



Interactive Physically-based Simulation of the Suction Phenomenon

Antonin Bernardin

► To cite this version:

Antonin Bernardin. Interactive Physically-based Simulation of the Suction Phenomenon. Robotics [cs.RO]. INSA de Rennes, 2021. English. NNT : 2021ISAR0001 . tel-03258570

HAL Id: tel-03258570

<https://theses.hal.science/tel-03258570>

Submitted on 11 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES
APPLIQUÉES RENNES

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

Par

Antonin BERNARDIN

Interactive Physically-based Simulation of the Suction Phenomenon

Thèse présentée et soutenue à Rennes (laboratoire IRISA), le 18/01/2021

Unités de recherche : INSA & IRISA Rennes

Thèse N° : 21ISAR 02 / D21 - 02

Rapporteurs avant soutenance :

Miguel OTADUY Professor, Université Rey Juan Carlos, Espagne
Fabrice JAILLET Maître de conférences HDR, Université Lyon 1

Composition du Jury :

Président :	Bruno ARNALDI	Professeur, INSA de Rennes
Examineurs :	Paul KRY	Associate Professor, Université McGill à Montréal, Canada
	Sheldon ANDREWS	Assistant Professor, ETS Montréal, Canada
Dir. de thèse :	Maud MARCHAL	Professeur, INSA de Rennes
Co-dir. de thèse :	Christian DURIEZ	Directeur de recherche, Inria Lille

RÉSUMÉ EN FRANÇAIS

Contexte et motivations

La thèse s'est déroulée dans le contexte du projet européen H2020 Imagine (<https://imagine-h2020.eu/>) qui a pour objectif de mieux comprendre les actions des robots en imaginant leurs effets. Le concept d'imagination s'apparente dans ce contexte à l'utilisation de techniques d'apprentissage couplées avec des simulations basées physiques afin de prédire les actions d'un robot en réponse à des actions déjà réalisées par le passé. Au sein de ce projet, une pince robotique multi-usages a été conçue afin d'expérimenter la méthodologie proposée dans le contexte du recyclage et du démantèlement d'objets électroniques potentiellement dangereux. La pince réalisée contient notamment une ventouse qui lui permet d'agripper les objets à manipuler (voir Figure 1 pour un exemple de démantèlement d'un disque dur à l'aide d'une simulation réalisée dans le cadre de la thèse). Un des objectifs du projet était donc de pouvoir simuler les effets de succion de cette ventouse afin de réaliser des simulations prédictives des actions de la pince robotique dans différentes situations applicatives.



Figure 1 – Illustration de l'utilisation du modèle de ventouse dans le cadre du projet Imagine: les différentes actions de démantèlement d'un disque dur sont simulées, en utilisant notamment une ventouse fixée à l'extrémité d'un bras de robot. Les simulations sont issues des travaux réalisés pendant la thèse.

La réalisation de simulations interactives et basées physique du phénomène de succion a été très peu étudiée dans la littérature et l'objectif de la thèse était donc de concevoir un nouveau modèle de ce phénomène physique complexe afin de pouvoir réaliser des simulations interactives et réalistes du comportement de ventouses. Au-delà du projet Imagine, un modèle du phénomène de succion trouve son intérêt dans d'autres contextes robotiques, les robots porteurs de ventouses étant utilisés dans plusieurs applications, notamment pour les robots déformables. Des simulations interactives du phénomène de ventouses trouvent également leur intérêt dans d'autres domaines tels que l'animation en informatique graphique ou bien dans les jeux vidéo.

Contributions de la thèse

Les contributions scientifiques de la thèse se situent à plusieurs niveaux: (1) tout d'abord du point de vue de la *modélisation*, nous avons proposé un nouveau modèle basé physique permettant de modéliser le phénomène de succion au sein d'une ventouse en interaction avec son environnement en s'intéressant aux variations de pression à l'intérieur de la cavité de la ventouse, (2) puis du point de vue de la *simulation*, nous avons proposé de nouveaux algorithmes permettant de réaliser une simulation interactive des phénomènes de succion, (3) enfin du point de vue de la *validation*, nous avons proposé d'évaluer notre modèle et les simulations associées au travers de plusieurs protocoles expérimentaux avec des données réelles et simulées.

Un nouveau modèle basé physique pour le phénomène de succion

Notre nouveau modèle basé physique du phénomène de succion est construit à partir de matériaux déformables simulés avec la méthode des éléments finis. Nous avons tout particulièrement choisi la méthode des éléments finis corotationnelle permettant de prendre en compte les grands déplacements. La dynamique de l'air sous la ventouse est modélisée sous la forme de différents états qui dépendent des contacts en cours sur la ventouse ainsi que les différentes forces qui sont appliquées.

Notre première contribution est la mise en place d'un algorithme permettant de détecter les cavités d'air qui surviennent au moment d'une interaction physique entre un objet de type ventouse et un objet de surface quelconque. Cet algorithme est illustré sur la Figure 2 et peut se découper en trois étapes: (1) la surface de contact entre la ventouse et l'objet en collision est déterminée au travers des points de contact localisés par la détection de collision; (2) à l'aide d'une procédure par inondation, la surface interne de la cavité de la ventouse et sa bordure interne (approximée) sont identifiées, les points de contact sont mis à jour en conséquence; (3) la bordure interne est raffinée puis projetée sur l'objet en collision pour faire place à une seconde surface interne qui est associée à la première pour former la cavité géométrique. La topologie des maillages de la ventouse et de la surface sur laquelle elle adhère est dynamiquement modifiée au moment du calcul de la bordure interne raffinée.

Notre deuxième contribution porte sur la modélisation de la dynamique de l'air sous la ventouse. Nous avons fait le choix de modéliser les pressions d'air contenues à l'intérieur des cavités détectées non pas en considérant précisément la dynamique du fluide au travers d'un système de particules, mais en considérant uniquement la distribution de la pression d'air au niveau des parois de la cavité à la place. Ce choix nous permet d'obtenir de meilleures performances en termes de calcul et ainsi obtenir des simulations en temps

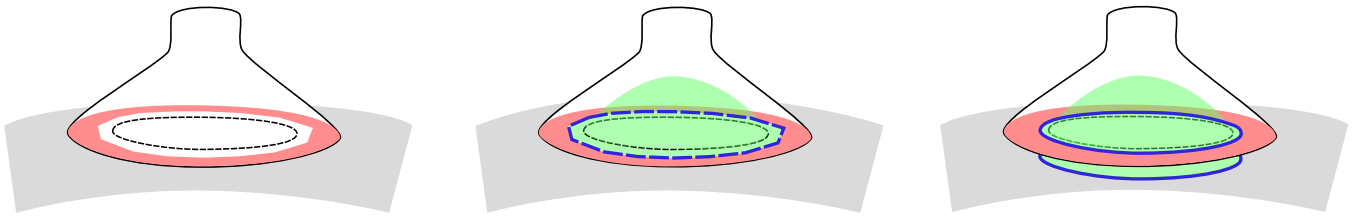


Figure 2 – Etapes de notre algorithme de détection de cavité : A gauche: la surface de contact est déterminée à partir des données de la détection de collision (rouge). Au centre: les surfaces internes (vert) et les bordures internes approximées (ligne bleue en pointillés) sont identifiées au travers d’une procédure par inondation. A droite: Les bordures internes raffinées (ligne bleue continue) sont calculées en utilisant de nouveaux sommets ajoutés au maillage, et les surfaces internes sont ensuite associées par paire pour constituer la cavité géométrique.

interactif. Les pressions d’air des cavités sont formulées sous la formes de contraintes qui sont couplées avec les contraintes de contact et de frottement que l’on retrouve communément dans le domaine de la simulation physique. Deux types de succion sont considérés:

- la succion *active* où la cavité de la ventouse est contrôlée directement par l’utilisateur à l’aide d’une pompe à air par exemple (dans ce cas, la valeur de la pression est connue).
- la succion *passive* qui est celle qu’on retrouve dans le cas d’une ventouse traditionnelle. La résolution des contraintes de pressions sont alors basées sur la loi des gaz parfaits.

Simulation interactive

Nous avons implémenté notre modèle de succion en C++ à l’aide de SOFA, un framework spécialisé dans la simulation d’objets déformables. Ce dernier a l’avantage d’être très modulaire, open source, et aussi doté de multiple fonctionnalités notamment en termes de gestion de collisions et traitement de contraintes. Afin d’optimiser les performances, nous avons utilisé un préconditionneur asynchrone implémenté sur GPU, basé sur une décomposition LDL , disponible dans un plugin SOFA. La décomposition LDL consiste à décomposer la matrice système A sous la forme $A = LDL^T$ avec L une matrice triangulaire et D une matrice diagonale. Cette technique permet de préconditionner le système afin de pouvoir le résoudre de manière plus efficiente.

Afin de tester les aptitudes de notre modèle de succion, nous l’avons soumis à plusieurs scénarios pour lesquels nous avons par ailleurs mesurer les performances de calcul. Les résultats sont indiqués dans le tableau ci-après.

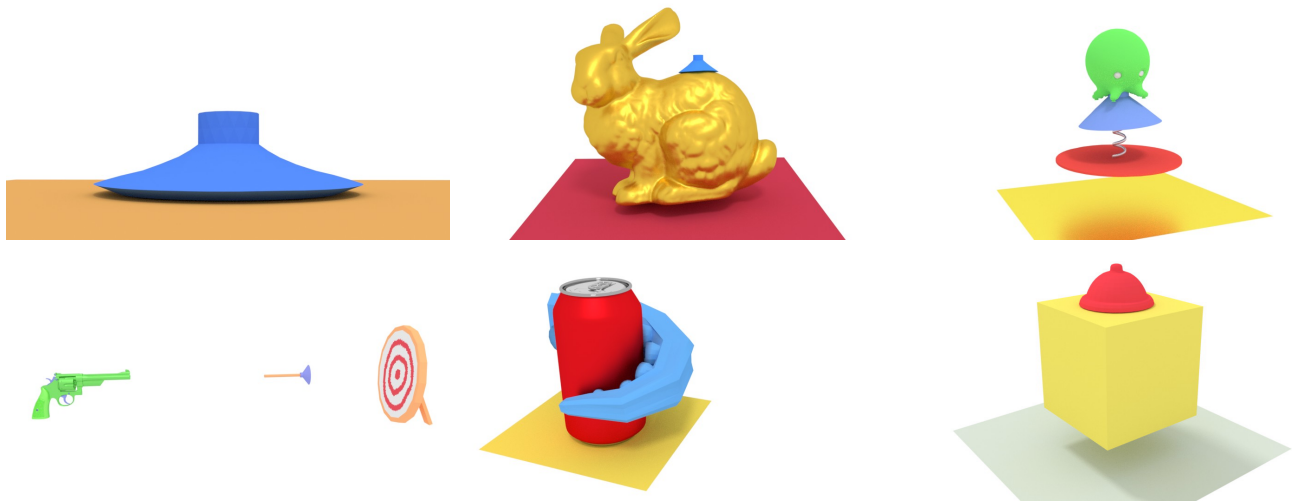


Figure 3 – Aperçu des différents scénarios de simulation pour lesquels nous avons mesuré les performances de calcul.

Les temps de calcul sont indiqués à l'échelle d'une frame (le pas de temps utilisé est de 10 ms), sur la base d'une moyenne d'un échantillon de 150 frames. Ils sont exprimés en millisecondes et sont détaillés pour chaque étape de notre pipeline. Les pourcentages écrit entre parenthèses indique la portion du temps utilisé durant l'étape de manière relative aux autres étapes de la pipeline. Les scénarios dotés d'un astérisques sont ceux n'ayant pas pu bénéficier de l'accélération GPU pour des raisons de stabilité (l'accélération GPU induit des approximations en termes de déformations).

Scénario	Nb de sommets	Nb de tetras	Nb moy. de contraintes	Tps moy. par frame	Mouvement libre	Détection de collision	Détection de cavités	Construction du sys. contraintes	Résolution du sys. contraintes
static plane	576	1685	2406.41	451.39	0.51 (0.23%)	8.31 (3.68%)	33.18 (22.05%)	31.57 (13.99%)	86.13 (57.24%)
rigid bunny	576	1685	2707.56	678.19	0.58 (0.17%)	5.65 (1.67%)	2.58 (0.76%)	97.24 (28.68%)	224.54 (66.22%)
static sphere	576	1685	4527.12	827.68	0.51 (0.12%)	7.84 (1.90%)	1.49 (0.54%)	69.63 (16.82%)	215.19 (78.00%)
rigid cube	576	1685	3164.74	910.04	0.58 (0.13%)	12.53 (2.75%)	56.72 (12.46%)	104.33 (22.93%)	273.71 (60.15%)
octopus arm*	731	1935	878.01	300.15	6.24 (4.16%)	6.47 (4.94%)	22.98 (30.62%)	24.78 (18.01%)	59.50 (39.65%)
toy*	576	1685	1444.31	477.74	27.60 (11.56%)	4.81 (3.67%)	20.92 (8.76%)	15.11 (10.95%)	149.05 (62.40%)
deformable cube*	576	1685	3361.09	1498.57	8.60 (1.15%)	4.89 (0.98%)	36.39 (4.86%)	125.72 (16.78%)	364.42 (72.95%)
dart*	723	2002	2686.25	1133.17	28.09 (4.96%)	13.71 (2.54%)	69.16 (12.21%)	52.00 (9.47%)	393.53 (69.46%)

Pour aller plus loin, nous avons ensuite exécuté le scénario "rigid cube" avec différentes résolutions de maillages puis nous avons comparé les performances. Les résultats sont indiqués dans le tableau ci-dessous:

Nb de noeuds	Nb de tetras	Nb de contraintes	Tps moy. par frame	Mouvement libre	Détection de collision	Détection de cavités	Construction du sys. de contraintes	Résolution du sys. de contraintes
76	205	244.88	11.66	0.10 (1.71%)	0.88 (15.04%)	0.41 (10.55%)	0.81 (13.88%)	0.87 (22.25%)
125	372	452.93	52.95	0.14 (0.54%)	3.56 (13.46%)	8.49 (48.09%)	2.34 (8.84%)	3.33 (18.87%)
141	413	623.56	75.94	0.16 (0.41%)	4.07 (10.73%)	11.56 (45.66%)	3.51 (9.26%)	6.59 (26.03%)
151	449	642.34	75.48	0.17 (0.45%)	3.96 (10.48%)	11.29 (44.86%)	3.68 (9.75%)	6.80 (27.03%)
159	470	830.41	104.65	0.18 (0.35%)	4.55 (8.70%)	14.76 (42.30%)	4.92 (9.40%)	11.61 (33.27%)
203	613	996.07	132.82	0.23 (0.35%)	5.03 (7.58%)	17.39 (39.27%)	6.94 (10.45%)	16.41 (37.07%)
363	1078	2294.55	486.40	0.36 (0.15%)	9.00 (3.71%)	45.15 (27.85%)	26.37 (10.84%)	88.23 (54.42%)
415	1241	2584.75	603.13	0.39 (0.13%)	10.50 (3.49%)	53.29 (26.50%)	32.11 (10.65%)	113.98 (56.70%)
455	1378	2837.21	703.27	0.44 (0.13%)	11.54 (3.29%)	60.05 (25.62%)	39.68 (11.29%)	134.81 (57.51%)
502	1472	3133.83	839.36	0.48 (0.11%)	12.66 (3.02%)	69.39 (24.80%)	48.05 (11.45%)	163.78 (58.54%)
552	1598	3323.07	896.17	0.51 (0.11%)	13.33 (2.98%)	71.87 (24.06%)	51.57 (11.51%)	177.27 (59.34%)
576	1685	3450.77	993.11	0.55 (0.11%)	14.14 (2.84%)	78.33 (23.66%)	55.08 (11.10%)	199.68 (60.32%)

Comme nous pouvons l’observer, plus la résolution des maillages est élevée, plus les performances diminuent. Ceci est lié au fait que des maillages haute résolution induisent une quantité plus importante de points de contacts lors de collisions, et que par conséquent, le nombre de contraintes de contact frictionnel à résoudre augmente. Nous avons pu vérifier, au travers de tests plus approfondis, que les performances sont en effet principalement impactées par le nombre de contraintes à résoudre au cours d’un pas de temps.

Validation

A l’égard du processus de validation de notre modèle de succion, nous avons réalisé plusieurs expériences, d’une part sous la forme de simulations, puis avec des comparaisons avec des données réelles.

Premièrement, dans le contexte de la succion active, nous avons mis en place une expérience afin d’évaluer la géométrie des ventouses simulées, ainsi que les forces générées sur la ventouse. Pour cela, nous avons mis en place un scénario dans lequel nous avons fabriqué plusieurs ventouses en silicone avec des propriétés rhéologiques connues. Nous avons dans un premier temps comparé la géométrie des ventouses réelles avec les ventouses virtuelles de nos simulations pour un même scénario donné. Ce scénario consistait à appliquer à l’intérieur des ventouses une pression que nous contrôlions. La géométrie des ventouses réelles a été reconstruite par le biais de techniques de photogrammétrie afin de pouvoir être comparée avec la géométrie des ventouses virtuelles. Puis, dans un second temps et toujours dans le cas de la succion passive, nous avons réalisé une expérimentation physique qui a consisté à comparer les forces nécessaires pour décoller une ventouse en fonction de son élasticité et de la pression d’air appliquée à l’intérieur de la cavité. Les résultats mesurés ont été comparés avec ceux de nos simulations et montrent que les géométries obtenues et les forces comparées étaient similaires.

De la même manière que pour la succion active, nous avons réalisé une expérience similaire dans le cadre de la succion passive cette fois. Nous avons de plus considéré plusieurs niveaux de courbures pour la surface sur laquelle la ventouse adhère. Les expériences sont illustrées sur la Figure 4. Enfin, nous avons également réalisé plusieurs expériences virtuelles permettant d'observer les limites des capacités de notre modèle de succion.

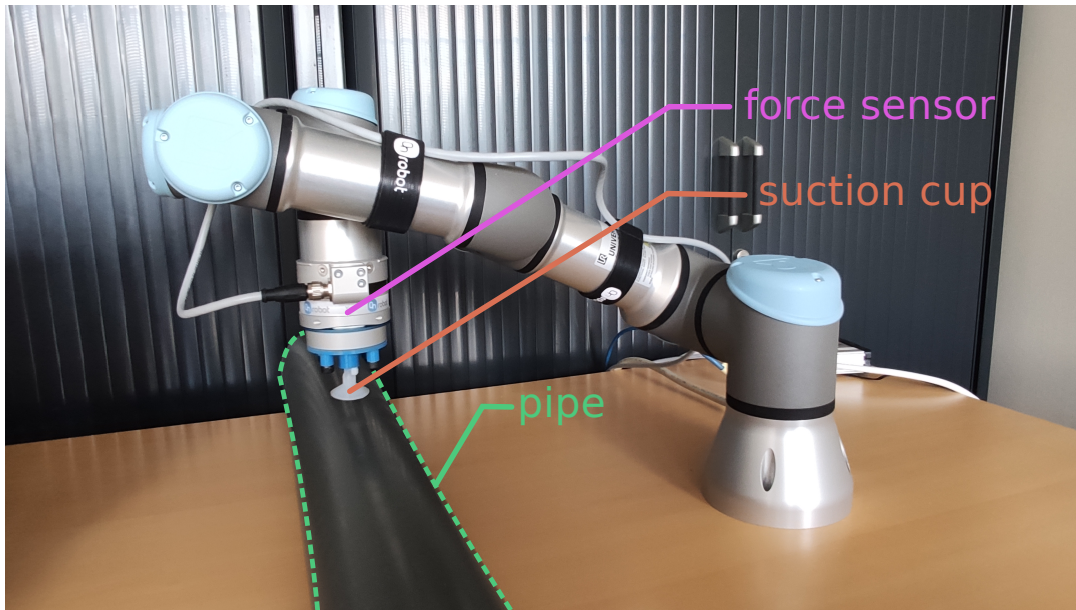


Figure 4 – Expérience menée dans le contexte de la succion passive ayant pour objectif de valider les forces nécessaires pour décoller des ventouses de formes et d'élasticités différentes. La ventouse est appuyée contre une surface puis tirée vers le haut. Plusieurs degrés de courbure ont été testés.

TABLE OF CONTENTS

1	Introduction	13
1.1	Context and motivations	13
1.2	Scientific challenges	16
1.3	Contributions of the manuscript	18
1.4	Published research papers	19
2	State of the Art	21
2.1	Suction phenomenon	22
2.2	Physics-based modeling of deformable objects	23
2.2.1	Spatial integration with FEM methods	25
2.2.2	Mesh-based methods for spatial discretization	30
2.2.3	Mesh-free methods for spatial discretization	32
2.2.4	Time integration schemes	33
2.2.5	Discussion	35
2.3	Collision detection	36
2.3.1	Spatial subdivision	37
2.3.2	Bounding volume hierarchies	38
2.3.3	Distance fields	39
2.3.4	Stochastic methods	41
2.3.5	Image-space techniques	41
2.4	Contact handling and constraints	41
2.4.1	Signorini’s law	42
2.4.2	Coulomb friction model	43
2.4.3	Penalty-based models	43
2.4.4	Impulse-based models	44
2.4.5	Constraint-based models	45
2.5	Linear solvers	47
2.6	Conclusion	48

TABLE OF CONTENTS

3	Modeling & algorithms	49
3.1	Description of the suction phenomenon	50
3.2	Geometric cavity detection	52
3.2.1	Contact surface determination	53
3.2.2	Flood-fill of inner and outer surfaces	54
3.2.3	Inner border refinement	55
3.2.4	Pairing cavity inner surfaces	56
3.2.5	Tracking cavities over time	57
3.2.6	Volume computation	57
3.2.7	Dynamic topology	58
3.3	Numerical methods	59
3.3.1	Simulation pipeline	59
3.3.2	Corotational Finite Element Method	60
3.3.3	Constraint formulation	61
3.3.4	Constraint resolution	63
3.4	Conclusion	66
4	Implementation & Simulation	69
4.1	Implementation	69
4.1.1	General organization of SOFA	70
4.1.2	Cavity detection	70
4.1.3	Constraint system	73
4.1.4	GPU acceleration	74
4.2	Illustrative scenarios	74
4.3	Computation time performance	77
4.4	Conclusion	79
5	Validation	81
5.1	Experiments for validating active suction	81
5.1.1	Geometry validation	82
5.1.2	Force validation	83
5.2	Experiments for validating passive suction	86
5.2.1	Forces and curvatures	86
5.2.2	Virtual experiments	90
5.3	Conclusion	95

6 Conclusion	97
6.1 Summary of the contributions and future work	97
6.2 Short-term possible improvements	98
6.3 Long-term future work and applications	99
Bibliography	99

INTRODUCTION

1.1 Context and motivations

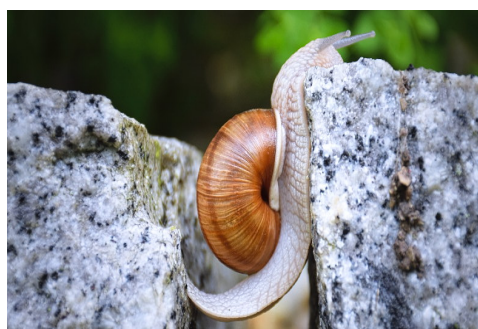


Figure 1.1 – Examples of animals using suction as a mean of locomotion: octopus on the top image and snail on the bottom image.

The overall goal of this thesis is to provide novel approaches based on physics-based simulations in order to improve the dexterity of robots. Our research focus is on the understanding and the simulation of suction phenomenon because of its interesting features in regards to robot manipulation. Furthermore, controlling suction phenomenon would be of high interest in the design of soft robots as well as other applications beyond robotics.

In daily life, we encounter a considerable amount of objects and situations involving suction phenomena. For instance, suction phenomenon occurs when we join and separate our hand palms. Another everyday example is the act of holding a plastic cup around our mouth by inhaling air and then stop breathing. In nature, the locomotion of snails and octopuses also takes advantage of suction (Figure 1.1). But the most obvious example is likely the traditional suction cup. The latter is able to stick to some surfaces and we can grasp objects with it too. Suction cups are a part of many objects such as tea towel hooks, suction cup unblockers, care cups, dart guns, pop up spring toys and glass suction cups (Figure 1.2).

Nowadays, suction cups have been widely adopted in robotics (Figure 1.3). Vacuum grippers for instance are often used through assembly line wherein machines have to manipulate brittle and smooth materials like glass. Indeed, a suction cup is made of a deformable material (silicon) and is consequently suitable to manipulate delicate



Figure 1.2 – Examples of common objects made with suction cups: tea towel hooks, suction cup unblocker, care cups, dart gun, pop up spring toys, glass suction cups.

materials while avoiding damage. In addition, a vacuum pump can be linked to the cavity of the cup in order to reinforce the suction effect and ensure the grasping operation without accidental dropping. Another example is the case of the wall-climbing robots. Some of them are designed with suction cup actuators. Thanks to these actuators, they are able to climb flat surfaces. Beyond that, we think that controlling the suction phenomenon has a great potential in robotics especially with the emerging soft robots. These robots are commonly made of silicon and have the advantage of limiting damage, being flexible and adapt themselves for accomplishing tasks. That is why they are particularly used in surgery. The design of soft robots is mostly inspired by nature. In our context, we could imagine a robot whose locomotion is motivated by octopus. The animal disposes of cups which are arranged along its multiple arms. Each cavity is surrounded by muscles that the octopus can activate in order to increase the volume of this cavity. This action has the effect of inducing a suction phenomenon allowing the animal to stick to surfaces. In fact, suction cups rely on this same principle. Such a robot would be very complex to design and the challenge would be multiple: on the one hand, soft robots tend to have an infinite number of degrees of freedom meaning their motion is hard to predict and control. On the other hand, whereas we perfectly know forcing

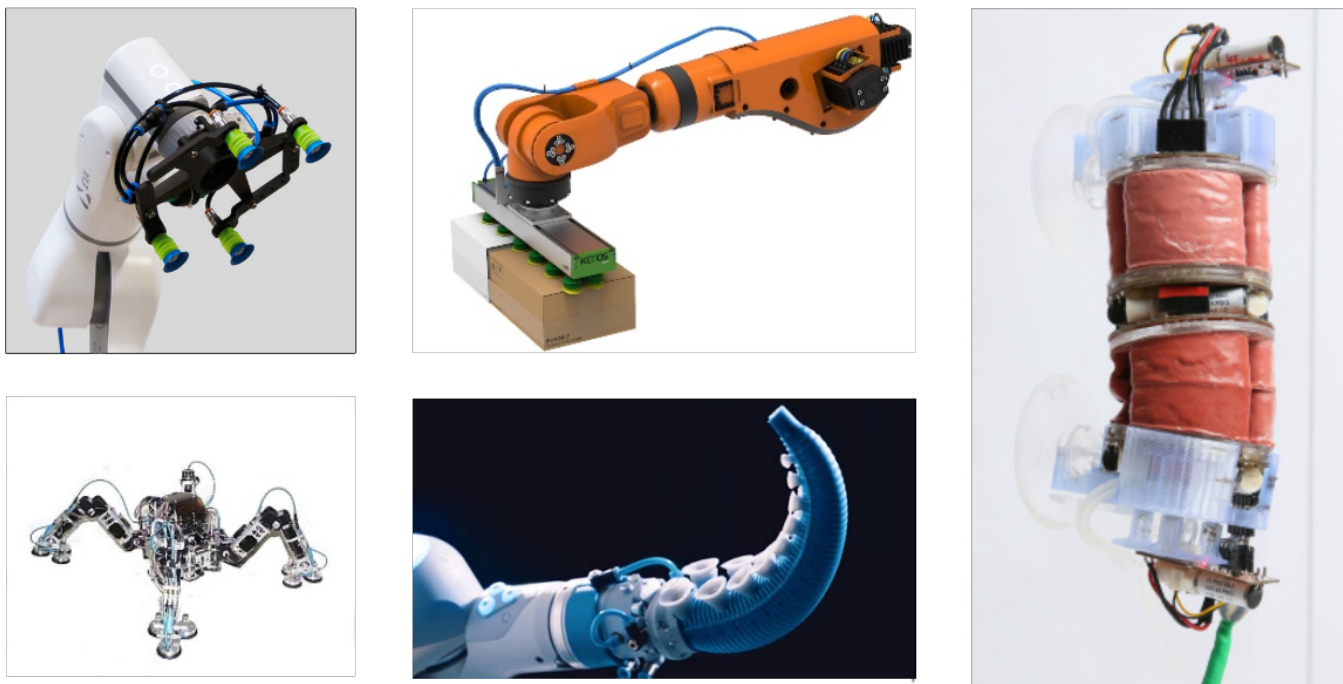


Figure 1.3 – Examples of robots using suction cups: vacuum grippers, wall-climbing robots [RP17], soft pneumatic actuators [XDA⁺20].

suction thanks to a vacuum pump (active suction), dealing with suction without any external device (passive suction) is more difficult especially in terms of control and design. From this observation, we are convinced that physics simulation of the suction phenomenon will become a powerful tool when designing soft robots. This concern about control is even more important in the medical field wherein any motion error could be critical.

This thesis is funded by the European H2020 project Imagine (<https://imagine-h2020.eu>). The overall objective of this project is to give to robots the ability to understand their environment and the way they are affected by their own actions. The project takes place in the context of electronic devices dismantlement and it relies on multiple software components coming from the different research partners. In fact, those components are gathered together in order to constitute the whole artificial intelligence of the robot. An association engine allows the robot to visually analyze the electronic device that it has to dismantle, and then generate a list of actions (manipulation tasks) to complete its mission. When an action seems subject to risks, this one is beforehand simulated thanks to a physics engine which is actually our project contribution as a research partner. On the one hand, robots could be able to judiciously choose their actions and parameters regarding to the simulated performance. On the other hand, they can monitor their progression by comparing the outcome of the real manipulation with the predicted behavior inside the

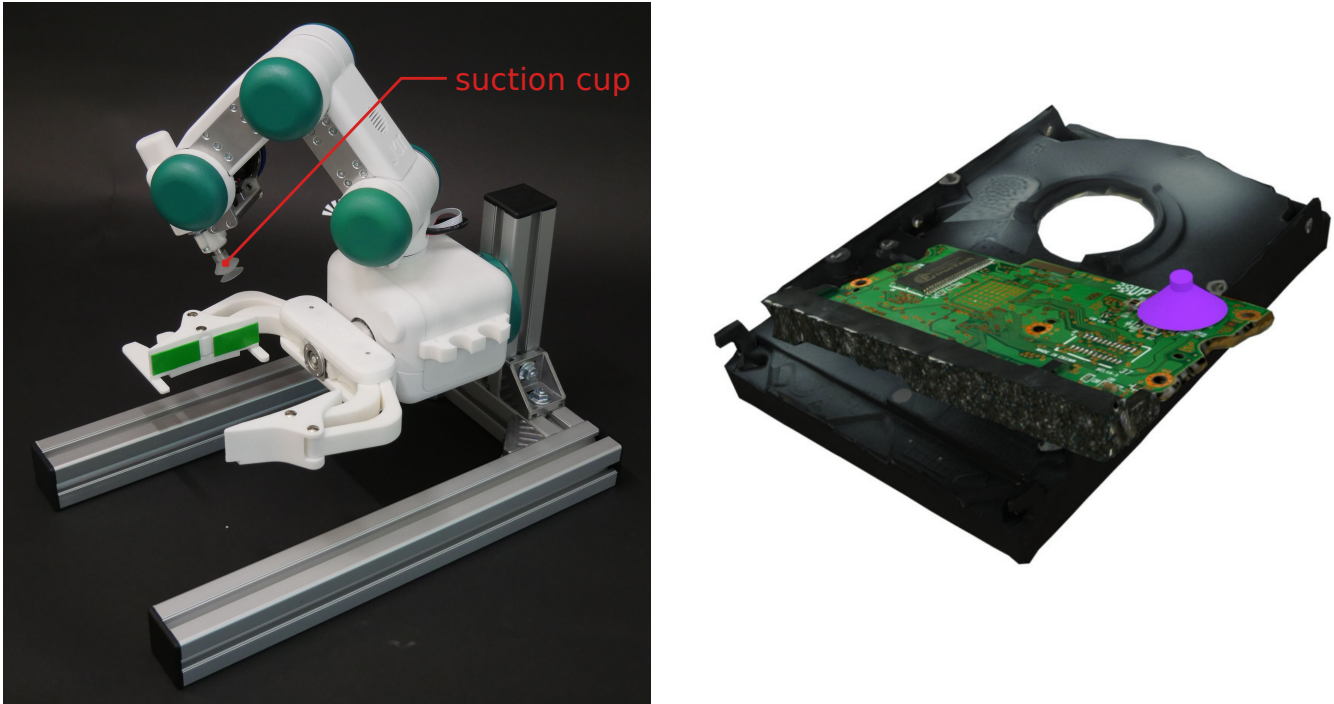


Figure 1.4 – Left: gripper robot from the Imagine project and its actuator equipped with a suction cup (image of courtesy from Karlsruhe Institute of Technology). Right: illustration of a physics-based simulation using our physics-based model of suction cup: the suction cup at the end of a gripper is used to dismantle a hard drive [SHGSA⁺20].

simulation. In this respect, the robot can improve its own performance by learning through the executed scenarios. The robot can use different kinds of actuator for object manipulation purposes. Among these actuators, one is a suction cup as shown in Figure 1.4. For this reason, my task was to realize and provide physics simulation of suction cups in the context of this project.

1.2 Scientific challenges

The suction phenomenon is difficult to control due to its most important features: on the one hand, the rim of the cup must be absolutely sealed to guarantee the air-tightness of the cavity. Otherwise the suction effect would not succeed or it would just partially work. Scenarios involving irregular surfaces like bumpy surfaces are especially difficult to handle. On the other hand, the dynamics of the deformable material are correlated with the dynamics of the fluid (air) which both drive suction. Moreover, friction between the interacting materials has an impact on suction.

Modeling: In terms of modeling challenges, one difficulty is that we deal with deformable objects (suction cups are commonly made of silicon). In addition, they could interact with other involved deformable objects, leading to more complex situations. This aspect means that the modeling of the object dynamics (displacement and distortion) is likely based on continuum mechanics. The air dynamics has to be considered likewise because it plays a crucial role about how the inner air pressure is distributed. A last problem we have to manage is the cavity detection of the suction cup. Indeed, depending on the contact configuration of the suction cup with the colliding object, the rim would be more or less sealed. On top of that, the air trapped in cavity could escape and it would be difficult to estimate the precise amount of air loss.

Simulation: In physics-based simulation, we often look for a trade-off between stability, accuracy, and computation-time performance. In our situation, we mainly focus on robotics applications meaning the accuracy of our simulations is important. But we also would like to be able to obtain simulations at interactive time. However, computation-time performance is faced off by some costly aspects within the simulation: First, the computation of the object deformations itself is already challenging because the object has a lot of degrees of freedom so it often necessitates to solve a large linear system. Secondly, interacting objects implies dealing with collision detection, resulting in a lot of contact points to handle. Thirdly, simulating the dynamics of the fluid (air) could be very costly depending on the numerical method we choose. To finish, the more the scene is complex, the more the above listed elements are amplified. This complexity essentially depends of the granularity of the geometric representations of the simulated objects, the number of involved objects and how much they interact together.

Validation: We distinguish two kinds of suction: active suction and passive suction. Active suction corresponds to the situation wherein the cavity of the cup is connected to a vacuum pump for instance, meaning the inner air pressure is directly controlled by the user. In contrast, passive suction means that the air dynamics inside the cavity is only driven by the distortion of the materials, the friction of the rim, and other effects but without any artificial device. In other words, passive suction coincide with the traditional suction cups scenarios. From a model validation point of view, the results of our virtual physics simulations must be compared with ground truth data. It implies achieving real physics experiments to measure forces, geometry, and other interesting metrics associated to scenarios involving the suction phenomenon. This part is particularly challenging in our context. Among the encountered difficulties, we mention the tracking of the shapes of the deformable objects plus the measure of the pressures and forces. The validation of the model is particularly important in robotics because we want to be sure our simulation is meaningfully close to reality.

1.3 Contributions of the manuscript

We introduced a novel physics-based model of the suction phenomenon in two versions: active suction and passive suction. Our model includes an algorithm which detects the geometries of air cavities which may arise when a suction cup (or another kind of suction object) interacts with another object. In addition, we use a constraint formulation wherein we introduced a new air pressure constraint that we coupled with contact and friction constraints. Besides, we designed several simulation scenarios illustrating our suction model and we measured their computation-time performance. We also made several physical and virtual experiments (additional scenarios) to test and validate our suction model in terms of geometry and forces. These above contributions are detailed in the following.

Regarding numerical methods, we decided to use the popular Co-rotational Finite Element Method [MG04] to simulate the deformations of the objects due to its accuracy. Our first contribution is the elaboration of an algorithm which is able to detect the cavities wherein air is trapped when a suction object interacts with another object. Each detected cavity is associated to mechanical values related to the ideal gas law, namely the internal air pressure, the volume of the cavity and its quantity of air. In terms of modeling, we decided to not represent air dynamics (through a particles system for instance) because, even if the pressure distribution is impacted by the air dynamics, we think this aspect is negligible in our context. Instead, we solely consider a uniform pressure distribution along the walls of the cavities. This simplification allows us to obtain better time performances and therefore achieve interactive simulations. Our second contribution is that we formulated the air pressures inside cavities as constraints coupled with the traditional contact and friction constraints we use in physics simulation. The resolution of these pressure constraints is based on the ideal gas law in the case of passive suction (the pressure value is already known in the case of active suction).

We have established several simulation scenarios for illustrative and benchmark purposes about our suction model. We use an asynchronous GPU-implemented preconditioner which accelerates the resolution of the constraint system. The time performances of our simulations has been measured and compared using different mesh resolutions. We observed that the time performance is essentially slowed down by the numerous contact and friction constraints. As expected, the more the resolution of the meshes is high, the more we have contact points meaning more constraints to solve. Thus, computation-time performance is basically proportional to the degree of granularity of the meshes.

Our last contribution is the validation of our model through several physical and virtual experiments. Regarding the active suction version of our model, we compared the geometry of photogrammetry-

reconstructed suction cups coming from reality with the simulated ones. To achieve this, we linked the cavities of the real cups to a vacuum pump with a regulator in-between in order to obtain the desired material configurations. Still about active suction, we measure the necessary pull forces to detach our suction cups in a real experiment, and we compared the results with the ones from our simulation. Later, we did a similar experiment in the context of passive suction, except we additionally tested different degrees of surface curvatures (contact surface). Furthermore, we also made some virtual experiments to illustrate the features of our suction model. Roughly, they consist of grasping objects by suction and test different object shapes and curvatures.

To summarize, our research focuses on the modeling and simulation of suction phenomenon due to its very interesting features in regards to object manipulation purposes, soft robots locomotion and design. The organisation of this thesis is as follows: In Chapter 2, we sketch the state of the art of the suction phenomenon: physics-simulation of deformable objects, collision detection, constraint formulation and resolution. In Chapter 3, we explain our modeling and the algorithmic aspects of our method. In Chapter 4, we provide some implementation details, our illustrative scenarios, and the measured time performances. In Chapter 5, we detail the physical and virtual experiments we made to validate our model. After that, we conclude with a discussion of our work.

1.4 Published research papers

- Antonin Bernardin, Christian Duriez, and Maud Marchal. An interactive physically-based model for active suction phenomenon simulation. In IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), 2019.
- Alejandro Suárez-Hernández, Thierry Gaugry, Javier Segovia-Aguas, Antonin Bernardin, Carme Torras, Maud Marchal, and Guillem Alenyà. Leveraging multiple environments for learning and decision making: a dismantling use case. In IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), 2020.
- Antonin Bernardin, Guillaume Cortes, Rebecca Fribourg, Tiffany Luong, Florian Nouviale, and Hakim Si-Mohammed. Toward intuitive 3d user interfaces for climbing, flying and stacking. In IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pages 860–861, 2018.

STATE OF THE ART

Contents

2.1	Suction phenomenon	22
2.2	Physics-based modeling of deformable objects	23
2.2.1	Spatial integration with FEM methods	25
2.2.2	Mesh-based methods for spatial discretization	30
2.2.3	Mesh-free methods for spatial discretization	32
2.2.4	Time integration schemes	33
2.2.5	Discussion	35
2.3	Collision detection	36
2.3.1	Spatial subdivision	37
2.3.2	Bounding volume hierarchies	38
2.3.3	Distance fields	39
2.3.4	Stochastic methods	41
2.3.5	Image-space techniques	41
2.4	Contact handling and constraints	41
2.4.1	Signorini's law	42
2.4.2	Coulomb friction model	43
2.4.3	Penalty-based models	43
2.4.4	Impulse-based models	44
2.4.5	Constraint-based models	45
2.5	Linear solvers	47
2.6	Conclusion	48

As teased in Section 1.3, we elaborated a physically-based model of the suction phenomenon that can notably be applied to simulate suction cups. Obviously, such a simulation does not solely depend upon our contributions. It also relies on existing methods in matter of physics simulation of deformable objects.

Because of this, the purpose of this chapter is to sketch an overview of related work while explaining the different modeling and numerical approaches regarding time integration, object distortion, collision detection, and contact handling.

The chapter is organized as follows: Section 2.1 draws existing work related to the modeling of the suction phenomenon. Section 2.2 gives an overview of the state-of-the-art methods and related background to simulate deformable objects, with a specific focus on the Finite Element Method that we use in the context of our work. Next, some explanations about time integration schemes are given plus a comparison between the different simulation methods. Section 2.3 deals with collision detection algorithms having the special ability to handle self-collision, an important feature in the case of soft bodies. Section 2.4 is about the different approaches to simulate contacts between colliding objects. In particular, we detail the constraint-based approach and the underlying possible constraint formulations knowing that our model relies on a novel air pressure constraint. Finally, Section 2.5 quickly discusses about linear solvers while mentioning some commonly used ones.

2.1 Suction phenomenon

A significant amount of work have been recently published in order to improve the design of suction cups. One problem with traditional rubber suction cups is that they are not suitable to grip rough surfaces due to air leakages. To tackle this issue, Xin [XZK⁺14] developed a new pneumatic sucker having a cylindrical cavity in which a rotating air flow is produced through tangential nozzles. The air flow generates a cupped negative pressure distribution which creates a suction phenomenon. The author compared its new sucker to traditional ones and proved that it is superior in handling rough work-pieces. Other recent works seek to design very small suction cups: Knowing the traditional suction mechanism with an air pump is difficult to miniaturize, Hu [HY18] designed a valveless piezoelectric pump which is integrated into a micro-suction cup. Additionally, Nishita [NO17] proposed a novel liquid-filled flexible micro suction-controlled array (MISCA) for humanoid robot hands. The array includes multiple tiny suction units onto a sheet (membrane) of 1 centimeter squared area, made of a flexible material. The latter lays onto a body wherein a micro-channel network was dug to let an incompressible fluid circulate. When the array is put against a surface and that the syringe pump is then enabled, the fluid is moved in a way the membrane distorts itself. It has the effect of increasing the volumes of the suction units cavities and decreasing the respective air pressures in return, leading to suction phenomena. This system, which looks similar to the suckers of an octopus arm, allows robots to hold curved or grooved surface objects. Speaking of octopus, there exists an impressive amount of bio-inspired robots and actuators especially in the domain of soft robots. One special mention

would be the octopus-like suction cups from Tramacere [TFPM15].

Whereas suction cups are widely used in Robotics, physics-based modeling of the suction phenomenon has been reported by relatively few papers in the literature. One of the earlier, but relevant, work on the topic from Bahr [BLN96] focuses on the design and analysis of a climbing robot that was equipped with suction cups at the tip of its two legs. In their work, they established a theoretical analysis of the suction cup to find the minimal vacuum pressure for avoiding both falling down and sliding. Some years later, Liu [LTBY06] presented an analytic modeling of suction cups for window-cleaning robots, in which the friction was omitted by considering that the window surface was always perfectly wet. Through a quantitative study of the attachment and detachment of a passive suction cup, Ge [GMS⁺15] modeled the forces acting on the cup considering only the vertical ones for a sake of simplicity. More recently, Mahler [MML⁺18] proposed a compliant suction contact model wherein the suction cup is seen as a quasi-static spring system. Their model estimates the deformation energy to maintain a seal and it quantifies the ability to resist to external wrenches. Nevertheless, and to the best of our knowledge, there is no existing physics-based model of suction cup with a coupling between its deformation and the inner pressure. Such methods require simulations of both deformable models and contact model between different objects. We therefore gave a response to this challenge through our suction model presented in this thesis.

2.2 Physics-based modeling of deformable objects

Soft body simulation is often based on physical laws and considerations coming from the field of continuum mechanics: An elastic object is usually defined by its undeformed shape, also called rest configuration, and by its set of material parameters such as its modulus of elasticity. The closed surface of the rest shape is considered continuous and defines the boundary of the material domain D of the object. We assume \mathbf{m} the material coordinates (location in the initial configuration) of an arbitrary particule in the domain such as $\mathbf{m} \in D$, with $\mathbf{x} = \mathbf{x}(\mathbf{m}, t)$ its position at a time t and $\dot{\mathbf{x}} = \dot{\mathbf{x}}(\mathbf{m}, t)$ its associated velocity vector.

When forces are applied to the object, each material particle \mathbf{m} is moved to a new location \mathbf{x} (Figure 2.1). From the displacement vector field $\mathbf{u} = \mathbf{x} - \mathbf{m}$, we can calculate the elastic strain ϵ , a dimensionless quantity which is simply equal to $\Delta l / l$ in the linear 1D case with l the rest length of a spring and Δl the offset between its rest length and its current length. A spatially constant displacement field represents a translation of the object with no strain. From this observation, we now understand that the strain of the deformable object is directly related to the variation of its displacement field \mathbf{u} . For elastic bodies, the strain

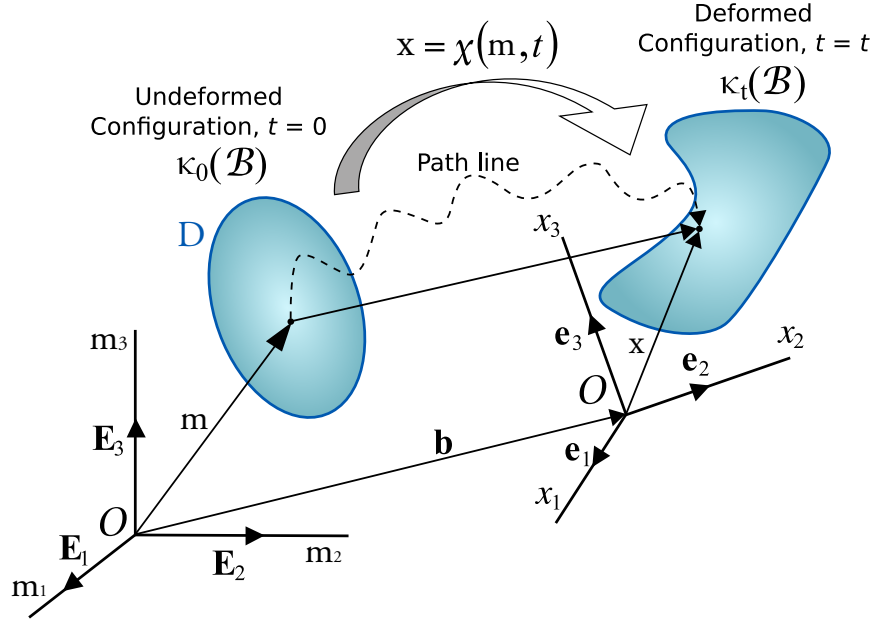


Figure 2.1 – Continuum mechanics: a deformable object moves from its initial configuration to a deformed configuration.

is often described using the Green's nonlinear symmetric strain tensor ϵ_G :

$$\epsilon_G = \frac{1}{2}(\nabla \mathbf{u} + [\nabla \mathbf{u}]^T + [\nabla \mathbf{u}]^T \nabla \mathbf{u}) \quad (2.1)$$

However, in case of small displacements, the nonlinear part of the definition can be neglected, leading to a linearization of ϵ_G , also known as Cauchy's linear strain tensor ϵ_C :

$$\epsilon_C = \frac{1}{2}(\nabla \mathbf{u} + [\nabla \mathbf{u}]^T) \quad (2.2)$$

with the gradient $\nabla \mathbf{u}$ representing the spatial derivatives along each axis of our euclidean space.

When a soft body is strained by external forces, it implies internal forces which appear as a consequence. These forces, which are called internal stresses, tend to return the strained body to its rest configuration. It means there exists a law which connects the stress tensor with the strain tensor according to the mechanical properties of the deformable object. It is the purpose of a constitutive law. If the material is considered as perfectly elastic and isotropic, the linear Hooke's law can be used, which is a popular choice in computer graphics:

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \gamma \text{tr}(\boldsymbol{\epsilon})\mathbf{I} \quad (2.3)$$

with \mathbf{I} the identity matrix and tr the trace function. The term μ is the first Lamé coefficient and γ is the second Lamé coefficient, also called shear modulus. These two coefficients only depend of two properties of the material which are respectively the Young modulus Y and the Poisson ratio ν such that:

$$\mu = \frac{Y}{2(1 + \nu)} , \quad (2.4)$$

$$\gamma = \frac{Y\nu}{(1 + \nu)(1 - 2\nu)} . \quad (2.5)$$

The Young modulus represents the elasticity of the material, whereas the Poisson's ratio characterizes the ratio of contraction or expansion in perpendicular directions relative to the applied load.

Deformable object modeling is often governed by continuum mechanics but computer computation capabilities are however limited. In other words, simulation of soft bodies needs to represent the geometry of the objects as non-continuous representations (see Section 2.2.1 and Section 2.2.2). And the same idea applies for the representation of time, for which a time integration scheme has to be used (see Section 2.2.4). Physics simulation needs numerical methods to discretize time and space.

Several kinds of spatial discretization exist in the literature. They can be divided into two classes according to the way they spatially represent the objects. On the one hand, the mesh-based methods represent the deformable object using a mesh. On the other hand, the mesh-free methods represent the object using something else than a mesh. All mesh-based methods and the mesh-free methods are detailed and compared in Section 2.2.2. The only exception is the mesh-based Finite Element Method which has a dedicated Section (Section 2.2.1) because it is a popular method and the one we use in the context of our work.

2.2.1 Spatial integration with FEM methods

The dynamic behavior of the object is governed by the Newton's second law which can conveniently be formulated as an ordinary differential equation (ODE):

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} = \mathbf{g}(t) - \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) \quad (2.6)$$

where the right-hand terms represents the gathering of all forces which impact the motion of the deformable object, while $\mathbf{M}(\mathbf{x})$ represents its constant diagonal mass matrix and $\ddot{\mathbf{x}}$ its acceleration. Two types of forces are considered in this equation: $\mathbf{g}(t)$ the external forces which apply onto the object's surface and $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$

the internal forces. Assuming we deal with elastic objects, these internal forces are actually elastic forces. They try to restore the object to its rest configuration from inside.

In the following, the Newton's second law will be denoted $\ddot{\mathbf{x}} = N(\mathbf{x}, \dot{\mathbf{x}}, t)$. One of the most popular numerical methods for deformable object is the Finite Element Method (FEM). This one performs a discretization (mesh) of the deformable object and often transforms the partial differential equation of motion (PDE) into a piece-wise linear function (ODE). The Finite Element Method (FEM) considers the deformable objects as continuous volumes but discretized as irregular meshes, giving a finite number of elements. The latter are typically hexahedra or tetrahedra in order to capture the volumetric aspect of the object in terms of deformation computation. Besides, triangles can be used when the simulation takes place within a 2D euclidean space. Even if acceleration techniques exist, one drawback of the FEM is typically the slowness of the simulation, especially when dealing with high resolution meshes. However, one advantage of the method is it respects the law of conservation of matter.

Knowing the spatial representation of the simulated objects is done through irregular meshes, the FEM needs to solve Partial Differential Equations (PDE) on them. The PDE which is used to simulate dynamic elastic materials is formulated as follows:

$$\rho \ddot{\mathbf{u}} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{g} \quad (2.7)$$

where ρ is the density of the material, \mathbf{u} the displacement vector, \mathbf{g} the external forces and ∇ the divergence operator. This equation is often approximated with a very specific subset of functions which are solved numerically. Some methods use linear interpolation within each element. The consequence is that it restricts the solution to a piece-wise linear function. However, the solution can be represented more accurately by using quadratic functions instead of piece-wise linear functions, or even more elaborated bases. Knowing that in the FEM, the continuous deformation is approximated by a sum of linear basis functions (one per each node of the mesh), the goal is to find all the unknown positions $\mathbf{x}_i(t)$ for the next time step, according to the following equation:

$$\mathbf{x}(\mathbf{m}, t) = \sum_i \mathbf{x}_i(t) b_i(\mathbf{m}) \quad (2.8)$$

where b_i is the Kronecker Delta property. In the explicit FEM, the masses, the internal forces, and the external forces are lumped to the vertices. The strain field $\boldsymbol{\epsilon}(\mathbf{m})$ and stress field $\boldsymbol{\sigma}(\mathbf{m})$ are computed from $\mathbf{u}(\mathbf{m})$. Then, the deformation energy can be determined according to the following equation:

$$E = \oint_D \boldsymbol{\epsilon}(\mathbf{m}) \cdot \boldsymbol{\sigma}(\mathbf{m}) d\mathbf{m} . \quad (2.9)$$



Figure 2.2 – Large rotational deformations yields to inflation artifacts when not using Co-rotational Finite Element Method [MG04].

After that, the forces can be computed as the derivatives of the energy according to the nodal positions. The relationship between nodal forces and nodal positions is often nonlinear but can be linearized as:

$$\mathbf{f}_e = \mathbf{K}_e \mathbf{u}_e \quad (2.10)$$

for which \mathbf{f}_e contains the n_e nodal forces and \mathbf{u}_e the n_e nodal displacement of an element. Thus, $\mathbf{K}_e \in \mathbb{R}^{3n_e \times 3n_e}$ is the stiffness matrix of the element e . The stiffness matrix of the entire mesh can be expressed as $\mathbf{K} = \sum_e \mathbf{K}_e$. Here \mathbf{K} is a sparse matrix containing zeros at positions related to nodes not adjacent to the element. Once \mathbf{K} has been determined, the linear algebraic equation of motion can be used at each time step of the simulation (between time t_0 and t with $\mathbf{u} = \mathbf{x} - \mathbf{x}_0$):

$$\mathbf{g} = \mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} \quad (2.11)$$

with \mathbf{M} the mass matrix, \mathbf{C} the damping matrix and \mathbf{g} the externally applied forces. Besides, \mathbf{M} and \mathbf{C} can be transformed to diagonal matrices using a technique called mass lumping. If we use a linear strain measure and Hooke's law for isotropic materials (a material having identical values of a property in all directions), the PDE to solve becomes the Lamé's linear PDE:

$$\rho \ddot{\mathbf{x}} = \nu \Delta \mathbf{u} + (\mu + \gamma) \nabla (\nabla \cdot \mathbf{u}) \quad (2.12)$$

wherein the constants μ and γ are respectively the first and second Lamé coefficients that we have defined in Equation 2.4 and Equation 2.5.

Co-rotational FEM: Although using linear PDE's yields linear algebraic systems which can be solved faster while keeping a good stability compared to nonlinear PDE, linearized elastic forces are only valid

for small deformations. Indeed, as it is represented on Figure 2.2, large rotational deformations yield highly inaccurate restoring forces, causing inflation artifacts on parts on which the object is distorted. In order to avoid this artifact, Müller et al. [MG04] proposed a technique based on his previous work called Stiffness Warping [MDM⁺02]. The author computes the elastic forces for every element (tetrahedron) in a local unrotated coordinate frame. This method is now commonly called co-rotational in the community. Assuming we know the rotational part \mathbf{R}_e of the deformation of the tetrahedron e and its stiffness matrix \mathbf{K}_e , the force \mathbf{f}_e on the element is computed as:

$$\mathbf{f}_e = \mathbf{R}_e \mathbf{K}_e \cdot (\mathbf{R}_e^{-1} \mathbf{x} - \mathbf{x}_0) = \mathbf{R}_e \mathbf{K}_e \mathbf{R}_e^{-1} \mathbf{x} - \mathbf{R}_e \mathbf{K}_e \mathbf{x}_0 = \mathbf{K}'_e \mathbf{x} + \mathbf{f}'_{0_e} \quad (2.13)$$

The elastic forces of the entire mesh are given as $\mathbf{f} = \mathbf{K}' \mathbf{x} + \mathbf{f}'_0$ wherein \mathbf{K}' is the global stiffness matrix. \mathbf{f}'_0 represents the sums of the element's rotated stiffness matrices $\mathbf{K}'_e = \mathbf{R}_e \mathbf{K}_e \mathbf{R}_e^{-1}$ and force offsets $\mathbf{f}'_{0_e} = \mathbf{R}_e \mathbf{f}_{0_e}$ with global vertex numbers. By using the co-rotational technique, there is no ghost force because the elastic forces in \mathbf{f}_e are guaranteed to sum up to zero, meaning the forces in \mathbf{f} sum to zero too.

Reduced deformation models: In order to speed up the simulation at the expense of loosing a bit of accuracy, one solution is to use reduced deformation models. A reduced deformation model approximates deformed point locations \mathbf{x} by a linear superposition of n displacement fields, given by the columns of \mathbf{U} . As illustrated in Figure 2.3, the amplitude of each displacement field is given by \mathbf{q} the reduced coordinates, such that $\mathbf{x} = \mathbf{x}_0 + \mathbf{U} \mathbf{q}$. One major way to obtain reduced deformable models is to use modal analysis [PW89]. The principle behind using modal analysis for deformable objects is the decoupling of the linear algebraic equation $\mathbf{M} \ddot{\mathbf{u}} + \mathbf{C} \dot{\mathbf{u}} + \mathbf{K} \mathbf{u} = \mathbf{f}$ into several linearly independent ODE, by use of the whitening transform which simultaneously diagonalizes \mathbf{M} , \mathbf{C} and \mathbf{K} . And to find the whitening transform, it is necessary to solve the eigenvalue problem $\Lambda \Phi = \mathbf{M}^{-1} \mathbf{K} \Phi$ where Λ and Φ are the eigenvalues and eigenvectors of $\mathbf{M}^{-1} \mathbf{K}$. Note that Φ is likewise called the modal matrix. One interesting aspect with modal analysis is that, once the analysis has been performed, we are able to select only a few modes in order to reduce the number of degrees of freedom (DOF).

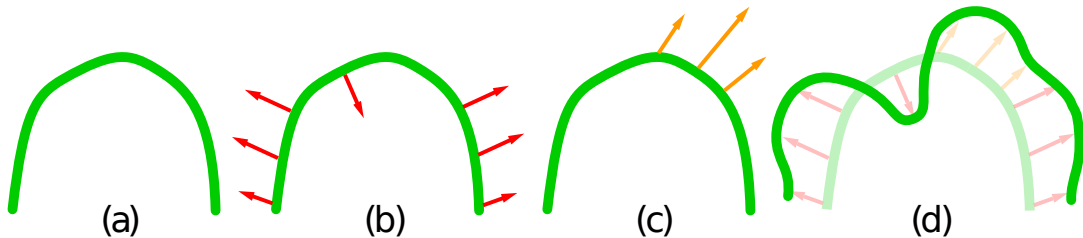


Figure 2.3 – Reduced deformation models [NMK⁺06]: (a) reference shape \mathbf{x}_0 , (b) displacement field \mathbf{U}_1 , (c) displacement field \mathbf{U}_2 , and (d) one possible deformed shape $\mathbf{x} = \mathbf{x}_0 + \mathbf{U}_1 + 0.5\mathbf{U}_2$.

After the introduction of modal analysis by Pentland [PW89], a few significant contributions have been presented since then: Stam [Sta97] applied modal analysis to the simulation of tree branches subjected to turbulence. Hauser [HSO03] combined constraints with reduced deformation models to achieve fast and stable simulations with complex objects. James and Pai [JP02] invented a precomputed Dynamic Response Texture (DyRT) storing mode shapes and other quantities into graphics hardware. In terms of implementation, displacements are efficiently applied to the undeformed mesh thanks to a vertex program. Barbic and James [BJ05] presented an approach for fast subspace integration of reduced-coordinate nonlinear deformable models for interactive applications. James [JP04] introduced the Bounded Deformation Tree (BD-Tree), a collision detection method using reduced deformable models wherein costs are comparable to collision detection with rigid objects. James and Fatahalian [JF03] elaborated a reduced deformation model without using modal analysis. Instead, the approach consists in precomputing data-driven models of interactive physically based deformable scenes meaning that collision, contact, and dynamics of the deformable objects are all precomputed. To support runtime interaction, the method builds a lookup table of Impulse Response Functions (IRF) for a limited range of external impulses from the user. These IRFs are dimensionally reduced through the technique of Principal Component Analysis (PCA), and are blended based on the user interaction. Another data-driven model has been elaborated by Wang [WObR11]. The author developed a new measurement techniques for studying the elastic deformations of real cloth samples. And he additionally proposed a piece-wise linear elastic model having parameters which can be fit to observed data with a well-posed optimization procedure. The main purpose of this technique is to accurately animate clothes materials such as silk or denim for which elastic behavior deals with anisotropic stretching and bending phenomena.

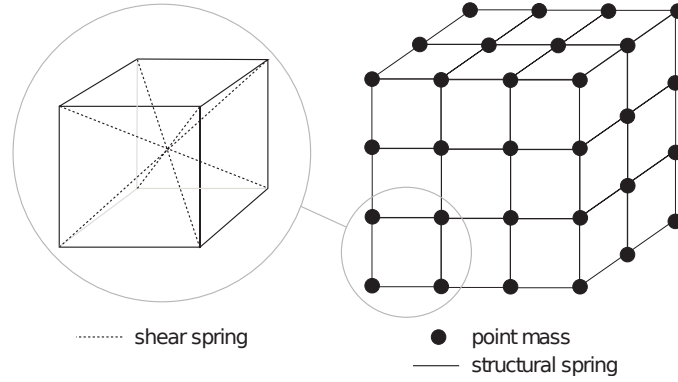
Adaptive models: Another kind of model allows to speed up simulation: the Adaptive Models. Regardless of the chosen method, choosing the degree of space discretization is part of the challenge of achieving a good trade-off between accuracy and computational resources. This observation leads to the emerging spatial adaptive models: The principle is to dynamically adapt the spatial resolution of the simulated object according to the current situation. According to one or several carefully chosen criteria, the method dynamically determines on which geometric regions higher resolution is needed. Then, a refinement/coarsening strategy allows the dynamic modification of the geometric representation of the simulated object to adapt it to the desired resolution. The quality of the simulation depends a lot of the refinement criteria. Often, simple heuristics such as the distance to boundaries, surface curvature, presence of contacts are used. Otherwise, the refinement criteria can be tied to the dynamics of the system, especially in case of elastic and plastic solids. Wicke et al. [WRK⁺10] used the strain gradient as the refinement criterion for dynamic local remeshing. In this way, regions undergoing severe deformation are refined locally. Another approach

is to perform view-dependent adaptivity using a refinement criteria based on visibility and distance from the camera. It means refining the coarser regions which come into the field of view. Koh et al. [KNO14] invented a technique to increase the resolution in advance using camera path information. Depending on the chosen numerical method to simulate the deformation of the object (see Table 2.1), accelerating spatial structures can be used to speed up the simulation. These structures are either structured meshes, grids (kD-trees for instance), or unstructured meshes. Structured meshes allows more core re-usability when converting from a regular grid to an adaptive one whereas fully unstructured meshes gives more flexibility. Note that time-stepping adaptivity can also be applied, by using freezing techniques or Asynchronous Variational Integrators [TPS08] for instance. For more information about adaptive models, the reader can refers to the survey from Manteaux [MWN⁺17].

Geometry of simulated objects can be represented in different ways. They can be split into two categories: the mesh-based methods and the mesh-free methods. Mesh-based methods, like the Finite Element Method (FEM), represent the simulated objects using meshes. Indeed, FEM is actually an accurate mesh-based method which discretize the geometry of the object while considering volume and energy. However, FEM is rather slow to compute and it is sometimes necessary to turn towards efficient mesh-based methods. The efficient mesh-based methods often uses rough representations of the simulated objects, using regular grids for instance. Contrary to mesh-based methods, the free-based methods represent a given simulated object as something else than an mesh. They typically use particles. It induces no mesh generation nor runtime re-meshing operations, meaning a better suitability when dealing with spatial adaptation. Another interesting feature of free-based methods is the easy construction of complex shape functions allowing to obtain any desired order of spatial continuity. However, these shape functions often have to be re-evaluated at each time-step. Consequently, the free-based methods are generally more time-consuming than mesh-based methods.

2.2.2 Mesh-based methods for spatial discretization

Mass-spring system: Mass-spring systems [Kim20] (Figure 2.4) are popular in some real-time applications like video game wherein the computation speed is crucial. These systems are especially fast when combined with an explicit time-integration scheme (see Section 2.2.4). Instead of solving the equation of motion as a PDE, the deformable model is a set of point masses which are linked together by a network of springs (mesh). The motion of each point mass is expressed as a single Ordinary Differential Equation (ODE) which corresponds to the Newton's second law (equation of motion). Consequently, a mass spring system amounts to solve a system of coupled Ordinary Differential Equations (ODE) at each time step, which is relatively fast. It is a simple and intuitive model in spite of a lack of precision.

Figure 2.4 – Mass spring system [NMK⁺06].

In a mass-spring system, each mass point i also carries its respective position \mathbf{x}_i and its velocity $\dot{\mathbf{x}}_i$ both varying over time. At each time step, the force \mathbf{f}_i of the mass point i is computed with respect to its spring connections with its neighbors in addition of forces such as gravity and friction. The time-integration scheme can be, like explained in Section 2.2.4, either implicit or explicit, depending of the desired precision, stability, and performance of the mass-spring system. The springs are modeled as being elastic but sometimes a viscous force (damping) is added to produce energy dissipation during deformation. Therefore, the force \mathbf{f}_{ij} of a spring between two mass points i and j is governed by the following formula:

$$\mathbf{f}_{ij} = k_s \left((|\mathbf{x}_{ij}| - l_{ij}) \right) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} + k_d \left((\dot{\mathbf{x}}_j - \dot{\mathbf{x}}_i) \cdot \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} \right) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} \quad (2.14)$$

where \mathbf{x}_{ij} is the difference between the two mass position vectors ($\mathbf{x}_j - \mathbf{x}_i$), meaning $\frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|}$ is the direction of the spring. The term l_{ij} is the rest length of the spring while k_s is the spring stiffness and k_d the spring's damping constant. The modeling of the spring can obviously be modeled in more appropriate ways depending of the application and its underlying purposes. For instance, in the case of cloth simulations [BW98], the modeling has to reproduce physics behavior like bending and wrinkling. Besides, deformation energies can eventually be defined and minimized through soft constraints in order to improve the stability of the mass-spring system.

Finite Difference Method: The Finite Differences Method (FDM) [TPBF87] uses a regular spatial grid (mesh) in order to discretize the equation of motion. But the boundaries of the object are difficult to determine with such a rough representation of the object geometry. Moreover, local adaptations are only possible with irregular meshes.

Finite Volume Method: The Finite Volume Method (FVM) computes the nodal forces using the stress tensor in a direct way: Let's assume that the mesh is composed of planes, the internal forces per unit area with respect to a certain plane orientation is $\mathbf{f} = \boldsymbol{\sigma} \hat{\mathbf{n}}$ with $\hat{\mathbf{n}}$ the normalized vector of the plane. From this observation, we can say that the total force applied on the face S of a finite element is $\mathbf{f}_S = \int_S \boldsymbol{\sigma} dS$. Thus, if we use linear basis function, the stress tensor is constant within an element. For instance, if the elements of the mesh are planar faces, the previous equation can be simplified as $\mathbf{f}_S = S \boldsymbol{\sigma} \hat{\mathbf{n}}$ where S is the area of the face and $\hat{\mathbf{n}}$ its normal. To compute all the nodal force, we just have to iterate among all the faces S_i of the elements, and distribute the force \mathbf{f}_{S_i} among the nodes which are adjacent to that face.

Boundary Element Method: The Boundary Element Method (BEM) handles the elasticity of the object on the surface (boundary) instead of its volume. Thus, the volume integral is transformed into a surface integral by applying the Green-Gauss theorem, reducing the model to two dimensions. This complexity reduction allows to gain an important speed up in terms of computation. However, the approach is not compatible with non-linear distortions.

2.2.3 Mesh-free methods for spatial discretization

Spatially Coupled Particle Systems Particles have been used for decades to simulate fuzzy objects [Ree83] such as fire or clouds. They are also commonly used in fluid simulation through Smooth Particle Hydrodynamics (SPH) systems [YWH⁺09]. But particles can also be applied to simulate deformable objects. The principle of a particle system is the following: particles are actually points placed in a 3D space. They are generated to make a cloud of points which fits with a desired shape. Each particle has a certain number of attributes like position, velocity, lifetime, temperature. The values of these attributes can be either fixed or stochastically determined. Positions of the particles are updated according to their respective velocities. Often, a particle dies when it reaches its lifetime, and other particles can eventually spawn. In short, the attributes globally drive the dynamic behavior of the particles over time.

Particles interacting with each other are called Spatially Coupled Particle Systems. This type of representation presents some advantages: First, the simplicity of it. Secondly, it enables a huge number of particles in complex scenes. Thirdly, topological changes can be easily modeled with this method. The motion of the particles is determined from their energies. Each particle has its own potential energy which is related to the pairwise potential energies between itself and all the other particles within the system. The computation of the potential energy between two particles can be performed by the Lennard-Jones potential. Once the energies have been computed, the forces can be computed as the derivative of these energies. In contrast with the FEM, we notice that Spatially Coupled Particle Systems simulate the deformable object

on a microscopic scale. The resulting Lennard-Jones forces are indeed very small compared to the order of magnitude of the forces that we expect using FEM.

Point-based animation: Another technique is the combination with mesh-free physics and point sampled surfaces. It is called Point-based Animation [MKN⁺04]. In this approach, elastic body forces are derived via the strain energy density $\phi = \frac{1}{2}(\epsilon \cdot \sigma)$. The elastic force per unit volume at a particle p_i with material coordinates \mathbf{m}_i is computed using the directional derivative $\nabla_{\mathbf{u}_i}$. Combining with linear Hooke's law ($\sigma = \mathbf{L} \cdot \epsilon$ with \mathbf{L} constant), the elastic force is computed using the following formula:

$$\mathbf{f}_i = -\nabla_{\mathbf{u}_i} \phi = -\frac{1}{2} \nabla_{\mathbf{u}_i} (\epsilon_i \cdot \mathbf{L} \cdot \epsilon_i) = -\sigma \cdot \nabla_{\mathbf{u}_i} \epsilon_i. \quad (2.15)$$

Note that the approximation of $\nabla_{\mathbf{u}_i}$ from the displacement vectors \mathbf{u}_j of the neighboring particles is often computed using the Moving Least Square Method (linearization).

Shape matching: Shape Matching from Müller et al. [MHTG87] is a meshless method which, like mass-spring systems, represents the object as a cloud of mass points. However, these mass points correspond to the nodes of the volumetric mesh of the object, and there are no connections between the different nodes. The physics simulation performs like a simple particle system but without particle-particle integrations: after each time step, each particle is pulled towards its respective goal position. These goal positions are determined via a generalized shape matching of the undeformed rest state with the current deformed state of the point cloud.

2.2.4 Time integration schemes

As mentioned earlier, in the context of physics simulation, we are interested in numerical solutions knowing we have to discretize time and space. Most popular spatial discretization methods have been described above but these need to be combined with a time-integration scheme. In the following, we will consider that the domain D of our deformable object is constituted by a finite set of material particles. Let \mathbf{m} be one of these particles. It actually exists multiple time-stepping integration method but the simplest one is the Euler's method. Assuming that our initial value for \mathbf{x} is $\mathbf{x}_0 = \mathbf{x}(t_0) = \mathbf{x}(\mathbf{m}, t_0)$ at a time t_0 , we have to find \mathbf{x} at a later time $t_0 + h$ with h a fixed time-step. Euler's method simply computes $\mathbf{x}(t_0 + h)$ by taking a step in the derivative direction $\dot{\mathbf{x}}(t_0)$. It means the derivatives are numerically approximated with finite differences.

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h \dot{\mathbf{x}}(t) \quad \text{and} \quad \dot{\mathbf{x}}(t+h) = \dot{\mathbf{x}}(t) + h N(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) \quad (2.16)$$

The two above equations formulate the Euler's method using an explicit scheme, which is the simplest one. The latter provides explicit formulas for the quantities $\dot{\mathbf{x}}$ and \mathbf{x} at the next time step. An explicit scheme is simple to implement but it is stable only if h is smaller than a value (that depends on the ratio between mass and stiffness in the model and the size of the elements). Otherwise, the forces could get too large causing an exponential gain of energy and finally resulting in an explosion. To avoid this conditional stability, an implicit scheme can be used instead. This one uses quantities at the next time step on both sides of the equations. For instance, the implicit Euler scheme can be written:

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h \dot{\mathbf{x}}(t+h) \quad \text{and} \quad \dot{\mathbf{x}}(t+h) = \dot{\mathbf{x}}(t) + h N(\mathbf{x}(t+h), \dot{\mathbf{x}}(t+h), t) . \quad (2.17)$$

This improvement allows to use larger time-steps without a stability loss, and implies to know an expression of the derivable force. However, it induces to solve an algebraic system at each time step, which is more costly. For this reason, explicit scheme is often used in the case of computer games for which the performance is crucial. Besides, it is also possible to use an implicit scheme only for the evaluation of the position \mathbf{x} , whereas the velocity $\dot{\mathbf{x}}$ is computed in the first place using an explicit scheme:

$$\dot{\mathbf{x}}(t+h) = \dot{\mathbf{x}}(t) + h N(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) \quad \text{and} \quad \mathbf{x}(t+h) = \mathbf{x}(t) + h \dot{\mathbf{x}}(t+h) . \quad (2.18)$$

This mix between explicit and implicit has the advantage to be more stable than a pure explicit scheme, without any computational overhead. In Equation 2.16 related to Euler's method, we notice that the time derivatives are approximated as the first order Taylor series expansion between two consecutive times. From this observation, we can conclude that this approximation leads to an error of $O(h^2)$. To reduce this error, we could intuitively use a larger order Taylor serie expansion. For instance, using a second order expansion leads to an error of $O(h^3)$, and so on. In conclusion, all the time-stepping methods are actually based on the Taylor series.

In our case, we deal with simulations wherein collisions frequently occur, implying discontinuous phenomena. For this reason, using a time-integration scheme of higher order than 1 such as the middle-point method (order 2) or the Runge Kunta method (order 4) would be not suitable. Indeed, high order expansion of Taylor series allows to know the motion of the object along a larger time-step, which is only useful when this motion is not disturbed during that time-step. In addition, the more the order is high, the more the polynomial will be costly to compute, resulting in poor time performances.

2.2.5 Discussion

Spatial discretization method	Mesh-based or Mesh-free	Lagrangian or Eulerian	Advantages	Drawbacks
Finite Element Method (FEM)	Mesh-based	Lagrangian	Accurate method. Respect the law of conservation of matter.	Rather slow to compute.
Mass-Spring Systems	Mesh-based	Lagrangian	Efficient method when using explicit time-integration scheme.	Lack of accuracy.
Finite Difference Method (FDM)	Mesh-based	Lagrangian	Simple object representation: a regular spatial grid.	Object boundaries are difficult to determine with that kind of mesh.
Finite Volume Method (FVM)	Mesh-based	Lagrangian	-	-
Boundary Element Method (BEM)	Mesh-based	Lagrangian	-	Not compatible with non-linear distortions.
Spatially Coupled Particle Systems	Mesh-free	Eulerian	Simple object representation: particles. Easy topological changes.	Simulate at a microscopic scale which is not suitable in our case.
Point-based animation	Mesh-free	Lagrangian	-	-
Shape matching	Mesh-free	Lagrangian	No particle-particle integrations. Efficient method.	Lack of accuracy.

Table 2.1 – Comparison between the state-of-the-art spatial discretization methods.

Choosing the most suitable spatial discretization method (see Table 2.1) in regards to the desired application lays on numerous criteria knowing that the most important ones are probably accuracy, stability, and efficiency (time-performances). The time-performances factor is crucial in computer games and virtual reality applications, in order to guarantee a smooth experience in despite of accuracy. That is why efficient methods like mass-spring systems are commonly used in these cases, in combination with an explicit time-integration scheme (fast to compute). In contrast, virtual simulations of soft robots, as well as surgery applications, requires to be precise enough to be significant, and could be more tolerant about slow

simulations. From this observation, Finite Element Method is generally used in this situation, and with an implicit time-integration scheme in order to guarantee a better stability. Besides, the time performances of the simulations are also impacted by the selected collision detection methods (see Section 2.3) as well as the selected contact handling technique (see Section 2.4). Nowadays, the most advanced implementations of physics simulations actually boost the time performances by taking advantage of the GPU while using acceleration methods like the Reduced Deformations Models or the Adaptive Models that we explained earlier.

2.3 Collision detection

In a virtual scene using FEM, when a lot of objects may collide each other, the naive approach is to test each element-element collision in an exhaustive way. Unfortunately, it is a very inefficient approach because the complexity is equals to $O(n^2)$ in this way, knowing n the total number of elements. To overpass this issue, collision detection is often performed through a pipeline containing several stages corresponding to different levels of granularities. The pipeline is typically constituted of three stages:

- Broad-phase: the bodies are approximated by simple geometric primitives, such as the sphere for example, for which the computation of the euclidean distances between the primitives is very fast. This step allows to keep only the bodies which are roughly close enough and put all the other bodies aside.
- Mid-phase: the remaining bodies are split in several parts in the case they are complex shapes. During this phase, some of these parts are culled in local body space.
- Narrow-phase: the parts of the remaining bodies are used in order to find the precise body feature in contact and the location of the contact points.

By structuring the collision pipeline as such, we prune a lot of collision tests progressively through the pipeline, improving time performances. In addition, collision detection stages often carry spatial data structures to speed up the simulation. These structures are built in a preprocessing stage in the case of rigid bodies whereas they need to be updated frequently in the context of soft body simulation. Most popular spatial data structures are Bounding-Volume Hierarchy (Section 2.3.2) and Distance Fields (Section 2.3.3). In addition to time performances, an other important aspect with collision detection is the formatting of the output data. Indeed, collision detection returns either penetration depth, or a minimal distance between object meshes, or interpenetration volumes, or pairs of primitive. Also, it is important to mention that, contrary to rigid body, self-collision is an aspect which is necessary to handle for realistic behavior in the context of simulation of deformable objects.

2.3.1 Spatial subdivision

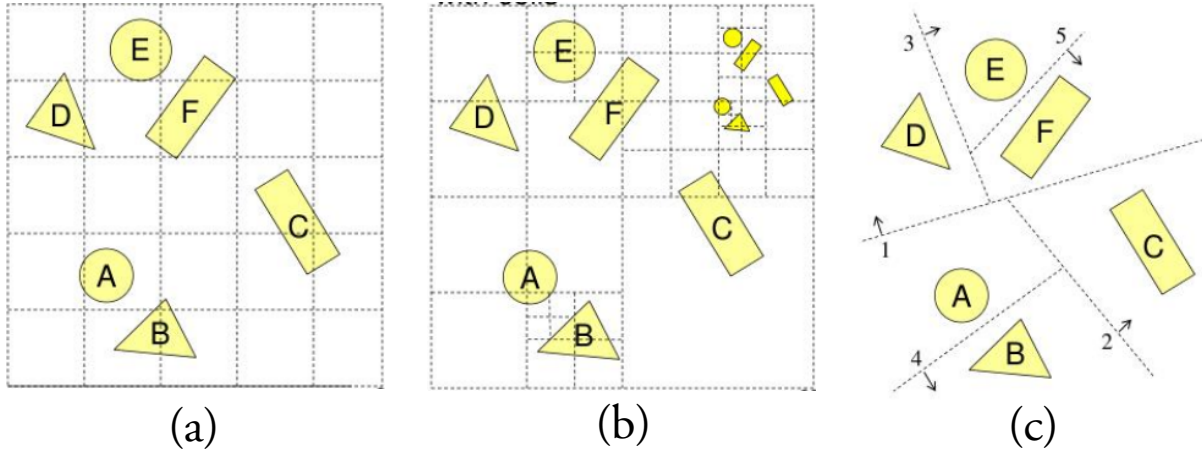


Figure 2.5 – Different spatial structures, here represented in 2D: (a) uniform grid, (b) kD-tree, (c) BSP-tree.

One aspect when dealing with collision detection of deformable objects is to partition the space. The simulated objects are consequently wrapped around spatial structures. These can be uniform grids, kD-trees, or BSP-trees as shown in Figure 2.5.

The uniform grids are the simplest structures. They partition the space uniformly into axis-aligned cells. But hashed storage can however be applied for non-uniform distribution.

An octree [Sam88] is a hierarchical structure (tree) which partition the space into rectangular and axis-aligned cells. In fact, an octree corresponds to a kD-tree with $k = 3$ where k is the number of dimensions of the space wherein the simulation takes place. The octree's root corresponds to the Axis-Aligned Bounding Box (AABB) of the objects while its internal nodes correspond to subdivisions of the AABB. Each node can own until 8 children, meaning the space is recursively subdivided in 8 octants. Octrees take less memory than uniform 3D grids but more than BSP-trees. The subdivision of space can be uniform or non-uniform when using octrees. Moreover, the hierarchy can be dynamically updated.

BSP means Binary Space Partitioning [TN87], and BSP-trees are structures which recursively partition the space in convex zones and in a hierarchical way. It means that each node of the tree is associated to a subspace of its parent node. In other words, the space is divided in two subspaces in order to obtain the same number of polygons inside each subspace. And then, each one of these subspaces is also subdivided

in two subspaces and so on, until there is no any possible division. One interesting observation is: if all the subspaces are axis-aligned, then the BSP-tree is actually a kD-tree.

2.3.2 Bounding volume hierarchies

Bounding-Volume hierarchies (BVH) are efficient data structures for collision detection. Whereas they are commonly used for rigid body collision detection, they are especially suitable for soft bodies because they can be employed to accelerate self-collisions. Often, the 3D objects of the scene are made up using a set of homogeneous primitives like polygons, tetrahedrons or sometimes NURBS patches. Knowing that, the principle of BVH is, at a scale of an object, to partition these primitives recursively until some leaf criterion is encountered. It leads to the construction of a hierarchy wherein each node is associated with a subset of primitives and a Bounding Volume (BV) which encloses this subset. The hierarchies are commonly binary trees in cases of rigid objects, meaning one child by node. Instead, the number of children by node is typically 4 or 8 for deformable objects. The reason is that that fewer nodes need to be updated, improving the time performances of the update step.

As illustrated in Figure 2.6, it exists different classes of Bounding Volumes (BV): OBBs, DOPs, AABBs, spherical shells, convex hulls to name just a few. Among those, the most interesting ones are Oriented Bounding Boxes (OBB) and Discrete Orientation Polytope (k-DOP). In the one hand, an OBB wraps primitives into a rectangular bounding box at an arbitrary orientation in the 3D space. The main advantage of OBBs is they can bound geometry more tightly than Axis-Aligned Oriented Boxes (AABB) and spheres, thanks to their variable orientation. In the other hand, a k-DOP is a convex hull whose precision is determined by a chosen k value, which is very convenient.

The BVHs are traversed top-down in order to test overlap between two colliding objects. During the traversal, there are three possible cases to manage: If the overlapping nodes are both leaves then the enclosed primitives are directly tested for intersection. In the case of one is a leaf but the other is an internal node, then the leaf is tested against each of the children of the internal node. The last case is when the overlapping nodes are both internal nodes. In this situation, it is tried to minimize the probability of intersection as fast as possible.

The BVHs can be built following one of the three different approaches: top-down, bottom-up, and insertion. The most used strategy is the top-down strategy. In this method, the set of the object primitives are split until a threshold is reached. An heuristic guides the splitting in order to yield an appropriate BVH regardless to the chosen criterion.

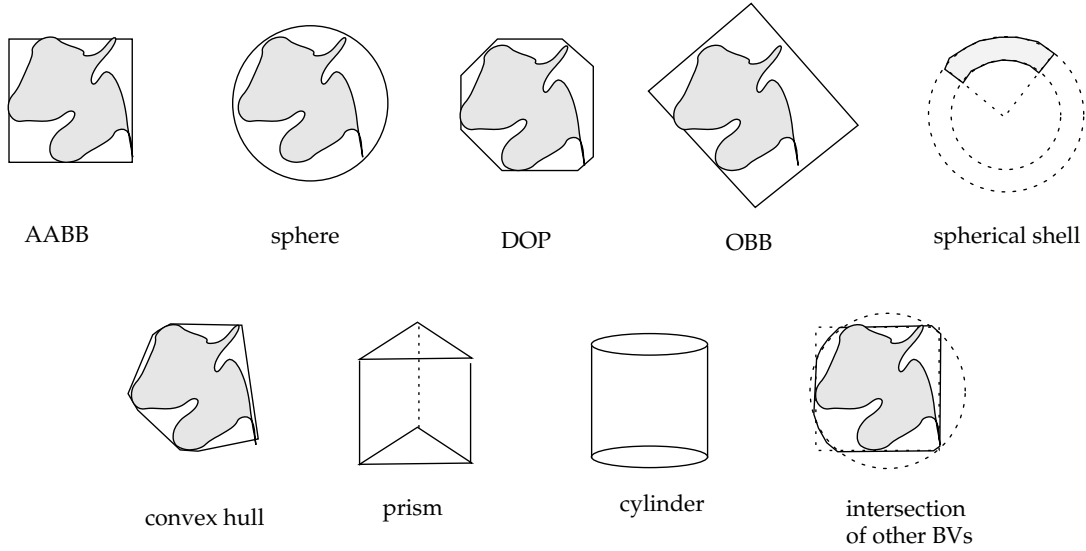


Figure 2.6 – It exists numerous types of Bounding Volumes [TKH⁺05].

With deformable objects, the BVHs have to be updated or refitted each time a deformation happens. Therefore, these operations need to be efficient. Usually, during the simulation, the structure of the tree is kept and only the extents of the BVs are updated.

2.3.3 Distance fields

Distance fields (or distance volumes), illustrated in Figure 2.7, specify the minimum distance to a closed surface (an object in the scene) for all points in the field. From this observation, the distance may be signed to distinguish between inside (positive value) and outside (negative value), which is very convenient. Besides, distance fields works with any kind of object topology. One of the other advantages is that the collision detection response is very fast and does not depend of the geometric complexity of the object. It is also important to notice that in this approach, collision detection between deformable and rigid object can be performed efficiently without updating the distance fields. Before collision detection, the distance fields must be generated first, and this step is unfortunately very slow in practice. The distance field can be represented through a variety of different data structures. The most popular are uniform 3D grids, BSP-trees and octrees we explained in Section 2.3.1.

An uniform 3D grid is the most easy way to implement distance fields: Distances are computed for each grid point and intermediate values are obtained using trilinear interpolation. Distance queries can be computed in constant time, which is a very interesting property for interactive applications. Most collision response schemes require normals computation which can be performed by normalizing the analytical

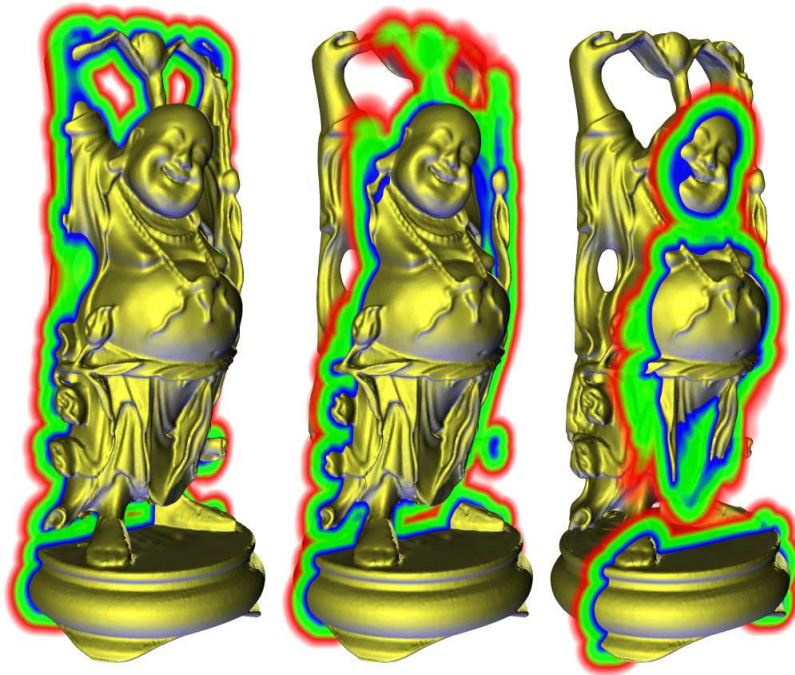


Figure 2.7 – Happy Buddha [TKH⁺05] and three color-mapped slices coming of its distance field. Blue indicates close distances while red indicates medium distances.

gradient of the trilinear interpolation. Although the resolution of the grid can be adapted, it is often huge and therefore requires a large memory capability.

The advantage of using a BSP-tree is mainly that the memory consumption can be reduced using a piece-wise linear approximation of the distance field, which is not necessarily continuous. Furthermore, several algorithms exist for selecting appropriate splitting planes in order to partition the space in a very compact way. Unfortunately, the construction of a BSP-tree is costly.

Friskén and al. [FPRJ00] proposed Adaptively sampled Distance Fields (ADF) stored in a hierarchy, typically an octree. In this approach, the data is able to increase the sampling rate in regions of fine detail. Compared to uniform 3D grids, an ADF provides a good compression ratio.

The computation of the distance field's Euclidean distances to the surface is generally a simple evaluation function. For example, in a case of an object mesh made up of triangles, finding the minimum distance between a given point and a triangle can be done efficiently using Voronoi [SOM04] regions of the features of the triangle. Moreover, this method can use the GPU capabilities to speed up computation.

2.3.4 Stochastic methods

Another approach to detect collisions efficiently is to use stochastic methods. Although they are inexact methods, these methods are motivated by the fact that the geometry of the 3D objects are already approximated when using polygonal models. A first approach from Klein and al. [KZ03] is to estimate the possibility of a collision in respect to a quality criterion chosen by the user. This model is convenient for time-critical systems, because it is possible to assess the quality according to a chosen maximum time-budget. The author's framework uses extended BVH called average-distribution trees or ADB-trees which additionally stores characteristics about the average distribution of the set of polygons in the scene. Another technique consists in performing a stochastic sampling within colliding bodies in order to guess colliding pairs.

2.3.5 Image-space techniques

Image-space techniques are efficient ways to calculate collision without any processing and by using GPU capabilities. An early approach was from Shinya et al. [SF91] using rasterization: the projection of the target objects are rasterized in a depth buffer, allowing the detection of overlapping objects. Faure et al [FAFB08] extended the method using Layer Depth Images (LDI) to process contacts between objects bounded by triangular surfaces. Unlike previous techniques of the same category, this one relies on image-based volume minimization. It has the advantage to eliminate complex geometrical computations and robustly handles deep overlap, which eventually occurs between the unconstrained configurations of two colliding objects.

2.4 Contact handling and constraints

In the context of our work, we seek to simulate the suction phenomenon which typically implies an interaction between a deformable object and another object which may be deformable too. It means our simulation has to manage object collisions by handling contact forces as the third law of Newton states, in order to avoid object-object interpenetration. But additional phenomena can be taken into account, like friction, depending of the goal of the simulation and the desired realism. In fact, the principle of contact handling consists in producing a collision response according to the output of the collision detection algorithm. Most of the time, at the scale of a time step, the simulation firstly computes the new material configurations of the deformable objects without considering contacts (unconstrained motion). These configurations are commonly called free-motion configurations (or unconstrained configurations) and can overlap each other. The simulation is consequently able to detect collisions regarding to the overlap between two interpenetrating free-motion configurations. Once collision detection is finished, the stage regarding

the contact handling typically occurs. In our case, we introduced the suction phenomenon into the contact handling stage. Several approaches exist for contact handling. They can be classified into three categories:

- Penalty-based: spring-damper forces are used to separate objects at contact (Section 2.4.3).
- Impulse-based: iterative solver which solves multiple contacts as a sequence of individual collisions (Section 2.4.4).
- Constraint-based: the contact is designed as a complementarity problem (Section 2.4.5).

Among these three categories of approaches, we are particularly interested by the Constraint-based methods in our context because of their accuracy (see Section 2.4.5). In the following, we explain the main laws and concepts related to contact handling. Then, we give an overview of the state of the art of the contact handling techniques in each category. For a more complete survey, the reader can refer to the one from Benedetti [Ben02].

2.4.1 Signorini's law

In continuum mechanics, the Signorini problem is known to deal with contacts between deformable bodies. It consists in finding the elastic equilibrium configuration of an anisotropic non-homogeneous elastic body, resting on a rigid friction-less surface and subject only to its mass forces. Moreau [Mor66] introduced the use of quadratic programming (QP) to solve it. QP is a particular type of nonlinear programming which is intended to optimize a quadratic function of several variables subject to a set of linear constraints. On a surface ξ , the contact forces are an integration of elementary surface forces df_s that exert on the elementary surfaces ds (triangle for instance) are defined as:

$$df_s = -\sigma \cdot \hat{n} \cdot ds = -\sigma_s \cdot ds$$

where \hat{n} is the normal vector of the elementary surfaces ds and σ the stress tensor of the deformable object. Thanks to QP, the Signorini's formulation becomes a Complementarity Problem (CP):

$$0 \leq \delta_n(P) \perp \sigma_n(P) \geq 0$$

$$\sigma_t(P) = 0$$

where \perp represents a complementarity between the two constraints $0 \leq \delta_n(P)$ and $\sigma_n(P) \geq 0$ (see Section 2.4.5). P is a point of the object surface ξ and δ_n represents the minimal distance between every point P and its near surrounding. σ_n and σ_t are respectively the normal stress and the shearing stress which both

belong to the surface stress tensor σ_s . These terms are expressed as follows:

$$\sigma_n = \sigma_s \cdot \hat{n} \quad \text{and} \quad \sigma_t = \sigma_s \cdot \hat{t}$$

where \hat{n} and \hat{t} are respectively the normal vector and the tangent vector of the elementary surface ds which carries the point. The Signorini's formulation considers two states: Either the point P is actually a contact point and in this case $\delta_n(P) = 0$ and $\sigma_n(P) > 0$. Otherwise P is not yet a contact point which means that $\delta_n(P) > 0$ and $\sigma_n(P) = 0$. Further details about the formulation of contact phenomena using complementarity constraints are given in Section 2.4.5.

2.4.2 Coulomb friction model

Friction is often simulated using the Coulomb's law, also called the Coulomb friction cone (Figure 2.8). This friction model is an isotropic model which describes friction in a simplified way. This law distinguishes two cases: When the solids slide one on the other (slip state or dynamic friction) and when there is an adhesion between both solids (stick state or static friction). In both cases, we assume that each mutual action exerted between the two solids contains a normal vector \mathbf{f}_n which press them one against other, and a tangent vector \mathbf{f}_t which is an opposing force to slipping. The two solids are in a static friction state while $\mathbf{f}_t < \mathbf{f}_t^0$, knowing that \mathbf{f}_t^0 is a limit force defined as $\mathbf{f}_t^0 = \mu_0 \times \mathbf{f}_n$. In this formula, μ_0 is the static friction coefficient which depends of the material properties and the condition of solid surfaces. In a geometric point of view, we are in a stick state while the contact force $\mathbf{f}_c = \mathbf{f}_t + \mathbf{f}_n$ stays inside a cone, called friction cone (Figure 2.8). The aperture of this cone is twice the angle $\phi_0 = \arctan(\mu_0)$. In the contrary, we are in a slip state when $\mathbf{f}_t^0 \geq \mathbf{f}_t$ or in other words, when \mathbf{f}_c is outside of the cone. In this case, $\mathbf{f}_t = \mu \times \mathbf{f}_n$ with μ the dynamic friction coefficient. Leine et al. [LG03] extended the Coulomb model to also take into account a normal friction torque (drilling friction) and spin. Their work was inspired by the work from Contensou whose conducted drilling friction experiments and established the dependence of the sliding friction force on the sliding velocity and spin. For this reason, the extended model is called Coulomb–Contensou friction.

2.4.3 Penalty-based models

One approach to resolve detected collisions consists in applying penalty forces [TMOT12] when two objects collide: the collision detection stage compute the interpenetration metric from the respective free-motion configurations. After that, temporary springs are attached between the contact points. Each spring compresses and applies equal and opposite forces to separate the two bodies. These forces are the penalty contact forces. They increase with the interpenetration. It means the penalty forces acts as a repulsion

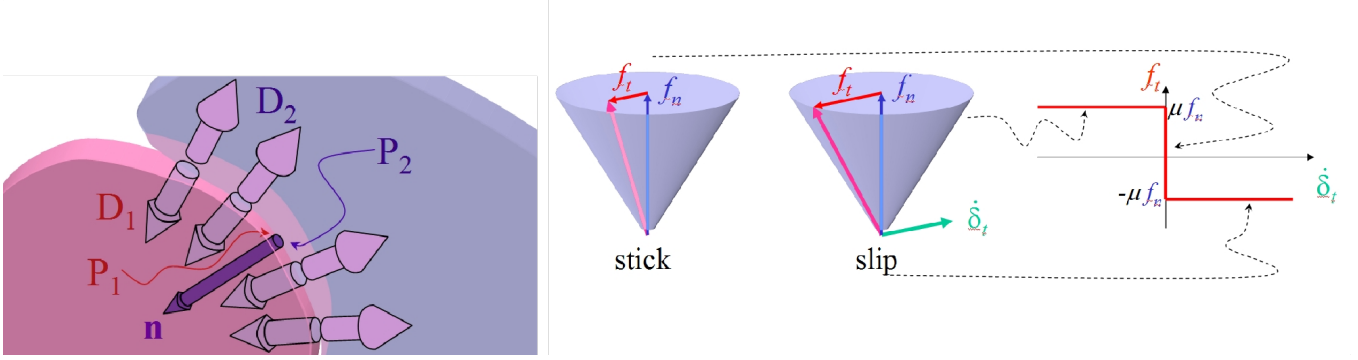


Figure 2.8 – Illustration of the Coulomb friction cone [Dur13].

between the two free-motion configurations to prevent the interpenetration. Mathematically speaking, a penalty force \mathbf{b} is often formulated as it follows:

$$\mathbf{b} = -(k_s(\|\mathbf{s}\| - l) + k_d \|\dot{\mathbf{x}}h\|) \frac{\mathbf{s}}{\|\mathbf{s}\|} \quad (2.19)$$

with k_s the stiffness of the spring and k_d the damping of the spring. The vector \mathbf{s} is the difference vector between the two contact points and l the initial length of the spring. The penalty method is simple particularly when combined with an explicit time-integration scheme. Consequently, it requires small integration time steps to guarantee stability when dealing with large penalty contact forces.

2.4.4 Impulse-based models

The idea of the impulse-based approach is to model all contacts between objects through a series of impulses [Pro97]. The latter aim to avoid interpenetration between colliding objects by correcting their trajectories. An impulse \mathbf{g} is the product of the force \mathbf{b} and the time step h such as $\mathbf{g} = \mathbf{b}h$ in Newtons per second. The appliance of the impulse instantaneously change the velocity of the two colliding objects. The change of velocity $\Delta \mathbf{v}$ is equals to: $\Delta \mathbf{v} = \frac{\mathbf{g}}{m}$ with m the mass. Impulse-based approaches work as it follows: When a collision is detected, meaning the free-motion configurations of two object overlap each other, the collision impulse \mathbf{g} is then computed:

$$\mathbf{g} = \mathbf{M}^{-1} \Delta \mathbf{u} \quad (2.20)$$

with $\Delta \mathbf{u}$ the change in the contact point velocity over the course of the collision, and \mathbf{M} a matrix dependent only upon the masses and mass matrices of the colliding bodies. The impulse \mathbf{g} is instantaneously applied on the first object while $-\mathbf{g}$ is applied on the second one. The advantage of the impulse-based approach in contrast with penalty-based is its ability to estimate the exact time of every impact event. Contrary to

the penalty-based approach, the impulse-based approach usually guarantees a collision-free state of the colliding objects. The major disadvantage of this method is its inability to efficiently handle simultaneous and persistent contacts.

2.4.5 Constraint-based models

The contacts between deformable objects can be handled using constraints [OTSG09]. It leads to reformulate the second law of Newton (Equation 2.6) into:

$$M(\mathbf{x})\ddot{\mathbf{x}} = \mathbf{g}(t) - \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{H}^T \boldsymbol{\lambda} \quad (2.21)$$

with $\mathbf{g}(t)$ the external forces, $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$ the elastic forces, \mathbf{H} the constraint matrix which embed the directions of the constraint forces and $\boldsymbol{\lambda}$ the intensities of these constraint forces. Generally speaking, a constraint can be either bilateral (equality) or unilateral (inequality). Among the constraints, some of them could have complementarities each other, such as contact constraints with friction constraints when using the Coulomb friction cone (Section 2.4.2). It consequently leads to formulate a Complementarity Problem (CP) in the form of an algebraic system. Complementarity between two constraints c_1 and c_2 is commonly noted as $c_1 \perp c_2$ such that $c_1 \cdot c_2 = 0$, meaning they are orthogonal. The CP can be either a Non-Linear Complementarity Problem (NCP), a Linear Complementarity Problem (LCP), a Mixed Non-Linear Complementarity Problem (MNCP) or a Mixed Linear Complementarity Problem (MLCP). A mixed complementarity problem contains inequalities, equalities and unrestricted variables, whereas non-mixed include only inequalities. Mathematically speaking, these four categories of CP can be illustrated using the following examples:

NCP	$0 \leq f(\mathbf{x}) \perp \mathbf{x} \geq 0$	$f(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a non-linear function. $\mathbf{x} \in \mathbb{R}^m$ is the seeking vector.
LCP	$0 \leq y(\mathbf{x}) \perp \mathbf{x} \geq 0$	$y(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a linear function such that $y(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{b} \in \mathbb{R}^m$ are known.
MNCP	$0 \leq f(\mathbf{x}, \mathbf{w}) \perp \mathbf{x} \geq 0$ $g(\mathbf{x}, \mathbf{w}) = 0$	$\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^n$ are the seeking vectors. $f(\mathbf{x}, \mathbf{w}) : \mathbb{R}^{m+n} \mapsto \mathbb{R}^m$ and $g(\mathbf{x}, \mathbf{w}) : \mathbb{R}^{m+n} \mapsto \mathbb{R}^n$ are both non-linear functions.
MLCP	$0 \leq y(\mathbf{x}, \mathbf{w}) \perp \mathbf{x} \geq 0$ $z(\mathbf{x}, \mathbf{w}) = 0$	$y(\mathbf{x}, \mathbf{w}) : \mathbb{R}^{m+n} \mapsto \mathbb{R}^m$ and $z(\mathbf{x}, \mathbf{w}) : \mathbb{R}^{m+n} \mapsto \mathbb{R}^n$ are both linear functions. $y(\mathbf{x}, \mathbf{w}) = \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{w} + \mathbf{a}$ with known values for $\mathbf{F} \in \mathbb{R}^{m \times m}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\mathbf{a} \in \mathbb{R}^m$ $z(\mathbf{x}, \mathbf{w}) = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{w} + \mathbf{r}$ with known values for $\mathbf{C} \in \mathbb{R}^{n \times m}$, $\mathbf{D} \in \mathbb{R}^{n \times n}$, $\mathbf{r} \in \mathbb{R}^n$.

Most often, the constraints we include into the system are unilateral contact constraints and non-linear friction constraints. These are respectively those coming from the Signorini's law (Section 2.4.1) and the Coulomb friction cone (Section 2.4.2). Knowing Coulomb's friction involves nonlinear constraints, we deal with an inconvenient NCP from this observation. A popular approach is to simplify the friction phenomenon

by using a pyramid discretization of the friction cone in order to transform the current NCP into a LCP. Note that the number of sides of the pyramid must be carefully chosen in respect to the desired precision.

Lagrange multipliers: One method to solve the constraint system is to beforehand transform it into an optimization problem without constraints using Lagrange multipliers. In Equation 2.21, the constraint matrix \mathbf{H} can be considered as the Jacobian of the mapping between the physics space and the constraint space while $\boldsymbol{\lambda}$ actually represent the vector of Lagrange multipliers. Considering a single linearization by time step, Equation 2.6 can be reformulated as the following linear system:

$$\mathbf{M}(\mathbf{x})\Delta\dot{\mathbf{x}} = -h(\mathbf{g}(t) + \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) - \mathbf{H}^T \boldsymbol{\lambda})$$

that we can write in a simpler way as:

$$\mathbf{A}\Delta\dot{\mathbf{x}} = \mathbf{b} + h\mathbf{H}^T \boldsymbol{\lambda}$$

In case of two interacting objects, it actually leads to two Equations (one per object):

$$\mathbf{A}_1\Delta\dot{\mathbf{x}}_1 = \mathbf{b}_1 + h\mathbf{H}_1^T \boldsymbol{\lambda} \text{ and } \mathbf{A}_2\Delta\dot{\mathbf{x}}_2 = \mathbf{b}_2 + h\mathbf{H}_2^T \boldsymbol{\lambda} .$$

This system can then be solved in two steps: We first compute the free-motion configurations $\Delta\dot{\mathbf{x}}_1^{free}$ and $\Delta\dot{\mathbf{x}}_2^{free}$ for each object by setting $\boldsymbol{\lambda} = 0$. Secondly, we found the vector of Lagrange multipliers $\boldsymbol{\lambda}$ which counteract the vector of constraint violations $\dot{\boldsymbol{\delta}}$. The latters are evaluated in terms of velocity from the collision detection stage and expressed relatively to the Lagrange multipliers as:

$$\dot{\boldsymbol{\delta}} = \mathbf{H}_1\Delta\dot{\mathbf{x}}_1 - \mathbf{H}_2\Delta\dot{\mathbf{x}}_2 + \mathbf{W}\boldsymbol{\lambda}$$

where \mathbf{W} is the Schur complement of \mathbf{A} and defined as:

$$\mathbf{W} = h(\mathbf{H}_1\mathbf{A}_1^{-1}\mathbf{H}_1^T + \mathbf{H}_2\mathbf{A}_2^{-1}\mathbf{H}_2^T)$$

Once the Lagrange multipliers have been found thanks to a Gauss-Seidel solver for instance, the motion corrections of the two interacting objects are applied onto their respective free-motion configurations.

Volume constraints: Contact constraints can be represented as points and solved according to the penetration distances. But to be accurate, the number of points must be important, often resulting to a large LCP, expensive to compute. Another approach is to use volume constraints. In this case, the penetration volumes have to be solved. It results to a MCP but which can be solved as a LCP using a compliance

matrix. The number of volumes depends of the size of a regular grid over the space. In most cases, the number of volume constraints is much less than point constraints, which consequently reduces the size of the LCP, which becomes more efficient to solve. Allard et al. [AFC⁺10] introduced a new method for simulating frictional contact between volumetric objects using interpenetration volume constraints. Contact between highly detailed meshes can be simplified to a single unilateral constraint equation, or accurately processed at arbitrary geometry-independent resolution with simultaneous sticking and sliding across contact patches. The intersection volumes and gradients are efficiently provided thanks to a GPU implementation of the Layered Depth Image (LDI) collision detection method from Heidelberger [HTG03]. Talvas et al. [TMDO15] designed a physics model for virtual dexterous manipulation with soft fingers using a constraint-based formulation. The method takes advantage of volume constraints to reduce the size of the complementarity problem and consequently speeds up the simulation. The authors actually generate aggregate volume constraints from multiple classical contact constraints, hence the word aggregation.

2.5 Linear solvers

We explained in Section 2.2.4 the importance of choosing either an implicit or explicit time integration scheme. To summarize, an implicit scheme allows the use of larger time steps while guaranteeing a better stability compared to an explicit scheme. However, an explicit scheme is very fast to solve contrary to an implicit scheme, and is more suitable to efficiently solve non-stiff problems. In the end, using an explicit or implicit scheme impacts on the construction of a complete linear system $\mathbf{Ax} = \mathbf{b}$ arising from the dynamics of a given deformable object. More precisely, explicit contributions at the current time step $\mathbf{x}(t)$ will contribute to the right hand side vector \mathbf{b} , while implicit contributions at the next time step $\mathbf{x}(t+h)$ will contribute to the left hand side matrix \mathbf{A} . Besides, the combination of explicit and implicit methods is also possible, it is called semi-implicit or semi-explicit schemes. Eventually, the problem can otherwise be formulated as a non-linear system to simulate non-linear deformations. Besides, assuming the constraints such as contact constraints and friction constraints have been gathered into a LCP, the latter may have been transformed into a linear system without constraints thanks to the Lagrange multipliers method.

Once the linear system $\mathbf{Ax} = \mathbf{b}$ has been built, the latter then needs to be solved in order to find the material configuration of the object at the next time step. The solving process can be performed using either a direct linear solver or an iterative linear solver. On the one hand, direct solvers find the exact solution by computing $\mathbf{A}^{-1}\mathbf{b}$ in one single step. They are suitable in the context of well-conditioned and even some quite ill-conditioned problems. On the other hand, an iterative solver gradually converges towards the solution: it firstly finds an approximated solution which is then refined through multiple next iterations. The

convergence of an iterative solver remains monotonic for well-conditioned problems and might be slower for ill-conditioned systems. The iterative solver computes the residual $r = \mathbf{A}\mathbf{x} - \mathbf{b}$ at each iteration. It continues to iterate until either the current solution is satisfactory (the residual r is below a given tolerance) or the number of executed iterations overreaches a fixed maximum value. In other words, the formulation becomes an optimization problem wherein the goal is to minimize the residual r . Assuming that \mathbf{A} is a positive definite matrix, the optimization function (evaluation function of the residue) becomes:

$$r(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} \quad (2.22)$$

One popular iterative solver for linear matrix problems is the conjugate gradient solver. As a matter of fact, it exists a wide literature about linear solvers, especially when the \mathbf{A} matrix is sparse, which is our case.

In our context, we decided to use a conjugate gradient solver combined with an asynchronous preconditioner based on *LDL* decomposition to compute the free-motion configurations of the simulated objects. *LDL* decomposition means the system matrix \mathbf{A} is decomposed through the form $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ with \mathbf{L} a lower unit triangular matrix and \mathbf{D} a diagonal matrix. This technique allows to precondition the system meaning that it can be solved efficiently.

2.6 Conclusion

In conclusion of this chapter, we have drafted the state of the art of the simulation of deformable objects and contact handling. In our context, we chose to use the Co-rotational Finite Element Method (with an implicit time scheme) because it is an interesting trade-off between accuracy and time-performance. In order to compute the free-motion configuration of the simulated objects, we decided to use a Projected Conjugate Gradient solver combined with an asynchronous preconditioner which takes advantage of *LDL* decomposition. Besides, in terms of collision detection, knowing our simulations include only a few objects, our broad-phase is a simple brute force collision detection with simple Axis-Aligned Bounding Boxes (AABB) wrapped around the simulated objects. And our narrow-phase collision detection use virtual cones for which their tips are positioned along the vertices of the collision meshes. When a vertex from a collision mesh (external to the current object) enters into the cone, it means one precise portion of contact area is close. Furthermore, we use a constraint formulation to handle frictional contact between colliding objects. It leads to a Non-linear Complementarity Problem (NCP) which is built using Lagrange multipliers, and we use the iterative Projected Gauss-Seidel (PGS) solver to find their values. In the next chapter, we will see how we add new constraints into this NCP formulation in order to simulate the suction behavior.

MODELING & ALGORITHMS

Contents

3.1	Description of the suction phenomenon	50
3.2	Geometric cavity detection	52
3.2.1	Contact surface determination	53
3.2.2	Flood-fill of inner and outer surfaces	54
3.2.3	Inner border refinement	55
3.2.4	Pairing cavity inner surfaces	56
3.2.5	Tracking cavities over time	57
3.2.6	Volume computation	57
3.2.7	Dynamic topology	58
3.3	Numerical methods	59
3.3.1	Simulation pipeline	59
3.3.2	Corotational Finite Element Method	60
3.3.3	Constraint formulation	61
3.3.4	Constraint resolution	63
3.4	Conclusion	66

The suction phenomenon implies the consideration of multiple physical aspects such as the dynamics of the air (fluid) as well as the dynamics of the deformable materials while considering contact and friction phenomena. Our choices in terms of modeling were driven by the most relevant features we need for an accurate model without overloading the complexity and time performances.

The purpose of this chapter is to explain the suction phenomenon while giving details about our scientific contributions from a modeling and algorithmic point of view. In the first part of this chapter, we provide an illustrated explanation of how the suction phenomenon works through an example involving a suction cup (Section 3.1). Knowing the cavity of the suction cup must be sealed to produce the suction effect,

we propose a cavity detection algorithm that we describe in Section 3.2. After that, we clarify the chosen numerical methods and we present the mathematical formulation of our novel air pressure constraint in Section 3.3.3.

3.1 Description of the suction phenomenon

In the context of our research, we distinguished two types of suction: active suction and passive suction. Active suction is the simplest case in terms of modeling. It assumes that the air pressure inside the cavity of the suction cup is directly controlled by an external device such as a vacuum pump which would be connected to the cavity. In this case, the time-variation of the pressure inside the cavity is known. On the contrary to active suction, the cavity is not linked to any external device in the passive suction case. Consequently, the time-variation of the pressure inside the cavity is driven by the ideal gas law and the suction cup undergoes deformation in function of the elasticity and friction phenomena.

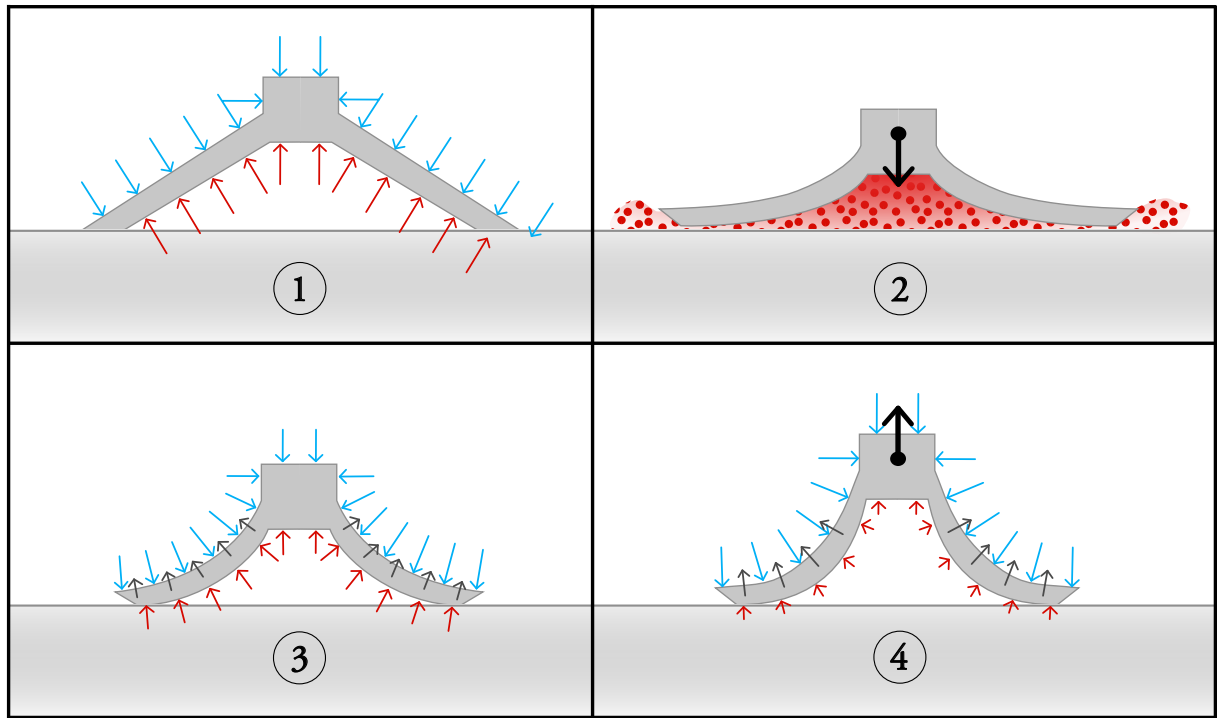


Figure 3.1 – Illustration of the suction phenomenon through a scenario in the context of passive suction: (1) The suction cup is positioned on the floor. The red arrows represent the internal pressure while the blue ones represent the atmospheric pressure. Both pressure values are the same at this moment. (2) We push the cup onto the ground. Some air escapes. (3) The cup tends to an equilibrium state. The atmospheric pressure is stronger than the internal pressure. The dark grey arrows represent the elastic forces. (4) We pull the suction cup thus increasing the volume of the cavity and decreasing the internal pressure accordingly.

In a nutshell, the suction phenomenon occurs when there is a difference in terms of air pressurization between two neighboring air volumes. The explanation of the suction phenomenon is sketched in Figure 3.1 as a series of four states along a simple scenario in which we push a traditional suction cup against a flat ground surface and then pull it. Note that we deal with passive suction in this scenario but the same principles apply in the context of active suction except that the air pressure value is already known (because it is controlled by the user). We detail the four states of the scenario below in order to give a better understanding of how passive suction works:

1. We simply put the suction cup onto the floor. At this state, it results in an airtight cavity. In other words, a volume of air is trapped inside the cavity. The air pressure has approximately the same air pressure intensity inside as outside the cavity at this time. The relation between the air pressure p , the volume v , and the air quantity n inside the cavity relies on the ideal gas law $p v = n R T$ with R the ideal gas law constant and T the temperature. Assuming that T is constant, the right-hand term of this equation is likewise constant as long as the cavity is airtight.
2. We then push the suction cup against the ground. The volume of the cavity progressively decreases, the gas particles inside the cavity are compressed which consequently create an over-pressurization. The more the gas particles are compressed, the more the air pressure inside the cavity is strong. But the internal pressure quickly becomes strong enough to slightly leverage the rim of the cup and let some gas particles escape, meaning the cavity is suddenly not airtight anymore. Due to the fact that the number of gas particles has decreased and in respect of the ideal gas law, the air pressure decreases too.
3. We stop pushing against the suction cup. No air escape anymore which means the air-tightness of the cavity is restored. We assume $p v = \text{const}$. At this moment, we know we have a pressure difference between the air pressure and the atmospheric pressure: we just reached a state in which the suction phenomenon occurs. More precisely, the atmospheric pressure is stronger than the air pressure inside cavity. In this situation, the force field of the atmospheric pressure pushes the walls of the suction cups towards the inside. But in contrast, the elastic forces of the material try to restore the rest configuration of the object. It progressively results in an equilibrium wherein elastic forces and the internal pressure confront the atmospheric pressure. With regards to $p v = \text{const}$, the elastic forces tend to increase the cavity volume while decreasing the internal air pressure. This behavior actually reinforces the effect of the atmospheric pressure which keeps the suction cup stuck to the ground.

4. We now pull the suction cup. The cavity is still airtight at this state. The pulling increases the cavity volume and reinforces more intensively the effect of the atmospheric pressure. That is why the more we pull onto the cup, the more it becomes difficult to go further. But in the case the pulling force becomes strong enough, the suction cup finally detaches from the ground.

The above description of the suction phenomenon reveals several aspects which seem important to highlight in terms of modeling. On the one hand, the material deformation has to be simulated thanks to a numerical method. We choose the Co-rotational Finite Element Method to achieve it (Section 3.3.2). On the other hand, the non-constant terms of the ideal gas law has to be determined. In our approach, we decided to elaborate an algorithm which is able to detect these cavities, detailed in Section 3.2. Thereby, the cavity volume can be geometrically computed afterwards (Section 3.2.6).

For modeling the pressure exerted by the air in the cavity, we chose to not focus on the air dynamics as we are more interested on the effect of the pressures on the structure than on a precise modeling of the air flows. Although the fluid dynamics has an impact on the evolution of the air pressure distribution, we think that impact is negligible in our context knowing we deal with simple cavity geometries. For this reason, we decided to not represent the dynamics of the fluid (air) in our modeling, but solely the pressures which apply onto the surfaces of the materials using uniform pressure distributions. In our modeling, we chose a constraint-based formulation to represent the pressure forces. We introduced a novel air pressure constraint in our model that we combine with the traditional contact and friction constraints. The underlying constraint formulation is described in Section 3.3.3 as well as the constraint resolution.

3.2 Geometric cavity detection

In our model, we seek to detect if the cavity of the suction cup is sealed or not. In other words, we are interested in determining the state of the cavity. On top of that, it is possible to obtain multiple small air cavities due to the object distortion. Several cavities can also arise when dealing with unusual shapes of suction objects.

Knowing air can be trapped in cavities when a suction cup interacts with a physical object, one key point of our method is therefore to detect the geometry of these cavities. In our approach, a cavity consists of three parts: the internal surface region of the suction cup, as detected by the collision model when the suction cup is colliding an other body; the surface region under the cavity and belonging to the body in collision

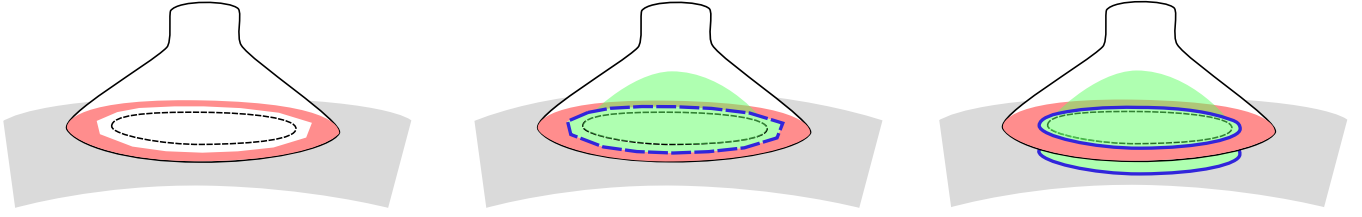


Figure 3.2 – Workflow of the cavity detection. Left: the contact surface is determined from the narrow-phase collision response (red). Middle: inner surfaces (green) and approximate inner borders (dashed blue) are identified thanks to flood-fill operations. Right: exact inner borders (continuous blue line) are computed using new vertices that are added to the mesh, and inner surface regions are paired to constitute the unified geometric cavities.

with the suction cup; and the mechanical values needed for pressure computation (pressure, volume, and air quantity). Our method detects a cavity and then handles the pressure and volume computations both from a geometrical and a mechanical points of view.

The workflow of the cavity detection process is explained in Figure 3.2 and detailed in the following subsections. First, the contact surface is determined using information from narrow-phase collision output (Section 3.2.1). Then, flood-fill operations are performed on mesh surfaces to find the inner cavity surfaces and their respective approximate inner borders (Section 3.2.2). After that, we compute what we call the exact inner borders from the approximated ones using a refinement operation (Section 3.2.3). The vertices along these exact inner borders correspond to new vertices which are added into the topology of the suction object. Thanks to a projection of the computed exact inner border plus another executions of flood-fill procedures, we pair the inner surfaces from the suction object with those of the colliding object (Section 3.2.4). The result is a set of unified geometric cavities, each composed of two inner surfaces from the two colliding objects respectively. On top of that, we use the centroids of cavities to track them over time, as we must keep track of their varying mechanical properties, such as the quantity of air in a cavity and the pressure (Section 3.2.5).

3.2.1 Contact surface determination

The contact points that make up the contact surfaces are detected thanks to proximity queries. The narrow-phase collision detection and threshold distances used in our simulation are illustrated in Figure 3.3. Each vertex below an *alarm distance* of the opposite mesh creates a pair of contacting points (one point on each mesh) with an associated contact constraint. This *alarm distance* should not be too close to zero otherwise there is a great chance that contact points are missed in this case. In fact, the value of the *alarm distance* must be carefully chosen according to the granularity of the mesh, its velocity, and the magnitude

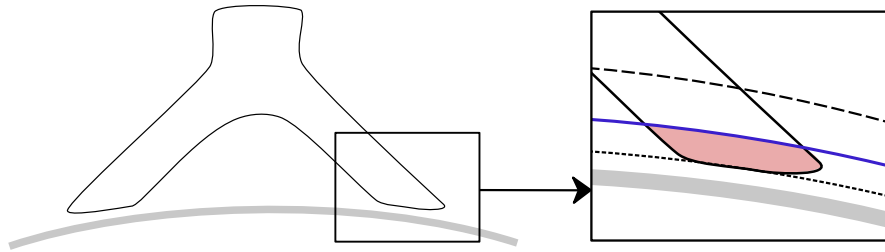


Figure 3.3 – Collision distances. All vertices below the alarm distance (dashed line) generate contact constraints. The contact distance (dotted line) delimits a gap between the two colliding objects which must be satisfied after constraint resolution. We use a sealing threshold (continuous blue line) to identify the vertices of the contact surface. The intersection of the sealing distance with the inner surface of the cavity delimits the exact inner border.

of the time step. The gap function of the contact constraint (normal constraint) is equal to the signed distance between the free-motion configurations of the overlapping objects plus a contact distance (an additional threshold to ensure that objects never interpenetrate, see dotted line in Figure 3.3). However, we use an additional threshold, the *sealing distance*, to classify points as being in contact before they strictly interpenetrate the other object. The more this value is high, the more the sealing is reinforced. We found that it especially helps to improve the robustness of the cavity detection algorithm when dealing with collisions against bumpy surfaces. Once the collision detection is complete and potential contact points have been filtered according to the distance thresholds, we label the set of all vertices which have been classified as contact points to identify the contact surface.

3.2.2 Flood-fill of inner and outer surfaces

Once the contact surface has been determined, the next stage of the cavity detection is to classify the remaining vertices from the suction object, and in particular to determine which vertices are inside the cavity, but not part of the contact surface. This is accomplished by executing a series of flood-fill procedures until all vertices have been classified as being inside the cavity, or outside. The process is illustrated in Figure 3.4.

Our flood-fill works on vertices and uses mesh topology to identify connected vertices within the collision mesh. It starts from an initial seed vertex, that we add to a stack. For each vertex in the stack, the algorithm examines the neighboring vertices in the collision mesh. If the neighboring vertex is not yet classified, it is classified and added to the stack. The algorithm proceeds through all the vertices in this stack until the latter becomes empty.

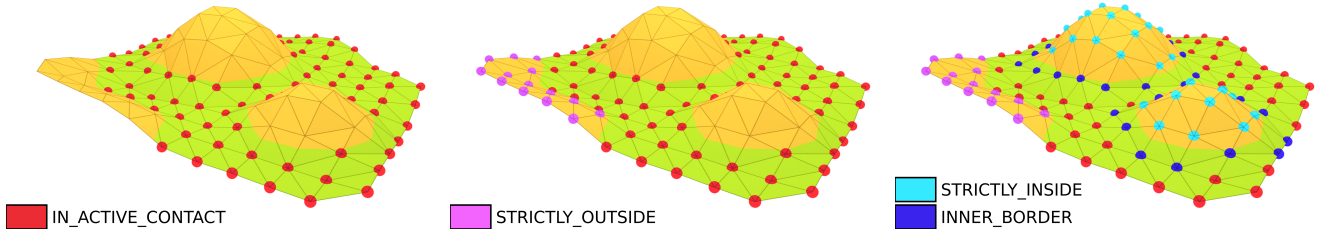


Figure 3.4 – Illustrations of the vertex classification stages before inner border refinement. The illustrated surface is a sample of the closed surface of an arbitrary suction object. The green area represents the exact contact area. Left: The contact points are classified first. Middle: Vertices outside cavity are flood-filled starting from a user-defined seed vertex. Right: Remaining vertices are flood-filled to be classified as points inside cavities and the approximated inner borders are revealed.

We first flood-fill the region of vertices that are outside the cavity with the help from a user-defined seed vertex on each collision mesh. We choose a vertex which is unlikely to be contained within the suction cavity (e.g., the top of the suction cup stem). After the first filling of the outside regions, the remaining vertices are inevitably the inner surfaces. Then we use a first unclassified vertex as a seed, to flood-fill a first inner surface. If there is only one cavity, the process stops here and all vertices are classified, otherwise, we take a new unclassified vertex as a new seed to find a new inner surface and we proceed sequentially until all vertices have been classified. Note that each time a inner surface is classified, we identify its approximated inner border.

3.2.3 Inner border refinement

When the flood-fill step identifies an inner surface, we also obtain a set of vertices approximating the inner border. We make an ordered list of these vertices along the inner border such that they form a line loop, that is, vertices in the ordered list are adjacent if they are also adjacent in the mesh.

However, as specific vertices that make up the contact surface may change between time steps, this introduces severe discontinuities in the computation of the inner border and hence the cavity volume. In order to avoid these artifacts, we refine the approximated inner border by computing the exact inner border as the contour line defined by the contact surface vertices. This set of vertices is found by the proximity queries below a given *sealing distance* (see Figure 3.3).

For each inner-border node, there is an edge that connects it to a node that is strictly inside the cavity. For each of these edges, we use linear interpolation to find the point on the edge that intersects the isocontour

defined by the sealing distance. Given that the approximated inner border nodes are sorted, the newly computed points along the contour line are sorted as well. These points constitute the exact inner border of the cavity we are looking for. We insert them as vertices into the topology of the suction cup's collision mesh. In practice, we remove the triangles which are crossed by the contour line beforehand and we apply an ear clipping triangulation algorithm [Ebe08] after adding the new vertices (Additional implementation details are given in Section 3.2.7). We refresh vertex classification of the approximated inner border after each refinement.

The reason for modifying the topology of the collision geometry is due to the fact we later compute the cavity volume gradient and the air pressure distribution from it (see Section 3.3.3), which implies to get a clean discretization of the geometry of the detected cavity beforehand in order to obtain accurate values. Furthermore, the contour line on one object is then projected onto the other, and another flood-fill operation is used to determine the cavity. This flood-fill, which we describe in the subsequent section, relies on having the exact inner border as part of the mesh.

3.2.4 Pairing cavity inner surfaces

The exact inner border determined by the contour line refinement is projected onto the collision mesh of the colliding object. As a result, the topology of the colliding object is modified in the same way as described in Section 3.2.3 whereby the collision mesh is modified to include the exact inner border. The purpose of this projection is to pair the two inner surfaces, which finally constitute a unified cavity geometry. The classification of the regions from the colliding object is performed through the application of a flood-fill procedure similar to the method described in Section 3.2.2 for the suction object. One small difference here is that one of the vertices that lies on the projected contour line is used as the seed vertex used to initialize the flood-fill on the paired surface. Each inner surface is closed by the centroid of its exact inner border. The outcome is a discretization of the cavity as two associated closed surfaces.

The projection of the inner border has two advantages. First, it means that the two interacting objects do not necessary need to have the same mesh resolution for the cavity to be detected, and furthermore the result is a clean discretization of the inner cavity. Note that at the end of the time-step, the initial topologies of both objects are restored.

3.2.5 Tracking cavities over time

The vertex classification algorithm is independently executed at each time-step without considering the classifications coming from the previous time step. Therefore, the detected geometric cavities are not persistent. However, our method introduces persistent mechanical cavities which carry the mechanical values (pressure, air quantity, volume). These persistent mechanical cavities are linked with the non-persistent geometric cavities which allow to track the cavities over time. If the cavity was already detected at the previous time step, the previous mechanical values are re-used. The purpose of our cavity tracking algorithm is to decide if we should link detected cavities to existing mechanical cavities at the previous time step, or to create new ones.

The algorithm proceeds as follows: first, it associates each detected cavity to the closest cavity at the previous time-step. The association is handled through a proximity criterion, corresponding to the Euclidean distance between the cavity centroids. The proximity is only considered if the given Euclidean distance does not exceed a maximal value, which is defined by the user. Then, our algorithm identifies the surface cavities which are not assigned to any previously existing mechanical cavity, and creates new mechanical cavities for them. Finally, the previously existing mechanical cavities that have not been associated to a detected surface cavity are deleted.

3.2.6 Volume computation

Once we have a clean discretization of the geometric cavity thanks to the exact inner border, we are able to compute its volume. To this end, we close the two surface regions of the mechanical cavity to sum-up their signed volume. This aspect elucidates why curved surfaces inside cavities can be handled with our method, no matter if it is a bump or a hollow region.

Each geometric cavity is virtually closed by the centroid of the inner border, and the volume is then geometrically-computed using the technique from [ZC01]: it consists in creating an arbitrary reference point P^O whereby we connect the vertices P_t^A, P_t^B, P_t^C of the triangles from the collision mesh. The term t denotes the triangle indices and we assume it exists n triangles. In practice, P^O is actually the centroid of the inner border of the geometric cavity. Each triangle t thus reveals a tetrahedron for which we compute its signed volume v_t . The sign of its volume actually depends on the position of P^O according to the plane of the concerned triangle. In case the created point is located on the side of the plane where the triangle's normal \hat{n}_t points out, the sign is negative. In the opposite case, the sign is positive. By summing the signed volumes v_t of all the revealed tetrahedra, the outcome is v the volume of the geometric cavity. Mathematically

speaking, we can write down the calculation of v as:

$$v = \sum_{t=0}^n v_t = \sum_{t=0}^n s_t \frac{U_t^{OA} \cdot (U_t^{OB} \times U_t^{OC})}{6} ,$$

$$\text{with } U_t^{OA} = (P_t^A - P^O) ; U_t^{OB} = (P_t^B - P^O) ; U_t^{OC} = (P_t^C - P^O) ,$$

$$\text{and } s_t = \begin{cases} 1 & \text{if } U_t^{OA} \cdot \hat{n} > 0 \\ -1 & \text{if } U_t^{OA} \cdot \hat{n} < 0 \end{cases}$$

3.2.7 Dynamic topology

As seen in Figure 3.2, the cavity detection process performs the projection of the exact inner border of the suction cup onto the colliding object. The aim is to pair the two geometric cavities while obtaining a clean discretization of these, and then summing their volumes to finally get the volume of the mechanical cavity. This projection implies to dynamically change the topologies of the two meshes.

In the context of the suction cup, the exact inner border is actually a closed chain of segments knowing that the points between the consecutive segments belong to existing edges. The topology adaptation is consequently straightforward: each existing triangle is split in two polygons by a portion of the chain of segments. From this state, the only remaining operation consists in applying a triangulation algorithm onto the revealed polygons. We chose to apply the well-known ear clipping algorithm [Ebe08] for its simplicity but any triangulation algorithm would work.

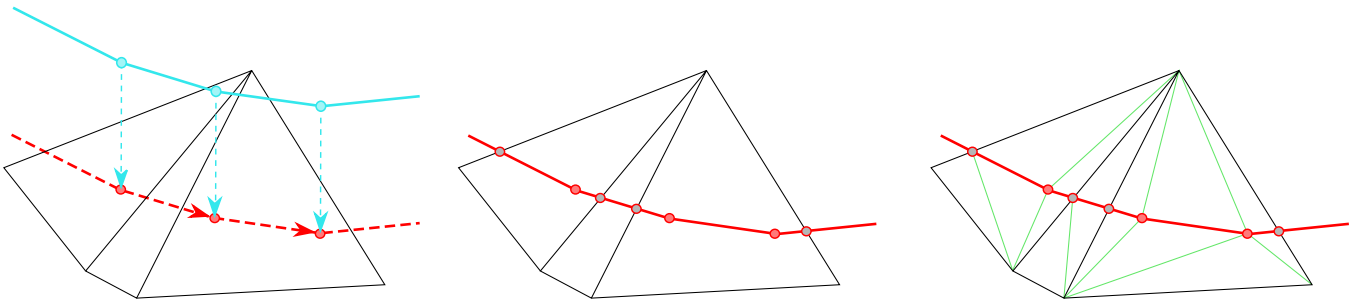


Figure 3.5 – Illustration of the procedure consisting in dynamically adapting the topology of the colliding mesh after inner border projection. From left to right: 1) Points along the exact inner border (blue) are projected onto the colliding object. 2) Projected points (red) are linked and intersection points with existing edges are added (Grey). 3) Each original triangle is split in two polygons through the projected inner border. These polygons are triangulated (green) thanks to the well-known ear clipping algorithm.

The topology adaptation is illustrated in Figure 3.5 through three steps from left to right: In the first step, the sorted points along the exact inner border (blue) are projected onto the triangles of the colliding object. Note that each projected point (red) now belongs to a triangle and this data is kept in memory. In the second step, those points are linked together and intersection points with existing edges are added in-between (green). More precisely, we trace a vector from a first projected point to the next one and we are looking for the intersected edge of the current triangle. Then we know on which neighbor triangle we must jump to continue the operation until we reach the next projected point. We do not omit to compute the intersection points (Grey) along this process. We repeat the operation until all the projected points have been treated. The outcome is that all existing triangles are now split in two polygons. In the third step, we execute the ear-clipping algorithm (triangulation) onto the revealed polygons (green). As a result, original triangles are respectively substituted by sets of smaller triangles.

3.3 Numerical methods

We previously described the physical and geometrical aspects of the simulation of the suction phenomenon. In this subsection we detail the numerical aspects of the modeling.

3.3.1 Simulation pipeline

Our simulation pipeline proceeds by first simulating the unconstrained motion (also called free-motion) of deformable objects. It is performed thanks to an asynchronous preconditioner using *LDL* decomposition followed by a direct solver. *LDL* decomposition means the system matrix \mathbf{A} is decomposed through the form $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ with \mathbf{L} a lower unit triangular matrix and \mathbf{D} a diagonal matrix. After this first step, a collision detection stage determines the contact points, namely the points of the collision meshes included in the overlapping of the free-motion configurations. The cavity detection algorithm described in Section 3.2 is then executed. Finally, a complementarity problem is defined and then solved through a Gauss-Seidel solver. The aim is to resolve contacts, simulate friction and the suction phenomenon all together using constraints. The resolution of the complementarity problem results in a motion correction applied over the unconstrained motion (see Section 3.3.3). This simulation pipeline is summarized in Algorithm 1 in which the mathematical terms are detailed further in this chapter.

Algorithm 1 General pipeline of our suction simulation.

\mathcal{O}_∞ and \mathcal{O}_ϵ represent the two colliding objects while \mathcal{C} represent the collision response data.

p, v, n are respectively the pressure, volume, and air quantity in the cavity.

\mathbf{A} is the system matrix, \mathbf{H} the constraint matrix, $\boldsymbol{\lambda}$ the vector of Lagrange multipliers, and h the time step.

```

1: procedure STEP
2:   solve  $\mathbf{A}\Delta\dot{\mathbf{x}} = b$ 
3:   // Update configuration
4:    $\mathbf{x}_0 \leftarrow \mathbf{x} + h(\dot{\mathbf{x}} + \Delta\dot{\mathbf{x}})$ 
5:    $\mathbf{H}_c, \mathcal{C} \leftarrow \text{COLLISIONDETECTION}(\mathcal{O}_\infty, \mathcal{O}_\epsilon)$ 
6:    $\mathbf{H}_p, n, v \leftarrow \text{CAVITYDETECTION}(\mathcal{O}_\infty, \mathcal{O}_\epsilon, \mathcal{C})$ 
7:    $\boldsymbol{\lambda}, p \leftarrow \text{SOLVECONTACTPRESSURE}(\mathbf{H}_c, \mathbf{H}_p, n, v)$ 
8:   update  $\Delta\dot{\mathbf{x}} \leftarrow \Delta\dot{\mathbf{x}} + h\mathbf{A}^{-1}\mathbf{H}_c^T\boldsymbol{\lambda} + h\mathbf{A}^{-1}\mathbf{H}_p^Tp$ 
9:    $\dot{\mathbf{x}} \leftarrow \dot{\mathbf{x}} + \Delta\dot{\mathbf{x}}$ 
10:   $\mathbf{x} \leftarrow \mathbf{x} + h\dot{\mathbf{x}}$ 
11: end procedure

```

3.3.2 Corotational Finite Element Method

The simulation of the deformable objects is performed through the Co-rotational Finite Element Method (FEM) and an Euler implicit time-integration scheme. The word cororotational means the rotational part of the deformation is computed separately from the rest in order to avoid an inflating artifact (Figure 2.2). Each deformable object is represented as a set of volumetric tetrahedra (FEM mesh). Additional triangle meshes representing the surface of each object are used for collision detection purpose (collision mesh). Theses two topological representations are illustrated in Figure 3.6.

We suppose that the state of the mechanical system is given by the position \mathbf{x} and the velocity $\dot{\mathbf{x}}$ of a total of N nodes across the finite element models. The implicit stepping of the unconstrained elastic system involves computing updates to the nodal velocities $\Delta\dot{\mathbf{x}}$ by solving

$$\underbrace{(M - hC - h^2K)}_A \Delta\dot{\mathbf{x}} = \underbrace{hf(\mathbf{x}, \dot{\mathbf{x}}) + h^2K\dot{\mathbf{x}} + hg}_b. \quad (3.1)$$

Here, h is the time step, M , C , and K are respectively the mass, damping, and stiffness matrices of the finite element models, $\dot{\mathbf{x}}$ are the nodal velocities, and elastic forces are given by $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$, and external forces, such as gravity, given by \mathbf{g} .

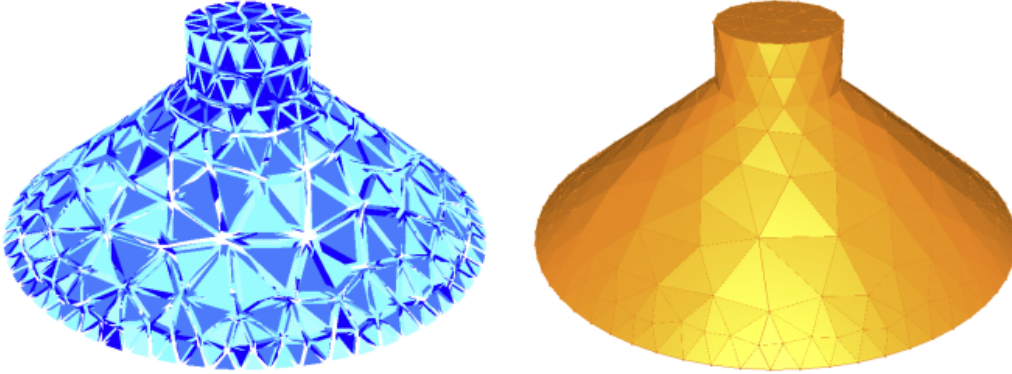


Figure 3.6 – Two topological representations of a *suction object*. Left: FEM mesh composed of tetrahedral elements. Right: Collision mesh composed with triangles along the object surface.

3.3.3 Constraint formulation

First of all, we decided not to take into account the dynamics of the air in our model but only the air pressure distributions along surfaces that we consider uniform. Indeed, although the air dynamics has an impact on the pressure distribution, we think it is negligible due to the simple geometries of the cavities. Moreover, computing fluid dynamics would consume time performance knowing we already have to simulate deformable objects which is likewise costly.

A reduced system of equations involving only the constraint variables is built and the Lagrange multipliers are computed by defining a Complementarity Problem (CP). This problem accounts for the bounds of the non-interpenetration, friction, and our novel air pressure constraints. As the gas law is non-linear, we favored the use of a Projected Gauss-Seidel (PGS) solver that allows iteratively solving small blocks of non-linear constraints while taking into account the coupling between them. Finally, we apply the correction forces resulting from the constraint solve to the nodal degrees of freedom (DOFs). It results in a penetration-free configuration of the FEM meshes, but also accounts for the depressurization and suction of the cavities.

Since contacts are an essential component of suction simulation, we constrain the elastic bodies by introducing non-interpenetration and friction constraints, that we regroup under the name of frictional contact constraints. Suction is also modeled in our simulations using a novel air pressure constraint we coupled with the other constraints. The constraint matrix \mathbf{H} , as illustrated in Figure 3.7, embed the directions of the frictional contact constraint forces and the uniform air pressure distributions of the detected cavities. In contrast, the intensities of the frictional contact forces as well as the scalar internal pressures from the cavities are represented by Lagrange multipliers.

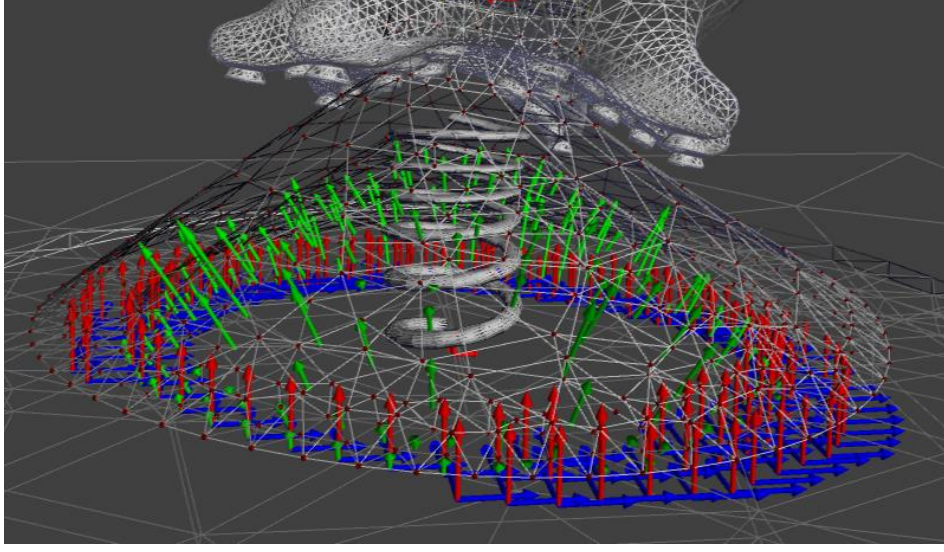


Figure 3.7 – When a cavity exists due to collision, three types of rows exist in the constraint matrix: non-penetration constraint directions (red), friction constraint directions (two per contact, blue), and the uniform negative pressure distribution in the cavity (green).

For reasons of clarity in our formulation, let's split \mathbf{H} into two sub-parts: \mathbf{H}_c and \mathbf{H}_p . On the one hand, $\mathbf{H}_c \in \mathbb{R}^{3k \times 3N}$ encodes the non-interpenetration and friction constraints of each contact point, assuming there are k contact points. We observe that \mathbf{H}_c provides important information about the separation and tangent slip velocity at contacts $\mathbf{H}_c \dot{\mathbf{x}}$. By the way, the Lagrange multipliers related to these frictional contact constraints are gathered into the vector $\boldsymbol{\lambda}$. On the other hand, $\mathbf{H}_p \in \mathbb{R}^{l \times 3N}$ encodes the uniform air pressure distribution in the cavities with l the number of cavities. Assuming there is only one detected cavity, implying $l = 1$, the scalar internal pressure is represented by the Lagrange multiplier p . Letting $\mathbf{H}_p \in \mathbb{R}^{1 \times 3N}$ be the volume gradient of a sealed cavity, we observe that this is a row vector only has non-zero components at nodes that participate in defining the volume of the cavity. This constraint matrix also provides the rate of volume change in the cavity based on nodal velocities as

$$\dot{v} = \mathbf{H}_p \dot{\mathbf{x}} \quad (3.2)$$

With the system in (3.1) being solved to obtain $\Delta \dot{\mathbf{x}}$, that is, at the velocity level, we have constraint forces acting according to Lagrange multipliers $\boldsymbol{\lambda}$ and p , which respectively enforce the contact and air pressure constraints. These forces act in constraint directions determined by \mathbf{H}_c^T and \mathbf{H}_p^T , and a time step factor of h is used to make these constraint forces act as impulses. An update to the unconstrained motion

computed in (3.1) can then be obtained by:

$$\Delta \dot{\mathbf{x}} \leftarrow \Delta \dot{\mathbf{x}} + h \mathbf{A}^{-1} \mathbf{H}_c^T \boldsymbol{\lambda} + h \mathbf{A}^{-1} \mathbf{H}_p^T p \quad (3.3)$$

We note that the terms $\mathbf{A}^{-1} \mathbf{H}_c^T$ and $\mathbf{A}^{-1} \mathbf{H}_p^T$ can be computed efficiently by using a sparse linear solve. For each contact point, a block is added to the matrix \mathbf{H}_c that encodes the contact normal and friction directions. We use $\lambda_n \in \mathbb{R}$ to denote the non-interpenetration constraint force of a contact, and $\boldsymbol{\lambda}_t \in \mathbb{R}^2$ for the frictional forces.

3.3.4 Constraint resolution

The inner loop of the Projected Gauss-Seidel (PGS) solver (Algorithm 2) computes contact forces block-wise by solving for the tuple of constraint forces that satisfy Signorini and Coulomb laws. This involves solving three non-linear equations with complementarity conditions:

$$0 \leq \lambda_n \perp \mathbf{H}_{c,n}(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}}) \geq 0 \quad (3.4)$$

$$(\|\boldsymbol{\lambda}_t\|_2 < \mu \lambda_n) \oplus \left(\boldsymbol{\lambda}_t = -\mu \lambda_n \frac{\mathbf{H}_{c,t}(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}})}{\|\mathbf{H}_{c,t}(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}})\|_2} \right) \quad (3.5)$$

where \oplus denotes exclusive disjunction. In (3.4), the term $\mathbf{H}_{c,n}(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}})$ is the velocity in the normal direction at the contact at the end of the time step. Similarly, in (3.5), $\mathbf{H}_{c,t}(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}})$ is the 2D velocity in the tangent plane at the contact, and when $\|\boldsymbol{\lambda}_t\|_2 < \mu \lambda_n$, this tangential velocity is constrained to be zero.

The Schur complement is used to solve for contact constraint forces, accounting for the unconstrained motion update $\Delta \dot{\mathbf{x}}$ and the current estimate of the suction pressure p :

$$h \mathbf{H}_c \mathbf{A}^{-1} \mathbf{H}_c^T \boldsymbol{\lambda} = \mathbf{H}_c (\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}} - h \mathbf{A}^{-1} \mathbf{H}_p^T p) . \quad (3.6)$$

When multiplied by the constraint matrix, we obtain the nominal constraint gap function velocity $g_{cb} = \mathbf{H}_c(\dot{\mathbf{x}} + \Delta \dot{\mathbf{x}})$ due to the known forcing terms and the current velocity. Likewise, we solve $h \mathbf{A}^{-1} \mathbf{H}_p^T$ in advance, allowing us to write the compliance, i.e., rate at which the constraint velocity changes due to pressure, as $\mathbf{W}_{cp} = h \mathbf{H}_c \mathbf{A}^{-1} \mathbf{H}_p^T$. Thus:

$$h \mathbf{H}_c \mathbf{A}^{-1} \mathbf{H}_c^T \boldsymbol{\lambda} = -g_{cb} - \mathbf{W}_{cp} p . \quad (3.7)$$

Note that \mathbf{W}_{cp} could be updated if the vertices that define the boundary of the cavity were to change, but contacts included in the solution are fixed at the beginning of the time step. The additional challenge in

coupling contact and pressure is that the directions of \mathbf{H}_p would also need to change in the situation where contact forces show a gap forming at the inner boundary of the contact patch. We avoid this complication not including vertices that form the inner boundary in the volume gradient \mathbf{H}_p .

As mentioned above, the PGS solver for frictional contact follows the well proven approach [JAJ98] of solving in sequence blocks of 3 equations (contact and friction) for each contact. The corresponding Lagrange multipliers λ_n and λ_t are updated in turn and projected to bounds as necessary. The pressure constraint, however, is non-linear and requires closer inspection to ensure the correct solution. The initial quantity of air in a cavity is set to n , and at each time step we compute the pressure p and a change in air quantity $\Delta n \leq 0$. If the gas law constraint can be satisfied, then the cavity remains airtight and the quantity of trapped air remains constant, i.e., $\Delta n = 0$. However, if the gas law cannot be satisfied without increasing the pressure above the maximum P_{max} , then we allow an air quantity change $\Delta n < 0$ during the time step.

We elaborated on two versions of our suction model by considering *active suction* on the one hand and *passive suction* on the other hand. *Active suction* is the simplest version: it assumes that the cavity of the cup is linked to a vacuum pump in practice, meaning the air pressure is directly controlled by user through a regulator. In this case, the pressure p is known and the only remaining unknown becomes the air quantity n we can simply compute as $n = \frac{pv}{RT}$ by following the ideal gas law.

In contrast, *passive suction* states that we deal with a traditional suction cup without any artificial vacuum device. It means p and air quantity n still have to be determined. In the case of *passive suction*, the pressure computation must account for how the volume will change over the next time step due to contact forces, the change in the amount of trapped air, and the pressure itself. Thus we solve for p in:

$$p(v + h\dot{v}) = (n + \Delta n)RT \quad (3.8)$$

and using Equation 3.2 with the velocity at the next time step we obtain:

$$p(v + h\mathbf{H}_p(\dot{\mathbf{x}} + \Delta\dot{\mathbf{x}})) = (n + \Delta n)RT . \quad (3.9)$$

Recall that before starting the contact PGS solve, we compute solutions for the velocity changes due to b and the pressure constraint direction \mathbf{H}_p^T , to multiply by \mathbf{H}_c and thus prepare the contact constraint velocities g_{cb} and \mathbf{W}_{cp} used in Equation 3.7. Here, we similarly prepare in advance the *volume velocity* due to forcing terms b and the current FEM velocity as $g_{pb} = \mathbf{H}_p(h\mathbf{A}^{-1}b - \dot{\mathbf{x}})$, and the compliance, i.e., volume velocity produced by pressure, $\mathbf{W}_{pp} = \mathbf{H}_ph\mathbf{A}^{-1}\mathbf{H}_p^T$ (note that \mathbf{W}_{pp} is a scalar if there is only one cavity). Furthermore, pre-computing (at start of time step) compliance $\mathbf{W}_{pc} = (\mathbf{H}_ph\mathbf{A}^{-1})\mathbf{H}_c^T$, Equation 3.9 can be

expanded as:

$$p(v + hg_{pb} + h\mathbf{W}_{pc}\boldsymbol{\lambda} + h\mathbf{W}_{pp}p) = (n + \Delta n)RT. \quad (3.10)$$

Collecting terms on the left hand side yields:

$$\underbrace{(h\mathbf{W}_{pp})}_{a>0}p^2 + \underbrace{(v + hg_{pb} + h\mathbf{W}_{pc}\boldsymbol{\lambda})}_{b>0}p + \underbrace{(-(n + \Delta n)RT)}_{c<0} = 0. \quad (3.11)$$

This quadratic in p is easy to solve for one cavity (and cavity by cavity with Gauss-Seidel in case of multiple cavities), if roots are not complex. Let's prove the impossibility of complex roots. First, note that \mathbf{W}_{pp} , and quadratic term coefficient a , are positive, provided that \mathbf{A} and its inverse are symmetric positive definite (which is typically the case, and can be guaranteed by choosing a small enough time step even if the simulation state involves a large and indefinite elastic stiffness \mathbf{K}). Furthermore, the value of c must be negative (or zero) because $n + \Delta n$ must always be a positive (or zero) quantity of air trapped in the cavity at the next time step, and constants R and T are positive. Thus, the discriminant $d = b^2 - 4ac$ is always positive, and it is also greater or equal to b^2 .

Now let us understand the impossibility of multiple positive solutions for p . Notice that the linear term coefficient b measures the volume at the next time step accounting for the current velocity, external and elastic forces, and the contact forces, while ignoring the influence of pressure on the cavity. The total will be non-negative provided that we can rely on the resolution of contacts to ensure contact forces that prevent the cavity going past a zero volume state with interpenetration between the surfaces forming the cavity. Thus, with $b > 0$, we have one negative solution $p = (-b - \sqrt{d})/(2a)$, and with the discriminant $d \geq b^2$, we will also always have a positive (or zero) solution:

$$p = (-b + \sqrt{d})/(2a). \quad (3.12)$$

This is the solution we chose.

When the solution for p exceeds P_{max} , then p must be clamped at maximum value, and a change in air quantity over this time step is computed by evaluating the quadratic Equation 3.11 with the pressure set to P_{max} , that is:

$$\Delta n = \frac{aP_{max}^2 + bP_{max}}{RT} - n. \quad (3.13)$$

We observe that Δn is zero if p is exactly P_{max} (i.e., the pressure has not yet exceeded the limit, does not need to be clamped, and the gas law is satisfied). When the solution p is clamped to P_{max} , we notice that Δn becomes negative as the first term in Equation 3.13 becomes smaller. That is, the air quantity will never increase when p is clamped. This first term is positive because coefficients a and b , and the maximum pressure are positive (the air quantity can never be reduced below zero over a time step). Thus obtain at a complementarity condition:

$$0 \leq (P_{max} - p) \perp -\Delta n \geq 0 . \quad (3.14)$$

A summary of our constraint resolution through our PGS solver is unfolded below (Algorithm 2).

Algorithm 2 Solver algorithm for pressure and contact forces (Projected Gauss-Seidel).

```

1: procedure SOLVECONTACTPRESSURE( $\mathbf{H}_c, \mathbf{H}_p, n, v$ )
2:    $\Delta n = 0$ 
3:   for  $i < \text{max iterations}$  do
4:     // Solve for contacts
5:     for each contact
6:       solve and project  $\lambda_n, \lambda_t$  // Equations 3.4-3.7
7:     end for
8:     // Solve for pressure
9:      $p = (-b + \sqrt{d})/2a$  // Equations 3.11 and 3.12
10:    if  $p > P_{max}$  then
11:       $p = P_{max}$ 
12:       $\Delta n = (aP_{max}^2 + bP_{max})/(RT) - n$  // Equation 3.13
13:    end if
14:     $i = i + 1$ 
15:  end for
16:  return  $\lambda, p, \Delta n$ 
17: end procedure

```

3.4 Conclusion

In conclusion of this chapter, we proposed a new suction model composed of several features that we described from a modeling point of view.

The first important feature of the suction model is a cavity detection algorithm. The latter is able to handle multiple cavities meaning we can simulate suction effect on objects with arbitrary geometry. Moreover,

the algorithm computes exact inner borders involving dynamic topological changes to obtain a clear discretization of the detected air cavities and thus avoid discontinuities.

The second important feature is a novel pressure constraint we coupled with non-interpenetration and friction constraints. These constraints are gathered to form a Non-linear Complementarity Problem (NCP) which is built using Lagrange multipliers and solved thanks to a Projected Gauss-Seidel solver. In the context of passive suction, the scalar pressure values of the cavities are found thanks to a resolution which is based on the ideal gas law. The next chapter clarifies implementation details about these two contributions.

IMPLEMENTATION & SIMULATION

Contents

4.1	Implementation	69
4.1.1	General organization of SOFA	70
4.1.2	Cavity detection	70
4.1.3	Constraint system	73
4.1.4	GPU acceleration	74
4.2	Illustrative scenarios	74
4.3	Computation time performance	77
4.4	Conclusion	79

In Chapter 3, we described the equations and algorithms related to our scientific contributions about the modeling of the suction phenomenon. In the first part of this chapter, we explained how we implemented these equations and algorithms into SOFA, a C++ simulation framework, by giving details in terms of software design. Then, in the second part of this chapter, we describe the multiple simulation scenarios we implemented in order to test our simulation model in terms of features and time-performance. Besides, the scenarios related to our model validation are presented in Chapter 5.

4.1 Implementation

We implemented our modeling and algorithms in C++ using SOFA [ACF⁺07], a simulation framework with facilities for handling the simulation of deformable objects. This decision was essentially motivated by several reasons:

- The framework already contains a lot of interesting features. In particular, we were interested in the collision detection features, the time-integration schemes, a simulation pipeline suiting our requirements, an existing constraint solver (Gauss-Seidel), and the graphical user interface. The fact that such features are provided resulted in time saving.

- It is an open-source framework. Thus, when we detect a bug in the framework, we do not have to wait for a patch because we can solve it ourselves. In addition, we are completely free to change the design and behavior of the core components.
- SOFA is a modular framework relying on scene components (see Section 4.1.1).
- The framework contains a large community whose members are mainly from the Inria laboratory, which is very convenient and helpful in the context of our research.

In order to implement the simulation of the suction phenomenon from our modeling presented in Chapter 3, we created new software components that we assembled into a dedicated SOFA plugin.

4.1.1 General organization of SOFA

In SOFA, a simulation is setup by the definition of a scene in XML or Python language wherein we describe a set of SOFA components. These components determine the techniques we use through the simulation pipeline as well as the parameters of the simulated objects. In SOFA, one simulated object can have multiple representations, namely a FEM mesh, a collision mesh and a visual model. This kind of configuration is the most common case. Each representation relies on a mesh. For each vertex of this mesh, the degrees of freedom including positions and velocities are stored in a `MechanicalObject` component. And those representations are linked together thanks to `Mapping` components. Most of the time, the topology of the FEM is composed of tetrahedra, leading to a suitable geometric and volumetric representation of the object. In the case of the collision mesh, knowing that it is a surface mesh, its topology is commonly made of triangles. These topologies are likewise defined by SOFA components inside the described scene. Note that the physics parameters of the materials are also defined through components.

4.1.2 Cavity detection

Vertex classification: As explained in Section 3.2, the points of the collision meshes are classified just after the collision detection stage and the classification is then adjusted after the computation of the exact inner border because of vertices which are dynamic added into the topology. A simplified UML class diagram about vertex classification operations is given in Figure 4.1. On collision, the two interacting objects, namely the suction cup and the other colliding object, each own classification data as an instance of the `VertexClassificationData` class. In fact, the latter stores vertex classification as a vector of `PointTags` whereby order follows the indices of the vertices of the collision mesh. In this way, we can easily access to the classification value (tag) of a given vertex knowing its index. But we also would like to efficiently browse the set of points of a given tag. That is why `VertexClassificationData` additionally stores one vector of point indices per tag. Theses accelerating data structures are populated once the entire

classification of the collision mesh is completed. About the vertex classification operations, notably the flood-fill procedure, those are defined in a dedicated class called `VertexClassifier`.

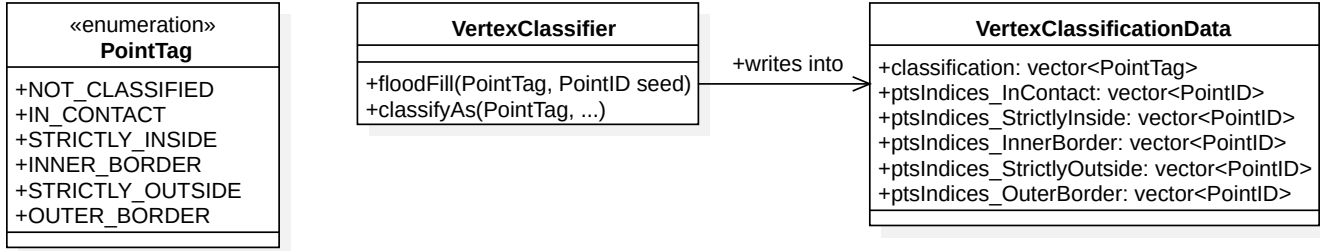


Figure 4.1 – Simplified UML class diagram about vertex classification.

Dynamic topology: In the process of the cavity detection, the approximated inner border is classified and then refined by adding new vertices into the topology of the collision meshes (Section 3.2). As shown in Figure 4.2, we introduced the class `TopologyAdaptator` which computes the exact inner border of the cavity and let it appear onto the topology of the two collision meshes. In fact, the projection of the exact inner border is performed thanks to `RayCastingProjector`. In SOFA, topological changes can already be handled by a native `TriangleSetTopologyModifier`. However, in our case, we need to change the mesh topology but we also want to restore the original one at the end of the time-step. To answer to this problematic, we extended the base class by creating the child class `TriangleSetTopologyModifierWithRollback`. This one is able to register all the topological changes in a way they can be rolled back. Besides, note that the triangulation method we use (ear clipping) has been written in a separated class called `EarClipping` for better code flexibility.

Cavity linkage: We also explained in Section 3.2 that a cavity is composed of two cavity meshes (inner surfaces that we close using centroids) and an associated mechanical cavity. As illustrated in Figure 4.3, related data is respectively stored in classes `SingleGeometricCavity` and `SingleMechanicalCavity`. On the one hand, `SingleGeometricCavity` stores the geometrically-computed volume of the associated cavity mesh. On the other hand, `SingleMechanicalCavity` stores the real volume of the cavity, computed as the sum of the volumes of the two cavity meshes. In fact, the aim of `SingleMechanicalCavity` is to store all the mechanical values related to the ideal gas law (the volume of the cavity, the quantity of air inside, and the inner pressure). In addition, it contains the cavity state and a boolean indicating that the mechanical cavity is either a fresh new cavity or an existing one coming from the previous time-step. Whereas the two geometric cavities are directly paired through the projection of the exact inner border, they have to be linked to mechanical cavities over time (Section 3.2.4). This feature is fulfilled

by `CavityLinker`. As a reminder, the linkage is based on the centroids of the geometric and mechanical cavities. These centroids are defined as an attribute in the base class `SingleAbstractCavity`. Besides, the instances of `SingleGeometricCavity` and `SingleMechanicalCavity` are contained into the respective containers `GeometricCavitiesData` and `MechanicalCavitiesData`. These two classes are actually template specializations of the template class `AbstractCavityData`. The reason of this implementation choice is that `AbstractCavityData` is able to generate cavity IDs and data is indexed in a way that a given cavity, either mechanical or geometric, can be accessed directly from its own ID. It also explains why there is an *id* attribute in class `SingleAbstractCavity`.

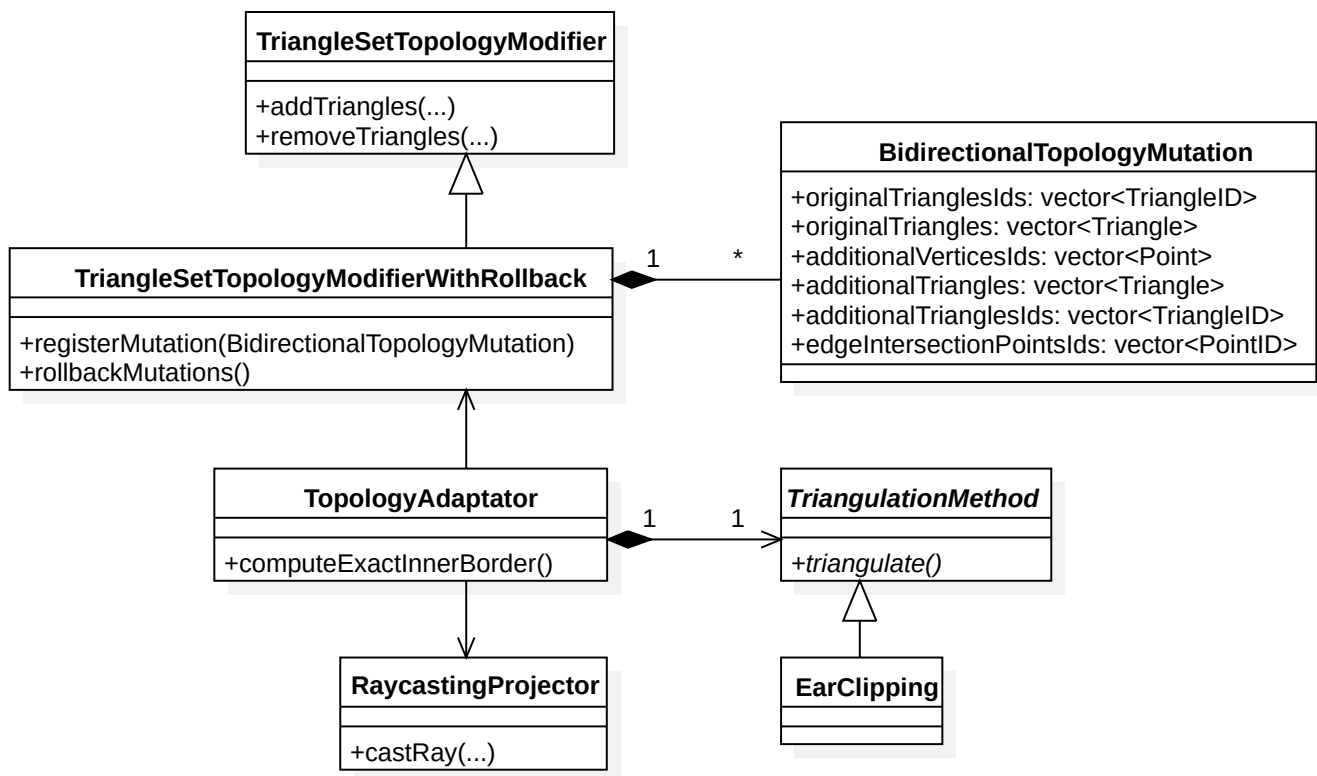


Figure 4.2 – Simplified UML class diagram about how we handle dynamic topology in order to compute the exact inner border of the cavities.

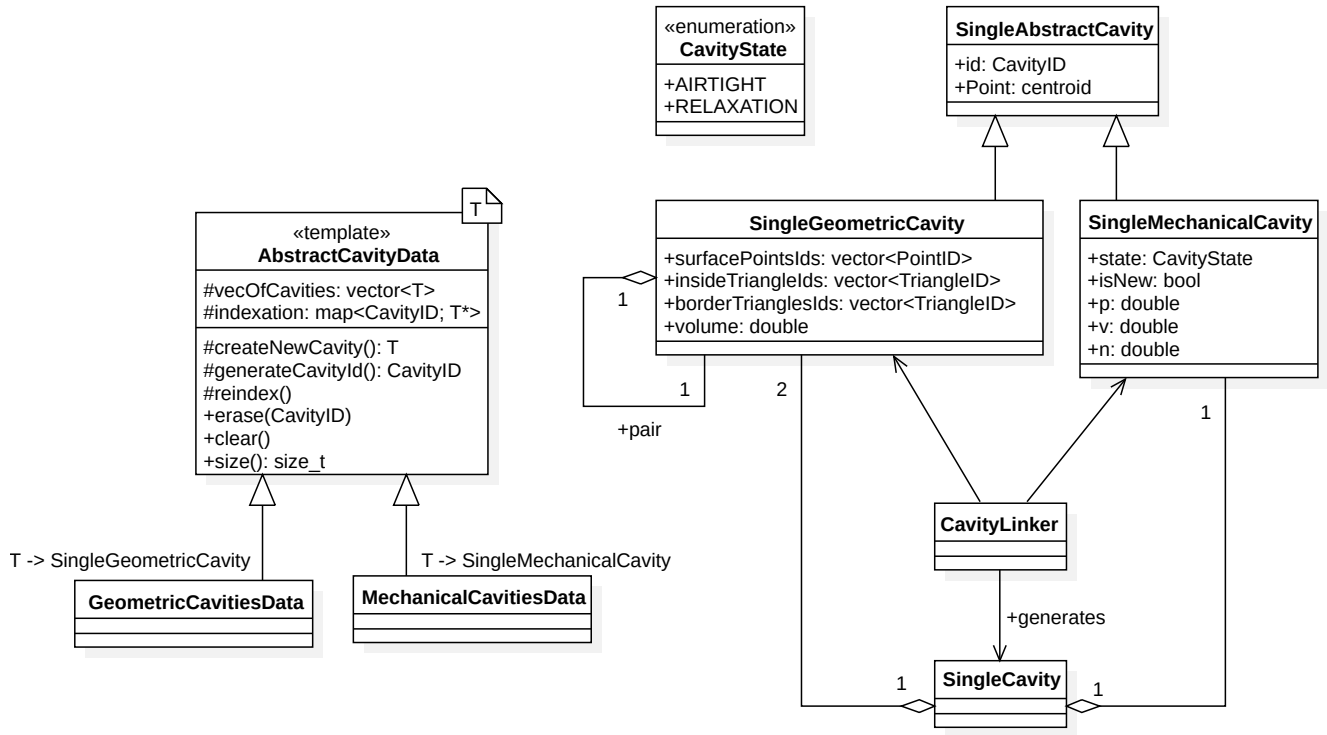


Figure 4.3 – Simplified UML class diagram about cavity linkage.

4.1.3 Constraint system

Regarding the constraint system, namely its construction and the resolution of it, this aspect is essentially performed thanks to the classes illustrated in Figure 4.4. More precisely, it works as follows: once the unconstrained configurations of the scene objects have been computed and the collision detection is finished, **GenericConstraintSolver** triggers the construction of the constraint system through a visitor mechanism (design pattern) which is native to SOFA. A visitor browses all the possible **PairInteractionConstraint** instances which are actually components of the current SOFA scene and call their **buildConstraintViolations** methods. It consequently generates multiple kinds of constraints which are aggregated into the constraint matrix: **ContactConstraintSet** generates contact and friction constraints into the constraint matrix while **PressureConstraintSet** generates air pressure constraints. In addition, respective **ConstraintResolution** instances are generated knowing that **GenericConstraintSolver** has access to this data.

The entire complementarity problem (or constraint system) is solved by the **GenericConstraintSystem** (native class in SOFA) which iterates over the generated **ConstraintResolution** objects and calls their resolution methods until the tolerance or the maximum number iterations criterion is satisfied. Note that

`GenericConstraintSolver` is actually a Gauss-Seidel solver with successive over-relaxation.

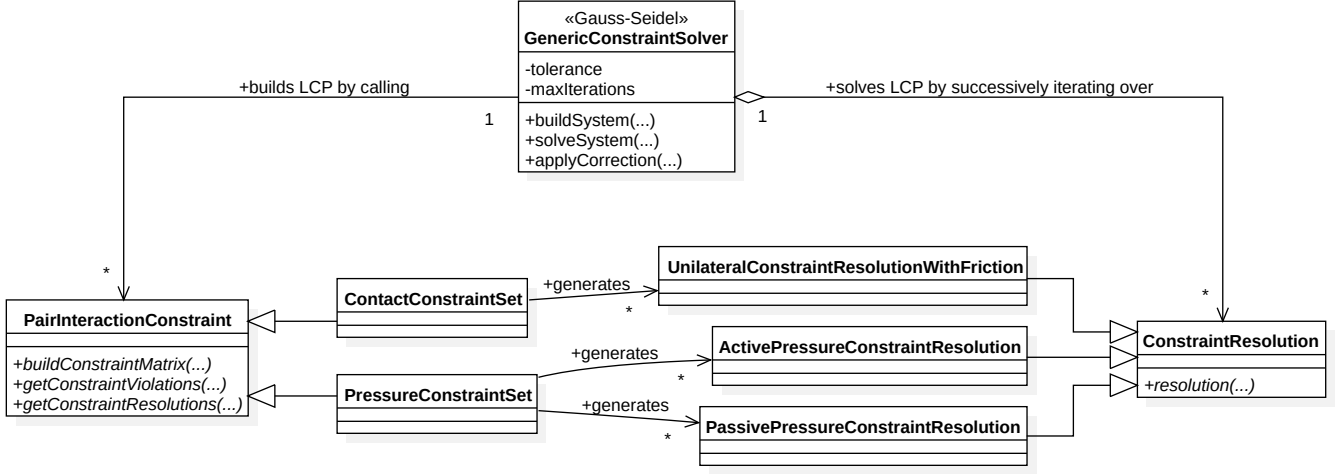


Figure 4.4 – Simplified UML class diagram about constraint implementation.

4.1.4 GPU acceleration

We use an asynchronous preconditioner [CADC10] with a GPU implementation. It is based on a LDL decomposition which allows us to reduce the condition number of our system (i.e., Equation 3.6), by approximating the system matrix \mathbf{A} in a manner that is easy to invert. The GPU implementation also allows us to accelerate the computation of the \mathbf{W} matrix defined as $\mathbf{W} = \mathbf{h} \mathbf{H} \mathbf{A}^{-1} \mathbf{H}^T$. This matrix is actually built to solve the constraint system knowing $\boldsymbol{\delta} = \boldsymbol{\delta}^0 + \mathbf{W} \boldsymbol{\lambda}$ with $\boldsymbol{\delta}$ the current constraint violations, $\boldsymbol{\delta}^0$ the initial violations which are evaluated from the overlapping of the free-motion configurations of the two colliding objects (in the case of frictional contact constraints), and $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. The preconditioner is asynchronously recomputed on a dedicated thread, while the primary thread uses the last-computed \mathbf{A} matrix approximation for solving purpose. In terms of implementation, the preconditioner was already provided in a SOFA plugin called `SofaAsyncSolvers`, which is actually an extend of the plugin `SofaCUDASolvers`.

4.2 Illustrative scenarios

We simulated different scenarios to illustrate the performances of our approach. These scenarios are listed and described in Table 4.1. In all these scenarios except the octopus arm scenario, the Young modulus of the suction cup is set to 338 kPa, which corresponds to soft silicone. In all experiments, we set the

Poisson ratio to 0.42. While our experiments include suction cups of several different shapes, the first four scenarios below involve a standard suction cup shape, and its mass is equal to 30 g. The diameter of the outside border measures 40 mm. The diameter of the cavity is 8 mm at the lowest and 32 mm at the highest (inner border). The height of the cavity is 9 mm. And the thickness of the rim is 4 mm.

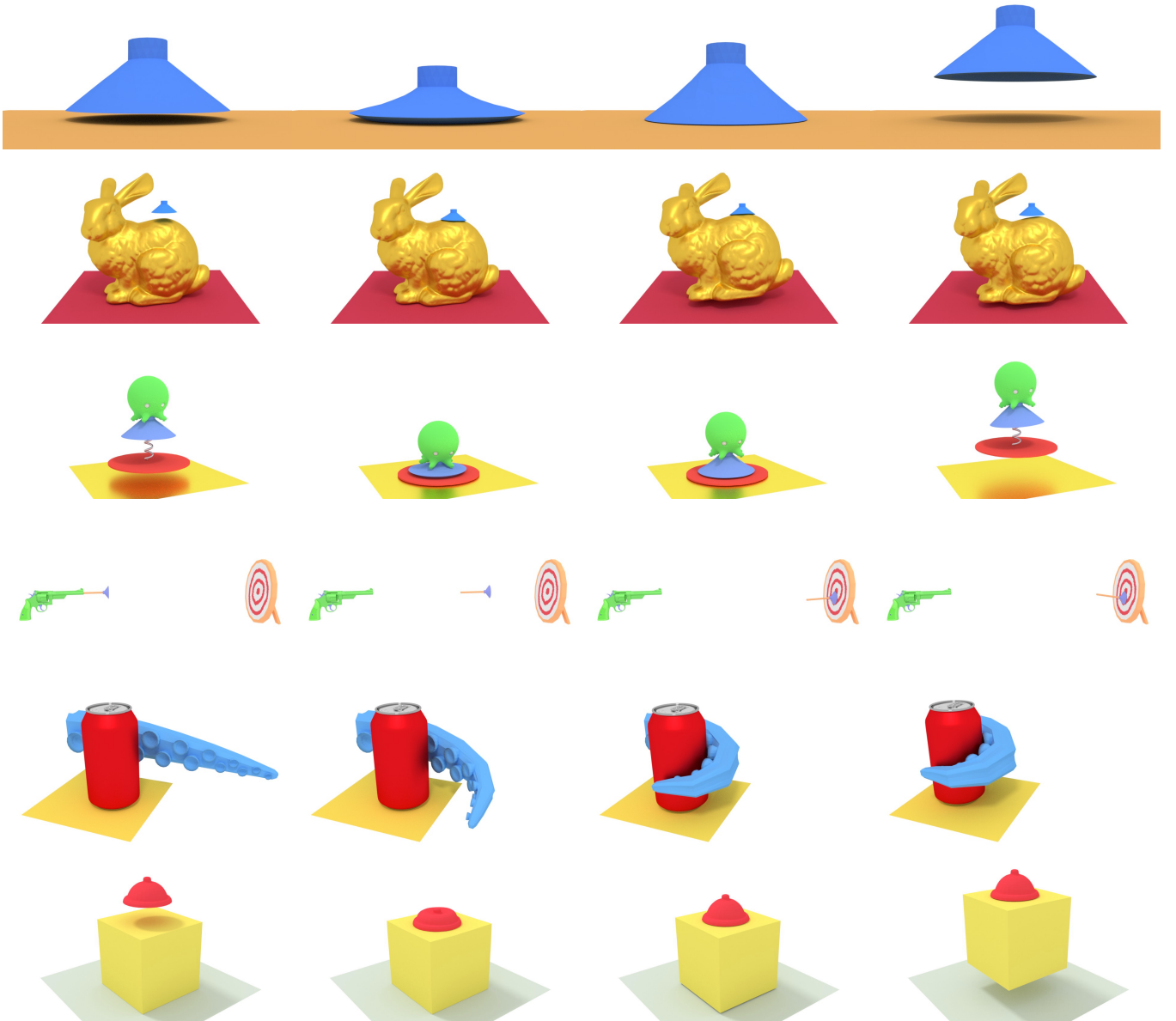
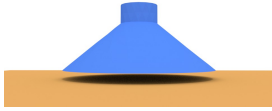


Figure 4.5 – Animation, from left to right, of our different simulation scenarios illustrating our suction model that we described in Table 4.1.



Static Plane: The suction cup is pushed onto a static plane and then pulled out vertically. The suction cup first sticks on the plane before being released when the pressure constraints are not strong enough to compensate for the elasticity and pulling forces.



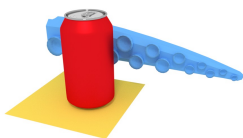
Bunny: The suction cup is used to lift a rigid bunny. This scenario is more complex than the cube scenario since the bunny surface is not planar. The cup manages to slightly lift the bunny by suction, but it finally drops. The air-tightness of the cavity is weak due to the curved and bumpy surface.



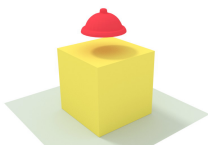
Toy This scenario involves a jumping toy that combines a spring and a suction cup. The toy has a solid base (red) attached to a suction cup (blue) by a spring. The spring is physically simulated with a finite element model using a wrapped deformable model [NKJF09] composed of hexaedra. A rigid monster head is placed above the suction cup and can be pushed, sticking the suction cup on the base. Pressure forces and elasticity forces coming from the spring are then applied to the suction cup. With the right spring stiffness, the toy finally jumps into the air after a short amount of time.



Dart This scenario consists of a dart gun shooting a dart (with a suction cup at the tip) at a target. Note that the dart is a deformable model, but the elasticity of the rod has been made rigid. When the suction cup hits the target, we can observe an oscillating motion of the rigid rod driven by the elastic deformation of the suction cup.



Octopus Arm. This scenario is composed of an octopus arm with suction cups of varying sizes. The arm can be manipulated to grasp a can. The scenario illustrates the ability of our approach to detect multiple cavities because the arm is composed of multiple cavities. The arm is driven by user using pulling cables. We use a Young modulus of 150 kPa for the arm in this example.



Deformable Cube In this last scenario, the suction cup is used to lift a deformable cube. The cube is modeled as an elastic FEM model, has side length 115 mm, a mass of 30 g, and it is very soft with a Young modulus is 10 kPa. The suction cup deforms the cube by pushing on it, then deforms in turn and finally lifts the cube.

Table 4.1 – Illustrative scenarios of our suction model.

4.3 Computation time performance

We performed timings for simulations running on a laptop with an Intel Core i7-7820HQ CPU at 2.90 GHz, 32GB of DDR4 system memory, and a NVIDIA Quadro P3000 Mobile graphics card. Below, in Table 4.2, we respectively report timings for the above scenarios (Table 4.1).

Scenario	Nb of vertices	Nb of tets	Avg. Nb of constraints	Avg. time per frame	Free motion	Collision detection	Cavity detection	Constraint sys. construction	Constraint sys. resolution
static plane	576	1685	2406.41	451.39	0.51 (0.23%)	8.31 (3.68%)	33.18 (22.05%)	31.57 (13.99%)	86.13 (57.24%)
rigid bunny	576	1685	2707.56	678.19	0.58 (0.17%)	5.65 (1.67%)	2.58 (0.76%)	97.24 (28.68%)	224.54 (66.22%)
static sphere	576	1685	4527.12	827.68	0.51 (0.12%)	7.84 (1.90%)	1.49 (0.54%)	69.63 (16.82%)	215.19 (78.00%)
rigid cube	576	1685	3164.74	910.04	0.58 (0.13%)	12.53 (2.75%)	56.72 (12.46%)	104.33 (22.93%)	273.71 (60.15%)
octopus arm*	731	1935	878.01	300.15	6.24 (4.16%)	6.47 (4.94%)	22.98 (30.62%)	24.78 (18.01%)	59.50 (39.65%)
toy*	576	1685	1444.31	477.74	27.60 (11.56%)	4.81 (3.67%)	20.92 (8.76%)	15.11 (10.95%)	149.05 (62.40%)
deformable cube*	576	1685	3361.09	1498.57	8.60 (1.15%)	4.89 (0.98%)	36.39 (4.86%)	125.72 (16.78%)	364.42 (72.95%)
dart*	723	2002	2686.25	1133.17	28.09 (4.96%)	13.71 (2.54%)	69.16 (12.21%)	52.00 (9.47%)	393.53 (69.46%)

Table 4.2 – Computation time performance of the different scenarios. Average computation time per frame is reported in milliseconds as well as the average number of constraints and the detailed computation times among the different components of the simulation. For each measurement we additionally indicate in parentheses the portion of total simulation time as a percentage. The scenarios which are marked with an asterisk "*" do not use GPU asynchronous preconditioners for stability reasons (they involve larger deformations than the others).

Furthermore, we ran the "rigid cube" scenario with different mesh resolutions and compared time-performance as shown below, in Table 4.3.

Nb of nodes	Nb of tetrahedra	Nb of constraints	Avg. time per frame	Free motion	Collision detection	Cavity detection	Constraint system construction	Constraint system resolution
76	205	244.88	11.66	0.10 (1.71%)	0.88 (15.04%)	0.41 (10.55%)	0.81 (13.88%)	0.87 (22.25%)
125	372	452.93	52.95	0.14 (0.54%)	3.56 (13.46%)	8.49 (48.09%)	2.34 (8.84%)	3.33 (18.87%)
141	413	623.56	75.94	0.16 (0.41%)	4.07 (10.73%)	11.56 (45.66%)	3.51 (9.26%)	6.59 (26.03%)
151	449	642.34	75.48	0.17 (0.45%)	3.96 (10.48%)	11.29 (44.86%)	3.68 (9.75%)	6.80 (27.03%)
159	470	830.41	104.65	0.18 (0.35%)	4.55 (8.70%)	14.76 (42.30%)	4.92 (9.40%)	11.61 (33.27%)
203	613	996.07	132.82	0.23 (0.35%)	5.03 (7.58%)	17.39 (39.27%)	6.94 (10.45%)	16.41 (37.07%)
363	1078	2294.55	486.40	0.36 (0.15%)	9.00 (3.71%)	45.15 (27.85%)	26.37 (10.84%)	88.23 (54.42%)
415	1241	2584.75	603.13	0.39 (0.13%)	10.50 (3.49%)	53.29 (26.50%)	32.11 (10.65%)	113.98 (56.70%)
455	1378	2837.21	703.27	0.44 (0.13%)	11.54 (3.29%)	60.05 (25.62%)	39.68 (11.29%)	134.81 (57.51%)
502	1472	3133.83	839.36	0.48 (0.11%)	12.66 (3.02%)	69.39 (24.80%)	48.05 (11.45%)	163.78 (58.54%)
552	1598	3323.07	896.17	0.51 (0.11%)	13.33 (2.98%)	71.87 (24.06%)	51.57 (11.51%)	177.27 (59.34%)
576	1685	3450.77	993.11	0.55 (0.11%)	14.14 (2.84%)	78.33 (23.66%)	55.08 (11.10%)	199.68 (60.32%)

Table 4.3 – Performance of the cube grasping scenario with different mesh resolutions for the suction cup. The average computation time for a time step is listed in milliseconds and for each component of the algorithm we additionally indicate between brackets the portion of total simulation time as a percentage.

Measured performances

Table 4.2 summarizes the computation time performance of our approach for different scenarios. We measured the total time needed to achieve the entire simulation with respect to the complexity of the

scenario. In addition to the total time, we report the average times needed per frame for the computation of the free-motion configuration, the collision detection, as well as the times needed more specifically for simulating the suction cup phenomenon: the cavity detection, the constraint system construction (including the pressure constraints, the friction constraints, the contact constraints) and the constraint resolution. The time units are in milliseconds and for each measure we additionally indicate in parentheses the coverage percentage over the time-step simulation. For all scenarios, we used a time-stepping value of 10 ms. The maximum number of Gauss-Seidel iterations was clamped to 50. All the results were acquired using averages onto a sample of 150 time-step iterations, starting from the beginning of the simulation.

Note that the scenarios in Table 4.2 that have an asterisk next to their name are those for which we do not use the GPU asynchronous preconditioners. This is because we observe that the asynchronous behavior leads to a loss of precision in the deformations, and produces instabilities in these scenarios. Note also that the toy and octopus arm scenarios both have a low time-performance average despite the large number of nodes involved. This is because there is a longer period without contact at the start of these simulations in comparison to the other scenarios.

Different mesh resolutions: We ran the cube scenario with several resolutions in order to illustrate how the different computation times evolve as a function of the resolution of the suction cup. The results are shown in Table 4.3. As expected, the total time increases with the mesh resolution. The times of all the components of the simulation are impacted by the mesh resolution, since the number of vertices is involved in both the flood-fill procedures of the cavity detection and the constraint system construction and resolution.

In order to confirm that the time-performance is primarily impacted by the number of constraints, we collected the number of constraints at each frame and the corresponding time-performance at the scale of a time-step. The results are shown in Figure 4.6. We displayed the dots using a gradient of color from black to blue which respectively indicates if a dot has been collected early or lately in the simulation. Note that the computation time stays low even for a large number of constraints when there is no contact at the very beginning of the simulation (i.e., when vertices fall below the alarm distance, but before constraint forces are needed to prevent them from passing the contact surface). That is why we can see on the graph that the blackest dots form a horizontal line on the graph. Those have been collected when the suction cup is close from the ground without touching it. Most importantly, we observe that the blue dots on top of the graph trace a line which show that the time-performance of the simulation seems almost linearly proportional to the number of constraints within the range of the number of constraints we used for our tests. It should be

noted that the low number of iterations of Gauss-Seidel (without relying on a precision criterion to stop the iterations) artificially reduces the cost for systems with many constraints. Indeed, the theoretical cost of a Gauss-Seidel type algorithm executed entirely sequentially is quadratic.

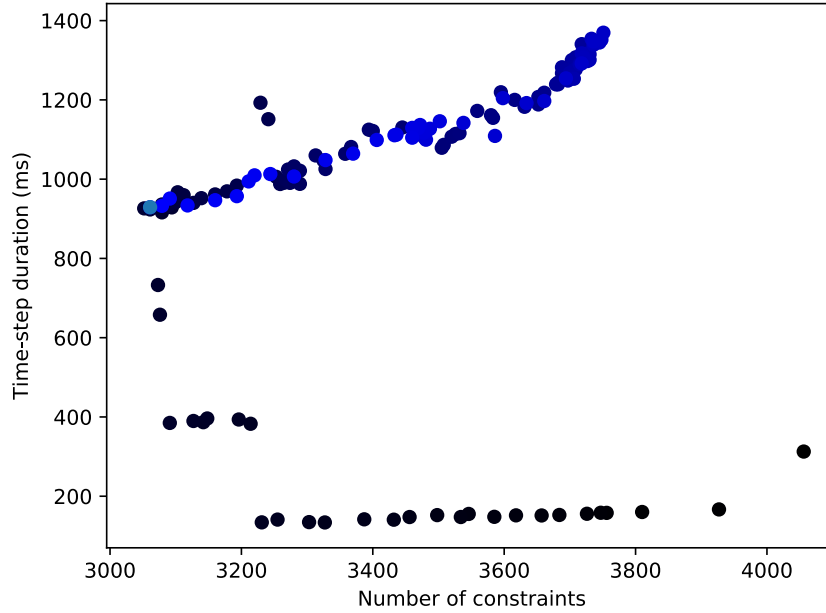


Figure 4.6 – Computation time performance in function of the number of constraints. The blue dots on top of the graph trace a line. The black dots on bottom of the graph denote simulation steps where constraints are generated but contact has not yet formed (i.e., they have trivial solutions).

4.4 Conclusion

In conclusion of this chapter, we briefly presented the SOFA framework that we used in our research and we provided implementation details regarding both our cavity detection and the constraint resolution. Secondly, we established some illustrative scenarios (simulations) for which we measured time-performance and we tested different mesh resolutions. As expected, we observed that the more the resolution of the meshes is high, the more we potentially have contact points meaning frictional contact constraints that must be solved, leading to a time-performance drop. To go further in terms of tests, we performed a validation of our model through physical experiments and additional simulations (virtual experiments) that we detail in Chapter 5.

VALIDATION

Contents

5.1 Experiments for validating active suction	81
5.1.1 Geometry validation	82
5.1.2 Force validation	83
5.2 Experiments for validating passive suction	86
5.2.1 Forces and curvatures	86
5.2.2 Virtual experiments	90
5.3 Conclusion	95

We listed illustrative scenarios in Chapter 4 that were also used for time-performance benchmarking. But other additional scenarios, called virtual experiments, were implemented to test the abilities of the suction model. In addition, we did several physical experiments to ensure our model is close from reality, notably in terms of forces and geometry. Those physical and virtual experiments seek to validate our simulation model. In this chapter, we first describe the physical experiments related to our suction model in the context of active suction. And we then describe the experiments which take place in the context of passive suction.

5.1 Experiments for validating active suction

The goal of the two first experiments was to evaluate the *active suction* version of our suction model. Both experiments constitute evaluations based on comparisons with real data. One of these experiments aims to validate the geometry of the suction cups that we obtained after deformation with our model, whereas the other experiment seeks to validate the forces computed from our simulations. The two experiments are presented in the following subsections and in this order.

We designed two different types of silicone suction cups. We cast 3D printed suction cups, using respectively Dragon Skin A20 silicon (Young modulus = 843 kPa) and Dragon Skin A30 silicon (Young

modulus = 1067 kPa). In the following, we will refer to the names A20 and A30 for the two suction cups. We generated the meshes using the CAD models built for 3D printing (see Figure 5.1).

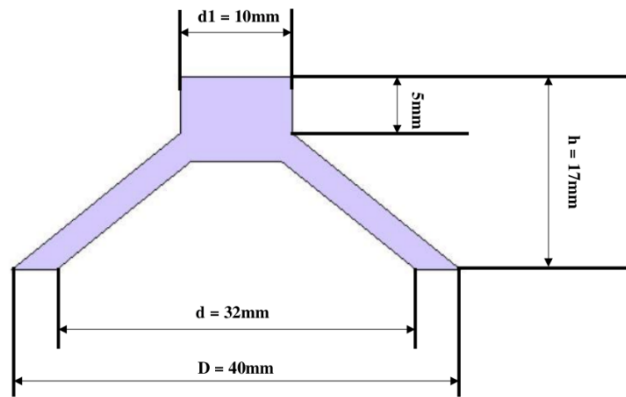


Figure 5.1 – Dimensions of our two suction cups A20 and A30.

5.1.1 Geometry validation

The objective of the geometry validation experiment with active suction was to compare the shape of the real suction cup with the shape of the virtual one, in order to validate the geometry under a static approach. We designed an experimental setup wherein the suction cup is positioned on a flat surface, on top of a small hole which is connected to a vacuum pump via a tube. The pressure inside the suction cup's cavity is controlled (active suction) using a pressure regulator and an associated numeric pressure sensor both placed along the tube (Figure 5.2). In this way, we were able to control the shape of the cup by gently varying the internal pressure through the regulator. We applied an internal depressurization of -11 kPa and -6 kPa respectively to the suction cups A30 and A20 and we recorded the initial shapes of the real suction cups as well as the final ones.

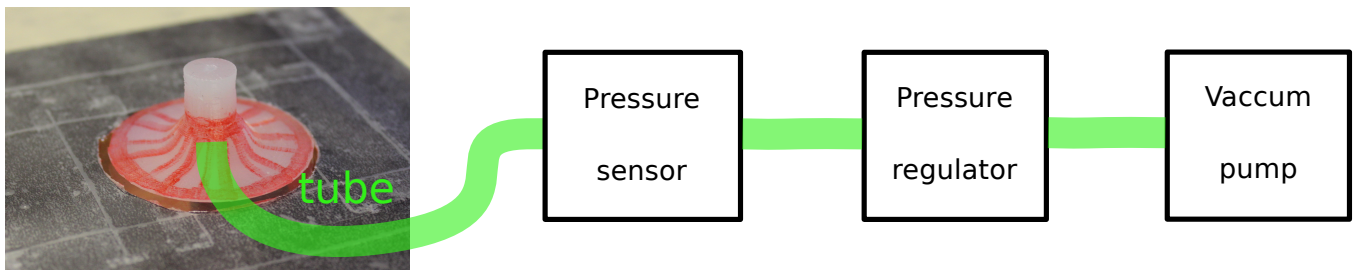


Figure 5.2 – The cavity of the suction cup is connected to a vacuum pump via a tube. The user can control the internal pressure using a pressure regulator and an associated numeric pressure sensor both placed along the tube.



Figure 5.3 – Photogrammetry reconstruction of the real suction cup.

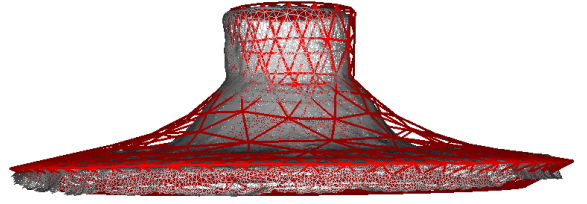


Figure 5.4 – Shape comparison between the real suction cup (Grey) and the virtual one (red) at their final states.

We extracted the shapes of the real suction cups using a photogrammetry procedure. For that purpose, we used the Meshroom software (<https://alicevision.github.io/>) to reconstruct the 3D shapes from pictures we took during the experiment (see Figure 5.3). Once the 3D real data were reconstructed, we could compare them with the resulting shapes from our simulation.

In order to avoid the noise resulting from the photogrammetry reconstruction, we applied smoothing and filtering operations afterwards. As illustrated in Figure 5.4, we compared the real suction cup with the virtual one using the Hausdorff distance. This metric is simply the greatest of all the distances from a vertex in the virtual mesh to the closest vertex in the real mesh. In a nutshell, it measures the error between our simulations and reality. When comparing the final states, we obtained the following measurements: for the suction cup A20, we measured a Hausdorff distance of 1.59 mm. For the suction cup A30, the Hausdorff distance was 3.29 mm.

These measures are not so close to zero which means that the virtual shape coming from simulation and the photogrammetry-reconstructed shape does not match perfectly together. Intuitively, when we look at Figure 5.4, we observe that the walls of the two cups are not aligned so it would mean the error essentially comes from there. But in fact, the error potentially come from different factors: noise in the photogrammetry-reconstruction, approximation dues to the linearization of the deformations in our simulations, resolution of the meshes.

5.1.2 Force validation

The objective of the force experiment in the context of active suction was to measure the intensity of the pull force required to detach the suction cup from the surface, assuming it was completely stuck thanks to

a vacuum. As illustrated in Figure 5.5, we attached the top of the suction cup to an actuator with a hook. In addition, a force sensor was positioned between the tip of the actuator and the hook. In this setup, we consequently were able to measure the pulling force. The actuator had only one degree of freedom, which means it can exclusively move vertically. As for the geometry experiment (Section 5.1.1), we connected the cavity of the suction cup to the vacuum and were thus able to regulate the internal pressure.

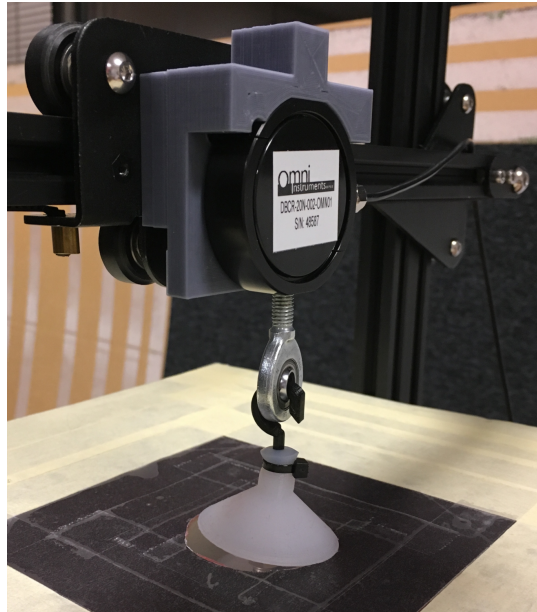


Figure 5.5 – Experimental setup for the force measurements. The suction cup is attached to a force sensor. When it is positioned on a flat surface, its cavity is linked to a vacuum pump with a regulator in-between.

We ran the experiments with both suction cups A20 and A30. For each one, as illustrated in Figure 5.6, we proceeded as it follows: first of all, thanks to the actuator, the suction cup was positioned on top of the hole without pushing it too much against the flat surface. After that, the vacuum was switched on and we set the cavity depressurization to a fixed value (either -5 kPa, -10 kPa, -15 kPa or -20 kPa) using the pressure regulator, achieving a deformation of the suction cup. To finish, we pulled the suction cup by moving the actuator up at a speed of 5 mm/s.

Regarding this experiment, we reproduced a similar scenario in the simulation. We were then able to compare the force variation as a function of time and to detect the necessary pull force to detach the suction cup from the flat surface. The forces comparison between the virtual one and the ground truth to reach detachment is provided in Figure 5.7 with suction cup A20 and different internal pressures.

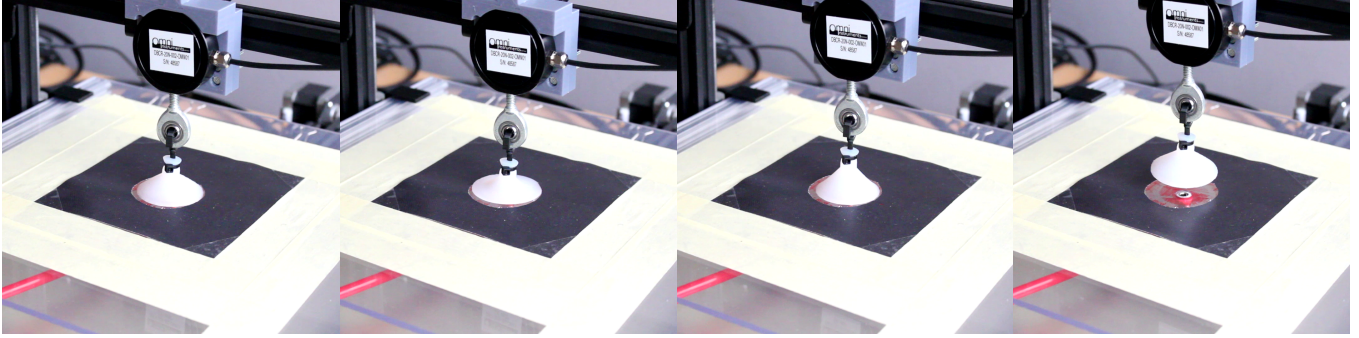


Figure 5.6 – Workflow of the force experiment (active suction). From left to right: first, the suction cup is positioned on a flat surface and on top of the hole which is linked to a vacuum. Secondly, we use the pressure regulator to set the desired pressure in the cavity, leading to a distortion of the material. Thirdly, the actuator goes up to pull the cup. Finally, the suction cup detaches.

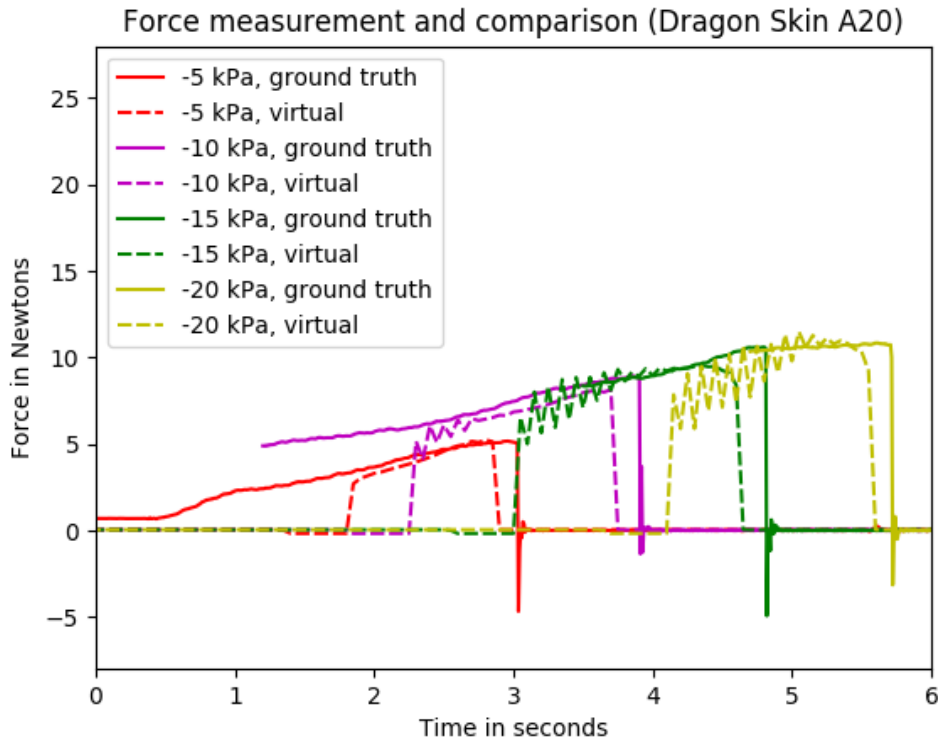


Figure 5.7 – Force measurement from real data and simulations (called virtual) for different depressurizations. The force increases because the actuator pulls the suction to the top. The force drops to zero when the suction cup detaches from a flat surface. Simulation forces are represented with dashed lines. A time offset of 0.2 s has been voluntarily introduced between the curves from real data and simulation for better readability.

According to the results (Figure 5.7), we notice that the shapes of the curves representing real data and the ones representing our simulations tend to be globally the same. However, we can see there are discontinuities in the time-variation of the vertical forces in the context of the simulation. The reason is that the computed inner borders of the cavity were just the approximated inner borders because the computation of the exact inner borders (Section 3.2.3) were not yet implemented at the time we plotted these curves. Therefore, there were discontinuities in the time-variation of the cavity volume, and thus in the time-variation of the internal pressure too, leading to vibrations of the suction cup, especially when the pressure becomes high.

5.2 Experiments for validating passive suction

After finishing the experiments with active suction, we decided to extend the validation by making more experiments but now in the context of passive suction. More precisely, we made a new force experiment similar to the one presented in Section 5.1.2 and we additionally performed several virtual experiments, namely simulation scenarios which aim to test the abilities of our suction model. We present these experiments in the following subsections.

5.2.1 Forces and curvatures

As an extend of the force validation experiment with active suction, we decided to make another similar second experiment but in the context of passive suction. As illustrated in Figure 5.8, the setup of this one relies on a robotic arm (CB3 UR3 from Universal Robots) which push and pull passive suction cups (vertical motion only) against surfaces with different degrees of curvatures. In practice, we used plastic pipes with different diameters that we fixed onto the workbench just under the robot actuator. The diameters of the pipes that we used are 40 mm, 50 mm, 63 mm and 101 mm. In addition of running the experiment by varying the curvatures, we also ran it with different suction cups in terms of material elasticity and dimensions. We used three silicon materials having different elasticities to cast the suction cups: Dragon Skin A10, Dragon Skin A20, and Dragon Skin A30. We actually made four suction cups: three small ones made of A10, A20, A30 respectively and a bigger one made of A20. They however all have the same wall thickness as well as the same stem's diameter. We detail their properties in Table 5.1.

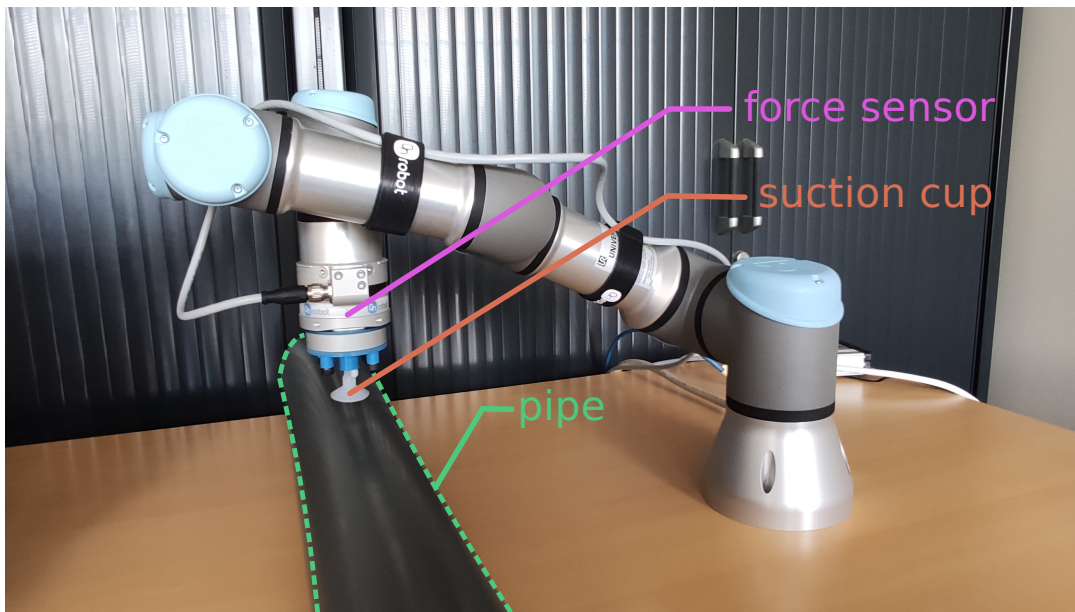


Figure 5.8 – Setup of our force experiment in the context of passive suction: the cup is attached at the tip of a robot arm and is pushed against a pipe (curved surface) and then pulled. The necessary detachment force is measured thanks to a force sensor near the actuator.

Cup ID	S10	S20	S30	B20
Size	small	small	small	big
Material	A10	A20	A30	A20
Young modulus	667 kPa	843 kPa	1067 kPa	843 kPa
Mass	2.4 g	2.5 g	2.6 g	4.5 g

	Small	Big
Inner diameter	24 mm	38 mm
Cavity's height	4.5 mm	10 mm
Cavity's volume	1195 mm ³	6063 mm ³
Wall thickness	4 mm	4 mm
Stem's diameter	10 mm	10 mm

Table 5.1 – Properties of the different suction cups from our force experiment related to passive suction.



Figure 5.9 – We cast 4 models of suction cups for the force experiment in the passive suction case.

At each run of the experiment, as illustrated in Figure 5.10, the actuator goes down at a speed of 10 mm/s until the suction cup is completely flattened (i.e. stuck) against the pipe. After that, it takes a short break to let the suction cup reaching an equilibrium state and it then goes up at the same speed than the descending phase in order to detach the suction cup. The vertical forces are measured over time thanks to a native force sensor located at the level of the robot's actuator.

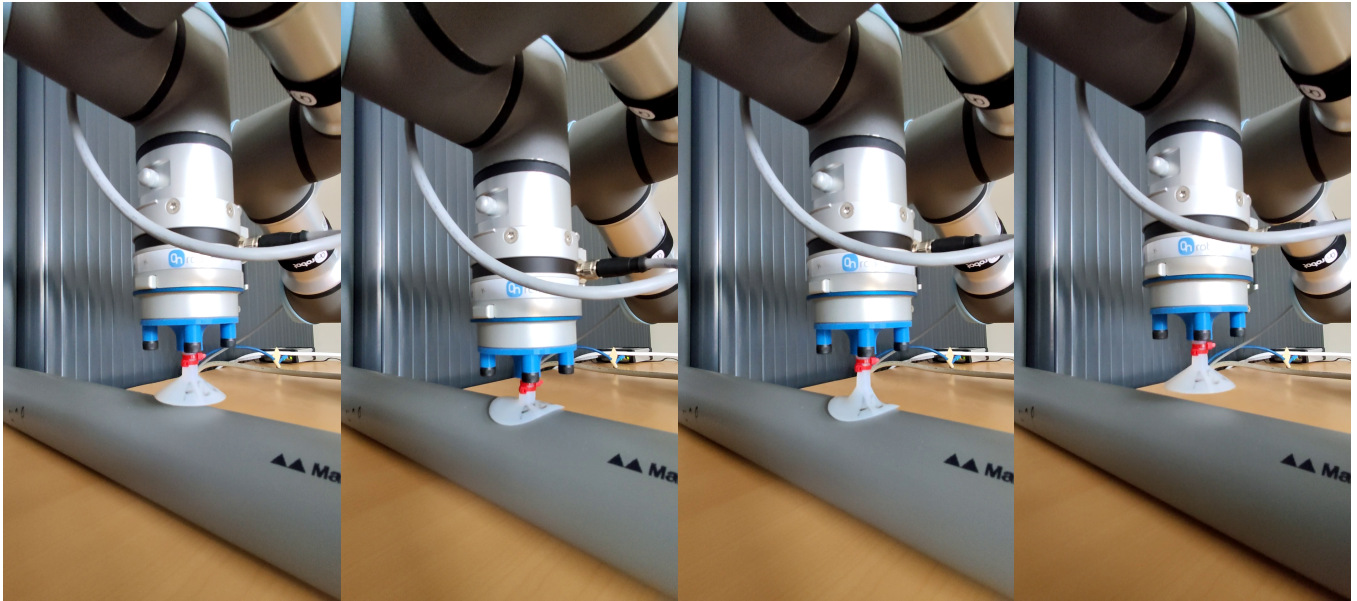


Figure 5.10 – Workflow of the force experiment with different curvatures (passive suction). From left to right: first, the actuator goes down. Secondly, the suction cup becomes completely flattened against the pipe, and the actuator takes a short break at this moment. Thirdly, it goes up to pull the suction cup. Finally, the suction cup ends up to detach.

The force measurements are given in Figure 5.11. Generally, we observe that the signed vertical force goes into negative values due to the fact we push the cup against the pipe. And then, the force goes into positive value because we pull the cup whereas the suction effect resists to the detachment. In the end, the force returns to zero when the cup has been detached.

We plan to compare the results with those from our simulations. Unfortunately, it has not been done yet due to some difficulties about keeping the tracking of the cavity in our simulation. In fact, knowing our cavity detection is based on a flood-fill procedure (Section 3.2.2), the vertices located on the bottom part of the cup are all considered as contact points when the cavity is completely flattened onto the surface, meaning no vertex is classified as being inside the cavity. And as a result, the cavity detection is lost. This case raises a limitation that we need to overpass. Indeed, in reality, when the suction cup is completely

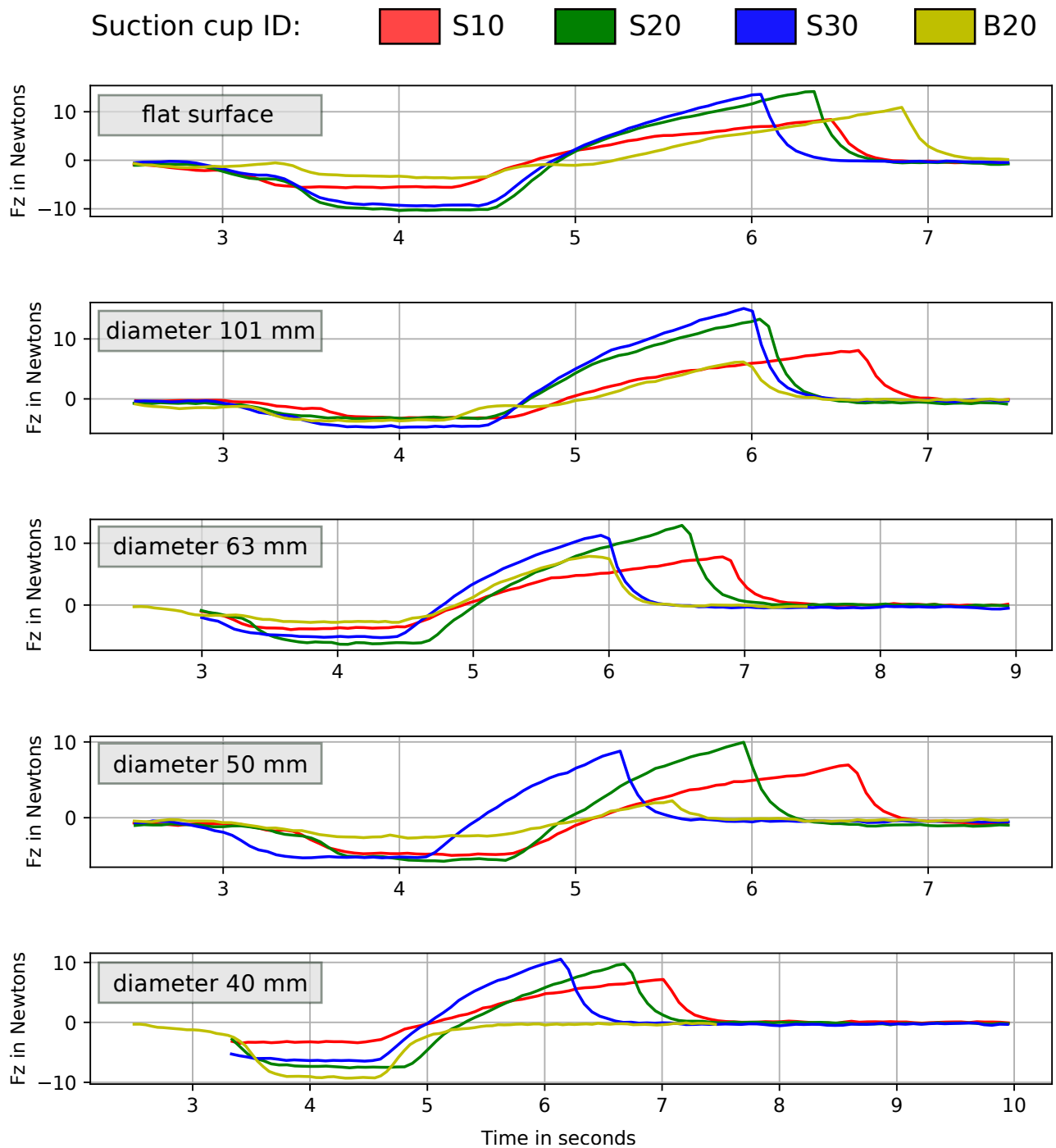


Figure 5.11 – Force experiment with passive suction: measurement of vertical forces over time with different curvatures (pipes with different diameters). The colors represent the IDs of the suction cups (see Table 5.1).

flattened, it remains numerous tiny air cavities along the contact surface that allows the suction cup to stay stuck. However, we do not capture them in our simulation because the resolution of the mesh is not high enough. Assuming we increase the mesh resolution, it would be likewise necessary to manage the merges and splits of the air cavities over time. Otherwise we would be not able to correctly track the pressure values inside the cavities. This discussion reveals a new feature we should implement into our cavity detection algorithm as future work.

Although the comparison between the virtual simulations and real data is still an ongoing work, we have learned several lessons through this experiment. First, we noticed the same trends about the influence of the curvature of the surface on which we push the suction cup, namely the more the degree of curvature is important, the less the suction effect is strong. Furthermore, we made one virtual experiment related to this aspect (presented in Section 5.2.2) wherein we try to lift an object by varying the curvature of its shape. Secondly, the boundary conditions related to the location on which we attach the suction cup has a meaningful impact on the result.

5.2.2 Virtual experiments

In order to test the abilities of the suction model, we created some appropriate simulation scenarios that we executed using different parameters. These are described in the following.

Lifting of curved objects: Figure 5.12 shows an experiment in which we vary the curved shaped of an object until the suction cup is no longer able to lift the object. We used a cylinder (diameter: 120 mm) with different non-uniform scales to produce test surfaces, using scale factors from 0.2 up to 1 along the vertical axis. The suction cup is able to lift by suction all the flattened versions of the cylinder, but not the original cylinder. In the high curvature case of the original cylinder, the suction cup only lifts the object briefly, at which point it swings slightly, breaking the seal, and the object falls. This behavior tends to be the same in reality whereas we did not perform a complete validation about this.

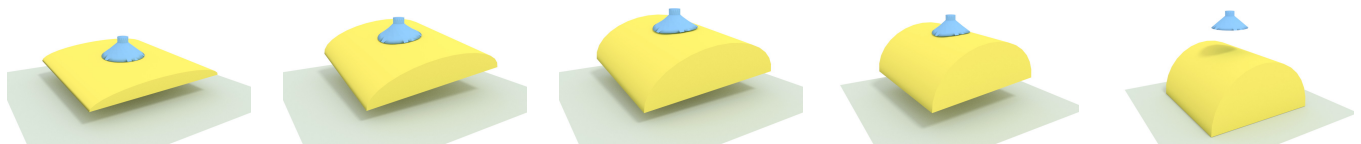


Figure 5.12 – The suction cup is able to lift objects with curved surfaces by suction, but fails when the curvature is too high. From left to right: we increase the curvature of the object.

Cube lifting with different shapes: The scenario consists in trying to lift a deformable cube by suction using different shapes of suction cups (Figure 5.13). We experimented three different shapes:

- Red: a cup with a round border, like a plunger, with the diameter of its cavity being 7 mm at the lowest and 51 mm at the highest (inner border). The diameter of its outside border is 55 mm and the thickness of the rim is 2 mm. The height of the cavity is 21 mm.
- Blue: a cup with straight borders, where the diameter of its cavity is 8 mm at the lowest and 32 mm at the highest (inner border). The diameter of its outside border is 40 mm, and the thickness of the rim is 4 mm. The height of the cavity is 9 mm.
- Green: a cup with an asymmetric shape, where the longest Euclidean distance between two points of its inner border is 57 mm. The longest euclidean distance between two points on its outer border is 60.5 mm. The diameter of the cavity at the top is 4.5 mm, the thickness of the rim is 2 mm, and the height of the cavity is 8 mm.

The red and blue suction cups are able to lift the cube, but the green one cannot. Because of its unusual shape, a well-sealed cavity cannot form due to its borders which rise when the cup is pressed against the cube. This behavior is obviously the same in reality.

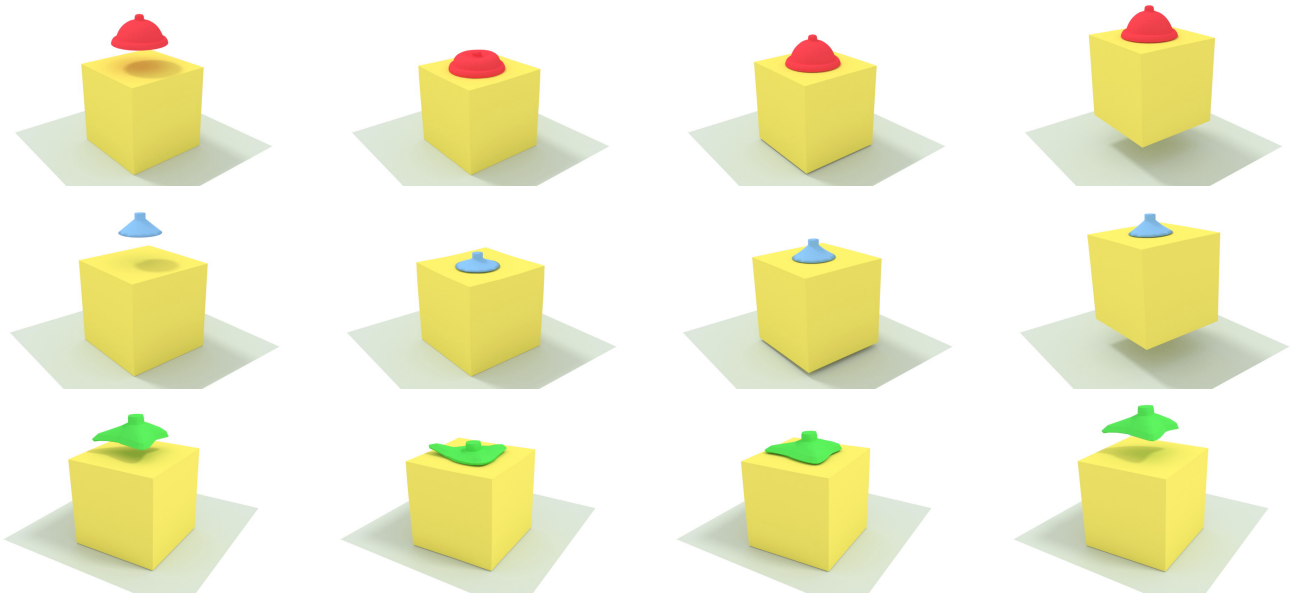


Figure 5.13 – We attempt to lift a deformable cube with 3 different suction cup models. Left-to-right shows the progression of the 3 benchmark scenarios. The red one and the blue cups succeed, whereas the shape of the green cup does not allow it to make a large sealed cavity because its borders rise when it is pushed onto the cube.

Cube lifting with different masses: The first experiment consists in grasping a rigid cube by suction force. We vary the mass of the cube at each run of the experiment while collecting the variation of the vertical pull force along time. We plot in Figure 5.14 the time-variation of the vertical force for each mass. The purpose is to determine how much weight a suction cup is able to grasp. In this experiment, we used two different suction cups:

- Small: the diameter of its cavity is 8 mm at the lowest and 32 mm at the highest (inner border), the diameter of the outside border is 40 mm. The thickness of the rim is 4 mm. The height of the cavity is 9 mm.
- Large: the diameter of its cavity is 12 mm at the lowest and 48 mm at the highest (inner border), the diameter of the outside border is 69 mm. The thickness of the rim is 10.5 mm. The height of the cavity is 13.5 mm.

They both have the same mass (30 g) and the same elastic properties, namely 338 kPa for the Young modulus.

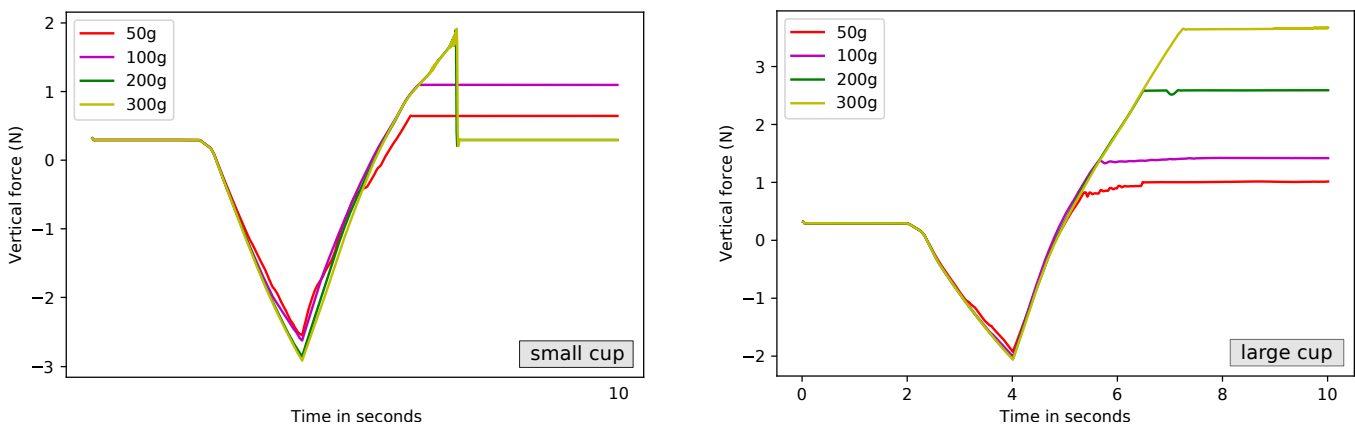


Figure 5.14 – Benchmark involving different masses where two suction cups of different sizes are pressed against a cube (2-4 s), and then slowly lifted (4-6 s). Left: the small suction cup is able to lift 50 g and 100 g, but fails for masses 200 g and above. Right: the large suction cup has a larger cavity which allows it to lift heavier objects.

In Figure 5.14, from 0 s to 2 s, the cup goes down and ends up touching the cube. The force is not zero during this time due to the mass of the cup. From 2 s to 4 s, the cup continues to go down while gently deforming due to contact with the cube. During this motion, the force applied in the vertical axis becomes negative because of the counteracting elastic force of the suction cup.

The force continues to decrease until we stop pushing the cup. From 4 s, the ascending phase begins and continues until the end of the simulation. Here, we observe that the force increases to the point that it supports the weight of both the cup and the cube knowing the latter is lifted by suction. Finally, the scenario has one of two possible endings according to the physical properties. In all but one case, the cup is able to lift the cube and we observe an approximately constant force from the moment the cube lifts off the ground. In the case of the small cup and the heaviest cube, the cup is not able to maintain a seal and it detaches, and that is why we see that the force returns to its initial value.

In Figure 5.14, we observe that the small suction cup (top plot) is able to grasp by suction 50 g and 100 g, but not 200 g, while the larger cup (bottom plot) succeeds with all masses in the experiment. It can be explained by the fact the larger suction cup has a larger cavity, meaning the internal pressure acts on a larger area, and can also go lower due to its larger volume. As already mentioned in Section 5.2.1, we also noticed during the experiments that this is not possible to obtain an absolute void in the cavity. The latter still contains some air molecules.

Air tunnel: This scenario involves a deformable cube which includes a tunnel connecting two holes on its top surface. With this scenario, we demonstrate that cavity detection is able to handle the tunnel as a closed cavity when we place the suction cup so as to simultaneously cover the two holes. In contrast, when the suction cup only covers one hole, no cavity is formed as we correctly identify that the tunnel communicates between the cavity of the cup and the outside. Figure 5.15 shows the result.

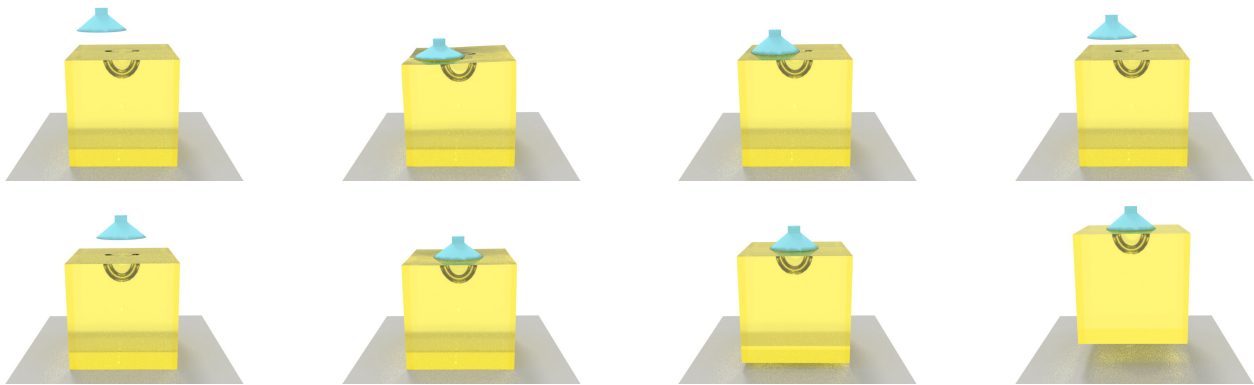


Figure 5.15 – In this experiment, a deformable cube includes a tunnel connecting two holes. Left-to-right shows the progression of the 2 benchmark scenarios. If the suction cup is pushed down onto only one hole, the cavity detection algorithm correctly identifies the air passage and an airtight cavity is not formed. In contrast, we can lift the cube by suction when the rim of the suction cup covers both holes. In this case the tunnel is part of the cavity.

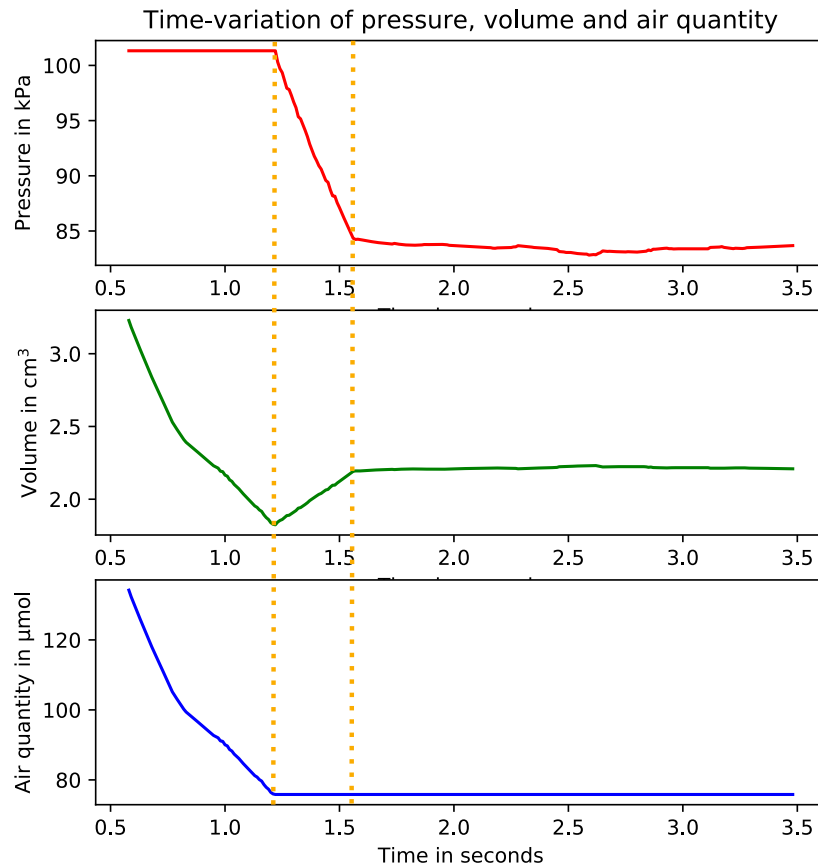


Figure 5.16 – Time-variation of the mechanical values of the cavity during the scenario wherein a rigid cube is lifted by the suction cup. The timeline is split in 3 time periods: pushing the cup onto the targeted object, pulling it while the object is still on the ground, and finally lifting the object assuming it is not too heavy and that the cavity stays airtight.

Ideal gas law: The purpose of this experiment is to run the scenario wherein the rigid cube is grasped by suction in order to observe the time-variation of the air pressure, the volume, and the air quantity inside the detected cavity. The results of this experiment are shown in Figure 5.16. To explain the variation of the curves, we split the graph into three time periods using dotted orange vertical lines. In the first time period, the suction cup is pushed onto the cube and the curves in the graph start once the cavity has been detected. The volume of the cavity decreases, as well as the air quantity because our pressure constraint lets air escapes from the cavity when the pressure is clamped to P_{max} . In the second time period, we begin to pull on the suction cup. Here, the cavity is airtight, which can be seen by the constant quantity of air, while the

volume of the cavity increases due to deformation of the suction cup. The consequence is a depressurization in the cavity to satisfy the gas law. In the third time period, the cube is lifted by suction. The cavity is still airtight, and we observe a constant quantity of air (the cup is not completely flattened to keep track of the cavity). The pressure and volume are likewise almost constant, but show some variation as there are some dynamic deformations of the suction cup as the cube swings gently when it is raised into the air.

Regarding this virtual experiment, we think it would be very interesting to measure the time-variation of the pressure inside cavity in a physical experiment. The latter could be performed using an experimental setup similar to the one from the force experiment (Section 5.2.1) but with a pressure sensor. In this way, we will be able to compare the results of the virtual experiment with virtual data.

5.3 Conclusion

In conclusion to this chapter, we made multiple experiments in order to validate the suction model. We learned some lessons through these, notably the fact we need to improve the cavity detection algorithm to capture the remaining air molecules along the contact surface when the cup is completely flattened. Furthermore, it would be very interesting to measure the pressure inside cavity over time in a physical experiment and compare measured data with the results of the virtual experiment related to the ideal gas law (end of Section 5.2.2). We think further experiments related to friction and air pressure measurements over time would be very interesting to complete our model validation.

CONCLUSION

6.1 Summary of the contributions and future work

For the last three years, our research on the simulation of the suction phenomenon led to several contributions.

Modeling: In Chapter 3, we proposed a new suction model composed of two important features: a cavity detection algorithm and a novel air pressure constraint to simulate suction. In terms of modeling, we decided to solely represent uniform air pressure distributions because, although air dynamics has an influence on the pressure distribution, we thought it can be neglected in our context to get better time-performance. In terms of numerical methods, the object deformations are achieved through the Co-rotational Finite Element Methods as a trade-off between time-performance and accuracy. We have proposed and implemented a cavity detection algorithm which is able to handle multiple cavities meaning we can simulate suction objects beyond traditional suction cups. Moreover, the algorithm computes exact inner borders involving dynamic topological changes to obtain a clear discretization of the detected air cavities. It allows us to obtain an accurate geometrically-computed volume of the cavity without discontinuities over time, regardless of the granularity of the meshes. In order to simulate suction, we decided to introduce an air pressure constraint that we coupled with classical contact and friction constraints. These constraints are gathered to form a Non-linear Complementarity Problem (NCP) which is built using Lagrange multipliers and solved thanks to a Projected Gauss-Seidel solver. The suction can be simulated as active suction (the internal pressure is directly controlled by the user) or passive suction (case of traditional suction cups). In that latter case, the pressure values inside cavities are found thanks to a resolution based on the ideal gas law.

Simulation: In Chapter 4, we explained that we implemented the above contributions in C++ using SOFA, a specific framework for deformable objects. We have designed several simulation scenarios to illustrate our suction model. In addition, we measured time-performance in each of these scenarios using different mesh resolutions. We mentioned that we use an asynchronous GPU-implemented preconditioner based on a sparse LDL factorization to accelerate performance. As expected, the higher the resolution, the

more we have contact and friction constraints, but we did not observe any important drop in the computation time-performance.

Validation: In order to validate our model, we made some experiments as described in Chapter 5. In the context of active suction, we achieved a geometry experiment and a force experiment whereas in the context of passive suction, we also made a force experiment plus some virtual experiments to test the abilities of our model. The geometry experiment with active suction compares the material configuration of the suction cup between our virtual simulations and reality using a photogrammetry-reconstructed mesh. The measurements (Hausdorff distance) reveals that the two shapes do not match perfectly together. The force experiment in active suction consisted in measuring the necessary pull force to detach the suction cup once this one is stucked to a flat surface. The results showed that the forces in our simulations and in reality globally have the same trends. The force experiment with passive suction is similar to the one with active suction except that we test the suction onto surfaces having different curvatures. We figured out through this experiment that the cavity detection algorithm has some limitations which prevent the comparison of the measures with those from our simulations. However, we noticed the same trends about the influence of the curvature of the surface. About the virtual experiments with passive suction, we performed cube lifting attempts (by suction) with different masses while measuring the vertical force. We observe that, at a certain threshold in terms of mass, the suction is not able to lift the cube anymore, which is obviously the same behavior in reality. Still in the context of cube lifting, we tested different shapes of suction cups. As expected, one suction cup failed to lift the cube because of its unusual shape for which its borders rise when the cup is pushed against the cube, leading to a not-sealed cavity. Another virtual experiment consisted in lifting a half cylinder this time, instead of the cube. We varied the degree of curvature of the surface at each run by scaling the half cylinder on one axis. In this experiment, the suction cup ends to dropping the object when its curvature is too high. Whereas we did not compare this virtual experiment with reality, we observed the same trends than in reality. Next, we performed a virtual experiment involving a tunnel connecting two holes on a top surface of the cube. The experiment demonstrated that our cavity detection algorithm is able to handle the tunnel as a closed cavity when the cup lays onto the two holes. To finish, a last virtual experiment traced the time-variation of the mechanical values inside the cavity, namely the pressure, volume and air quantity. The curves revealed an accordance regarding the ideal gas law.

6.2 Short-term possible improvements

On further consideration, it would be interesting to improve our cavity detection algorithm by implanting a new feature which manage the merges and splits of the air cavities over time. In this way, the suction

model would be able to correctly track over time the numerous tiny cavities which may arise when the cup is completely flattened. However, this technique would work provided that the resolution of the collision mesh is high enough. Knowing a high resolution mesh means numerous constraints and consequently lower computation time-performance, one optimization may be to dynamically adapt its resolution according to the geometries and the locations of the detected cavities for instance. Besides, it would also be very interesting to adapt the friction coefficient with regards to the rugosity of the surface on which the cup eventually slip. Regarding the Imagine project that we briefly presented in Chapter 1, this aspect would be especially relevant knowing the cup (located at the tip of the robot actuator) may be lead to lift electronic cards by suction, knowing these constitute noisy surfaces which could be considered as rough surfaces.

Another point is that it would be necessary to find a way to improve time-performance when the number of contacts points becomes high. One solution could be, for instance, to gather some contact points which are close together. In addition, we could optimize computation-time performances of the constraint solver.

Regarding model validation, we think it would be interesting to go further with additional experiments especially in the passive suction case. For instance, we could do an experiment wherein we measure the pressure inside the cavity over time and compare the values with the results from our simulations. Furthermore, measuring the impact of friction regarding the suction effect would be also a very interesting experiment. Besides, we only tested suction onto flat or curved surfaces for now, so another idea of experiments would be to test suction onto surfaces with different levels of rugosity. We could even let the suction cup slide along these surfaces and measure friction as well as the sealing reliability.

6.3 Long-term future work and applications

We believe our suction model will be of great interest in robotics, especially with the emerging soft robots. It could be used as a tool to design robots which are equipped of suction cups. Such a tool would be useful to decide the shape of the cups, their positions, their rheological properties, and so on. The concerned robots could use suction to grasp so in this case, the objective in terms of design would be to optimize this task. But we can also imagine soft robots using suction as a means of locomotion. These robots could be inspired by animals like the octopus or even the snail (Figure 1.1). Finally, we think our suction model will let emerge more sophisticated robots by exploiting suction the best way.

BIBLIOGRAPHY

- [ACF⁺07] J. Allard, S. Cotin, F. Faure, P-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni. Sofa - an open source framework for medical simulation. *Studies in Health Technology and Informatics*, 2007.
- [AFC⁺10] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry. Volume contact constraints at arbitrary resolution. In *Proc. of ACM SIGGRAPH 2010 papers*, pages 1–10, 2010.
- [Ben02] F. Benedetti. Physical response to collision between deformable objects. *Postgraduate course on graphics visualization and comm.*, EPFL, VRLab, 2002.
- [BJ05] J. Barbič and D. L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Transactions on Graphics (TOG)*, 24(3):982–990, 2005.
- [BLN96] B. Bahr, Y. Li, and M. Najafi. Design and suction cup analysis of a wall climbing robot. *Computers & Electrical Engineering*, 22(3):193–209, 1996.
- [BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH Computer Graphics and Interactive Techniques*, pages 43–54. ACM, 1998.
- [CADC10] H. Courtecuisse, J. Allard, C. Duriez, and S. Cotin. Asynchronous preconditioners for efficient solving of non-linear deformations. In *Proc. of Virtual Reality Interactions and Physical Simulations (VRIPHYS)*, pages 59–68, 2010.
- [Dur13] C. Duriez. Real-time haptic simulation of medical procedures involving deformations and device-tissue interactions, 2013.
- [Ebe08] D Eberly. Triangulation by ear clipping. *Documentation on GeometricTools.com*, 2008.
- [FAFB08] F. Faure, J. Allard, F. Falipou, and S. Barbier. Image-based collision detection and response between arbitrary volumetric objects. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2008.
- [FPRJ00] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proc. of Computer graphics and Interactive Techniques*, pages 249–254, 2000.
- [GMS⁺15] D. Ge, T. Matsuno, Y. Sun, C. Ren, Y. Tang, and S. Ma. Quantitative study on the attachment and detachment of a passive suction cup. *Vacuum*, 116:13–20, 2015.

-
- [HSO03] K. K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Proc. of Graphics Interface (GI)*, volume 3, pages 16–17, 2003.
- [HTG03] B. Heidelberger, M. Teschner, and M. Gross. Real-time volumetric intersections of deforming objects. In *Proc. of Vision, Modeling, and Visualization (VMV)*, pages 461–468. AKA, 2003.
- [HY18] B. Hu and H. Yu. Optimal design and simulation of a microsuction cup integrated with a valveless piezoelectric pump for robotics. *Shock and Vibration*, 2018, 2018.
- [JAJ98] F. Jourdan, P. Alart, and M. Jean. A gauss-seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, 155(1-2):31–47, 1998.
- [JF03] D. L. James and K. Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (TOG)*, 22(3):879–887, 2003.
- [JP02] D. L. James and D. K. Pai. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 582–585, 2002.
- [JP04] D. L. James and D. K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. In *Proc. of ACM SIGGRAPH 2004 Papers*, pages 393–398, 2004.
- [Kim20] T. Kim. A finite element formulation of baraff-witkin cloth. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2020.
- [KNO14] W. Koh, R. Narain, and J. F. O'Brien. View-dependent adaptive cloth simulation. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2014.
- [KZ03] J. Klein and G. Zachmann. Adb-trees: controlling the error of time-critical collision detection. In *Proc. of Vision, Modeling, and Visualisation (VMV)*, pages 37–45, 2003.
- [LG03] R. I. Leine and C. Glocker. A set-valued force law for spatial coulomb-contensou friction. *European Journal of Mechanics - A/Solids*, 22(2):193–216, 2003.
- [LTBY06] J. Liu, K. Tanaka, L. M. Bao, and I. Yamaura. Analytical modelling of suction cups used for window-cleaning robots. *Vacuum*, 80(6):593–598, 2006.
- [MDM⁺02] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 49–54, 2002.
- [MG04] M. Müller and M. Gross. Interactive virtual materials. In *Proc. of Graphics Interface (GI)*, pages 239–246, 2004.

-
- [MHTG87] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 1987.
- [MKN⁺04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 141–151, 2004.
- [MML⁺18] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.
- [Mor66] J-J. Moreau. Quadratic programming in mechanics: dynamics of one-sided constraints. *SIAM Journal on Control*, 4(1):153–158, 1966.
- [MWN⁺17] P-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M-P. Cani. Adaptive physically-based models in computer graphics. *Computer Graphics Forum*, 36(6):312–337, 2017.
- [NKJF09] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure. Preserving topology and elasticity for embedded deformable models. *ACM Transactions on Graphics (TOG)*, pages 1–9, 2009.
- [NMK⁺06] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum*, 2006.
- [NO17] S. Nishita and H. Onoe. Liquid-filled flexible micro suction-controller array for enhanced robotic object manipulation. *IEEE Journal of Microelectromechanical Systems*, 26(2):366–375, 2017.
- [OTSG09] M. A. Otaduy, R. Tamstorf, D. Steinemann, and M. Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum*, 28(2):559–568, 2009.
- [Pro97] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Proc. of Computer Animation and Simulation*, pages 177–189. Springer, 1997.
- [PW89] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. In *Proc. of ACM SIGGRAPH Computer Graphics and Interactive Techniques*, pages 215–222, 1989.
- [Ree83] W. T. Reeves. Particle Systems A Technique for Modeling a Class of Fuzzy Objects. *ACM Transactions on Graphics (TOG)*, 1983.
- [RP17] M. A. Robertson and J. Paik. New soft robots really suck: vacuum-powered systems empower diverse capabilities. *Science Robotics*, 2(9), 2017.

-
- [Sam88] H. Samet. An overview of quadtrees, octrees, and related hierarchical data structures. In *Theoretical Foundations of Computer Graphics and CAD*, pages 51–68. Springer, 1988.
- [SF91] M. Shinya and M-C. Forgue. Interference detection through rasterization. *The Journal of Visualization and Computer Animation*, 2(4):132–134, 1991.
- [SHGSA⁺20] A. Suárez-Hernández, T. Gaugry, J. Segovia-Aguas, A. Bernardin, C. Torras, M. Marchal, and G. Alenyà. Leveraging multiple environments for learning and decision making: a dismantling use case. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [SOM04] A. Sud, M. A. Otaduy, and D. Manocha. Difi: Fast 3d distance field computation using graphics hardware. *Computer Graphics Forum*, 23(3):557–566, 2004.
- [Sta97] J. Stam. Stochastic dynamics: simulating the effects of turbulence on flexible structures. *Computer Graphics Forum*, 16:C159–C164, 1997.
- [TFPM15] F. Tramacere, M. Follador, N. M. Pugno, and B. Mazzolai. Octopus-like suction cups: from natural to artificial solutions. *Bioinspiration & Biomimetics*, 10(3):035004, 2015.
- [TKH⁺05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, et al. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, 2005.
- [TMD015] A. Talvas, M. Marchal, C. Duriez, and M. A. Otaduy. Aggregate constraints for virtual manipulation with soft fingers. *IEEE Transactions on Visualization and Computer Graphics*, 2015.
- [TMOT12] M. Tang, D. Manocha, M. A. Otaduy, and R. Tong. Continuous penalty forces. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.
- [TN87] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. In *Proc. of ACM SIGGRAPH Computer Graphics and Interactive Techniques*, pages 153–162, 1987.
- [TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of ACM SIGGRAPH Computer Graphics and Interactive Techniques*, pages 205–214, 1987.
- [TPS08] B. Thomaszewski, S. Pabst, and W. Straßer. Asynchronous cloth simulation. In *Proc. of Computer Graphics International (CGI)*, volume 2, page 2, 2008.
- [WObR11] H. Wang, J. F. O ’brien, and R. Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. *ACM Transactions on Graphics (TOG)*, 30(4), 2011.

-
- [WRK⁺10] M. Wicke, D. Ritchie, Bryan M. K., S. Burke, J. R. Shewchuk, and James F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics (TOG)*, 29(4):1–11, 2010.
- [XDA⁺20] Z. Xie, A. G. Domel, N. An, C. Green, Z. Gong, T. Wang, E. M. Knubben, J. C. Weaver, K. Bertoldi, and L. Wen. Octopus arm-inspired tapered soft actuators with suckers for improved grasping. *Soft Robotics*, 2020.
- [XZK⁺14] L. Xin, W. Zhong, T. Kagawa, H. Liu, and G. Tao. Development of a pneumatic sucker for gripping workpieces with rough surface. *IEEE Transactions on Automation Science and Engineering*, 13(2):639–646, 2014.
- [YWH⁺09] H. Yan, Z. Wang, J. He, X. Chen, C. Wang, and Q. Peng. Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds*, 20(2-3):417–426, 2009.
- [ZC01] C. Zhang and T. Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *Proc. of International Conference on Image Processing*, volume 3, pages 935–938, 2001.

LIST OF FIGURES

1.1	Examples of animals using suction as a mean of locomotion: octopus on the top image and snail on the bottom image.	13
1.2	Examples of common objects made with suction cups: tea towel hooks, suction cup unblocker, care cups, dart gun, pop up spring toys, glass suction cups.	14
1.3	Examples of robots using suction cups: vaccum grippers, wall-climbing robots [RP17], soft pneumatic actuators [XDA ⁺ 20].	15
1.4	Left: gripper robot from the Imagine project and its actuator equipped with a suction cup (image of courtesy from Karlsruhe Institute of Technology). Right: illustration of a physics-based simulation using our physics-based model of suction cup: the suction cup at the end of a gripper is used to dismantle a hard drive [SHGSA ⁺ 20].	16
2.1	Continuum mechanics: a deformable object moves from its initial configuration to a deformed configuration.	24
2.2	Large rotational deformations yields to inflation artifacts when not using Co-rotational Finite Element Method [MG04].	27
2.3	Reduced deformation models [NMK ⁺ 06]: (a) reference shape x_0 , (b) displacement field U_1 , (c) displacement field U_2 , and (d) one possible deformed shape $x = x_0 + U_1 + 0.5U_2$	28
2.4	Mass spring system [NMK ⁺ 06].	31
2.5	Different spatial structures, here represented in 2D: (a) uniform grid, (b) kD-tree, (c) BSP-tree.	37
2.6	It exists numerous types of Bounding Volumes [TKH ⁺ 05].	39
2.7	Happy Buddha [TKH ⁺ 05] and three color-mapped slices coming of its distance field. Blue indicates close distances while red indicates medium distances.	40
2.8	Illustration of the Coulomb friction cone [Dur13].	44

3.1	Illustration of the suction phenomenon through a scenario in the context of passive suction: (1) The suction cup is positioned on the floor. The red arrows represent the internal pressure while the blue ones represent the atmospheric pressure. Both pressure values are the same at this moment. (2) We push the cup onto the ground. Some air escapes. (3) The cup tends to an equilibrium state. The atmospheric pressure is stronger than the internal pressure. The dark grey arrows represent the elastic forces. (4) We pull the suction cup thus increasing the volume of the cavity and decreasing the internal pressure accordingly.	50
3.2	Workflow of the cavity detection. Left: the contact surface is determined from the narrow-phase collision response (red). Middle: inner surfaces (green) and approximate inner borders (dashed blue) are identified thanks to flood-fill operations. Right: exact inner borders (continuous blue line) are computed using new vertices that are added to the mesh, and inner surface regions are paired to constitute the unified geometric cavities.	53
3.3	Collision distances. All vertices below the alarm distance (dashed line) generate contact constraints. The contact distance (dotted line) delimits a gap between the two colliding objects which must be satisfied after constraint resolution. We use a sealing threshold (continuous blue line) to identify the vertices of the contact surface. The intersection of the sealing distance with the inner surface of the cavity delimits the exact inner border.	54
3.4	Illustrations of the vertex classification stages before inner border refinement. The illustrated surface is a sample of the closed surface of an arbitrary suction object. The green area represents the exact contact area. Left: The contact points are classified first. Middle: Vertices outside cavity are flood-filled starting from a user-defined seed vertex. Right: Remaining vertices are flood-filled to be classified as points inside cavities and the approximated inner borders are revealed.	55
3.5	Illustration of the procedure consisting in dynamically adapting the topology of the colliding mesh after inner border projection. From left to right: 1) Points along the exact inner border (blue) are projected onto the colliding object. 2) Projected points (red) are linked and intersection points with existing edges are added (Grey). 3) Each original triangle is split in two polygons through the projected inner border. These polygons are triangulated (green) thanks to the well-known ear clipping algorithm.	58
3.6	Two topological representations of a <i>suction object</i> . Left: FEM mesh composed of tetrahedral elements. Right: Collision mesh composed with triangles along the object surface. . .	61
3.7	When a cavity exists due to collision, three types of rows exist in the constraint matrix: non-penetration constraint directions (red), friction constraint directions (two per contact, blue), and the uniform negative pressure distribution in the cavity (green).	62

4.1	Simplified UML class diagram about vertex classification.	71
4.2	Simplified UML class diagram about how we handle dynamic topology in order to compute the exact inner border of the cavities.	72
4.3	Simplified UML class diagram about cavity linkage.	73
4.4	Simplified UML class diagram about constraint implementation.	74
4.5	Animation, from left to right, of our different simulation scenarios illustrating our suction model that we described in Table 4.1.	75
4.6	Computation time performance in function of the number of constraints. The blue dots on top of the graph trace a line. The black dots on bottom of the graph denote simulation steps where constraints are generated but contact has not yet formed (i.e., they have trivial solutions).	79
5.1	Dimensions of our two suction cups A20 and A30.	82
5.2	The cavity of the suction cup is connected to a vacuum pump via a tube. The user can control the internal pressure using a pressure regulator and an associated numeric pressure sensor both placed along the tube.	82
5.3	Photogrammetry reconstruction of the real suction cup.	83
5.4	Shape comparison between the real suction cup (Grey) and the virtual one (red) at their final states.	83
5.5	Experimental setup for the force measurements. The suction cup is attached to a force sensor. When it is positioned on a flat surface, its cavity is linked to a vacuum pump with a regulator in-between.	84
5.6	Workflow of the force experiment (active suction). From left to right: first, the suction cup is positioned on a flat surface and on top of the hole which is linked to a vacuum. Secondly, we use the pressure regulator to set the desired pressure in the cavity, leading to a distortion of the material. Thirdly, the actuator goes up to pull the cup. Finally, the suction cup detaches.	85
5.7	Force measurement from real data and simulations (called virtual) for different depressurizations. The force increases because the actuator pulls the suction to the top. The force drops to zero when the suction cup detaches from a flat surface. Simulation forces are represented with dashed lines. A time offset of 0.2 s has been voluntarily introduced between the curves from real data and simulation for better readability.	85
5.8	Setup of our force experiment in the context of passive suction: the cup is attached at the tip of a robot arm and is pushed against a pipe (curved surface) and then pulled. The necessary detachment force is measured thanks to a force sensor near the actuator.	87
5.9	We cast 4 models of suction cups for the force experiment in the passive suction case.	87

5.10	Workflow of the force experiment with different curvatures (passive suction). From left to right: first, the actuator goes down. Secondly, the suction cup becomes completely flattened against the pipe, and the actuator takes a short break at this moment. Thirdly, it goes up to pull the suction cup. Finally, the suction cup ends up to detach.	88
5.11	Force experiment with passive suction: measurement of vertical forces over time with different curvatures (pipes with different diameters). The colors represent the IDs of the suction cups (see Table 5.1).	89
5.12	The suction cup is able to lift objects with curved surfaces by suction, but fails when the curvature is too high. From left to right: we increase the curvature of the object.	90
5.13	We attempt to lift a deformable cube with 3 different suction cup models. Left-to-right shows the progression of the 3 benchmark scenarios. The red one and the blue cups succeed, whereas the shape of the green cup does not allow it to make a large sealed cavity because its borders rise when it is pushed onto the cube.	91
5.14	Benchmark involving different masses where two suction cups of different sizes are pressed against a cube (2-4 s), and then slowly lifted (4-6 s). Left: the small suction cup is able to lift 50 g and 100 g, but fails for masses 200 g and above. Right: the large suction cup has a larger cavity which allows it to lift heavier objects.	92
5.15	In this experiment, a deformable cube includes a tunnel connecting two holes. Left-to-right shows the progression of the 2 benchmark scenarios. If the suction cup is pushed down onto only one hole, the cavity detection algorithm correctly identify the air passage and an airtight cavity is not formed. In contrast, we can lift the cube by suction when the rim of the suction cup covers both holes. In this case the tunnel is part of the cavity.	93
5.16	Time-variation of the mechanical values of the cavity during the scenario wherein a rigid cube is lifted by the suction cup. The timeline is split in 3 time periods: pushing the cup onto the targeted object, pulling it while the object is still on the ground, and finally lifting the object assuming it is not too heavy and that the cavity stays airtight.	94

LIST OF TABLES

2.1	Comparison between the state-of-the-art spatial discretization methods.	35
4.1	Illustrative scenarios of our suction model.	76
4.2	Computation time performance of the different scenarios. Average computation time per frame is reported in milliseconds as well as the average number of constraints and the detailed computation times among the different components of the simulation. For each measurement we additionally indicate in parentheses the portion of total simulation time as a percentage. The scenarios which are marked with an asterisk "*" do not use GPU asynchronous preconditioners for stability reasons (they involve larger deformations than the others).	77
4.3	Performance of the cube grasping scenario with different mesh resolutions for the suction cup. The average computation time for a time step is listed in milliseconds and for each component of the algorithm we additionally indicate between brackets the portion of total simulation time as a percentage.	77
5.1	Properties of the different suction cups from our force experiment related to passive suction.	87

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:

Simulation Interactive basée Physique du Phénomène de Succion

Nom Prénom de l'auteur : BERNARDIN ANTONIN

Membres du jury :

- Monsieur JAILLET Fabrice
- Monsieur OTADUY Miguel
- Madame MARCHAL Maud
- Monsieur ARNALDI BRUNO
- Monsieur KRY Paul G.
- Monsieur ANDREWS Sheldon
- Monsieur DURIEZ Christian

Président du jury : ARNALDI Bruno

Date de la soutenance : 18 Janvier 2021

Reproduction de la these soutenue

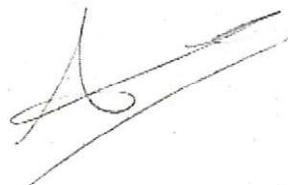
- ☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 18 Janvier 2021.

Signature du président de jury

Le Directeur,


Abdellatif MIRAOU



Titre : Simulation Interactive basée Physique du Phénomène de Suction

Mot clés : Suction, Contact, Déformable, Simulation Physique, Manipulation, Robotique

Résumé : Cette thèse porte spécifiquement sur la modélisation ainsi que la simulation du phénomène de suction en raison de ses vertus très intéressantes vis-à-vis de la manipulation, surtout dans le domaine de la robotique. Au sein de nos recherches, nous faisons la distinction entre la suction active (pression d'air contrôlée directement par une pompe à air par exemple) et la suction passive (cas d'une ventouse traditionnelle). En termes de contributions, nous avons proposé un modèle de suction composé de deux composants principaux : un algorithme de détection de cavités et une contrainte de pression d'air. L'algorithme de détection de cavités permet de localiser et identifier les géométries des poches d'air qui surviennent lorsqu'une ventouse (ou objet similaire) rentre en contact avec un autre objet. Quant à la contrainte de pression d'air, celle-ci est couplée avec les contraintes traditionnelles de contact et de frottement. Afin d'accé-

lérer la simulation, la distribution de la pression d'air à l'intérieur des cavités est représentée au sein de notre modèle sans pour autant simuler le comportement complexe des dynamiques du fluide d'air. Au delà de la conception de notre modèle de suction, nous avons réalisé plusieurs expérimentations physiques et virtuelles : Nous avons mesuré les forces nécessaires pour détacher des ventouses dans le contexte de la suction active puis de la suction passive. Nous avons lancé ces expérimentations en utilisant plusieurs élasticités de matériaux au niveau de la ventouse. Par rapport à la suction passive, nous avons en plus fait varier la courbure de la surface contre laquelle la ventouse est appuyée. En outre, nous avons mis en place plusieurs scénarios illustratifs pour mesurer les performances de calcul, et aussi des scénarios de test pour analyser les capacités de notre modèle de suction.

Title: Interactive Physically-based Simulation of the Suction Phenomenon

Keywords: Suction, Contact, Deformable, Physics simulation, Manipulation, Robotics

Abstract: This thesis focus on the modeling and physics simulation of the suction phenomenon in regards to its very interesting features related to manipulation, especially in the robotics context. In our research, we distinguish active suction (air pressure value is directly controlled by a vacuum pump for instance) from passive suction (case of a traditional suction cup). In terms of contributions, we proposed a suction model composed of two main features: a cavity detection algorithm and a novel air pressure constraint. The cavity detection algorithm allows to locate and identify the geometries of air cavities which may arise when a suction cup (or similar object) collides with another object. As for the air pressure constraint, it is coupled with the traditional contact

and friction constraints. In order to speed up the simulation, the air pressure distribution inside the cavities is represented in our model without simulating the complex behavior of air dynamics. In addition our elaborated suction model, we made several physical and virtual experiments: We measure the necessary forces to detach suction cups in the context of active and passive suction. We ran these experiments using different elasticities regarding the suction cup material. About passive suction, we additionally varied the degrees of curvature of the surface onto which the suction cup is pushed. Furthermore, we performed different illustrative to test time-performances and also benchmarking scenarios to test the abilities of our suction model.

