

ROBOTIC MANIPULATION OF CLOTH

Mechanical Modeling and Perception

FRANCO COLTRARO



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Doctoral Programme in Applied Mathematics
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

SUPERVISORS:

Jaume Amorós

Maria Alberich-Carramiñana

February 2023

Franco Coltraro: *Robotic Manipulation of Cloth*, Mechanical Modeling and Perception, © February 2023

A thesis submitted to the Universitat Politècnica de Catalunya for the degree of Doctor of Philosophy

DOCTORAL PROGRAMME:
Applied Mathematics

LOCATION:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Barcelona, Spain

FUNDING:
This project was developed in the context of the project CLOTHILDE ("CLOTH manipulation Learning from DEMonstrations") which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 741930) and was also supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656).

SUPERVISORS:
Jaume Amorós
Maria Alberich-Carramiñana

A mi Nonna.

ABSTRACT

In this work, we study various mathematical problems arising from the robotic manipulation of cloth. First, we develop a locking-free continuous model for the physical simulation of inextensible textiles. We present a novel *finite element* discretization of our inextensibility constraints which results in a unified treatment of triangle and quadrilateral meshings of the cloth. Next, we explain how to incorporate contacts, self-collisions and friction into the equations of motion, so that frictional forces and inextensibility and collision constraints may be integrated implicitly and without any decoupling. We develop an efficient *active-set* solver tailored to our non-linear problem which takes into account past active constraints to accelerate the resolution of unresolved contacts and moreover can be initialized from any non-necessarily feasible point. Then, we embark ourselves on the empirical validation of the developed model. We record in a laboratory setting –with depth cameras and motion capture systems– the motions of seven types of textiles (including e.g. cotton, denim and polyester) of various sizes and at different speeds and end up with more than 80 recordings. The scenarios considered are all dynamic and involve rapid shaking and twisting of the textiles, collisions with frictional objects and even strong hits with a long stick. We then, compare the recorded textiles with the simulations given by our inextensible model and find that on average the mean error is of the order of 1 cm even for the largest sizes (DIN A2) and the most challenging scenarios.

Furthermore, we also tackle other relevant problems to robotic cloth manipulation such as cloth perception and classification of its states. We present a reconstruction algorithm based on Morse theory that proceeds directly from a point-cloud to obtain a cellular decomposition of a surface with or without boundary: the results are a piecewise parametrization of the cloth surface as a union of Morse cells. From the cellular decomposition, the topology of the surface can be then deduced immediately. Finally, we study the configuration space of a piece of cloth: since the original state of a piece of cloth is flat, the set of possible states under the inextensible assumption is the set of developable surfaces isometric to a fixed one. We prove that a generic simple, closed, piecewise regular curve in space can be the boundary of only finitely many developable surfaces with nonvanishing mean curvature. Inspired by this result we introduce the dGLI cloth coordinates, a low-dimensional representation of the state of a piece of cloth based on a directional derivative of the Gauss Linking Integral. These coordinates –computed from the position of the cloth’s boundary– allow us to distinguish key qualitative changes in folding sequences.

RESUMEN

En este trabajo estudiamos varios problemas matemáticos relacionados con la manipulación robótica de textiles. En primer lugar, desarrollamos un modelo continuo libre de *locking* para la simulación física de textiles inextensibles. Presentamos una novedosa discretización usando *elementos finitos* de nuestras restricciones de inextensibilidad resultando en un tratamiento unificado de mallados triangulares y cuadrangulares de la tela. A continuación, explicamos cómo incorporar contactos, autocolisiones y fricción en las ecuaciones de movimiento, de modo que las fuerzas de fricción y las restricciones de inextensibilidad y colisiones puedan integrarse implícitamente y sin ningún desacoplamiento. Desarrollamos un *solver* de tipo *conjunto-activo* adaptado a nuestro problema no lineal que tiene en cuenta las restricciones activas pasadas para acelerar la resolución de nuevos contactos y, además, puede inicializarse desde cualquier punto no necesariamente factible. Posteriormente, nos embarcamos en la validación empírica del modelo desarrollado. Grabamos en un entorno de laboratorio -con cámaras de profundidad y sistemas de captura de movimiento- los movimientos de siete tipos de textiles (entre los que se incluyen, por ejemplo, algodón, tela vaquera y poliéster) de varios tamaños y a diferentes velocidades, terminando con más de 80 grabaciones. Los escenarios considerados son todos dinámicos e implican sacudidas y torsiones rápidas de los textiles, colisiones con obstáculos e incluso golpes con una varilla cilíndrica. Finalmente, comparamos las grabaciones con las simulaciones dadas por nuestro modelo inextensible, y encontramos que, de media, el error es del orden de 1 cm incluso para las telas más grandes (DIN A2) y los escenarios más complicados.

Además, también abordamos otros problemas relevantes para la manipulación robótica de telas, como son la percepción y la clasificación de sus estados. Presentamos un algoritmo de reconstrucción basado en la teoría de Morse que partiendo directamente de una nube de puntos obtiene una descomposición celular de una superficie con o sin borde: los resultados son una parametrización a trozos de la superficie de la tela como una unión de celdas de Morse. A partir de la descomposición celular puede deducirse inmediatamente la topología de la superficie. Por último, estudiamos el espacio de configuración de un trozo de tela: dado que el estado original de la tela es plano, el conjunto de estados posibles bajo la hipótesis de inextensibilidad es el conjunto de superficies desarrollables isométricas a una fija. Demostramos que una curva genérica simple, cerrada y regular a trozos en el espacio puede ser el borde de un número finito de superficies desarrollables con curvatura media no nula. Inspirándonos en este

resultado, introducimos las coordenadas dGLI, una representación de dimensión baja del estado de un pedazo de tela basada en una derivada direccional de la integral de enlazamiento de Gauss. Estas coordenadas -calculadas a partir de la posición del borde de la tela- permiten distinguir cambios cualitativos clave en distintas secuencias de plegado.

AGRADECIMIENTOS

A mis tutores Jaume y Maria por hacer de guías en lo académico y muchas veces en lo personal en este largo camino.

A Amanda por su apoyo emocional.

A mis padres y hermano por su amor incondicional.

A mis amigos de Barcelona y Madrid por sus palabras de ánimo cuando más las necesitaba.

A mis estudiantes de TFG y TFM por todo lo que aprendimos juntos.

Al personal del IRI y en especial al *Grupo de Percepción y Manipulación* por ayudarme, especialmente en mis peleas con el WAM.

A todos ellos,

Gracias.

CONTENTS

1	Introduction	1
1.1	Motivation	3
1.2	Contributions	6
1.3	Organization	7
1.3.1	Chapter 2: inextensibility modeling	7
1.3.2	Chapter 3: collisions for inextensible cloth	7
1.3.3	Chapter 4: experimental validation	8
1.3.4	Chapter 5: surface reconstruction via Morse theory	10
1.3.5	Chapter 6: developable surfaces	10
1.3.6	Chapter 7: semantic classification of cloth states	11
1.4	Videos	11
1.5	Notation	11
I	Modeling cloth	
2	Inextensible cloth model	15
2.1	Related work	15
2.2	Inextensibility modeling	17
2.2.1	Equations of inextensibility	18
2.3	Discretization of the cloth with FEM	19
2.4	Constraint function enforcing inextensibility	21
2.4.1	Constraints for boundary curves	21
2.4.2	Weighted Galerkin residual method	21
2.4.3	Efficient computation of the coefficients of the metric	23
2.4.4	Constraints evaluation	23
2.4.5	Practical implementation	24
2.4.6	Shearing Energy	25
2.5	Equations of motion of the cloth	26
2.5.1	Modeling of aerodynamics through virtual mass	27
2.6	Discretization of the equations of motion	28
2.6.1	Fast projection algorithm	28
2.7	Evaluation and results	29
2.7.1	Locking test	29
2.7.2	Cusick's test	30
2.7.3	Tank-top simulation	32
3	Cloth collisions	35
3.1	Related Work	35
3.2	Modeling of contacts and friction	38
3.3	Self-collisions	39
3.3.1	Detection of self-collisions	40

3.3.2	Constraint definition for self-collisions	41
3.3.3	Proximity constraints and cloth thickness	42
3.4	Numerical integration of the system	43
3.4.1	Addition of self-collision constraints	44
3.5	Efficient solution of the quadratic problems	45
3.5.1	Factorization of the matrix system	47
3.5.2	Updates of the Cholesky decomposition	48
3.5.3	Detailed algorithm for collisions	50
3.5.4	Similarities and differences with common active-set methods	51
3.6	Evaluation and Results	51
3.6.1	Frictional cylinder: https://youtu.be/_nh-ejHcJAg	52
3.6.2	Rotating sphere: https://youtu.be/-XS3pKpoVbA	52
3.6.3	Collision with a sharp obstacle: https://youtu.be/z7L_0_nSfrM	54
3.6.4	Folding sequence of short pants: https://youtu.be/2gdnjUICb0g	56
4	Experimental validation	59
4.1	WAM-arm experiments	61
4.1.1	Camera data	61
4.1.2	Parameter fitting	63
4.1.3	Sensitivity analysis	65
4.1.4	Comparison with other models	66
4.1.5	Discussion of the results	68
4.2	Aerodynamics study	70
4.2.1	Movements and textiles	72
4.2.2	Parameter fitting	73
4.2.3	A priori forecast of α and δ	75
4.2.4	Discussion of the results	77
4.3	Validation of collisions	77
4.3.1	Tablecloth scenario	78
4.3.2	Hitting scenario	80
4.3.3	Discussion of results	84
II Reconstructing garments		
5	Surface reconstruction via Morse Theory	87
5.1	Related work	87
5.2	Morse theory for manifolds	88
5.2.1	Morse theory for surfaces without boundary	89
5.2.2	Morse theory for surfaces with boundary	90
5.3	Extension to point-clouds	94
5.4	Practical implementation	96
5.4.1	Neighbors identification	96
5.4.2	Tangent space estimation	97

5.4.3	Boundary recognition	97
5.4.4	Reconstruction of curves	98
5.4.5	Morse function and flows	99
5.4.6	Hyperplane sections and computation of critical values	99
5.4.7	Identification of Morse cells	103
5.4.8	Attachment maps of the Morse cells	105
5.4.9	Parametrization of the 2-cells	108
5.4.10	Results	109

III Classifying cloth states

6	Developable Surfaces	119
6.1	The space of developable surfaces	119
6.2	The boundary of developable surfaces	122
7	Semantic Classification of Cloth States	125
7.1	Related work	125
7.2	Preliminaries	127
7.3	Derivation of the Cloth Coordinates	129
7.3.1	GLI of two segments	129
7.3.2	Directional derivative of the GLI	130
7.3.3	Practical computation of dGLI	131
7.3.4	Definition of the dGLI Cloth Coordinates	132
7.4	Results	133
7.4.1	Analysis of folding sequences	134
7.4.2	Confusion matrix of the full data-base	135
7.4.3	Comparison with other shape representations	136
7.4.4	Real images classification	138
7.4.5	Discussion of the results	140
8	Conclusions	143
8.1	Further work	146

IV Appendix

A	Appendix	153
A.1	Tables	153
A.2	Figures	155

	Bibliography	157
--	--------------	-----

LIST OF FIGURES

- Figure 1.1 Various deformation states that a shirt can present: crumpled, extended, hanging, etc. Image adapted from [61]. 1
- Figure 1.2 Comparison at 4 time instants of the fast shaking by a WAM robot of DIN A3 felt (right) versus its simulation with the inextensible model (left). The mean absolute error is 0.39 cm. 2
- Figure 1.3 Final frame of a draping simulation: when all edges in a triangular mesh are forced to maintain their length the cloth faces a spurious resistance to bending. 3
- Figure 1.4 Four frames comparing the recorded hitting of A2 polyester (left) with its inextensible simulation (right); its average error being 1.44 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines, we highlight the moments in which the stick is in contact with the cloth. 9
- Figure 2.1 Isometric deformation of a surface: in both surfaces, the length of the depicted (geodesic) curve is the same. Image adapted from <https://solitaryroad.com/c334.html>. 18
- Figure 2.2 Bilinear parametrization of one of the elements of a quadrangulated surface (shorts). 20
- Figure 2.3 In order to have an isometry between the times t_0 and t_f , we impose the preservation of the first fundamental form at the central node 5, we constraint the length of the boundary edges $\vec{12}, \vec{23}, \dots, \vec{74}, \vec{41}$ and we enforce $F_k(t_0) = F_k(t_f)$ at the corners $k \in \{1, 3, 7, 9\}$. 22
- Figure 2.4 Different meshes used for the locking test. The triangular meshes (left) are irregular whereas the quadrilateral ones (right) are uniform. 29
- Figure 2.5 Four different meshes used for the experiment: the red ones are made of triangles and the blue ones of quadrilaterals. From left to right the number of nodes is: 472, 941, 529, 900. On the bottom of each cloth, we display the euclidean position of the free corner. 30
- Figure 2.6 Photo of a circular cloth draping on top of a circular table. Image taken from [42]. 31

- Figure 2.7 After the cloth has draped on top of the table, the drape coefficient is calculated as the ratio between the area of the plane projection of the cloth C , divided by the area of the flat ring A of width 6cm. Image adapted from [42]. 31
- Figure 2.8 Simulation of Cusick's test with the inextensible model and a mesh of 768 nodes. From left to right we vary the stiffness of the cloth and get respectively DCs of 26.1%, 54.9%, 77.2%. They correspond to *peach-skin* polyester (low stiffness), *imitation* wool (medium) and *tussore* cotton (high). 32
- Figure 2.9 Computation of the drape coefficient (DC) for 3 stiffness values (low, medium and high) and 24 different meshes of cloth from 250 to 1300 nodes. The three mean values for the computed DCs are 23.2%, 52.4%, 77.3%. 32
- Figure 2.10 Simulation with the inextensible model of the shaking of a triangulated tank-top. At each time instant ($t = 1.1s, 1.75s, 2.5s, 3.5s$) we plot the area error with sign (2.26) of each individual triangle. 33
- Figure 2.11 Plot of the time-dependent errors: total area (2.23), mean element (2.24) and dispersion (2.25). Note how the errors increase during the complex parts of the simulation (from $t = 0.5s$ to $t = 3s$) and decrease afterward. 34
- Figure 3.1 If the moving triangles were not intersecting before, then there exists a time in which the edges ab and $a'b'$ were co-planar. 40
- Figure 3.2 Final frame ($t = 2$ seconds) of 3 separate simulations of the fall of a sheet of cloth on top of an off-center cylinder. All the physical parameters are kept constant but friction, which varies among $\mu \in \{0.2, 0.4, 0.55\}$. 52
- Figure 3.3 Final frame ($t = 2.5$ seconds) of 2 different simulations of the collision of a sheet of cloth with a frictional sphere and the floor. One second after the textile has fallen, the sphere performs half a rotation along the z -axis during one second. All the physical parameters are kept constant but in the top image, a small amount of shearing is allowed while in the bottom one, all the inextensibility constraints are enforced. 53

- Figure 3.4 Simulated final frame of cloth's collision with a collection of needle-like obstacles seen from 3 different angles. The cusps must be taken into account separately from the rest of the surface and are treated with the same algorithm we treat self-collisions. 55
- Figure 3.5 Trajectory of the controlled nodes for the dynamic folding of shorts. 56
- Figure 3.6 Simulated sequence of the dynamical folding of a pair of shorts. The first part of the motion (frames two and three) is performed fast enough so that the shorts lay partially flat on top of the table after a lowering phase (frame four). The final fold is completed by dropping the top two corners on top of the leg loops (frames five and six). 57
- Figure 4.1 In the picture we can see all the fabrics (size A3) used in the experiments. From left to right we have: paper, polyester, light cotton, felt, wool, denim and stiff cotton. 60
- Figure 4.2 Experimental setup for the recording of the motion of the real textiles. On the left, the depth camera. On the right, the robotic arm with an affixed hanger. 61
- Figure 4.3 Point-cloud obtained using a depth camera (bottom) and RGB image (top). Note that the camera only gets the depth of the objects visible to it, all the rest (e.g. the robot behind the cloth) is occluded. 62
- Figure 4.4 Quadrilateral meshing (left) of one of the frames of the filtered and de-noised point-cloud (right). Each quadrilateral is divided into triangles for plotting purposes. 63
- Figure 4.5 Comparison at four time instants of the recorded fast motion of paper (right) versus its simulation with the inextensible model (left) with $\delta = 0.37$ and $\alpha = 2.49$. The mean absolute error is 0.41 cm. 64
- Figure 4.6 Mean of the absolute error (4.3) for the fast motion of paper using different values of the parameters of the model (damping α and virtual mass δ). In red we highlight the error with the optimal value of the parameters. 65

- Figure 4.7 Comparison at 4 time instants of the fast motion of light cotton (right) versus its simulation with the inextensible model (left) with $\delta = 0.52$ and $\alpha = 2.69$. The mean absolute error is 0.31 cm. 66
- Figure 4.8 Comparison of the fast motion of light cotton between the inextensible and the other 3 models' errors (bottom curves) and dispersions (top curves), with respect to the recorded motion (using a 9×9 meshing). The mean absolute error for the inextensible model is 0.31cm. 67
- Figure 4.9 Comparison at 4 time instants of the slow motion of wool (right) versus its simulation with the inextensible model (left) with $\delta = 0.47$ and $\alpha = 1.19$. The mean absolute error is 0.37 cm. 68
- Figure 4.10 Absolute error (bottom curves) and dispersion (top curves) of the position of the simulated textile vs. the recorded motion (with the depth-camera and WAM robot) for the slow movement (top) and the fast one (bottom) using the inextensible model. 69
- Figure 4.11 Reflective markers attached to the denim sample (encircled in red). The markers are very small, with a diameter of 3 mm and a weight of 0.013 g. We use 20 reflective markers when the size of the textile is A2 and 12 when it is A3. 70
- Figure 4.12 Setup used to record the motion of the textiles: 5 cameras surround the scene so that every marker (highlighted in red in the photo) is visible to at least 2 cameras at the same time. This ensures that the system can be certain of the 3D position of the marker. 71
- Figure 4.13 Shaking motion sequence (left to right): the cloth is shaken back and forwards. 72
- Figure 4.14 Twisting motion sequence (left to right): the cloth is rotated with respect to the z-axis back and forth several times. 72
- Figure 4.15 Surface plot of the error function $\bar{e}(\delta, \alpha)$ for the fast shaking motion of A2 wool (left) and a close-up near the detected minimum (right). Notice the presence of noise in the close-up. 74

- Figure 4.16 Three frames comparing the recorded fast twisting of A2 polyester (left) with its inextensible simulation (right). The error at the three depicted frames from left to right is 1.82, 1.73 and 1.36 cm respectively; being the average error of the whole simulation 1.15 cm. 75
- Figure 4.17 Three frames comparing the recorded fast shaking of A2 denim (left) with its inextensible simulation (right). The error at the three depicted frames from left to right is 0.63, 0.92 and 1.001 cm respectively; being the average error of the whole simulation 0.84 cm. 75
- Figure 4.18 Putting a tablecloth motion sequence (right to left): the cloth starts suspended and is afterwards laid dynamically (only partially) onto the table. 78
- Figure 4.19 Three frames comparing the recorded tablecloth low friction scenario of A2 wool (left) with its inextensible simulation (right). The error at the three depicted frames from right to left is 0.80, 1.17 and 0.76 cm respectively; being the average error of the whole simulation 0.58 cm. 79
- Figure 4.20 Sensitivity analysis for high friction case, i.e. we vary the value of μ and compute the absolute error (4.3) for the four A2 fabrics. In red we encircle the error found with the optimal parameter of μ . 79
- Figure 4.21 Long-stick hits sequence (left to right): the cloth is held by its two upper corners and then is hit repeatedly with a long stick. The hits are aimed at different locations with varied intensities. 80
- Figure 4.22 Long-stick (with a length of 75 cm and a diameter of 1.5 cm) used to hit the textiles. Two markers are put at both ends of the stick to record its trajectory. 81
- Figure 4.23 Four frames comparing the recorded hitting of A2 stiff-cotton (left) with its inextensible simulation (right); being its average error 1.07 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines, we highlight the moments in which the stick is in contact with the cloth. 83
- Figure 5.1 Critical points of the Morse-Smale function $f(x, y, z) = z$ on an example surface. 88

- Figure 5.2 Three types of critical points (from top to bottom: minima, saddles and maxima) and their Morse data for surfaces without boundary. 90
- Figure 5.3 Four types of critical points located at the boundary and their tangential and normal Morse data. 92
- Figure 5.4 Change in level set when crossing a critical value in a surface (left) and point cloud (right): note the change in neighbors among the 4 marked points after the flow. 95
- Figure 5.5 Typical problems associated to k -nearest neighbors (left) and Voronoi neighbors (right). On the left, the majority of the closest points to v are clustered at one side of it. On the right, vertices that are too far apart from each other have a neighboring cell. 96
- Figure 5.6 In principle, a boundary point can be identified easily because after projecting it and its neighbors on the tangent plane, they cluster in a semi-space of \mathbb{R}^2 (left panel). Nevertheless, this is not always the case for every boundary point (middle panel). To overcome this, we declare a point as a boundary point when none of the plane projections of its neighbors enclose the point (right panel). 98
- Figure 5.7 Intersection of the oriented graph G_{down} with a plane. Changes in the number of connected components of the reconstructed curves $\Gamma(c) = G_{\text{down}} \cap H_c \approx \cup \mathbb{S}^1$ tell us that we have crossed critical points. 100
- Figure 5.8 Apparition of a maximum when following the downwards flow: a new connected component appears. 101
- Figure 5.9 Disappearance of a minimum when following the downwards flow: an existing connected component disappears. 101
- Figure 5.10 Apparition of a saddle point when following the downwards flow: a connected component appears or disappears. 101
- Figure 5.11 Different generic and local level set transformations for surfaces without boundary. The reconstructed curves $\Gamma(c)$ are homeomorphic to a disjoint union of \mathbb{S}^1 . 102

- Figure 5.12 Different local level set transformations for all possible cases involving boundary points. The reconstructed curves $\Gamma(c)$ are homeomorphic to a union of S^1 and closed intervals $[0, 1]$. The previous cases (interior minima, maxima and saddles) are not shown again but can also occur. Saddle points are also drawn, in blue. 103
- Figure 5.13 A sampled *dumbbell*: the black line is the direction of the height function; local maxima, resp. minima, are painted red, resp. black; saddle points are painted blue; their 1-cells are outlined in blue. There are two 2-cells: one in dark red (left) and one in light blue (right). 104
- Figure 5.14 Parametrization of a 2-cell (left) using a rectangle D with isometric sides to the boundary of the 2-cell (right). 108
- Figure 5.15 A sampled knotted torus: the black line is the direction of the height function; local maxima, resp. minima, are painted red, resp. black; saddle points are painted blue; their 1-cells are outlined in blue. On the bottom, we plot the level set curves highlighting when critical points appear. 110
- Figure 5.16 A sampled vest: the black line is the direction of the height function; maxima are painted in red, minima in black; 1-cells corresponding to boundary minima are outlined in blue and the boundary curves in black. The two purple points where the 1-cells meet each other or the boundary curves are added to the decomposition. The numbers correspond to the different formal 1-cells that, when identified (e.g. 7 with 7'), reconstruct the entire surface from 2 pieces homeomorphic to disks. 112
- Figure 5.17 Parametrization by a rectangle of the rightmost 2-cell of Figure 5.16. The bounding 1-cells are mapped isometrically to a flat rectangle and then interior points are obtained using the neighboring relationships of the cloud. The 2-cell consisting of 27 000 points (shown in red) is down-sampled interpolating linearly to a cloud of 900 points (shown in blue). 113

- Figure 5.18 A 3D-scan of real pants: the black line is the direction of the height function; maxima are painted red, minima in black, saddle points are painted blue; 1-cells are outlined in blue and the boundary curves in black. The purple point where the 1-cells meet is added to the decomposition. On the bottom, we plot the level-set curves obtained by intersecting the surface with planes perpendicular to the height function. 115
- Figure 6.1 A smooth simple closed curve (in black) which is the boundary of two developable surfaces (indicated in red and blue respectively) 123
- Figure 7.1 Folding sequence of a quadrangular cloth with its associated *dGLI cloth coordinates*, represented as upper triangular matrices. Each matrix element m_{ij} is a geometrical value corresponding to the *dGLI* between the segments i and j highlighted in red in the corresponding folded state. Notice how some values of the matrix change sign when corners are folded or cross each other. 126
- Figure 7.2 The subset of chosen segments is marked in red. 132
- Figure 7.3 Study of the index during 3 folding sequences. In the left column, we show a representation of the cloth frames, and in the right column the confusion matrix of all of them. In red we highlight the clear class changes that can be identified. 134
- Figure 7.4 Confusion matrix that computes all the distances between the states shown in the top table. 135
- Figure 7.5 Synthetic representatives chosen for each class. When only one is chosen, it is the closest to the centroid of the class. When a class has more sparsity, additional representatives are chosen to represent the subgroups in the class. 139
- Figure 7.6 Results of the real image classification using the data-base presented in Figure 7.4 as reference. The first column shows the ground truth class of the images, and at the bottom of every image the classified class. 140

- Figure A.1 Four frames comparing the recorded hitting of A2 denim (left) with its inextensible simulation (right); being its average error 0.98 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines we highlight the moments in which the stick is in contact with the cloth. 155
- Figure A.2 Four frames comparing the recorded hitting of A2 wool (left) with its inextensible simulation (right); being its average error 1.39 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines we highlight the moments in which the stick is in contact with the cloth. 156

LIST OF TABLES

Table 2.1	Physical parameters of the inextensible model and their meaning. 27
Table 2.2	Distances (cm) between the bottom corner of the cloths (of dimensions $1\text{m} \times 1\text{m}$). 30
Table 4.1	Density, sizes and examples of all the materials used in all the experiments. 60
Table 4.2	Estimated parameters and mean absolute errors (cm) for the slow and fast oscillatory motions performed by the WAM robot. 64
Table 4.3	Comparison of the 4 models. The first column is the mean of the absolute error (4.3) of the simulated cloth with respect to the recorded motion. 67
Table 4.4	Average velocities ($\text{m} \cdot \text{s}^{-1}$) for the twisting motion of the A2 textiles. We display the speeds for the first repetition (with a hanger) and for the second (with bare hands). 73
Table 4.5	Mean absolute error \bar{e} and the mean standard deviation \bar{s} with the optimal value of the parameters averaged over: fabric's material, size (A2 or A3), type of movement and speed for the first repetition of the recordings. 74
Table 4.6	Optimal values of the parameters δ_k, α_k for both repetition I (with hanger) and II (with bare hands) obtained by minimizing the function \mathcal{R} . 76

Table 4.7	Optimal values of the friction coefficients along with the mean absolute error and spatial standard deviation for the low and high friction scenario 78
Table 4.8	Mean absolute error and spatial standard deviation with the optimal value of the parameters α^* and δ^* . In the two last columns, we display the quotient (4.9), for our active-set collision algorithm and a standard interior-point method. 82
Table 4.9	Comparison of the parameters $(\hat{\delta}, \hat{\alpha})$ obtained with the aerodynamic formulas (4.10) with the optimal ones (δ^*, α^*) along with their respective mean absolute errors. 84
Table 7.1	Comparison between different shape representations* 137
Table A.1	Summary of results of the aerodynamic study for the first repetition (with bare hands) of the experiments. For all the 32 recordings we display the characteristics of the recording (fabric, size, motion and speed), and the optimal value of the fitted parameters along with their associated absolute error and spatial standard deviation. 153
Table A.2	Summary of results of the aerodynamic study for the second repetition (with hanger) of the experiments. For all the 32 recordings we display the characteristics of the recording (fabric, size, motion and speed), and the optimal value of the fitted parameters along with their associated absolute error and spatial standard deviation. 154

INTRODUCTION

Robotic manipulation of cloth in a domestic environment, i.e. not in its manufacturing plant, is an increasingly relevant problem because of the ubiquity and versatility of textiles in human lives and activities; with promising applications ranging from folding clothes to dressing people with impaired mobility [31, 40]. One of the main challenges faced is the high variety of deformation states that textiles can present [100] (see Figure 1.1). In contrast to rigid body manipulations, where the dynamics of the manipulated object are very well understood [110], there is not one single model that can be considered best in terms of describing the dynamics of real textiles [85]. Having a faithful and easy-to-adjust physical model of cloth behavior is useful for planning and control strategies in the task of robotic cloth manipulation [26, 71]; as well as for generating the massive data required to train learning algorithms before their deployment and tuning in the real world [25, 58].

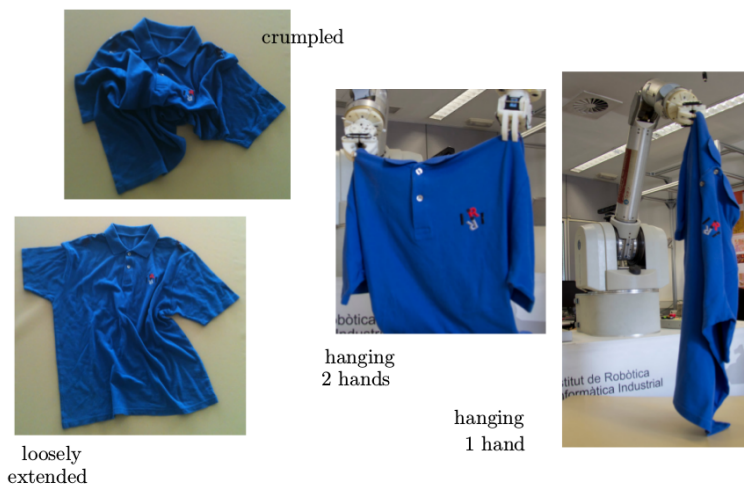


Figure 1.1: Various deformation states that a shirt can present: crumpled, extended, hanging, etc. Image adapted from [61].

Therefore, automatic manipulation of textiles is being tackled nowadays by combining analytical modeling with data-driven learning, so as to overcome the shortcomings of either individual approach. Cloth models need to retain the physical properties relevant for dynamic manipulation and, to be time-affordable, dispense with other features such as appealing rendering. The need for a compact, quick-to-compute model for deformable objects –and textiles, in particular– has

been widely acknowledged within the robot manipulation community [26, 106] and it remains as a key challenge in this field.

Textiles are difficult to model from a material viewpoint. They can be naturally treated as thin plates or shells, but they show an extreme dichotomy: they almost do not stretch in tangent directions, but bend almost freely in the out-of-plane ones. This combination means that textiles are prone to buckling under the slightest compression. Moreover, they are usually non-isotropic because of their weaved inner structure, and nonlinear responses to stresses appear very soon [50]. An additional difficulty for the modeling of the motion of cloth in everyday uses is posed by aerodynamics: most textiles are light, so they are considerably affected by the resistance of air to their movement. One can try simple drag-and-lift models of this resistance in the surface of a cloth away from its edges [77], but those edges bend easily and thus interact with flow turbulences around them in a more complicated way than a rigid solid, even a vibrating one [104].

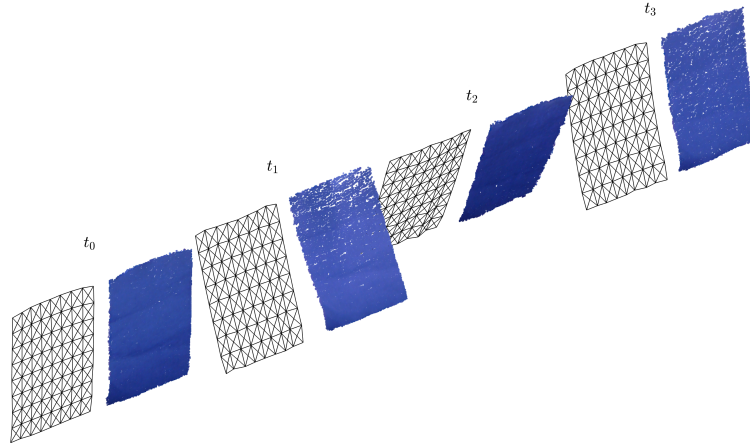


Figure 1.2: Comparison at 4 time instants of the fast shaking by a WAM robot of DIN A3 felt (right) versus its simulation with the inextensible model (left). The mean absolute error is 0.39 cm.

The first main purpose of this work is to introduce and discuss a model of the dynamics of cloth intended for its control under robotic manipulation in a human environment, which means that the textiles will be subjected to moderate or low stresses [15]. Because of its control purpose, we sought a model with which simulations of the motion of cloth could be computed fast, but with a small margin of error (of the order of 1 cm) in the position of every point of the cloth when compared to its position in the real cloth subjected to the same manipulation by the robot (see Figure 1.2, an mean absolute error under 0.5 cm is obtained fitting only 2 parameters). Thus, the main aims of this model are shape realism and speed, while dramatic effects and aesthetic considerations will be disregarded. The achievement

of computational speed taking advantage of the allowed cm margin of error, led us to the most basic hypothesis in the model: we will assume that textiles are *inextensible*, that is, the surfaces that represent them only deform isometrically through space, aiming at preserving not only the area but also both dimensions of each piece of the cloth. This assumption simplifies the model by removing all considerations of elasticity in it, at a cost of introducing errors in the mm range in the position of the cloth under the low stresses at which domestic robots operate. The inextensibility assumption is relevant because most garments stretch very little under their own weight [44] and under the typical tensions of human manipulation [15], making this assumption quite realistic. Moreover, it simplifies the empirical validation of the model, reducing drastically the number of parameters to be fitted since stretching and shearing forces [10, 50] are no longer needed. We will model inextensibility as a set of constraints, and since they will be derived from continuous conditions, the resulting model becomes very stable under different resolutions of the meshed fabric, allowing its use in coarse-mesh simulations.

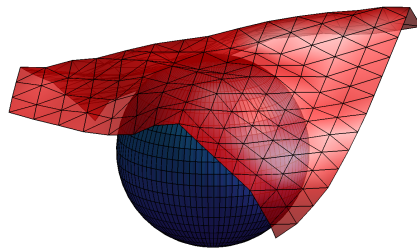


Figure 1.3: Final frame of a draping simulation: when all edges in a triangular mesh are forced to maintain their length the cloth faces a spurious resistance to bending.

1.1 MOTIVATION

Since the publication of the seminal paper [44] a lot of attention has been given to the simulation of inextensible and quasi-inextensible textiles. The main difficulty of inextensible cloth simulation is *locking*, i.e. the meshed garment becoming very rigid or facing a spurious resistance to bending (see Figure 1.3). With limited exceptions such as creasing along straight lines, this is an unavoidable consequence of applying elasticity or spring models to a triangulated fabric and then tuning them for full inextensibility: in the inelastic limit, the lengths of all edges in the mesh become fixed, and often the only deformations preserving all these lengths are those in which the mesh moves as a rigid body (see [8] and [89]). One of the underlying reasons of why this happens is the classical Cauchy rigidity theorem for polytopes, extended by Gluck [43], who showed that generic closed (i.e. compact

and without boundary) polyhedral surfaces in space are rigid, i.e. they lock once one fixes the length of their edges. Although this result does not apply verbatim to cloth meshes (since they usually have a boundary), we believe that a similar mechanism is responsible for cloth locking.

To overcome this difficulty, two approaches have been proposed in literature. In [44] a discrete model is presented where a quad-mesh discretized cloth is made inextensible in the horizontal and vertical directions but still elastic diagonally, i.e. allowing shearing but not stretching. For triangle meshes, [32] uses non-conforming triangles to impose full inextensibility, albeit sacrificing continuity of the polyhedral surface at the edges. One may wonder then, why is another inextensible model necessary? We give several reasons for this:

1. To our knowledge, all inextensible or quasi-inextensible models in literature are discrete in nature, i.e. they impose inextensibility constraining areas, edge lengths, etc. This may create mesh-related artifacts [11] and often impedes convergence as the mesh is refined [85].
2. The other family of models, which are continuous, are all based on elasticity theory, with the drawback that, in the inelastic limit when Poisson's ratio tends to zero, exhibit locking (see [8] and [89]).
3. The development of a continuous model that performs well with coarse meshes is useful, particularly in the context of robotic manipulation of deformable objects, where speed is of critical importance.

In this work, we avoid the problem of locking by using an inextensible cloth model that is continuous and based on differential geometry. This in turn causes all the edges of the meshed garment not to have fixed length and allows the cloth to deform freely but at the same time conserving the total area of the cloth within an error of less than 1% (see Section 2.7.3). Once we have a working inextensible cloth simulator, a related problem that always arises with inextensibility simulations is how to conciliate them with collision and friction forces (i.e. contacts with an obstacle or self-collisions), since both inextensibility and contact forces are hard constraints that the cloth must satisfy at the same time (moreover, friction forces depend on the magnitude of the contact force). In the conclusions of [44], the authors state:

"(...) there is no longer an efficient way to perfectly enforce both ideal inextensibility and ideal collision handling, since one filter must execute before the other, and both ideals correspond to sharp constraints. To enforce both perfectly would require combining them in a single pass, an elegant and

exciting prospect from the standpoint of theory, but one which is likely to introduce considerable complexity and convergence challenges."

To our knowledge, the vast majority of models in literature decouple friction, contacts and strain limiting, with all the possible artifacts that this could introduce. In this dissertation, we develop a novel active set solver that can be seen as an extension of the *fast projection algorithm* of [44], which incorporates contacts, friction and inextensibility in a single pass (see Chapter 3).

Cloth perception and classification of its states: in addition to cloth modeling and simulation, in this memoir, we tackle other problems relevant to the robotic manipulation of cloth. These are fundamentally perception problems (see [62]), namely, surface reconstruction, parametrization of developable surfaces and classification of cloth states.

To succeed in a robotic cloth manipulation, the robot needs to identify and "understand" the textile it is trying to manipulate. When an arbitrary and unknown garment is presented before the robot, a point-sample of it can be obtained through the use of depth-cameras. Nevertheless, this sample of points will in general have no structure and is difficult to understand. This is where *reconstructing* the textile from the point-cloud (i.e. parametrize it and deduce its topology) becomes of great importance if one wishes the robot to control (e.g. fold) the garment [122]. Moreover, from this reconstruction the textile can be simulated with any cloth model, so that different control algorithms can be applied (e.g. model predictive control methods). In this work, we present a reconstruction algorithm that proceeds directly from a point-cloud to obtain a cellular decomposition of the cloth surface: a global piecewise parametrization of the surface is found, with a small number of pieces. From the cellular decomposition, the topology of the surface can be deduced immediately.

Once we have identified the garment we are trying to manipulate, we need to understand its configuration space, i.e. the space of its positions (or states) in the ambient Euclidean space. As already discussed, the main reason why textile objects are challenging to handle for robots is because they change shape under contact and motion. When a robot is faced with the task of e.g. folding a towel in four, it can very easily and rapidly lose track of the current state of the cloth. This is where the study of *developable surfaces* becomes relevant: since the original state of a piece of cloth is flat, the set of possible states under the inextensible assumption is the set of developable surfaces (i.e. Gaussian curvature 0) isometric to a fixed one. Then, to understand the configuration space of the cloth we only need to study the space of developable surfaces isometric to the initial state of the garment. A step in this direction is Theorem 4: we prove that a generic simple, closed, piecewise regular curve in space can be the boundary

of only finitely many developable surfaces with nonvanishing mean curvature. This implies that in order to track the motion of a garment it is not necessary to know the position of all its points, but only those at the boundaries. Inspired by this result, we present a coordinate representation of the configuration of a rectangular cloth which is computed from the position of its border. This representation (which is invariant by rotations and translations) enables the recognition and classification of high-level states, allowing us to classify different cloth configurations into states that we identify as qualitatively *different* (e.g. two folded corners vs one folded corner). This opens the door to performing a more organic robotic control of cloth, where in each step the robot "knows" the current state of cloth and therefore can act accordingly.

1.2 CONTRIBUTIONS

In the following we list, the most important contributions of this thesis:

- A locking-free continuous model for the simulation of inextensible textiles based on natural geometrical constraints is presented together with a new and efficient discretization of the model using the Finite Element Method (FEM). Our approach results in a unified treatment of triangles and quadrilaterals.
- We develop a novel and efficient active-set solver together with a new discretization of friction and self-collisions which results in a non-decoupled implicit resolution of contacts, friction and inextensibility constraints in a single pass.
- We propose a simple way of accounting for the full aerodynamic contribution to dynamics in our cloth model, which leads to an empirical validation under challenging scenarios involving fast motions, friction and even strong hits with a long stick with mean errors of the order of 1 cm.
- An algorithm is developed for the reconstruction of smooth surfaces (with or without boundary) from a point-sample embedded in an ambient space of any dimension. The results are a piecewise parametrization of the surface as a union of Morse cells and a cell complex of small rank determining the surface topology.
- We prove that a generic simple, closed, piecewise regular curve in space can be the boundary of only finitely many developable surfaces with nonvanishing mean curvature. This implies that during a continuous cloth motion, the position of a garment is determined by the location of its boundary.
- We introduce the dGLI cloth coordinates, a low-dimensional representation of the state of a rectangular piece of cloth –based on

the position of its boundary– that allows us to distinguish key topological changes in folding sequences.

1.3 ORGANIZATION

We now proceed to explain the organization of this memoir chapter by chapter.

1.3.1 *Chapter 2: inextensibility modeling*

We explain how to impose the inextensibility conditions (see Equation 2.2) as hard constraints using *Lagrange multipliers* in Section 2.2. In order to treat computationally this, we employ finite elements [17] in the following way: we use them to discretize the Lagrangian of the mechanical system (see Section 2.4) but not the equations of motion, obtaining this way as *Euler-Lagrange equations* an ordinary differential equation (ODE) system and not a (difficult to integrate, e.g. [111]) partial differential equation (PDE) system. We then discuss practical aspects of evaluating the inextensibility constraints in Section 2.4.4. These will be the most technical parts of this chapter and this is due to the proposed model having aspects that are delicate and original from a mathematical viewpoint. Particularly important is how to constrain nodes lying at the boundaries (if not done properly this could lead to *locking*), and the parametrization of the surface, in order to allow arbitrary topologies. After this is done, we discuss how to integrate numerically the equations of motion (Section 2.6). Bending is modeled using Willmore’s Energy as described in [12].

We then test the performance of the presented inextensible model in several challenging scenarios: first, we perform a simple quasi-static test to show the locking-free nature of our model, along with its independence with respect to different meshed topologies (Section 2.7.1). We use different triangle and quadrilateral meshings to prove our point. Second, we show how to simulate Cusick’s test with our model (Section 2.7.2). With this experiment, we show the stability of our model when the mesh is refined. Lastly, we present a scenario with non-trivial cloth topology, where we simulate the motion of a tank-top and check to what extent our theoretical inextensibility assumptions are being met in practice, computing several area errors (Section 2.7.3).

1.3.2 *Chapter 3: collisions for inextensible cloth*

In this chapter, we encounter a ubiquitous problem in cloth simulation: collision modeling and response. First, we explain how to include contact forces with the aid of the very natural Signorini’s conditions (Section 3.2). In that same section, we present a simple model to

account for friction between cloth and possible obstacles (e.g. a table) and with itself. This is done in a manner that will allow us to consider later all constraints (inextensibility and contacts) and friction forces at the same time without any decoupling. In Section 3.3 we explain how to detect and include self-collision constraints under the framework presented in Section 3.2. Particularly important for efficiency and to avoid unwanted oscillations is how to take into account the thickness of cloth. This is explained in Section 3.3.3. Afterwards, in Section 3.4 a novel numerical discretization is presented in order to integrate the extended equations of motion. This can be seen as a natural extension of the *fast projection algorithm* presented in [44] in order to include inextensibility, contacts and friction in a single pass. This discretization leads naturally to a sequence of quadratic problems with inequality constraints. We explain in detail how to include self-collision constraints under this scheme. In Section 3.5 we enter the most mathematical part of the chapter, where we study how to solve efficiently the sequence of quadratic problems defined before. We present a novel *active-set* method tailored to our problem. The main advantage of this new algorithm will be its ability to start from any point and not necessarily from a feasible one. Lastly, we discuss how to implement it efficiently with the use of Cholesky factorizations. A detailed procedure is laid out in pseudo-code in Algorithm 2.

To close the chapter we present several scenarios that put to a test the developed collision model. They will be qualitative in nature, i.e. we only show that our simulator is capable of dealing with them (more quantitative experiments will be performed in Chapter 4, Section 4.3). We will show that the model of friction is effective in static (cylinder experiment, Section 3.6.1) and dynamic (rotating sphere experiment, Section 3.6.2) settings. Next, we show how we can easily include collisions with sharp needle-like objects in Section 3.6.3. Finally, we simulate complicated folding sequences of cloth with non-trivial topologies (a pair of shorts) in Section 3.6.4. The second and the third experiments are challenging scenarios suggested by [70] as tests that every robust collision model should be able to pass.

1.3.3 Chapter 4: experimental validation

In Section 4.1 we compare the inextensible model with recorded motions of real fabrics under different scenarios. First, we show how with as few as two parameters, we are able to model within an error of less than 0.5 cm different types of DIN A3 textiles (such as cotton, wool and felt) under fast and slow shaking movements. To perform this real-world validation we use a Barret robotic arm together with a depth camera to record the real motion of garments being shaken at different velocities. Afterwards, in Section 4.1.3 we perform a sensitivity analysis in order to understand how stable is our model with

respect to the fitted parameters. Finally, the performance of the inextensible model is compared to other popular cloth models (Section 4.1.4).

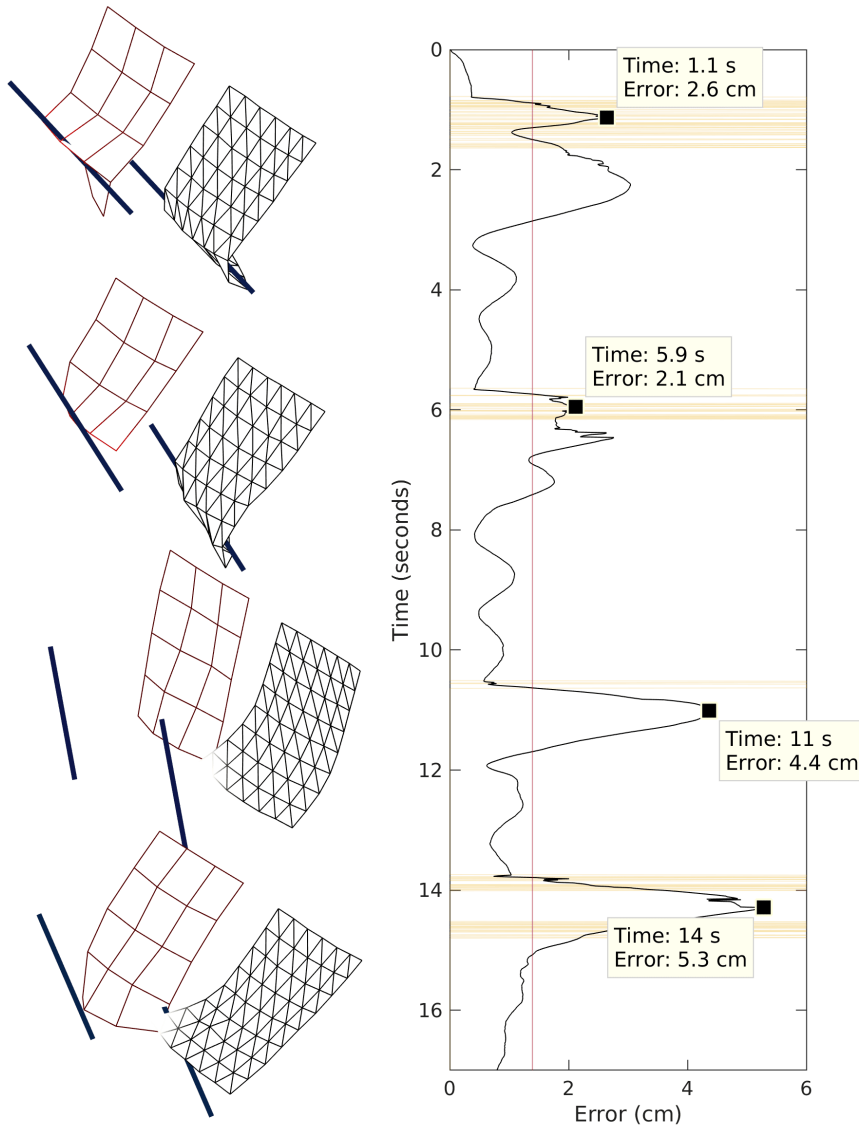


Figure 1.4: Four frames comparing the recorded hitting of A2 polyester (left) with its inextensible simulation (right); its average error being 1.44 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines, we highlight the moments in which the stick is in contact with the cloth.

In the second round of experiments, we perform a more exhaustive set of recordings than before. We enlarge our database by adding a new twisting movement and a larger cloth size (DIN A2). The fabrics are shaken and twisted by a human at two different speeds and are recorded with a *Motion Capture System*. We carry out two repetitions of each motion and at the end have 64 different recordings of about 15

seconds. As before we estimate the optimal values of the two physical parameters and achieve mean errors of less than 1 cm (see Tables 4.5, A.1 and A.2), even for the A2 textiles and fast motions. Finally, we try to understand how the speed and size of the textiles affect their motion by developing a predictive and *a priori* formula for the value of the cloth's physical parameters (see Section 4.2.3).

To close up the chapter we perform a final set of experiments where we use again the motion capture system, this time to record the collision of four (size A2) textiles in two different scenarios. In one of them, the fabrics are laid dynamically on top of a table in a putting-a-tablecloth fashion. In the other, they are hit by a long stick four times at various places and with different strengths (see Figure 1.4). For the first scenario, we find the optimal friction parameter for both a high and a low friction case and study the stability of the model with respect to this parameter (see Figure 4.20). For the hitting experiment, we find again the optimal parameters of the model but this time we also put to use the predictive formulas found before in Section 4.2.3 by using them to compute an estimate of the physical parameters without performing any optimization (see Table 4.9). To conclude, we check computational times and compare our active-set solver with a standard interior-point method.

1.3.4 Chapter 5: surface reconstruction via Morse theory

In this chapter, we present an algorithm for the reconstruction of surfaces from point samples. Our method proceeds directly from the point-cloud to obtain a cellular decomposition of the surface derived from a Morse-Smale function (see Section 5.2). After running the algorithm we obtain a piecewise parametrization of the surface as a union of a small number of Morse cells, suitable for tasks such as noise-removal or meshing, and a cell complex of small rank determining the topology of the manifold (see Section 5.4 for all the implementation details). We explain how to apply this algorithm to samples of smooth surfaces with or without boundary, embedded in an ambient space of any dimension.

1.3.5 Chapter 6: developable surfaces

In this short chapter, we study from a geometric viewpoint developable surfaces, i.e. smooth surfaces with Gaussian curvature 0. The original state of a piece of cloth is flat, so the set of possible states under the inextensible assumption is the set of developable surfaces isometric to a fixed one. Section 6.1 discusses two candidates to the role of generalized coordinates in the space of states of a developable surface under our isometric strain model, and explains their common limitation from the viewpoint of their application. Section 6.2 proposes

an alternative approach: to track the motion of the surface by following its boundary. This is not straightforward because the boundary does not determine completely the position of the surface, but as we explain below our Main Theorem 4 establishes the feasibility of this approach.

1.3.6 Chapter 7: semantic classification of cloth states

In this chapter, we define the dGLI *Cloth Coordinates*, a low-dimensional representation of the configuration state of a rectangular cloth piece that allows us to distinguish topologically different folded states. In Section 7.3 we introduce the novel concept of the directional derivative of the *GLI* (Gauss Linking integral) which allows us to overcome the main limitation of the *GLI* in planar settings: it becomes zero and therefore uninformative. We derive first an expression for the *GLI* of two segments, then we prove that we can perturb the segments slightly to obtain information when they are co-planar. Finally in Section 7.4, we apply these new coordinates to a data-base of cloth configurations taken from simulated folding sequences and then we test experimentally the index on real images of folded clothes.

1.4 VIDEOS

Along this work, several videos will be presented in order to illustrate and show several simulations. All videos (in order or appearance) can be found in the playlist:

<https://youtube.com/playlist?list=PL1XXvX-KsL9puQ79M03K8AhzCoXxNIVu1>

1.5 NOTATION

Before we proceed, for reference, we present a list of the most important symbols used in this thesis. Moreover, for Part I (the most involved notationally) we will follow the following notational conventions:

1. Matrices, tensors and vector functions will be denoted by bold capital letters.
2. Vectors (except for points in \mathbb{R}^3) will be denoted by bold lower-case letters.
3. The rest (e.g. scalar parameters, points in \mathbb{R}^3 , scalar functions, etc.) will be denoted by capital and lower-case italics.

LIST OF MOST IMPORTANT SYMBOLS

S	Surface used to model the cloth.
Ω_e	Elements (e.g. triangles) of the discretized surface.
$\varphi_e(\xi, \eta)$	Local parametrization of the element Ω_e .
$\partial_\xi \varphi_e, \partial_\eta \varphi_e$	Partial derivatives of the parametrizations.
n	Number of vertices/nodes of the discrete surface.
$p_i(t)$	Coordinates in \mathbb{R}^3 of the node i of the surface.
\mathcal{N}_i	Indicator functions, i.e. $\mathcal{N}_i(p_j) = \delta_{ij}$.
$\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t)$	x -coordinates (resp. y and z) of all the nodes of the surface in \mathbb{R}^n .
$\boldsymbol{\varphi}(t)$	Position of all nodes together, i.e. $(\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t))^\top \in \mathbb{R}^{3n}$.
E_k, F_k, G_k	Coefficients of the metric of the discrete surface at node k .
$\mathbf{C}(\boldsymbol{\varphi}(t))$	Constraint function, e.g. $C_i(\boldsymbol{\varphi}(t)) = E_k(t) - E_k(0)$.
$\mathbf{H}(\boldsymbol{\varphi}(t)) \geq 0$	Set of collision constraints.
$\lambda(t), \gamma(t)$	Lagrange multipliers associated to the inextensibility and contact constraints respectively.
$\mathbf{T}_E, \mathbf{T}_F, \mathbf{T}_G$	Tensors used to compute $E_k(t), F_k(t), G_k(t)$.
$\dot{\boldsymbol{\varphi}}(t), \ddot{\boldsymbol{\varphi}}(t)$	Velocity and acceleration of all nodes.
$\mathbf{M}, \mathbf{K}, \mathbf{D}$	Mass, stiffness and damping matrices.
$\mathbf{f}_\mu(\dot{\boldsymbol{\varphi}})$	Friction force.
$\mathbf{V}(\dot{\boldsymbol{\varphi}})$	Unit relative tangent velocities at the points of contact.
$dt > 0$	Time step used to discretize the equations of motion.
$\boldsymbol{\varphi}^m, \dot{\boldsymbol{\varphi}}^m$	Position and velocity of the nodes at $t_m = m \cdot dt$.
$\boldsymbol{\varphi}^m \rightarrow_{dt} \boldsymbol{\varphi}^{m+1}$	Detection of self-collisions between two states.
\mathcal{W}, \mathcal{O}	Working and observation set of the active-set solver.
$\mathbf{QAQ}^\top = \mathbf{LL}^\top$	Cholesky decomposition of the symmetric positive definite matrix \mathbf{A} .
$\boldsymbol{\phi}^m$	Nodes of the meshed recording of real textiles at t_m .
e_m	Time-dependent mean absolute error in cm at t_m .
s_m	Time-dependent spatial standard deviation in cm at t_m .

Part I

MODELING CLOTH

In this first part we deduce the inextensible cloth model: under the assumption that cloth is a surface that moves through space whose metric is preserved, we derive novel inextensibility equations that we later discretize using the Finite Element Method. We test the new model with scenarios involving mesh independency, cloth's area preservation and complex topology simulations.

Then we introduce the problem of collision modeling for inextensible cloth. We develop a new active-set algorithm tailored for resolving contacts (including self-collisions), friction and inextensibility in a single pass without any decoupling of contact and inextensibility constraints. We put to a test the developed collision procedure with scenarios involving static and dynamic friction, sharp objects and complex-topology folding sequences.

Finally, we compare the inextensible model developed in the previous two chapters with collected experimental data from real textiles. We study how faithfully our model is capable of reproducing recorded textiles under challenging scenarios involving fast motions, friction and even strong hits with a long stick.

INEXTENSIBLE CLOTH MODEL

After reviewing the *state of the art* in Section 2.1, we explain how to impose the inextensibility conditions (2.2) as hard constraints using *Lagrange multipliers* in Section 2.2. We discretize the cloth using the Finite Element Method (FEM) in Section 2.3 and derive an original discretization of the inextensibility equations in Section 2.4. There we discuss practical and computational aspects of evaluating the inextensibility constraints. The system of ordinary differential equations modeling the dynamics of the cloth is presented in Section 2.5 along with our physical model of aerodynamics. We discuss how to integrate numerically the equations of motion in Section 2.6. Finally, in Section 2.7 we test the performance of the presented inextensible model in several challenging scenarios.

2.1 RELATED WORK

The mechanic behavior of cloth has been extensively studied from the engineering and computer graphics viewpoints. The engineering focus has usually been on cloth manufacture and its mechanical resistance, using elasticity theory to examine local properties of the material. The aim in computer graphics has always been the simulation and representation of cloth motion, using mostly mass-spring models for speed of computation and paying attention to global problems such as nontrivial cloth topology and collisions. In both fields, the textiles are usually modeled as two-dimensional surfaces. Since the pioneering work of [111] dozens of different models have been proposed for cloth simulation.

Our approach retakes the original idea of [111]: understanding cloth's internal dynamics as the preservation of the first fundamental form of the surface. For surveys and research problems about cloth modeling and simulation see [11, 22, 85]. We now review some of the most popular methods for simulating the internal dynamics of cloth, noting that there is not a clear-cut separation between some of them. We deliberately only reference lines of research that model internal dynamics of cloth in an essentially different *physical* manner. Approaches that develop novel numerical methods to integrate existing physical models will not be considered.

TEXTILE ENGINEERING: The modeling of fabrics from a Structural Mechanics viewpoint has been a topic of interest in Textile Engineering since the start of the industrial manufacture of cloth. See [50] for a survey, and completion in many respects, of this theory, or [56] for

updates. This modeling has been hampered by the complexity of fabric as a material: it is inhomogeneous, even discontinuous, already at a relatively sizable scale; highly anisotropic and capable of a nonlinear response even to relatively modest strains; prone to buckling under very small compression.

Because of these complexities, investigations of the explicit mathematical expression of the stress-strain relationships of fabrics start from the elasticity theory of solids, and typically consider fabric as an elastic thin plate, more rarely as a shell. Ignoring its thickness, the resulting surface has 6 strain fields that determine its displacement and deformation, and a 6×6 symmetric stiffness matrix giving the constitutive equations that relate stress and strain ([56] p.10). Particular properties of fabric such as orthotropy, reduce the number of stiffness parameters to take into account and lead to simplifications of this general elastic model (e.g.[50] and [120]), or to the introduction of nonlinear responses in the model [74].

The development of these models has been based foremost on experimentation, using procedures such as the Kawabata Evaluation System (KES) [65], which accurately measures the tensile, shear, pure bending, compression and friction behavior of a cloth sample under stresses that are typical in the cloth manufacturing process, in order to establish the stiffness parameters of the model. The increase in available computing power has made viable, although not for real-time computing, sophistications of this basic model to take into account the fiber structure of yarns and fabrics (see [23] and [60]).

The computational complexity of these models opened the way to the development of simpler, descriptive models that would later become of common use in Graphic Computer Science (as we explain below), treating fabrics as a viscoelastic material that can be modeled as a mass-spring system [88]. While such descriptive models have been extensively used in fabric simulation, their tenuous connection to reality has made them of limited use for the industrial handling of cloth. Here the thin-plate with linear elasticity models prevail.

ELASTIC MODELS: these models, e.g. [34, 99] and [10], derive the internal forces of the garment from elasticity theory. They are the practical realization of the textile engineering models previously discussed, aimed at graphical simulations rather than at the study of physical properties of cloth. They are all continuous in nature and have the advantage of being physically based, stable under different meshing of the garment and convergent when the mesh is refined [22]. Mostly, they use finite elements to discretize the equations of motion [17]. They can be expensive to evaluate (especially when using non-linear elasticity or the co-rotational method, necessary for ensuring rotational invariance [99]) and may exhibit locking of triangular

meshes if elasticity is heavily reduced: see [32] and [63] and more in general [8] and [89] for a more detailed discussion.

MASS-SPRING SYSTEMS: these models, e.g. [93] and [21], derive internal forces from spring-like energies. They are cheap to evaluate and very intuitive, but less physically sound: they are mesh dependent, have a lot of (non-physical) parameters to be tuned (see [81] for an evolutionary algorithm to tune them) and do not show a convergent behavior when the mesh is refined [85].

CONSTRAINED DYNAMICS: these models derive internal forces from explicit conditions that the cloth must satisfy [11]. Most of the methods use some kind of Lagrange multipliers to impose the constraints. They mostly differ in what the conditions are and the algorithm used to impose the constraints. Our method fits in this category. Sometimes they are used as velocity filters that complement the previous methods. There are mainly two kinds:

1. *Continuous:* in this case the constraints are continuous in nature, e.g. imposing bounds to the strain tensor. Examples of this approach are [76, 84, 112] and [118]. To our knowledge, all methods use elasticity theory to some extent, and therefore in the limit, when elasticity is reduced, face the same problems commented previously. Our method lies in this category with the important difference that it does not use elasticity, but differential geometry of the surface to derive the constraints.
2. *Discrete:* in this case constraints are discrete in nature, e.g. preserving the length of the edges of the meshed garment or the area of the elements. They are derived concretely for the mesh at hand. Examples are [32, 44, 47, 63]. Their main drawback is the possible lack of convergence of the methods (as opposed to the continuous case) and the possible introduction of mesh-related artifacts [76]. Nevertheless, they can be fast and handle better the inelastic scenario than the elasticity-based continuous models.

OTHERS: There are two different recent trends: modeling woven cloth at the fiber level (e.g. [23] and [6]) as opposed to the macroscopic level (i.e. as a surface) and considering cloth as a non-Newtonian fluid using the Material Point Method (MPM) adapted to co-dimensional elasticity [60]. The first method is computationally very intensive and the second while being more efficient depicts cloth as very elastic.

2.2 INEXTENSIBILITY MODELING

As explained before, our main assumption will be to consider cloth as a *continuous* and *inextensible* two-dimensional *surface*. Formulating the model at the continuous level is important because it avoids as much as possible mesh dependencies: i.e. without changing the physical

parameters of the model (e.g. stiffness, damping, etc.) the results are mostly independent of mesh topology (this will be checked in our experiments, see Section 2.7.2). This is not the case when using mass-spring systems [85]. On the other hand, since most cloth materials do not stretch under their own weight, inextensibility is a very reasonable approximation for most textiles (specially in a robotics context, where fine details such as wrinkles are not needed). We will show this by comparing our model to real-life experimental data (see Chapter 4). Moreover, this assumption reduces greatly the amount of tuneable parameters of the model (stretching and their damping forces [10, 50] are no longer present).

Definition 1 (Inextensibility). A smooth surface is said to be inextensible if over time its metric is preserved. This is equivalent to requiring, that at all times the length of any given curve inside the surface remains constant (see Figure 2.1).

Given a parametrization $\varphi = \varphi(\xi, \eta) \in \mathbb{R}^3$ of a smooth surface $S \subset \mathbb{R}^3$ its *metric* (first fundamental form) is given by:

$$d\varphi^\top \cdot d\varphi = \begin{bmatrix} \langle \varphi_\xi, \varphi_\xi \rangle & \langle \varphi_\xi, \varphi_\eta \rangle \\ \langle \varphi_\eta, \varphi_\xi \rangle & \langle \varphi_\eta, \varphi_\eta \rangle \end{bmatrix} = \begin{bmatrix} E & F \\ F & G \end{bmatrix},$$

where $\varphi_\xi = \partial_\xi \varphi$ denotes partial differentiation. The importance of this 2×2 matrix relies on the fact that it gives a unique way of measuring angles and distances intrinsically on the surface [19].

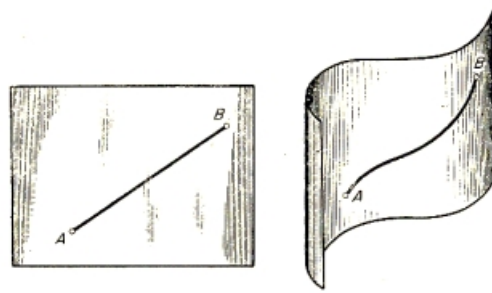


Figure 2.1: Isometric deformation of a surface: in both surfaces, the length of the depicted (geodesic) curve is the same. Image adapted from <https://solitaryroad.com/c334.html>.

Definition 2 (Quasi-inextensibility). When the coefficients E and G of the metric are preserved, but no necessarily F , the surface is said to be *quasi-inextensible*. In physical terms, this means that the surface shears but does not stretch.

2.2.1 Equations of inextensibility

Assume now that S is inextensible and it is moving through space; by virtue of the previous definition this means that we have a continuum

of surfaces $\{S_t\}_{t \geq 0}$ such that $S_0 = S$ and for every $t > 0$ there exists a (smooth) isometry $H_t : S_0 \rightarrow S_t$. We think of the parameter $t \geq 0$ as time. If φ is any parametrization of S , then $\varphi(t) = H_t \circ \varphi$ is a parametrization of S_t and inextensibility means:

$$d\varphi(t)^\top \cdot d\varphi(t) = d\varphi^\top \cdot d\varphi \quad \text{for all } t \geq 0. \quad (2.1)$$

These are 3 independent conditions (depending on time and space):

$$\langle \varphi_{\xi}, \varphi_{\xi} \rangle(t) = E_0, \quad \langle \varphi_{\xi}, \varphi_{\eta} \rangle(t) = F_0, \quad \langle \varphi_{\eta}, \varphi_{\eta} \rangle(t) = G_0 \quad \text{for } t \geq 0. \quad (2.2)$$

Remark 2.2.1 (Rigidity invariance). Noting that $d\varphi(t) = \nabla H_t|_{\varphi} \cdot d\varphi$, we can rewrite (2.1) as

$$d\varphi^\top (\nabla H_t|_{\varphi}^\top \cdot \nabla H_t|_{\varphi} - I) \cdot d\varphi = 0 \quad (2.3)$$

where ∇H_t is the Jacobian of H_t and I is the identity 3×3 matrix. Therefore if $H_t(x) = Ax + b$ is any rigid transformation of \mathbb{R}^3 , we must have that A is an orthogonal matrix and hence it preserves the metric of S as expected. This rigidity invariance is not obtained when using linear elasticity models, e.g. large rotations can lead to different elastic restoring forces [99].

Remark 2.2.2 (Relationship to elasticity). In elasticity theory, the *Cauchy-Green strain tensor* is defined as: $E = \frac{1}{2} [\nabla H_t|_{\varphi}^\top \cdot \nabla H_t|_{\varphi} - I]$. Note that all deformations that have zero strain are isometric (see Equation (2.3)), but not all isometries are strain-free (e.g. bending along a line). This means that imposing (2.3) as a hard constraint (as we will do) is not equivalent to imposing zero strain deformations with classical elasticity theory.

2.3 DISCRETIZATION OF THE CLOTH WITH FEM

For the rest of this chapter, we will assume that our surface S has been discretized, i.e., that we have a polyhedron consisting on an ensemble of *quadrilaterals* or *triangles* Ω_e for $e \in \{1, \dots, n_q\}$ that approximates our original surface: $S \simeq \cup_e \Omega_e$ (see Figure 2.2). For the sake of clarity, we will assume the use of quadrilaterals, although everything works the same with triangles. Let us state precisely the choices in our discretization scheme. Each element will have its own local parametrization:

$$\varphi_e : \Omega_0 := [-1, 1] \times [-1, 1] \rightarrow \Omega_e; \quad \varphi_e(\xi, \eta) = \sum_{i=1}^4 p_i^e N_i(\xi, \eta) \quad (2.4)$$

where the $p_i^e \in \Omega_e \subset \mathbb{R}^3$ are the 4 corners of the quadrilateral and the functions N_i are called the shape functions:

$$\begin{aligned} N_1 &= \frac{1}{4}(1 - \xi)(1 - \eta), & N_2 &= \frac{1}{4}(1 + \xi)(1 - \eta), \\ N_3 &= \frac{1}{4}(1 + \xi)(1 + \eta), & N_4 &= \frac{1}{4}(1 - \xi)(1 + \eta) \end{aligned}$$

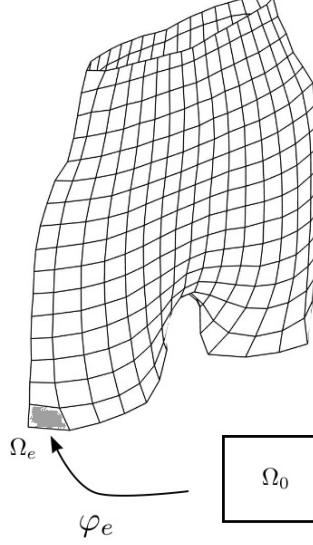


Figure 2.2: Bilinear parametrization of one of the elements of a quadrangulated surface (shorts).

and have the property that $N_i(q_j) = \delta_{ij}$ where δ_{ij} is Kronecker delta and the $q_j \in \mathbb{R}^2$ are the four corners of Ω_0 ordered starting at $(-1, -1)$ and going counter clockwise. Hence, $\varphi_e(q_i) = p_i^e$. If the surface is moving through space, the parametrizations $\varphi_e = \varphi_e(t)$ depend on time and this means that $p_i^e = p_i^e(t)$ vary with time.

Definition 3 (Nodes of the surface). Let us fix a *global* ordering of the n vertices of S . We will call them the nodes of the surface: $\text{Nodes}(S) = [p_1, \dots, p_n]$, where $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$. Then, we can construct the vector of positions of the textile for any time $t \geq 0$:

$$\begin{aligned} \boldsymbol{\varphi}(t) &:= (\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t))^\top \\ &= (x_1(t), \dots, x_n(t) | y_1(t), \dots, y_n(t) | z_1(t), \dots, z_n(t))^\top \in \mathbb{R}^{3n}. \end{aligned}$$

Definition 4 (Indicator functions). We can define global (but non-smooth) indicator functions $\mathcal{N}_i : \cup_e \Omega_e \rightarrow \mathbb{R}$ such that $\mathcal{N}_i(p_j) = \delta_{ij}$ as follows: given any element $\Omega_e \ni p_i$ we define $\mathcal{N}_i|_{\Omega_e} = N_k$ where $N_k(\varphi_e^{-1}(p_i)) = 1$. Thus, if $p = \varphi(t) \in S_t$ is any point of the moving surface, it can be written as

$$\boldsymbol{\varphi}(t) = \sum_{i=1}^n p_i(t) \cdot \mathcal{N}_i(p), \quad (2.5)$$

and therefore $\partial_{\xi} \boldsymbol{\varphi}(t) = \sum_{i=1}^n p_i(t) \partial_{\xi} \mathcal{N}_i$ and $\partial_{\eta} \boldsymbol{\varphi}(t) = \sum_{i=1}^n p_i(t) \partial_{\eta} \mathcal{N}_i$.

Remark 2.3.1. Usually with FEM methodologies one parametrizes a region $R \subset \mathbb{R}^2$ where the values of a function φ are searched; however we parametrize each quadrilateral $\Omega_e \subset S$ independently. Therefore, with this approach, S can have any given topology. Also,

since $\varphi_e(0,0) = \frac{1}{4} \sum_{i=1}^4 p_i^e$, the average of the nodes of each quadrilateral belongs to the element (even though the quadrilaterals are **not** flat, they are bi-linear), so when rendering we can get higher resolution by adding the middle point.

2.4 CONSTRAINT FUNCTION ENFORCING INEXTENSIBILITY

We will construct a smooth and easy-to-evaluate constraint function

$$\mathbf{C} : \text{Nodes}(S_t) \simeq \mathbb{R}^{3n} \rightarrow \mathbb{R}^{n_c} \quad (2.6)$$

such that it is identically zero when all the n_c conditions ensuring inextensibility are satisfied (see Equation (2.11)). This function must depend only on the position vector of the nodes $\varphi(t)$, and on nothing else.

2.4.1 Constraints for boundary curves

In the case of the isometric motion of a curve $\gamma : [0, 1] \rightarrow \mathbb{R}^3$, its metric has only one coefficient given by $\langle \gamma', \gamma' \rangle$. If the curve is discretized into a polygon with vertices $\{\gamma(s_0), \dots, \gamma(s_f)\}$, we have that $\gamma'(s) = \frac{\gamma(s_{l+1}) - \gamma(s_l)}{s_{l+1} - s_l}$ at interior points $s \in [s_l, s_{l+1}]$ of the polygon and hence preserving its metric is equivalent to preserving the lengths of all its edges, so for each edge k we add the constraint:

$$C_k = \|p_{k_1}(t) - p_{k_2}(t)\|_{\mathbb{R}^3}^2 - l_k^2, \quad (2.7)$$

where p_{k_1}, p_{k_2} are the two endpoints of edge k and $l_k > 0$ is the length of the edge at rest. We will apply these constraints to the boundary curves of our discretized cloth S .

2.4.2 Weighted Galerkin residual method

On the other hand, we now explain how to discretize Equation (2.2) for constraining interior nodes. Substituting Equation (2.5) in the left-hand side of the first equation of (2.2) we get:

$$\langle \varphi_{\xi}, \varphi_{\xi} \rangle(t) = \sum_{i,j=1}^n \langle p_i(t), p_j(t) \rangle \cdot \partial_{\xi} \mathcal{N}_i \partial_{\xi} \mathcal{N}_j.$$

The only problem with the previous equation is that it cannot be evaluated at the nodes because the derivative of the indicator functions is not defined there. Nevertheless, if we write:

$$\langle \varphi_{\xi}, \varphi_{\xi} \rangle(t) = \sum_{l=1}^n E_l(t) \cdot \mathcal{N}_l,$$

where $E_l(t)$ are the values of the first coefficient of the metric at the nodes, and then multiply by any indicator function \mathcal{N}_k and integrate over the surface, we obtain:

$$\sum_{l=1}^n E_l(t) \int_S \mathcal{N}_k \mathcal{N}_l dA = \sum_{i,j=1}^n \langle p_i(t), p_j(t) \rangle \cdot \int_S \mathcal{N}_k \partial_{\xi} \mathcal{N}_i \partial_{\xi} \mathcal{N}_j dA, \quad (2.8)$$

and therefore the coefficients $E_l(t)$ can be found by pre-multiplying the right-hand side of (2.8) by the inverse of the *mass matrix*

$$M_{ij} = \int_{\cup_e \Omega_e} \mathcal{N}_i \mathcal{N}_j dA. \quad (2.9)$$

In practical implementations the mass matrix M is substituted by a diagonal matrix called the *lumped mass matrix* M_L defined as

$$(M_L)_{ii} = \sum_j M_{ij}. \quad (2.10)$$

We can imagine this process as collapsing all the mass of the surface to the nodes [17]. This is very convenient because it allows computing the inverse of the mass matrix as the multiplication by the reciprocals of the constants $m_k := (M_L)_{kk}$.

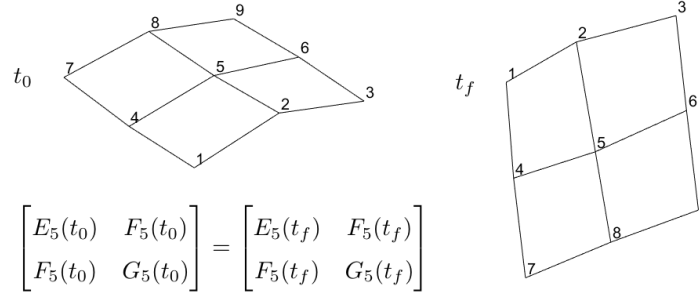


Figure 2.3: In order to have an isometry between the times t_0 and t_f , we impose the preservation of the first fundamental form at the central node 5, we constraint the length of the boundary edges $\vec{12}, \vec{23}, \dots, \vec{74}, \vec{41}$ and we enforce $F_k(t_0) = F_k(t_f)$ at the corners $k \in \{1, 3, 7, 9\}$.

Therefore

$$E_k(t) = \frac{1}{m_k} \sum_{i,j=1}^n \langle p_i(t), p_j(t) \rangle \cdot \int_S \mathcal{N}_k \partial_{\xi} \mathcal{N}_i \partial_{\xi} \mathcal{N}_j dA. \quad (2.11)$$

Naturally, similar equations are obtained when discretizing $\langle \varphi_{\eta}, \varphi_{\eta} \rangle(t)$ and $\langle \varphi_{\xi}, \varphi_{\eta} \rangle(t)$:

$$G_k(t) = \frac{1}{m_k} \sum_{i,j=1}^n \langle p_i(t), p_j(t) \rangle \cdot \int_S \mathcal{N}_k \partial_\eta \mathcal{N}_i \partial_\eta \mathcal{N}_j dA,$$

$$F_k(t) = \frac{1}{m_k} \sum_{i,j=1}^n \langle p_i(t), p_j(t) \rangle \cdot \int_S \mathcal{N}_k \partial_\xi \mathcal{N}_i \partial_\eta \mathcal{N}_j dA.$$

In summary, to impose constraints (2.2), for every interior node k and $t > 0$, we enforce $E_k(t) = E_k(0)$, $F_k(t) = F_k(0)$, $G_k(t) = G_k(0)$ (see Figure 2.3).

2.4.3 Efficient computation of the coefficients of the metric

We now discuss how to compute the right-hand side of (2.11). Let $\text{Int}(S)$ denote the interior (non-boundary) nodes of S . We can define three *time independent* 3-tensors that will allow us to compute the 3 coefficients of the metric of S . First we have \mathbf{T}_E :

$$T_E^{kij} = \frac{1}{m_k} \int_{\cup_e \Omega_e} \mathcal{N}_k \partial_\xi \mathcal{N}_i \partial_\xi \mathcal{N}_j dA, \quad (2.12)$$

where $k \in \text{Int}(S)$ and $i, j \in \text{Nodes}(S)$. Similarly, \mathbf{T}_G is

$$T_G^{kij} = \frac{1}{m_k} \int_{\cup_e \Omega_e} \mathcal{N}_k \partial_\eta \mathcal{N}_i \partial_\eta \mathcal{N}_j dA, \quad (2.13)$$

and lastly \mathbf{T}_F , where $k \in \text{Int}(S) \cup \text{Corners}(S)$, is

$$T_F^{kij} = \frac{1}{2m_k} \int_{\cup_e \Omega_e} \mathcal{N}_k (\partial_\xi \mathcal{N}_i \partial_\eta \mathcal{N}_j + \partial_\eta \mathcal{N}_i \partial_\xi \mathcal{N}_j) dA. \quad (2.14)$$

For a pseudo-algorithm for the computation of the 3 tensors, see Algorithm 1.

Remark 2.4.1 (Symmetry of \mathbf{T}_F). This choice of the \mathbf{T}_F tensor is so that it is symmetric (as the other two tensors) in the i, j indexes and hence computing the Jacobian of \mathbf{C} will be simpler.

Remark 2.4.2 (Treatment of corners). On the other hand, in the presence of corners special care must be taken in order to avoid shearing at those points; that is why we include the corner's indices in the definition of \mathbf{T}_F (see Figure 2.3).

2.4.4 Constraints evaluation

This way, evaluating all the n_c constraints simultaneously reduces to computing the dot product matrix $\mathbf{P}(t) \in \mathbb{R}^{n \times n}$ defined by $P_{ij}(t) = \langle p_i(t), p_j(t) \rangle$ and then doing a tensor product (see Equation (2.11)):

$$\mathbf{C}(\boldsymbol{\varphi}(t)) = \mathbf{T}_{E,F,G} \otimes \mathbf{P}(t) - \mathbf{C}_0 \quad (2.15)$$

Algorithm 1 Computation of $\mathbf{T}_E, \mathbf{T}_F, \mathbf{T}_G$

```

1:  $m \leftarrow \# \text{nodes of each element} \triangleright (m = 3 \text{ (triangles)} \text{ or } m = 4 \text{ (quads)})$ 
2: for element  $\Omega_e$  do
3:    $\varphi_e := \sum_{i=1}^4 p_i^e N_i$   $\triangleright$  (parametrization of  $\Omega_e$ )
4:    $I \leftarrow \text{Index}(\Omega_e)$   $\triangleright$  (node's indices of  $\Omega_e$  e.g. [4, 48, 50, 13])
5:   for Gauss point  $x_l$  and weight  $w_l$  do
6:      $E_l \leftarrow \varphi_{\xi}(x_l) \cdot \varphi_{\xi}(x_l);$ 
7:      $F_l \leftarrow \varphi_{\xi}(x_l) \cdot \varphi_{\eta}(x_l);$ 
8:      $G_l \leftarrow \varphi_{\eta}(x_l) \cdot \varphi_{\eta}(x_l);$ 
9:      $dA_l \leftarrow \sqrt{|E_l \cdot G_l - F_l^2|} \cdot w_l;$ 
10:    for  $k, i, j = 1, \dots, m$  do
11:       $t_F^{kij} + = N_k(x_l) \cdot \partial_{\xi} N_i(x_l) \cdot \partial_{\xi} N_j(x_l) \cdot dA_l;$ 
12:       $t_F^{kij} + = \frac{1}{2} N_k(x_l) \cdot [\partial_{\xi} N_i(x_l) \cdot \partial_{\eta} N_j(x_l) + \partial_{\eta} N_i(x_l) \cdot \partial_{\xi} N_j(x_l)] \cdot$ 
13:       $dA_l;$ 
14:       $t_G^{kij} + = N_k(x_l) \cdot \partial_{\eta} N_i(x_l) \cdot \partial_{\eta} N_j(x_l) \cdot dA_l;$ 
15:    end for
16:  end for
17:  for  $k, i, j = 1, \dots, m$  do  $\triangleright$  (global assembly of the tensors)
18:     $\mathbf{T}_E^{I(k)I(i)I(j)} + = \frac{1}{m_{I(k)}} t_E^{kij};$ 
19:     $\mathbf{T}_F^{I(k)I(i)I(j)} + = \frac{1}{m_{I(k)}} t_F^{kij};$ 
20:     $\mathbf{T}_G^{I(k)I(i)I(j)} + = \frac{1}{m_{I(k)}} t_G^{kij};$ 
21:  end for

```

where we have grouped the previous tensors in just one $\mathbf{T}_{E,F,G} = [\mathbf{T}_E; \mathbf{T}_F; \mathbf{T}_G] \in \mathbb{R}^{n_c \times n \times n}$, and of course $\mathbf{C}_0 = \mathbf{T}_{E,F,G} \otimes \mathbf{P}(0)$. With this definition, computing the Jacobian matrix $\nabla \mathbf{C} \in \mathbb{R}^{n_c \times 3n}$ is very easy (because the tensors are symmetric in the last two indexes):

$$\nabla \mathbf{C}(\boldsymbol{\varphi}(t)) = 2 \cdot [\mathbf{T}_{E,F,G} \otimes \mathbf{x}(t), \mathbf{T}_{E,F,G} \otimes \mathbf{y}(t), \mathbf{T}_{E,F,G} \otimes \mathbf{z}(t)], \quad (2.16)$$

recall that $\boldsymbol{\varphi}(t) = (\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t))^{\top} \in \mathbb{R}^{3n}$.

2.4.5 Practical implementation

We now list some considerations to take into account in a practical implementation of our method:

- As mentioned before, the constraints (2.7) are added to \mathbf{C} to impose that all edges of all boundary curves are preserved. Also, we remark again that the tensor constraints are only applied for interior nodes, except in the presence of corners, where we include their indices in the definition of \mathbf{T}_F in order to avoid shearing there.
- The tensor $\mathbf{T}_{E,F,G}$ needs to be computed only once at the beginning of the simulation because it is time independent. All the integrals

involved are evaluated exactly using standard *Gaussian quadratures* (see [17] for details) as shown in Algorithm 1.

- In order to be efficient we first calculate the Jacobian (according to Equation (2.16)) and then put $\mathbf{C}(\boldsymbol{\varphi}) = \frac{1}{2} \nabla \mathbf{C}(\boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} - \mathbf{C}_0$, so the matrix $\mathbf{P}(t)$ is actually never computed.

- The tensor $\mathbf{T}_{E,F,G}$ is highly sparse due to the fact that most products of the 3 indicator functions (Equation (2.11)) are zero. It is not difficult to see that it has of the order of $\sim 150n$ non-zero elements.

2.4.6 Shearing Energy

Although we are assuming that we can model cloth as inextensible (reasonable for denim, stiff cotton, felt, etc.) in all directions, it is known that this is not completely true in other cases (silk, light cotton, wool). Especially relevant for some types of textiles is shearing, i.e. stretching in the *diagonal* direction. Nevertheless, we will show in Chapter 4 that the inextensible assumption is still a very realistic and accurate assumption. For a smooth surface S parametrized by $\boldsymbol{\varphi}$, a shearing energy can be modeled (see [111]) as

$$\mathcal{S} = \frac{k_s}{2} \int_S \langle \varphi_\xi, \varphi_\eta \rangle^2 dA,$$

where $k_s > 0$ is a constant that controls the amount of shearing allowed. With the ideas previously presented, we can discretize this energy in a very sound and efficient way. Indeed for a discrete surface S with nodes $\boldsymbol{\varphi}(t)$, writing

$$\mathbf{C}_F(\boldsymbol{\varphi}(t)) = \mathbf{T}_F \otimes \mathbf{P}(t) - \mathbf{C}_F^0 \in \mathbb{R}^n \quad (2.17)$$

where \mathbf{T}_F is defined by Equation (2.14), $k \in \{1, \dots, n\}$ and $\mathbf{C}_F^0 = \mathbf{T}_F \otimes \mathbf{P}(0)$; we can define the *fundamental shearing energy* as

$$\mathcal{S}(\boldsymbol{\varphi}(t)) = \frac{k_s}{2} \mathbf{C}_F(\boldsymbol{\varphi}(t))^\top \cdot M \cdot \mathbf{C}_F(\boldsymbol{\varphi}(t))$$

where M is the mass matrix (see Equation 2.9). It is clear that this shearing energy is invariant under rigid motions (because \mathbf{C}_F is). Computing the Jacobian $\nabla \mathbf{C}_F \in \mathbb{R}^{n \times 3n}$ as before (see Equation 2.16), the resulting force derived from this energy is

$$\mathbf{F}_s(\boldsymbol{\varphi}(t)) = -\nabla \mathcal{S}(\boldsymbol{\varphi}(t)) = -k_s \nabla \mathbf{C}_F(\boldsymbol{\varphi}(t))^\top \cdot M \cdot \mathbf{C}_F(\boldsymbol{\varphi}(t)) \in \mathbb{R}^{3n}. \quad (2.18)$$

The fundamental shearing energy is the discrete version of a continuous energy. This is important because keeping the value $k_s > 0$ fixed, as the mesh is refined ($n \rightarrow +\infty$) we observe a convergent and stable behavior.

Remark 2.4.3. For defining the fundamental shearing energy, ideas from [10] were used. Nevertheless, their shearing model is very different from ours: unlike theirs, we derive the discrete shearing energy from a continuous integral equation. From a more practical point of view, they obtain only one condition for every triangle, whereas we have one for each node. The integration of this new energy must be performed implicitly (since it is very stiff), and hence we need to compute the gradient of the force (the Hessian of the energy). As mentioned in [10] the full Hessian leads to an indeterminate matrix (which causes many problems when solving linear systems), and hence it is approximated by:

$$\nabla \mathbf{F}_s(\boldsymbol{\varphi}) = -\nabla^2 \mathcal{S}(\boldsymbol{\varphi}) \simeq -k_s \nabla \mathbf{C}_F(\boldsymbol{\varphi})^\top \cdot \mathbf{M} \cdot \nabla \mathbf{C}_F(\boldsymbol{\varphi}).$$

2.5 EQUATIONS OF MOTION OF THE CLOTH

Finally, we can write down the Lagrangian (kinetic minus potential energies) of our discretized surface:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\varphi}(t), \dot{\boldsymbol{\varphi}}(t)) &= \frac{\rho}{2} \dot{\boldsymbol{\varphi}}(t)^\top \cdot \mathbf{M} \cdot \dot{\boldsymbol{\varphi}}(t) - \rho \mathbf{g}^\top \cdot \mathbf{M} \cdot \boldsymbol{\varphi}(t) \\ &\quad - \frac{\kappa}{2} \boldsymbol{\varphi}(t)^\top \cdot \mathbf{K} \cdot \boldsymbol{\varphi}(t) - \boldsymbol{\lambda}(t)^\top \cdot \mathbf{C}(\boldsymbol{\varphi}(t)), \end{aligned}$$

where

1. $\rho > 0$ is the density of the cloth (assuming homogeneous mass), \mathbf{M} is the augmented mass matrix and $\mathbf{g} = (0, \dots, 0 | 0, \dots, 0 | g, \dots, g)^\top$ where $g = 9.8m/s^2$ is gravity,
2. the stiffness matrix (we are using the isometric bending model described in [12]) is $\mathbf{K} = \mathbf{L}^\top \mathbf{M} \mathbf{L}$ where \mathbf{L} is an approximation of the point-wise Laplacian and $\kappa > 0$ is a bending constant,
3. and finally $\boldsymbol{\lambda}(t)$ are the Lagrange multipliers ensuring inextensibility (at the interior nodes and boundaries) and other possible positional constraints (e.g. the corners of the textiles to be manipulated) included.

Thus, we get as Euler-Lagrange equations [110] the following ODE system:

$$\begin{cases} \rho \mathbf{M} \ddot{\boldsymbol{\varphi}} = \mathbf{f}_\rho - \kappa \mathbf{K} \boldsymbol{\varphi} - \mathbf{D} \dot{\boldsymbol{\varphi}} - \nabla \mathbf{C}(\boldsymbol{\varphi})^\top \boldsymbol{\lambda} \\ \mathbf{C}(\boldsymbol{\varphi}) = 0 \end{cases} \quad (2.19)$$

where $\mathbf{f}_\rho = -\rho \mathbf{M} \mathbf{g}$ is the force of gravity and we have added Rayleigh damping: $\mathbf{D} = \alpha \mathbf{M} + \beta \mathbf{K}$, where α and β are positive parameters [127].

Notice how our inextensibility assumption reduces greatly the number of physical parameters of the model (we do not have shearing or

stretching parameters and their respective dampings). Nevertheless, cloth dynamics can be very complicated and there is one important factor we are not accounting for in the previous equations: air resistance. Although there exist some simplified models [77], aerodynamic forces on a deformable object (i.e. cloth) submerged in a fluid (i.e. air) are difficult to model (see [41] and [73]), especially near the boundaries of cloth, because turbulences appear and the nonrigid response of cloth to such turbulences is way more unpredictable than that of a rigid, even vibrating, body (see [104]).

Remark 2.5.1 (Shearing forces). In the case we want to allow some shearing of the cloth, we simply introduce the force $F_s(\varphi(t))$ defined in Equation (2.18) into the first equation of the ODE system (2.19). Obviously, in that case, we would have one parameter more $k_s > 0$. For the rest of this thesis unless explicitly stated (namely in Sections 3.6.2, 3.6.4 and Chapter 7) we assume that no shearing is allowed and set $\kappa_s = 0$.

PARAMETER	MEANING
ρ	Density (inertial mass)
δ	Virtual (gravitational) mass
κ	Bending/stiffness
α	Damping of slow oscillations
β	Damping of fast oscillations

Table 2.1: Physical parameters of the inextensible model and their meaning.

2.5.1 Modeling of aerodynamics through virtual mass

The equivalence principle states that for any object its inertial mass is equal to its gravitational mass, which means that an object in vacuum of inertial mass m would fall freely under the action of a force of magnitude mg , where g is gravity. Nevertheless, experiments show that in the presence of air, the free-falling velocities depend on the (shape of the) object at hand. In our experiments, we have found that aerodynamic effects can be modeled without explicitly including them by allowing inertial and gravitational masses to be different. Although this is not true in the physical world, the consideration of the two masses as virtual in our model allows us to account for all aerodynamic contributions to cloth dynamics (drag, lift, turbulences at the boundaries) with great accuracy. Hence, we put $\mathbf{f}_\delta = -\delta\mathbf{Mg}$, setting δ as a new parameter to be fitted. In Table 2.1 we summarize the meaning of all the parameters of our model.

2.6 DISCRETIZATION OF THE EQUATIONS OF MOTION

As usual, we approximate $\boldsymbol{\varphi}(t)$ and $\dot{\boldsymbol{\varphi}}(t)$ with $\{\boldsymbol{\varphi}^0, \boldsymbol{\varphi}^1, \dots\}$ and $\{\dot{\boldsymbol{\varphi}}^0, \dot{\boldsymbol{\varphi}}^1, \dots\}$, where $\boldsymbol{\varphi}^n$ and $\dot{\boldsymbol{\varphi}}^n$ are the position and velocities of the nodes of the mesh at time $t_n = n \cdot dt$ and $dt > 0$ is the size of the time step. Using an implicit Euler scheme to integrate Equations (2.19) we obtain:

$$\begin{cases} \boldsymbol{\varphi}^{n+1} = \boldsymbol{\varphi}_0^{n+1} - \mathbf{M}^{-1} \nabla \mathbf{C}(\boldsymbol{\varphi}^{n+1})^\top \boldsymbol{\lambda}^{n+1} \\ \mathbf{C}(\boldsymbol{\varphi}^{n+1}) = 0, \end{cases} \quad (2.20)$$

where $\boldsymbol{\varphi}_0^{n+1}$ is the unconstrained step (which depends on $\boldsymbol{\varphi}^n$ and $\dot{\boldsymbol{\varphi}}^n$). Note that at time $t_n = n \cdot dt$ the only unknown in previous equations is $\boldsymbol{\varphi}^{n+1}$ (and $\boldsymbol{\lambda}^{n+1}$).

2.6.1 Fast projection algorithm

Finding $\boldsymbol{\varphi}^{n+1}$ can be done solving the following constrained optimization problem:

$$\begin{cases} \min_{\boldsymbol{\varphi}^{n+1}} (\boldsymbol{\varphi}^{n+1} - \boldsymbol{\varphi}_0^{n+1})^\top \cdot \mathbf{M} \cdot (\boldsymbol{\varphi}^{n+1} - \boldsymbol{\varphi}_0^{n+1}) \\ \text{s.t. } \mathbf{C}(\boldsymbol{\varphi}^{n+1}) = 0, \end{cases} \quad (2.21)$$

because Equations (2.20) are the critical (stationary) points of the optimization problem. This is a quadratic program with quadratic constraints which can be solved with Newton's method. Nevertheless, in order to make it computationally tractable in front of difficulties such as indefinite system matrices, it is approximated by a *sequence* of quadratic programs with linear constraints. Write $\boldsymbol{\varphi}_{j+1} = \boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}$ and make the approximation $\mathbf{C}(\boldsymbol{\varphi}_{j+1}) = \mathbf{C}(\boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}) \simeq \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla \mathbf{C}(\boldsymbol{\varphi}_j) \Delta\boldsymbol{\varphi}_{j+1}$, then the sequence is:

$$\begin{cases} \min_{\Delta\boldsymbol{\varphi}_{j+1}} \Delta\boldsymbol{\varphi}_{j+1}^\top \cdot \mathbf{M} \cdot \Delta\boldsymbol{\varphi}_{j+1} \\ \text{s.t. } \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla \mathbf{C}(\boldsymbol{\varphi}_j) \cdot \Delta\boldsymbol{\varphi}_{j+1} = 0, \end{cases}$$

the initial point being $\boldsymbol{\varphi}_0 = \boldsymbol{\varphi}_0^{n+1}$. This is called the *fast projection algorithm* [44]. Each of these quadratic programs (which has a diagonal matrix system \mathbf{M}) can be reduced to solving a linear system and we iterate until the constraints are satisfied to a given relative tolerance (usually 0.1%). We have found this algorithm very stable and fast for our purposes.

Remark 2.6.1. In the presence of an obstacle (e.g. a table) given by an implicit equation $\mathbf{H}(\boldsymbol{\varphi}) = 0$ with a well-defined outwards normal $\nabla \mathbf{H}(\boldsymbol{\varphi})$, we can model collisions by including new constraints in every iteration of the *fast projection algorithm* described earlier, i.e. we solve

iteratively the following sequence of quadratic programs with linear equality and inequality constraints:

$$\begin{cases} \min_{\Delta\boldsymbol{\varphi}_{j+1}} \frac{1}{2} \Delta\boldsymbol{\varphi}_{j+1}^\top \cdot \mathbf{M} \cdot \Delta\boldsymbol{\varphi}_{j+1} \\ \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla\mathbf{C}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} = 0, \\ \mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} \geq 0, \end{cases}$$

where we have made the approximation

$$\mathbf{H}(\boldsymbol{\varphi}_{j+1}) = \mathbf{H}(\boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}) \simeq \mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1}.$$

A theoretical justification together with an algorithm to solve efficiently the previous quadratic problem will be discussed in detail in Chapter 3.

2.7 EVALUATION AND RESULTS

In this section, we study experimentally several desirable properties of the inextensible cloth model. The section is organized as follows: first, we perform a simple quasi-static test to show the locking-free nature of our model, along with its independence with respect to different meshed topologies (Section 2.7.1). We use different triangle and quadrilateral meshings to prove our point. Second, we show how to simulate Cusick's test with our model (Section 2.7.2). With this experiment, we show the stability of our model when the mesh is refined. Lastly, we present a scenario with non-trivial cloth topology, where we simulate the motion of a tank-top and check to what extent our theoretical inextensibility assumptions are being met in practice, computing several area errors (Section 2.7.3).

2.7.1 Locking test

With this experiment, we intend to show the locking-free nature of our inextensible model (no shearing is allowed $\kappa_s = 0$), and its stability with respect to mesh topology. We fix three corners of a flat sheet of cloth (with added random noise of standard deviation 3mm) of 1m by 1m, and let the fourth corner fall freely.



Figure 2.4: Different meshes used for the locking test. The triangular meshes (left) are irregular whereas the quadrilateral ones (right) are uniform.

We use four different meshings of the cloth: two with triangles and two with quadrilaterals (see Figure 2.4), but we keep the physical parameters fixed. The visual results of the experiment can be seen in Figure 2.5. The four textiles fold diagonally without any locking artifact.

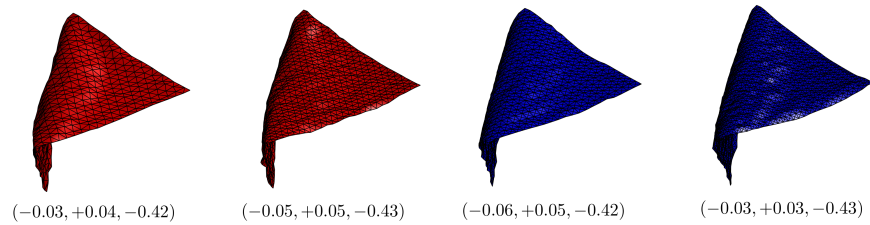


Figure 2.5: Four different meshes used for the experiment: the red ones are made of triangles and the blue ones of quadrilaterals. From left to right the number of nodes is: 472, 941, 529, 900. On the bottom of each cloth, we display the euclidean position of the free corner.

NODES / ELEMENT	472/ \triangle	941/ \triangle	529/ \square	900/ \square
472/ \triangle	0	2.45 cm	3.16 cm	1.41 cm
941/ \triangle	-	0	1.41 cm	2.83 cm
529/ \square	-	-	0	3.74 cm
900/ \square	-	-	-	0

Table 2.2: Distances (cm) between the bottom corner of the cloths (of dimensions $1\text{m} \times 1\text{m}$).

In Table 2.2 we compute the euclidean distance of the bottom corner of every textile with respect to each other. Note how, doubling the number of nodes, or changing from irregular triangles to regular quadrilaterals, does not alter the result within a margin of error of a few centimeters (recall that the sheets are $1\text{m} \times 1\text{m}$).

2.7.2 Cusick's test

This subsection will demonstrate the stability of our model under refinements of the mesh. We will explain how to simulate Cusick's test [56] using our simulator. Cusick's test consists in letting a circular cloth of radius 15cm drape on top of an also circular table of radius 9cm (see Figure 2.6) with their centers aligned.



Figure 2.6: Photo of a circular cloth draping on top of a circular table. Image taken from [42].

Then, one computes the area of the plane projection of the draped cloth C and divides it by the area of the flat ring A of width 6cm (see Figure 2.7).

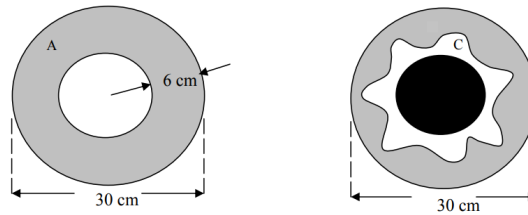


Figure 2.7: After the cloth has draped on top of the table, the drape coefficient is calculated as the ratio between the area of the plane projection of the cloth C , divided by the area of the flat ring A of width 6cm. Image adapted from [42].

This is called the **drape coefficient** of the cloth:

$$\text{DC \%} = 100 \times \frac{C}{A}. \quad (2.22)$$

It is known that this coefficient depends heavily on the stiffness of the cloth [56]. This can be understood intuitively: a completely rigid sheet of metal would not bend and would have DC equal to 100%, whereas a really light material (e.g. silk) has a DC pretty close to 0. The measurement of this coefficient is not trivial: it usually requires a dedicated machine (Cusick's Drape Tester [65]); and variation of the measured coefficient for the same cloth is common, so several specimens of the same textile have to be employed and then their DC's averaged [42].

Naturally, these complications have caused the creation of alternative measuring methods using computer simulations (e.g. [38]). In order to simulate Cusick's test with our model, we quadrangulate the ring A of Figure 2.7, we fix in space its inner boundary nodes and let gravity act (see Figure 2.8). In order to study the dependency of the DC with respect to mesh resolution, in Figure 2.9 we plot the DC for three stiffness values κ (low: κ_0 , medium: $5\kappa_0$ and high: $10\kappa_0$; where $\kappa_0 = 0.005$) and 24 different meshes of the ring A ranging from 250 nodes to 1300. The three mean values for the computed DCs are

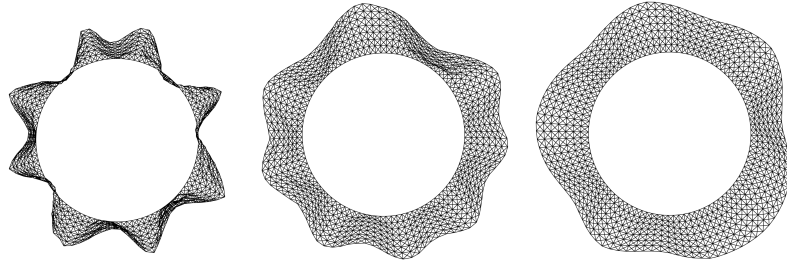


Figure 2.8: Simulation of Cusick’s test with the inextensible model and a mesh of 768 nodes. From left to right we vary the stiffness of the cloth and get respectively DCs of 26.1%, 54.9%, 77.2%. They correspond to *peach-skin* polyester (low stiffness), *imitation* wool (medium) and *tussore* cotton (high).

23.2%, 52.4%, 77.3% and they approximately correspond to *peach-skin* polyester (low stiffness), *imitation* wool (medium stiffness) and *tussore* cotton (high stiffness) (see the Appendix of [42]: IDs: 38, 37, 22). Figure 2.9 shows that the computation of the DC with our model is very stable (especially from 700 nodes on), and hence the model has a robust behavior with respect to mesh resolution. As we already said, this is very relevant for robotic applications, where the use of coarse meshes becomes a necessity because of performance constraints.

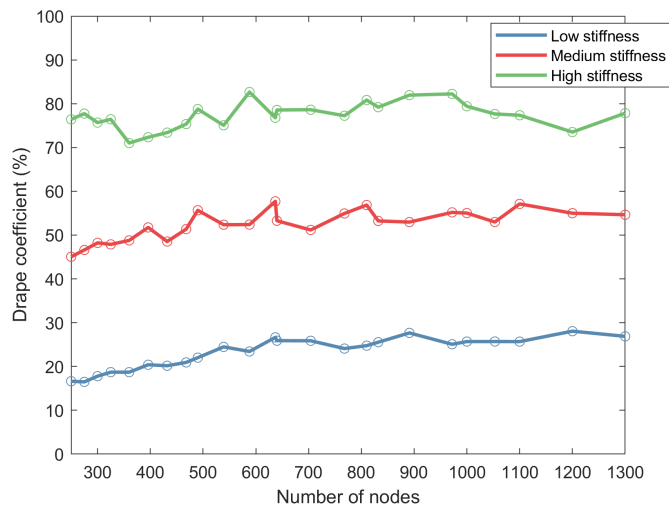


Figure 2.9: Computation of the drape coefficient (DC) for 3 stiffness values (low, medium and high) and 24 different meshes of cloth from 250 to 1300 nodes. The three mean values for the computed DCs are 23.2%, 52.4%, 77.3%.

2.7.3 Tank-top simulation

With this experiment, we aim to show the ability of our model to simulate complex topologies (e.g. a tank-top shirt, see Figure 2.10 and

<https://youtu.be/DvSWEXdw6Bo>) and study different area errors. We track the variation of area of the garments through time. We think area is a good measure since it is a physical property of cloth and not a mesh-dependent metric (e.g. stretching of the edges of the elements). We compute the area of each element of the discretized cloth using its local parametrization (2.4).

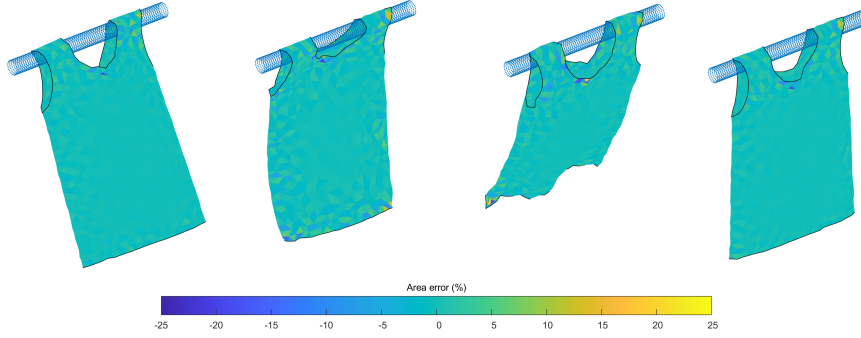


Figure 2.10: Simulation with the inextensible model of the shaking of a triangulated tank-top. At each time instant ($t = 1.1s, 1.75s, 2.5s, 3.5s$) we plot the area error with sign (2.26) of each individual triangle.

We use three area errors:

1. *Total area error:*

$$e_t(t) = \frac{|A(t) - A_0|}{A_0}, \quad (2.23)$$

where A_0 is the total area of the garment at time $t = 0$ and $A(t)$ is its area at time $t > 0$.

2. *Mean element error:*

$$e_m(t) = \frac{1}{n_e} \sum_e \frac{|a_e(t) - a_e(0)|}{a_e(0)}, \quad (2.24)$$

where $a_e(t)$ is the area of element Ω_e at time $t \geq 0$.

3. *Dispersion:*

$$d_a(t) = e_m(t) + 2\sqrt{\text{Var}_e \left(\frac{|a_e(t) - a_e(0)|}{a_e(0)} \right)}. \quad (2.25)$$

We simulate the shaking with a *hanger* of a meshed tank-top with 1676 triangles (see Figure 2.10) during 4.5s. There, we also plot the area error with sign of each individual triangle for each time instant:

$$e(t) = \frac{a_e(t) - a_e(0)}{a_e(0)}. \quad (2.26)$$

It is interesting to notice the concentration of error near the boundaries of the cloth and at the points where the tank-top makes contact

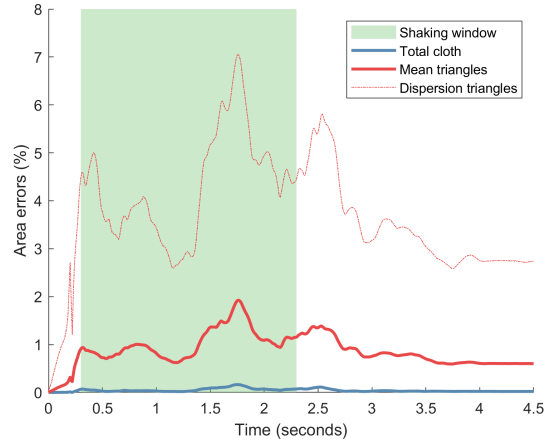


Figure 2.11: Plot of the time-dependent errors: total area (2.23), mean element (2.24) and dispersion (2.25). Note how the errors increase during the complex parts of the simulation (from $t = 0.5\text{s}$ to $t = 3\text{s}$) and decrease afterward.

with the moving cylinder. Nevertheless, note that when a triangle is stretched (color yellow), next to it appears a triangle that contracts (color blue). This makes the total area error to be very small (see Figure 2.11). Finally in Figure 2.11, we plot the three area errors (2.23), (2.24) and (2.25). The total area error is very low during all the simulation, almost zero. The mean area error is higher, but still very low (it peaks around $t = 1.75\text{s}$, the second frame in Figure 2.10), keeping itself around 1%. The dispersion error, on the other hand, tells us that if the distribution of the errors were normal, the error of 95% of the mesh triangles would be under 7% during all the simulation.

In this chapter, we study the problem of collision detection and response tailored for our inextensible cloth model. After reviewing the *state of the art* in Section 3.1, we explain how to include contact reaction forces with the aid of the Signorini’s conditions (Section 3.2). In that section, we also present a simple model to account for friction between cloth and possible obstacles (e.g. a table) and with itself. In Section 3.3 we explain how to detect and include self-collision under the framework presented in Section 3.2. Afterward in Section 3.4 a novel numerical discretization is presented in order to integrate the extended equations of motion. This discretization leads naturally to a sequence of quadratic problems with inequality constraints. In Section 3.5 we study how to solve efficiently the sequence of quadratic problems defined before. A detailed procedure is laid out in pseudo-code in Algorithm 2. To close the chapter in Section 3.6 we present several scenarios that put to the test the developed collision model.

3.1 RELATED WORK

There is a rich history of research on contact and collisions for cloth simulation; in the following, we will review what we consider the most relevant methods, with an emphasis on newer works. We focus especially on articles that model collisions in physically different manners or use novel numerical algorithms to resolve them (and not so much in performance, e.g. GPU implementations of existing methods). Most of these works come from the Computer Graphics (CG) community and not so much from the Textile Engineering fields. This is due to the fact that in CG applications a core concern has always been simulating dressed moving mannequins (e.g. for movies and video games), whereas for textile engineers the focus has been on measuring intrinsic fabric properties. There are 3 main types of collision-response methods:

Penalty-based methods: these include very stiff spring-like forces of the form $kf(\epsilon)$ (where ϵ is the detected penetration depth) into the dynamical system when a penetration is detected. They are easy to implement and can work for simple cases, but are not physically accurate (e.g. they do not conserve momentum during the collision, see [105]) and introduce a lot of stiffness into the system when k is large (making it harder to integrate numerically). For example, Provot [92] is one of the first to propose a penalty-based approach to solve

collisions for cloth modeled as a mass-spring system. However, his method has no theoretical guarantees when there is more than one simultaneous collision. That is why, when many collisions accumulate during the same time step, he must resort to a fail-safe consisting of rigidifying zones of the cloth. On the other hand, he is among the first to give a formula to detect continuous-time collisions (i.e. when two moving edges or a node and triangle cross, see Section 3.3). Finally, one of the main theoretical problems with penalty-based methods is that there is always a fast or strong enough collision where they fail because the spring force is not strong enough (although there are sophistications with more guarantees such as [48]). These types of models have somehow fallen out of fashion in recent times (at least in the research literature) in favor of constraint-based approaches.

Impulse-based methods: these methods include impulse forces (mostly based on rigid-body mechanical ideas) which are then used to modify velocities (and thus positions) instantaneously. They work well for individual collisions and are fast to compute, but run into problems for multiple simultaneous collisions. In this line, the work by Bridson et al. [18] is considered to be by the CG community the first truly robust method for handling collisions, contact and friction for cloth simulation. More than a unified physical model, their method consists of a list of procedures used to get a state of cloth that is collision-free but not necessarily physically realistic. They first apply penalty forces as a prevention method and then impulse forces for detected collisions in continuous time. When there are simultaneous collisions in the same time step, after a fixed number of iterations they also resort to rigid impact zones (but with corrected formulas with respect to [92]). Some alternatives exist to avoid the rigidification or areas of the cloth, e.g. [115] and [49]. These two methods derive impulse forces for the case of simultaneous collisions with the aid of constraints that are being violated by the detected penetrations. Their main problem is that in order to be efficient they derive the impulses from equality constraints, and this introduces *sticking* artifacts into the simulation (some nodes are forced to stay in contact, when they otherwise would depart). Moreover, another issue with impulse-based methods is that since one is modifying positions instantaneously, strain-limiting procedures must be performed prior to (or after) collision response and thus constraints such as inextensibility cannot be maintained exactly.

Constraint-based methods: with the increase of computing power in the last decades these methods have flourished from a research viewpoint. Their idea is simple: once a collision is detected, a constraint is defined (which is being violated because of the collision) and an optimization problem should be solved with all the detected constraints. Most methods vary mainly in how the optimization problem is solved

and how friction is modeled (going from exact Coulomb models to linearized ones). These restrictions can be imposed as equalities or inequalities. As already said, imposing the constraints as equalities can be very efficient but one runs into *sticking artifacts* (since some constraints can pull from others) and thus it is better to consider them inequalities. This is in turn known as the Signorini-contact model. Otaduy et al. [87] were among the first to propose a physically sound constrained dynamics formulation for cloth simulation and contact. They employ Signorini’s contact model and add to it a linearized Coulomb’s friction model. The optimization program is stated formally as a linear complementary problem (LCP) and friction and contacts are afterwards decoupled in order to be solved numerically.

Years later, Li et al. [69] implement exact Coulomb friction for cloth simulation using adaptive meshes. Their constraint-based solver (released later as an open-source simulator called ARGUS) is costly to run but treats contacts (and friction) simultaneously and implicitly. Recently, Ly et al. [75] have proposed an alternative numerical algorithm based on Projective Dynamics that accelerates by an order of magnitude results obtained with ARGUS. Its main drawback is that it inherits the limitations of Projective Dynamics, in particular, the lack of a simple rule to ensure convergence. One of the main limitations of all the previous methods is the difficulty of integrating at the same time strain limiting (e.g. inextensibility) with the collision handling algorithm. In this line, Li et al. [70] have developed a method that integrates strain limiting and collisions in a single pass with the extensive use of barrier functions. They also propose a benchmark set of challenging tests. Naturally, constraint-based methods also have some drawbacks: most of the time they need dedicated solvers and thus can be cumbersome to implement and since several optimization programs must be solved, they are slower than impulse or penalty-based methods.

To finish this section we would like to mention a research line, that although is not a collision model *per-se* is relevant to the topic:

Untangling methods: these methods have the particularity that they do not focus on solving new collisions to come but on resolving pre-existing cloth intersections. They appeared out of practical necessity (e.g. in the movie industry) since these pre-existing penetrations often arise when dressed mannequins move in nonphysical or impossible ways (pinching cloth between extremities that intersect each other). For example, [9] and [116] give algorithms to untangle cloth in an efficient manner. More recently, in [121] a method is proposed where new collisions and pre-existing penetrations are all handled in a unified manner. The main challenge of these methods is actually a topological one: in which normal direction one needs to *push* the intersecting

elements so that the penetration is resolved without knowing a past intersection-free state?

Overview

The model we will derive in this chapter lies in the category of constraint-based methods and hence can handle efficiently simultaneous collisions. We will solve a quadratic problem with inequality constraints and therefore we will be employing the physically accurate model of Signorini. In order to do so, we will develop a novel active-set solver in order to resolve collisions efficiently. Moreover, we derive a simple friction model that allows us to integrate all forces and constraints in a simple pass without the need to decouple contact and friction forces like it has been traditionally done for rigid body contacts (see [64]). Finally, our method considers strain limiting (inextensibility) and contact at the same time, unlike most current methods. Our algorithm can be seen as an extension of the *fast projection algorithm* (see [44] and Section 2.6.1) developed in order to incorporate contacts, friction and inextensibility in a single pass.

3.2 MODELING OF CONTACTS AND FRICTION

For its application in the real world, we need to include in our model collisions of the cloth with an object (e.g. a table) and with itself. We will model this by enforcing a set of constraints $\mathbf{H}(\boldsymbol{\varphi}) \geq 0$, which we assume have a well-defined outwards normal $\nabla\mathbf{H}(\boldsymbol{\varphi})$ (almost everywhere). Observe that the obstacle could move in time, but we need to know its position. We can then model collisions by including new (non-smooth, see [67] and [125]) forces in our dynamical system. Signorini's contact model then reads (see [59]):

$$\begin{cases} \mathbf{M}\ddot{\boldsymbol{\varphi}} = \mathbf{F}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) - \nabla\mathbf{C}(\boldsymbol{\varphi})^\top\boldsymbol{\lambda} + \nabla\mathbf{H}(\boldsymbol{\varphi})^\top\boldsymbol{\gamma}, \\ \mathbf{C}(\boldsymbol{\varphi}) = 0, \\ \mathbf{H}(\boldsymbol{\varphi}) \geq 0, \quad \boldsymbol{\gamma} \geq 0, \quad \boldsymbol{\gamma}^\top \cdot \mathbf{H}(\boldsymbol{\varphi}) = 0, \end{cases} \quad (3.1)$$

where $\boldsymbol{\gamma} \geq 0$ are new contact Lagrange multipliers and we have grouped in the force term $\mathbf{F}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}})$ damping, gravity, etc. Now the system is non-smooth, which is why we will need to use a first-order (implicit) integration scheme [67]. A simple model for friction can be introduced if we add yet another force of the form:

$$\mathbf{f}_\mu(\dot{\boldsymbol{\varphi}}) = -\mu\mathbf{V}(\dot{\boldsymbol{\varphi}})^\top\boldsymbol{\beta} \quad (3.2)$$

where $\mu > 0$ is a friction constant, $\boldsymbol{\beta}$ are new multipliers (one for each contact constraint) satisfying that they belong to the friction's cone, i.e. they satisfy component-wise $\beta_i \leq \|\nabla H_i(\boldsymbol{\varphi})^\top\boldsymbol{\gamma}_i\|$, and $\mathbf{V}(\dot{\boldsymbol{\varphi}})$ are unit

(relative) tangent velocities at the points of contact, i.e. for the case of a collision with a static obstacle:

$$kV_i(\dot{\boldsymbol{\varphi}}) = \dot{\boldsymbol{\varphi}} - \langle \dot{\boldsymbol{\varphi}}, \mathbf{n}_i \rangle \cdot \mathbf{n}_i,$$

where $\mathbf{n}_i = \frac{\nabla H_i(\boldsymbol{\varphi})}{\|\nabla H_i(\boldsymbol{\varphi})\|}$ and k is a normalization constant. For theoretical details and more sophisticated models for friction see [1].

Remark 3.2.1. We now list some implicit assumptions we are making in stating the collision model as Equation (3.1):

1. When $H = 0$ defines a surface (e.g. a plane or a sphere), the condition $\mathbf{H}(\boldsymbol{\varphi}) \geq 0$ means that for each node of the surface we impose

$$H_i(\boldsymbol{\varphi}) := H(p_i(t)) \geq 0.$$

This only forces the vertices of the cloth to be outside the obstacle (but there could be some penetrations of the faces). When the mesh is fine this is not really a problem, in the case of coarse meshes, one can add yet another constraint for the middle point of each face.

2. Signorini's condition implies that when there is no contact taking place, i.e. $H_i(\boldsymbol{\varphi}) > 0$, then there is no repulsive force acting, i.e. $\nabla H_i(\boldsymbol{\varphi})^\top \gamma_i = 0$. Therefore there is also no friction force acting, i.e. $\beta_i = 0$.
3. Without any other condition the multipliers β are not uniquely defined. A common approach is to assume that these multipliers cause *maximal dissipation* (see [59, 64, 105]). This amounts to solving a linear program. In practice, we will assume that $\beta_i = \|\nabla H_i(\boldsymbol{\varphi})^\top \gamma_i\|$ (which is anyways always the case when the tangent velocity is nonzero).
4. This model assumes that the collision is inelastic (there is no bouncing). This is a reasonable assumption for cloth; we will corroborate this in Chapter 4 when we perform the empirical validation of the collision model.

3.3 SELF-COLLISIONS

We now explain how to define the constraints H_k that account for modeling self-collisions of the cloth inside the function $\mathbf{H}(\boldsymbol{\varphi}) \geq 0$. In principle we need to integrate numerically the equations of motion and advance the simulation from $\boldsymbol{\varphi}^n$ to $\boldsymbol{\varphi}^{n+1}$, then check if in the process self-collisions took place, and in case they did, add new constraints H_k to the system and repeat the numerical integration. This process must be repeated until no new collisions are found. In practice this is costly and thus we will develop a more efficient procedure that

takes advantage of the way we integrate numerically the equations of motion. For the time being, assume we have both φ^n and φ^{n+1} (and their velocities) available to make computations.

3.3.1 Detection of self-collisions

In general, we assume that the cloth is triangulated (in case of a quadrangulation we can always divide the quads in two); then in case of collision, there are only two stable (i.e. detectable) possibilities: an edge-edge collision and a node-face collision. In these two cases, we have four nodes involved which at some instant of time belong to the same plane (see Figure 3.1). We must then only check if two co-planar edges cross or if a point belongs to a triangle. These two problems are readily solved using barycentric coordinates.

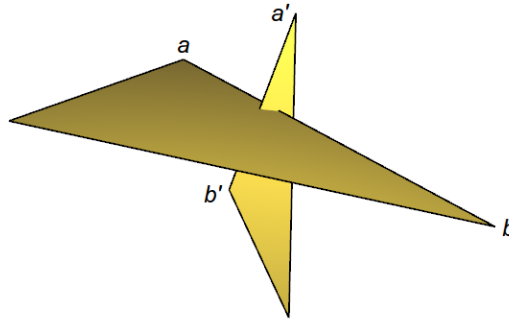


Figure 3.1: If the moving triangles were not intersecting before, then there exists a time in which the edges ab and $a'b'$ were co-planar.

Now we describe in more detail the process: in order to save computational time, we only check if a collision has happened for pairs of edges (or nodes and faces) that at time t_n are *sufficiently* close (and not for every pair, which would result in combinatorial explosion). To obtain this list of sufficiently close (up to some tolerance) pairs, there are several methods: one of the most widely used is the *hierarchical method* [92], where the mesh is divided into large regions, which are in turn also divided in smaller regions, up to a certain number of times (the number of hierarchies). Then one only checks if two pairs are sufficiently close when all of the larger regions containing these are close enough (e.g. by computing the distance between their center of masses), otherwise, they are discarded. Since our meshes are fairly coarse we avoid hierarchies (which can be cumbersome to implement) and compare the center of masses of our pairs in order to detect those that are candidates for collision.

Next, denoting by x_1, x_2, x_3, x_4 the position of the four *candidate* nodes at time t_n and by v_1, v_2, v_3, v_4 their velocities, we must only check if

$$\det(\tilde{x}_1 + t \cdot \tilde{v}_1, \tilde{x}_2 + t \cdot \tilde{v}_2, \tilde{x}_3 + t \cdot \tilde{v}_3) = 0$$

where $\tilde{x}_i = x_i - x_4$ and $\tilde{v}_i = v_i - v_4$, since $\boldsymbol{\varphi}^{n+1} = \boldsymbol{\varphi}^n + dt \cdot \dot{\boldsymbol{\varphi}}^{n+1}$. This is a cubic equation $a_3 t^3 + a_2 t^2 + a_1 t + a_0 = 0$ in t , with coefficients:

$$\begin{cases} a_3 = \det(\tilde{v}_1, \tilde{v}_2, \tilde{v}_3); \\ a_2 = \det(\tilde{x}_1, \tilde{v}_2, \tilde{v}_3) + \det(\tilde{v}_1, \tilde{x}_2, \tilde{v}_3) + \det(\tilde{v}_1, \tilde{v}_2, \tilde{x}_3); \\ a_1 = \det(\tilde{x}_1, \tilde{x}_2, \tilde{v}_3) + \det(\tilde{x}_1, \tilde{v}_2, \tilde{x}_3) + \det(\tilde{v}_1, \tilde{x}_2, \tilde{x}_3); \\ a_0 = \det(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3); \end{cases} \quad (3.3)$$

When $t \ll dt$ is small, the solution of the previous equation can be approximated linearly by $-\frac{a_0}{a_1}$. In any case, if there is a root for some $t_c \in [0, dt]$, we must then do two different calculations with the four co-planar points $y_i = x_i + t_c \cdot v_i$, namely:

1. Edge-edge case: say the endpoints of the first edge are y_1, y_2 and of the second y_3, y_4 , then we compute the real numbers α, β where $y_1 + \alpha(y_2 - y_1) = y_3 + \beta(y_4 - y_3)$ (the intersection of the two lines defined by the segments) and if they belong to the interval $[0, 1]$ a collision has occurred.
2. Node-face case: say the node is y_4 and the other 3 points are the corners of the triangle, then we compute the real numbers u, v, w such that $y_4 = uy_1 + vy_2 + wy_3$ and if they are in the interval $[0, 1]$ a collision has occurred.

3.3.2 Constraint definition for self-collisions

We now describe the computation of the self-collision constraint H_k . It will be linear in $\boldsymbol{\varphi}$ and naturally have slightly different forms depending on our two cases:

1. Edge-edge case:

$$H_k(\boldsymbol{\varphi}) := \langle \pi_\alpha(x_1, x_2) - \pi_\beta(x_3, x_4), \nu \rangle \geq 0,$$

where x_i are the four endpoints of the two edges, $\pi_\alpha(x_1, x_2) = (1 - \alpha)x_1 + \alpha x_2$ and $\pi_\beta(x_3, x_4) = (1 - \beta)x_3 + \beta x_4$ are the closest points between the two segments and ν is the normal vector to both edges. In general, the values ν, α, β vary with time. We will nevertheless assume that they are constant during the time-step, and compute them with the positions of the segments given by $\boldsymbol{\varphi}^{n+1}$. The normal vector ν is oriented such that $H_k(\boldsymbol{\varphi}^n) \geq 0$.

2. Node-face case:

$$H_k(\boldsymbol{\varphi}) := \langle x_4 - \pi(x_1, x_2, x_3), \nu \rangle \geq 0,$$

where x_4 is the node, x_i are the 3 corners of the triangle, again $\pi(x_1, x_2, x_3) = ux_1 + vx_2 + wx_3$ is the closest point inside the face to the node and ν is the normal vector to the triangle. In general, the values u, v, w vary with time. We will again assume that they are constant in time, and compute them with the positions given by φ^{n+1} . The normal vector ν is oriented such that $H_k(\varphi^n) \geq 0$.

Remark 3.3.1. Notice that:

1. By construction $H_k(\varphi^{n+1}) < 0$.
2. The constraint H_k is an approximation of the signed distance between the pairs edge-edge and node-face (only an approximation since ν and the barycentric coefficients are fixed in time).
3. Since in practice cloth has thickness, say τ_0 , the constraint we actually must impose is $H_k(\varphi) \geq \tau_0$.

3.3.3 Proximity constraints and cloth thickness

Adding the constraints we have just defined is enough to correct all present self-collisions. Nevertheless, there are two main drawbacks:

1. Efficiency: most cloth collisions can be avoided before they happen by adding preventive constraints.
2. Vibrations: since we are assuming that the cloth has a thickness $\tau_0 > 0$, when we integrate the system again and go from $H_k(\varphi^n) < 0$ to $H_k(\varphi^{n+1}) \geq \tau_0$, the change between the position of the nodes can be too large, and since our cloth is inextensible, this could create unwanted oscillations.

In order to avoid these two problems, we apply the detection procedure previously explained in 3.3.1 with one small difference: during the detection phase we move the pairs (edge-edge or face-node) closer, using their normal vectors and taking into account the thickness of the cloth, so that pairs that are too close and/or are approaching each other, are kept at a minimum distance of τ_0 before they actually cross. Since the restrictions we are considering are inequalities, we can add these to the system because they only affect the dynamics of cloth in case the constraint will actually get violated. In symbols, this means that we compute the coefficients (3.3) of the third degree polynomial using the altered positions given by $\hat{x}_i = x_i \pm \omega\tau_0\nu$, where ν is the unit normal vector (the cross product for the edge-edge case and the normal to the triangle for node-face case), $\omega \approx 0.5$ is what we will call a *proximity parameter* and the sign \pm is chosen so that the pairs approach each other. Afterwards the response constraint H_k is calculated as usual (i.e. the normals and the barycentric coordinates) with the unaltered positions x_i given by φ^{n+1} .

Remark 3.3.2. It is usually enough to use the positions $\hat{x}_i = x_i \pm \omega\tau_0\nu$, where $\omega \approx 0.5$ to detect all self-collisions, nevertheless some can sometimes be missed because the nodes have moved too much. In that case, we enter an iterative process reducing gradually the value of ω until all are resolved. We will explain this in more detail in Section 3.4.1.

Definition 5. (Self-collision constraints). We will denote by

$$\mathcal{C} = \text{Collisions}_\omega \left(\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}^{n+1} \right)$$

the set of self-collisions constraints that must be imposed from the state $\boldsymbol{\varphi}^n$ to the state $\boldsymbol{\varphi}^{n+1}$ with proximity parameter $\frac{1}{2} > \omega \geq 0$.

3.4 NUMERICAL INTEGRATION OF THE SYSTEM

The friction force and the contact constraints introduced in the previous section are in general highly non-linear and stiff and thus must be integrated implicitly (like the inextensibility constraints, see Section 2.6.1). To integrate the system numerically from time t_n to t_{n+1} (i.e. to advance the simulation from $\boldsymbol{\varphi}^n$ to $\boldsymbol{\varphi}^{n+1}$), we perform as before an iterative process $\boldsymbol{\varphi}_{j+1} = \boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}$ where the initial point is the unconstrained step $\boldsymbol{\varphi}_0 = \boldsymbol{\varphi}_0^{n+1}(\boldsymbol{\varphi}^n, \dot{\boldsymbol{\varphi}}^n)$ given by an implicit Euler scheme. Also, we write:

$$\mathbf{H}(\boldsymbol{\varphi}_{j+1}) = \mathbf{H}(\boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}) \simeq \mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1},$$

and similarly

$$\mathbf{C}(\boldsymbol{\varphi}_{j+1}) = \mathbf{C}(\boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}) \simeq \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla\mathbf{C}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1},$$

and then solve iteratively the following sequence of quadratic programs with linear equality and inequality constraints:

$$\begin{cases} \min_{\Delta\boldsymbol{\varphi}_{j+1}} \frac{1}{2}\Delta\boldsymbol{\varphi}_{j+1}^\top \cdot \mathbf{M} \cdot \Delta\boldsymbol{\varphi}_{j+1} - \Delta\boldsymbol{\varphi}_{j+1}^\top \cdot \mathbf{f}_\mu(\dot{\boldsymbol{\varphi}}_j) \\ \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla\mathbf{C}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} = 0, \\ \mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} \geq 0, \end{cases} \quad (3.4)$$

where

1. $\dot{\boldsymbol{\varphi}}_{j+1} = \frac{\boldsymbol{\varphi}_{j+1} - \boldsymbol{\varphi}^n}{dt}$ is an approximation of $\dot{\boldsymbol{\varphi}}^{n+1}$,
2. $\mathbf{f}_\mu(\dot{\boldsymbol{\varphi}}_j) = -\mu\mathbf{V}(\dot{\boldsymbol{\varphi}}_j)^\top \Delta\boldsymbol{\beta}_j$ is the friction force at iteration j ,
3. $\mathbf{V}(\dot{\boldsymbol{\varphi}}_j)$ are the relative unit tangent velocities,
4. $(\Delta\boldsymbol{\beta}_j)_i = \|\nabla H_i(\boldsymbol{\varphi}_j)^\top (\Delta\boldsymbol{\gamma}_j)_i\|$ is the magnitude of the contact forces at iteration j ,

5. and $\Delta\gamma_j \geq 0$ are the multipliers associated to the contact constraints.

We iterate until

$$\max |\mathbf{C}(\boldsymbol{\varphi}_j)| < \epsilon_0, \quad \min \mathbf{H}(\boldsymbol{\varphi}_j) \geq -\epsilon_1, \quad \max |\Delta\boldsymbol{\varphi}_j| < \epsilon_2 \quad (3.5)$$

for some tolerances $\epsilon_0, \epsilon_1, \epsilon_2 > 0$. This third condition ensures that the friction force has stabilized. Note that the critical points of the previous quadratic problems (3.4) are:

$$\begin{cases} \mathbf{M} \cdot \Delta\boldsymbol{\varphi}_{j+1} = -\nabla\mathbf{C}(\boldsymbol{\varphi}_j)^\top \Delta\boldsymbol{\lambda}_{j+1} + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)^\top \Delta\boldsymbol{\gamma}_{j+1} - \mu \mathbf{V}(\dot{\boldsymbol{\varphi}}_j)^\top \Delta\boldsymbol{\beta}_j, \\ \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla\mathbf{C}(\boldsymbol{\varphi}_j) \Delta\boldsymbol{\varphi}_{j+1} = 0, \\ \mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j) \Delta\boldsymbol{\varphi}_{j+1} \geq 0, \\ \Delta\boldsymbol{\gamma}_{j+1} \geq 0, \quad \Delta\boldsymbol{\gamma}_{j+1}^\top \cdot [\mathbf{H}(\boldsymbol{\varphi}_j) + \nabla\mathbf{H}(\boldsymbol{\varphi}_j) \Delta\boldsymbol{\varphi}_{j+1}] = 0, \\ (\Delta\boldsymbol{\beta}_j)_i = \|\nabla H_i(\boldsymbol{\varphi}_j)^\top (\Delta\boldsymbol{\gamma}_j)_i\|. \end{cases} \quad (3.6)$$

Remark 3.4.1. In order to integrate friction force we have made the approximation $\mathbf{f}_\mu(\dot{\boldsymbol{\varphi}}_{j+1}) \simeq \mathbf{f}_\mu(\dot{\boldsymbol{\varphi}}_j)$. That is, we have dropped the gradient we would normally have with a first-order approximation (this is what we also do with the gradient of the constraint forces in the first equation of (3.6)).

3.4.1 Addition of self-collision constraints

Instead of checking and generating all self-collision constraints only with the states $\mathcal{C} = \text{Collisions}_\omega(\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}^{n+1})$, we take advantage of the fact that we perform an iteration process. We now explain how we introduce self-collisions into the sequence of problems (3.4) for every step. For every iteration j we check for self-collisions (see Section 3.3.1) taking into account the thickness of the cloth (Section 3.3.3) between the states $\boldsymbol{\varphi}^n$ and $\boldsymbol{\varphi}_j$ and generate the corresponding constraints (Section 3.3.2). In symbols this means that all the constraints $\mathcal{C}_j = \text{Collisions}_\omega(\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}_j)$ for $j \geq 0$ and $\omega \approx 0.5$ are added to the system. In the rare case that the same collision is found in two different iterations we only keep the constraint defined by the later iteration. Then, when we find a state $\boldsymbol{\varphi}_{j+1}$ that satisfies the stopping criteria (3.5), we check for self-collisions with $\omega = 0$, and in case no self-collision is detected, we put $\boldsymbol{\varphi}^{n+1} = \boldsymbol{\varphi}_{j+1}$. Otherwise, we repeat the whole iteration process with a smaller value of ω (see Remark 3.3.2).

3.5 EFFICIENT SOLUTION OF THE QUADRATIC PROBLEMS

Definition 6 (Active constraint). In a constrained optimization problem (such as (3.4)), we say that an inequality constraint $g(x) \geq 0$ is active at a feasible point y if $g(y) = 0$. Note that all equality (in our case inextensibility) constraints are always active.

In order to solve the sequence of problems (3.4) we could employ any quadratic problem solver, but we would not be taking advantage of the structure of our problem. That is, if in one of the iterations j one of the contact constraints H_i is active (see the previous definition), then it is likely that it will be active again at the next iteration. Physically, this means that nodes of the cloth that are in contact with an obstacle (or among themselves) at some iteration, are likely to remain in contact. This suggests the use of active-set-methods [86] to solve the quadratic problems. We will develop a novel *active-set* algorithm in the following pages. Although we could use one of the many existing ones, they always require that one begins with a feasible (albeit not optimal) solution to the problem. Our method will not have this requirement.

The main idea of active set methods is to find the *active set* of constraints at the solution, because then, once known, the program can be solved by ignoring inactive constraints, and assuming that all active inequality constraints are equality constraints. Recall that solving quadratic problems with equality constraints is very cheap and can be done by solving a linear system (see [44]). This will be precisely what we will do for every iteration of the sequence (3.4). In order to find the active set, one splits the constraints in two sets:

The working set, \mathcal{W} : these are the constraints believed to be active ($g = 0$) and therefore are imposed as equality constraints when one solves the optimization problem. This can be initialized as the set consisting only of equality constraints.

The observation set, \mathcal{O} : these are the constraints believed to be inactive ($g > 0$) and therefore are not imposed as equality constraints. Since they are not included in the problem one must be careful that they do not become violated.

Then one proceeds as follows:

- 1) solve the equality problem defined by the working set;
- 2) compute the Lagrange multipliers of the working set for the inequality constraints;
- 3) send a subset of the constraints with negative Lagrange multipliers to the observation set;
- 4) if all multipliers are positive, check if all constraints in the observation set remain feasible;

- 5) send a subset of the infeasible constraints to the working set;
- 6) repeat.

Then, if at some iteration we have found an increment $\Delta\boldsymbol{\varphi}_{j+1}$ such that all contact constraints in the working set have positive Lagrange multipliers $\Delta\gamma_{j+1} \geq 0$ (see Equation (3.6)) and all constraints in the observation set are not violated, we have found the active set (see [86]) and we can make the update $\boldsymbol{\varphi}_{j+1} = \boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}$. The following proposition ensures that we do not enter in a never-ending cycle:

Proposition 1 (Entry and exit of constraints). Given the system (with unknowns $\Delta\boldsymbol{\varphi}$)

$$\begin{cases} \mathbf{M}\Delta\boldsymbol{\varphi} = \nabla\mathbf{H}(\boldsymbol{\varphi})^\top\Delta\boldsymbol{\gamma}, \\ \mathbf{H}(\boldsymbol{\varphi}) + \nabla\mathbf{H}(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} = 0, \end{cases} \quad (3.7)$$

and the system (with unknowns $\Delta\tilde{\boldsymbol{\varphi}}$)

$$\begin{cases} \mathbf{M}\Delta\tilde{\boldsymbol{\varphi}} = \nabla\mathbf{H}^{-k}(\boldsymbol{\varphi})^\top\Delta\tilde{\boldsymbol{\gamma}}, \\ \mathbf{H}^{-k}(\boldsymbol{\varphi}) + \nabla\mathbf{H}^{-k}(\boldsymbol{\varphi})\Delta\tilde{\boldsymbol{\varphi}} = 0, \end{cases} \quad (3.8)$$

where we have removed the constraint $H_k(\boldsymbol{\varphi}) + \nabla H_k(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} = 0$ from the first system; then it holds that

$$\Delta\gamma_k \cdot (H_k(\boldsymbol{\varphi}) + \nabla H_k(\boldsymbol{\varphi})\Delta\tilde{\boldsymbol{\varphi}}) \leq 0, \quad (3.9)$$

where $\Delta\gamma_k$ are the Lagrange multipliers of the removed constraint k .

Proof. Subtracting the first two equations of the systems, we get:

$$\mathbf{M}(\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi}) = \nabla\mathbf{H}^{-k}(\boldsymbol{\varphi})^\top(\Delta\tilde{\boldsymbol{\gamma}} - \Delta\boldsymbol{\gamma}^{-k}) - \nabla H_k(\boldsymbol{\varphi})^\top\Delta\boldsymbol{\gamma}_k.$$

Then, multiplying both sides by $(\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi})^\top$, we deduce that

$$0 \leq (\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi})^\top \cdot \mathbf{M} \cdot (\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi}) = 0 - (\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi})^\top \cdot \nabla H_k(\boldsymbol{\varphi})^\top \Delta\boldsymbol{\gamma}_k,$$

since $(\Delta\tilde{\boldsymbol{\varphi}} - \Delta\boldsymbol{\varphi})^\top \cdot \nabla\mathbf{H}^{-k}(\boldsymbol{\varphi})^\top = \mathbf{H}^{-k}(\boldsymbol{\varphi})^\top - \mathbf{H}^{-k}(\boldsymbol{\varphi})^\top = 0$. Finally, using that $\nabla H_k(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} = -H_k(\boldsymbol{\varphi})$, and rearranging terms we get

$$0 \leq -\Delta\tilde{\boldsymbol{\varphi}}^\top \nabla H_k(\boldsymbol{\varphi})^\top \Delta\boldsymbol{\gamma}_k - H_k(\boldsymbol{\varphi})\Delta\boldsymbol{\gamma}_k.$$

From here (3.9) follows easily. \square

Corollary 1. If a constraint in the working set has a negative Lagrange multiplier, when it is taken out of the system and put in the observation set, it becomes feasible. Conversely, when a constraint in the observation set is infeasible and we sent it to the active set, its associated Lagrange multiplier is positive.

Remark 3.5.1. The heuristic that is usually followed to decide which constraint to remove or to add is: delete from the working set the constraint with the most negative Lagrange multiplier and add to the working set the constraint from the observational set that is being most violated (the most negative one).

To finish this section we study the case of linearly dependent constraints. This is relevant since in general, we do not want to introduce linearly dependent constraints into the system because they give raise to (near) singular matrices.

Lemma 1. If a constraint G in the observation set can be written as a linear combination of constraints of the working set, i.e. $G(\boldsymbol{\varphi}) = \sum \alpha_i H_i(\boldsymbol{\varphi})$, then the linearized constraint is feasible $G(\boldsymbol{\varphi}) + \nabla G(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} \geq 0$.

Proof.

$$G(\boldsymbol{\varphi}) + \nabla G(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} = G(\boldsymbol{\varphi}) + \sum \alpha_i \nabla H_i(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} = G(\boldsymbol{\varphi}) - \sum \alpha_i H_i(\boldsymbol{\varphi}) = 0.$$

□

Remark 3.5.2. The previous lemma ensures that in general, we do not send linearly dependent constraints from the observation set to the working set. Nevertheless, it is possible to have a degenerate case where the constraints are not linearly dependent but their gradients are. In symbols, this would mean that a constraint in the observation set satisfies $G(\boldsymbol{\varphi}) + \nabla G(\boldsymbol{\varphi})\Delta\boldsymbol{\varphi} \leq 0$ and moreover $\nabla G(\boldsymbol{\varphi}) = \sum \alpha_i \nabla H_i(\boldsymbol{\varphi})$. What we do then is to introduce G in the working set while removing the H_i with the greatest $\alpha_i \neq 0$ in absolute value. This new working set is linearly independent (otherwise it would contradict the assumption that the original working set without G was linearly independent) and the process can continue.

3.5.1 Factorization of the matrix system

Every time that a constraint goes from the working set to the observation set (or viceversa), i.e. when the index sets \mathcal{W} and \mathcal{O} are updated, the Lagrange multipliers must be recomputed, i.e. a linear system must be solved in order to find the solution of (3.10).

$$\begin{cases} \mathbf{M} \cdot \Delta\boldsymbol{\varphi}_{j+1} = -\nabla\mathbf{C}(\boldsymbol{\varphi}_j)^\top \Delta\boldsymbol{\lambda}_{j+1} + \nabla\mathbf{H}(\boldsymbol{\varphi}_j)^\top \Delta\boldsymbol{\gamma}_{j+1} - \mu\mathbf{V}(\boldsymbol{\varphi}_j)^\top \Delta\boldsymbol{\beta}_j, \\ \mathbf{C}(\boldsymbol{\varphi}_j) + \nabla\mathbf{C}(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} = 0, \\ H_i(\boldsymbol{\varphi}_j) + \nabla H_i(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} = 0 \text{ for } i \in \mathcal{W}. \end{cases} \quad (3.10)$$

In order to ease readability we will include inextensibility constraints and the contact constraints of the working set in only one

function denoted by $\mathbf{G}^\top = [\mathbf{C}^\top, \mathbf{H}^\top]$. Now, since in general only one constraint will be entering or exiting at the time, the linear systems that we have to solve are almost identical with the exception of a few rows and columns. That is why, the use of factorizations becomes an important tool to achieve efficiency. The linear system we need to solve to find the multipliers is:

$$\left(\nabla \mathbf{G}(\boldsymbol{\varphi}_j) \mathbf{M}^{-1} \nabla \mathbf{G}(\boldsymbol{\varphi}_j)^\top \right) \Delta \boldsymbol{\zeta}_{j+1} = -\mathbf{G}(\boldsymbol{\varphi}_j) - \nabla \mathbf{G}(\boldsymbol{\varphi}_j) \mathbf{M}^{-1} \mathbf{f}_\mu(\boldsymbol{\varphi}_j),$$

where $\Delta \boldsymbol{\zeta}_{j+1}^\top = [\Delta \boldsymbol{\lambda}_{j+1}^\top, \Delta \boldsymbol{\gamma}_{j+1}^\top]$. Hence the system matrix (let us call it \mathbf{A}) is positive definite (since \mathbf{M} is positive definite because it is the mass matrix); therefore we can use Cholesky decomposition [45], provided our constraints are linearly independent (see again Lemma 1 and Remark 3.5.2). This is a factorization of the form

$$\mathbf{Q} \mathbf{A} \mathbf{Q}^\top = \mathbf{L} \mathbf{L}^\top, \quad (3.11)$$

where \mathbf{L} is an invertible lower triangular matrix and \mathbf{Q} is an (orthonormal) permutation matrix (this is done in order to take advantage of sparsity patterns). Solving the linear system in this way reduces to a couple of triangular substitutions and two matrix multiplications:

$$\mathbf{A}^{-1} = (\mathbf{Q}^\top \mathbf{L} \mathbf{L}^\top \mathbf{Q})^{-1} = \mathbf{Q}^\top \mathbf{L}^{-1} (\mathbf{L}^\top)^{-1} \mathbf{Q},$$

since \mathbf{L} and \mathbf{L}^\top are lower and upper triangular, and we can use forward and backward substitutions (no matrix is actually inverted).

3.5.2 Updates of the Cholesky decomposition

Every time a constraint enters or exits the working set, \mathbf{L} (and \mathbf{Q}) can be efficiently updated without recomputing the factorization from scratch (see, e.g. [28, 102]). In the following, we describe two simple methods (not necessarily the most efficient) to perform such a task. When introducing constraints, we will do so, only one at a time, whereas to remove them we will usually delete all constraints with negative Lagrange multipliers. Assume first that we want to add a single contact constraint H_k to the working set. Then the updated matrices are:

$$\mathbf{L}_+ = \begin{bmatrix} \mathbf{L} & \vec{0} \\ \vec{l}^\top & \lambda \end{bmatrix}, \quad \mathbf{Q}_+ = \begin{bmatrix} \mathbf{Q} & \vec{0} \\ \vec{0} & 1 \end{bmatrix}, \quad (3.12)$$

where \vec{l} is found by solving $\mathbf{L} \cdot \vec{l} = \mathbf{Q} \cdot \nabla \mathbf{H} \cdot \mathbf{M}^{-1} \cdot (\nabla H_k)^\top$ and

$$\lambda = \sqrt{\nabla H_k \cdot \mathbf{M}^{-1} \cdot (\nabla H_k)^\top - \vec{l}^\top \cdot \vec{l}}$$

This can be easily seen by writing the expanded equations for:

$$\mathbf{Q}_+ \mathbf{A}_+ \mathbf{Q}_+^\top = \mathbf{L}_+ \mathbf{L}_+^\top, \quad (3.13)$$

where

$$\mathbf{A}_+ = \begin{bmatrix} \nabla \mathbf{G} \\ \nabla H_k \end{bmatrix} \cdot \mathbf{M}^{-1} \cdot \begin{bmatrix} \nabla \mathbf{G} \\ \nabla H_k \end{bmatrix}^\top. \quad (3.14)$$

Remark 3.5.3. The case when $\lambda \leq 0$ occurs when we try to introduce a linearly dependent constraint into the system. We use this as a test to check if we must interchange constraints between the working and observation set as explained in Remark 3.5.2.

On the other hand, updating \mathbf{L} when taking out an equation is more complicated since when we remove a row (or a column) from \mathbf{L} it loses its triangular form. There are several methods to accomplish this update without recomputing everything from scratch, among them: using low-rank downdates [45], using artificial multipliers to set to zero the solution's coordinates corresponding to exiting constraints or employing Givens' rotations (also known as Householder transforms, see [102]). Among these three methods, we have found that the most practical and fastest for vectorized languages such as Python or MATLAB is the artificial multipliers method, which we now describe. Assume we have the system:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where every coordinate k of \mathbf{x} is the Lagrange multiplier corresponding to a constraint G_k . Then, if we want to remove the constraints i_1, \dots, i_n from the system, we set their values to zero and hence solve the system

$$\begin{cases} \mathbf{A}\mathbf{x} + \mathbf{S}^\top \mathbf{y} = \mathbf{b}, \\ \mathbf{S}\mathbf{x} = \vec{0}, \end{cases} \quad (3.15)$$

where $S_k \cdot \mathbf{x} = x_{i_k}$. This is the case since equations (3.15) are the critical points of:

$$\begin{cases} \min_{\mathbf{z}} \frac{1}{2} \mathbf{z}^\top \mathbf{A} \mathbf{z} - \mathbf{z}^\top \mathbf{b} \\ \mathbf{S} \mathbf{z} = \vec{0}. \end{cases}$$

Remark 3.5.4. As already mentioned, recomputing from scratch the full Cholesky factorization is usually not desirable, especially in cases with very fine meshes. Nevertheless, when the number of variables is not too high, it can be competitive in the case of the removal of constraints, since we can delete more than one condition a time without compromising linear independence (as opposed to the case of adding constraints, where adding more than one at the time can be problematic). With this method, the heuristic used is that all constraints with negative Lagrange multipliers are removed from the working set.

3.5.3 Detailed algorithm for collisions

To finish this section we give a detailed description of the full numerical algorithm written in pseudo-code in Algorithm 2.

Algorithm 2 Collisions active-set algorithm

Require: $\boldsymbol{\varphi}^n, \dot{\boldsymbol{\varphi}}^n$

```

1:  $\boldsymbol{\varphi}_0 \leftarrow \text{unconstrained}(\boldsymbol{\varphi}^n, \dot{\boldsymbol{\varphi}}^n, \dots)$   $\triangleright$  (Implicit Euler integration step)
2:  $\mathcal{C}_0 = \text{Collisions}_\omega(\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}_0)$   $\triangleright$  (Self-collision constraints)
3:  $\mathcal{W} \leftarrow \{i : G_i(\boldsymbol{\varphi}^n) = 0\}$ ,  $\mathcal{O} \leftarrow \mathcal{W}^c$   $\triangleright$  (Working and observation sets)
4:  $\mathbf{J} \leftarrow \text{gradient}(\boldsymbol{\varphi}_0, \mathcal{C}_0, \mathcal{W}, \dots)$   $\triangleright$  (i.e.  $\nabla G_i : i \in \mathcal{W}$ )
5:  $\mathbf{L} \leftarrow \text{cholesky}(\mathbf{J} \cdot \mathbf{M}^{-1} \cdot \mathbf{J}^\top)$ 
6: while  $\max |\mathbf{C}(\boldsymbol{\varphi}_j)| \geq \epsilon_0$  or  $\min \mathbf{H}(\boldsymbol{\varphi}_j) \leq -\epsilon_1$  do
7:    $[\Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\gamma}] = \text{multipliers}(\boldsymbol{\varphi}_j, \mathbf{L})$ 
8:   if  $\min(\Delta\boldsymbol{\gamma}) \geq 0$  then
9:      $\Delta\boldsymbol{\varphi}_{j+1} \leftarrow \text{increment}(\Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\gamma}, \mathbf{J})$ 
10:    if  $H_i(\boldsymbol{\varphi}_j) + \nabla H_i(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1} \geq 0$  for  $i \in \mathcal{O}$  then
11:       $\boldsymbol{\varphi}_{j+1} \leftarrow \boldsymbol{\varphi}_j + \Delta\boldsymbol{\varphi}_{j+1}$ 
12:       $\mathcal{C}_{j+1} = \text{Collisions}_\omega(\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}_{j+1})$ ,  $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{C}_{j+1}$ 
13:       $\mathbf{J} \leftarrow \text{gradient}(\boldsymbol{\varphi}_{j+1}, \cup_j \mathcal{C}_j, \mathcal{W}, \dots)$ 
14:       $\mathbf{L} \leftarrow \text{cholesky}(\mathbf{J} \cdot \mathbf{M}^{-1} \cdot \mathbf{J}^\top)$ 
15:    else
16:       $i_{in} \leftarrow \text{indmin}_{i \in \mathcal{O}}(H_i(\boldsymbol{\varphi}_j) + \nabla H_i(\boldsymbol{\varphi}_j)\Delta\boldsymbol{\varphi}_{j+1})$ 
17:       $\mathcal{O} \leftarrow \mathcal{O} \setminus i_{in}$ ,  $\mathcal{W} \leftarrow \mathcal{W} \cup i_{in}$ 
18:       $\mathbf{L} \leftarrow \text{update}(\mathbf{L}, i_{in})$ 
19:    end if
20:    else
21:       $i_{out} \leftarrow \text{indmin}(\Delta\boldsymbol{\gamma})$ 
22:       $\mathcal{O} \leftarrow \mathcal{O} \cup i_{out}$ ,  $\mathcal{W} \leftarrow \mathcal{W} \setminus i_{out}$ 
23:       $\mathbf{L} \leftarrow \text{update}(\mathbf{L}, i_{out})$ 
24:    end if
25:  end while
26:  $\boldsymbol{\varphi}^{n+1} \leftarrow \boldsymbol{\varphi}_{j+1}$ ,  $\dot{\boldsymbol{\varphi}}^{n+1} \leftarrow \frac{\boldsymbol{\varphi}^{n+1} - \boldsymbol{\varphi}^n}{dt}$ 
27: return  $\boldsymbol{\varphi}^{n+1}, \dot{\boldsymbol{\varphi}}^{n+1}$ 

```

Remark 3.5.5. We now make some comments about Algorithm 2:

- 1 The working set is always initialized at least with the inextensibility constraints, but we can also add the active contact constraints from the previous time-step t_{n-1} .
- 2 We have not explicitly written the friction force, but it obviously comes up in the computation of the multipliers and the increment (lines 7 and 9, see Equations (3.10)).
- 3 Note that after successfully finding the active set (line 10), for the next step $j + 1$, we do not change the working set \mathcal{W} . Only the

observation set \mathcal{O} is updated with the possible new self-collision constraints found (line 12).

- 4 When there is a negative multiplier (line 20), note that we take out from the system the constraint with the smallest (most negative) multiplier. Likewise, when one of the constraints in the observation set must be introduced (line 15), we choose the one that is being most violated.

3.5.4 *Similarities and differences with common active-set methods*

Now that we have presented the full algorithm we use to solve the quadratic problems, we can talk in more detail about how it compares to standard active-set methods like the one described in [86]. The main difference was already mentioned: to our knowledge, all active-set methods require that one begins at a feasible point and then iterates from there. This has the disadvantage that one must find a feasible point to begin with, e.g. solving a linear program with equality and inequality constraints. On the other hand, those classic methods have the advantage that all the constraints in the observation set are kept non-violated, and this allows one to take smaller steps towards the solution when all the multipliers are positive (potentially causing the algorithm to converge faster). Since we are solving so many relatively large sparse quadratic problems in a row, we have found that the requirement of starting at a feasible point is way more expensive than employing the novel algorithm here presented. This is the case because lower-rank updates of sparse Cholesky decompositions can be carried out very efficiently.

3.6 EVALUATION AND RESULTS

In this section, we present several experiments that put to the test our collision model. They will be qualitative in nature, i.e. we only show that our simulator is capable of dealing with such scenarios. We will show that our modelization of friction is effective in static (cylinder experiment, Section 3.6.1) and dynamic (rotating sphere experiment, Section 3.6.2) settings, that we can easily include collisions with sharp objects (Section 3.6.3) and that we can simulate complicated folding sequences of cloth with non-trivial topologies (shorts experiment, Section 3.6.4). The second and the third experiments, are challenging scenarios suggested by [70] as benchmarking tests for a robust cloth collision model. All the videos of the experiments can be visualized online at <https://youtube.com> (in each section we will give the appropriate link for the relevant video). In Chapter 4, Section 4.3 we will perform several quantitative experiments where we will compare the collision model with real experimental data.

3.6.1 Frictional cylinder: https://youtu.be/_nh-ejHcJAg

This is the most basic of the four experiments: a flat sheet of cloth falls on top of a frictional cylinder during 2 seconds. In Figure 3.2 we show the final configuration of the textile for $t = 2$. All the physical parameters are kept constant but friction, which varies among $\mu \in \{0.2, 0.4, 0.55\}$. The cylinder is 30 cm off-center (with respect to the center of mass of the cloth whose measures are 130 cm \times 130 cm) and the textile is slightly rotated (20 degrees with respect to the z-axis). This means that in the absence of friction (or with a small friction coefficient), the cloth collides with the cylinder and then falls to the floor. We show with this scenario that our implementation of friction is effective and can handle scenarios with persistent contact. In the first

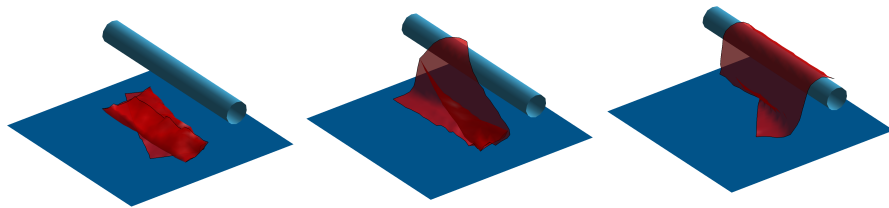


Figure 3.2: Final frame ($t = 2$ seconds) of 3 separate simulations of the fall of a sheet of cloth on top of an off-center cylinder. All the physical parameters are kept constant but friction, which varies among $\mu \in \{0.2, 0.4, 0.55\}$.

image of Figure 3.2, we have that $\mu = 0.2$ is too small and therefore the cloth falls onto the floor. In the second image, the friction $\mu = 0.4$ is somehow bigger and the cloth can be seen still in the process of falling but at a later stage, which shows that the friction forces have acted and delayed the fall. Finally, in the third image with $\mu = 0.55$, the friction is high enough so that the sheet lies stably on top of the cylinder.

3.6.2 Rotating sphere: <https://youtu.be/-XS3pKpoVbA>

In this second experiment, we simulate the collision of a sheet of cloth with a frictional sphere and the floor. The cloth measures 190 cm \times 190 cm whereas the sphere has a radius of 35 cm. One second after the textile has fallen, the sphere performs half a rotation along the z-axis during one second. The discrepancy in size is intentional so that after the fall the textile is also in contact with the floor and can then wrap around the sphere. In Figure 3.3 we can see the final frame of the simulation at $t = 2.5$ seconds; in the top image a small amount of shearing is allowed (following the shearing model described in Section 2.4.6) and in the bottom one all the inextensibility constraints are enforced. The rest of the parameters are all kept constant.

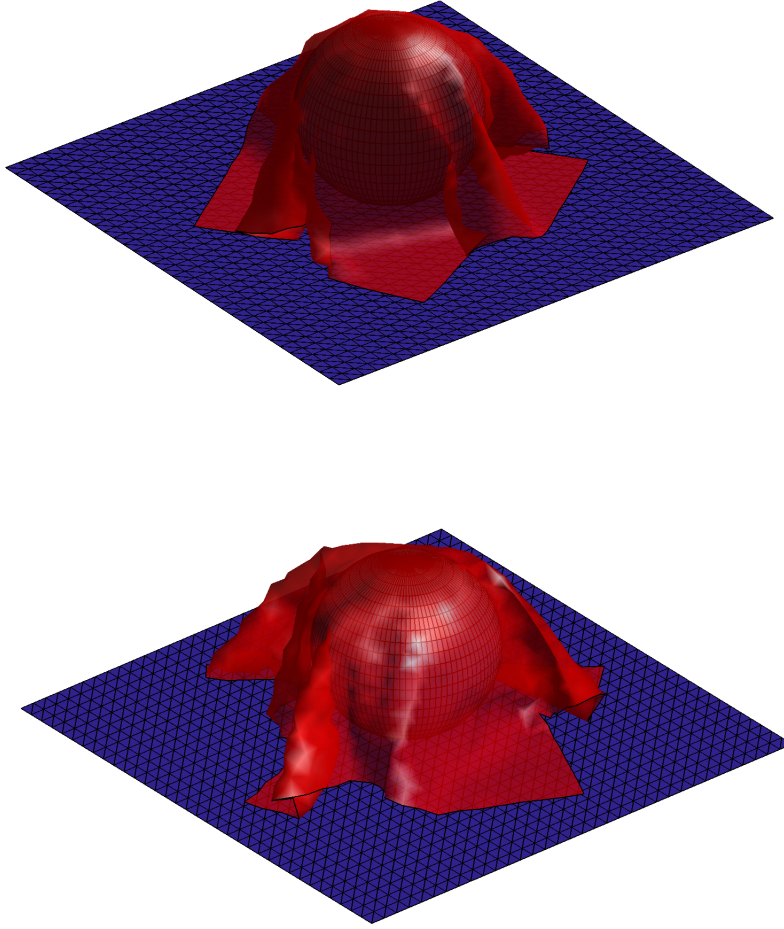


Figure 3.3: Final frame ($t = 2.5$ seconds) of 2 different simulations of the collision of a sheet of cloth with a frictional sphere and the floor. One second after the textile has fallen, the sphere performs half a rotation along the z -axis during one second. All the physical parameters are kept constant but in the top image, a small amount of shearing is allowed while in the bottom one, all the inextensibility constraints are enforced.

In order to simulate properly the dynamical friction between the cloth and the sphere, we must take into account the speed of the rotation for the points of the mesh that are in contact with the sphere. In practice this means that when computing at every iteration j of our solver the tangent velocities, we must account for a new term:

$$\mathbf{V}(\dot{\boldsymbol{\phi}}_j) = \dot{\boldsymbol{\phi}}_j - \langle \dot{\boldsymbol{\phi}}_j, \mathbf{n} \rangle \cdot \mathbf{n} - \mathbf{v}^{sphere},$$

where \mathbf{n} is the outwards normal to the sphere and \mathbf{v}^{sphere} is its speed at the current time step (as before the tangent velocities are afterwards

normalized).

In order to obtain an interesting behavior of the simulation, it is important to calibrate carefully the interplay between the friction with the floor and with the sphere. We select $\mu_{sphere} = 0.5$ and $\mu_{floor} = 0.4$, so that the cloth follows the rotation of the sphere but with considerable resistance from the floor. On the other hand, despite the fact that the shearing constant $k_s > 0$ was chosen so that the mean area errors during both simulations are basically the same (1.3431% for the inextensible vs 1.3520% for the other), we can see clearly in Figure 3.3 that the qualitative behavior is very different, one resembling a stiffer and heavier material (e.g. a towel or a stiff napkin), whereas the other seems lighter and smoother.

3.6.3 Collision with a sharp obstacle: https://youtu.be/z7L_0_nSfrM

In this third experiment, we simulate the collision of a piece of cloth with a collection of needle-like obstacles. They are given by the set of implicit equations:

$$\mathbf{H}(\boldsymbol{\varphi}) = c_1 c_2 \mathbf{z} - \sin(c_1 \mathbf{x}) \sin(c_1 \mathbf{y}), \quad (3.16)$$

where we take $c_1 = 20$ and $c_2 = 0.075$ (see Figure 3.4). The interest of this scenario lies in the fact that it is not enough to impose the previous equation (3.16) as a hard constraint (like we did with the sphere and the cylinder); but that we need in addition to take into account the cusps of the surface. These difficulties are typical for most physical simulators and they arise when the obstacles we are simulating present characteristics of lower dimensional objects (e.g. a really thin cylinder or the cusps in this case). It is easy to see that the cusps are given by:

$$x = \frac{2\pi m \pm \frac{\pi}{2}}{c_1}, \quad y = \frac{2\pi m \pm \frac{\pi}{2}}{c_1}, \quad z = \frac{1}{c_1 c_2}, \quad (3.17)$$

where $m \in \mathbb{Z}$.

Let us denote them by $\{q_1, \dots, q_f\}$. Then, for every iteration j of the solver, similarly like we do with self-collisions, we must check if a collision occurred during the motion $\boldsymbol{\varphi}^n \rightarrow_{dt} \boldsymbol{\varphi}_j$ between these cusps and the (triangular) faces of our meshed cloth.

This means that for every detected collision, in the next iteration $j + 1$ we must add a constraint of the form:

$$\langle q_i - \pi(x_1, x_2, x_3), \nu \rangle \geq 0,$$

where q_i is the corresponding cusp, x_i are the 3 corners of the triangle, $\pi(x_2, x_3, x_4) = ux_1 + vx_2 + wx_3$ is the closest point between the face and q_i , and ν is the normal vector to the triangle.

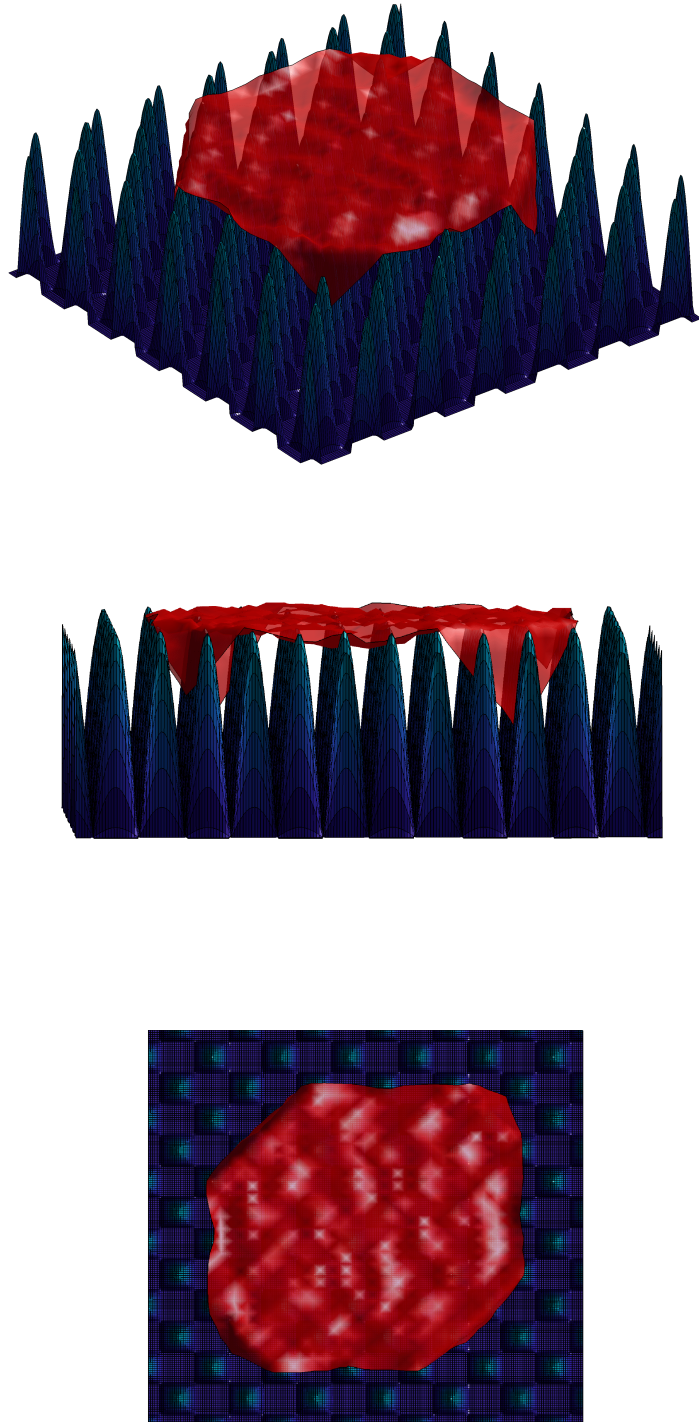


Figure 3.4: Simulated final frame of cloth's collision with a collection of needle-like obstacles seen from 3 different angles. The cusps must be taken into account separately from the rest of the surface and are treated with the same algorithm we treat self-collisions.

The values v, u, v, w are (like in the case for self-collisions) constant in time, and are computed with the positions given by φ_j . The normal vector v is oriented such that $H_i(\varphi^n) \geq 0$.

Remark 3.6.1. As with self-collisions we consider cloth's thickness in practice by imposing $H_k(\varphi) \geq \tau_0 > 0$. Moreover, as before this thickness is taken into account in the detection process (see Section 3.3.3).

In Figure 3.4 we can observe the result of the simulation from three different viewpoints. The cloth lies stably on top of the cusps without any noticeable artifact. In Chapter 4, we will encounter again sharp objects when we simulate the hitting of a piece of cloth with a thin stick.

3.6.4 *Folding sequence of short pants:* <https://youtu.be/2gdnjUICb0g>

In this final experiment, we simulate the dynamical folding of a pair of shorts (the same ones of Figure 2.2) on top of a table. In order to do so, we control two nodes at the top of the shorts. In this experiment, we also allow the garment to shear by introducing the force described in Section 2.4.6.

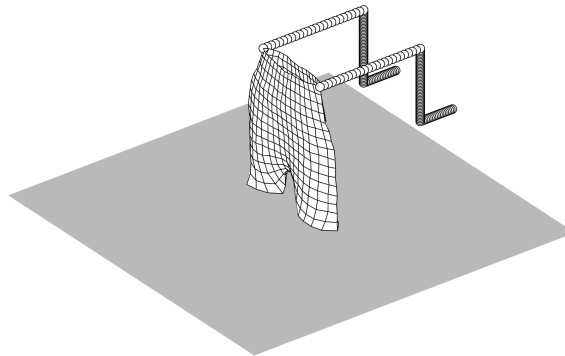


Figure 3.5: Trajectory of the controlled nodes for the dynamic folding of shorts.

The first part of the motion is performed fast enough so that the shorts have sufficient momentum to lay partially flat on top of the table after lowering them. Finally, the fold is completed by dropping the top two corners on top of the leg loops. The followed trajectory of the controlled nodes can be seen in Figure 3.5.

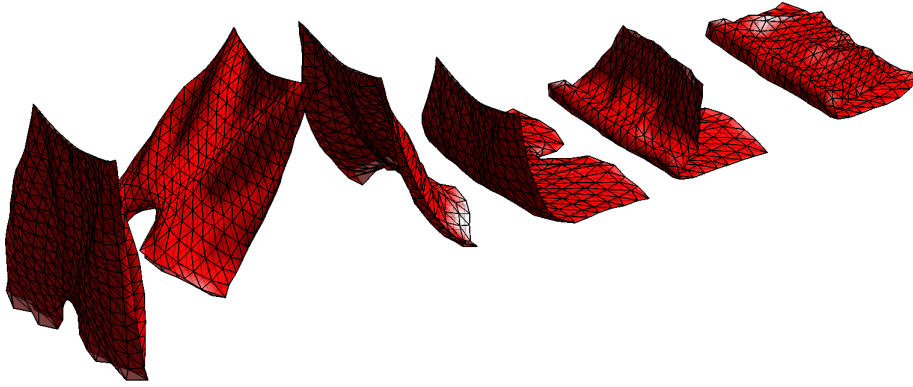


Figure 3.6: Simulated sequence of the dynamical folding of a pair of shorts. The first part of the motion (frames two and three) is performed fast enough so that the shorts lay partially flat on top of the table after a lowering phase (frame four). The final fold is completed by dropping the top two corners on top of the leg loops (frames five and six).

In Figure 3.6 we depict six frames of the simulation. Notice how crucial is the well-functioning of the self-collisions algorithm for a realist outlook of the whole folding sequence.

The definitive test for a cloth's model is its comparison to reality. Textile engineering models are focused on such comparison, to the point of developing specialized testing equipment. But the object of their study are local properties of cloth, such as elasticity parameters, which are tested in static scenarios (e.g. [24, 56, 79, 117]). Other, more recent lines of research [96, 97] focus on estimating friction coefficients using non-intrusive video images. To the knowledge of the authors, none of the models previously mentioned has been able to compare its results with the motion of cloth dynamically.

In this chapter, we seek to study how faithfully our inextensible model can reproduce recordings of real textiles under different circumstances. This chapter is divided in 3 parts:

WAM-shaking experiments (Section 4.1): in this first round of experiments we record the motion of four (size A3) textiles with a depth camera. The fabrics are shaken by a robotic WAM arm using two sets of different amplitudes and frequencies. The goal is to assess with how much accuracy our model is capable of reproducing the dynamics of the textiles.

Aerodynamics study (Section 4.2): in this second round of experiments we record the motion of eight (four size A3 and four size A2) textiles with a *Motion Capture System*. The fabrics are shaken and twisted by a human at two different speeds. We carry out two repetitions of each motion and therefore have 64 different recordings of about 15 seconds. The goal is to study how the speed and size of the textiles affect their motion and afterwards develop a predictive and *a priori* formula for the value of the cloth's physical parameters.

Collision's validation (Section 4.3): in this final set of experiments we use again motion capture, and record the collision of four (size A2) textiles. In one of the scenarios, the fabrics are laid dynamically on top of a table in a putting-a-tablecloth fashion. In the other, they are hit by a long stick four times at various places and with different strengths. The goal is to assess the accuracy of the collision and friction model previously developed and put to use the predictive formulas found before.

Cloth's materials and sizes

For the experiments in this chapter, we employ seven cloth materials (see Figure 4.1) and two different sizes: A3 (0.297 x 0.420 m with area 0.1247 m²) and A2 (0.42 x 0.594 m with area 0.2495 m²). Before performing the experiments they were ironed to remove all considerations of plasticity from the validation process. Not all the textiles are used in all experiments, in every scenario, we will specify what materials and sizes are used. In Table 4.1 we can see the density of all the fabrics and some typical examples of garments made from them.

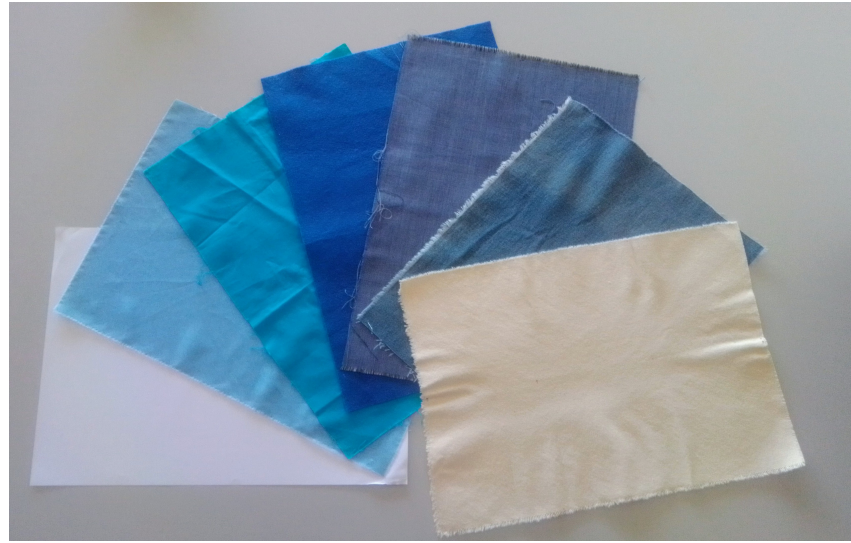


Figure 4.1: In the picture we can see all the fabrics (size A3) used in the experiments. From left to right we have: paper, polyester, light cotton, felt, wool, denim and stiff cotton.

FABRIC	DENSITY (KG · M ⁻²)	SIZES	EXAMPLES
Paper	0.0802	A3	-
Polyester	0.1042	A3, A2	Silk-like.
Light-cotton	0.1604	A3	Dressing shirt.
Felt	0.1764	A3	-
Wool	0.1804	A3, A2	Formal suit.
Denim	0.3046	A3, A2	Jeans.
Stiff-cotton	0.3046	A3, A2	Sack.

Table 4.1: Density, sizes and examples of all the materials used in all the experiments.

4.1 WAM-ARM EXPERIMENTS

A Barret robotic arm together with a depth camera are employed to record the real motion of garments being shaken at different velocities (see Figure 4.2). We implement oscillatory motions: a forward and backward shaking (see Figure 4.5) at two different speeds. Afterwards, the resulting point-cloud recordings are de-noised, interpolated and meshed, so that we end up with the spatial trajectory of the vertices of a polyhedron. In order to keep this first validation manageable, we focus only on four different size A₃ textiles: light cotton, wool, felt and paper. We use a trivial rectangular topology in order to avoid occlusions in the point-cloud when using the depth camera to record the motions. Finally, we find the physical parameters of the model that best approximate the motion of the polyhedron using a least squares approximation and a minimization algorithm. We study the evolution of absolute errors and dispersion measures (i.e. standard deviations) of the simulated garments versus the real ones for all textiles and motions. In the following sections, we give more specific details of the whole recording and comparison process.



Figure 4.2: Experimental setup for the recording of the motion of the real textiles. On the left, the depth camera. On the right, the robotic arm with an affixed hanger.

4.1.1 Camera data

As mentioned before, comparison with reality is performed by recording in the laboratory the motion of a piece of cloth when subjected to exactly the same movements of the robotic arm controlling it as specified in the simulations with our model. The real motion is captured

by a depth camera Kinect XB360-607 (see Figure 4.3). The garment is a rectangular piece of cloth, with its two upper corners fixed to a rigid hanger, which is affixed itself to a robotic arm Barrett WAM. To keep the garment completely in view of the camera, the robot moves it along the depth axis of the camera (axis x in our reference) following a curve of equation:

$$(x(t), y(t), z(t)) = (A \cos(2\pi ft) + c, y_0, z_0). \quad (4.1)$$

The two corners of the garment fixed to the robot arm follow the same oscillation, with different but constant values y_1, y_2 instead of y_0 .

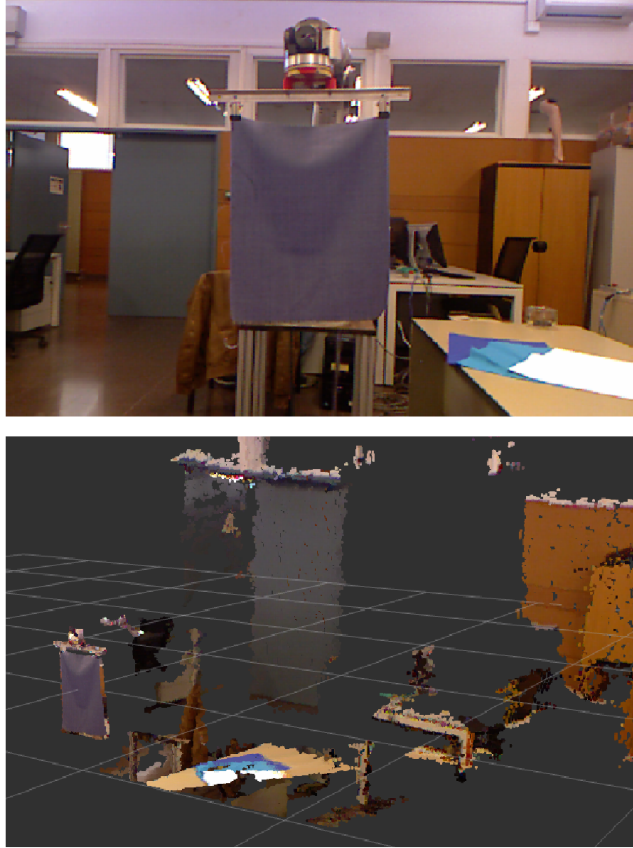


Figure 4.3: Point-cloud obtained using a depth camera (bottom) and RGB image (top). Note that the camera only gets the depth of the objects visible to it, all the rest (e.g. the robot behind the cloth) is occluded.

Two motions are recorded: the *slow* one has $A = 0.15\text{m}$ and $f = 0.3\text{Hz}$, while the *fast* one has $A = 0.075\text{m}$ and $f = 0.6\text{Hz}$. Surrounding objects are filtered using planes for each recorded frame, and outlying points are detected on the basis of distance and removed. Only garments with a homogeneous color are tested, so we can remove further noise through the use of a color filter. After the point-cloud has been thus filtered, it is meshed using the algorithm described in [55] (see Figure 4.4). The model that we are validating is continuous, producing

simulations that are very stable under remeshing. Because of this, it suffices to select as a baseline for all the experiments a coarse 9×9 mesh.

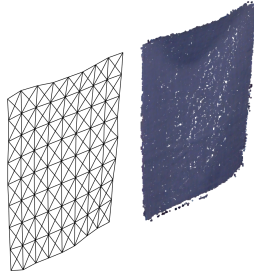


Figure 4.4: Quadrilateral meshing (left) of one of the frames of the filtered and de-noised point-cloud (right). Each quadrilateral is divided into triangles for plotting purposes.

The robotic arm follows the oscillation of Equation (4.1) for 10 seconds. The camera starts recording 1 second before the start, and finishes 3 seconds after the end of the robotic motion, for a total of 14 seconds of recorded garment motion. The original recording has its frames at irregular time steps. We replace them through linear interpolation to obtain the meshed cloth's position each $dt = 0.01$ seconds.

4.1.2 Parameter fitting

We will be adjusting only 2 parameters: the damping parameter α and the (virtual) gravitational mass δ . The other parameters in Table 2.1 are set to 0 except for $\rho = 1$, since α and δ are the most relevant for the motions performed in this experimental validation (β and κ affect mostly the local deformation of the cloth, see [7]). In Section 4.1.1 we explained how to obtain a sequence of positions of the nodes of the real cloth $\{\boldsymbol{\phi}^0, \boldsymbol{\phi}^1, \dots, \boldsymbol{\phi}^m\}$. If we integrate numerically Equation (2.19) using the same trajectories (4.1) for the two upper corners, we get a sequence $\{\boldsymbol{\varphi}^0(\delta, \alpha), \boldsymbol{\varphi}^1(\delta, \alpha), \dots, \boldsymbol{\varphi}^m(\delta, \alpha)\}$ of positions of the nodes of the simulated cloth (where $\boldsymbol{\varphi}^0 = \boldsymbol{\phi}^0$) for each value of the two parameters. Hence, a natural error metric to minimize is:

$$L(\delta, \alpha) = \sum_{i=1}^{\lfloor \frac{m}{2} \rfloor} \|\boldsymbol{\varphi}^i(\delta, \alpha) - \boldsymbol{\phi}^i\|_{\mathbf{M}}^2, \quad (4.2)$$

where $\|\cdot\|_{\mathbf{M}}$ is the L^2 norm with respect to the matrix \mathbf{M} (i.e. $\|\mathbf{x}\|_{\mathbf{M}}^2 = \mathbf{x}^T \cdot \mathbf{M} \cdot \mathbf{x}$), and we only use the first half (7 seconds) of the recorded frames to perform the fitting. We call this the *training window*. Finally, we minimize L using a derivative-free algorithm (the Nelder-Mead Simplex Method) and find the optimal values of δ, α . The resulting parameters are shown in Table 4.2.

Remark 4.1.1. Notice that for these experiments we have taken $\rho = 1$ and hence the optimal values of δ, α will not be comparable to those found in the other next experiments when we use the real values of ρ of Table 4.1.

We will analyze the evolution of the time-dependent absolute error:

$$e_i(\delta, \alpha) = \sqrt{\|\boldsymbol{\varphi}^i(\delta, \alpha) - \boldsymbol{\phi}^i\|_{\mathbf{M}}^2}. \quad (4.3)$$

MATERIAL	δ_{slow}	δ_{fast}	α_{slow}	α_{fast}	\bar{e}_{slow}	\bar{e}_{fast}
Paper	0.32	0.37	1.40	2.49	0.45 cm	0.41 cm
Light-cotton	0.52	0.52	1.29	2.69	0.41 cm	0.31 cm
Felt	0.46	0.61	1.05	2.65	0.39 cm	0.30 cm
Wool	0.47	0.50	1.19	2.52	0.37 cm	0.34 cm

Table 4.2: Estimated parameters and mean absolute errors (cm) for the slow and fast oscillatory motions performed by the WAM robot.

In the last two columns of Table 4.2 we show the average values of e_i over the *testing window* (i.e. the last 7 seconds of the motion). Moreover, for plotting purposes we use two times the standard deviation of the error at each node j as a dispersion measure:

$$d_i(\delta, \alpha) = e_i(\delta, \alpha) + 2\sqrt{\text{Var}_{j \in \text{Nodes}(S)} \left(\|\boldsymbol{\varphi}_j^i(\delta, \alpha) - \boldsymbol{\phi}_j^i\|_{\mathbb{R}^3} \right)}. \quad (4.4)$$

Remark 4.1.2. Notice that the variance is taken along the nodes of the surface and not on time, hence we are measuring a spatial standard deviation.

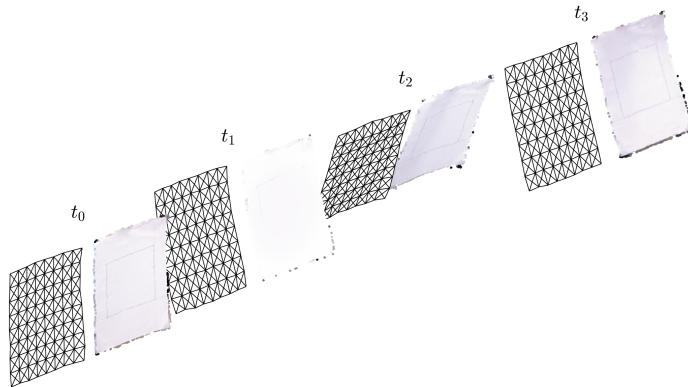


Figure 4.5: Comparison at four time instants of the recorded fast motion of paper (right) versus its simulation with the inextensible model (left) with $\delta = 0.37$ and $\alpha = 2.49$. The mean absolute error is 0.41 cm.

4.1.3 Sensitivity analysis

In order to show how robust our model is when away from the optimal values found in Table 4.2 we perform a sensitivity analysis. We carry out 100 different simulations with different values of the parameters $\alpha \geq 0$ and $\delta > 0$ (with an upper limit of two times their optimal value), and compute the mean of the absolute error (4.3) on the testing window formed by the last 7 seconds of the movement. The results for the fast motion of paper (see Figure 4.5) are shown in Figure 4.6.

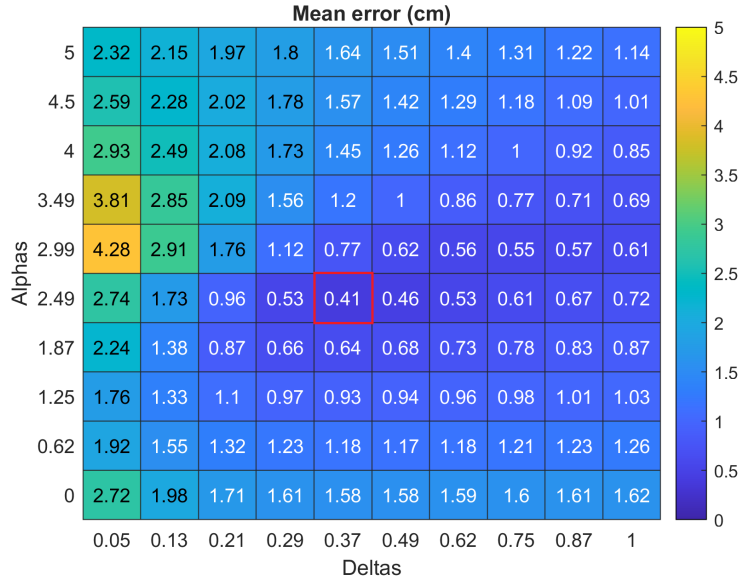


Figure 4.6: Mean of the absolute error (4.3) for the fast motion of paper using different values of the parameters of the model (damping α and virtual mass δ). In red we highlight the error with the optimal value of the parameters.

First of all, note that for the discrete set of values considered, the optimal values found are in fact a global minimum. It is interesting to note that the model is very robust with respect to the virtual mass δ , and in fact even when $\delta = 1$ (no aerodynamics), there is a value of $\alpha = 2.99$ that gives a very low overall error (0.61 cm). Other interesting limit cases are $\alpha = 0$ (no damping) and $\alpha = 5$ (over-damped), but they give higher errors. Nevertheless, we do not consider $\delta = 0$ (no gravity), since it is very unrealistic, and that is confirmed in the picture since the largest errors occur around $\delta = 0.05$.

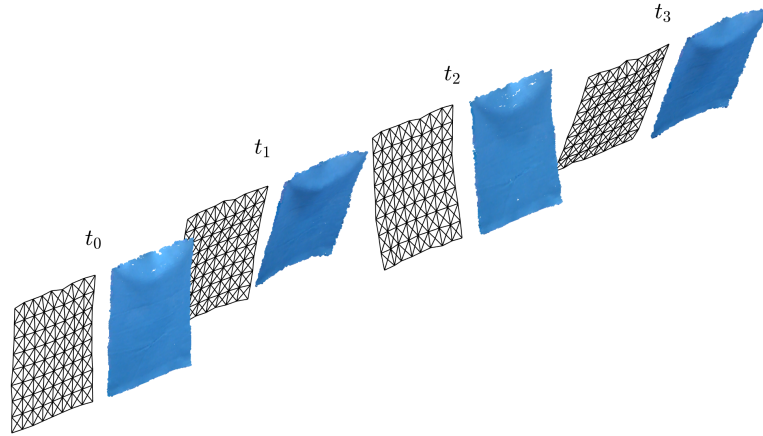


Figure 4.7: Comparison at 4 time instants of the fast motion of light cotton (right) versus its simulation with the inextensible model (left) with $\delta = 0.52$ and $\alpha = 2.69$. The mean absolute error is 0.31 cm.

4.1.4 Comparison with other models

We compare our method with three other models for the fast motion of cotton (see Figure 4.7). We study the evolution in time of the absolute error (4.3) and the dispersion error (4.4); the curves are shown in Figure 4.8. Moreover, we look at 4 different metrics: the mean of the absolute error (4.3), the maximum of the dispersion error (4.4), the total simulation time (in seconds) and, when applicable, the mean number of iterations per step of the *fast projection algorithm* described in Section 2.6.1. In Table 4.3 we display the metrics, the first two only computed over the last 7 seconds of the movement. The models considered are:

1. Inextensible model: this is our cloth model presented in Chapter 2 (without any shearing).
2. Quasi-inextensible model [44]: this is a discrete model in which stretching is not allowed because all the edges of the quadrangulated cloth are constrained to maintain their length, and only shearing is permitted by using non-stiff diagonal springs.
3. Continuum-elastic model [10]: this is a very popular elastic continuous model. Stretching and shearing are both permitted by using elastic energies instead of constraints.
4. Mass-spring system e.g. [93] and [21]: this is arguably the most widespread cloth model, used by popular animation and robotics software such as Blender [14] and MuJoCo [82]. Stretching and shearing are both allowed by using springs to connect the nodes of the cloth. There are no constraints.

We fit the relevant parameters of each of the models (shearing, stretching, damping, etc.) using the same framework applied to find the optimal parameters of our model (Section 4.1.2). All comparisons are performed using an Intel Core i7-8700K with 12 cores of 3.70 GHz.

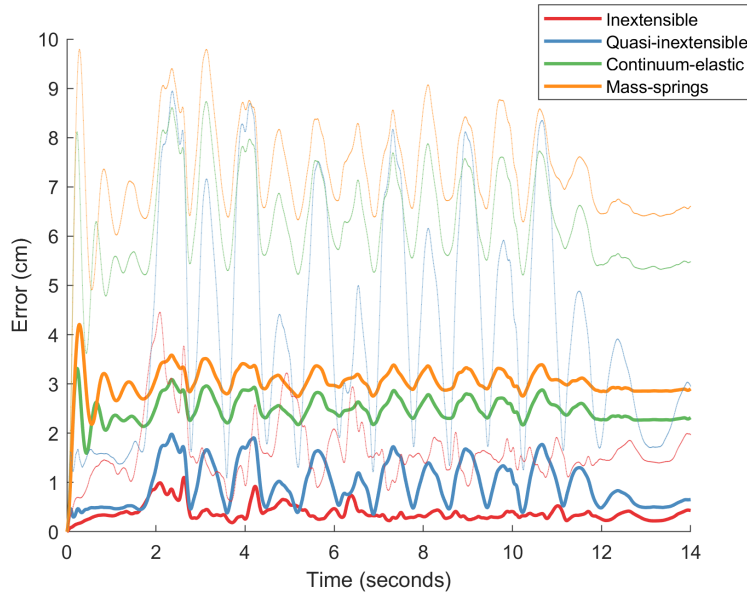


Figure 4.8: Comparison of the fast motion of light cotton between the inextensible and the other 3 models' errors (bottom curves) and dispersions (top curves), with respect to the recorded motion (using a 9×9 meshing). The mean absolute error for the inextensible model is 0.31cm.

MODEL	$\bar{\epsilon}$	$\max(d)$	TOTAL TIME	ITERATIONS
Inextensible	0.31 cm	2.02 cm	2.10 sec	1.60
Quasi-inextensible	0.95 cm	8.35 cm	2.20 sec	4.51
Continuum-elastic	2.46 cm	7.87 cm	1.16 sec	-
Mass-springs	3.03 cm	9.07 cm	1.30 sec	-

Table 4.3: Comparison of the 4 models. The first column is the mean of the absolute error (4.3) of the simulated cloth with respect to the recorded motion.

The first thing to remark is that the inextensible model has overall the lowest mean absolute error, being 3 times smaller than the quasi-inextensible one, and almost 10 times smaller than the elastic models (see Table 4.3). This supports the idea that inextensibility is a very realistic assumption. Moreover, the dispersion curves in Figure 4.8 tell us even more: in the second half of the motion there are nodes

that have absolute errors over 7cm with the other models, whereas with ours, they are at most 2cm apart from the recorded motion. On the other hand, the two elastic models have the smallest simulation times, and this is easily explainable by the fact that they do not have constraints and hence the numerical integration is simpler. Nevertheless, our model simulates 14 seconds of real time in only 2.10, which implies that we get a speed-up of seven times with respect to real time. Finally, the inextensible model requires fewer iterations of the *fast projection algorithm* than the quasi-inextensible one, but each iteration is more expensive, so in the end they have similar total simulation times.

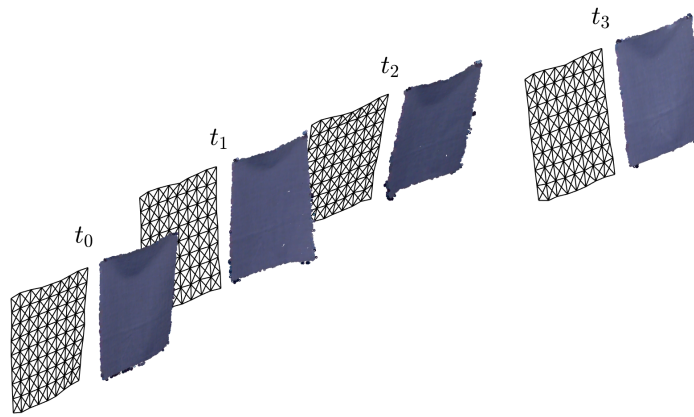


Figure 4.9: Comparison at 4 time instants of the slow motion of wool (right) versus its simulation with the inextensible model (left) with $\delta = 0.47$ and $\alpha = 1.19$. The mean absolute error is 0.37 cm.

4.1.5 Discussion of the results

As shown in Figure 4.10, the errors (4.3) for all four textiles and both motions are very low (its mean is under 5mm, see Table 4.2). For a visual comparison of the results, see Figures 4.5, 4.7, 4.9 and the video at https://youtu.be/VIorL_mTW5U. A phenomenon that is noticeable is the concentration of error (clearly seen in the dispersion curves) at the beginning of the movement; likely due to the fact that air turbulences are more complicated at this stage. In regards to the parameter values, we note their discrepancy when comparing the fast and the slow motions (using the same textile), showing that they actually account for the aerodynamics of the motion. While this approach is not standard, note that using only 2 parameters we are able to predict cloth dynamics accurately.

It is relevant to emphasize that these results are obtained using a 9×9 mesh (with which our simulations are 7 times faster than real time) for a piece of cloth of size A3 (29.7×42 cm). Keeping the obtained optimal parameters, we also computed the error (4.3) for other three

mesh resolutions 9×17 , 17×9 and 17×17 , getting respectively 0.37, 0.34 and 0.35cm as the mean of the absolute error (4.3) for the fast motion of light cotton (see Figure 4.7; with the 9×9 mesh the error is 0.31cm). This shows again (see Section 2.7.2) the robustness of the model with respect to mesh resolution.

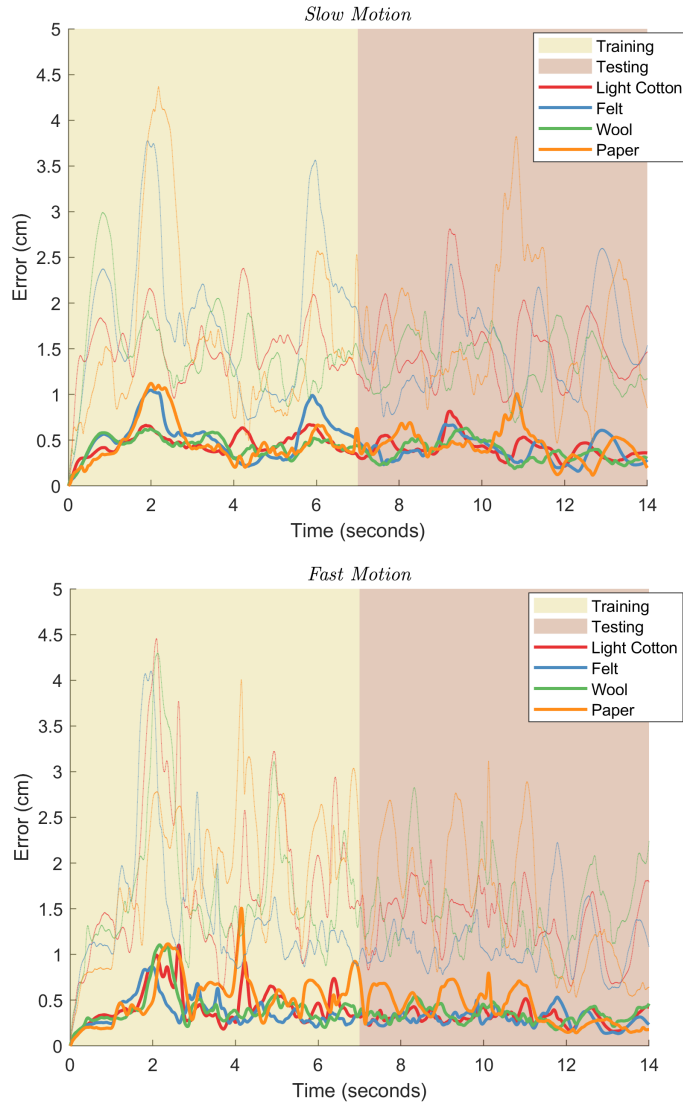


Figure 4.10: Absolute error (bottom curves) and dispersion (top curves) of the position of the simulated textile vs. the recorded motion (with the depth-camera and WAM robot) for the slow movement (top) and the fast one (bottom) using the inextensible model.

In the application of our model for robotic manipulation purposes, it may be necessary to simulate the motion of the garment prior to any possibility of calibration of its two relevant parameters. In such cases, estimates of these parameters should be made from a prior study of the model and the involved fabric. This topic will be the

subject of the next section, but we have found that a simple, interim solution is already accurate within our desired error bounds: as a prediction of motion regardless of cloth speed one may take as an estimate of each parameter the arithmetic average of the values of its calibrations found in Table 4.2 for a comparable fabric. Performing the simulations of both the slow and fast motions for light cotton using these a priori estimates instead of calibration yielded 0.51cm and 0.59cm respectively as the mean of error (4.3) (compared to 0.41cm and 0.31cm with the optimal parameters). This is in agreement with the stability of the model with respect to its parameters shown in the sensitivity analysis previously performed.

4.2 AERODYNAMICS STUDY

This second set of experiments is performed using *Motion Capture Technology*. To record the motion of the textiles a system of cameras detects and tracks reflective markers that are hooked on the cloth (see Figure 4.11). These markers, with a diameter of 3 mm and a weight of 0.013 g reflect infrared light, so the cameras are able to follow their motion through space. We use hardware and software from the manufacturer *NaturalPoint Inc*: five *Optitrack Flex 13* cameras surround the scene we wish to record (see Figure 4.12) and afterwards the recordings are processed with the software *Motive*. This combination of software and hardware offers sub-millimeter marker precision, in most applications less than 0.10 mm according to the manufacturers.



Figure 4.11: Reflective markers attached to the denim sample (encircled in red). The markers are very small, with a diameter of 3 mm and a weight of 0.013 g. We use 20 reflective markers when the size of the textile is A2 and 12 when it is A3.

This technology has been extensively used to track the motion of rigid and articulated bodies (e.g. human movements by following the trajectories of all joints). Nevertheless, its use for deformable objects has been less common since the weight of the markers could affect

the dynamics of the object. This does not happen in our case since the markers we use have a diameter of 3 mm and a weight of 0.013 g, and therefore account for less than 1% of cloth's weight even for the lightest materials.



Figure 4.12: Setup used to record the motion of the textiles: 5 cameras surround the scene so that every marker (highlighted in red in the photo) is visible to at least 2 cameras at the same time. This ensures that the system can be certain of the 3D position of the marker.

Remark 4.2.1. We now list some practical considerations and characteristics of the system:

1. Since we have 5 cameras surrounding a scene, we can record more varied and faster movements without losing track of the textiles, as opposed to just one depth camera.
2. At the beginning of a recording session the cameras are calibrated automatically with respect to a user-defined reference system. We define the plane $z = 0$ to be either at the floor or at a table (when we record collisions).
3. The cameras cannot face each other, since this causes blind spots (areas where the markers become invisible to the cameras). Reflecting lights (e.g. a window) also cause blind spots, so one must isolate the recording area as much as possible.
4. The software *Motive* does an automatic labeling and tracking of the markers, nevertheless when the movements are too rapid it loses some of them and creates new labels. Hence, different labels that correspond the same marker must be afterwards identified and merged.
5. Inevitably some markers are lost some of the time (especially with fast or abrupt movements), for instance when the textiles deform so much that the corners are no longer visible to the cameras. We have taken care that in our recordings these disappearances only happen for short periods of time.

4.2.1 Movements and textiles

For this second set of experiments, we employ the following materials: polyester, wool, denim and stiff-cotton. Both A2 and A3 sizes are used. For the A2 textiles we use 20 reflective markers, whereas for the A3 ones 12 are used. In both cases, the markers are placed equidistantly in order to obtain a faithful representation of the dynamics of the fabrics. In contrast to the first experiment with the WAM robot, this time the motions are performed by a human. This introduces way more uncertainty than before, since every movement has its own unique variabilities. Therefore every motion was recorded twice on different days: **repetition I** with a special hanger (see Figure 4.11) and **repetition II** with bare hands (see Figure 4.12). The motions are:

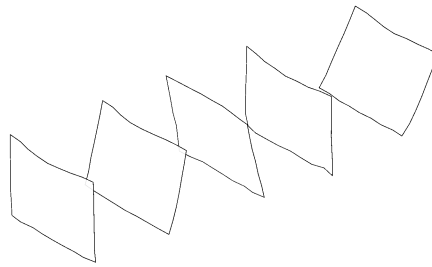


Figure 4.13: Shaking motion sequence (left to right): the cloth is shaken back and forwards.

1. Shaking: this is similar to the motion performed with WAM robot, the cloth is shaken back and forwards (see Figure 4.13).

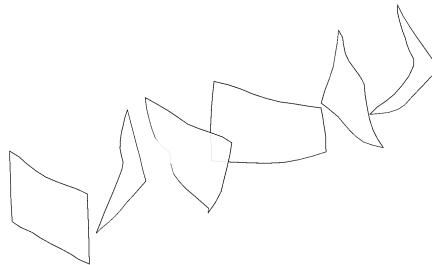


Figure 4.14: Twisting motion sequence (left to right): the cloth is rotated with respect to the z -axis back and forth several times.

2. Twisting: this is a new motion where the cloth is rotated multiple times (approximately 30 degrees) with respect to the z -axis (see Figure 4.14).

Each motion lasts approximately 15 seconds (with a frame every $dt = 0.01$ seconds) and is performed at two different speeds: *slow* and *fast*. In Table 4.4 we can see some values of average speeds ($\text{m} \cdot \text{s}^{-1}$) comparing fast and slow motions for the twisting movement of the A2 textiles.

MATERIAL	SLOW I	FAST I	SLOW II	FAST II
Polyester	0.081	0.250	0.090	0.247
Wool	0.093	0.228	0.085	0.256
Denim	0.092	0.308	0.090	0.292
Stiff-cotton	0.091	0.218	0.107	0.246

Table 4.4: Average velocities ($\text{m} \cdot \text{s}^{-1}$) for the twisting motion of the A2 textiles. We display the speeds for the first repetition (with a hanger) and for the second (with bare hands).

We can of course observe some variability but overall the speeds are maintained pretty consistently.

Remark 4.2.2. Notice that we have two motions at two speeds for four textiles with two different sizes repeated two times, which makes a total of 64 recordings.

4.2.2 Parameter fitting

As before we will be adjusting only 2 parameters: the damping parameter α and the (virtual) gravitational mass δ . The other parameters in Table 2.1 are set to 0 except for the density ρ , which this time is set to its corresponding value in Table 4.1. We denote the sequence of positions (this time given by following the reflective markers) of the recorded fabric's nodes by $\{\boldsymbol{\phi}^0, \boldsymbol{\phi}^1, \dots, \boldsymbol{\phi}^m\}$. Again, we integrate numerically Equation (2.19) using the recorded trajectories of the two upper corners (i.e. the two markers placed at the top corners), and get a simulated sequence $\{\boldsymbol{\varphi}^0(\delta, \alpha), \boldsymbol{\varphi}^1(\delta, \alpha), \dots, \boldsymbol{\varphi}^m(\delta, \alpha)\}$ (where $\boldsymbol{\varphi}^0 = \boldsymbol{\phi}^0$) for each value of the two parameters. For the simulations we consider a refinement of the initial meshes given by the markers: for the A3 case we employ a 5×7 meshing and for the A2 case a 7×9 one.

As metrics we use again the absolute error (4.3) (only at the recorded nodes) and the following time-dependent (on i) spatial (on j) standard deviation:

$$s_i(\delta, \alpha) = \sqrt{\text{Var}_{j \in \text{Nodes}(S)} \left(\|\boldsymbol{\varphi}_j^i(\delta, \alpha) - \boldsymbol{\phi}_j^i\|_{\mathbb{R}^3} \right)}. \quad (4.5)$$

Remark 4.2.3. As mentioned before some of the markers disappear for small amounts of time, in those cases, they are simply excluded from the computation of the errors (no interpolation is performed).

REPETITION I	\bar{e}	\bar{s}
Polyester	0.67 cm	1.09 cm
Wool	0.51 cm	0.97 cm
Denim	0.50 cm	1.00 cm
Stiff-cotton	0.38 cm	0.74 cm
A2	0.71 cm	1.08 cm
A3	0.33 cm	0.83 cm
Shake	0.57 cm	1.01 cm
Twist	0.47 cm	0.89 cm
Slow	0.40 cm	0.80 cm
Fast	0.63 cm	1.11 cm
Global	0.51 cm	0.96 cm

Table 4.5: Mean absolute error \bar{e} and the mean standard deviation \bar{s} with the optimal value of the parameters averaged over: fabric's material, size (A2 or A3), type of movement and speed for the first repetition of the recordings.

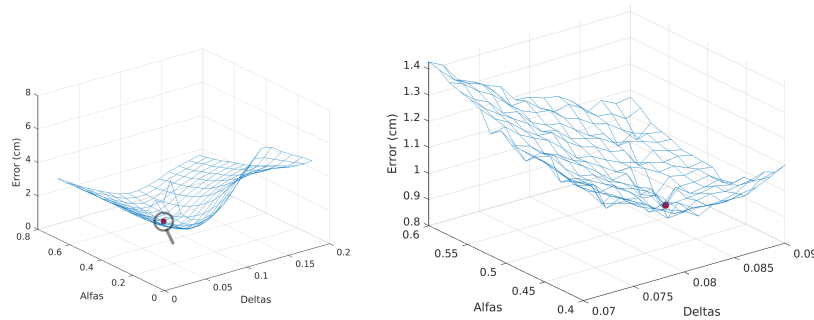


Figure 4.15: Surface plot of the error function $\bar{e}(\delta, \alpha)$ for the fast shaking motion of A2 wool (left) and a close-up near the detected minimum (right). Notice the presence of noise in the close-up.

To find the optimal value of the parameters, we minimize the average on time of the absolute error (4.3) by performing a sweep search on the space $(\delta, \alpha) \in [0, \rho] \times [0, 4\rho]$. The lower and upper limits are selected based on physical (they have to be positive) and empirical (if they are too large they drag the cloths too much, as if they were underwater) considerations. We found this optimization method to be faster and more robust than the more sophisticated minimization algorithm used in Section 4.1. This is likely due to the fact that the function we are trying to minimize is not completely smooth (see Figure 4.15) and hence has many local minima. The results for all 64 experiments can be checked in Appendix A. In Table 4.5 we can see the mean absolute error and the mean standard deviation averaged over: material, size,

type of movement and speed for the first repetition of recordings. In Figure 4.16, 4.17 and the video at <https://youtu.be/UWda9cw0uI4>, we can see a visual comparison between the recording and its simulation with the optimal value of the parameters.

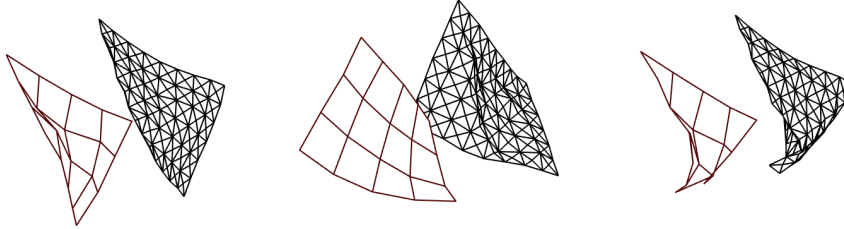


Figure 4.16: Three frames comparing the recorded fast twisting of A2 polyester (left) with its inextensible simulation (right). The error at the three depicted frames from left to right is 1.82, 1.73 and 1.36 cm respectively; being the average error of the whole simulation 1.15 cm.

From the table, we can deduce that for the inextensible model, the most challenging material to modelize is polyester. This is somehow to be expected because of its silk-like properties. On the other hand, the errors are larger for the bigger textiles, this is again reasonable since they have double the area. The fast motions have larger errors, this is likely due to the fact that in that case aerodynamics are harder to model. Finally, we can see that both the shaking and twisting motions have comparable errors.

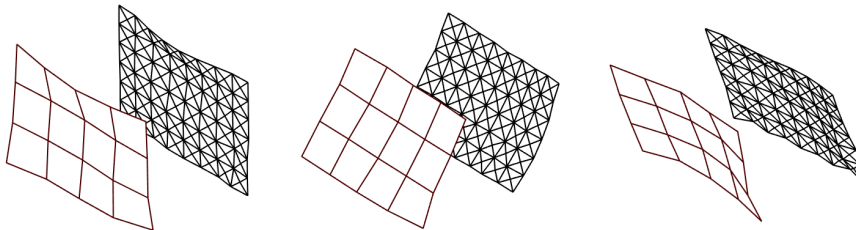


Figure 4.17: Three frames comparing the recorded fast shaking of A2 denim (left) with its inextensible simulation (right). The error at the three depicted frames from left to right is 0.63, 0.92 and 1.001 cm respectively; being the average error of the whole simulation 0.84 cm.

4.2.3 *A priori* forecast of α and δ

As we can see in Tables A.1 and A.2 in Appendix A, the values of α and δ that minimize the absolute error \bar{e} vary substantially with respect to material, size, speed, etc. We would like to find an *a priori* formula that we can use to forecast the value of these parameters without

minimizing $\bar{e}(\alpha, \delta)$. Apart from intrinsic properties of the textile (the density ρ and its size), this formula will also depend on the cloth's speed. We look then for formulas of the form:

$$\delta = \delta_0 + \delta_1 S + \delta_2 V + \delta_3 \rho, \quad \alpha = \alpha_0 + \alpha_1 S + \alpha_2 V + \alpha_3 \rho, \quad (4.6)$$

where S is a measure of size and V a measure of velocity. We have found that for our purposes using as S a normalized area of the cloth (1 for size A3 and 2 for A2) and as $[V] = \text{m}^2 \cdot \text{s}^{-2}$ the average (in time and over all the nodes) of 50% of the highest squared velocities gives the best results.

Now, in order to find the optimal values of $\{\delta_k, \alpha_k\}$ we minimize the function given by:

$$\mathcal{R}(\delta_0, \dots, \delta_3, \alpha_0, \dots, \alpha_3) = \frac{1}{32} \sum_{m=1}^{32} \bar{e}_m(\delta, \alpha), \quad (4.7)$$

where \bar{e}_m corresponds to the mean on time of the absolute error (4.3) of the m -recording of either repetition I or II obtained using the values α, δ given by Equation (4.6). To minimize this function we employ again a derivative-free algorithm (the Nelder-Mead Simplex Method). We denote by \mathcal{R}^* Function (4.7) evaluated at the optimal values of $\{\delta_k, \alpha_k\}$. For comparison let \mathcal{E}^* be the mean of the optimal errors found in the previous section (which will be by definition smaller), i.e. the mean of the errors displayed in Table A.1 or A.2.

	REPETITION I	REPETITION II
δ_0	-0.0223	-0.0359
δ_1	-0.0178	-0.0117
δ_2	0.0714	0.0780
δ_3	0.7664	0.7890
α_0	0.2082	0.2155
α_1	-0.1481	-0.1711
α_2	1.1804	1.4410
α_3	1.7440	1.9387
\mathcal{R}^*	0.578 cm	0.646 cm
\mathcal{E}^*	0.516 cm	0.560 cm

Table 4.6: Optimal values of the parameters δ_k, α_k for both repetition I (with hanger) and II (with bare hands) obtained by minimizing the function \mathcal{R} .

In Table 4.6 we can see the optimal values of the parameters $\{\delta_k, \alpha_k\}$ along with the mean absolute errors averaged over all recordings for

both repetition I (with hanger) and II (with bare hands). Notice that the error \mathcal{R}^* is comparable to \mathcal{E}^* and hence the fitting is quite accurate. Moreover, both sets of parameters are estimated independently for the two repetitions and have very similar values and the same signs, which shows that they are significant and have a consistent meaning. In particular, this justifies the introduction of the novel aerodynamic parameter δ done in Section 2.5.

4.2.4 Discussion of the results

In this second round of experiments, we have performed a more exhaustive set of recordings than before. We already knew from Section 4.1 that our model was capable of reproducing faithfully the shaking motion of A3 textiles, but we have enlarged the set of recordings by adding a new twisting movement and a larger cloth size (DIN A2). This has resulted in two sets of 32 recordings (each cloth being recorded two times on different days: with a special hanger or bare hands) each lasting approximately 15 seconds. As before we have estimated the optimal values of the physical parameters and the model has achieved very low mean errors (less than 1 cm) and standard deviations (see Tables 4.5, A.1 and A.2), even for the A2 textiles and fast motions. Finally, we have found a predictive formula in order to obtain *a priori* estimates for the values of the parameters α and δ of the model. This formula depends on the density of the textile, its size and more importantly its speed. The formula was found to produce parameter's values which in turn still give rise to very low absolute errors. In the next section, we will further make use of this *a priori* formula and test its accuracy.

4.3 VALIDATION OF COLLISIONS

For this final set of experiments, we will use again the following materials: polyester, wool, denim and stiff-cotton. We employ exactly the same recording setup described in the previous section (the motion capture system, Section 4.2). This time only the A2 size is used (as before with 20 reflective markers placed equidistantly). We denote the sequence of positions of the recorded fabric's nodes by $\{\phi^0, \phi^1, \dots, \phi^m\}$ and the simulated sequence by $\{\varphi^0, \varphi^1, \dots, \varphi^m\}$. For the simulations, we utilize a refined 7×9 mesh. As error metrics, we use again the absolute error (4.3) and the spatial standard deviation (4.5). In order to validate the realism of our collision model, we must fit this time three parameters: as before α (damping) and δ (virtual mass); and for the first time μ (friction coefficient). In order to obtain their optimal value, as usual, we minimize the mean on time of the absolute error:

$$\sum_i e_i(\delta, \alpha, \mu) = \sum_i \sqrt{\|\boldsymbol{\varphi}^i(\delta, \alpha, \mu) - \boldsymbol{\phi}^i\|_{\mathbf{M}}^2}. \quad (4.8)$$

The experiments are performed by a human (without any hanger) and consist of two scenarios:

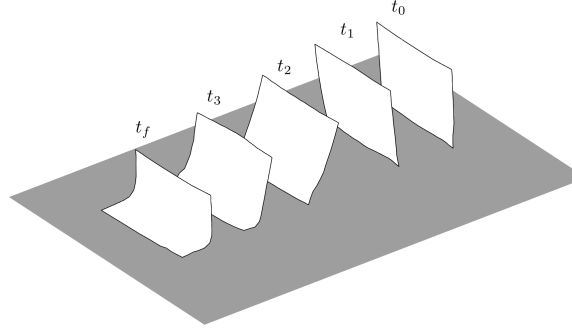


Figure 4.18: Putting a tablecloth motion sequence (right to left): the cloth starts suspended and is afterwards laid dynamically (only partially) onto the table.

4.3.1 Tablecloth scenario

The textile starts suspended at about 10 cm of height and is afterwards laid dynamically (only partially, so that half of the cloth is still suspended) onto the table (see Figure 4.18). Each motion lasts approximately 4 seconds (with a frame every $dt = 0.01$ seconds) and is performed with two different surfaces as the table, one with *low* friction (a raw polished table) and one with *high* friction (a table with a tablecloth). The goal here is to estimate the friction coefficient μ (see Equation (3.6)) for the two different surfaces and to study the sensitivity of the model with respect to friction.

MATERIAL	μ_{LOW}	\bar{e}_{LOW}	\bar{s}_{LOW}	μ_{HIGH}	\bar{e}_{HIGH}	\bar{s}_{HIGH}
Polyester	0	0.95 cm	1.20 cm	1	0.84 cm	1.03 cm
Wool	0	0.58 cm	0.73 cm	2	0.52 cm	0.75 cm
Stiff-cotton	0	0.60 cm	0.86 cm	2	0.58 cm	0.77 cm
Denim	0	0.77 cm	1.11 cm	1.6	0.61 cm	0.80 cm

Table 4.7: Optimal values of the friction coefficients along with the mean absolute error and spatial standard deviation for the low and high friction scenario

In Table 4.7 we can see the optimal values of the friction coefficients along with their optimal errors and deviations for the low and high

friction scenarios. The optimal friction coefficients for the low friction case (raw polished table) were all smaller than 10^{-3} and that is why they were rounded up to zero on the table. For a visual comparison of the results see Figure 4.19 and the video at <https://youtu.be/SWJcxFTwKHE>.

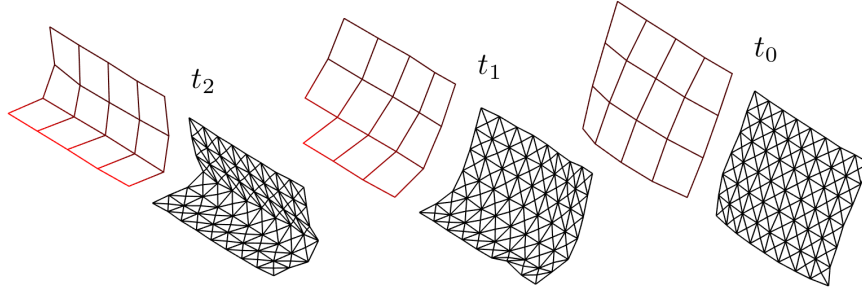


Figure 4.19: Three frames comparing the recorded tablecloth low friction scenario of A2 wool (left) with its inextensible simulation (right). The error at the three depicted frames from right to left is 0.80, 1.17 and 0.76 cm respectively; being the average error of the whole simulation 0.58 cm.

In order to understand how friction influences the dynamics of the textiles we perform a sensitivity analysis for the high friction case, i.e. we vary the value of μ (keeping all the other parameters fixed), and compute the mean of the absolute error (4.3). The results can be seen in the heat-map depicted in Figure 4.20. Notice that in general the model is quite stable with respect to the optimal friction value.

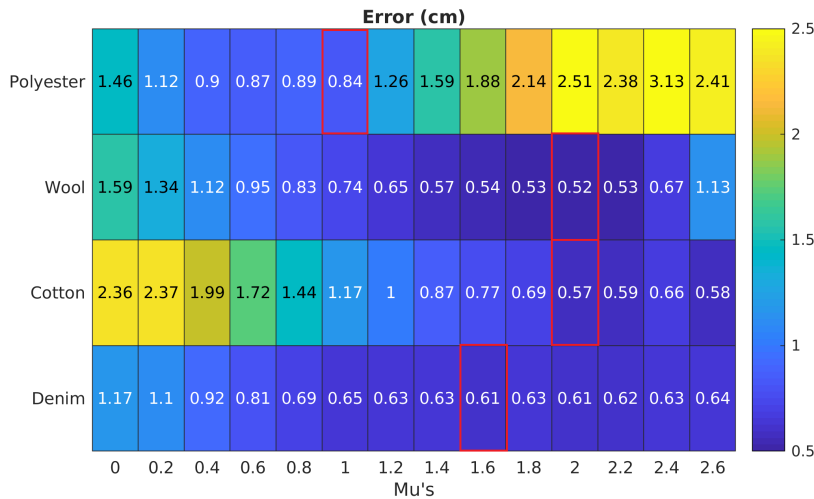


Figure 4.20: Sensitivity analysis for high friction case, i.e. we vary the value of μ and compute the absolute error (4.3) for the four A2 fabrics. In red we encircle the error found with the optimal parameter of μ .

4.3.2 Hitting scenario

In this final scenario, the fabrics are held suspended in the air (with the long sides perpendicular to the floor) and hit repeatedly with a long stick. The hits are aimed at various locations of the cloth with varied strengths and speeds (see Figure 4.21). In order to simulate the hits, the stick is subdivided into small edges and we employ a procedure similar to the one used for self-collisions of the cloth in the case of an edge-edge collision (see Section 3.3).

Remark 4.3.1. The stick could be represented as a moving cylinder of small radius, but then we would be forced to use a very fine mesh to simulate the hits.

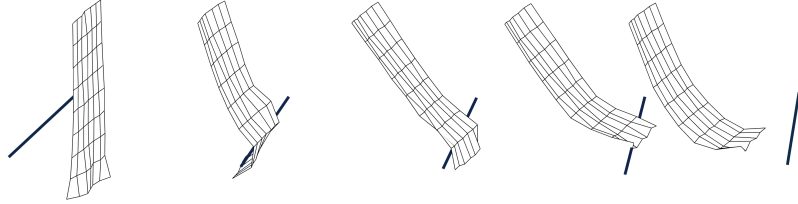


Figure 4.21: Long-stick hits sequence (left to right): the cloth is held by its two upper corners and then is hit repeatedly with a long stick. The hits are aimed at different locations with varied intensities.

Let us denote by $\{a_1(t), \dots, a_m(t)\}$ the endpoints of the edges of the stick. Then, for every iteration j of the iterative process (3.4), as we do with self-collisions (see Section 3.4.1), we must check if a collision occurred during the motion between the stick-edges

$$\{a_1(t_n), \dots, a_m(t_n)\} \rightarrow_{dt} \{a_1(t_{n+1}), \dots, a_m(t_{n+1})\}$$

and the edges of our triangulated cloth $\varphi^n \rightarrow_{dt} \varphi_j$. This means that for every detected collision, in the next iteration $j+1$ of the sequence of quadratic problems (3.4) we must add a constraint of the form:

$$H(\varphi_{j+1}) = \langle \pi_\alpha(a_1, a_2) - \pi_\beta(x_1, x_2), \nu \rangle \geq 0,$$

where a_1, a_2 are the two endpoints of the corresponding edge of the stick, x_1, x_2 are likewise the two endpoints of the edge's cloth, $\pi_{\alpha'}(a_1, a_2) = (1 - \alpha')a_1 + \alpha'a_2$ and $\pi_{\beta'}(x_1, x_2) = (1 - \beta')x_3 + \beta'x_4$ are the closest points between the two segments and ν is the normal vector to both edges. The values ν, α', β' are constant in time, and are computed in the case of the cloth with the positions of the segments defined by φ_j and for the stick at time t_{n+1} . The normal vector ν is oriented such that $H(\varphi^n) \geq 0$.

On the other hand, the real long stick has a length of 75 cm and a diameter of 1.5 cm (see Figure 4.22). Two (rather large, with a diameter of 1.5 cm) markers are put at both ends of the stick to record its trajectory.

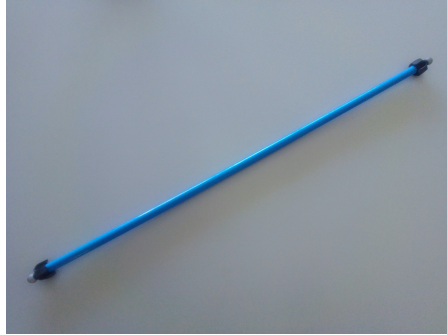


Figure 4.22: Long-stick (with a length of 75 cm and a diameter of 1.5 cm) used to hit the textiles. Two markers are put at both ends of the stick to record its trajectory.

Remark 4.3.2. We consider the stick's thickness by imposing $H(\varphi) \geq \tau_0$, where $\tau_0 = 0.75\text{cm}$ is the radius of the stick. Moreover, this thickness is taken into account in the detection process (see Section 3.3.3).

The stick is made of polished plastic and hence we consider friction between the cloth and the stick to be negligible (moreover, since the cloth is held firmly by the two upper corners the small amount of friction that could exist is always overcome by the stick). Each textile is hit four times with recordings varying between 12 and 18 seconds (as usual with a frame every $dt = 0.01$ seconds). On top of fitting as usual the damping parameter α and the (virtual) gravitational mass δ , the goal of this scenario is to assess the realism of our collision algorithm when modeling the hits and put to use the predictive aerodynamic formula (Section 4.2.3).

Remark 4.3.3. By the nature of this collision experiment, some movements of the textiles are very abrupt and therefore as mentioned before the markers disappear some of the time. This problem is also present in the recording of the trajectory of the stick. This is more problematic, since we need its position at every time to be able to simulate the collisions. We have interpolated the missing positions of the stick linearly.

In this experiment, we also study the performance of the **active-set** collision algorithm described in Chapter 3. We compare it with a standard **interior-point** algorithm implemented to solve quadratic problems (see [86]). As before, all comparisons are performed using an Intel Core i7-8700K with 12 cores of 3.70 GHz. Since the four recordings have different durations, we compute the quotient

$$q = \frac{T_{\text{sim}}}{T_{\text{rec}}}, \quad (4.9)$$

where T_{rec} is the duration of the recording and T_{sim} is the amount of time it takes to simulate it. Hence $q \approx 1$ would mean that the simulations work in real-time, $q \approx 0.5$ means that they are twice as fast, etc. In Table 4.8 we can see the value of the absolute error and standard deviation with the optimal value of the parameters (α^* and δ^* shown in Table 4.9).

MATERIAL	\bar{e}	\bar{s}	ACTIVE-SET	INTERIOR-POINT
Polyester	1.44 cm	2.13 cm	0.456	1.344
Wool	1.39 cm	2.23 cm	0.437	1.298
Denim	0.98 cm	1.86 cm	0.425	1.235
Stiff-cotton	1.07 cm	1.85 cm	0.510	1.576

Table 4.8: Mean absolute error and spatial standard deviation with the optimal value of the parameters α^* and δ^* . In the two last columns, we display the quotient (4.9), for our active-set collision algorithm and a standard interior-point method.

For a visual comparison of the results, together with a plot of how the absolute error varies with time for the four textiles, see Figure 4.23 (stiff-cotton), Figure 1.4 (polyester), Figure A.1 (denim) and Figure A.2 (wool) and the video at https://youtu.be/U7-p_1E09L8. With yellow lines we highlight the moments in which the stick is in contact with the cloth (during the simulations). Notice that precisely in those instants is where more missing data is found. In the figures we see clearly that the error concentrates after the hit and not during it, showing that the collision model is very realistic but afterwards the aerodynamics become dominant and the errors increase. Overall the fitting is quite good, with errors slightly bigger than those found in Section 4.2 for the fast motions of the A2 size. Finally, our active-set algorithm is found to be almost 3 times faster than a standard interior-point method, with simulations still going faster than real time for a 7×9 mesh (see Table 4.8).

Finally, to finish this section we assess the accuracy of the formulas found during the aerodynamic experiments shown in Table 4.6; i.e. we do not perform any optimization and compute the value of the aerodynamic parameters with the formulas

$$\hat{\delta} = -0.0223 - 0.0178S + 0.0714V + 0.7664\rho, \quad (4.10)$$

$$\hat{\alpha} = +0.2082 - 0.1481S + 1.1804V + 1.7440\rho, \quad (4.11)$$

where, as before, ρ is the density of the cloth, S is a normalized area measure (in this scenario is always 2), V is the average (in time and

over all the nodes) of 50% of the highest squared velocities. We have used the coefficients of the first repetition in Table 4.6. A comparison of the value of these estimated parameters ($\hat{\delta}, \hat{\alpha}$) with the optimal ones (δ^*, α^*) along with their respective absolute errors is displayed in Table 4.9.

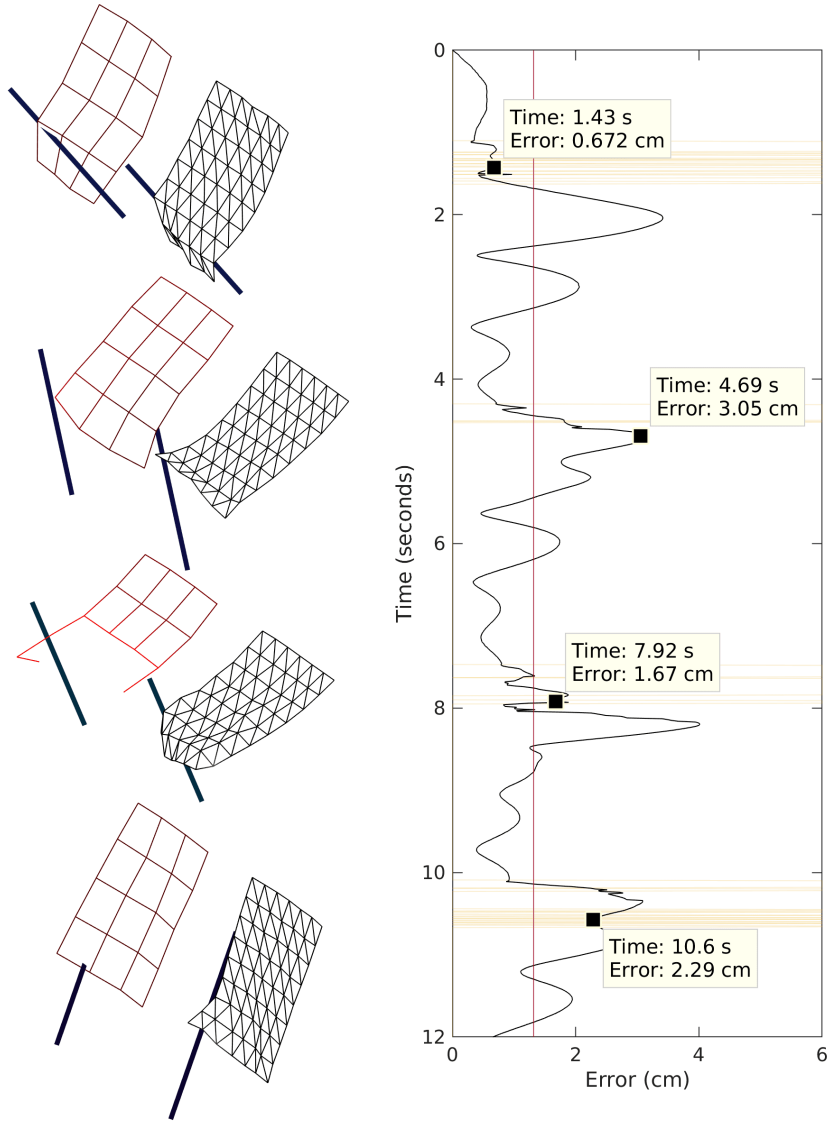


Figure 4.23: Four frames comparing the recorded hitting of A2 stiff-cotton (left) with its inextensible simulation (right); being its average error 1.07 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines, we highlight the moments in which the stick is in contact with the cloth.

We can see that in general, the formula gives a very reasonable estimation of the parameters (especially for α), but it always overestimates δ and this causes the errors \hat{e} to be somehow larger. Still the results are very accurate considering how challenging this scenario is and

MATERIAL	$\hat{\delta}$	δ^*	$\hat{\alpha}$	α^*	\hat{e}	e^*
Polyester	0.032	0.025	0.26	0.28	1.59 cm	1.44 cm
Wool	0.099	0.078	0.54	0.50	1.96 cm	1.39 cm
Denim	0.198	0.127	0.82	0.67	1.76 cm	0.98 cm
Stiff-cotton	0.203	0.165	0.90	0.89	1.34 cm	1.07 cm

Table 4.9: Comparison of the parameters ($\hat{\delta}, \hat{\alpha}$) obtained with the aerodynamic formulas (4.10) with the optimal ones (δ^*, α^*) along with their respective mean absolute errors.

that the parameters are computed with an *a priori* equation without any optimization at all.

4.3.3 Discussion of results

With these final experiments, we have validated two different but related aspects of the collision model: its ability to simulate properly friction and to model the reaction of cloth to fast and strong hits with a long stick.

We have found the optimal friction parameters for both a high and a low friction case (see Table 4.7), with absolute errors still under 1 cm for all DIN A2 textiles. We have shown that the simulations are very stable with respect to friction by performing again a sensitivity analysis (see Figure 4.20).

On the other hand, we have been able to model the most challenging scenario in this work: the cloths were held by its two upper corners and then were hit repeatedly at different locations with varied intensities with a stick. The average errors are still of the order of 1 cm (see Table 4.8) and we are able to properly simulate the hits, appearing the biggest errors not during the hits but just after because of aerodynamic effects (see Figure 4.23). Finally, we have put to use the *a priori* aerodynamic formulas found before (see Equations (4.6) and Table 4.6), by using them to compute an estimate of the physical parameters without performing any optimization with very satisfactory results (see Table 4.9). Last but not least, we have checked that the simulations are still two times faster than real-time (for the hitting scenario and a simulated mesh of 7×9 nodes), being our novel active-set solver three times faster than a standard interior-point method.

Part II

RECONSTRUCTING GARMENTS

In this second part we study the perception problem for textiles: the identification of their geometry and position from point-cloud samples, as obtained e.g. with depth cameras. We present a reconstruction algorithm based on Morse theory that proceeds directly from a point-cloud to obtain a cellular decomposition of the cloth surface: a global piecewise parametrization of the surface is found, with a small number of pieces (Morse cells). From the cellular decomposition, the topology of the surface can be deduced immediately and the point-cloud can be filtered and/or simplified with topological guarantees.

Until now we have not studied in depth the problem of meshing an arbitrary point-cloud. When we have a rectangular and nearly flat topologically trivial piece of cloth (as in Chapter 4), this problem does not pose a great challenge. Arbitrary point-clouds with any topology however are a different story altogether. We now present an algorithm for the reconstruction of a surface from a point sample. It proceeds directly from the point-cloud to obtain a cellular decomposition of the surface derived via a Morse function. No intermediate triangulation or local implicit equations are used, saving on computation time and reconstruction-induced artifices. No a priori knowledge of surface topology, density or regularity of its point sample is required to run the algorithm. The results are a piecewise parametrization of the surface as a union of Morse cells, suitable for tasks such as noise-filtering or mesh-independent reparametrization, and a cell complex of small rank determining the surface topology. This algorithm can be applied to smooth surfaces with or without boundary, embedded in an ambient space of any dimension.

5.1 RELATED WORK

Reconstruction of a surface in space from a sample of points on it is a question to which considerable attention has been devoted in the areas of Computational Geometry and Computer Graphics (see [30] for a variety of methods). Our goal is a fast algorithm for topology identification and parametrization of surfaces with boundary. The qualities are required for robotic handling of textiles, but are hoped to make the algorithm fit to study higher dimensional algebraic varieties.

Differential Topology has tackled the piecewise parametrization problem for manifolds through Morse functions. Applying this idea directly to the sample point cloud of a surface was suggested by [39, 126], who propose an algorithm for point clouds with a known, homogeneous density of sampling. In [20] a Morse decomposition scheme from point-clouds sampling manifolds without boundary of any dimension is proposed. All these works, however, stop short of questions such as cell parametrization or attachment maps, which are relevant to robotic applications where point-clouds of textiles may need to be filtered and down-sampled in order to e.g. be simulated.

We report in this work a complete Morse cell decomposition algorithm for surfaces of any topology, with or without boundary, which can be applied to sample-point clouds without a priori knowledge

of sampling density or regularity, or of surface topology. It can be applied to surfaces in any ambient dimension. We use the gradient flows of [39, 126] as the starting point, but then detect saddle points and their Morse cells differently, proposing a new procedure based on studying the level sections of these flows.

5.2 MORSE THEORY FOR MANIFOLDS

Let M be a smooth compact manifold without boundary. A map $f : M \rightarrow \mathbb{R}$ is *Morse* if it is \mathcal{C}^2 , has only finitely many critical points, and at all of these the Hessian $H(f)$ is nondegenerate. Classical Morse theory (see [52]) shows that a generic Morse function f induces, through its gradient flow, two decompositions of the manifold M :

1. *As a CW complex* (see [83]): Each critical point of f , together with its unstable manifold for the vector field $-\nabla f$, forms a cell which is topologically a ball, whose boundary attaches to lower-dimensional cells (see Figure 5.1). A global piecewise parametrization of M is achieved, and a Morse-Smale complex, with the critical points of f as a basis, giving the singular homology of M .

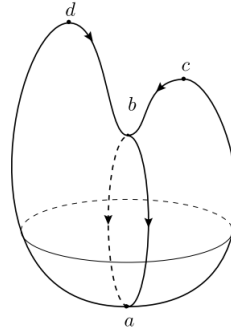


Figure 5.1: Critical points of the Morse-Smale function $f(x, y, z) = z$ on an example surface.

2. *As level sets*: M is foliated by the level sets $f^{-1}(c)$. For regular values c these level sets are submanifolds of M with codimension 1, with $f^{-1}(c_1) \cong f^{-1}(c_2)$ if no critical value of f lies between c_1 and c_2 . The transformation of the level set when c crosses a critical value of f is a surgery (see [52]).

The success of Morse theory comes from the fact that Morse functions, and the Morse-Smale transversality conditions required for the above analysis, are generic among \mathcal{C}^2 maps from M to \mathbb{R} . For instance, the height function in a random direction in \mathbb{R}^N has probability 1 of being a Morse-Smale function. In the following subsection, we present a summary of Morse theory (as level sets) for surfaces without boundary with views towards explaining the case with boundary next.

5.2.1 Morse theory for surfaces without boundary

Let $S \subset \mathbb{R}^N$ be a smooth compact surface without boundary. As before, a map $f : S \rightarrow \mathbb{R}$ is *Morse* if it is \mathcal{C}^2 , has only finitely many critical points (i.e. points p where $d_p f = \nabla f(p) = 0$), and at all of these its Hessian has rank 2.

Definition 7 (Morse data). For each critical point $p \in S$, the Morse data are the pair of sets $(A(p), B(p))$ where

$$A(p) := \mathcal{B}(p) \cap f^{-1}([f(p) - \epsilon, f(p) + \epsilon]),$$

with $\mathcal{B}(p) \subset \mathbb{R}^N$ a closed ball around p of sufficiently small radius, the value of $\epsilon > 0$ is such that there are not more critical points of f in $f^{-1}([f(p) - \epsilon, f(p) + \epsilon])$ and

$$B(p) := \mathcal{B}(p) \cap f^{-1}(f(p) - \epsilon).$$

Notice that $B(p) \subseteq \partial A(p)$.

Let us now denote

$$f_{\leq c} = f^{-1}((-\infty, c]), \quad f_c = f^{-1}(c).$$

Then it is well known (see [52]) that as $c \in \mathbb{R}$ increases two things can happen:

- A If $c_1 < c_2$ and there are no critical values of f between them, then $f_{\leq c_1}$ and $f_{\leq c_2}$ have the same topology (they are actually diffeomorphic).
- B If there is a critical point $p \in S$ such that $c_1 < f(p) < c_2$ then $f_{\leq c_2}$ is obtained, up to diffeomorphism, from $f_{\leq c_1}$ by attaching the cell $A(p)$ along $B(p)$, i.e. $f_{\leq c_2} \simeq (f_{\leq c_1} \cup A(p)) / \sim$ where the equivalence relation is given by identifying $B(p) \subseteq f_{\leq c_1}$ with points of $\partial A(p) \supseteq B(p)$.

Now, since $H(f)$ has rank 2 at p , we can only have three types of critical points (see Figure 5.2) according to their index (number of negative eigenvalues):

1. Minima: $A(p)$ is homeomorphic to a disk and $B(p) = \emptyset$. In this case, there is no surgery, the cell $A(p)$ just appears. This cell retracts to a point, the local minimum, which will be a 0-cell in the Morse-Smale complex.
2. Saddles: $A(p)$ is a quadrilateral (homeomorphic to a disk) and $B(p)$ consists of two segments. Two opposite sides of $\partial A(p)$ are identified with $B(p)$ (see Figure 5.2). The attachment of the cell $A(p)$ along $B(p)$ is homotopy-equivalent to the attachment of a 1-cell, namely the medial axis of $A(p)$ to the middle points of $B(p)$.

3. Maxima: $A(p) = D$ is homeomorphic to a disk and $B(p) = \partial D$ is its boundary. The attachment map identifies $\partial A(p)$ with $B(p)$. This surgery adds a 2-cell to the complex.

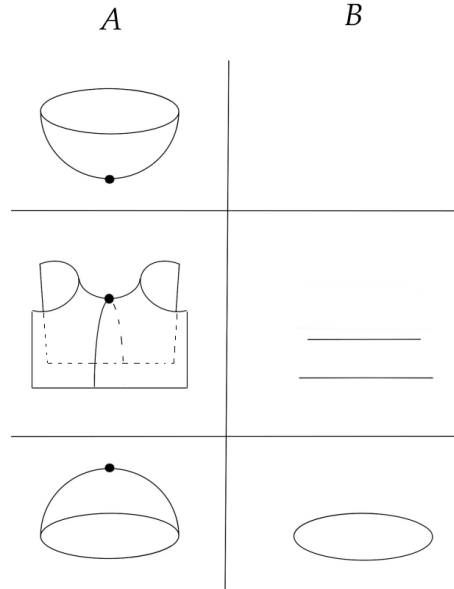


Figure 5.2: Three types of critical points (from top to bottom: minima, saddles and maxima) and their Morse data for surfaces without boundary.

5.2.2 Morse theory for surfaces with boundary

Morse theory also extends to manifolds with boundary [46]. Since this theory is less well known, in the following we give more details on how the boundaries affect the cellular decomposition and the transition between level sets.

Definition 8. We say that a C^2 map $f : M \rightarrow \mathbb{R}$ defined in a manifold M with boundary ∂M is *Morse* if

1. it is Morse in the interior of M ,
2. its restriction to ∂M , $g := f|_{\partial M}$ is also a Morse function,
3. if $p \in \partial M$ is a critical point of g then $\ker(d_p f) = \{0\}$.

As in the previous section, we will focus on the case $M = S$ is a surface. Notice that now we can have critical points of g in the boundary curves ∂S that are not critical points of f in the whole of S . Moreover, the third condition of the previous definition ensures that critical points located at ∂S are not saddle points of f in S . Notice that minima (resp. maxima) points p of f located at ∂S will satisfy that

$d_p f \neq 0$, and thus, they will not be strictly speaking critical points of f (but they will be of g).

Hence, we will adopt a *Whitney stratification* dividing the surface into two **strata**: the first $E_1 = \partial S$ being the boundary and the second $E_2 = \text{int}(S)$ the interior. Each stratum will have as before Morse data, but this time apart from the sets $(A(p), B(p))$ defined before, which we will call *tangential data*, we will have also *normal data*.

Definition 9 (Tangential and normal data). Let $p \in S$ be a critical point for some stratum E_i of the Morse function $f|_{E_i}$. Then

1. The tangential Morse data are the pair of sets (A_T, B_T) defined as in Definition 7 for $f|_{E_i}$.
2. The normal Morse data are the sets (A_N, B_N) given by

$$\begin{aligned} A_N(p) &:= \mathcal{N}(p) \cap f^{-1}([f(p) - \epsilon, f(p) + \epsilon]), \\ B_N(p) &:= \mathcal{N}(p) \cap f^{-1}(f(p) - \epsilon). \end{aligned}$$

where $\mathcal{N}(p) = S \cap D(p)$ is called the *normal slice* at p and $D(p) \subset \mathbb{R}^N$ is a sufficiently small closed disk around p of dimension $N - \dim(E_i)$ transversal to E_i at p and the value of $\epsilon > 0$ is such that there are no more critical points in $f^{-1}[f(p) - \epsilon, f(p) + \epsilon]$.

Notice that by construction $E_i \cap D(p) = \{p\}$, and as before $B_{N,T} \subseteq \partial A_{N,T}$.

Remark 5.2.1. When $p \in \text{int}(S)$ and $N = 3$ then $D(p)$ is homeomorphic to a piece of curve normal to S at p and hence $\mathcal{N}(p) = \{p\}$. Therefore $(A_N, B_N) = (p, \emptyset)$. It is not hard to see that this is also the case when $N > 3$.

We are now ready to state the main theorem of stratified Morse theory [46] (SMT theorem, pages 6-8).

Theorem 1 (Goresky-MacPherson). As $c \in \mathbb{R}$ increases two things can happen:

- A If between $c_1 < c_2$ there are no critical points of f then $f_{\leq c_1}$ and $f_{\leq c_2}$ are diffeomorphic.
- B If there is a critical point $p \in S$ such that $c_1 < f(p) < c_2$ then $f_{\leq c_2}$ is obtained from $f_{\leq c_1}$ by performing a *surgery* around p with Morse data (A, B) diffeomorphic to the topological product

$$(A_N, B_N) \times (A_T, B_T) = (A_N \times A_T, A_N \times B_T \cup B_N \times A_T),$$

i.e. $f_{\leq c_2} \simeq (f_{\leq c_1} \cup A) / \sim$ where the equivalence relation is given by identifying $B \subseteq f_{\leq c_1}$ with points of $\partial A \supseteq B$.

Remark 5.2.2. We already established that at interior critical points of S , we have $(A_N, B_N) = (p, \emptyset)$. Therefore in that case the above product is trivial and the attachment maps are the same ones described earlier.

The new cases occur when p is a critical point of $f|_{\partial S}$ lying on ∂S . Since ∂S is a one-dimensional curve, $p \in \partial S$ can only be a maximum or a minimum. Nevertheless, depending on whether p is also a local minimum (resp. maximum) of f or just of $g = f|_{\partial S}$, we will have four different cases (see Figure 5.3). Recall that for minima (and maxima) points p of f located at ∂S we have that $d_p f \neq 0$. Nevertheless, in order not to complicate the discussion semantically we will still call them critical points since they satisfy $d_p g = 0$.

Remark 5.2.3. We will denote by $(\blacksquare, \sqcup, |, _)$ a quadrangular 2-cell, all its sides minus the top one, two lateral sides and the bottom side, respectively.

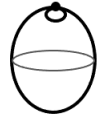









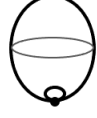








	A_T	B_T	A_N	B_N	A	B
		$\cdot \cdot$		\cdot		
		$\cdot \cdot$				
						
				\cdot		

Figure 5.3: Four types of critical points located at the boundary and their tangential and normal Morse data.

The four new different critical points that we can have are:

Maxima of $f|_{\partial S}$: In this case, A_T is a closed concave piece of curve (which we denote by \cap) and B_T two points, i.e. $(A_T, B_T) = (\cap, \cdot \cdot)$. We now have two sub-cases:

- p is also a local maximum of f . Then A_N is a closed interval and B_T one point, i.e. $(A_N, B_N) = (|, \cdot)$. Therefore the topological product is

$$(A, B) = (\cap, \cdot \cdot) \times (|, \cdot) = (\blacksquare, \sqcup),$$

and the three of the sides of ∂A where p is not present are attached to B . During this surgery, a 1-cell (containing p) is attached to the boundary and a 2-cell to the interior (closing a void in the process).

2. p is not a local maximum of f (only of $f|_{\partial S}$). Then A_N is again a closed interval but B_T is empty, i.e. $(A_N, B_N) = (|, \emptyset)$. Then the Morse data is

$$(A, B) = (\cap, \dots) \times (|, \emptyset) = (\blacksquare, ||),$$

and the two opposite sides of ∂A where p is not present, attach to B . This surgery is equivalent to attaching a 1-cell to the boundary (but no 2-cell is attached to the interior).

Minima of $f|_{\partial S}$: In this case, A_T is a closed convex piece of curve (which we denote by \cup) and B_T is empty, i.e. $(A_T, B_T) = (\cup, \emptyset)$. We again have two sub-cases:

1. p is also a local minimum of f . Then A_N is a closed interval and B_T is empty, i.e. $(A_N, B_N) = (|, \emptyset)$. Therefore the topological product is

$$(A, B) = (\cup, \emptyset) \times (|, \emptyset) = (\blacksquare, \emptyset).$$

In this case there is no surgery, the cell A just appears. This cell retracts to a point, which will be a 0-cell in the cell complex.

2. p is not a local minimum of f (only of $f|_{\partial S}$). Then A_N is a closed interval and B_T is a point, i.e. $(A_N, B_N) = (|, \cdot)$. The Morse data is

$$(A, B) = (\cup, \emptyset) \times (|, \cdot) = (\blacksquare, _).$$

and the opposite side of ∂A where p is, attaches to B . This surgery is equivalent to attaching a 0-cell to the boundary and a 1-cell to the interior.

We now make a summary of the effect of each critical point on the cell complex once we have made the appropriate deformation retracts.

1. Interior maximum: attach a 2-cell to the 1-cells or to the boundary curves.
2. Interior saddle: attach a 1-cell to the 0-cells or to a point of the boundary.
3. Interior minimum: add a 0-cell (a point) to the skeleton.
4. Local maximum of S located in ∂S : attach a 2-cell to the 1-cells and attach a 1-cell to a point of the boundary.

5. Boundary maximum (not of S): attach a 1-cell to a point of the boundary.
6. Local minimum of S located in ∂S : add a 0-cell to the skeleton (this point will be on the boundary).
7. Boundary minimum (not of S): add a 0-cell (boundary point) to the skeleton and attach a 1-cell to a local minimum (0-cell) or to a point of the boundary.

In order to construct the complex, we first add all the 0-cells (all interior and boundary minima and possibly some boundary points), then the 1-cells (the boundary curves, the 1-cells corresponding to each saddle point and the 1-cells joining boundary minima to local minima or boundary points) and finally we add the 2-cells corresponding to each local maximum.

5.3 EXTENSION TO POINT-CLOUDS

In this section, we outline the main ideas of the reconstruction algorithm for point-clouds. First, we explain the case without boundary and then with boundary; in the next section, we will give full details for both cases.

Let $X \subset \mathbb{R}^N$ be a point cloud sampling a compact surface S , possibly with boundary. Determine the neighbors of each point p in the sample, e.g. proceeding as outlined in Section 5.4. Choose a unit vector $v \in \mathbb{R}^N$ such that the height function $f(p) = p \cdot v$ has different values in all points of X , and define the positive gradient (or *upwards*), resp. negative gradient (or *downwards*) flows of f by sending every point p in the cloud to its neighbor that maximizes the slope of growth of f , resp. makes f decrease with the most negative slope. Points where f cannot grow, resp. decrease, are local maxima, resp. minima of f in X . The delicate critical points to compute are saddles.

Interpret the downwards flow of f on X as an embedded graph whose vertices are the points in the cloud and the edges are given by connecting each point in the cloud to its downwards neighbor. The intersections of this graph with the hyperplane $x \cdot v = c$ can be considered point samples for the level set $f^{-1}(c)$ on the original surface S , with some noise added by the linear interpolation. This level set consists of point samples of curves, either closed or with edges in the boundary of S . These level curves can be reconstructed, e.g. as explained in Section 5.4, identifying the different components.

Perform these level set intersections at n different levels $c_i = c_0 + i \cdot h$ ranging from $c_0 = \min f(X)$ to $c_n = \max f(X)$. The number of level sections n must be selected, the idea is that all surface *features* (e.g. saddle points) whose range in height is h or greater will be detected.

In Section 5.4 we will explain how changes in the topology of the level set correspond to variations in the number or type of curves. For example, going over a saddle point of f can be tracked by detecting two pairs of neighbors in different connected components of a level set which end up in the same connected component of the next level set after applying the downwards flow (see Figure 5.4).

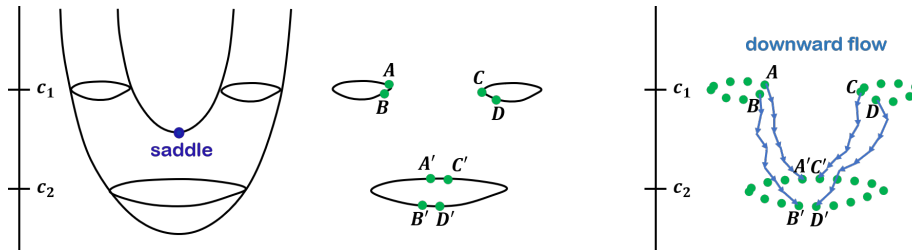


Figure 5.4: Change in level set when crossing a critical value in a surface (left) and point cloud (right): note the change in neighbors among the 4 marked points after the flow.

Following the 4 points in these 2 pairs in the downwards flow, and their pairing according to closeness, the level at which the pairing changes marks the position of the saddle point (see Figure 5.4). The *unstable variety* of this saddle point for the flow of $-\nabla f$ (i.e. the 1-cell joining the saddle point to local minima) is approximated by taking the two pairs of neighboring points at the level of f immediately below the saddle point ($\{B', D'\}$ and $\{A', C'\}$ in the figure), and averaging the downward orbits of each pair (which end up in a minimum, but not necessarily the same). These computations have a margin of error $O(d)$, where d is the local variation of height among neighboring cloud points. Once the saddle points of the height function and their (un)stable varieties have been found, the piecewise parametrization for the entire surface without boundary follows: 0-cells are the local minima, 1-cells have been parametrized at the saddle point detection, and each 2-cell can be identified from the tree formed in itself by the upwards flow to its unique maximum. Finally, the boundary relations given by the downwards flow on the cells give us the Morse-Smale complex and singular homology of the surface S .

The case of surfaces with boundaries poses several complications that we will discuss in detail in the next section, but the main aspect to take into account with respect to 1-cells is that when we have boundary minima (i.e. a critical point of $f|_{\partial S}$ that is a minimum when restricted to the boundary but not in the whole point-cloud) we must add to the cellular decomposition of S a 1-cell that joins the boundary minimum with a local minimum or to a boundary point of the surface. This is

simply obtained by applying the downwards flow repeatedly starting at the boundary minimum.

5.4 PRACTICAL IMPLEMENTATION

In this section, we give detailed algorithms for all the ideas presented in the previous section.

5.4.1 *Neighbors identification*

The first step is the identification of a set of neighbors of each point v in the cloud $X \subset \mathbb{R}^N$. There are two classical approaches:

1. k -nearest neighbors (KNN): given a value for k and a point v , the k nearest points $\{v_1, \dots, v_k\}$ with respect to the Euclidean distance are declared as its neighbors. This is quite efficient to compute but runs into problems when the point-cloud has irregular densities and k is not big enough, e.g. when all the closest points to v are clustered at one side of it and do not *enclose* the vertex (see Figure 5.5, left).
2. Voronoi-Delaunay neighbors: we perform Voronoi's cellular decomposition of the point-cloud and then declare as neighbors of v the points v_i with neighboring cells (i.e. are connected to it by an edge in the Delaunay triangulation). This has the virtue of enclosing the vertex v even with irregular densities, but it can be expensive to compute and produces neighbors which are too apart from each other (see Figure 5.5, right).

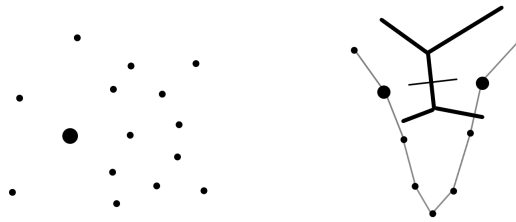


Figure 5.5: Typical problems associated to k -nearest neighbors (left) and Voronoi neighbors (right). On the left, the majority of the closest points to v are clustered at one side of it. On the right, vertices that are too far apart from each other have a neighboring cell.

Therefore we merge these two criteria, and declare two points as neighbors when (i) each point is among the k -nearest neighbors of the other and (ii) their Voronoi cells in the decomposition of the ambient space \mathbb{R}^N induced by X are adjoining. In order to be efficient we first choose a $k \approx 12$ to compute the k -nearest points and then

only keep as neighbors the vertices that are connected to v in the Delaunay triangulation of these few points. Finally, the relationship of neighborhood is made symmetric by reciprocating neighboring relationships where needed. The neighbors of v will be denoted by $\text{Neigh}(v)$. This produces a (locally non-planar) graph, which gives an idea of the local structure of X , but which will be in general very complicated.

Remark 5.4.1 (Pruning of the point-cloud). If the original cloud X has a very irregular density, it can be wise to discard some points. In order to do this, several heuristics can be employed, one of them is to estimate the density at each point (e.g. the reciprocal of the logarithm of the distance to the closest point) and then discard points whose density is less than the mean plus a fixed multiple of a standard deviation. Special care must be taken to avoid removing complete clusters of points (at least one should be kept).

5.4.2 Tangent space estimation

This task is performed through *Principal Component Analysis*: if the point v and all its neighbors $\text{Neigh}(v) = \{v_1, \dots, v_k\}$ were co-planar we would have that for every i : $\langle \vec{v}_i, n_j \rangle = 0$, where n_j are all the normal vectors to the surface (recall that in general we are in \mathbb{R}^N). Since in general this will not be the case, we find the n_j 's by minimizing the function $\sum_j \sum_{i=1}^k \langle \vec{v}_i, n_j \rangle^2$. This is equivalent to finding the regression plane in the least squares sense, and it can be done efficiently by means of a *singular value decomposition* of the matrix with vectors \vec{v}_i as rows.

5.4.3 Boundary recognition

Once we have an estimation of the tangent spaces, in principle a boundary point of the surface can be easily identified because after orthogonally projecting it and its neighbors on its tangent plane, they cluster in a semi-space (see the first panel of Figure 5.6). Nevertheless, this method is difficult to implement robustly (e.g. on points of high curvature of the boundary curves, see the second panel of Figure 5.6). In order to obtain a robust detection, the idea will be to declare points as lying on the boundary only when the projections do not enclose the point.

In points with high curvature where the tangent plane may not be perfectly estimated the previous method can give false positives. In order to overcome this difficulty, we will also project the point and all its neighbors in the tangent planes estimated for the neighbors. We will build a graph for every projection and only declare v as boundary point when none of the graphs enclose v .

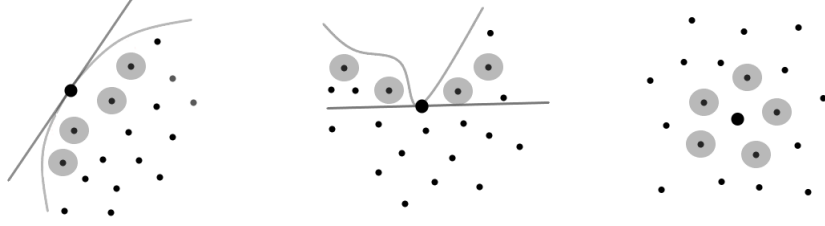


Figure 5.6: In principle, a boundary point can be identified easily because after projecting it and its neighbors on the tangent plane, they cluster in a semi-space of \mathbb{R}^2 (left panel). Nevertheless, this is not always the case for every boundary point (middle panel). To overcome this, we declare a point as a boundary point when none of the plane projections of its neighbors enclose the point (right panel).

Now we proceed to describe our method in detail: let $v \in X$, and $T_v S$ the estimated tangent plane at v . We follow the following steps:

1. Given $\{v_1, \dots, v_k\}$ the neighbors of $v_0 = v$ we project them on the planes $T_{v_j} S$ for $j = 0, \dots, k$. These projections will be denoted by $\pi(v_i)$.
2. We create a plane graph G_j with the projected points for every $j = 0, \dots, k$, where we add an edge between $\pi(v_i)$ and $\pi(v_l)$ only when v_i, v_l are themselves neighbors in the cloud and $i, l \neq 0$.
3. We declare the vertex v as a boundary point only when none of the plane graphs G_j enclose $\pi(v)$ for every projection to $T_{v_j} S$.

Remark 5.4.2. In point-clouds with irregular densities, the previous algorithm can still give false positives because of small boundary holes. In that case, clusters of boundary points with small diameter (i.e. clusters contained in small balls) are discarded.

5.4.4 Reconstruction of curves

Curve reconstruction from a point-cloud sample is used for boundary parametrization, and later for level set reconstruction when we intersect the gradient flow graph with level hyperplanes. We employ a variant of the NN-Crust algorithm described in [30]; which can be called KNN-Crust: we limit the search of edges to the k -nearest neighbors on the curve, usually with $k = 5$. The rest of the reconstruction follows as in [30].

Thus, from now on, we assume that boundary curves of the cloud (if present) have been reconstructed and parametrized.

5.4.5 Morse function and flows

We define $f : X \rightarrow \mathbb{R}$ as the height function $f(v) = \langle v, \nu \rangle$ where ν is a fixed unitary direction. We try several ν randomly until all the $\{f(v)\}_{v \in X}$ are different and the number of local maxima of f (the future number of 2-cells) is small. In the presence of boundaries, we also choose the direction ν that minimizes the number of critical points at the boundaries. This keeps the quantity of boundary 1-cells small.

In order to compute the maxima and minima, we define the **upward flow** of f as the function $\text{Up} : X \rightarrow X$ such that

$$\text{Up}(v) = \operatorname{argmax}_{v_k \in \text{Neigh}(v), f(v_k) > f(v)} \frac{f(v_k) - f(v)}{\|v_k - v\|}. \quad (5.1)$$

Analogously, the **downward flow** is the function $\text{Down} : X \rightarrow X$ defined by

$$\text{Down}(v) = \operatorname{argmin}_{v_k \in \text{Neigh}(v), f(v_k) < f(v)} \frac{f(v_k) - f(v)}{\|v_k - v\|}. \quad (5.2)$$

Then, when $f(v) > f(v_k)$ for every $k \in \text{Neigh}(v)$, we have a local maximum and redefine $\text{Up}(v) = v$. Likewise, when $f(v) < f(v_k)$ for every $k \in \text{Neigh}(v)$, we have a local minimum and write $\text{Down}(v) = v$.

Since we have identified and parametrized the boundary curves of X , we can define the two flows $\partial \text{Down}(\cdot)$ and $\partial \text{Up}(\cdot)$ analogously (considering that each boundary point has two natural neighbors in ∂X) so that they send boundary points to boundary points. Then we can easily compute boundary maxima (resp. minima). Notice that in general these points do not need to be local maxima (resp. minima) of the full cloud X .

Remark 5.4.3. Notice that in general we do **not** have the equalities $v = \text{Up}(\text{Down}(v)) = \text{Down}(\text{Up}(v))$ as we would in the continuous case.

5.4.6 Hyperplane sections and computation of critical values

Next, we perform n level set intersections at equispaced levels $c_i = c_0 + i \cdot h$ ranging from $c_n = \max f(X)$ to $c_0 = \min f(X)$ (in the case of surfaces with boundary it will be better to select levels not necessarily equispaced as explained in Remark 5.4.4). This means that we intersect the oriented graph G_{down} , with vertices X and edges given by $\text{Down}(\cdot)$ (i.e. two vertices $v, w \in X$ share an edge if $w = \text{Down}(v)$) and the boundary curves, with the planes $H_c = \{p \in \mathbb{R}^N : p \cdot \nu - c = 0\}$ (see Figure 5.7). These intersections $\Gamma(c) = G_{\text{down}} \cap H_c$ are plane point-samples of one-dimensional curves (possibly with boundary, i.e. intervals) that we can reconstruct and parametrize. Notice that

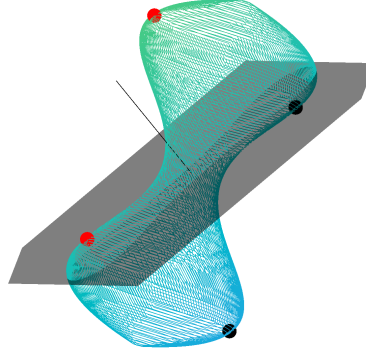


Figure 5.7: Intersection of the oriented graph G_{down} with a plane. Changes in the number of connected components of the reconstructed curves $\Gamma(c) = G_{\text{down}} \cap H_c \approx \cup \mathbb{S}^1$ tell us that we have crossed critical points.

by construction we can track the correspondence of points between different levels $\Gamma(c_{i+1})$ and $\Gamma(c_i)$ using the flow $\text{Down}(\cdot)$. Changes in the number or topological type of connected components of these level sets tell us that we have crossed critical points of f . We will study each possible case separately.

In order to get the correspondence of points between the levels $\Gamma(c_{i+1})$ and $\Gamma(c_i)$ using $\text{Down}(\cdot)$, we may need to apply the flow more than once depending on the size h of the jump between level sets that we have defined. The explicit correspondence is obtained as follows: for each node v , we consider the trajectory given by the sequence $v^n = \text{Down}(v^{n-1})$ where $v^0 = v$ and then we intersect this polygonal curve with the planes $H_{c_{i+1}}$ and H_{c_i} respectively.

Remark 5.4.4. When the surface has non-empty boundary we select level sections that are not necessarily equispaced, so that between two consecutive critical points of f there is always a regular level set (without any critical value) in between.

Case without boundary

When the underlying surface S of the point-cloud X has no boundary, the reconstructed curves $\Gamma(c)$ are all homeomorphic to the disjoint union of \mathbb{S}^1 and hence we can only have 3 cases:

1. *Apparition of a maximum:* when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ a new connected component appears (see Figure 5.8). These new points can **not** be reached from points of $\Gamma(c_{i+1})$ using the flow $\text{Down}(\cdot)$.
2. *Disappearance of a minimum:* when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ an existing connected component disappears (see Figure

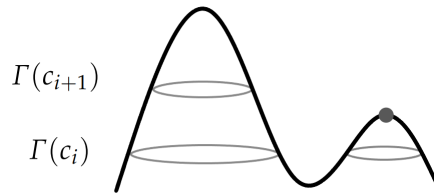


Figure 5.8: Apparition of a maximum when following the downwards flow: a new connected component appears.

5.9). These points do **not** go to points of $\Gamma(c_i)$ using the flow $\text{Down}(\cdot)$.

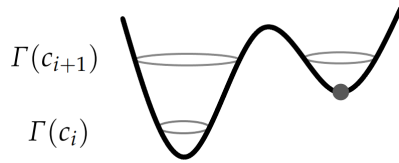


Figure 5.9: Disappearance of a minimum when following the downwards flow: an existing connected component disappears.

- 3. *Apparition of a saddle*: when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ a connected component appears or disappears (see Figure 5.10). Points on any component of $\Gamma(c_{i+1})$ have images in every component of $\Gamma(c_i)$ using the flow $\text{Down}(\cdot)$.

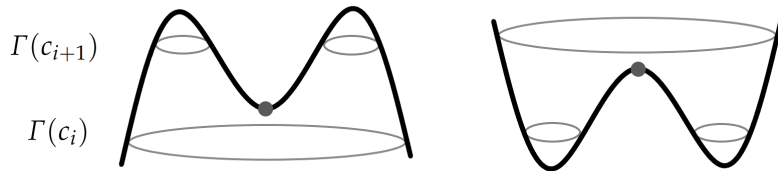


Figure 5.10: Apparition of a saddle point when following the downwards flow: a connected component appears or disappears.

A summary of the generic (i.e. non-degenerate) level set transformations that we get locally around each critical point can be seen in Figure 5.11.

Case with boundary

When the underlying surface S of the point-cloud X has a boundary ∂S , the reconstructed curves $\Gamma(c)$ are homeomorphic to a union of S^1 and closed intervals $[0, 1]$. These intervals always join two points

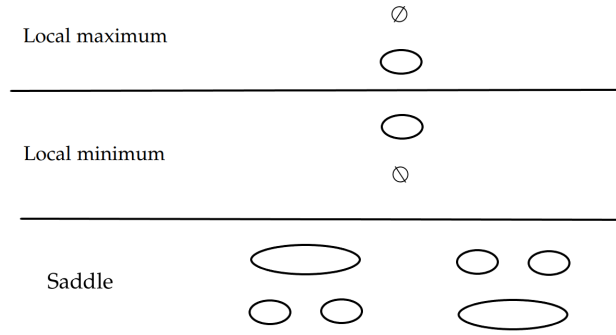


Figure 5.11: Different generic and local level set transformations for surfaces without boundary. The reconstructed curves $\Gamma(c)$ are homeomorphic to a disjoint union of S^1 .

of ∂S , introducing a bordism equivalence relation in $\partial S \cap H_c$. Therefore, we have the three previous cases, plus two new ones, since now we can have minima and maxima at the boundaries. Saddle points can not be located at the boundaries since that would violate the third condition of Definition 8 as already explained. We will make a distinction between (local) minima (resp. maxima) of the whole point-cloud, and boundary minima (resp. maxima) which are points that are critical when we restrict f to the boundary curves, but not in the whole surface. In Figure 5.12 we can see a summary of the generic (i.e. non-degenerate) level set transformations that we get locally around each critical point for all possible cases involving boundary points.

We can have the following six cases:

1. *Apparition of a local maximum*: this is similar as before, when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ a new connected component S^1 or $[0, 1]$ whose points can **not** be reached from $\Gamma(c_{i+1})$ using $\text{Down}(\cdot)$ appears.
2. *Disappearance of a local minimum*: similarly as before, when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ an existing connected component S^1 or $[0, 1]$ whose points do **not** go to $\Gamma(c_i)$ using $\text{Down}(\cdot)$ disappears.
3. *Apparition of a boundary maximum*: when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ either a S^1 splits into a $[0, 1]$ or a $[0, 1]$ opens in two intervals.
4. *Disappearance of a boundary minimum*: when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ either a $[0, 1]$ becomes a S^1 or two $[0, 1]$ become one interval.
5. *Apparition of a saddle*: this is the most complex case, when going down from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ a connected component S^1 or $[0, 1]$

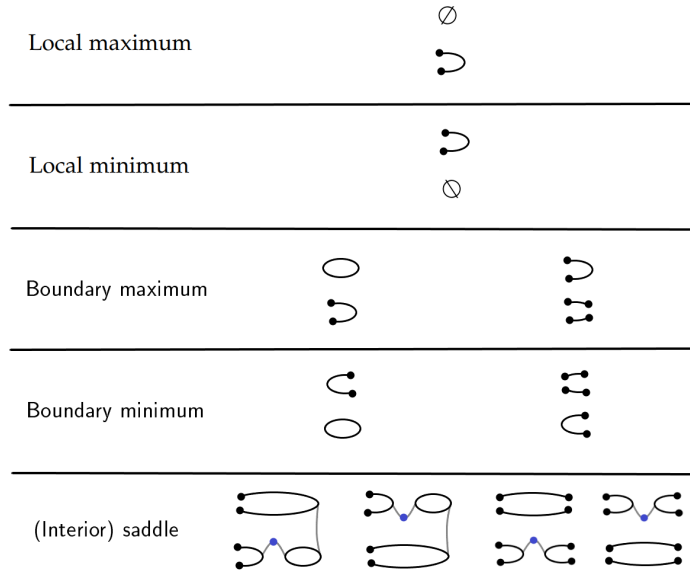


Figure 5.12: Different local level set transformations for all possible cases involving boundary points. The reconstructed curves $\Gamma(c)$ are homeomorphic to a union of S^1 and closed intervals $[0, 1]$. The previous cases (interior minima, maxima and saddles) are not shown again but can also occur. Saddle points are also drawn, in blue.

appears or disappears. We can also have a case where the number of connected components stays the same but in passing from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ the bordism pairing of boundary points of ∂S given by the $[0, 1]$ component changes. Points on any component of $\Gamma(c_{i+1})$ have images in every component of $\Gamma(c_i)$ using the flow $\text{Down}(\cdot)$.

5.4.7 Identification of Morse cells

In this section, we explain how to identify all the Morse cells of the Morse-Smale complex that give us a cellular decomposition of the surface (see Figure 5.13).

0-cells

They correspond to local minima or to boundary minima of f , we already found them; they are the fixed points of $\text{Down}(\cdot)$ and $\partial \text{Down}(\cdot)$ respectively.

1-cells

There are three cases that generate the 1-cells of the skeleton:

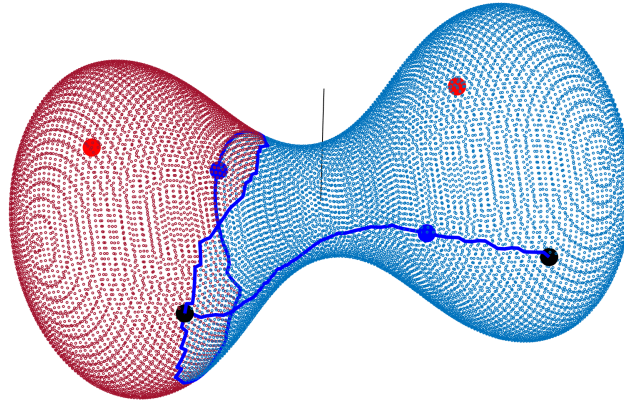


Figure 5.13: A sampled *dumbbell*: the black line is the direction of the height function; local maxima, resp. minima, are painted red, resp. black; saddle points are painted blue; their 1-cells are outlined in blue. There are two 2-cells: one in dark red (left) and one in light blue (right).

1. Boundaries: when S has non-empty boundary, the boundary curves are also part of the 1-skeleton. We already parametrized those. Each boundary maximum gives rise to a 1-cell that attaches to a 0-cell (point) located at the boundaries.
2. Saddle points: there are one-dimensional curves that go from one local minimum m_1 or boundary point to another (not necessarily distinct) local minimum m_2 or boundary point passing through the saddle point.
3. Boundary minima: there are 1-cells in the complex not associated with a saddle point or to boundary curves. In this last case, they always connect a boundary minimum to a local minimum or boundary point of the cloud. In order to compute this 1-cell we simply flow down every boundary minimum using $\text{Down}(\cdot)$.

Remark 5.4.5. When 1-cells introduced by saddle points or boundary minima end at points of the boundary ∂S not previously labeled as 0-cells, we introduce the point where they meet to the 0-skeleton and divide the 1-cell into two.

Computation of saddles and their 1-cells

We now explain how to compute saddle points and afterwards their associated 1-cells. Recall that we know when we are in the presence of a saddle: when going from $\Gamma(c_{i+1})$ to $\Gamma(c_i)$ a connected component appears, disappears or there is a change of bordism pairing of boundary points. In any case, all points of $\Gamma(c_{i+1})$ have images in $\Gamma(c_i)$ using the flow $\text{Down}(\cdot)$. Therefore, there exist four points v_1, v_2, v_3, v_4 in $\Gamma(c_{i+1})$ that can be paired by proximity (i.e. are neighbors in the level

set curves) as $\{v_1, v_2\}$ and $\{v_3, v_4\}$ whose images by $\text{Down}(v_i) = \tilde{v}_i$ are now paired differently as $\{\tilde{v}_1, \tilde{v}_3\}$ and $\{\tilde{v}_2, \tilde{v}_4\}$. The saddle point is approximated by the average:

$$s = \frac{1}{8} \left(\sum_{i=1}^4 v_i + \sum_{i=1}^4 \tilde{v}_i \right).$$

One branch of the 1-cell is obtained by taking the average of the two orbits generated by $\text{Down}(\cdot)$ starting from $\{\tilde{v}_1, \tilde{v}_3\}$. The other branch is approximated in the same manner but starting from and $\{\tilde{v}_2, \tilde{v}_4\}$. This averaging operation generates new points not previously in the cloud. All these new points are added to the cloud, declaring as neighbors of s the 8 points used for its computation, and for each new point of the branches its two neighbors in the 1-cell plus the two points that were used for its computation. The flows $\text{Down}(\cdot)$ and $\text{Up}(\cdot)$ are simply defined at those points by following the newly created trajectories down or up (except the saddle points which are fixed points of both flows). This ensures that this newly defined curve (the 1-cell) is invariant by both flows.

2-cells

There is one 2-cell for each local maximum of f (recall that boundary maxima only generate 1-cells, i.e. the boundary curves), we already found them (they are the fixed points of $\text{Up}(\cdot)$), but we will now explain how to deduce which points of the cloud correspond to each 2-cell (or each maximum). The idea is simply to flow up every point $v \in X$ by $\text{Up}(\cdot)$ and see at which maximum it ends up. In symbols, this means that we consider the limit of the sequence $v^n = \text{Up}(v^{n-1})$ where $v^0 = v$ when $n \rightarrow +\infty$ (this limit exists because we have a finite number of points and maxima are fixed by $\text{Up}(\cdot)$). It can happen that the sequence $\{v^n\}_n$ converges to a saddle s . This can only happen for points of the 1-cells, but must be corrected for the rest: when a point v ends at a saddle s but it is not part of the 1-cell, we look at the neighbors $\text{Neigh}(v)$ and see in which maximum they finished. The most common maximum among the neighbors is selected as the corrected destination of v . It may be necessary to iterate this procedure a finite number of times.

5.4.8 Attachment maps of the Morse cells

Now that we have the skeleton of S , i.e. the 0, 1, 2-cells, we encounter a delicate problem: figuring out the attachment maps between the cells. We already know how the 1-cells attach to the 0-cells. We now discuss how 2-cells attach to 1-cells. We must figure out which 1-cells are the boundary of the 2-cells, in which order, orientation and how many times they appear: once (when they are part of the boundary of

S or when they attach to a different 2-cell) or twice (when they will be, formally, two different sides of the 2-cell that are identified, see Figure 5.13). We first discuss the case without boundary.

Smooth case

We first describe the process for the smooth case (without boundary) and then explain how to adapt it to point-clouds. Notice that the 2-cell corresponding to each maximum \hat{m} , let us call it $D_{\hat{m}}$, is the set of points that converge to \hat{m} under the flow $+\nabla f$ (in Dynamical System terminology, its stable manifold). The main idea to deduce how the boundary of $D_{\hat{m}}$ attaches to the 1-cells is to choose a simple closed curve around each maximum and flow it down by $-\nabla f$ until it reaches the 1-cells. To achieve this properly, we must perturb f so that in a neighborhood of the 1-cells the gradient flow is *transversal* (i.e. not parallel) to the 1-cells. This is needed since otherwise the points of $D_{\hat{m}}$ would converge to minima under the downwards flow $-\nabla f$. In order to obtain this perturbed \tilde{f} we consider an arbitrary small tubular neighborhood around the 1-cells and a normal vector field to the 1-cells. These two flows are joined smoothly in the tubular neighborhood with the aid of a partition of unity (i.e. bump functions, see [52]).

Discrete case

In the discrete case, we choose a simple (i.e. without self-intersections) closed curve of neighbors around each maximum (a *crown*) and flow it by $\text{Down}(\cdot)$ until it reaches the 1-cells. We do this in a way such that when a point of the curve has neighbors that are in the 1-cell, we stop following the flow $\text{Down}(\cdot)$ and match the point in question to the point in the 1-cell with results in the most negative downwards slope. This method would work perfectly if simple closed curves were preserved by $\text{Down}(\cdot)$; since this is not the case, we must *repair* the curve every time we flow it down. There are three main situations:

1. *Splitting of points*: it may happen that two points w_1 and w_2 were neighbors but $\text{Down}(w_1)$ and $\text{Down}(w_2)$ are not. In that case, we define the sub-graph of neighbors given by points

$$\{v \in X : f(v) \leq \max [f(\text{Down}(w_1)), f(\text{Down}(w_2))]\},$$

and compute the shortest path joining $\text{Down}(w_1)$ to $\text{Down}(w_2)$ (taking as the distance of a path, that given by the edges of the graph).

2. *Collapse of points*: it may happen that two points w_1 and w_2 have the same image $\text{Down}(w_1) = \text{Down}(w_2)$. In that case, we simply delete one of the repeated points from the curve.

3. *Creation of spikes*: it may happen that one of two neighboring points w_1 and w_2 has image $\text{Down}(w_2) = w_1$. This could happen to more than one point at a time. In that case, we simply delete the *spiky* points (i.e. $\text{Down}(w_2) = w_1$) in the new curve.

The final problem we face is that the flowing crown may arrive at a stage where it is stationary by the downward flow but some points of it have not reached the 1-cells. In that case, we pair each unmatched point of the crown with the closest (as measured by the edges of the neighboring relationship of the cloud) point of the 1-cells.

We now have a matching between the initial crown and the 1-cells (although not a bijection). But this is enough to deduce which 1-cells are the boundary of $D_{\hat{m}}$ (only the 1-cells that are reached), in which order (this can be deduced since the crown is parametrized), the orientation (again thanks to the parametrization) and how many times they appear (once or twice, depending on the matching).

Attachment maps for surfaces with boundaries

When the surface S has a boundary ∂S the boundaries of the 2-cells are no longer only the 1-cells derived from the saddle points, but also possibly curves of ∂S . We distinguish several cases depending on the type of maximum we have:

1. Interior maxima (not belonging to ∂S): as before we flow down a crown around the maximum until it reaches a point that is either a neighbor of a boundary point or of a 1-cell. In this way, we obtain a matching between the boundary of the 2-cell and the 1-cells and boundary curves of S .
2. Boundary maxima (but not a local maximum of S): they do not intervene in the attachment maps of 2-cells to 1-cells (although they give rise to a 1-cell in the boundary).
3. Local maxima (located at ∂S): we consider a *semi-crown* around the maximum \hat{m} , meaning that we take an arc of the boundary centered at \hat{m} and complete it with neighboring interior nodes so that we obtain a closed simple curve. Then we flow down this semi-crown in such a way that the arcs of the boundary stay in the boundary (we use the restricted flow $\partial \text{Down}(\cdot)$) and the interior nodes of the curve flow down normally by $\text{Down}(\cdot)$. As before we flow down the semi-crown until each point of it reaches a node that is either a neighbor of a boundary point or of a 1-cell.

For an example of this process in action, see the video at <https://youtu.be/8cgr54oRf6w>.

5.4.9 Parametrization of the 2-cells

Once we have each Morse cell and their attachment maps identified, the last problem we face is how to parametrize the 2-cells, i.e. finding a flat domain $D \subset \mathbb{R}^2$ and a map $\phi : D \rightarrow X$, such that each $\phi(D) \subset X$ corresponds to one of the 2-cells found before (see Figure 5.14). We will further require D to be a convex polygon and that ∂D is isometric to $\phi(\partial D)$. Once we have a parametrization of each 2-cell, since they attach well (their boundaries are the 1-cells), we have a full piece-wise parametrization of X .

Remark 5.4.6. We may require ∂D to also have the same number of sides of $\phi(\partial D)$ (i.e. the different 1-cells), their length and their order (these are known in advance and will be determined by the attachment map of the 2-cell to the 1-cells). In that case, we denote the sides' lengths by l_1, \dots, l_d .

In order to find D and ϕ , we follow two steps:

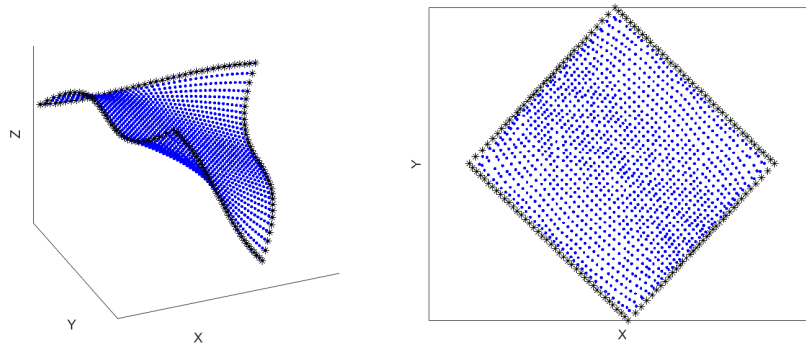


Figure 5.14: Parametrization of a 2-cell (left) using a rectangle D with isometric sides to the boundary of the 2-cell (right).

1. Obtain a convex polygon $D \subset \mathbb{R}^2$ whose sides are l_1, \dots, l_d : we find the polygon with maximal area and predetermined sides l_1, \dots, l_d inscribed in a circumference. To do so, we solve an optimization problem in order to find the circumference in question. By a result of Bramagupta [13], this is done by finding the radius that makes the polygon's interior angles add up to 2π . Once we have D , we obtain a bijection between ∂D and the points of the corresponding 1-cells in X .

Alternatively, we can take any convex polygon in the plane whose boundary ∂D is isometric to the boundary of the 2-cell (e.g. a rectangle).

2. Obtain a correspondence of interior points of D mapping to the cloud points in the 2-cell: for each interior point v in the 2-cell we

want to find an interior point $x = \phi^{-1}(v)$ in D . We assume that each point p_i of D (including ∂D) is a barycentric combination of its neighbors (as given by the neighboring relationship of the cloud, see [37]) with Tutte's weights (all coefficients are equal to $\frac{1}{|\text{Neigh}(v)|}$). Then we have to solve a linear system with a unique solution (see [37]), which in turn gives us the coordinates of interior points of D mapping to the cloud.

Once we have these points, we can extend the parametrization to the whole interior of D by taking its Delaunay triangulation with the newly found vertices, and interpolating linearly for the images. Then we can re-mesh (or even quadrangulate) the polygon D if desired. This re-meshing allows us to obtain C^∞ parametrizations of the surface via splines, to de-noise the point-sample if needed or to obtain a synthetic resampling of the point-cloud with better regularity properties.

Remark 5.4.7. As already mentioned, the polygon does not need to have the same number of sides as the 2-cell, in fact, in order to apply some de-noising algorithms it is necessary to take the polygon D as a square or rectangle. Moreover, if the 2-cell is too far from being flat (e.g. because of high curvature) the algorithm may produce interior points in D with non-uniform densities and thus it may be better to subdivide the cell into smaller and flatter pieces and afterwards parametrize each piece independently. On the other hand, using a polygon where the sides are mapped isometrically to each of the bounding 1-cells, has the advantage of allowing the gluing of the resulting parametrizations of the 2-cells into a global piece-wise defined and continuous parametrization of the entire surface.

5.4.10 Results

In this section, we reconstruct three different surfaces (one without boundary –a *torus*– and two with it –a *vest* and a pair of *pants*–) which pose various challenges to the presented algorithm. The first two point-clouds are synthetic whereas the last one is a real scan of an actual textile. The three cases are challenging and interesting for different reasons: the *torus* is really slim and its embedding describes a (2,3)-toric knot which causes far away parts of the surface (as measured by geodesic distance) to be really near each other in euclidean space. The *vest* was obtained by cutting out parts of an ellipsoid in order to obtain a surface with the same topology as an open vest. Therefore, it has positive Gaussian curvature everywhere and very large boundary curves. Finally, the *pants* are a 3D-scan of a real pair of jeans and thus its point-cloud presents wrinkles, noise and an irregular density distribution of points.

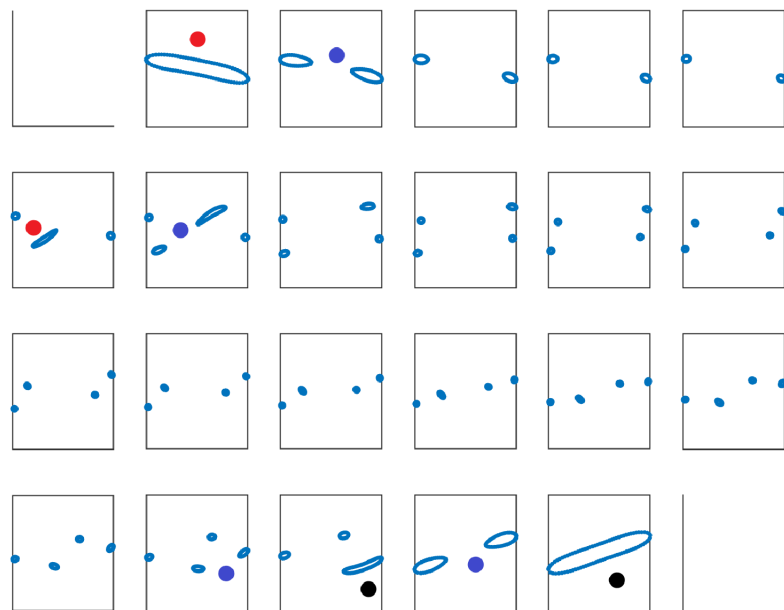
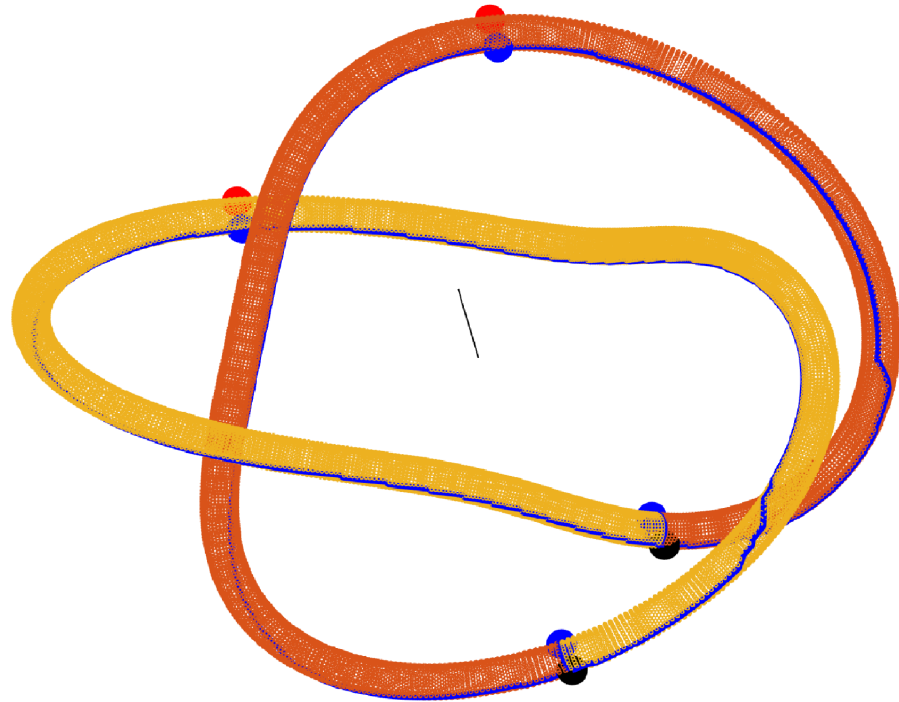


Figure 5.15: A sampled knotted torus: the black line is the direction of the height function; local maxima, resp. minima, are painted red, resp. black; saddle points are painted blue; their 1-cells are outlined in blue. On the bottom, we plot the level set curves highlighting when critical points appear.

Knotted torus

Figure 5.15 shows our algorithm applied to a point-sample from a torus embedded in \mathbb{R}^3 along a (2,3)-toric knot. The algorithm correctly detects 2 local maxima, 2 local minima and 4 saddle points for the height function depicted in the figure. Out of a point cloud of 30 000 points, a decomposition of the surface into 8 Morse cells is found (two 0-cells: the minima, four 1-cells associated to the saddle points and two 2-cells, one for each maximum). On the bottom of Figure 5.15 we display the level set curves $\Gamma(c) = G_{\text{down}} \cap H_c$ (see Section 5.4.6). Notice that for this point-cloud we have all possible local transformations of level-set curves for surfaces without boundary (see Figure 5.11).

Ellipsoidal vest

Figure 5.16 shows our algorithm applied to a sample of 36 000 points from a vest embedded in \mathbb{R}^3 . The cloud was obtained by cutting out parts of an ellipsoid in order to obtain a surface with the same topology as an open vest. After detecting and parametrizing the boundary successfully, the algorithm correctly detects 2 local maxima, 2 boundary maxima, 1 local minimum and 3 boundary minima for the height function depicted in the figure. All critical points are located at the boundary. Moreover, two new points (shown in purple) are added where the 1-cells meet each other or the boundary curves (see Remark 5.4.5).

Then, a decomposition of the surface into 16 Morse cells is found (five 0-cells, nine 1-cells and two 2-cells). In order to deduce how the two 2-cells attach and which 1-cells are their boundary we apply the curve flow explained in Section 5.4.8. This process in action for one of the 2-cells can be visualized at <https://youtu.be/8cgr54oRf6w>. Thus, we deduce how the cells attach with each other (e.g. the 1-cell number 5 appears on both 2-cells and it is precisely one of the curves where they attach, see Figure 5.16). From this, we recover the entire topology of the vest.

In Figure 5.17 we show a parametrization by a rectangle of one of the 2-cells of the vest (the red one on the right in Figure 5.16). This is done as explained in Section 5.4.9: the bounding 1-cells are mapped isometrically to a flat rectangle (notice that we consider the 1-cells number 6 and 6' as different) and then interior points are obtained using the neighboring relationships of the cloud (taking care of removing neighbors at opposite sides of the 1-cell number 6). Finally, the 2-cell consisting of 27 000 points (shown in red in Figure 5.17) is down-sampled using the parametrization and interpolating linearly to a cloud of 900 points (shown in blue Figure 5.17).

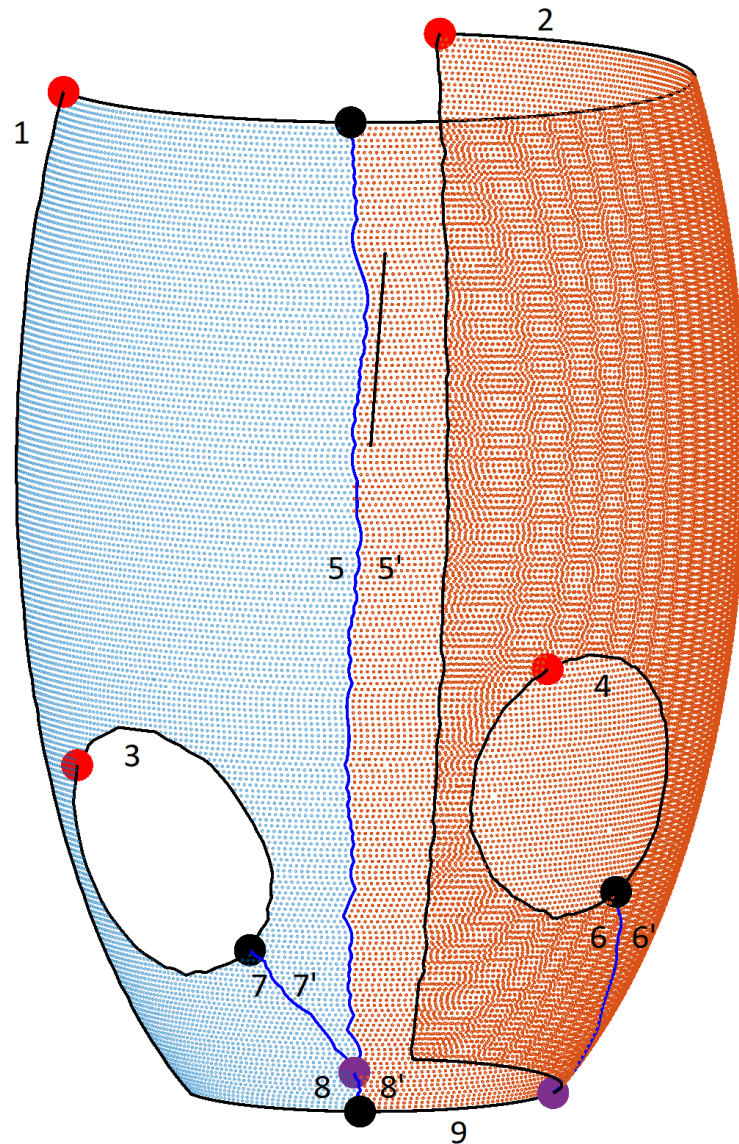


Figure 5.16: A sampled vest: the black line is the direction of the height function; maxima are painted in red, minima in black; 1-cells corresponding to boundary minima are outlined in blue and the boundary curves in black. The two purple points where the 1-cells meet each other or the boundary curves are added to the decomposition. The numbers correspond to the different formal 1-cells that, when identified (e.g. 7 with 7'), reconstruct the entire surface from 2 pieces homeomorphic to disks.

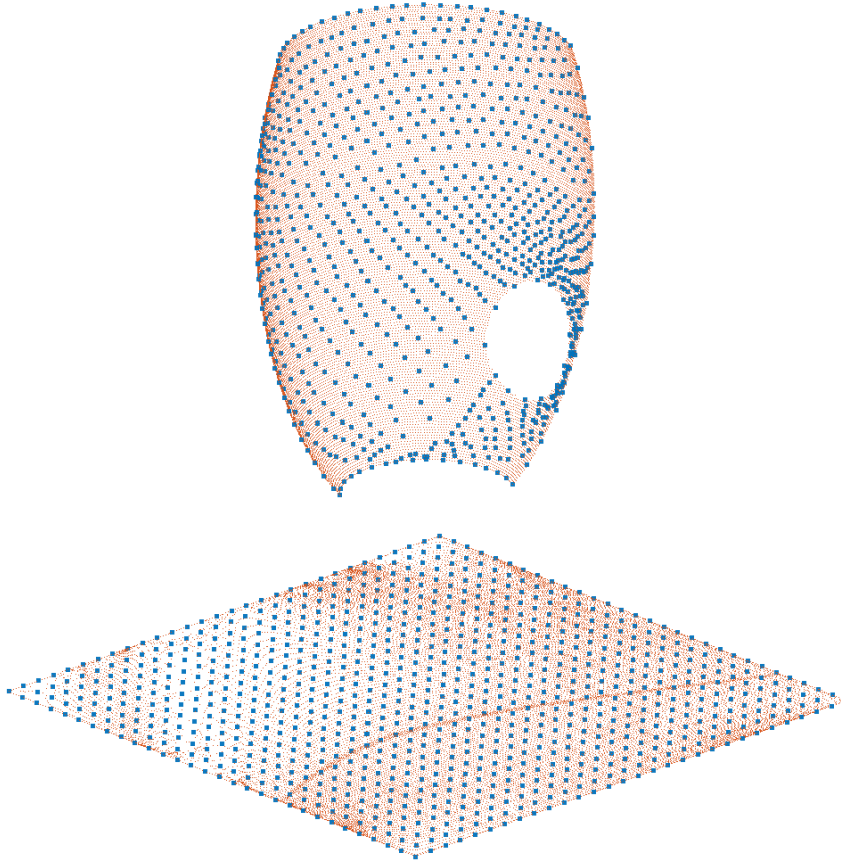


Figure 5.17: Parametrization by a rectangle of the rightmost 2-cell of Figure 5.16. The bounding 1-cells are mapped isometrically to a flat rectangle and then interior points are obtained using the neighboring relationships of the cloud. The 2-cell consisting of 27 000 points (shown in red) is down-sampled interpolating linearly to a cloud of 900 points (shown in blue).

3D-scan of pants

Figure 5.18 shows our algorithm applied to a 3D-scan of a pair of real jeans. The scan was made by a *Artec Eva* professional handheld 3D scanner while a person was wearing the garment. The 3D scan is processed with the proprietary software *Artec Studio* in order to obtain a point-cloud sample. In order to apply our algorithm and to reduce irrelevant details and noise, we down-sample the initial cloud of more than 500 000 points to 11 000 by using a *box-grid filter*, i.e. an axis-aligned bounding box is computed for the entire point-cloud and then divided into grid boxes. Points within each grid box are merged by averaging their locations.

Still, the cloud has a lot of detail (e.g. wrinkles) that cause the proliferation of critical points and hence of Morse-cells. After computing for each point its neighbors as explained in Section 5.4.1, we apply a

discrete-curvature filter to the point-cloud. This means that we substitute each point v for a weighted average of its position and the location of its neighbors:

$$h(v) = \alpha v + (1 - \alpha) \frac{1}{|\text{Neigh}(v)|} \sum_{v_i \in \text{Neigh}(v)} v_i,$$

where in our case we take $\alpha = 0.4$. When applied a small number of times (in our case 8 times) this filter defines a bijection between the original cloud and the filtered one, which preserves the topology of the underlying surface. Hence, the decomposition we find for the filtered case is still valid and topologically accurate for the original cloud, which is the one shown in Figure 5.18.

The algorithm correctly detects 1 local maximum, 2 boundary maxima, 2 local minima, 1 boundary minimum and 1 saddle point for the height function depicted in Figure 5.18. A decomposition of the surface into 8 Morse cells is found (three 0-cells: the minima and a point added because the original 1-cells intersect, six 1-cells associated to the saddle points, boundary minima and the boundary curves and one 2-cell, for the local maximum). On the bottom of Figure 5.18 we display the level set curves $\Gamma(c) = G_{\text{down}} \cap H_c$ (see Section 5.4.6). Notice that for this point-cloud we encounter several of the possible local transformations of level-set curves for surfaces with boundary shown in Figure 5.12.

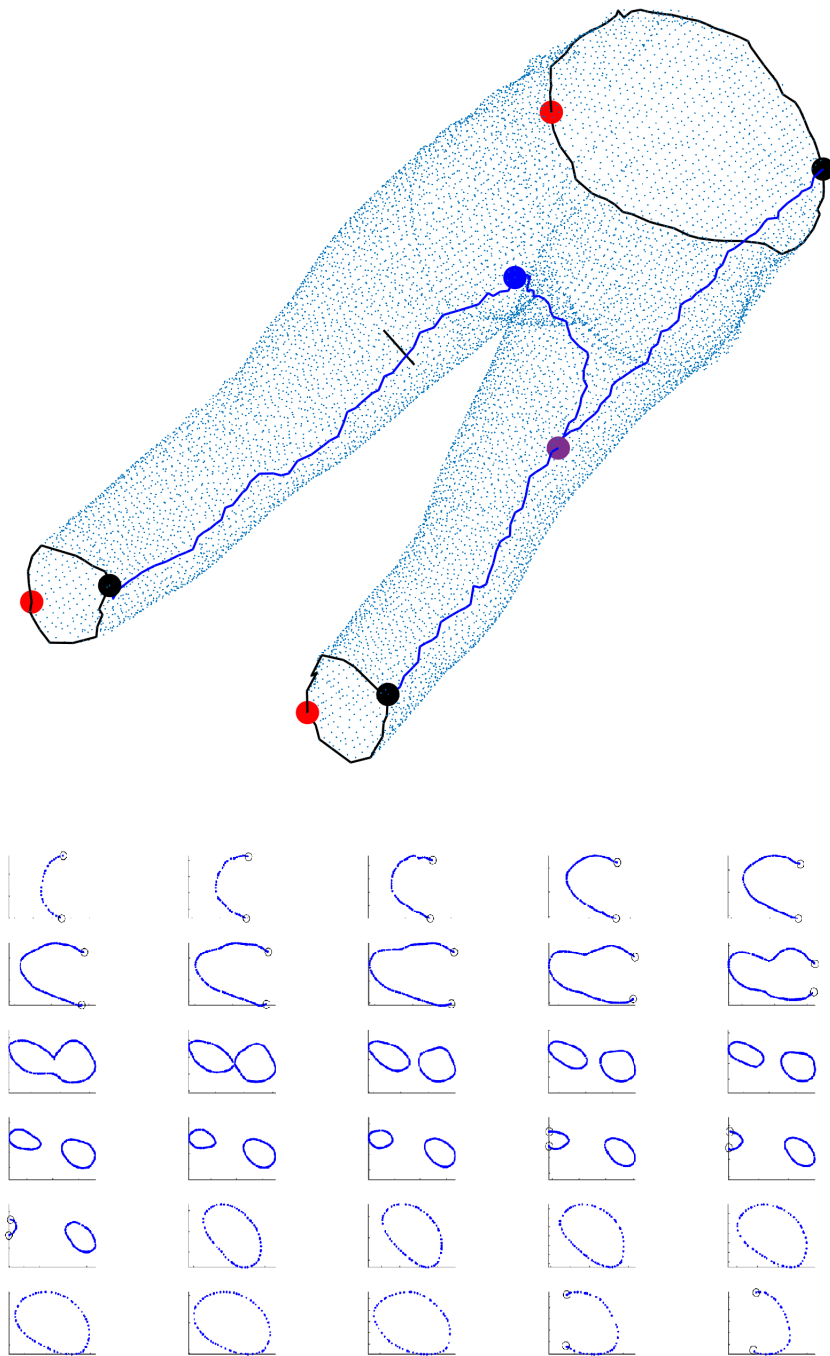


Figure 5.18: A 3D-scan of real pants: the black line is the direction of the height function; maxima are painted red, minima in black, saddle points are painted blue; 1-cells are outlined in blue and the boundary curves in black. The purple point where the 1-cells meet is added to the decomposition. On the bottom, we plot the level-set curves obtained by intersecting the surface with planes perpendicular to the height function.

Part III

CLASSIFYING CLOTH STATES

In this final part we study the configuration space of a piece of cloth, i.e. the space of all surfaces in Euclidean 3-space isometric to it. First, we show that under certain conditions the boundary curve of an inextensible textile determines completely its position in space. Afterward, we introduce the dGLI cloth coordinates, a low-dimensional representation of the state of a rectangular piece of cloth based on the position of its boundary. These intrinsic coordinates enable the recognition and classification of high-level states, allowing us to classify cloth configurations into states that we identify as different.

The original state of a piece of cloth is flat, so the set of possible states under the inextensible assumption is the set of developable surfaces isometric to a fixed one, which, using the Morse cellular decompositions of part II of this work if necessary, we may assume to be a domain R in the plane. In this chapter, it is proved that a generic simple, closed, piecewise regular curve in space can be the boundary of only finitely many developable surfaces with nonvanishing mean curvature. This implies that during a continuous cloth motion, the position of a garment is determined by the location of its boundary. The relevance of this result justifies theoretically our choice of developing cloth coordinates for semantic classification based only on the position of the boundary of the cloth (see Chapter 7).

In general, to study theoretically the dynamics of a piece of cloth with the Lagrangian formalism, we need adapted coordinates on the set of its states (akin to using an angular coordinate for a pendulum as opposed to cartesian coordinates). Apart from their theoretical appeal (see the Helfrich Hamiltonian of membrane dynamics [29]), such intrinsic, analytic coordinates should allow the formulation of discretization schemes where the resulting mechanics are more independent of how the garment is meshed, more frugal in computation time, and closer to reality.

Section 6.1 discusses two candidates for the role of generalized coordinates in the space of states of a developable surface, and explains their common limitation from the viewpoint of their application. Section 6.2 proposes an alternative approach: to track the motion of the surface by following its boundary. This is not straightforward because the boundary does not determine the position of the surface, but as we explain below Theorem 4 is a step in this direction.

6.1 THE SPACE OF DEVELOPABLE SURFACES

Definition 10. Given a smooth surface S embedded in \mathbb{R}^3 , we say it is *developable* if its Gaussian curvature K is 0.

These surfaces are exactly the surfaces that are locally isometric to a domain in the Euclidean plane \mathbb{R}^2 .

Definition 11. Given a developable surface S , we say it is *ruled* if through every point of the surface there is a line that lies inside S .

Developability places a strong constraint on the local structure of the surface (see [36]):

Theorem 2 (structure theorem for developable surfaces). A \mathcal{C}^3 developable surface S embedded in \mathbb{R}^3 has an open subset that is ruled, with unit normal vector constant along each line of the ruling but varying in a transverse direction. Every connected component of its complement is contained in a plane.

This structure can be deduced from the Gauss map of the surface: Gaussian curvature 0 makes its rank 0 or 1, the latter rank being reached on an open subset of the surface. The normal vector is locally constant in the rank 0 subset. The dichotomy in the rank of the Gauss map, and varied classical notations, motivate:

Definition 12. A developable surface is *torsal* if the Gauss map has rank 1 on a dense open subset. *Flat patches* are connected subsets of a developable surface with nonempty interior where the Gauss map is constant, i.e. they are contained in a plane.

The subdivision of a developable surface into torsal and flat patches is given by the boundary of the vanishing locus of the mean curvature (the trace of the Gauss map) and is not necessarily simple.

Let us fix a planar domain $R \subset \mathbb{R}^2$ which is compact, contractible and has a piecewise \mathcal{C}^∞ boundary (e.g. a convex polygon). Define \mathcal{S} to be the set of all \mathcal{C}^3 surfaces in \mathbb{R}^3 isometric to R . These surfaces are all developable, and \mathcal{S} may be seen as the *space of states* of an inextensible (i.e. isometric for the inner distance) deformation of R in Euclidean space. The space of states \mathcal{S} can also be defined as the set of \mathcal{C}^2 maps from R to \mathbb{R}^3 which are isometries with the image. As such, it is endowed with the compact-open topology derived from the Euclidean one in R and \mathbb{R}^3 . This topology furnishes valuable tips for the study of \mathcal{S} : the set of surfaces containing flat patches has an empty interior because there exist arbitrarily small deformations making the normal vector nonconstant on an open set. Torsal surfaces are *stably torsal* if the mean curvature function intersects transversely the zero function. These surfaces form an open subset $\mathcal{U} \subset \mathcal{S}$, and suffice for our practical study of \mathcal{S} .

We can try to develop coordinates for the stably torsal state space \mathcal{U} based on the classical structure theorem. First, let us recall how to identify developable surfaces among the ruled ones:

Proposition 2 (classical, see [19]). A ruled surface parametrized as $\phi(u, v) = \gamma(u) + v \cdot w(u)$, where γ is a regular parametrized curve and w a vector field over γ , is developable if and only if the 3 vectors $\gamma'(u), w(u), w'(u)$ are linearly dependent for all u .

Given a regular \mathcal{C}^2 curve γ there is a way to obtain systematically such rulings over γ resulting in regular torsal surfaces:

Proposition 3. Let n be a unit normal \mathcal{C}^1 vector field over a regular, \mathcal{C}^2 curve γ with $n' \neq 0$. Then $w = n \times n'$ defines a torsal surface in

a neighborhood of γ . Moreover, all regular, torsal rulings over γ are generated by such w , and only $n, -n$ define the same torsal surface.

Proof. n is normal to γ' and w by their definitions, and $w' = n \times n''$ so at every u the vectors γ', w, w' are normal to n . Also, note that $w \neq 0$ because $n' \neq 0$. If \tilde{n} is another unit normal vector field such that $\tilde{n} \times \tilde{n}' = \mu w$ for some function $\mu(u)$ then note that \tilde{n} has to be normal to both w and γ' , hence a multiple of n . Finally, let us point out that if w is a nonvanishing tangent vector field over γ defining a torsal surface around it, then we can select a unit vector field n normal to γ', w, w' . The fact that n is normal to w and w' imply that n' is also normal to w , so $n \times n'$ is a multiple of w . \square

To define coordinates in the space of stably torsal surfaces S isometric to a fixed bounded domain R , the pairs (γ, n) of Proposition 3 run into a practical difficulty: the condition that $n' \neq 0$ forces the Gauss map to have rank 1. If S is a stably torsal surface with mean curvature H of varying sign, we must subdivide it by the $H = 0$ curves and parametrize separately each component of the complement. To follow a motion of the surface, one has to track the boundary shifts, mergers and splits of these components.

On the other hand, Ushakov proposes in [113] an alternative, PDE-based, coordinate scheme viewing developable surfaces as solutions of the trivial Monge-Ampère equation. From an analytic viewpoint, developable surfaces can be seen locally, once they have been parametrized in the form $z = z(x, y)$, as solutions of the so-called trivial Monge-Ampère partial differential equation, which is indeed the simplest of Monge-Ampère equations

$$\begin{vmatrix} z_{xx} & z_{xy} \\ z_{xy} & z_{yy} \end{vmatrix} = 0. \tag{6.1}$$

This is the case since a surface described as the graph of a function $z = z(x, y)$, has Gaussian curvature:

$$K = \frac{z_{xx} \cdot z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2}.$$

The PDE (6.1) almost provides a first set of coordinates for the space of torsal states \mathcal{U} . If we have a developable surface S which is stably torsal, isometric to R , and admits a (non-isometric) parametrization by orthogonal projection to a plane then S admits a parametrization of the form $z = z(x, y)$ and we can use

Theorem 3 (Ushakov, [113]). The general solution to the equation (6.1) containing no flat patches is given in parametric form by

$$x(u, v) = g(u) - v \cdot f'(u) \quad (6.2)$$

$$y(u, v) = v \quad (6.3)$$

$$z(u, v) = u \cdot g(u) - \int_0^u g(t)dt + v \cdot \{f(u) - u \cdot f'(u)\} \quad (6.4)$$

where $f(u)$ and $g(u)$ are arbitrary functions such that $f \in \mathcal{C}^2, g \in \mathcal{C}^1$, and $g'(u) \neq 0$ everywhere.

The functions f, g in Ushakov's theorem give us a curve $\gamma(u) = (g(u), 0, u \cdot g(u) - \int_0^u g(t)dt)$ and a vector field $w(u) = (-f'(u), 1, f(u) - u \cdot f'(u))$ such that the ruled surface $\phi(u, v) = \gamma(u) + v \cdot w(u)$ is actually developable. We could use (f, g) as coordinates for our stably torsal states, and explore the tangent space of these states as first-order deformations of the solutions but, unfortunately, the existence of a parametrization of the form $z = z(x, y)$ can be assured only locally on a developable surface. Following the motion of the developable surface typically requires the dynamic subdivision of the original domain R in order to have such a parametrization, which leads even more intensely to the problem of tracking boundaries, mergers and splits of subdomains.

6.2 THE BOUNDARY OF DEVELOPABLE SURFACES

There is an alternative approach to study theoretically the dynamics of developable surfaces isometric to a fixed bounded planar domain R : follow the motion of the boundary ∂R in space, and derive from this the developable surface that fills it. This leads to:

Question. Given a piecewise smooth simple closed curve γ in \mathbb{R}^3 , what are the developable surfaces with boundary γ ?

The degeneracy nature of the trivial Monge-Ampère equation makes it fail to have a unique solution for this kind of boundary problem. Indeed, it is easy to find examples where there is more than one solution, as shown in Figure 6.1.

Nevertheless, for problems such as the study of cloth dynamics, it is not necessary for the boundary problem to have a unique solution. It suffices to know that it will always have a finite set of solutions, because this solution set is then discrete, with different solutions separated by a nontrivial jump in any tagging energy, local coordinates, etc. In such case, once one has a developable ruling with a boundary γ_0 at time $t = 0$, the evolution γ_t of the boundary will determine the analytic continuation of the $t = 0$ developable ruling, and identify a unique ruling for every time t . Herein lies the interest of the authors in

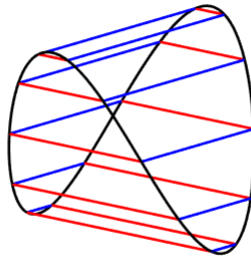


Figure 6.1: A smooth simple closed curve (in black) which is the boundary of two developable surfaces (indicated in red and blue respectively)

Theorem 4 (Theorem). Let γ be a simple closed curve in \mathbb{R}^3 which is piecewise \mathcal{C}^2 , has nonvanishing curvature, its torsion vanishes at finitely many points, and such that only for finitely many pairs $s \neq \tilde{s}$ does the tangent line to γ at s pass through $\gamma(\tilde{s})$. Then, there can be at most finitely many developable surfaces with boundary γ and nonzero mean curvature in its interior.

Let us point out that the preconditions that we impose on γ are generic, i.e. satisfied by a dense open subset of the embeddings of S^1 in \mathbb{R}^3 . The starting idea to prove the theorem is another classical result, analogous to Proposition 2:

Lemma 2. Let S be a torsal surface with boundary γ , and $l \subset S$ a segment with endpoints P, Q in γ . Then the common tangent plane to S along l is tangent to γ at both P, Q , i.e. $\gamma'(P)$ and $\gamma'(Q)$ are both contained in the plane.

The proof consists in pointing out that the normal vector to S stays constant over the segment l , and that γ is tangent to S . Lemma 2 presents developable rulings as arcs of bitangent planes (i.e., tangent to γ at 2 points). Such planes are given by pairs $s \neq \tilde{s}$ whose tangent lines are coplanar:

Proposition 4. Let $\gamma : [0, L] \subset \mathbb{R}^3$ be a simple, closed, arc-parametrized \mathcal{C}^3 curve. The function

$$D : [0, L]^2 \longrightarrow \mathbb{R} \\ (s, \tilde{s}) \longmapsto \det(\gamma(s) - \gamma(\tilde{s}), \gamma'(s), \gamma'(\tilde{s}))$$

is a Morse function at a neighborhood of its zeros (s, \tilde{s}) such that: $s \neq \tilde{s}$, γ has nonzero curvature and torsion at both s, \tilde{s} , and the tangent line to γ in each one does not pass through the other point of the curve.

Proof. It is a straightforward computation. With coordinates (s, \tilde{s}) we have that

$$dD = (\det(\gamma(s) - \gamma(\tilde{s}), \gamma''(s), \gamma'(\tilde{s})), \det(\gamma(s) - \gamma(\tilde{s}), \gamma'(s), \gamma''(\tilde{s})))$$

Let (s, \tilde{s}) be a zero of D with $s \neq \tilde{s}$, which is also a critical point of D . If any of the linear subspaces spanned by $\gamma(s) - \gamma(\tilde{s}), \gamma'(\tilde{s})$ and by $\gamma(s) - \gamma(\tilde{s}), \gamma'(s)$ has dimension less than 2, the tangent line to γ at one of the points $\gamma(s), \gamma(\tilde{s})$ contains the other. When both linear subspaces have dimension 2, the conditions $D(s, \tilde{s}) = 0, dD(s, \tilde{s}) = (0, 0)$ show that γ has the same osculating plane to γ at the points $\gamma(s), \gamma(\tilde{s})$. Because of this, the second differential d^2D of D is

$$\begin{pmatrix} \kappa_s \tau_s \det(\gamma(s) - \gamma(\tilde{s}), B_s, \gamma'(\tilde{s})) & 0 \\ 0 & \kappa_{\tilde{s}} \tau_{\tilde{s}} \det(\gamma(s) - \gamma(\tilde{s}), \gamma'(s), B_{\tilde{s}}) \end{pmatrix}$$

Here κ, τ, B are respectively the curvature, torsion, binormal vector of the Frenet frame, at the point given by their subindex. The determinants in the diagonal of d^2D are nonzero because each consists of a binormal vector and a basis for the osculating plane at the same point of the curve. \square

Proposition 4 has a version for piecewise \mathcal{C}^3 curves, saying just that D is Morse under the additional hypothesis that s, \tilde{s} do not correspond to corner points, at which D has two different definitions. We are now ready for

Proof of Theorem 4. A torsal developable surface S is foliated by segments that can only end at the boundary or at points of vanishing mean curvature. Having ruled out the latter, S is determined by an arc of bitangent planes $B(t)$, with $t \in [a, b]$, such that the curves $s(B(t)), \tilde{s}(B(t))$ formed by the two points of tangency of $B(t)$ cover γ .

Away from the finite set of horizontal and vertical lines in $[0, L]^2$ where one of the values s, \tilde{s} corresponds to a corner point, point with vanishing torsion, or point whose tangent line intersects γ again, the pairs of values $s \neq \tilde{s}$ for which there exists at all a bitangent plane to γ through $\gamma(s), \gamma(\tilde{s})$ lie by Proposition 4 in the zero set of a Morse function D from an open subset of $[0, L]^2 \subset \mathbb{R}^2$ to \mathbb{R} . The function D is proper, therefore it is Morse over a suitably small range of values $(-\varepsilon, \varepsilon)$, which implies that D_0 is a finite union of smooth curves with transverse intersections in $[0, L]^2$. The arc $B(t)$ is determined by its tangency points curve

$$(s(B(t)), \tilde{s}(B(t))) \subset [0, L]^2,$$

which must lie in the union of D_0 and finitely many vertical and horizontal lines, and cover γ , i.e. $\gamma = s(B) \cup \tilde{s}(B)$. There are only finitely many possibilities for that, once we specify a beginning point for the curves $s(B), \tilde{s}(B)$. \square

SEMANTIC CLASSIFICATION OF CLOTH STATES

One of the reasons why robotic manipulation of cloth is a very challenging task is the infinite-dimensional shape-state space of cloth, which makes its state estimation very difficult. In this chapter, we introduce the dGLI *Cloth Coordinates*, a low finite-dimensional representation of the state of a piece of cloth that allows us to efficiently distinguish key topological changes in a folding sequence. Our representation is based on a directional derivative of the well-known *Gauss Linking Integral*. The proposed dGLI Cloth Coordinates are shown to be more accurate in the separation of cloth states than other classic shape distance methods when applied to a rectangular cloth. In order to test this representation (and others), we use the full working simulator developed in the previous chapters to generate a full data-base of folded cloth states.

After reviewing the *state of the art* in Section 7.1, we present preliminary concepts, such as the Gauss Linking Integral, and explain its limitations in a planar setting in Section 7.2. Then, in Section 7.3 we introduce the novel concept of the directional derivative of the *GLI* which is also applicable in a flat configuration. We derive first an expression for the *GLI* of two segments, then we prove that we can perturb the segments slightly to obtain information when they are co-planar and we explain how to apply this to a fully meshed cloth. Finally, in Section 7.4 we apply this new index to a data-base of cloth configurations of a napkin taken from simulated folding sequences and then we test experimentally the index on real images of folded clothes.

7.1 RELATED WORK

As stated in the introduction, textile objects are important and omnipresent in many relevant scenarios of our daily lives, like domestic, healthcare, or industrial contexts. However, as opposed to rigid objects, whose pose is fixed with position and orientation, textile objects are challenging to handle for robots because they change shape under contact and motion, resulting in an infinite-dimensional configuration space. This huge dimensional jump makes existing perception and manipulation methods difficult to apply to textiles. Recent reviews on cloth manipulation, like [101, 122], agree on the need to find a simplified representation that enables more powerful learning methods to solve different problems related to cloth manipulation.

Different representations have been used in the literature of cloth manipulation, e.g. silhouette representations [80] or contours [31], assuming the high-level reasoning on cloth states was given. More modern end-to-end learning approaches use RGB-D images as direct input [58, 72, 78, 103, 109], but only very simple actions can be defined due to the limited state representation. In addition, these methods need large amounts of real or simulated data that are expensive to obtain and label, as no underlying previous knowledge is used to understand the geometric relationship between different states. Therefore, finding a low-dimensional representation of cloth based on low-level features remains an active open problem, while the high-level aspect of understanding cloth deformation is still almost unexplored.

Furthermore, to enable reasoning, abstraction and planning, *rigid object manipulation* applies object recognition methods in order to link objects to actions/affordances [16, 119]. Contacts are estimated among the objects to recognize states such as “on top of”, “inside of” [4]. However, when it comes to cloth manipulation, no work has explored the semantic state identification that could lead to particular actions depending on the task in mind. For simpler deformable objects like a box with an articulated lid, the open configuration clearly allows the action of closing the box or picking something from inside. An equivalent example for cloth would be to recognize a folded corner that needs to be either flattened back if the task is to lay it flat on the table, or picked up if the task is folding. In this context, we wish to classify the configuration space of a piece of cloth in macro-states (or just states), where each state is the set of cloth configurations that can be manipulated in the same way, i.e., that have similar grasping affordances.

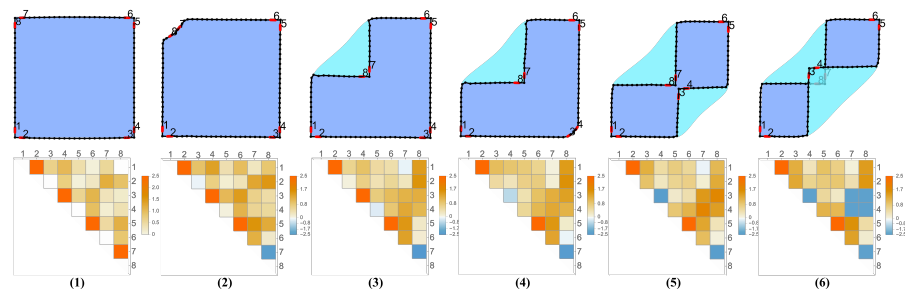


Figure 7.1: Folding sequence of a quadrangular cloth with its associated dGLI *cloth coordinates*, represented as upper triangular matrices. Each matrix element m_{ij} is a geometrical value corresponding to the *dGLI* between the segments i and j highlighted in red in the corresponding folded state. Notice how some values of the matrix change sign when corners are folded or cross each other.

In this work, we present a coordinate representation of the configuration of a rectangular cloth as an upper triangular matrix form (see Figure 7.1). This representation can be computed with a closed-form

formula from low-level features of the cloth, mainly the position of its border, and enables the recognition and classification of high-level states, since we can define a distance between cloth configurations. That allows us to classify different configurations into states that we identify as *different*, meaning that they afford different actions.

Our coordinates are based on a topological index, the *Gauss Linking Integral* (*GLI*). This index has been used in the past for robotic manipulation [57, 91, 107, 108, 124] but can only be applied to 3D curves. For a pair of almost coplanar curves, as the boundary curves of a folded garment, the *GLI* vanishes and it ceases to be informative. In order to consistently consider 2D curves as well as 3D curves, we introduce in this work the concept of the directional derivative of the *GLI*, *dGLI*, applied to a pair of curves. The *dGLI* is symmetric on the curves and it only depends on the relative position between them. We assign the *dGLI Cloth Coordinates* to a state of a garment as follows: first select a subset of edges (it may contain all of them) from a discretization of the boundary of the garment; then fix an ordering on these edges and compute the *dGLI* between any pair of edges in their spatial position of the current state of the garment; this gives a symmetric matrix from which only the upper triangular part is taken in order to avoid redundancies; the *dGLI cloth coordinates* of the state are precisely the entries of this upper triangular matrix (see Figure 7.1).

Our resulting representation can be computed efficiently and is invariant under isometric movements of the garment (i.e. rotations and translations), leaving invariant a distinguished direction normal to a predominant plane in the scene (e.g. a table used as support for the manipulations).

7.2 PRELIMINARIES

Given two non-intersecting 3D-space curves γ_1, γ_2 parameterized by $x(s)$ and $y(t)$, respectively, with $s, t \in I = [0, 1]$, the *Gauss Linking Integral* between them, *GLI* for short, is

$$GLI(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int_I \int_I \frac{(y(t) - x(s)) \cdot (y'(t) \times x'(s))}{|y(t) - x(s)|^3} dt ds$$

or written in a compact way

$$GLI(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int \int \frac{(\gamma_2 - \gamma_1) \cdot [\gamma_2' \times \gamma_1']}{\|\gamma_2 - \gamma_1\|^3}. \quad (7.1)$$

This double integral is invariant under re-parameterizations of the curves. In the case that both curves γ_1 and γ_2 are closed and smooth, their *GLI* is integer valued (due to the chosen normalization factor $\frac{1}{4\pi}$) and it is an invariant of the topology of the embedded curves (see [2]).

Historically, the *GLI* was first introduced by Gauss, presumably related to his works on magnetism (according to [98]) or on astronomy

(according to [33]). Considering the $GLI(\gamma, \gamma)$ of twice the same non-self-intersecting smooth curve γ , then the double integral (taking the domain of integration outside the diagonal of $I \times I$) defines another geometric invariant of the curve, known as *writhe* or *writhing number* of γ . Despite their resemblance, the GLI and the writhe measure different quantities: consider a normal vector field v of length $\epsilon > 0$ on γ , and the curve γ_v of endpoints of the vector field v , which is embedded and in one-to-one smooth correspondence with γ for sufficiently small ϵ . Then the GLI of these two close copies of the same γ differs from the writhe in $GLI(\gamma, \gamma_v) - GLI(\gamma, \gamma)$ equal to the total twist of v . This result is known as the Călugăreanu-White-Fuller theorem (see [90]). However, both indexes, GLI and writhe, are non-informative for planar curves, since they both vanish.

The GLI has been used for many applications after a version of the above formula for polygonal curves appeared in the context of DNA protein structures [68], with additional efficient formulations given in [66], from which we have chosen the following: given a discretization of the curves into N and M segments, that is, $\gamma_1 = \{\gamma_{P_i P_{i+1}}, i = 1, \dots, N\}$ and $\gamma_2 = \{\gamma_{Q_i Q_{i+1}}, i = 1, \dots, M\}$, where each segment is parameterized as $\gamma_{AB}(s) = A + s\vec{AB}$ for $s \in [0, 1]$, then the GLI between both curves is

$$GLI(\gamma_1, \gamma_2) = \frac{1}{4\pi} \sum_{i=1}^N \sum_{j=1}^M GLI(\gamma_{P_i P_{i+1}}, \gamma_{Q_j Q_{j+1}}) \quad (7.2)$$

where the GLI between a pair of segments γ_{AB} and γ_{CD} is computed as

$$GLI(\gamma_{AB}, \gamma_{CD}) = \arcsin(\vec{n}_A \vec{n}_D) + \arcsin(\vec{n}_D \vec{n}_B) + \arcsin(\vec{n}_B \vec{n}_C) + \arcsin(\vec{n}_C \vec{n}_A) \quad (7.3)$$

with

$$\vec{n}_A = \|\vec{AC} \times \vec{AD}\|, \vec{n}_B = \|\vec{BD} \times \vec{BC}\|, \\ \vec{n}_C = \|\vec{BC} \times \vec{AC}\|, \text{ and } \vec{n}_D = \|\vec{AD} \times \vec{BD}\|.$$

The above discrete formula was used by Ho [54] to identify and synthesize animated characters in intertwined positions [53, 54]. In the context of robotics, the GLI has been applied to representative curves of the workspace to guide path planning through holes [57, 124], for guiding caging grasps in [91, 107, 108], and for planning humanoid robot motions, using the GLI to guide reinforcement learning [123]. In this work, for the first time, we develop a further analysis of the notion to be able to apply it to planar or almost planar curves, which opens the door to a wider spectrum of applications.

7.3 DERIVATION OF THE CLOTH COORDINATES

As we have mentioned above, the *GLI* of two coplanar curves vanishes; so for many configurations of robotic interest — configurations where the cloth is nearly flat on a table, ready to be folded or already folded—the *GLI* does not provide much information. Our aim in this section is therefore to develop a similar index able to distinguish planar configurations. We shall see that a natural index to consider is in fact a directional derivative of the *GLI*, but to arrive at such an index we must first make a few observations about the *GLI* when applied to pairs of segments as in Equation (7.3); since the class of curves we will be working with computationally are piece-wise linear.

7.3.1 *GLI of two segments*

Since two segments AB and CD are uniquely defined by the four endpoints $A, B, C, D \in \mathbb{R}^3$, the *GLI* of two segments computed in Equation (7.3) can be viewed as a function from $(\mathbb{R}^3)^4 \equiv \mathbb{R}^{12}$ to \mathbb{R} . To emphasize that from now on we are considering segments we define $\mathcal{G} : \mathbb{R}^{12} \rightarrow \mathbb{R}$ as

$$\mathcal{G}(A, B, C, D) = GLI(\gamma_{AB}, \gamma_{CD}) = \frac{1}{4\pi} \int \int \frac{(\gamma_{CD} - \gamma_{AB}) \cdot [\gamma'_{CD} \times \gamma'_{AB}]}{\|\gamma_{CD} - \gamma_{AB}\|^3}. \tag{7.4}$$

Note that technically \mathcal{G} is not defined in the whole of \mathbb{R}^{12} , since it is not defined when γ_{AB} and γ_{CD} intersect. Next, we will find a reformulation for \mathcal{G} wherever it is defined.

Notice that the numerator in the integral expression of the *GLI* of Equation (7.4) is constant (for any t and s) and equals

$$\begin{aligned} (\gamma_{CD} - \gamma_{AB}) \cdot [\gamma'_{CD} \times \gamma'_{AB}] &= \\ &= (\vec{AC} + t\vec{CD} - s\vec{AB}) \cdot [\vec{CD} \times \vec{AB}] = \\ &= \vec{AC} \cdot [\vec{CD} \times \vec{AB}] = \\ &= \vec{AC} \cdot [(\vec{CA} + \vec{AD}) \times \vec{AB}] = \\ &= \vec{AC} \cdot [\vec{AD} \times \vec{AB}] = \vec{AB} \cdot [\vec{AC} \times \vec{AD}] = \\ &= \det(\vec{AB}, \vec{AC}, \vec{AD}) \end{aligned}$$

the signed volume of the tetrahedron $ABCD$ multiplied by 6. By writing

$$\mathcal{V}(A, B, C, D) = \det(\vec{AB}, \vec{AC}, \vec{AD})$$

and

$$\mathcal{I}(A, B, C, D) = \frac{1}{4\pi} \int \int \frac{1}{\|\gamma_{CD} - \gamma_{AB}\|^3},$$

we have

$$\mathcal{G} = \mathcal{V} \cdot \mathcal{I}. \tag{7.5}$$

Now, it is clear that the GLI vanishes when the segments are coplanar because $\mathcal{V} = 0$. Being \mathcal{G} the product of two differentiable functions and hence differentiable, it makes sense to consider its directional derivative.

7.3.2 Directional derivative of the GLI

In this section, we discuss how to perturb \mathcal{G} in order to make it informative in planar settings.

Definition 13 (Directional derivative of \mathcal{G}). Let $v_A, v_B, v_C, v_D \in \mathbb{R}^3$ be arbitrary directions and AB, CD two non-intersecting segments. The directional derivative of \mathcal{G} at the point (A, B, C, D) in the direction of $v = (v_A, v_B, v_C, v_D)$ is defined as the limit

$$\partial_v \mathcal{G}(A, B, C, D) = \lim_{\varepsilon \rightarrow 0} \frac{\mathcal{G}((A, B, C, D) + \varepsilon(v_A, v_B, v_C, v_D)) - \mathcal{G}(A, B, C, D)}{\varepsilon}.$$

Remark 7.3.1. Notice that this limit always exists since we have shown that \mathcal{G} is a differentiable function with respect to A, B, C, D . Moreover, $\partial_v \mathcal{G}$ can be equivalently written as

$$\lim_{\varepsilon \rightarrow 0} \frac{GLI(\gamma_{A^*B^*}, \gamma_{C^*D^*}) - GLI(\gamma_{AB}, \gamma_{CD})}{\varepsilon},$$

where $A^* = A + \varepsilon v_A$, $B^* = B + \varepsilon v_B$, $C^* = C + \varepsilon v_C$, $D^* = D + \varepsilon v_D$ and ε is sufficiently small so that A^*B^* and C^*D^* do not intersect. Also, from Equation (7.5) and by the product rule

$$\partial_v \mathcal{G} = \partial_v(\mathcal{V})\mathcal{I} + \mathcal{V}\partial_v(\mathcal{I}),$$

hence $\partial_v \mathcal{G} = \partial_v(\mathcal{V})\mathcal{I}$ when the segments γ_{AB} and γ_{CD} are coplanar.

Properties of $\partial_v \mathcal{G}$. By definition, $\partial_v \mathcal{G}$ is invariant under translations, rotations and scalings if v is rotated and scaled accordingly. These properties are a consequence of the fact that the GLI is invariant under such transformations. However, for a fixed choice of v , $\partial_v \mathcal{G}$ will not be invariant under rotations or scalings in general. For instance, no fixed choice of v can make $\partial_v \mathcal{G}$ invariant under scalings, since scaling by a factor of λ scales \mathcal{I} by $\frac{1}{\lambda^3}$ and $\nabla \mathcal{V}$ by λ^2 , and similarly scales $\nabla \mathcal{I}$ by $\frac{1}{\lambda^4}$ and \mathcal{V} by λ^3 , resulting in scaling $\nabla \mathcal{G}$ by $\frac{1}{\lambda}$. Depending on what this index is used for, one must keep this scaling relationship in mind or alternatively choose v depending on the segments. However, the distance we will use to compare different cloth states only depends on the correlation of values of the coordinates more than on the magnitude. That is why we can ignore the scaling factor that would appear when comparing two garments of different sizes (e.g. because of different meshings).

Choice of v . This is highly task-specific, but given the nature of our task –classifying planar cloth configurations based on affordances– it

is natural to perturb the vertices in the direction normal to the table plane. Making such a choice of v does in fact make $\partial_v \mathcal{G}$ invariant under rotations and translations of the XY plane, which is desirable for our purposes since such movements of a cloth configuration have the same affordances. Furthermore, to conserve the symmetry $\partial_v \mathcal{G}(A, B, C, D) = \partial_v \mathcal{G}(C, D, A, B)$, we must perturb A and C by the same amount and direction, and the same is the case with B and D . Finally, it is easy to see that in fact perturbing A and C both by the same amount normal to the plane yields the same result as perturbing B and D by the same amount in the opposite direction, so it really only makes sense to perturb A and C , or B and D , but not both pairs, and doing one or the other is equivalent except for a sign change. In summary, the most natural choice of v in our case is

$$v := (\vec{0}, e_3, \vec{0}, e_3) \quad (7.6)$$

(or $v = (e_3, \vec{0}, e_3, \vec{0})$, which is equivalent except for a sign change) where $e_3 = (0, 0, 1)$ is the normal to the plane of the table on which the cloth lies.

7.3.3 Practical computation of $dGLI$

We summarize the discussion of the previous section in the following definition.

Definition 14 (dGLI of two segments). Given two non-intersecting segments γ_{AB} and γ_{CD} , we define

$$dGLI(\gamma_{AB}, \gamma_{CD}) := \lim_{\varepsilon \rightarrow 0} \frac{GLI(\gamma_{AB^*}, \gamma_{CD^*}) - GLI(\gamma_{AB}, \gamma_{CD})}{\varepsilon}, \quad (7.7)$$

where $B^* = B + \varepsilon e_3$, $D^* = D + \varepsilon e_3$, $e_3 = (0, 0, 1)$ and each GLI function can be computed using Equation (7.3). This index is invariant under rotations and translations of the XY plane and moreover $dGLI(\gamma_{CD}, \gamma_{AB}) = dGLI(\gamma_{AB}, \gamma_{CD})$.

Remark 7.3.2. We have analyzed numerically the limit defined in Equation (7.7), and have found that it is sufficiently stable to be computed as

$$dGLI(\gamma_{AB}, \gamma_{CD}) \cong \frac{GLI(\gamma_{AB^*}, \gamma_{CD^*}) - GLI(\gamma_{AB}, \gamma_{CD})}{\varepsilon}$$

for a sufficiently small ε . Since we work with double precision floats (which allow a precision of around 14-15 decimals), it is known (see [35]) that when approximating derivatives numerically, one obtains better results when choosing perturbations that only affect 7 or 8 decimal places, for instance, $\varepsilon \approx 10^{-8}$. This is the value taken in our experiments.

Remark 7.3.3. In practical implementations, we may well be computing the $dGLI$ between segments γ_{AB} and γ_{CD} that are very close to intersecting (but not intersecting since the cloth has thickness), and then $dGLI(\gamma_{AB}, \gamma_{CD})$ becomes very large. As having such big quantities can dominate values of metrics and distances in a non-representative way, in practice we set a maximum value to the $dGLI$ once it surpasses a fixed threshold.

7.3.4 Definition of the $dGLI$ Cloth Coordinates

Since we are now equipped with a geometric index for pairs of segments, we are ready to introduce our cloth coordinates, which will parametrize the shape-state space of a piece of cloth. We assign the $dGLI$ Cloth Coordinates to a cloth configuration \mathcal{C} of a garment as follows:

Definition 15 ($dGLI$ of a cloth surface \mathcal{C}). Given a discretization of the boundary of the garment surface \mathcal{C} as a polygonal curve, select an ordered subset of edges of it $\mathcal{S}_{\mathcal{C}} = \{S_i : i = 1, \dots, m\}$. Then, the $dGLI$ Cloth Coordinates of configuration \mathcal{C} is the upper triangular matrix

$$dGLI(\mathcal{C}) = (dGLI(S_i, S_j))_{S_i, S_j \in \mathcal{S}_{\mathcal{C}}, i > j}. \quad (7.8)$$

To get an intuitive sense of what these upper triangular matrices look like for some cloth configurations, see the examples in Figure 7.1. If we were interested in a general direction v , we would take the $dGLI_v$ Cloth Coordinates

$$dGLI_v(\mathcal{C}) = (\partial_v \mathcal{G}(S_i, S_j))_{S_i, S_j \in \mathcal{S}_{\mathcal{C}}, i > j}.$$

Note that the full matrix when taking all the edges of the discretization is the equivalent rationale to computing the GLI of a polynomial curve used in [54, 68], where the GLI of all pairs of segments of the curves were first assembled in what was there called the GLI matrix [54].



Figure 7.2: The subset of chosen segments is marked in red.

Choice of edges. The subset $\mathcal{S}_{\mathcal{C}}$ of edges chosen in the discretization depends on the task one wants to carry out; tasks that demand finer distinctions between configurations of a similar class would require a greater subset of segments. For our task of classifying the configurations into relatively broad classes, we found experimentally that a

good choice of segments is given by the eight segments adjacent to the corner segments, marked red in Figure 7.2. This is a small subset that is nevertheless enough to provide an accurate affordance-based classification of near-to-flat configurations.

The upper triangular matrix in Equation (7.8), sorted as a vector, is a coordinate system that reduces the infinite dimensionality of the configuration space of cloth states to a mere $\frac{m(m-1)}{2}$ dimensional space. In our case $m = 8$, so this comes out to 28 dimensions. This reduction in dimensionality is well-suited and informative enough for practical purposes, as the validation results in the next section will show.

7.4 RESULTS

In this section, we study the ability of the cloth coordinates previously defined to tell apart different cloth states. First, we analyze 3 folding sequences (see Figure 7.3). We will show that our representation is capable of distinguishing different relevant cloth configurations (e.g. one folded corner vs two folded corners). Then, we will apply our method to a full data-base with 12 cloth classes (shown in Figure 7.4), and we will compare it to 4 alternative representations, proving that ours is more capable of differentiating between cloth states. All data in this section was simulated using the inextensible cloth model described in Chapter 3. All simulations are performed with a square cloth of dimensions $1\text{m} \times 1\text{m}$ and each computational mesh has a resolution of 400 nodes. A small amount of shearing (see Section 2.4.6) is allowed in order to facilitate the more complicated foldings. Finally, we will apply a simple classification method using our representation to real images of folded cloth states.

In order to compare different cloth configurations, once they are represented with our cloth coordinates $\text{dGLI}(\mathcal{C}) \in \mathbb{R}^{28}$, it is important to use a proper distance. Due to the scaling factor that we analyzed in the previous section, the most suitable distance was the *Spearman's distance*. Given two vectors x, y it is defined as

$$d(x, y) = 1 - \rho(\mathbf{R}(x), \mathbf{R}(y)), \quad (7.9)$$

where ρ is the Pearson correlation coefficient, and $\mathbf{R}(x)$ is the rank variable of x (i.e. ordering the coordinates of x from lowest to greatest and then assigning to each coordinate its position in the ranking). This distance assesses how well the relationship between two vectors x, y can be described using any monotonic function (not only a line). We found this distance to be more sensitive to changes of the cloth configuration than the euclidean distance. This may be due to the fact that this distance focuses on the ranking order between coordinates (with sign) rather than comparing their magnitudes, which is most relevant in our representation. Note that the Spearman's distance is

bounded with values between 0 and 2 and ignores scaling factors between different clothes.

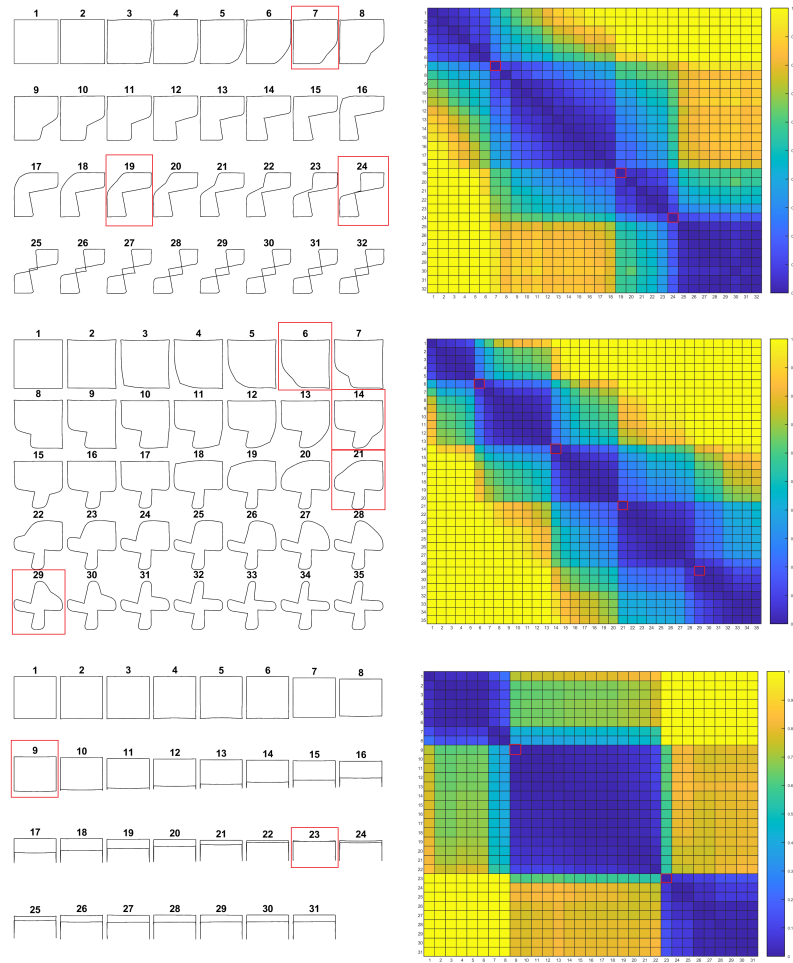


Figure 7.3: Study of the index during 3 folding sequences. In the left column, we show a representation of the cloth frames, and in the right column the confusion matrix of all of them. In red we highlight the clear class changes that can be identified.

7.4.1 Analysis of folding sequences

The first test compares different cloth states inside a folding sequence. Given the motion of the cloth $\{\mathcal{C}_1, \dots, \mathcal{C}_m\}$, where m is the number of discrete frames and \mathcal{C}_i is the state of the cloth at t_i , we compute the confusion matrix $\mathcal{M}_{ij} = d(d\text{GLI}(\mathcal{C}_i), d\text{GLI}(\mathcal{C}_j))$. The 3 folding sequences, shown at the left side of Figure 7.3 are: folding two opposite corners, folding 4 corners inwards, and folding the cloth in half. The results can be seen on the right side of the figure. Notice how our representation detects changes during the sequence that are topologically meaningful. For example, in Seq. 1, folding two opposite corners, at frame 7, there is a topological change, since a corner changes the orientation from flat

to folded, even before it is released. This can be seen in the confusion matrix (first two blue squares). This is also clear in Seq. 2, where four corners are folded inwards. Moreover, our method also detects when edges of the cloth cross (Seq. 1, frame 24, Seq. 3, frame 23). These changes are also meaningful from the manipulation point of view, as they afford different possible graspings or actions.

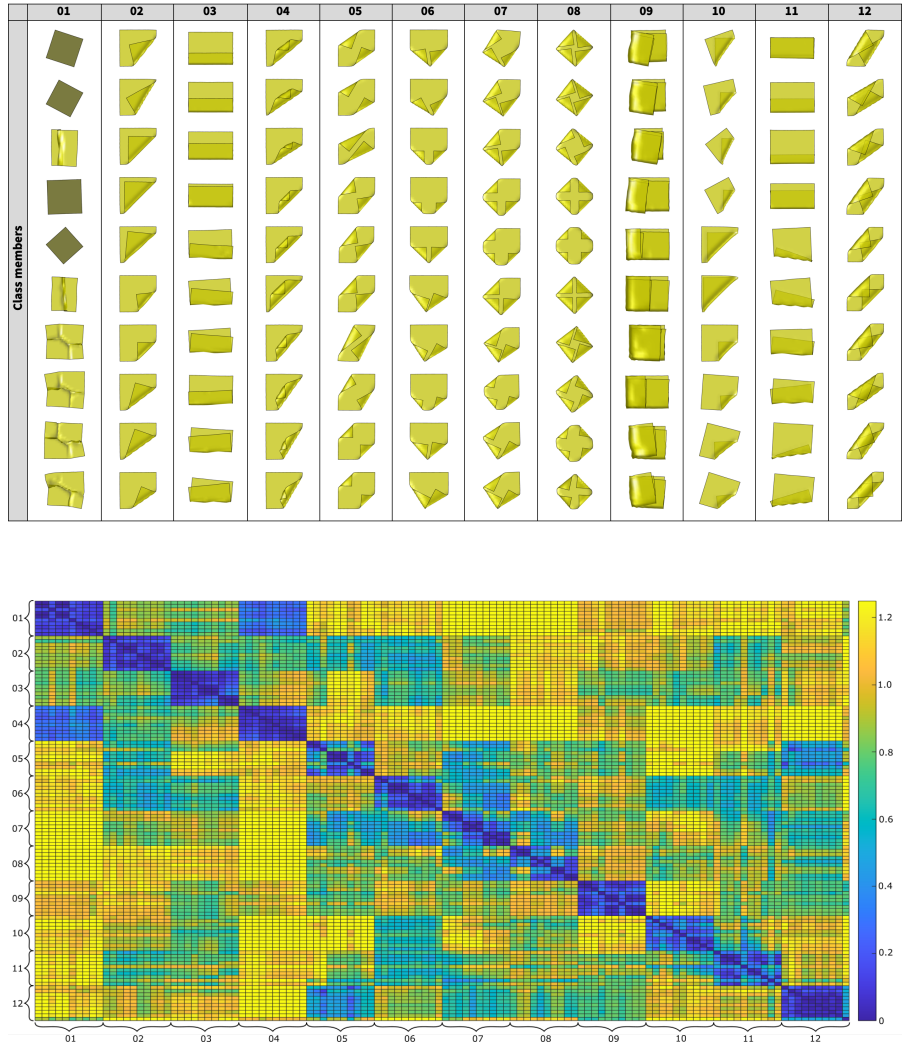


Figure 7.4: Confusion matrix that computes all the distances between the states shown in the top table.

7.4.2 Confusion matrix of the full data-base

We now analyze a complete data-base consisting of 120 examples classified in 12 different classes of states, shown in Figure 7.4. Most of them are self-explanatory. Note that in class 10, the upper left corner is folded under the cloth (likewise for class 11).

Each class has 10 samples corresponding to the final state of the cloth during a folding sequence simulation. We manually identified

samples that we considered to belong to the same state. We want to emphasize that once we fix an ordering of the corners, our method distinguishes, for example, between different folded corners and this does not contradict the rotational invariance previously shown.

Again we compute the confusion matrix $\mathcal{M}_{ij} = d(\text{dGLI}(\mathcal{C}_i), \text{dGLI}(\mathcal{C}_j))$ where \mathcal{C}_k is the k th example of the database. We order the samples, so that the samples from the same classes are consecutive. This way, the plot is more easily interpretable. In Figure 7.4 we can see how the classes group without confusion: i.e. the distance between members of a class tends to be smaller (color blue) than the distance to examples outside the class (color yellow).

The confusion matrix shows us interesting insights about our representation. For instance, we can see that the two classes 01 and 04 are relatively closer than others. That is because the relative position of all edges is indeed the same in these classes, resulting in a smaller distance in our representation. The same phenomenon can be seen between classes 05 and 12 in some cases, as they are indeed classes with similarities (in 05 the two corners do not cross, whereas in 12 they do). However, classes 03 and 11, which differ on whether or not the folding makes one side of the cloth hide its opposite, are perfectly separated. The borderline cases, that is, the fourth element in class 03 and the first element in class 11 are very similar, but our method distinguishes them because of the relative geometric position between edges (i.e. in these two cases, they are flipped). A similar thing occurs between classes 02 and 10. It is also worth mentioning that some classes that we have labeled as the same class have clear sub-classes shown in the confusion matrix. That is the case for classes 05, 07 and 08. These are folded corners with different orientations. It is possible, using our representation, to induce a partition of the space in order to separate this type of class into two.

7.4.3 Comparison with other shape representations

In this subsection, we perform a more quantitative comparison of our state representation with other competing methods in representing shapes. To evaluate a shape representation, we use the standard Davies-Bouldin index to measure cluster separation [27]:

$$DB = \frac{1}{n} \sum_{i=1}^n \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \quad (7.10)$$

where n is the number of classes (e.g. in the data-base it is 12), c_i is the centroid of class i (the average of the coordinates of members of class i), σ_i is a dispersion measure computed as the average distance of all elements in class i to the centroid c_i and $d(c_i, c_j)$ is the distance between centroids c_i and c_j . With the given classification in Figure 7.4

Table 7.1: Comparison between different shape representations*

	DATABASE	SEQUENCE I	SEQUENCE II	SEQUENCE III
dGLI	0.73	0.27	0.18	0.21
Edges	1.60	0.68	0.77	0.51
Corners	2.49	0.98	1.61	3.14
Fréchet	0.99	0.69	0.76	0.48
Hausdorff	1.45	0.71	0.84	0.49

*Each number is the Davies-Boulding index introduced in Equation (7.10), that measures cluster separation quality. A smaller value means a better separation. We mark in bold the smallest values in each column.

taken as ground truth, we want a representation that gives a small dispersion inside a class and a high distance between the classes, resulting in a low index. The representation and distance with the smallest DB is considered the one that better separates these clusters, and therefore, the best representation to identify different cloth states.

First, we use two simple cloth representations using similar low-level features like the ones we used:

- (i) **Edges:** for a given mesh we select the edges shown in Figure 7.2 and compute their pairwise minimal distance, i.e. for each pair of segments S_i and S_j in \mathcal{S}_C

$$v_{ij} = \min_{x \in S_i, y \in S_j} \|x - y\|$$

This results in a representation vector $v = (v_{ij})_{j>i}$ of length 28 just like those of the dGLI coordinates (notice that unlike the dGLI, the coordinates of this vector are always non-negative). We use the Spearman's distance to compare two different samples. This representation is invariant under rigid motions of the plane.

- (ii) **Corners:** for a given mesh we compute the pairwise distance between its 4 corners p_k , i.e. $v_{ij} = \|p_i - p_j\|$. These are 6 non-negative numbers $v = (v_{ij})_{j>i}$ that can be computed for any rectangular cloth, they are invariant by rigid motions and they give a trivial representation of the state of the cloth. We also use Spearman's distance to compare different samples.

In addition, we compare us with two classic methods to measure the distance between curves and polygons [5, 114], taking the full discrete boundary curve of the cloth as the state representation:

- (iii) **Fréchet**: to compare two different samples we compute the (discrete) Fréchet distance [3] between the curves. Let C_1 and C_2 be two continuous spatial curves, and $\alpha, \beta : [0, 1] \rightarrow \mathbb{R}^3$ parametrizations of them. Then, the Fréchet distance between the curves is defined as

$$d_F(C_1, C_2) = \inf_{\alpha, \beta} \sup_{t \in [0, 1]} \|\alpha(t) - \beta(t)\|,$$

where the infimum runs over all possible reparametrizations of the curves. The previous formula can be efficiently computed when the curves are polygons [3]. This is a distance that takes into account the location and ordering of the points along the curves. Since this distance is not invariant by rigid motions, special care must be taken to center and align the samples before comparing them. In order to do so we center the curves at the origin and perform a rigid alignment by computing the rotation that minimizes the distance between the curves' points.

- (iv) **Hausdorff**: to compare two different samples we compute the (discrete) Hausdorff distance between the boundary curves [51]. This distance can be defined as:

$$d_H(C_1, C_2) = \max \left\{ \sup_{x \in C_1} d(x, C_2), \sup_{y \in C_2} d(C_1, y) \right\},$$

where as before C_1 and C_2 are the (piece-wise linear) curves, and $d(\cdot)$ is the euclidean distance. Informally, two curves are close in the Hausdorff sense if every point of either curve is close to some other point of the other one. This distance disregards the fact that the sets it is comparing are curves and therefore is expected to be less sensitive than the Fréchet distance. As before, since this distance is not invariant by rigid motions, we center and align the samples before comparing them.

In Table 7.1, we display the computation of the DB index for our dGLI coordinates and the four discussed methods, using as testing scenarios the full data-base and the 3 folding sequences presented before (taking as classes those depicted in Figure 7.3). As seen in the table, our method results in the lowest overall DB in all 4 scenarios, indicating that our method is the one among those studied that best represents the different folded states of the cloths.

7.4.4 Real images classification

Once checked that our method was able to represent folded states of cloth accurately, we implemented a simple classifier of real folded cloth states to highlight its applicability. In order to do so, synthetic representative elements of each class in the data-base shown in Figure

7.4 are chosen, and we estimate the class of a new real unclassified sample by choosing its closest representative, using the Spearman's distance. As we can see in the confusion matrix in Figure 7.4, some classes have a bigger dispersion in distance because of the variation in orientations of the corners. For these classes, we have chosen 3 different representatives, corresponding to the three different subgroups that can be clearly seen in the confusion matrix. We show the silhouette of the representatives chosen for each class in Figure 7.5.

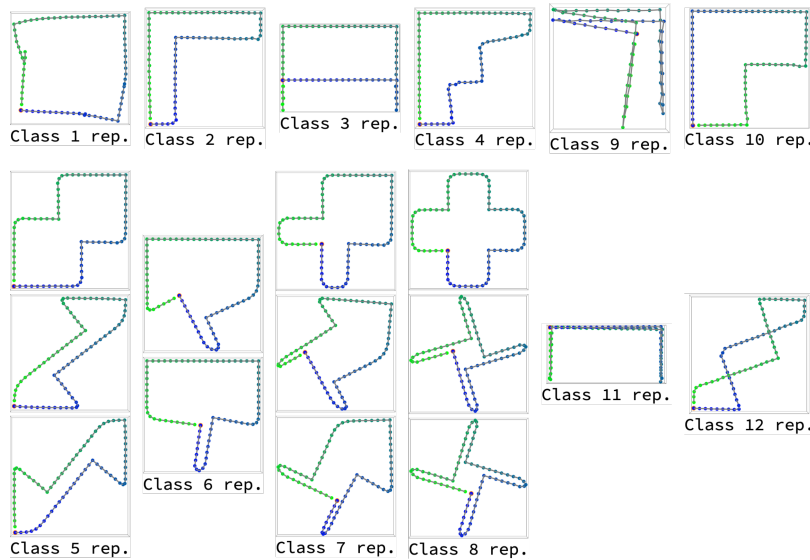


Figure 7.5: Synthetic representatives chosen for each class. When only one is chosen, it is the closest to the centroid of the class. When a class has more sparsity, additional representatives are chosen to represent the subgroups in the class.

The real images are taken from a zenithal position at 52 cm from the table using a Microsoft Azure Kinect DK 3D camera. A single napkin is used with 3 colored stickers attached along each edge, close to a corner and on both sides. We first use color segmentation to detect the center of each sticker and get the corresponding 3D point from the depth image. Once all markers are detected, with our combinations of colors on each edge, we can identify each individual corner of the cloth (there are four stickers of the same color around each corner), and its corresponding edge positions, following the same edge selection as in Figure 7.2. The obtained size of the observed edges is more than 400 times larger than the edges of the samples of the simulated data-base, but thanks to the Spearman's distance used, this does not affect the distance values when comparing shapes of different sizes. The table in Figure 7.6 shows the results of the classification. The only miss-classification is the last image of class 04. However, note that this is a very extreme case where the cloth is almost flat, and therefore, it is confused with the flat class 01. This is a reasonable mistake, as this cloth can be considered flat enough.

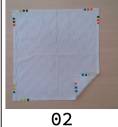



















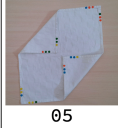






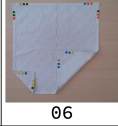







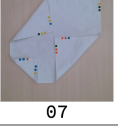

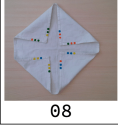
GT	Images of cloth with predicted class						
02							
	02	02	02	02	02	02	02
03							
	03	03	03	03	03	03	
04							
	04	04	04	04	04	04	01
05							
	05	05	05	05	05	05	05
06							
	06	06	06	06			
07							
	07	07	07	07	07	07	
08							
	08						

Figure 7.6: Results of the real image classification using the data-base presented in Figure 7.4 as reference. The first column shows the ground truth class of the images, and at the bottom of every image the classified class.

Notice that we can only perceive those textiles with all the stickers visible, therefore, classes with hidden edges, like for instance classes from 09 to 12 where the folding is under the cloth are not present in the real set of samples. However, the classifier still used all 12 classes of the simulated data-base. This shows that the missing classes don't create confusion in the classification process.

7.4.5 Discussion of the results

Our results show we can identify the topologically relevant changes along folding sequences, just by using a distance between the dGLI Cloth Coordinates as vectors. This same representation has allowed us to distinguish 12 different classes generated with the inextensible cloth simulator. These are promising results towards a low-dimensional

representation that can be used for high-level identification of states, but is still linked to low-level features such as the location of the border, which are fundamental to executing physical actions. Moreover, the fact that using our representation we can successfully classify real configurations of cloth from synthetically generated samples, as seen in Figure 7.6, shows great promise for applications in planning for cloth manipulation.

One limitation of our approach is that it assumes that the cloth boundaries are known. However, perception algorithms are starting to show solutions to overcome this problem. For instance, in [95] a method is developed to detect parts of clothes suitable for grasping. More recently, the deep-learning approach presented in [94] can identify corners and edges, but does not yet identify the full border. Our group is working on deep-learning methods to *hallucinate* the full border given an image of the cloth. Despite this limitation, our approach as it stands can be fully used in simulation, for instance, to automatically label cloth states.

CONCLUSIONS

Part I: modeling cloth

In this first part of the thesis, we presented a continuous model for the mechanical simulation of inextensible textiles, i.e. isometric motion of surfaces subject to the postulates of Classical Mechanics. Continuity is important because it guarantees that the model is very stable under different resolutions, allowing coarse meshes to be used (e.g. in robotic applications) soundly. On the other hand, inextensibility as implemented permits the conservation of area within an error of less than 1% without exhibiting *locking* and moreover, simplifies the empirical validation of the model, reducing drastically the number of parameters to be fitted since stretching and shearing forces are no longer needed. We derived this novel model of inextensibility as a set of continuous constraints (in fact PDEs) and then applied a non-trivial Finite Element discretization to these conditions. The model was shown to be locking-free, independent with respect to different meshed topologies, stable when the mesh is refined and capable of dealing with non-trivial cloth topologies (a tank-top). These properties make our model ideal for planning and control of cloth, manipulated by robots.

Then, we delved into the problem of modeling collisions, including the response. We incorporate contacts with an object (using Signorini's conditions), self-collisions and Coloumb friction into the equations of motion. Our collision model considers all constraints (inextensibility and contacts) and friction forces at the same time without any decoupling. We developed a novel numerical discretization of contact and friction forces that can be seen as a natural extension of the *fast projection algorithm* presented in [44] in order to include inextensibility, contacts and friction in a single pass. This discretization led naturally to a sequence of quadratic problems with inequality constraints. We presented a novel *active-set* method tailored to our problem which takes into account past active constraints to accelerate the resolution of unresolved contacts. The main advantage of this new algorithm with respect to standard *active-set* methods is its ability to start from any point and not necessarily from a feasible one. To close the chapter we showed that our model of friction is effective in static and dynamic settings, that collisions with sharp objects can be easily included and that complex folding sequences of cloth with non-trivial topologies (a pair of shorts) can be simulated.

Thirdly, we studied how faithfully our inextensible model is able to reproduce recordings of real textiles under different circumstances. First, we recorded the motion of four (size A3) textiles with a depth camera. The fabrics were shaken by a robotic WAM arm using two sets of different amplitudes and frequencies. The results are accurate: even with a coarse 9×9 mesh, the mean error stays under 5mm for different textiles and both fast and slow motions. Besides, the calibration depends only on two physical parameters. Moreover, in a novel, and remarkably simple way, those two parameters take into account even the aerodynamic perturbations to motion, which for cloth are harder to model than for rigid, even vibrating bodies.

We then performed a second more exhaustive set of recordings than before by adding a new twisting movement and a larger cloth size (DIN A2). As before we estimated the optimal values of the two physical parameters and the model achieved very low mean errors (less than 1 cm) and standard deviations, even for the A2 textiles and fast motions. Finally, we found a predictive formula in order to obtain *a priori* estimates for the values of the parameters α and δ of the model. This formula depends on the density of the textile, its size and more importantly its speed. The formula was found to be coherent between independent recordings of the textiles and moreover, it produced parameter values that in turn still give rise to very low absolute errors.

With the third round of experiments, we validated two different but related aspects of the collision model: its ability to simulate properly friction and to model the dynamics of fast and strong hits with a long stick. We were able to model the most challenging scenario in this thesis: the DIN A2 cloths were held by its two upper corners and then hit repeatedly at different locations and varied intensities with a stick. The average errors are still of the order of 1 cm and we were able to properly simulate the hits, appearing the biggest errors not during the hits but just after because of aerodynamic effects. The simulations are two times faster than real-time (for the hitting scenario with a 7×9 mesh), being our novel active-set solver three times faster than a standard interior-point method using the same mesh resolution.

Part II: reconstructing surfaces

In this second part, we studied the problem of reconstructing surfaces from point-clouds. We presented an algorithm that successfully reconstructs a surface S by finding a Morse cellular decomposition from a cloud of sampled points. The algorithm can be applied to surfaces in \mathbb{R}^N for any ambient dimension N , with or without boundary. We reconstructed with our method three different surfaces (one without boundary –a *torus*– and two with it –a *vest* and a pair of *pants*–) which posed various challenges. For instance, the *torus* was really slim and its

embedding described a $(2,3)$ -toric knot which caused far away parts of the surface (as measured by geodesic distance) to be really near each other in euclidean space. On the other hand, the *pants* were obtained as a 3D-scan of a real pair of jeans and thus its point-cloud presented wrinkles, noise and an irregular density distribution of points. For all surfaces, a global piecewise decomposition was found, with a very small number of pieces in the examples examined. From the cellular decomposition, the topology of the surface can be deduced immediately. Moreover, the algorithm is robust: it always produces a surface, and it captures the topological features of the sampled surface with a size greater than the average distance between sample points.

Part III: classifying cloth states

In this last part, we studied theoretical properties of developable surfaces, i.e. smooth surfaces with Gaussian curvature 0: since the original state of a piece of cloth is flat, the set of possible states under the inextensible assumption is the set of developable surfaces isometric to a fixed one. We proved that a generic simple, closed, piecewise regular curve in space can be the boundary of only finitely many developable surfaces. For problems such as the study of cloth dynamics, it is not necessary for the boundary problem to have a unique solution. It suffices to know that it will always have a finite set of solutions, because this solution set is then discrete, with different solutions separated by a nontrivial jump in the coordinates of the surface. This implies that during a continuous cloth motion, the position of a garment is determined by the location of its boundary and its initial state. The relevance of this result justifies theoretically our choice of defining the dGLI cloth coordinates based only on the position of the cloth boundary.

In the second chapter of this part, we introduced the dGLI *Cloth Coordinates*, a representation for cloth configurations based on a directional derivative of the Gauss Linking Integral that vastly reduces the dimensionality of the configuration space. This reduced representation nevertheless preserves enough information about the configurations to be able to distinguish them based on their grasping affordances (i.e. how they can be manipulated) using Spearman's distance. These coordinates bridge the gap between low-level features of different cloth configurations, such as the location of corners and edges, to high-level semantic identification of cloth states (e.g. how they are folded). Moreover, the fact that using our representation we could successfully classify real configurations of cloth from synthetically generated samples, as seen in Figure 7.6, shows great promise for applications in planning for cloth manipulation. Furthermore, our representation allows for different choices of the subset of edges chosen to compute the coordinates, so that one can fine-tune the representation to the

specific task at hand to boost results. Lastly, since our method is not learning-based, it does not require any training data, it is completely explainable, and it is robust against possible configurations that are not in the training set.

8.1 FURTHER WORK

Currently, we are working on overcoming occlusions when recording the motion of the garments. When parts of the cloth are not visible some of the time, we need to apply an identification scheme to our laboratory recordings to identify with accuracy these regions when they appear once again in the limelight. This re-identification scheme, which must have an error smaller than that of our model (i.e. in the mm range), is still under development. This would allow the recording of more complex garments, such as t-shirts. Moreover, the developed cloth model is expected to be included in a *Virtual Reality* environment so that data collection can be made faster and easier.

On the more theoretical side of things, the authors hope to carry out the program outlined in Chapter 6: to subdivide a developable surface S in patches according to the sign of its mean curvature, and follow its motion in a dynamical system by tracking the boundaries of the patches. This approach is promising because it works with a 1-dimensional set of space coordinates that satisfy few restrictions, rather than with a 2-dimensional set of space coordinates that are heavily restricted because of the assumption of isometry.

Future work also concerns an in-depth analysis of the configuration space defined by our dGLI coordinates. In particular, we would like to identify a partition of the space that corresponds to a partition of configurations by grasping affordance, which states are neighbors in this partition, and what the shortest paths from one state to another are. We look forward to carrying out this study analytically as well as through learning methods, which we believe will give better results when the data is enriched and given structure through our representation.

Finally, we intend to test the reconstruction algorithm with more *real* point-clouds of various textiles (i.e. as obtained with a 3D-scanner). This would allow the creation of a reconstructed data-base of textiles that could be later simulated with the inextensible model. Furthermore, we expect to extend the point-cloud reconstruction algorithm for surfaces to higher dimensional manifolds by iterating the hyperplane sections, and reconstructing the manifold from lower dimensional slices. A further extension would be to the study of real algebraic varieties of any dimension, where point cloud samples can be obtained from their equations and refined where necessary. This is hoped to lead our method to detect a Whitney stratification, and the Morse

cellular decomposition after [46] of the variety, by purely numerical methods.

PUBLICATIONS

Some of the research leading to this thesis has appeared previously in the following publications.

Journal Articles

- Franco Coltraro, Jaume Amorós, Maria Alberich-Carramiñana and Carme Torras: **An inextensible model for the robotic manipulation of textiles**. *Applied Mathematical Modelling*, Vol. 101 (2022), pp 832-858. DOI: <https://doi.org/10.1016/j.apm.2021.09.013>.

Proceedings

- Maria Alberich-Carramiñana, Jaume Amorós and Franco Coltraro. **Developable surfaces with prescribed boundary**. In *Extended Abstracts GEOMVAP 2019*, 127-132. Springer-Birkhäuser, 2021.

Under review

- F. Coltraro, J. Fontana, J. Amorós, M. Alberich-Carramiñana, J. Borràs and C. Torras. A Representation of Cloth States based on a Derivative of the Gauss Linking Integral. *Applied Mathematics and Computation*.

Under preparation

- F. Coltraro, J. Amorós, M. Alberich-Carramiñana and C. Torras. The isometric strain model for cloth simulation: contacts, friction, self-collisions, aerodynamics, and experimental validation.
- F. Coltraro, M. Verdaguer, J. Amorós, M. Alberich-Carramiñana and C. Torras. Morse cell decomposition and parametrization of surfaces from point-clouds.

Conferences and talks

- **Mechanics of inextensible surfaces**. *Workshop Trobada GEOM-VAP (Geometry of varieties and Applications Group)*. Cardona, Spain. January 23th, 2018.

- **Collisions and friction for inextensible cloth simulation.** *Conference Women in Geometry and Topology (organized by GEOMVAP)*. Barcelona, Spain. September 26th, 2019.
- **Morse cell decomposition and parametrization of surfaces from point-clouds.** *XVII EACA 2022 (Encuentro Álgebra Computacional y Aplicaciones)*. Castellón, Spain. June 21th, 2022.
- **Contacts, friction and self-collisions for inextensible cloth.** *XXVII CEDYA 2022 (Congreso de Ecuaciones Diferenciales y Aplicaciones)*. Zaragoza, Spain. July 21th, 2022.
- **Experimental validation of an inextensible cloth model.** *AICA 2022 (Applications to Industry of Computational Algebra)*. Barcelona, Spain. November 10th, 2022.
- **Mathematical problems related to the robotic manipulation of cloth.** *Seminar at Departamento Matemática Aplicada I, Universidad de Sevilla*. Seville, Spain. November 17th, 2022.

Supervision of Bachelor's and Master's theses

- Román Arañó Llach, July 2020. **Validación del modelo de tela del proyecto Clothlide mediante la simulación y el cálculo numérico.** *Bachelor Thesis*. UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Matemàtiques. (Co-directed with Jaume Amorós).
- José Maria Julià, February 2022. **Primeros pasos en el control de la tela por simulación isométrica.** *Master Thesis*. UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Matemàtiques.
- Julen Antonio Echevarria, April 2022. **Manipulació robòtica de tela: percepció i ajust de model.** *Bachelor Thesis*. UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Matemàtiques.

Part IV

APPENDIX

APPENDIX

A.1 TABLES

Material	Size	Motion	Speed	Error	STD	Alpha	Delta
"Stiff-cotton"	"A2"	"Shake"	"Slow"	0.433	0.661	0.336	0.16
"Stiff-cotton"	"A2"	"Shake"	"Fast"	0.516	0.77	0.657	0.181
"Stiff-cotton"	"A2"	"Twist"	"Slow"	0.324	0.533	0.496	0.222
"Stiff-cotton"	"A2"	"Twist"	"Fast"	0.562	0.918	0.817	0.284
"Stiff-cotton"	"A3"	"Shake"	"Slow"	0.268	0.711	0.336	0.201
"Stiff-cotton"	"A3"	"Shake"	"Fast"	0.336	0.798	0.817	0.222
"Stiff-cotton"	"A3"	"Twist"	"Slow"	0.267	0.711	0.336	0.243
"Stiff-cotton"	"A3"	"Twist"	"Fast"	0.315	0.823	0.898	0.263
"Wool"	"A2"	"Shake"	"Slow"	0.511	0.836	0.199	0.058
"Wool"	"A2"	"Shake"	"Fast"	0.991	1.485	0.437	0.082
"Wool"	"A2"	"Twist"	"Slow"	0.451	0.738	0.437	0.131
"Wool"	"A2"	"Twist"	"Fast"	0.844	1.314	0.532	0.095
"Wool"	"A3"	"Shake"	"Slow"	0.289	0.747	0.294	0.082
"Wool"	"A3"	"Shake"	"Fast"	0.447	1.083	0.579	0.107
"Wool"	"A3"	"Twist"	"Slow"	0.219	0.663	0.294	0.107
"Wool"	"A3"	"Twist"	"Fast"	0.361	0.919	0.579	0.107
"Polyester"	"A2"	"Shake"	"Slow"	0.887	1.187	0.17	0.026
"Polyester"	"A2"	"Shake"	"Fast"	1.36	1.819	0.307	0.041
"Polyester"	"A2"	"Twist"	"Slow"	0.444	0.685	0.197	0.041
"Polyester"	"A2"	"Twist"	"Fast"	1.152	1.617	0.307	0.048
"Polyester"	"A3"	"Shake"	"Slow"	0.463	0.903	0.142	0.033
"Polyester"	"A3"	"Shake"	"Fast"	0.471	0.99	0.28	0.041
"Polyester"	"A3"	"Twist"	"Slow"	0.243	0.689	0.115	0.048
"Polyester"	"A3"	"Twist"	"Fast"	0.359	0.854	0.307	0.055
"Denim"	"A2"	"Shake"	"Slow"	0.628	1.138	0.416	0.139
"Denim"	"A2"	"Shake"	"Fast"	0.841	1.252	0.898	0.201
"Denim"	"A2"	"Twist"	"Slow"	0.438	0.932	0.416	0.222
"Denim"	"A2"	"Twist"	"Fast"	0.905	1.35	0.978	0.263
"Denim"	"A3"	"Shake"	"Slow"	0.306	0.877	0.416	0.181
"Denim"	"A3"	"Shake"	"Fast"	0.334	0.959	0.737	0.201
"Denim"	"A3"	"Twist"	"Slow"	0.266	0.764	0.416	0.243
"Denim"	"A3"	"Twist"	"Fast"	0.288	0.804	0.737	0.305

Table A.1: Summary of results of the aerodynamic study for the first repetition (with bare hands) of the experiments. For all the 32 recordings we display the characteristics of the recording (fabric, size, motion and speed), and the optimal value of the fitted parameters along with their associated absolute error and spatial standard deviation.

Material	Size	Motion	Speed	Error	STD	Alpha	Delta
"Stiff-cotton"	"A2"	"Shake"	"Slow"	0.446	0.682	0.416	0.139
"Stiff-cotton"	"A2"	"Shake"	"Fast"	0.561	0.898	0.737	0.181
"Stiff-cotton"	"A2"	"Twist"	"Slow"	0.351	0.569	0.496	0.222
"Stiff-cotton"	"A2"	"Twist"	"Fast"	0.681	1.098	0.898	0.305
"Stiff-cotton"	"A3"	"Shake"	"Slow"	0.256	0.651	0.336	0.181
"Stiff-cotton"	"A3"	"Shake"	"Fast"	0.331	0.805	0.737	0.222
"Stiff-cotton"	"A3"	"Twist"	"Slow"	0.282	0.716	0.496	0.284
"Stiff-cotton"	"A3"	"Twist"	"Fast"	0.329	0.838	0.978	0.305
"Wool"	"A2"	"Shake"	"Slow"	0.559	0.842	0.247	0.07
"Wool"	"A2"	"Shake"	"Fast"	0.964	1.456	0.532	0.095
"Wool"	"A2"	"Twist"	"Slow"	0.421	0.72	0.294	0.095
"Wool"	"A2"	"Twist"	"Fast"	0.958	1.408	0.627	0.107
"Wool"	"A3"	"Shake"	"Slow"	0.316	0.796	0.342	0.082
"Wool"	"A3"	"Shake"	"Fast"	0.403	1.028	0.532	0.107
"Wool"	"A3"	"Twist"	"Slow"	0.235	0.627	0.294	0.095
"Wool"	"A3"	"Twist"	"Fast"	0.468	1.076	0.627	0.144
"Polyester"	"A2"	"Shake"	"Slow"	1.025	1.327	0.142	0.026
"Polyester"	"A2"	"Shake"	"Fast"	1.378	1.761	0.28	0.041
"Polyester"	"A2"	"Twist"	"Slow"	0.52	0.811	0.17	0.033
"Polyester"	"A2"	"Twist"	"Fast"	1.22	1.696	0.252	0.041
"Polyester"	"A3"	"Shake"	"Slow"	0.467	1.227	0.17	0.033
"Polyester"	"A3"	"Shake"	"Fast"	0.575	1.247	0.28	0.041
"Polyester"	"A3"	"Twist"	"Slow"	0.543	1.429	0.17	0.033
"Polyester"	"A3"	"Twist"	"Fast"	0.528	1.166	0.417	0.055
"Denim"	"A2"	"Shake"	"Slow"	0.512	0.978	0.336	0.139
"Denim"	"A2"	"Shake"	"Fast"	0.739	1.083	0.817	0.181
"Denim"	"A2"	"Twist"	"Slow"	0.526	0.983	0.336	0.222
"Denim"	"A2"	"Twist"	"Fast"	0.876	1.318	0.978	0.243
"Denim"	"A3"	"Shake"	"Slow"	0.376	1.194	0.416	0.181
"Denim"	"A3"	"Shake"	"Fast"	0.368	1.15	0.737	0.201
"Denim"	"A3"	"Twist"	"Slow"	0.334	0.991	0.978	0.222
"Denim"	"A3"	"Twist"	"Fast"	0.375	1.121	0.496	0.222

Table A.2: Summary of results of the aerodynamic study for the second repetition (with hanger) of the experiments. For all the 32 recordings we display the characteristics of the recording (fabric, size, motion and speed), and the optimal value of the fitted parameters along with their associated absolute error and spatial standard deviation.

A.2 FIGURES

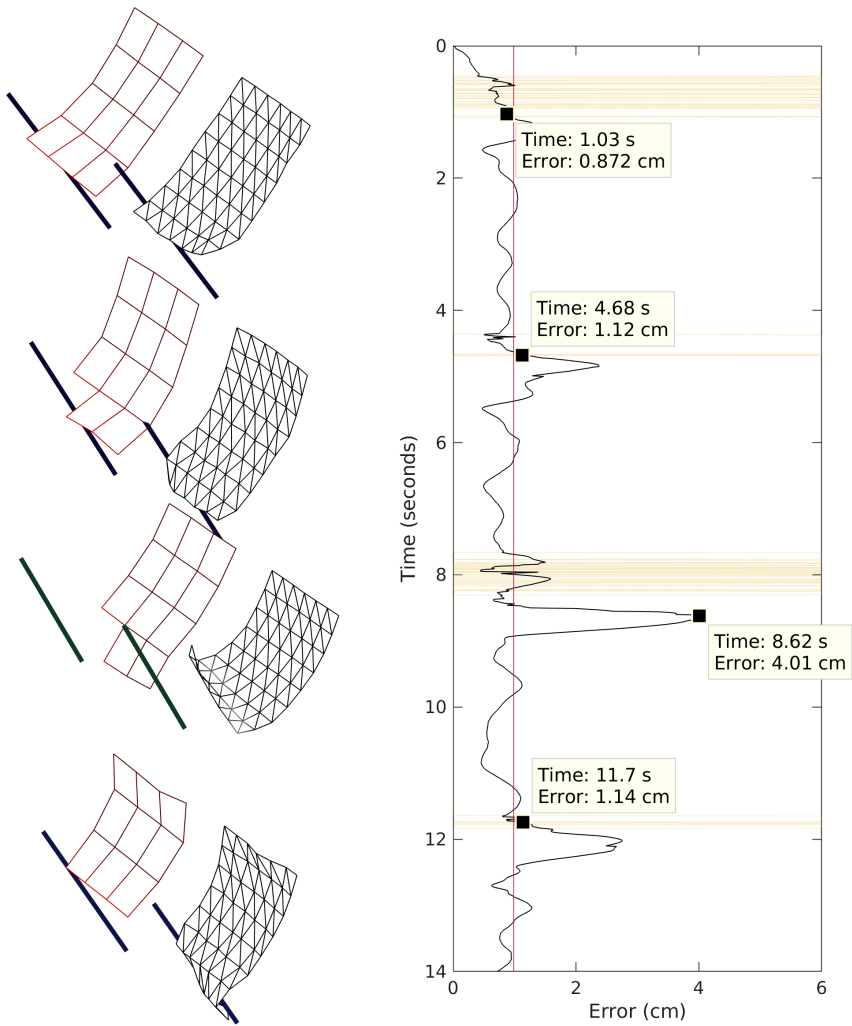


Figure A.1: Four frames comparing the recorded hitting of A2 denim (left) with its inextensible simulation (right); being its average error 0.98 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines we highlight the moments in which the stick is in contact with the cloth.

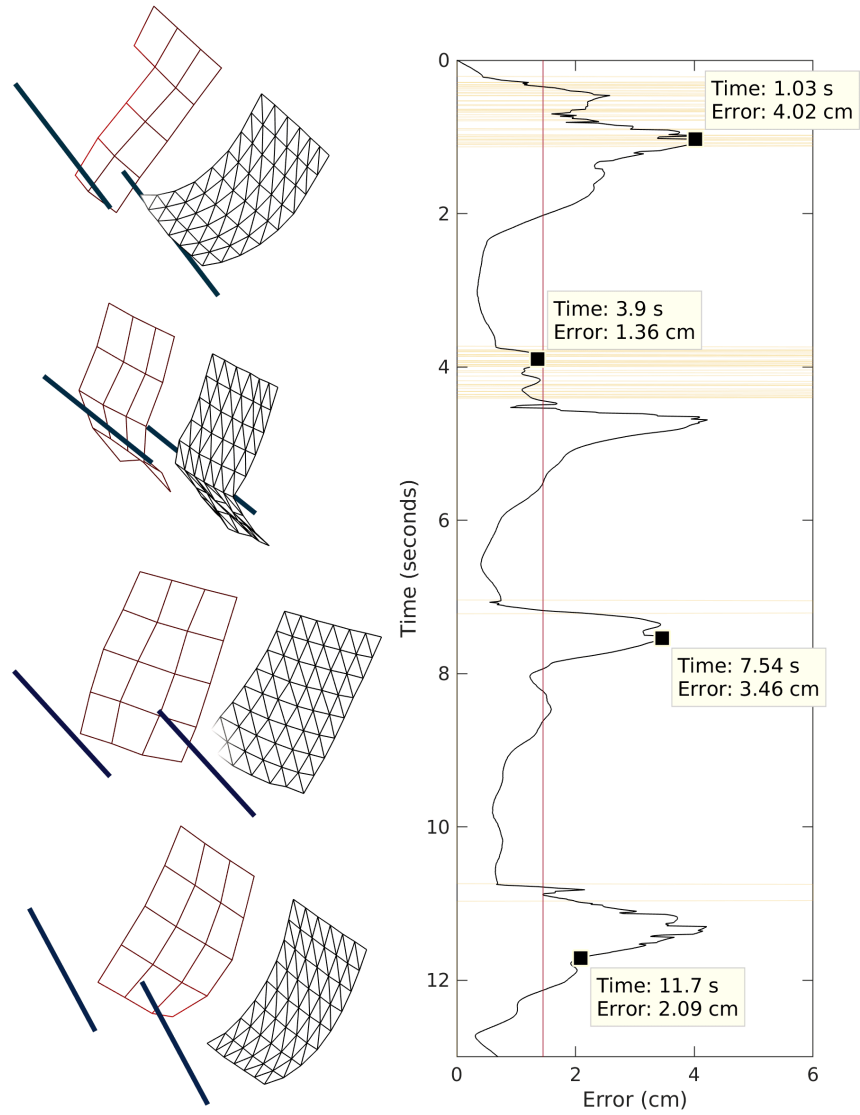


Figure A.2: Four frames comparing the recorded hitting of A2 wool (left) with its inextensible simulation (right); being its average error 1.39 cm. On the right, we show a full plot (vertically) of the absolute error and with yellow lines we highlight the moments in which the stick is in contact with the cloth.

BIBLIOGRAPHY

- [1] Vincent Acary and Bernard Brogliato. *Numerical methods for non-smooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.
- [2] Jürgen Adlinger, Isaac Klapper, and Michael Tabor. “Formulae for the calculation and estimation of writhe.” In: *Journal of Knot Theory and Its Ramifications* 4 (1995), pp. 343–372.
- [3] Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. “Computing the discrete Fréchet distance in subquadratic time.” In: *SIAM Journal on Computing* 43.2 (2014), pp. 429–449.
- [4] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. “Learning the semantics of object–action relations by observation.” In: *International Journal of Robotics Research* 30.10 (2011), pp. 1229–1249.
- [5] Helmut Alt and Michael Godau. “Computing the Fréchet distance between two polygonal curves.” In: *International Journal of Computational Geometry & Applications* 5.01n02 (1995), pp. 75–91.
- [6] Paola F. Antonietti, Paolo Biscari, Alaleh Tavakoli, Marco Verani, and Maurizio Vianello. “Theoretical study and numerical simulation of textiles.” In: *Applied Mathematical Modelling* 35.6 (2011), pp. 2669–2681. ISSN: 0307-904X. URL: <https://www.sciencedirect.com/science/article/pii/S0307904X10004725>.
- [7] Román Aranyó Llach. *Validación del modelo de tela del proyecto Clothlide mediante la simulación y el cálculo numérico*. Universitat Politècnica de Catalunya, Degree Thesis, 2020. URL: <http://hdl.handle.net/2117/330056>.
- [8] Ivo Babuška and Manil Suri. “Locking effects in the finite element approximation of elasticity problems.” In: *Numerische Mathematik* 62.1 (1992), pp. 439–463. ISSN: 0945-3245. URL: <https://doi.org/10.1007/BF01396238>.
- [9] David Baraff, Andrew P. Witkin, and Michael Kass. “Untangling cloth.” In: *ACM SIGGRAPH 2003 Papers* (2003).
- [10] David Baraff and Andrew Witkin. “Large Steps in Cloth Simulation.” In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’98. New York, NY, USA: ACM, 1998, pp. 43–54. ISBN: 0-89791-999-8. URL: <http://doi.acm.org/10.1145/280814.280821>.

- [11] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. "A Survey on Position-Based Simulation Methods in Computer Graphics." In: *Computer Graphics Forum* 33.6 (2014), pp. 228–251. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12346>.
- [12] Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. "A Quadratic Bending Model for Inextensible Surfaces." In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 227–230. ISBN: 3-905673-36-3. URL: <http://dl.acm.org/citation.cfm?id=1281957.1281987>.
- [13] T.S. Bhanumurthy and T.S.B. Murthy. *A Modern Introduction to Ancient Indian Mathematics*. New Age International, 2009. ISBN: 9788122426007. URL: <https://books.google.es/books?id=VosxlQJjRDAC>.
- [14] Blender. <https://www.blender.org/>.
- [15] Julia Borràs, Guillem Alenyà, and Carme Torras. "A Grasping-Centered Analysis for Cloth Manipulation." In: *IEEE Transactions on Robotics* 36.3 (2020), pp. 924–936. DOI: [10.1109/TR0.2020.2986921](https://doi.org/10.1109/TR0.2020.2986921).
- [16] C. Bousquet-Jette, S. Achiche, D. Beaini, Y.S. Law-Kam Cio, C. Leblond-Ménard, and M. Raison. "Fast scene analysis using vision and artificial intelligence for object prehension by an assistive robot." In: *Engineering Applications of Artificial Intelligence* 63 (2017), pp. 33–44. ISSN: 0952-1976. URL: <https://www.sciencedirect.com/science/article/pii/S0952197617300799>.
- [17] Dietrich Braess. *Finite elements. Theory, fast solvers and applications in elasticity theory*. Berlin: Springer., 2007.
- [18] Robert Bridson, Ronald Fedkiw, and John Anderson. "Robust treatment of collisions, contact and friction for cloth animation." In: *ACM SIGGRAPH 2005 Courses* (2005).
- [19] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. ISBN: 9780132125895.
- [20] Frédéric Cazals, Andrea Roth, Charles H. Robert, and Mueller Christian. "Towards Morse Theory for Point Cloud Data." In: *Research Report RR-8331, INRIA*. (2013), p. 37.
- [21] Kwang-Jin Choi and Hyeong-Seok Ko. "Stable but Responsive Cloth." In: *ACM Transactions on Graphics (TOG)* 21.3 (July 2002), pp. 604–611. ISSN: 0730-0301. URL: <http://doi.acm.org/10.1145/566654.566624>.

- [22] Kwang-Jin Choi and Hyeong-Seok Ko. “Research Problems in Clothing Simulation.” In: *Computer-Aided Design* 37.6 (May 2005), pp. 585–592. ISSN: 0010-4485. URL: <http://dx.doi.org/10.1016/j.cad.2004.11.002>.
- [23] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. “Yarn-Level Simulation of Woven Cloth.” In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia)* 33.6 (2014). URL: <http://www.gmr.v.es/Publications/2014/CLM014>.
- [24] David Clyde, Joseph Teran, and Rasmus Tamstorf. “Modeling and Data-Driven Parameter Estimation for Woven Fabrics.” In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '17. Los Angeles, California: Association for Computing Machinery, 2017. ISBN: 9781450350914. URL: <https://doi.org/10.1145/3099564.3099577>.
- [25] Adrià Colomé and Carme Torras. *Reinforcement Learning of Bimanual Robot Skills*. Volume 134 of Springer Tracts in Advanced Robotics. Springer, 2020.
- [26] Adrià Colomé and Carme Torras. “Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes.” In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 602–615. DOI: [10.1109/TR0.2018.2808924](https://doi.org/10.1109/TR0.2018.2808924).
- [27] David L. Davies and Donald W. Bouldin. “A Cluster Separation Measure.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: [10.1109/TPAMI.1979.4766909](https://doi.org/10.1109/TPAMI.1979.4766909).
- [28] Timothy A. Davis and William W. Hager. “Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves.” In: *ACM Transactions on Mathematical Software* 35 (2009), 27:1–27:23.
- [29] Markus Deserno. “Fluid lipid membranes: From differential geometry to curvature stresses.” In: *Chemistry and Physics of Lipids* 185 (2015), pp. 11–45.
- [30] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics, 2006.
- [31] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlavac, Tae-Kyun Kim, and Sotiris Malassiotis. “Folding Clothes Autonomously: A Complete Pipeline.” In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1461–1478.

- [32] Elliot English and Robert Bridson. "Animating Developable Surfaces Using Nonconforming Elements." In: *ACM TOG* 27.3 (Aug. 2008), 66:1–66:5. ISSN: 0730-0301. URL: <http://doi.acm.org/10.1145/1360612.1360665>.
- [33] Moritz Epple. "Orbits of Asteroids, a Braid, and the First Link Invariant." In: *The Mathematical Intelligencer* 20 (1998), pp. 45–52.
- [34] Olaf Etmuss, Michael Keckeisen, and Wolfgang Strasser. "A fast finite element solution for cloth modelling." In: *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.* 2003, pp. 244–251. DOI: [10.1109/PCCGA.2003.1238266](https://doi.org/10.1109/PCCGA.2003.1238266).
- [35] J. Douglas Faires and Richard L. Burden. *Numerical Methods, 4th.* Cengage Learning, 2012. ISBN: 9780495114765.
- [36] Gerd Fischer and Jens Piontkowski. *Ruled Varieties: An Introduction to Algebraic Differential Geometry.* Advanced Lectures in Mathematics. Vieweg and Teubner Verlag, 2012. ISBN: 9783322802170.
- [37] Michael S. Floater and Kai Hormann. "Parameterization of Triangulations and Unorganized Points." In: *Tutorials on Multiresolution in Geometric Modelling.* 2002.
- [38] L. Gan, N.G. Ly, and G.P. Steven. "A Study of Fabric Deformation Using Nonlinear Finite Elements." In: *Textile Research Journal* 65.11 (1995), pp. 660–668. URL: <https://doi.org/10.1177/004051759506501106>.
- [39] Jie Gao, Rik Sarkar, and Xianjin Zhu. "Morse-Smale Decomposition, Cut Locus and Applications in Sensor Networks." In: *Pre-print* (2008). URL: https://www.researchgate.net/publication/228414975_Morse-smale_decomposition_cut_locus_and_applications_in_wireless_sensor_networks.
- [40] Irene Garcia-Camacho et al. "Benchmarking Bimanual Cloth Manipulation." In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1111–1118. DOI: [10.1109/LRA.2020.2965891](https://doi.org/10.1109/LRA.2020.2965891).
- [41] Mohammad Ghalandari, Saeed Bornassi, Shahabbodin Shamshirband, Amir Mosavi, and Kwok Wing Chau. "Investigation of submerged structures' flexibility on sloshing frequency using a boundary element method and finite element analysis." In: *Engineering Applications of Computational Fluid Mechanics* 13.1 (2019), pp. 519–528. URL: <https://doi.org/10.1080/19942060.2019.1619197>.
- [42] Ayse Gider. *An Online Fabric Database to Link Fabric Drape and End-Use Properties.* LSU Master's Thesis. 3902, 2004. URL: https://digitalcommons.lsu.edu/gradschool_theses/3902.

- [43] Herman Gluck. "Almost all simply connected closed surfaces are rigid." In: *Geometric Topology*. Ed. by Leslie Curtis Glaser and Thomas Benjamin Rushing. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 225–239. ISBN: 978-3-540-37412-1.
- [44] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. "Efficient Simulation of Inextensible Cloth." In: vol. 26. 3. New York, NY, USA: ACM SIGGRAPH 2007 Papers, July 2007. URL: <http://doi.acm.org/10.1145/1276377.1276438>.
- [45] Gene H. Golub. *Matrix computations*. John Hopkins University Press, 1983.
- [46] Mark Goresky and Robert MacPherson. *Stratified morse theory*. Springer Verlag, 1988.
- [47] Dongsoo Han. "Area Preserving Strain Limiting." In: *Workshop on Virtual Reality Interaction and Physical Simulation*. Ed. by Fabrice Jaillet, Florence Zara, and Gabriel Zachmann. The Eurographics Association, 2015. ISBN: 978-3-905674-98-9. DOI: [10.2312/vriphys.20151340](https://doi.org/10.2312/vriphys.20151340).
- [48] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. "Asynchronous contact mechanics." In: *SIGGRAPH 2009*. 2009.
- [49] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. "Robust treatment of simultaneous collisions." In: *ACM SIGGRAPH 2008 papers* (2008).
- [50] John Hearle, Percy Grosberg, and Stanley Backer. *Structural mechanics of fibers, yarns, and fabrics*. Vol. 1. New York: Wiley-Interscience, 1969.
- [51] Jeff Henrikson. "Completeness and total boundedness of the Hausdorff metric." In: *MIT Undergraduate Journal of Mathematics* 1.69-80 (1999), p. 10.
- [52] Morris Hirsch. *Differential Topology*. Springer Verlag, 1976.
- [53] Edmond SL Ho, Taku Komura, Subramanian Ramamoorthy, and Sethu Vijayakumar. "Controlling humanoid robots in topology coordinates." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 178–182.
- [54] Shu Lim Ho. "Topology-based character motion synthesis." PhD thesis. University of Edinburgh, 2011.
- [55] Kai Hormann and Gunther Greiner. "Quadrilateral remeshing." In: *Proceedings of Vision, Modeling and Visualization*. 2000, pp. 153–162.
- [56] Jinlian Hu. *Structure and mechanics of woven fabrics*. The Textile Institute and Woodhead Publishing Ltd (UK), 2004.

- [57] Vladimir Ivan, Dmitry Zarubin, Marc Toussaint, Taku Komura, and Sethu Vijayakumar. "Topology-based representations for motion planning and generalization in dynamic environments with interactions." In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1151–1163.
- [58] Rishabh Jangir, Guillem Alenya, and Carme Torras. "Dynamic Cloth Manipulation with Deep Reinforcement Learning." In: *IEEE International Conference on Robotics and Automation (ICRA), Paris, France* (2020), pp. 4630–4636.
- [59] Michel Jean. "The non-smooth contact dynamics method." In: *Computer Methods in Applied Mechanics and Engineering* 177 (1999), pp. 235–257.
- [60] Chenfanfu Jiang, Theodore Gast, and Joseph Teran. "Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact." In: *ACM Transactions on Graphics* 36.4 (July 2017), 152:1–152:14. ISSN: 0730-0301. URL: <http://doi.acm.org/10.1145/3072959.3073623>.
- [61] Pablo Jiménez. "Visual grasp point localization, classification and state recognition in robotic manipulation of cloth: An overview." In: *Robotics and Autonomous Systems* 92 (2017), pp. 107–125.
- [62] Pablo Jimenez. "Learning in autonomous and intelligent systems: Overview and biases from data sources." In: *Arbor (Madrid)* 197.802 (2022), a627: 1–a627: 12. URL: <http://hdl.handle.net/2117/361981>.
- [63] Ning Jin, Wenlong Lu, Zhenglin Geng, and Ronald P. Fedkiw. "Inequality Cloth." In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '17. Los Angeles, California: ACM, 2017, 16:1–16:10. ISBN: 978-1-4503-5091-4. URL: <http://doi.acm.org/10.1145/3099564.3099568>.
- [64] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. "Staggered projections for frictional contact in multibody systems." In: *ACM Transactions on Graphics* 27 (2008), p. 164.
- [65] S. Kawabata. *The development of the objective measurement of fabric handle*. Kyoto, Textile Machinery Society, Japan: Eurographics Association, 1982, pp. 31–59.
- [66] Konstantin Klenin and Jörg Langowski. "Computation of writhe in modeling of supercoiled DNA." In: *Biopolymers* 54.5 (2000), pp. 307–317. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-0282%2820001015%2954%3A5%3C307%3A%3AAID-BIP20%3E3.0.CO%3B2-Y>.
- [67] Markus Kunze. *Non-Smooth Dynamical Systems*. Vol. 1744. Springer, Jan. 2000. ISBN: 978-3-540-67993-6. DOI: [10.1007/BFb0103852](https://doi.org/10.1007/BFb0103852).

- [68] Michael Levitt. “Protein folding by restrained energy minimization and molecular dynamics.” In: *Journal of molecular biology* 170.3 (1983), pp. 723–764.
- [69] Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E. Brown, and Laurence Boissieux. “An implicit frictional contact solver for adaptive cloth simulation.” In: *ACM Transactions on Graphics (TOG)* 37 (2018), pp. 1–15.
- [70] Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. “Codi-dimensional incremental potential contact.” In: *ACM Transactions on Graphics (TOG)* 40 (2021), pp. 1–24.
- [71] Yinxiao Li, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter Allen. “Folding Deformable Objects using Predictive Simulation and Trajectory Optimization.” In: *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems* (Dec. 2015), pp. 6000–6006.
- [72] Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. “Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation.” In: *IEEE/RSJ Int. Conf. on Intel. Rob. and Syst.* 2020, pp. 5619–5626.
- [73] Xiang Liu and Lisheng Liu. “An immersed transitional interface finite element method for fluid interacting with rigid/deformable solid.” In: *Engineering Applications of Computational Fluid Mechanics* 13.1 (2019), pp. 337–358. URL: <https://doi.org/10.1080/19942060.2019.1586774>.
- [74] D. W. Lloyd. “The Analysis of Complex Deformations.” In: *Mechanics of Flexible Fiber Assemblies* (1980), pp. 311–342. URL: <https://ci.nii.ac.jp/naid/10003779259/en/>.
- [75] Mickaël Ly, Jean louis Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. “Projective dynamics with dry frictional contact.” In: *ACM Transactions on Graphics (TOG)* 39 (2020), 57:1–57:8.
- [76] Guanghui Ma, Juntao Ye, Jituo Li, and Xiaopeng Zhang. “Anisotropic Strain Limiting for Quadrilateral and Triangular Cloth Meshes.” In: *Computer Graphics Forum* 35 (2015).
- [77] Tianlu Mao, Shihong Xia, and Zhaoqi Wang. “Evaluating simplified air force models for cloth simulation.” In: *11th IEEE International Conference on Computer-Aided Design and Computer Graphics*. 2009, pp. 81–86. DOI: [10.1109/CADCG.2009.5246929](https://doi.org/10.1109/CADCG.2009.5246929).
- [78] Jan Matas, Stephen James, and Andrew J Davison. “Sim-to-real reinforcement learning for deformable object manipulation.” In: *Proceedings of Conference on Robotic Learning*. 2018.

- [79] Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, W. Matusik, Miguel Otaduy, and S. Marschner. "Data-Driven Estimation of Cloth Simulation Models." In: *Computer Graphics Forum* 31 (May 2012), pp. 519–528. DOI: [10.1111/j.1467-8659.2012.03031.x](https://doi.org/10.1111/j.1467-8659.2012.03031.x).
- [80] Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. "A geometric approach to robotic laundry folding." In: *International Journal of Robotics Research* 31.2 (2012), pp. 249–267.
- [81] Domen Mongus, B. Repnik, Marjan Mernik, and Borut Alik. "A Hybrid Evolutionary Algorithm for Tuning a Cloth-simulation Model." In: *Applied Soft Computing* 12.1 (Jan. 2012), pp. 266–273. ISSN: 1568-4946. URL: <http://dx.doi.org/10.1016/j.asoc.2011.08.047>.
- [82] *MuJoCo (Multi-Joint dynamics with Contact)*. <http://www.mujoco.org/index.html>.
- [83] James R. Munkres. *Elements of algebraic topology*. Addison-Wesley, 1984.
- [84] Rahul Narain, Armin Samii, and James F. O'Brien. "Adaptive Anisotropic Remeshing for Cloth Simulation." In: *ACM Transactions on Graphics* 31.6 (Nov. 2012), 152:1–152:10. ISSN: 0730-0301. URL: <http://doi.acm.org/10.1145/2366145.2366171>.
- [85] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. "Physically Based Deformable Models in Computer Graphics." In: *Computer Graphics Forum* 25.4 (2006), pp. 809–836. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2006.01000.x>.
- [86] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006.
- [87] Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus H. Gross. "Implicit Contact Handling for Deformable Objects." In: *Computer Graphics Forum* 28 (2009).
- [88] S. A. Paipetis. "Mathematical modelling of composites." In: *Developments in Composite Material-2-stress Analysis Holister* (1981), pp. 1–29.
- [89] Phillip Joseph Phillips and Mary F. Wheeler. "Overcoming the problem of locking in linear elasticity and poroelasticity: an heuristic approach." In: *Computational Geosciences* 13.1 (2009), pp. 5–12. ISSN: 1573-1499. URL: <https://doi.org/10.1007/s10596-008-9114-x>.
- [90] William F. Pohl. "DNA and differential geometry." In: *The Mathematical Intelligencer* 3 (1980), pp. 20–27.

- [91] Florian T Pokorny, Johannes A Stork, and Danica Kragic. "Grasping objects with holes: A topological approach." In: *2013 IEEE international conference on robotics and automation*. IEEE. 2013, pp. 1100–1107.
- [92] Xavier Provot. "Collision and self-collision handling in cloth model dedicated to design garments." In: *Computer Animation and Simulation*. 1997.
- [93] Xavier Provot. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior." In: *Graphics Interface* 23(19) (Sept. 2001).
- [94] Jianing Qian, Thomas Weng, Luxin Zhang, Brian Okorn, and David Held. "Cloth Region Segmentation for Robust Grasp Selection." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 9553–9560.
- [95] Arnau Ramisa, Guillem Alenyà, Francesc Moreno-Noguer, and Carme Torras. "Learning RGB-D descriptors of garment parts for informed robot grasping." In: *Engineering Applications of Artificial Intelligence* 35 (2014), pp. 246–258. ISSN: 0952-1976. URL: <https://www.sciencedirect.com/science/article/pii/S095219761400147X>.
- [96] Abdullah Haroon Rasheed, Victor Romero, Florence Bertails-Descoubes, Stefanie Wuhrer, Jean-Sébastien Franco, and Arnaud Lazarus. "Learning to Measure the Static Friction Coefficient in Cloth Contact." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 9909–9918.
- [97] Abdullah Haroon Rasheed, Victor Romero, Florence Bertails-Descoubes, Stefanie Wuhrer, Jean-Sébastien Franco, and Arnaud Lazarus. "A Visual Approach to Measure Cloth-Body and Cloth-Cloth Friction." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2021).
- [98] Renzo L. Ricca and Bernardo Nipoti. "Gauss linking number revisited." In: *Journal of Knot Theory and Its Ramifications* 20 (2011), pp. 1325–1343.
- [99] Javier Rodriguez-Navarro and Antonio Susin. "Non structured meshes for Cloth GPU simulation using FEM." In: *Vriphys: 3rd Workshop in Virtual Reality, Interactions, and Physical Simulation*. Ed. by Cesar Mendoza and Isabel Navazo. The Eurographics Association, 2006. ISBN: 3-905673-61-4. DOI: [10.2312/PE/vriphys/vriphys06/001-007](https://doi.org/10.2312/PE/vriphys/vriphys06/001-007).
- [100] José Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Robotic manipulation and sensing of deformable objects in domestic and industrial applica-

- tions: a survey." In: *The International Journal of Robotics Research* 37.7 (2018), pp. 688–716. DOI: [10.1177/0278364918779698](https://doi.org/10.1177/0278364918779698).
- [101] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey." In: *International Journal of Robotic Research* 37.7 (2018), pp. 688–716.
- [102] Matthias W. Seeger. "Low Rank Updates for the Cholesky Decomposition." In: *Technical Report* (2004). URL: <https://infoscience.epfl.ch/record/161468>.
- [103] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, et al. "Deep imitation learning of sequential fabric smoothing policies." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 9651–9658.
- [104] Michael Shelley and Jun Zhang. "Flapping and Bending Bodies Interacting with Fluid Flows." In: *Annual Review of Fluid Mechanics* 43 (Jan. 2011), pp. 449–465. DOI: [10.1146/annurev-fluid-121108-145456](https://doi.org/10.1146/annurev-fluid-121108-145456).
- [105] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. "Reflections on simultaneous impact." In: *ACM Transactions on Graphics (TOG)* 31 (2012), pp. 1–12.
- [106] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V. Dimarogonas, and Danica Kragic. "Dual arm manipulation - A survey." In: *Robotics and Autonomous Systems* 60.10 (2012), pp. 1340–1353. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2012.07.005>.
- [107] Johannes A Stork, Florian T Pokorny, and Danica Kragic. "A topology-based object representation for clasping, latching and hooking." In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2013, pp. 138–145.
- [108] Johannes A. Stork, Florian T. Pokorny, and Danica Kragic. "Integrated motion and clasp planning with virtual linking." In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3007–3014.
- [109] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. "EMD Net: An encode–manipulate–decode network for cloth manipulation." In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1771–1778.
- [110] John R. Taylor. *Classical Mechanics*. University Science Books, 2005. ISBN: 9781891389221.

- [111] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. "Elastically Deformable Models." In: *SIGGRAPH Computer Graphics* 21.4 (Aug. 1987), pp. 205–214. ISSN: 0097-8930. URL: <http://doi.acm.org/10.1145/37402.37427>.
- [112] Bernhard Thomaszewski, Simon Pabst, and Wolfgang Strasser. "Continuum-based Strain Limiting." In: *Computer Graphics Forum* 28.2 (2009), pp. 569–576. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01397.x>.
- [113] Vitaly Ushakov. "The explicit general solution of trivial Monge-Ampère equation." In: *Commentarii Mathematici Helvetici* 75 (2000), pp. 125–133.
- [114] Remco C Veltkamp and Michiel Hagedoorn. "State of the art in shape matching." In: *Principles of visual information retrieval* (2001), pp. 87–119.
- [115] Pascal Volino and Nadia Magnenat-Thalmann. "Accurate collision response on polygonal meshes." In: *Proceedings Computer Animation 2000* (2000), pp. 154–163.
- [116] Pascal Volino and Nadia Magnenat-Thalmann. "Resolving surface collisions through intersection contour minimization." In: *SIGGRAPH 2006*. 2006.
- [117] Huamin Wang, Ravi Ramamoorthi, and James F. O'Brien. "Data-Driven Elastic Models for Cloth: Modeling and Measurement." In: *ACM Transactions on Graphics* 30.4 (July 2011). Proceedings of ACM SIGGRAPH 2011, Vancouver, BC Canada, 71:1–11. URL: <http://graphics.berkeley.edu/papers/Wang-DDE-2011-08/>.
- [118] Zhendong Wang, Tongtong Wang, Min Tang, and Ruofeng Tong. "Efficient and robust strain limiting and treatment of simultaneous collisions with semidefinite programming." In: *Computational Visual Media* 2.2 (2016), pp. 119–130. ISSN: 2096-0662. URL: <https://doi.org/10.1007/s41095-016-0042-8>.
- [119] Florentin Wörgötter, Eren Erdal Aksoy, Norbert Krüger, Justus Piater, Ales Ude, and Minija Tamosiunaite. "A simple ontology of manipulation actions based on hand-object relations." In: *IEEE Transactions on Autonomous Mental Development* 5.2 (2013), pp. 117–134.
- [120] Z. Wu, Chikit Au, and Matthew Yuen. "Mechanical properties of fabric materials for draping simulation." In: *International Journal of Clothing Science and Technology* 15 (Feb. 2003), pp. 56–68. DOI: [10.1108/09556220310461169](https://doi.org/10.1108/09556220310461169).
- [121] Juntao Ye, Guanghui Ma, Liguang Jiang, Lan Chen, Jituo Li, Gang-Yu Xiong, Xiaopeng Zhang, and Min Tang. "A Unified Cloth Untangling Framework Through Discrete Collision Detection." In: *Computer Graphics Forum* 36 (2017).

- [122] Hang Yin, Anastasia Varava, and Danica Kragic. "Modeling, learning, perception, and control methods for deformable object manipulation." In: *Science Robotics* 6.54 (2021), eabd8803.
- [123] Weihao Yuan, Kaiyu Hang, Haoran Song, Danica Kragic, Michael Y Wang, and Johannes A Stork. "Reinforcement learning in topology-based representation for human body movement with whole arm manipulation." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2153–2160.
- [124] Dmitry Zarubin, Vladimir Ivan, Marc Toussaint, Taku Komura, and Sethu Vijayakumar. "Hierarchical motion planning in topological representations." In: *Proceedings of Robotics: Science and Systems VIII* (2012).
- [125] Z. Zhong. *Finite Element Procedures for Contact-Impact Problems*. Oxford U.P., 1993.
- [126] Xianjin Zhu, Rik Sarkar, and Jie Gao. "Topological Data Processing for Distributed Sensor Networks with Morse-Smale Decomposition." In: *IEEE INFOCOM 2009* (2009), pp. 2911–2915.
- [127] Olgierd C. Zienkiewicz, Robert L. Taylor, and Jizhong Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, Sixth Edition*. Butterworth-Heinemann, May 2005. ISBN: 0750663200.