann, A., and Biermann, H. (2000). Recovering non-rigid 3d shape from image streams. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:690–696.

[Bregler and Omohundro, 1995] Bregler, C. and Omohundro, S. M. (1995). Nonlinear image interpolation using manifold learning. *Advances in neural information processing systems*, pages 973–980.

[Bright et al., 2013] Bright, I., Lin, G., and Kutz, J. N. (2013). Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(12):127102.

[Brunßen et al., 2007] Brunßen, S., Hüeber, S., and Wohlmuth, B. (2007). Contact dynamics with lagrange multipliers. *IUTAM Symposium on Computational Methods in Contact Mechanics*, pages 17–32.

[Brunton et al., 2016] Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

[Buchanan, 2005] Buchanan, B. G. (2005). A (very) brief history of artificial intelligence. *Ai Magazine*, 26(4):53–53.

[Carr and Zukowski, 2018] Carr, C. and Zukowski, Z. (2018). Generating albums with sampleRNN to imitate metal, rock, and punk bands. *arXiv preprint arXiv:1811.06633*.

[Chaconas and Höllerer, 2018] Chaconas, N. and Höllerer, T. (2018). An evaluation of bimanual gestures on the microsoft hololens. *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1–8.

[Chang et al., 2016] Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. (2016). A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, abs/1612.00341.

[Chaudhary and Bathe, 1986] Chaudhary, A. B. and Bathe, K.-J. (1986). A solution method for static and dynamic analysis of three-dimensional contact problems with friction. *Computers & Structures*, 24(6):855–873.

[Chen et al., 2017] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

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

[Chinesta et al., 2016] Chinesta, F., Aguado, J. V., Abisset-Chavanne, E., and Barasinski, A. (2016). Model reduction & manifold learning—based parametric computational electromagnetism: Fundamentals & applications. *IEEE Conference on Electromagnetic Field Computation (CEFC)*, pages 1–1.

[Chinesta et al., 2010] Chinesta, F., Ammar, A., and Cueto, E. (2010). Recent advances in the use of the Proper Generalized Decomposition for solving multidimensional models. *Archives of Computational Methods in Engineering*, 17(4):327–350.

[Chinesta and Cueto, 2014] Chinesta, F. and Cueto, E. (2014). *PGD-Based Modeling of Materials, Structures and Processes*. Springer International Publishing Switzerland.

[Chinesta et al., 2018] Chinesta, F., Cueto, E., Abisset-Chavanne, E., Duval, J. L., and El Khaldi, F. (2018). Virtual, digital and hybrid twins: A new paradigm in data-based engineering and engineered data. *Archives of Computational Methods in Engineering*.

[Chinesta et al., 2013a] Chinesta, F., Keunings, R., and Leygue, A. (2013a). *The proper generalized decomposition for advanced numerical simulations: a primer*. Springer Science & Business Media.

[Chinesta and Ladevèze, 2014] Chinesta, F. and Ladevèze, P. (2014). *Separated Representations and PGD-Based Model Reduction: Fundamentals and Applications*. Springer.

[Chinesta et al., 2011] Chinesta, F., Ladeveze, P., and Cueto, E. (2011). A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404.

[Chinesta et al., 2013b] Chinesta, F., Leygue, A., Bordeu, F., Aguado, J. V., Cueto, E., González, D., Alfaro, I., Ammar, A., and Huerta, A. (2013b). Pgd-based computational vademecum for efficient design, optimization and control. *Archives of Computational Methods in Engineering*, 20(1):31–59.

[Civera et al., 2009] Civera, J., Bueno, D. R., Davison, A. J., and Montiel, J. M. M. (2009). Camera self-calibration for sequential bayesian structure from motion. *IEEE International Conference on Robotics and Automation (ICRA)*.

[Cole and Newman, 2006] Cole, D. M. and Newman, P. M. (2006). Using laser range data for 3d slam in outdoor environments. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1556–1563.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

[Croft and Phillips, 2017] Croft, T. L. D. and Phillips, T. N. (2017). Least-squares proper generalized decompositions for weakly coercive elliptic problems. *SIAM Journal on Scientific Computing*, 39(4):A1366–A1388.

[Cueto et al., 2016] Cueto, E., González, D., and Alfaro, I. (2016). *Proper generalized decompositions: an introduction to computer implementation with Matlab*. Springer.

[Curley, 2010] Curley, R. (2010). *The 100 most influential inventors of all time*. Britannica Educational Pub. in association with Rosen Educational Services.

[Davis et al., 2015] Davis, A., Chen, J. G., and Durand, F. (2015). Image-space modal bases for plausible manipulation of objects in video. *ACM Trans. Graph.*, 34(6):239:1–239:7.

[Davison, 2018] Davison, A. J. (2018). Futuremapping: The computational structure of spatial ai systems. *arXiv preprint arXiv:1803.11288*.

[Davison, 2019] Davison, A. J. (2019). Novel hardware for spatial ai. *Second International Workshop on Event-based Vision and Smart Cameras, CVPR*.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

[De Lathauwer et al., 2000] De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

[Díez et al., 2017] Díez, P., Zlotnik, S., and Huerta, A. (2017). Generalized parametric solutions in stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 326:223–240.

[Dihlmann et al., 2011] Dihlmann, M., Drohmann, M., and Haasdonk, B. (2011). Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. *Proc. of ADMOS*, 2011.

[Dirichlet, 1829] Dirichlet, G. L. (1829). Sur la convergence des séries trigonométriques qui servent àreprésenter une fonction arbitraire entre des limites données». *Journal für die reine und angewandte Mathematik*, 3:157–169.

[Doble, 2012] Doble, R. (2012). *15 Years of Essay-Blogs About Contemporary Art & Digital Photography: In-Depth Articles from 1997-2012*. Lulu Press, Inc.

[Dupré, 2008] Dupré, S. (2008). Inside the camera obscura: Kepler's experiment and theory of optical imagery. *Early Science and Medicine*, 13(3):219–244.

[Edelsbrunner and Mücke, 1994] Edelsbrunner, H. and Mücke, E. P. (1994). Three dimensional alpha shapes. *ACM Transactions on Graphics*, 13:43–72.

[Emerling, 2013] Emerling, J. (2013). *Photography: History and theory*. Routledge.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. *European Conference on Computer Vision*, pages 834–849.

[Facil et al., 2019] Facil, J. M., Ummenhofer, B., Zhou, H., Montesano, L., Brox, T., and Civera, J. (2019). Cam-convs: Camera-aware multi-scale convolutions for single-view depth. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11826–11835.

[Falcó et al., 2013] Falcó, A., Hilario, L., Montés, N., and Mora, M. (2013). Numerical strategies for the galerkin–proper generalized decomposition method. *Mathematical and Computer Modelling*, 57(7):1694–1702.

[Fedorov et al., 2016] Fedorov, R., Frajberg, D., and Fraternali, P. (2016). A framework for outdoor mobile augmented reality and its application to mountain peak detection. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 281–301.

[Fefferman, 2006] Fefferman, C. L. (2006). Existence and smoothness of the navier-stokes equation. *The millennium prize problems*, 57:67.

[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. *Commun. ACM*, 24(6):381–395.

[Fish and Belytschko, 2007] Fish, J. and Belytschko, T. (2007). *A first course in finite elements*. John Wiley & Sons.

[Fua, 1993] Fua, P. (1993). A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine vision and applications*, 6(1):35–49.

[Fukunaga and Olsen, 1971] Fukunaga, K. and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 100(2):176–183.

[Gage, 1914] Gage, H. P. (1914). *Optic projection, principles, installation, and use of the magic lantern, projection microscope, reflecting lantern, moving picture machine*. Comstock.

[Garg et al., 2013] Garg, R., Roussos, A., and Agapito, L. (2013). Dense variational reconstruction of non-rigid surfaces from monocular video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1272–1279.

[Gerbeau and Lombardi, 2012] Gerbeau, J.-F. and Lombardi, D. (2012). Reduced-order modeling based on approximated lax pairs. *arXiv preprint arXiv:1211.4153*.

[Ghnatios et al., 2011] Ghnatios, C., Chinesta, F., Cueto, E., Leygue, A., Poitou, A., Breitkopf, P., and Villon, P. (2011). Methodological approach to efficient modeling and optimization of thermal processes taking place in a die: Application to pultrusion. *Composites Part A: Applied Science and Manufacturing*, 42(9):1169 – 1178.

[Ghnatios et al., 2012] Ghnatios, C., Masson, F., Huerta, A., Leygue, A., Cueto, E., and Chinesta, F. (2012). Proper generalized decomposition based dynamic data-driven control of thermal processes. *Computer Methods in Applied Mechanics and Engineering*, 213-216(0):29 – 41.

[Ghosh, 2019] Ghosh, P. (2019). Aaas: Machine learning 'causing science crisis'. Published online, https://www.bbc.com/news/science-environment-47267081.

[Gonzalez et al., 2010] Gonzalez, D., Ammar, A., Chinesta, F., and Cueto, E. (2010). Recent advances on the use of separated representations. *International Journal for Numerical Methods in Engineering*, 81(5).

[González et al., 2017] González, D., Badías, A., Alfaro, I., Chinesta, F., and Cueto, E. (2017). Model order reduction for real-time data assimilation through Extended Kalman Filters. *Computer Methods in Applied Mechanics and Engineering*, 326(Supplement C):679 – 693.

[González et al., 2018] González, D., Chinesta, F., and Cueto, E. (2018). Thermodynamically consistent data-driven computational mechanics. *Continuum Mechanics and Thermodynamics*, 31(1):239–253.

[González et al., 2015] González, D., Cueto, E., and Chinesta, F. (2015). Computational patient avatars for surgery planning. *Annals of Biomedical Engineering*, 44(1):35–45.

[Gonzalez et al., 2012] Gonzalez, D., Masson, F., Poulhaon, F., Cueto, E., and Chinesta, F. (2012). Proper generalized decomposition based dynamic data driven inverse identification. *Mathematics and Computers in Simulation*, 82:1677–1695.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, pages 2672–2680.

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

[Grady, 1993] Grady, D. (1993). The vision thing: Mainly in the brain. *Discover*, 14(6):56–66.

[Greene et al., 1993] Greene, N., Kass, M., and Miller, G. (1993). Hierarchical z-buffer visibility. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238.

[Guerrero-Viu et al., 2019] Guerrero-Viu, J., Fernandez-Labrador, C., Demonceaux, C., and Guerrero, J. J. (2019). What's in my Room? Object Recognition on Indoor Panoramic Images. *arXiv preprint arXiv:1910.06138*.

[Gutierrez-Gomez et al., 2016] Gutierrez-Gomez, D., Mayol-Cuevas, W., and Guerrero, J. J. (2016). Dense rgb-d visual odometry using inverse depth. *Robotics and Autonomous Systems*, 75:571–583.

[Hammond, 1986] Hammond, M. S. (1986). The camera obscura: a chapter in the prehistory of photography. *The Ohio State University*.

[Haouchine et al., 2015a] Haouchine, N., Cotin, S., Peterlik, I., Dequidt, J., Lopez, M. S., Kerrien, E., and Berger, M. O. (2015a). 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., 2015b] Haouchine, N., Dequidt, J., Berger, M. O., and Cotin, S. (2015b). Monocular 3d reconstruction and augmentation of elastic surfaces with self-occlusion handling. *IEEE Transactions on Visualization and Computer Graphics*, 21(12):1363–1376.

[Haouchine et al., 2012] Haouchine, N., Dequidt, J., Kerrien, E., Berger, M.-O., and Cotin, S. (2012). Physics-based Augmented Reality for 3D Deformable Object. *Eurographics Workshop on Virtual Reality Interaction and Physical Simulation*.

[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.

[Harwood et al., 2018] Harwood, A. R., Wenisch, P., and Revell, A. J. (2018). A real-time modelling and simulation platform for virtual engineering design and analysis. *Proceedings of 6th European Conference on Computational Mechanics (ECCM 6) and 7th European Conference on Computational Fluid Dynamics (ECFD 7)*.

[Hausdorf and Aumann, 1914] Hausdorf, F. and Aumann, J. R. (1914). *Grundzüge der mengenlehre*. Veit.

[Hernández et al., 2014] Hernández, J., Oliver, J., Huespe, A. E., Caicedo, M., and Cante, J. (2014). High-performance model reduction techniques in computational multiscale homogenization. *Computer Methods in Applied Mechanics and Engineering*, 276:149–189.

[Hinton et al., 1995] Hinton, G. E., Revow, M., and Dayan, P. (1995). Recognizing handwritten digits using mixtures of linear models. *Advances in neural information processing systems*, pages 1015–1022.

[Hödlmoser et al., 2012] Hödlmoser, M., Micusik, B., Liu, M.-Y., Pollefeys, M., and Kampel, M. (2012). Classification and pose estimation of vehicles in videos by 3d modeling within discrete-continuous optimization. *Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 198–205.

[Hofmann, 2006] Hofmann, M. (2006). Support vector machines-kernels and the kernel trick. *Notes*, 26.

[Hotelling, 1933] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

[Howe et al., 2016] Howe, K. B., Suharlim, C., Ueda, P., Howe, D., Kawachi, I., and Rimm, E. B. (2016). Gotta catch'em all! pokémon go and physical activity among young adults: difference in differences study. *British Medical Journal Publishing Group*, 355.

[Ibáñez et al., 2018] Ibáñez, R., Abisset-Chavanne, E., Ammar, A., González, D., Cueto, E., Huerta, A., Duval, J. L., and Chinesta, F. (2018). A multidimensional data-driven sparse identification technique: the sparse proper generalized decomposition. *Complexity*, 2018.

[Ibañez et al., 2017] Ibañez, R., Borzacchiello, D., Aguado, J. V., Abisset-Chavanne, E., Cueto, E., Ladeveze, P., and Chinesta, F. (2017). Data-driven non-linear elasticity: constitutive manifold construction and problem discretization. *Computational Mechanics*, 60(5):813–826.

[Ilardi, 2007] Ilardi, V. (2007). *Renaissance vision from spectacles to telescopes*, volume 259. American Philosophical Society.

[Jeong et al., 2015] Jeong, S., Solenthaler, B., Pollefeys, M., Gross, M., et al. (2015). Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)*, 34(6):199.

[Kalman, 1960] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45.

[Kambhatla and Leen, 1997] Kambhatla, N. and Leen, T. K. (1997). Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516.

[Karhunen, 1946] Karhunen, K. (1946). Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fennicae, ser. Al. Math. Phys.*, 37.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Kirchdoerfer and Ortiz, 2016] Kirchdoerfer, T. and Ortiz, M. (2016). Data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 304:81 – 101.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. *6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR.*, pages 225–234.

[Kolda and Bader, 2009] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[Kölsch and Turk, 2004] Kölsch, M. and Turk, M. (2004). Robust hand detection. *FGR*, pages 614–619.

[Kong and Lucey, 2016] Kong, C. and Lucey, S. (2016). Prior-less compressible structure from motion. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4123–4131.

[Kruskal, 1964a] Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

[Kruskal, 1964b] Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.

[Kutz, 2017] Kutz, J. N. (2017). Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4.

[Ladeveze, 1999] Ladeveze, P. (1999). *Nonlinear Computational Structural Mechanics*. Springer, N.Y.

[Ladeveze and Chamoin, 2011] Ladeveze, P. and Chamoin, L. (2011). On the verification of model reduction methods based on the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2032–2047.

[Ladeveze et al., 2010] Ladeveze, P., Passieux, J.-C., and Neron, D. (2010). The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22):1287 – 1296.

[Lam et al., 2017] Lam, R. R., Horesh, L., Avron, H., and Willcox, K. E. (2017). Should you derive, or let the data drive? an optimization framework for hybrid first-principles data-driven modeling. *arXiv preprint arXiv:1711.04374*.

[Lassila et al., 2013] Lassila, T., Manzoni, A., Quarteroni, A., and Rozza, G. (2013). A reduced computational and geometrical framework for inverse problems in hemodynamics. *International journal for numerical methods in biomedical engineering*, 29(7):741–776.

[Lassila and Rozza, 2010] Lassila, T. and Rozza, G. (2010). Parametric free-form shape design with pde models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23):1583–1592.

[Latorre and Montáns, 2014] Latorre, M. and Montáns, F. J. (2014). What-you-prescribe-is-what-you-get orthotropic hyperelasticity. *Computational Mechanics*, 53(6):1279–1298.

[Lax, 1968] Lax, P. D. (1968). Integrals of nonlinear equations of evolution and solitary waves. *Communications on pure and applied mathematics*, 21(5):467–490.

[LeBlanc and Chaput, 2016] LeBlanc, A. G. and Chaput, J.-P. (2016). Pokémon go: A game changer for the physical inactivity crisis? *Preventive medicine*.

[Lee and Verleysen, 2007] Lee, J. A. and Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer Science & Business Media.

[Lee and Civera, 2019] Lee, S. H. and Civera, J. (2019). Triangulation: Why Optimize? *arXiv e-prints*, page arXiv:1907.11917.

[Levenberg, 1944] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168.

[Liang et al., 2002] Liang, Y., Lee, H., Lim, S., Lin, W., Lee, K., and Wu, C. (2002). Proper orthogonal decomposition and its applications—part i: Theory. *Journal of Sound and vibration*, 252(3):527–544.

[Lin et al., 2004] Lin, H.-W., Tai, C.-L., and Wang, G.-J. (2004). A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. *Computer-Aided Design*, 36(1):1–9.

[Lindberg and Lindberg, 1981] Lindberg, D. C. and Lindberg, D. C. (1981). *Theories of Vision from al-Kindi to Kepler*. University of Chicago Press.

[Liu et al., 2015] Liu, F., Shen, C., and Lin, G. (2015). Deep convolutional neural fields for depth estimation from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170.

[Loève, 1963] Loève, M. M. (1963). *Probability theory*. The University Series in Higher Mathematics, 3rd ed. Van Nostrand, Princeton, NJ.

[Lundgren, 2017] Lundgren, B. (2017). A demonstration of vertical planes tracking and occlusions with arkit+scenekit. `https://github.com/bjarnel/arkit-occlusion`.

[Maday et al., 2002] Maday, Y., Patera, A., and Turinici, G. (2002). A priori convergence theory for reduced-basis approximations of single-parametric elliptic partial differential equations. *Journal of Scientific Computing*, 17/1-4:437–446.

[Maday and Ronquist, 2004] Maday, Y. and Ronquist, E. (2004). The reduced basis element method: application to a thermal fin problem. *SIAM J. Sci. Comput.*, 26/1:240–258.

[Mahmoud et al., 2019] Mahmoud, N., Collins, T., Hostettler, A., Soler, L., Doignon, C., and Montiel, J. M. M. (2019). Live tracking and dense reconstruction for handheld monocular endoscopy. *IEEE transactions on medical imaging*, 38(1):79–89.

[Manzoni et al., 2014] Manzoni, A., Lassila, T., Quarteroni, A., and Rozza, G. (2014). A reduced-order strategy for solving inverse bayesian shape identification problems in physiological flows. In Bock, H. G., Hoang, X. P., Rannacher, R., and Schlöder, J. P., editors, *Proceedings of the Fifth International Conference on High Performance Scientific Computing*, pages 145–155. Springer International Publishing.

[Manzoni et al., 2016] Manzoni, A., Pagani, S., and Lassila, T. (2016). Accurate solution of bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):380–412.

[Manzoni et al., 2012a] Manzoni, A., Quarteroni, A., and Rozza, G. (2012a). Computational reduction for parametrized pdes: Strategies and applications. *Milan Journal of Mathematics*, 80:283–309.

[Manzoni et al., 2012b] Manzoni, A., Quarteroni, A., and Rozza, G. (2012b). Model reduction techniques for fast blood flow simulation in parametrized geometries. *International Journal for Numerical Methods in Biomedical Engineering*, 28(6-7):604–625.

[Manzoni et al., 2012c] Manzoni, A., Quarteroni, A., and Rozza, G. (2012c). Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670.

[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.

[Marsland, 2014] Marsland, S. (2014). *Machine learning: an algorithmic perspective*. Chapman and Hall.

[Martin-Brualla et al., 2018] Martin-Brualla, R., Pandey, R., Yang, S., Pidlypenskyi, P., Taylor, J., Valentin, J., Khamis, S., Davidson, P., Tkach, A., and Lincoln, P. (2018). Lookingood: enhancing performance capture with real-time neural re-rendering. *SIGGRAPH Asia 2018 Technical Papers*, page 255.

[Milgram and Kishino, 1994] Milgram, P. and Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329.

[Milgram et al., 1994] Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1994). Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and telepresence technologies*, 2351(11):282–292.

[Moitinho de Almeida, 2013] Moitinho de Almeida, J. P. (2013). A basis for bounding the errors of proper generalised decomposition solutions in solid mechanics. *International Journal for Numerical Methods in Engineering*, 94(10):961–984.

[Montero et al., 2019] Montero, A., Zarraonandia, T., Diaz, P., and Aedo, I. (2019). Designing and implementing interactive and realistic augmented reality experiences. *Universal Access in the Information Society*, 18(1):49–61.

[Mousavian et al., 2017] Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082.

[Moya et al., 2019] Moya, B., Badías, A., Alfaro, I., Chinesta, F., and Cueto, E. (2019). Digital twins that learn and correct themselves. *International Journal for Numerical Methods in Engineering*.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[Nadal et al., 2015] Nadal, E., Chinesta, F., Díez, P., Fuenmayor, F., and Denia, F. (2015). Real time parameter identification and solution reconstruction from experimental data using the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 296:113 – 128.

[Nee et al., 2012] Nee, A. Y., Ong, S., Chryssolouris, G., and Mourtzis, D. (2012). Augmented reality applications in design and manufacturing. *CIRP Annals-manufacturing technology*, 61(2):657–679.

[Needham et al., 1971] Needham, J., Wang, L., Robinson, K. G., Lu, G.-D., Tsien, T.-H., Ho, P.-Y., Sivin, N., Kuhn, D., Harbsmeier, C., and Golas, P. J. (1971). *Science and Civilisation in China: Physics and Physical Technology*. Cambridge University Press.

[Neron and Ladeveze, 2013] Neron, D. and Ladeveze, P. (2013). Idelsohns benchmark. Technical report.

[Newcombe et al., 2011] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. *International conference on computer vision*, pages 2320–2327.

[Newton, 1833] Newton, I. (1833). *Philosophiae naturalis principia mathematica*. G. Brookman.

[Niépce, 2017] Niépce, N. (2017). Basic photography in 180 days. *Book XII Photographers*.

[Niroomandi et al., 2010] Niroomandi, S., Alfaro, I., Cueto, E., and Chinesta, F. (2010). Model order reduction for hyperelastic materials. *International Journal for Numerical Methods in Engineering*, 81(9):1180–1206.

[Niroomandi et al., 2013] Niroomandi, S., González, D., Alfaro, I., Bordeu, F., Leygue, A., Cueto, E., and Chinesta, F. (2013). Real-time simulation of biological soft tissues: a PGD approach. *International journal for numerical methods in biomedical engineering*, 29(5):586–600.

[Nouy, 2010] Nouy, A. (2010). A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1603–1626.

[Okabe, 2004] Okabe, D. (2004). Emergent social practices, situations and relations through everyday camera phone use. *International Conference on Mobile Communication, Seoul, Korea*.

[Oliehoek, 2018] Oliehoek, F. A. (2018). Interactive learning and decision making: Foundations, insights and challenges. pages 5703–5708.

[Oliver et al., 2009] Oliver, J., Hartmann, S., Cante, J., Weyler, R., and Hernández, J. (2009). A contact domain method for large deformation frictional contact problems. part 1: Theoretical basis. *Computer Methods in Applied Mechanics and Engineering*, 198(33-36):2591–2606.

[OpenAI, 2019] OpenAI (2019). Playing dota 2 world champions.

[Paavilainen et al., 2017] Paavilainen, J., Korhonen, H., Alha, K., Stenros, J., Koskinen, E., and Mayra, F. (2017). The pokémon go experience: A location-based augmented reality mobile game goes mainstream. *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 2493–2498.

[Pachoulakis and Kapetanakis, 2012] Pachoulakis, I. and Kapetanakis, K. (2012). Augmented reality platforms for virtual fitting rooms. *The International Journal of Multimedia & Its Applications*, 4(4):35.

[Paz et al., 2008] Paz, L. M., Piniés, P., Tardós, J. D., and Neira, J. (2008). Large-scale 6-dof slam with stereo-in-hand. *IEEE transactions on robotics*, 24(5):946–957.

[Peckham and Lindberg, 1972] Peckham, J. and Lindberg, D. C. (1972). *Tractatus de perspectiva*. Franciscan Institute.

[Peherstorfer and Willcox, 2015] Peherstorfer, B. and Willcox, K. (2015). Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21 – 41.

[Peng and Mohseni, 2016] Peng, L. and Mohseni, K. (2016). Nonlinear model reduction via a locally weighted pod method. *International Journal for Numerical Methods in Engineering*.

[Perić and Owen, 1992] Perić, D. and Owen, D. (1992). Computational model for 3-D contact problems with friction based on the penalty method. *International journal for numerical methods in engineering*, 35(6):1289–1309.

[Perret and Vander Poorten, 2018] Perret, J. and Vander Poorten, E. (2018). Touching Virtual Reality: A Review of Haptic Gloves. *16th International Conference on New Actuators*, pages 1–5.

[Pomerleau et al., 2013] Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. (2013). Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148.

[Potter et al., 2013] Potter, L. E., Araullo, J., and Carter, L. (2013). The leap motion controller: a view on sign language. *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, pages 175–178.

[Präkel, 2010] Präkel, D. (2010). *The visual dictionary of photography*. Ava Publishing.

[Press et al., 1988] Press, W. H., Teukolsky, S., Vetterling, W., and Flannery, B. (1988). Numerical recipes in c. *Cambridge University Press*, 1:3.

[Prudencio et al., 2013] Prudencio, E., Bauman, P., Williams, S., Faghihi, D., Ravi-Chandar, K., and Oden, J. (2013). A dynamic data driven application system for real-time monitoring of stochastic damage. *Procedia Computer Science*, 18:2056 – 2065.

[Prudencio et al., 2014] Prudencio, E., Bauman, P., Williams, S., Faghihi, D., Ravi-Chandar, K., and Oden, J. (2014). Real-time inference of stochastic damage in composite materials. *Composites Part B: Engineering*, 67:209 – 219.

[Quarteroni et al., 2015] Quarteroni, A., Manzoni, A., and Negri, F. (2015). *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer.

[Quarteroni and Rozza, 2003] Quarteroni, A. and Rozza, G. (2003). Optimal control and shape optimization of aorto-coronaric bypass anastomoses. *Mathematical Models and Methods in Applied Sciences*, 13(12):1801–1823.

[Quarteroni et al., 2011] Quarteroni, A., Rozza, G., and Manzoni, A. (2011). Certified reduced basis approximation for parametrized PDE and applications. *Journal of Mathematics in Industry*, 1(1):3.

[Quesada et al., 2018] Quesada, C., Alfaro, I., González, D., Chinesta, F., and Cueto, E. (2018). Haptic simulation of tissue tearing during surgery. *International journal for numerical methods in biomedical engineering*, 34(3):e2926.

[Raissi et al., 2017] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

[Reddy, 2013] Reddy, J. N. (2013). *An introduction to continuum mechanics*. Cambridge University Press.

[Ren et al., 2013] Ren, Z., Yuan, J., Meng, J., and Zhang, Z. (2013). Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia*, 15(5):1110–1120.

[Rines, 1920] Rines, G. E. (1920). Kodak. *The Encyclopedia Americana*.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Rozza et al., 2008] Rozza, G., Huynh, D., and Patera, A. (2008). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations – application to transport and continuum mechanics. *Archives of Computational Methods in Engineering*, 15/3:229–275.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *International Conference on Computer Vision*, pages 2564–2571.

[Russell and Norvig, 2016] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson Education Limited.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). *IEEE International Conference on Robotics and automation (ICRA)*, pages 1–4.

[Ryckelynck, 2005] Ryckelynck, D. (2005). A priori hyperreduction method: an adaptive approach. *Journal of computational physics*, 202(1):346–366.

[Ryckelynck et al., 2006] Ryckelynck, D., Chinesta, F., Cueto, E., and Ammar, A. (2006). On the a priori model reduction: Overview and recent developments. *Archives of Computational methods in Engineering*, 13(1):91–128.

[Schaaf, 2000] Schaaf, L. J. (2000). *The Photographic Art of William Henry Fox Talbot*. Princeton University Press.

[Schölkopf et al., 1997] Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. *International Conference on Artificial Neural Networks*, pages 583–588.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

[Schuon et al., 2008] Schuon, S., Durković, M., Diepold, K., Scheuerle, J., and Markward, S. (2008). Truly incremental locally linear embedding. *1st International Workshop on Cognition for Technical Systems*.

[Shepard, 1962a] Shepard, R. N. (1962a). The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140.

[Shepard, 1962b] Shepard, R. N. (1962b). The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246.

[Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). *Estimating Uncertain Spatial Relationships in Robotics*, pages 167–193. Springer.

[Stanford, 1993] Stanford (1993). The stanford 3d scanning repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

[Stoica et al., 2017] Stoica, I., Song, D., Popa, R. A., Patterson, D., Mahoney, M. W., Katz, R., Joseph, A. D., Jordan, M., Hellerstein, J. M., and Gonzalez, J. E. (2017). A berkeley view of systems challenges for ai. *arXiv preprint arXiv:1712.05855*.

[Suarez and Murphy, 2012] Suarez, J. and Murphy, R. R. (2012). Hand gesture recognition with depth images: A review. *21st IEEE international symposium on robot and human interactive communication*, pages 411–417.

[Swischuk et al., 2018] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K. (2018). Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*.

[Szeliski, 2010] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer International Publishing.

[Szulakowska, 2000] Szulakowska, U. (2000). *The alchemy of light: Geometry and optics in late Renaissance alchemical illustration*. Brill.

[Taketomi et al., 2017] Taketomi, T., Uchiyama, H., and Ikeda, S. (2017). Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16.

[Takeuchi and Koike, 2017] Takeuchi, I. and Koike, T. (2017). Augmented reality system with collision response simulation using measured coefficient of restitution of real objects. *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, page 49.

[Tomasi and Kanade, 1992] Tomasi, C. and Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154.

[Torresani et al., 2008] Torresani, L., Hertzmann, A., and Bregler, C. (2008). Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):878–892.

[Triggs et al., 1999] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. *International workshop on vision algorithms*, pages 298–372.

[Truesdell, 1960] Truesdell, C. (1960). *The rational mechanics of flexible or elastic bodies*, volume 10. Orell Füssli.

[Tucker, 1966] Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

[Umeta ni and Bickel, 2018] Umeta ni, N. and Bickel, B. (2018). Learning Three-dimensional Flow for Interactive Aerodynamic Design. *ACM Transactions on Graphics (TOG)*, 37(4):89.

[Ullman, 1979] Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426.

[Veltman, 1986] Veltman, K. H. (1986). Leonardo and the camera obscura. *Studi Vinciani in memoria de Nando de Toni*, pages 81–92.

[Welling, 2005] Welling, M. (2005). Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, 3(1).

[Wold et al., 1987] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

[Wu et al., 2013] Wu, H.-K., Lee, S. W.-Y., Chang, H.-Y., and Liang, J.-C. (2013). Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62:41–49.

[Xie et al., 2012] Xie, J., Xu, L., and Chen, E. (2012). Image denoising and inpainting with deep neural networks. *Advances in neural information processing systems*, pages 341–349.

[Yovcheva et al., 2012] Yovcheva, Z., Buhalis, D., and Gatzidis, C. (2012). Smartphone augmented reality applications for tourism. *E-review of tourism research (ertr)*, 10(2):63–66.

[Zhu et al., 2014] Zhu, Y., Huang, D., De La Torre, F., and Lucey, S. (2014). Complex non-rigid motion 3d reconstruction by union of subspaces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1542–1549.

[Zimmer et al., 2015] Zimmer, V. A., Lekadir, K., Hoogendoorn, C., Frangi, A. F., and Piella, G. (2015). A framework for optimal kernel-based manifold embedding of medical image data. *Computerized Medical Imaging and Graphics*, 41:93 – 107.

[Zou et al., 2018] Zou, X., Conti, M., Díez, P., and Auricchio, F. (2018). A nonintrusive proper generalized decomposition scheme with application in biomechanics. *International Journal for Numerical Methods in Engineering*, 113(2):230–251.