Reverse-Engineering Fashion Products : From a single-view Sketch to a 3D Model

Amélie Fondevilla

▶ **To cite this version:**

Amélie Fondevilla. Reverse-Engineering Fashion Products : From a single-view Sketch to a 3D Model. Artificial Intelligence [cs.AI]. Université Grenoble Alpes, 2019. English. NNT : 2019GREAM070 . tel-02908437

HAL Id: tel-02908437
https://theses.hal.science/tel-02908437

Submitted on 29 Jul 2020

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

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

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Amélie FONDEVILLA

Thèse dirigée par **Stefanie HAHMANN**
et codirigée par **Damien ROHMER**

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Modélisation 3D d'objets cousus à partir d'un unique croquis

Reverse-Engineering Fashion Products: From a single-view Sketch to a 3D Model

Thèse soutenue publiquement le 18/12/2019
devant le jury composé de :

Madame Stefanie Hahmann
Professeur, Grenoble INP, Directrice de thèse

Monsieur Damien Rohmer
Professeur, École Polytechnique, IP Paris, Co-directeur de thèse

Monsieur Loïc Barthe
Professeur, Université Paul Sabatier , Rapporteur

Monsieur Frédéric Cordier
Maître de Conférences (HDR), Université de Haute-Alsace, Rapporteur

Madame Géraldine Morin, *présidente du jury*
Professeur, Toulouse INP, Enseeiht Toulouse, Examinatrice

Monsieur Marc Daniel
Professeur, Université d'Aix-Marseille, Examinateur

Monsieur Adrien Bousseau
Chargé de Recherche, Inria, Université Côte d'Azur, Invité

Remerciements

Le chemin vers l'aboutissement de cette thèse fut long et sinueux. Aussi, de nombreuses personnes ont contribué à en faciliter la trajectoire, en augmenter la cadence, ou encore à rendre le trajet plus agréable. C'est avec grand plaisir que je prends ici un moment pour remercier ces personnes, qui, en plus de m'avoir apporté aide et soutien, ont laissé une trace indélébile dans ma vie.

En premier lieu, merci aux membres du jury, à Frédéric Cordier et Loïc Barthe pour leur lecture et rapports sur ce manuscrit, et aux examinateurs Géraldine Morin et Marc Daniel pour leur présence à la soutenance ainsi que leurs questions.
Je remercie ma directrice de thèse, Stefanie, pour ses précieux conseils, pour sa bienveillance et combativité à toute épreuve. Merci d'avoir toujours été là pour m'aider et me soutenir. Merci aussi à mon co-directeur Damien, qui, en plus de son soutien technique, a su par sa bonne humeur et son humour, me rassurer et me pousser à avancer pendant les moments de doute. De plus, je remercie Adrien et Marie-Paule, pour toutes leurs contributions apportées au travail de cette thèse, pour leur présence régulière aux réunions qui ont permis de le structurer.
Merci à Laurence pour ses contributions graphiques, sa créativité et sa patience. Un grand merci à toutes celles et ceux que cette thèse m'a apporté de rencontrer, et autour desquel.le.s j'ai appris et échangé énormément. L'équipe Imagine, dans toute sa fougue, sa créativité et son amour pour le fromage. Merci aussi à l'équipe GeoViC, pour m'avoir accueillie au sein du LIX pendant trois mois.

Un dernier paragraphe pour remercier mes proches, qui m'ont soutenue dans cette étape de ma vie et les difficultés qu'elle a pu apporter, bien qu'il n'aient pas forcément bien saisi ce que je faisais de mes journées. A tous mes ami.e.s, vivant à Grenoble ou non, mais toujours disponibles pour m'écouter me plaindre autour d'un verre.
Lo mot de la fin que s'en va entà la mia familha, los de qui cau. Un gran mercès entà tot çò qu'avetz heit tà jo.

Contents

1 Introduction **5**
- 1.1 Garment design . . . 5
- 1.2 Sketch and Image-based Modeling . . . 7
- 1.3 Style transfer . . . 8
- 1.4 Contributions . . . 10

2 State of the Art **13**
- 2.1 Design of developable surfaces . . . 13
  - 2.1.1 Mathematical background . . . 14
    - Developable Gaussian curvature . . . 14
    - A special case of ruled surfaces . . . 15
  - 2.1.2 Design of general developable surfaces . . . 16
    - Discrete developable surfaces . . . 17
    - Developable surfaces from boundaries . . . 18
  - 2.1.3 Garment design . . . 19
    - Cloth manufacturing . . . 20
    - Traditional approaches for virtual modeling . . . 21
- 2.2 Shape synthesis from 2D inputs . . . 22
  - 2.2.1 Perception of contour lines . . . 22
    - Contour lines in a drawing . . . 23
    - Interpretation of 3D shapes through contour lines . . . 25
  - 2.2.2 Sketch-based modeling . . . 26
    - Optimization-based reconstruction . . . 26
    - The Symmetry assumption . . . 27
  - 2.2.3 Modeling sewed objects from 2D inputs . . . 30
    - Geometric modeling from sketches . . . 30
    - Image-based and data-driven approaches . . . 31
- 2.3 Transferring styles and shapes . . . 33
  - 2.3.1 Example-based transfer . . . 33
    - Transfer by analogy . . . 33
    - Feature based style transfer . . . 34
  - 2.3.2 Style transfer of garments . . . 35
    - Transferring wrinkles . . . 35
    - Transferring garments to different morphologies . . . 36
- 2.4 Conclusion . . . 37

3 Modeling symmetric sewed objects using a single photo 39
3.1 Input image and annotations . . . 42
3.1.1 Hypothesis on the image . . . 42
Generic viewpoint . . . 42
Orthographic projection . . . 43
Global mirror-symmetry . . . 43
3.1.2 User annotations . . . 43
Contour annotations . . . 43
Symmetry annotations . . . 43
3.2 2D curve analysis and rulings extraction . . . 44
3.2.1 Extracting 2D patches . . . 44
3.2.2 Extracting symmetry . . . 45
Computing the plane of symmetry . . . 45
Matching points in symmetric curves . . . 47
3.3 Extracting rulings in a patch . . . 48
3.3.1 Silhouettes as projection of rulings . . . 48
Observations . . . 48
Generalization . . . 50
Conclusion . . . 52
3.3.2 Rulings propagation inside the patch . . . 52
Observations . . . 52
Dynamic Time warping algorithm . . . 55
3.4 3D contours optimization . . . 57
3.4.1 Energy formulation . . . 57
Regularization energies . . . 58
Symmetry . . . 60
Developability . . . 60
3.4.2 Optimization . . . 61
3.5 Developable surface generation . . . 61
Patches symmetrization . . . 61
Surface triangulation . . . 63
3.6 Results and Validation . . . 63
3.6.1 Results . . . 63
Synthetic models . . . 63
Real-world objects . . . 63
Influence of symmetry constraint . . . 64
3.6.2 Evaluation . . . 64
Metrics . . . 64
Analysis of the developability constraint . . . 65
Failure cases . . . 68
3.7 Conclusion and Discussion . . . 69

4 Sketch-based modeling of tubular folds 71
4.1 Representing tubular folds . . . 72
4.1.1 Tubular folds in fashion illustration . . . 73
Inner silhouettes . . . 73
Hemline . . . 75

4.1.2 Modeling folds in 3D . . . 75
Garment 3D representation . . . 75
Influence of perspective in sketched hemlines . . . 76
4.2 Sketch-based modeling of folded hemline . . . 78
4.2.1 Reconstruction algorithm . . . 79
Plane estimation . . . 80
Reverse perspective . . . 82
4.2.2 Flattening the fold curve . . . 82
Normalized polar representation . . . 84
Arc-length dependant representation . . . 84
4.2.3 Completing the occluded parts of the hemline . . . 85
Self-occlusions of the hemline . . . 85
The Hidden side of the hemline . . . 89
4.3 Transferring folds to existing geometries . . . 89
4.3.1 Applying a fold curve to a smooth surface . . . 89
Folding a boundary curve . . . 90
Fold propagation . . . 90
4.3.2 Fold analogies . . . 91
Transfer strategies . . . 91
Comparison . . . 92
4.4 Results and Evaluation . . . 93
4.4.1 Evaluation . . . 93
Plane estimation . . . 93
Fold reconstruction . . . 94
4.4.2 Results . . . 95
Completion of discontinuous hemlines . . . 96
Transferring folds distributions . . . 97
4.5 Conclusion . . . 100

5 Dressing arbitrary characters using a single sketch 101
5.1 Garment representations . . . 103
5.1.1 Style in garment sketches . . . 103
Garment drawings . . . 104
Style criteria . . . 104
5.1.2 User Inputs . . . 107
5.1.3 3D Primitive surface . . . 109
5.2 Positioning a garment patch using a sketch . . . 111
5.2.1 Proportion-preserving garment positioning . . . 111
Transferring location . . . 111
Transferring orientation . . . 113
5.2.2 Initial garment surface . . . 114
Computing boundary curves . . . 114
Straight primitive surface . . . 115
5.3 Transfer shape and fit . . . 116
5.3.1 Shape and fit . . . 116
5.3.2 Solving collisions with the body . . . 119
5.3.3 Adding tubular folds . . . 122

5.4 Results and discussion 123
5.4.1 Results 124
5.4.2 Technical details 127
Execution time 127
Complexity of the output models 128
5.4.3 Limitations 129
Junction areas 129
Adaptation to extreme poses 130
Fully tight garments 130
5.5 Conclusion 131

6 Conclusion **133**

Bibliography **137**

Résumé

Créer efficacement du contenu virtuel 3D est une problématique importante dans le domaine de l'Informatique Graphique. Cette thèse traite le sujet de la modélisation 3D d'objets cousus, tels que des vêtements, chaussures, ou accessoires à partir de croquis 2D. Des approches existantes parviennent à modéliser ces objets en utilisant plusieurs vues en entrée ou une interface de modélisation directement liée au personnage à habiller. Nous proposons ici d'utiliser un croquis unique annoté pour créer du contenu 3D satisfaisant des contraintes géométriques spécifiques aux tissus, comme la développabilité, ou l'apparition et la distribution de plis. Notre but est d'exploiter l'expressivité du dessin pour guider la modélisation d'objets plausibles, en utilisant des connaissances géométriques a priori.

Dans un premier temps, nous présentons une méthode de reconstruction 3D d'objets symétriques et développables par morceaux, à partir d'une unique photo annotée. Nous exploitons l'hypothèse de symétrie et les propriétés géométriques liées aux surfaces développables lisses, afin de proposer un système capable de donner de la profondeur aux courbes représentant silhouettes, bords et coutures dessinées sur la photo. Notre approche retrouve aussi des informations topologiques sur l'objet, permettant de calculer la forme 2D des patrons de tissus associés à chaque morceau de surface développable, information nécessaire pour la fabrication réelle de l'objet.

Dans un second temps, nous nous intéressons à la modélisation de vêtements virtuels, qui sont des objets cousus pouvant contenir des plis. La majorité des approches de modélisation de vêtement basées croquis reconstruisent en 3D des courbes de bord, silhouette et plis annotées dans une vue 2D du personnage à habiller. Notre approche, en revanche, utilise un unique croquis, et en extrait des caractéristiques liées aux style du vêtement : proportions, zones moulantes, silhouettes et plis. En particulier, nous proposons une méthode pour extraire une représentation générique de plis à partir du croquis, même dans le cas où de plis profonds qui entrainerait une représentation partielle de la courbe de bord sur le dessin. Nous synthétisons ensuite un vêtement 3D habillant un personnage virtuel cible et possédant les mêmes caractéristiques de style que celles extraites du dessin. Cette synthèse s'adapte aux cas où le personnage cible est dans une pose et/ou a une morphologie différente de celui représenté sur le dessin.

Abstract

The efficient creation of virtual 3D content is a major issue in Computer Graphics industry and research. This thesis addresses the modeling of sewed 3D objects such as clothes, shoes or accessories from 2D sketched inputs. While existing related approaches are either based on multi-view inputs or on character dependent interfaces, we propose in this thesis to start from an annotated single-view input in order to create 3D content satisfying geometric constraints that are specific to fabric, such as developability, or fold appearance and distribution. Our goal is to exploit the expressiveness user-drawn sketches to guide the modeling of plausible objects, using a priori geometric knowledge.

We first present a single-view reconstruction approach for symmetric piece-wise developable objects from an annotated photo. Using the assumption of mirror-symmetry along with properties of smooth developable surfaces, we propose a system able to lift in 3 the silhouette, borders and seams drawn on the picture and reconstruct the surface of the object. Our method also provides topology information on the object, allowing the computation of 2D patterns for each piece of developable surface, which is necessary for manufacturing.

While most of existing sketch-based modeling methods for garments aim at reconstructing border lines drawn on top of a view of the 3D virtual mannequin to be dressed, our approach uses a single 2D sketch as input. Our method first extracts generic features relative to the style of the sketched garment expressed as garment proportions, tight areas, silhouette shapes and folds. In particular, we propose a dedicated approach to extract robustly folds characteristics, even in the case of garment with deep folds leading to partially occluded garment borders. In a second step, we synthesize on a target virtual character a 3D garment surface exhibiting similar style, while adapting to the target pose and morphology that may be drastically different from the one depicted the sketch.

Chapter 1

Introduction

In the past few years, virtual 3D content has taken an increasing space in our live, in the fields of industry, entertainment, and education. Technologies to create and manipulate 3D content have democratized, whether it is for fabrication with personal 3D printers, for animation with free open-source softwares, such as Blender, or entertainment with the blast of virtual reality devices.

Virtual environments are able to represent a wide diversity of content, and have been proven of great interest for very different tasks. They allow the visualization and manipulation of inaccessible content, such as anatomy models, infinitely small ecosystems but also infinitely large ones, such as galaxies. It is also of great interest to model complex theories, such as fluid mechanics, and enforce geometric and physic constraints in design systems, for example in the automobile industry. Not only virtual environments can be used to model, deform and predict what exists around us, it can also be used to imagine, model and explore new shapes, motions, and visual effects.

1.1 Garment design

The design of intuitive tools for the creation of 3D content remains a challenging task. In particular, the creation of virtual garments is a subject of great interest with various applications. One example of application is the cloth manufacturing industry, where Computer-Aided-Design softwares are used to create virtual models in the scope of producing real garments. The users of these technologies are professionals of the fashion industry, and the main challenge for the interface is to account for specific technical and physical constraints imposed by the manufacturing process. Another field of application is entertainment, with animation movies and video games creation, in which one important task is the modeling and animation of garments that fit existing virtual characters. In this case, users are graphic artists, who may not have any expertise in the fashion field. Here one of the main challenges is to design sufficiently intuitive tool,

so that the artist can express his/her creativity, while creating garment surfaces which can be animated on the virtual characters.

The shape of a garment is conditionned by a multitude of factors of different nature :

Developability Garments are made of *developable surfaces*, which means that the shape of any cloth is the result of an isometric deformation of a planar surface. To put it differently, cloth can be flattened onto a plane without tearing, stretching or compressing the surface. This condition on the geometry of the surface prevents it to be *doubly-curved*, which implies that it can not locally fit any spheres. In practice, most of the clothes are not perfectly developable, and may be more or less extensible, depending on the fabric they are made of.

Fabric Cloth physics can be described by a list of numerous parameters, that are related to their fabric. These parameters, describing the mass, resistance to stretching, tearing, compressing, or folding of the cloth, are influencing the shape and location of folds on the surface. They may also determine how well the garment fits the character's body in tight areas.

Mannequin Last but not least, the body of the dressed character is also influencing the shape of the garment surface. In particular, the shape of the garment is conditionned by the size and morphology of the character in tight areas, and by the pose taken by the character in loose area.

One of the challenge in garment design is to provide a tool allowing the user to create and modify intuitively the garment surface while accounting for the specific constraints listed above. It is particularly true for the constraint of developability, which can be expressed in many ways. We contribute to this line of research by formulating a linear constraint on the boundary curves of a surface patch so that it remains developable during processing. Our approach, described in Chapter 3 exploits the fact that developable surfaces are ruled surfaces with constant tangent plane along their rulings.

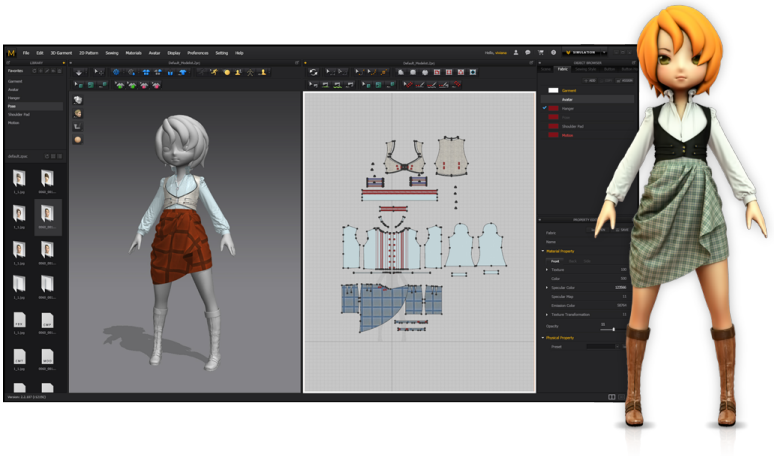


Illustration of the interface for creating virtual garments in Marvelous Designer

Due to the complexity of these various constraints, most garment design softwares such as Clo3D or Marvelous Designer, use a standard design pipeline inspired by the process of real cloth manufacturing. This process starts with the design of 2D pattern pieces, that are then triangulated into a 2D mesh, and assembled together in the 3D space. The user then needs to input several parameters to indicate the properties of the fabric, and run a cloth simulator based on physics laws.

One of the challenges implied by such method is the choice of the resolution of the garment mesh, which is determined before knowing what the garment will look like after simulation. If the mesh is not precise enough, it may not be able to represent the complexity of the garment, due to the tightness of the garment to complex parts of the body, or to a light fabric resulting into fine folds. On the other hand, increasing the resolution of mesh is at the cost of computational efficiency. In most cases, the optimal resolution is not uniform accross the garment. In Chapter 5, we propose an algorithm modeling 3D garments using a parametric primitive surface which adapts to the complexity of the garment which is modeled. The process starts with a basic initial surface, which is then subdivided when it encounters collisions with the character's body, or when folds are applied on it.

Another major drawback of the standard pipeline for garment modeling is the expertise it requires to design the 2D patterns and decide for the fabric's parameters. The shape of the resulting garment is then completely left to the cloth simulator. Another way of modeling 3D shapes without requiring any professional expertise is sketch-based modeling. Sketching is a widespread practice, for both 3D graphic artists and professionals of the fashion industry, designer using fashion illustration to convey new ideas and guide garment creation.
Can sketches and images ease the process of 3D garment modeling ?

1.2 Sketch and Image-based Modeling

Despite many advances in GPUs, touchscreens and virtual reality devices, providing intuitive interfaces to sketch surfaces directly in a virtual 3D environment still seems to be a challenging task. While most of the 3D sketching softwares, such as SketchUp, use 2D screen interfaces, some new systems, such as TiltBrush are proposing 3D sketching interfaces in immersive environments using virtual and augmented reality. While these immersive systems have the potential to improve fluidity of the creative gesture, it remains challenging for users to draw precise curves [AKA+17], and long sessions of 3D drawing can be physically demanding.

On the other hand, sketching with pencil and paper is one of the first known mean of human expression and communication. Despite the fact that an image only displays a partial representation of its subject, and despite the potential imprecision or lack of

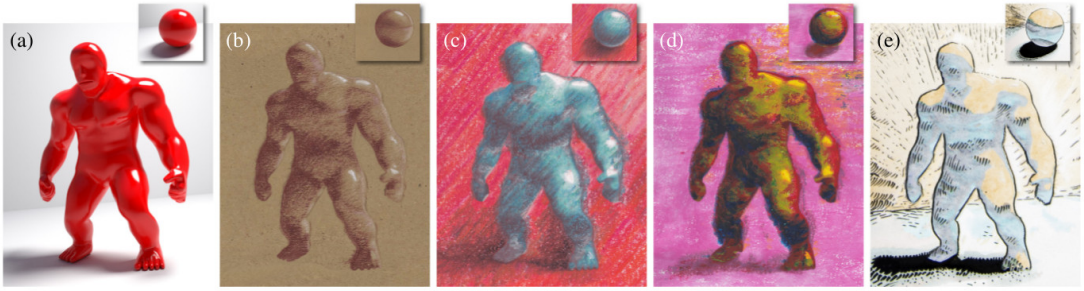
realism in sketches, one can usually imagine the 3D shape of the subject that an image is representing. This idea is generally shared by most human beings, independently of culture, nationality or education.

Reconstructing a 3D shape using a 2D view of it is however an ill-posed problem, because each point in a 2D image plane can be the projection of an infinite number of points in the 3D space. Sketch and image-based modeling techniques usually need additional a priori knowledge on the represented shape, whether it is given by geometric constraints, physics properties, or learned using examples.
We contribute to this line of research and propose in Chapter 3, a single-view reconstruction method for symmetric sewed object. In this work, we exploit the hypothesis of symmetry and developability of the surface that we want to reconstruct in order to lift a set of 2D sketched curves in 3D and generate a surface bounded by these curves.

One of the challenges that is inherent to single-view based modeling approaches is the reconstruction of the hidden parts of the shape. Because the input only represents one view of the object, a whole side of it is generally hidden. Moreover, some parts of the object can be occluded by the object itself.
In the context of garments, the latter observation is particularly true when the surface contains deep folds. In Chapter 4, we present an algorithm able to reconstruct the depth of planar curves representing folded boundaries of garment patches. We account to the case of discontinuous curves, that were partly occluded due to self-occlusions of the surface with a completion algorithm that infers the occluded parts of the curves based on the shape of the visible ones.

1.3 Style transfer

In addition to the challenges implied by 2D to 3D reconstruction, 3D modeling using sketches induces specific challenges, inherent to the expressiveness, and sometimes abstraction of drawings. This is particularly true for fashion drawings, which often represent exagerrated geometry, imprecise outlines, or characters with unrealistic morphologies. Using such stylized inputs thus adds further challenges to the task of reconstruction.
Creating shapes through stylized sketches can be formulated via the extraction of style properties from an example to transfer it on a target. The original idea of style transfer was formulated by Hertzmann et al. [HJO+01] in the context of image analogies. The idea is to use as inputs a source image A, a stylized version of the same image A', and a target non-stylized image B. Then the method of style transfer produces an output stylized image, representing the content of the image B in the same style than A'.



Examples of artistic renderings that can be generated using StyLit [FJL+16]

This analogy paradigm can be formulated for various applications, and can be used for the design of intuitive tools for creation of original content. Let us illustrate that statement with the example of StyLit [FJL+16]. The goal of this method is to create efficiently a stylized rendering of any 3D object. There exist a wide variety of styles of renderings, and defining a parametrization of rendering styles is a very challenging task. They overcome this issue using an approach of creation by analogy. In their system, the user designs the rendering style by drawing an basic examplar of it : a sphere lit from a unique light source. Because the geometry of the sphere and the illumination of the scene are known, this input is sufficient to extract texture patches and transfer them to renderings of objects with more complex geometries.
We apply this concept of creation by analogy in the context of the design of tubular folds on a surface. Our algorithm, described in Chapter 4, uses the boundary curve of a folded garment, along with its smooth non-folded counterpart to apply similar folds on another non-folded surface. We propose two different strategies to solve this issue : one preserving the number of folds in the surface, and another preserving the folds' shape.

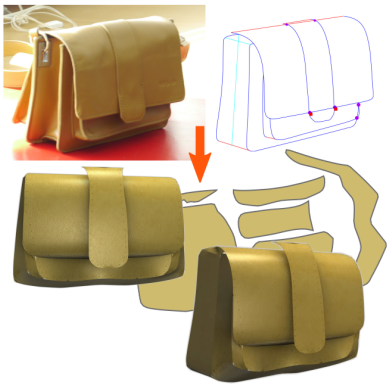
In alternative versions of the analogy paradigm, the non-stylized source object A is not accessible. In these cases, one wants to extract from the stylized source A' the features that are representative of its *style*, and need to be transferred to B, and the features relative to its *functionality*, which are not affected by the transfer.
One applicative example of such paradigm is the functionality-preserving style transfer of furnitures [LKWS16]. The main idea is to apply the style of a piece of furniture on another one which has different functionality, for example applying the style of a chair to a couch. In this work, they define functionality of the object as its rough shape and topological arrangement, while style is encoded via geometric details on the object's surface.
In the scope of garment modeling, an important challenge is to transfer garments to characters with different morphologies. The input is a source garment worn by one character, and the goal is to output a garment that fits a different character and that looks like the source garment. The problem is to determine what features of the source garment surface need to be preserved during transfer, and what should be adapted to the target. Brouet et al. [BSBC12] proposed a geometric definition of the style of a garment, in the context of 3D-to-3D style preserving transfer. We build on this definition

to propose a 2D-to-3D style preserving garment transfer. Our algorithm, described in Chapter 5 extracts style features of garment depicted with a single 2D input sketch and synthesizes a 3D garment surface with similar style adapted to an arbitrary target character.

1.4 Contributions

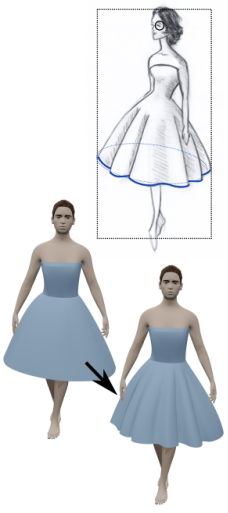
This thesis addresses the issue of synthesizing 3D sewed objects using a single 2D input. We propose three main scientific contributions, which are described in three different chapter of this thesis.

Chapter 3: Modeling symmetric sewed objects using a single photo



In this work, we propose a single-view reconstruction system to model piecewise C^2 developable surface that are symmetrical, using a single annotated photo. We develop an optimization system of geometric constraints to lift the border, silhouette and seam curves of the sewed object depicted in the 2D input into a network of 3D boundaries. We exploit both the hypothesis of global-symmetry and developability directly in our optimization system by extracting rulings and symmetric points from the 2D input curves and annotations. We show, in particular, a property of developable surface that can be applied in order to preserve developability during reconstruction. We then create a surface interpolating those 3D boundary curves.

Chapter 4: Sketch-based modeling of tubular folds



We then study garment sketching, and in particular sketches representing tubular folds in garments. Tubular folds are folds induced by the effect of gravity on the garment. We propose an algorithm to interpret a 2D sketch of the folded boundary of a garment, and synthesize a shape-independent fold representation that can be transferred to another garment surface. We use a few annotations in the input sketch to calibrate a virtual camera model and infer depth into the projected folded border curve. Our method adapts to the case of deep folds creating self occlusions in the garment surface, and leading to a discontinuous representation of the folded border curve. We then propose two

different strategies for transferring to an existing 3D garment : one preserving the number of folds depicted in the sketch, and another to preserve the shape and frequency of the folds.

Chapter 5: Dressing arbitrary characters using a single sketch.

Finally, we tackle the issue of style preserving garment synthesis and transfer using a stylized fashion sketch. Our algorithm uses as input an annotated sketch representing the contour curves of a garment along with the 2D skeleton of the dressed character. We then define some criteria for style of garments and show how to extract style features of a garment from a 2D sketch. We use these features to synthesize a 3D garment, having the same style as the garment drawn in the sketch, while suiting properly a target character of arbitrary size, pose, and morphology. :

Chapter 2

State of the Art

Both the topics of sketch-based modeling techniques, and design of garment and developable surfaces were widely discussed in literature. We report of most relevant work in this chapter.

In Section 2.1, we focus on the design of developable surfaces. After defining them mathematically (Section 2.1.1), we present some systems for designing such surfaces (Section 2.1.2). Then, we focus on the case of developable sewed objects and garments, present traditional techniques for modeling them, in real-life and in virtual environments.

In Section 2.2, we expose the main literature on reconstruction of 3D models from 2D inputs. We start by focusing on line drawings, explaining what they represent, and how they can interpreted (Section 2.2.1). Then, we present some of the existing sketch-based modeling methods for general shapes (Section 2.2.2). Finally, we refocus our study on garments and piecewise developable objects, and present the image- and sketch-based modeling techniques dedicated to them (Section 2.2.3).

This thesis is not only concerned with reconstruction, but also with extracting and using the artistic expressiveness of drawings to guide our synthesis. Therefore, we dedicate Section 2.3 to style transfer techniques for 3D modeling. After presenting the original analogy paradigm and its translation to various applications (Section 2.3.1), we study the case of style transfer to developable objects (Section 2.3.2).

2.1 Design of developable surfaces

We call developable surface a surface that can be flattened onto a plane without distortion, meaning without stretching, tearing or compressing. This is the case of objects made of developable materials, such as paper, metal sheets, and cloth to some extend. Design of developable surfaces is a widely discussed topic in Computer Aided Design and Computer Graphics.

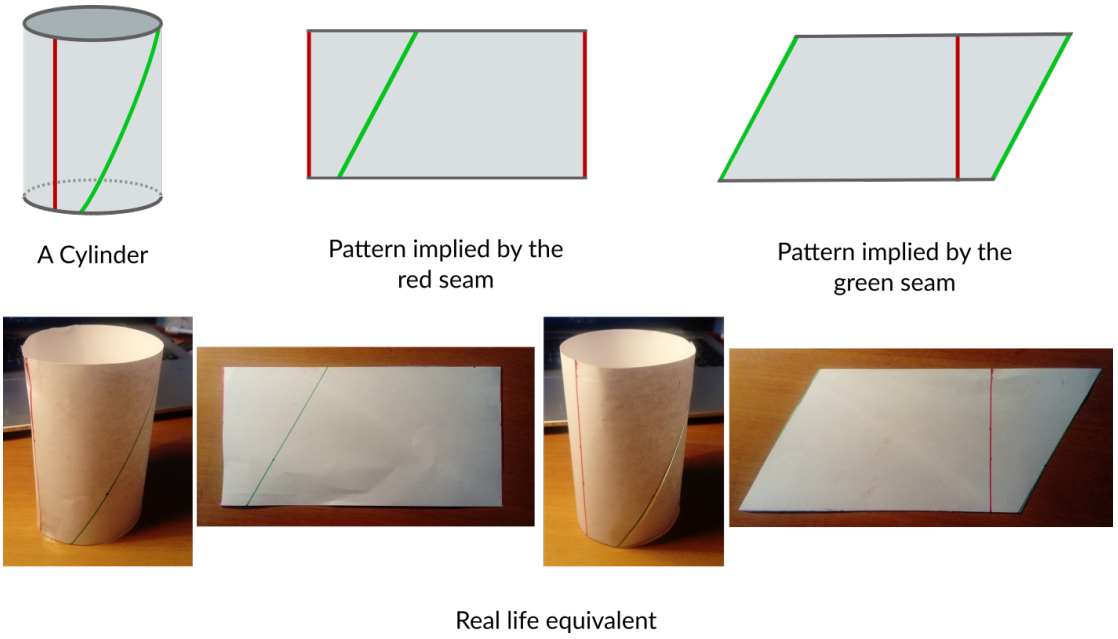


FIGURE 2.1: A Cylinder is a classical case of Developable Surface. Defining a seam in the surface determines the shape of the 2D pattern associated to the surface.

We first focus on presenting the mathematical definition and properties of such surfaces. Note that many of the theories presented in this section were discussed in depth in [Car76].

2.1.1 Mathematical background

Any developable surface can be associated to a 2D geometry, called pattern. In most cases, 3D geometric objects are not strictly developable. They may become developable once we add seams to them. Let's take the example of a continuous cylinder. There is no way to unfold it onto a planar surface without distortion. If we cut the cylinder along any curve linking one border of the cylinder to the other, then the flattening process becomes possible, see Figure 2.1. Different mathematical formulations for developability exist, some are expressed locally, like the Zero Gaussian curvature condition, others are expressed globally, like the *ruled surface* definition.

Developable Gaussian curvature Let S be a smooth C^2 regular surface. At each point $p \in S$, we can evaluate a tangent plane $T_p(S)$. For any vector $v \in T_p(S)$, we define the *normal section* C of S at p along v as the intersection of S with the plane containing v and $N(p)$, the normal vector to S at p. The curvature of C at p is called the normal curvature of S at p along v.

All normal curvature values at p can be computed from its two extremas k_1, k_2, called the *principal curvatures* of S at p. The directions associated to these extremas are called principal directions, and they form an orthonormal basis of $T_p(S)$. In Figure 2.2, we

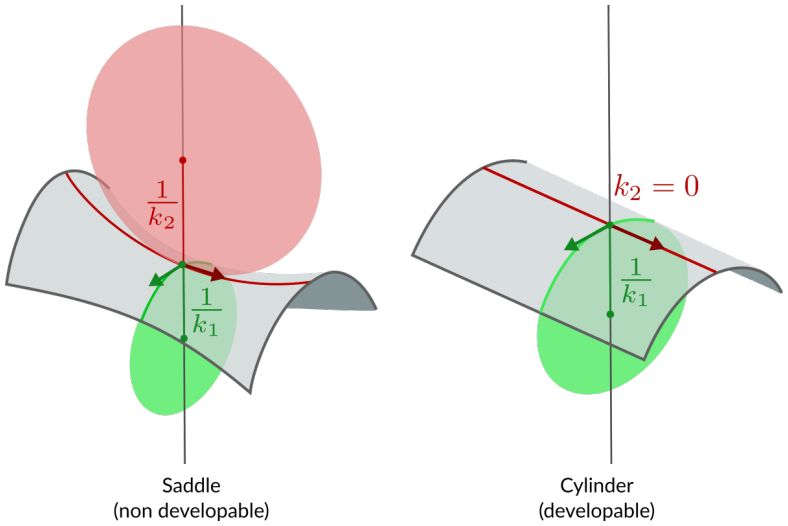


FIGURE 2.2: Illustration of the principal curvatures k_1, k_2 on two examples. The red and green curve represent respectively the normal sections associated to the minimal and the maximal directions, which are displayed with an arrow. We also represent the osculating circle of the normal sections, of radius $\frac{1}{k}$, with the same color code.

show an example of principal curvatures on two different surfaces and corresponding normal sections (green,red).

The Gaussian curvature K is defined as the product of both principal curvatures.

$$K = k_1 \cdot k_2 \tag{2.1}$$

This measure, representative of the local behavior of the surface, distinguishes elliptic points ($K > 0$), hyperbolic points ($K < 0$) and parabolic points ($K = 0$), with the specific case of planar points, for which all normal curvatures are 0.

Definition 2.1. A surface is developable if and only if at each point of the surface, the Gaussian curvature is zero. We also say that the surface is *singly-curved*.

A special case of ruled surfaces Developability can also be formulated with definitions accounting for the global shape of the surface. One of the most popular is the ruling configuration.

A ruled surface is a surface that can be obtained by moving a straight 3D line continuously in space. It can be parametrized with an equation of the form:

$$X(t, u) = \alpha(t) + u\omega(t) \ , \ t \in I \ , \ u \in J \tag{2.2}$$

where $I, J \subset \mathbb{R}$ and α, ω are functions assigning to any $t \in I$ respectively a point and a vector in $\mathbb{R}^3$.

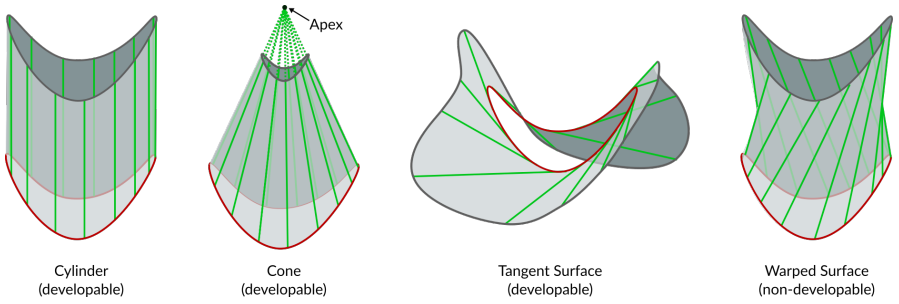


FIGURE 2.3: Different cases of ruled surface. Rulings are displayed with green lines, and a directrix is displayed in red. A ruled surface is developable if and only if it has a constant tangent plane along its rulings, which is not the case for the warped surface.

The lines $L_t : u \mapsto \alpha(t)+u\omega(t)$ are called *rulings* of the surface, and the curve $\{\alpha(t)|t \in I\}$ is called a *directrix* of the surface.

Definition 2.2. A C^2 developable surface is a ruled surface of constant tangent plane along its rulings.

Non-developable ruled surfaces are called warped surfaces. We distinguish three main, but non-exhaustive cases of developable surfaces (illustrated in Figure 2.3):

- Cylindrical surface, for which $\omega'(t) = 0$, $\forall t \in I$, $\omega \neq 0$
- Conical surfaces, of the form $X(t, u) = u \cdot \omega(t)$, $\omega \neq 0$, for which all rulings meet at a singular points, called *apex*
- Tangent surface of a space C^2 curve, of the form $X(t, u) = \alpha(t) + u\alpha'(t)$, $\forall t \in I, \forall u \in J$.

More generally, a developable surface is a union of pieces of cylindrical, conical and tangent surfaces.

2.1.2 Design of general developable surfaces

Design of virtual developable surfaces accounts for several challenges. First, the continuous definition of developability needs to be adapted to suit discrete representations of surfaces. Second, developability is highly constraining the geometry of a surface, and the design of a shape modeling tool for developable surfaces needs to be flexible enough to easily create and manipulate the surface, while ensuring that the surface remains developable.

In this section, we first review some of the proposed definitions for discrete developable surfaces, and how they were used in editing interfaces and approximation algorithm.

FIGURE 2.4: Architectural design made using subdivison of conical meshes, which is a specific case of PQ meshes. Illustration taken from [LPW+06].

Then, we will present some approaches dealing with the specific issue of inferring a developable surface using its boundary curves.

The specific case of garments and sewed objects will be treated in Section 2.1.3.

Discrete developable surfaces Discrete representation of surfaces, such as meshes, are at the core of the field of Computer Graphics and shape modeling. Accounting for developability in such discrete representations is a challenge, because the derivability of the surface itself is not well-defined. Different definitions of discrete developable surfaces were provided in litterature.

The most common measure for developability of meshes is the *angle defect*, introduced by Wang and Tang [WT04], and defined at each vertex q_i of the mesh by:

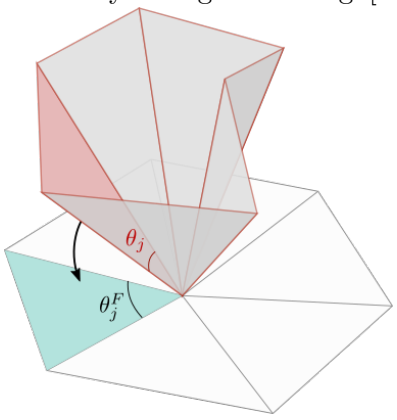


Illustration of the inner angles of a mesh and its flattened version (pattern), reproduced from [WT04].

$$\kappa_{q_i} = \frac{2\pi - \sum_j \theta_j}{\frac{1}{3}\sum_j A_j} \tag{2.3}$$

where θ_j are the inner angles incident to q_i (cf image on the left), and A_j are the corresponding triangle areas. The mesh is said locally flattenable at $q_i \iff \kappa_{q_i} = 0$. This function is a discrete approximation of the Gaussian curvature, and thus measures developpability locally, and does not account for a ruling representation of the surface. Wang and Tang [WT04] proposed an algorithm to optimize developability on a 3D mesh using this measure.

Another axis of research explored PQ meshes, which are composed of strips of planar quadrilateral faces (PQ strips). Such a strip can be trivially unfolded onto a plane without distortion, and is flattenable by construction. In particular, it contains a set of

rulings that are consistent in the whole mesh, directly given by the edges shared by the quad faces of the strip.
PQ meshes are used to approximate developable surfaces from an input quadrilateral mesh [LPW+06], and extended to curved developable surfaces with creases [KFC+08]. PQ meshes are especially suited for architecture design, allowing the creation of complex plausible shapes, cf Figure 2.4.

Solomon et al. [SVWG12] proposed an editing interface for origami-like shapes allowing the user to navigate through the space of perfectly developable surfaces. The surface is represented in both 3D and planar space, along with a set of rulings that are consistent all over the shape, and ensuring perfect developability of the surface. This representation is transparent to the user who manipulates boundary points of the surface. Following up this approach, Rabinovich et al [RHSH18] proposed a similar user interface to model developable surfaces. Their approach is based on discrete orthogonal geodesic nets. In general, discrete nets can be seen as a discrete version of a parametric representation of a surface. Discrete orthogonal geodesic nets are nets for which the incidental angles at each inner vertex of the discrete surface are equal. They proved that surfaces that can be parametrized with such orthogonal geodesic nets are developable surfaces. Their definition accounts for local developability, able to model flattenable, but not necessarily C^2 surfaces, because the consistency of the set of rulings of the surface is not guaranteed.

Recently, Stein et al [SGC18] proposed a new definition for discrete developable surfaces: they measure developability locally by finding, at each vertex of the surface, a direction along which the normal of the vertices remains constant. This definition both ensures local developability of the surface, measured by the angle defect, and provides a PQ mesh-like representation of the surface, which contains a consistent set of rulings. They also provide an applicative algorithm approximating a developable surface from an input mesh.

Other methods are using physics-based simulation to compute developable surfaces [NPO13, SRH+15]. These approaches, while working with discrete representations of surfaces, do not need a geometric definition for discrete developability, focusing on the physic laws that are modifying the surface shape.

Developable surfaces from boundaries Early work on design of developable surface was initiated for CAD and design of ship hull, automobile and aircrafts. Design of developable Bézier patches was at the core of this litterature. Some approaches proposed conditions so that two Bezier curves $A, B : I \mapsto \mathbb{R}^3$ form a developable patch, meaning a Bézier ribbon such that $\forall t \in I$, $[A(t)B(t)]$ is a ruling of the surface with constant tangent plane [CS02, CC04]. Some other presented algorithms to design developable surface using one Bezier curve as directrix [Aum03].

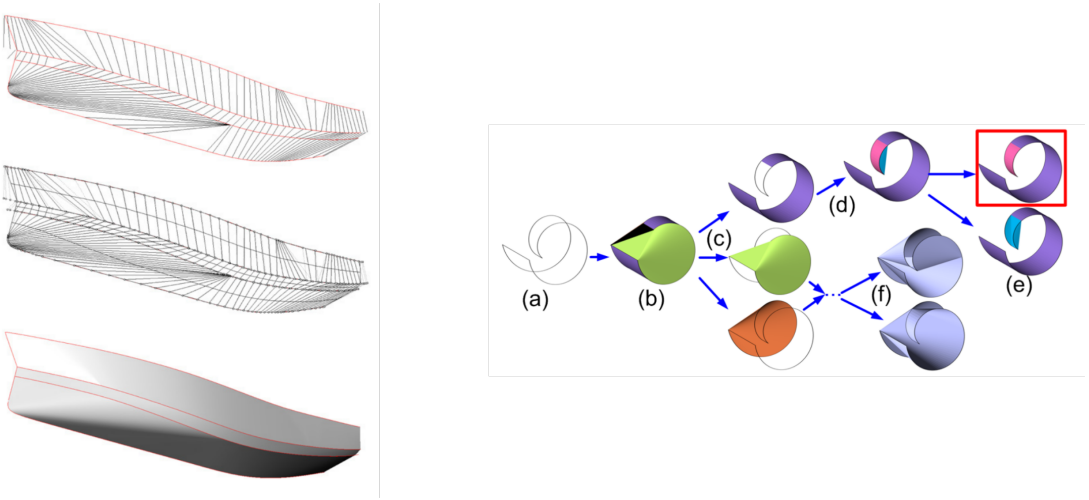
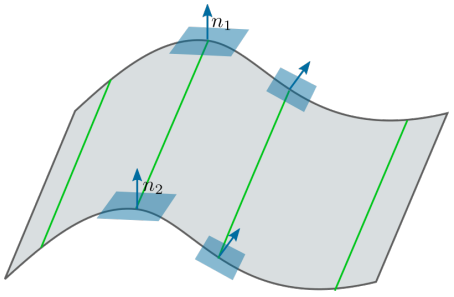


FIGURE 2.5: Two approaches for modeling developable surfaces from boundaries. On the left, a set of rulings is found to create the surface [PS07], on the right, the surface is generated by recursive computation and subdivision of the convex hull [RSW+07]

Perez et al. [PS07] extended these works to B-spline surfaces and proposed an algorithm to detect rulings as pairs of samples of the boundary curves (see an example of result in Figure 2.5-left).



The rulings are chosen to minimize the *warp angle* ϕ, defined as:

$$|n_1 \times n_2| = \sin\phi \tag{2.4}$$

where n_1, n_2 are the normals of the tangent planes at rulings extremities (cf image on the left).

Similar work was done for polygonal boundaries [Fre02], matching vertices to create rulings, using a discrete version of the warp angle, and was extended to isometry constraints by Rohmer et al. [RCHT11].

A different approach has been proposed by Rose et al. [RSW+07], exploiting the idea that a triangulation describes a developable surface if and only if each inner edge lies on the local convex hull formed by its neighbors. They propose a branch-and-bound algorithm, recursively computing and subdiving the convex hull of the boundary until a suitable triangulation is found (illustrated in Figure 2.5).

2.1.3 Garment design

For garment design, perfect developability is usually not the goal. The surface needs some elasticity to mimic the fabric while it is folded, or fits the body. The level of

FIGURE 2.6: Example of fashion sketches with different styles, and the technical drawings associated to the second sketch. Sources: left [NG09], right [Wat09]

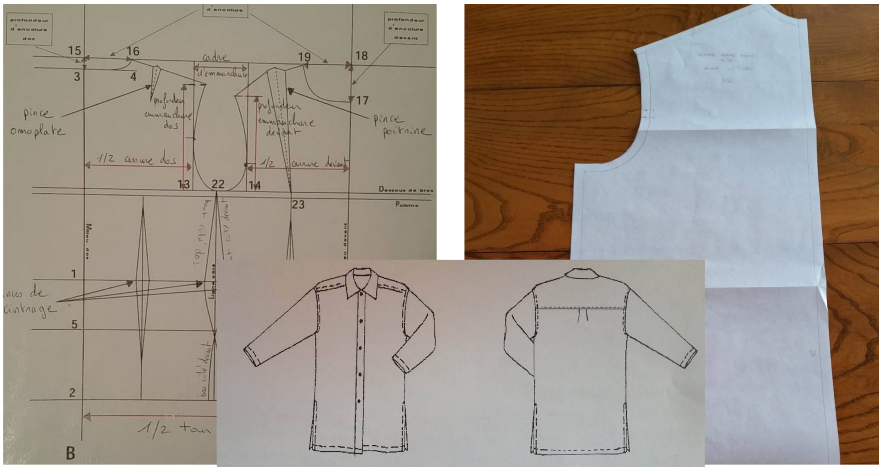


FIGURE 2.7: Example of pattern created from the technical drawing of a shirt.

elasticity depends on the material that is mimicked. We propose first to quickly review the process for real garment manufacturing, and then present methods mimicking this process for virtual garment modeling.

Cloth manufacturing The manufacturing process for real-life garments is composed of three steps, that may be iterated if need be.

First, the fashion designer, proposes a concept of garment using one or several sketches. Those sketches can have arbitrary expressiveness and level of abstraction, as displayed in Figure 2.6. It is then often completed with a technical drawing, to communicate more information about the balance, structure, and specifications of the garment [Wat09].

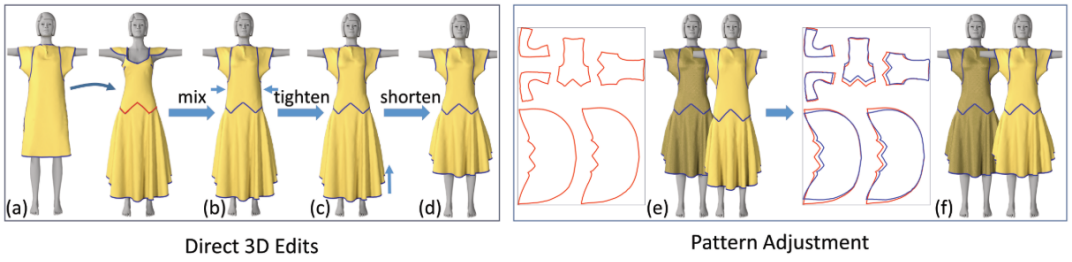


FIGURE 2.8: Example of editing interface for garment modeling from [BSK+16]. Transformations are induced by the user on the 3D surface (left), and the 2D patterns are adjusted accordingly (right).

The second step is done by the pattern maker, and consists in creating the 2D patterns associated to the drawing. An example of fashion pattern is displayed in Figure 2.7. Contours of 2D patterns are composed of straight and curved lines created with a specific tool called *french curve*. A geometrical study of this tool led to an interactive design system [Sin99], and an automatic method to decompose a 2D curve into parts of french curves segments [MS11]. However, as far as we know, there is no mathematical definition for the curves it can generate.

Finally, the dressmaker cuts the fabric according to the pattern pieces and sews them together. At this point, some adjustments can be done in the cloth so that it fits a person in particular.

Traditional approaches for virtual modeling The most common approach for virtual garment design mimicks this manufacturing process. The user designs 2D patterns, choses seam curves among its borders, and places the pattern pieces around the virtual body. The algorithms usually triangulate the patterns in the 2D space, and use numerical physics-based methods to compute the 3D shape of the garment around the character's body [VCMT05]. Berthouzoz et al. [BGK+13] proposed a method to automatically parse 2D patterns, assemble the pieces and create the 3D shape associated. 2D pattern based modeling approaches combined with physics simulation allows the creation of very realistic garments and are used in commercial softwares, such as Clo3D, Marvelous Designer, Optitex.

Umetani et al. [UKIG11] proposed a garment editing interface, were the user can modify either the patterns of the garment or the 3D surface itself on a virtual character. 3D to pattern edition of garment was refined by Bartle et al. [BSK+16], defining operations on garments such as mix, tighen, shorten, see an illustration in Figure 2.8.

All these approaches however require the design of 2D patterns corresponding to the desired garment. Physics-based simulation also depends on material parameters, which are often non intuitive for 3D artists.

Other approaches are based on images or sketches, which are a more intuitive representation of surfaces. These methods belong to the field of single-view based modeling, which is discussed in the next section.

2.2 Shape synthesis from 2D inputs

Images and sketches are a really mean of creation and communication. While understanding the 3D shape of an object represented with a picture is generally a trivial task for humans, it is a very challenging task for computers, mainly because an image only displays a 2D projection, and therefore partial representation of its subject. Interpretation of sketches presents additionnal challenges, dues to the potential imprecision or lack of realism they may represent. How could an algorithm recover the shape which is the most likely to be perceived by people, and therefore which was intended by the designer ?

In this section, we will describe previous work in the field of modeling for 2D inputs. First, we will present in Section 2.2.1 some insights on how we perceive full 3D shapes from such partial representation, then Section 2.2.2 will review some of the litterature related to sketch-based modeling of 3D shapes, and finally we will focus on sketch and image based modeling techniques for garments and developable objects in Section 2.2.3.

2.2.1 Perception of contour lines

Early computer vision work interprets brightness information in an image into geometric characteristics such as depth, orientation, reflectance, color. Perceptual experiments showed that the human vision is more sensitive to brightness discontinuities, occuring at surface discontinuities, than to the consistency of brightness distribution in smooth areas [BT81]. This would explain why we can communicate complex shapes using only line drawings, which represent these discontinuities, such as contour lines. In this section, we focus on the content and understanding of line drawings.

Interpretation of a 2D view of a smooth surface is an ill-posed problem. As stated by Barrow and Tenembaum in their study of the interpretations of line drawings [BT81], *"Since each point in an image determines only a ray in space and not a unique point, a two-dimensional line in the image could, in theory, correspond to a possible projection of an infinitude of three-dimensional space curves"*. A simple illustration of this inherent ambiguity is displayed in Figure 2.9: an ellipse drawn in a plane, could in theory be the projection of very different curves in space. However, some interpretations are more likely than others, especially if we have a priori knowledge on the 3D geometry of the object represented.

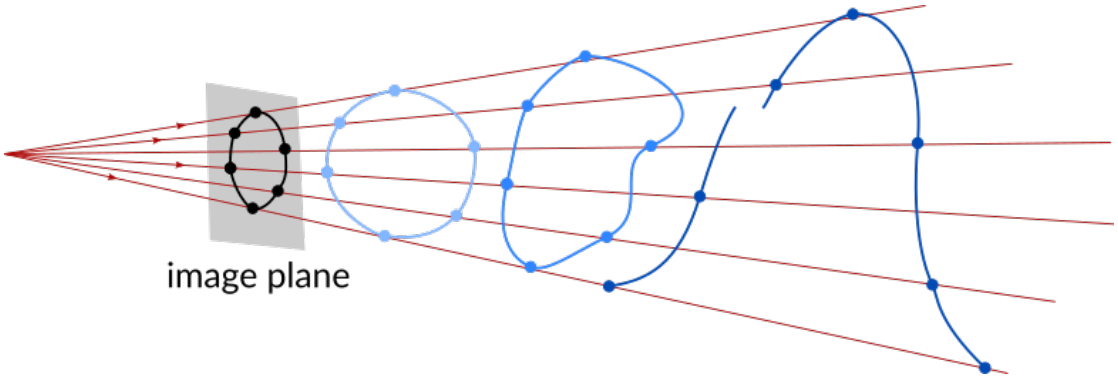


FIGURE 2.9: Image reproduced from [BT81]. If we follow the set of rays in red, all the 3D curves in blue would be projected onto the same 2D ellipse in the image plane.

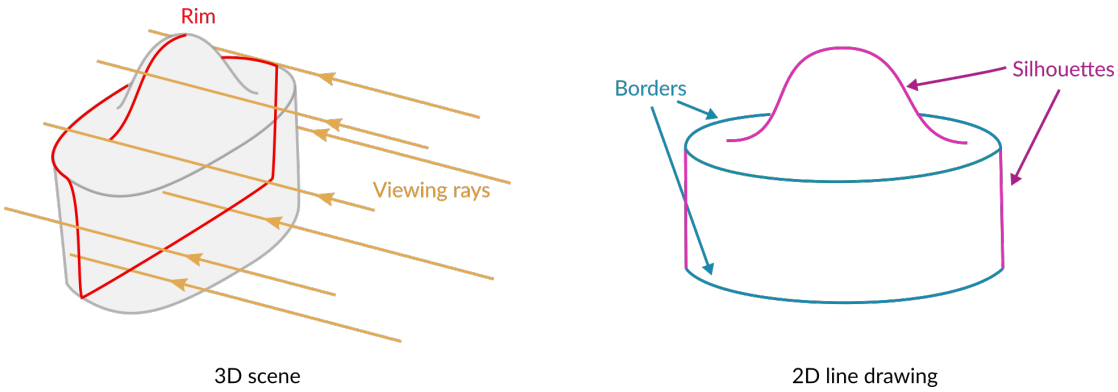


FIGURE 2.10: Illustration of a line drawing representing a piecewise smooth object seen from one viewpoint.

Contour lines in a drawing As previously stated, we focus on 2D drawings displaying lines of brightness discontinuity in the scene. Let's first consider a smooth object, meaning an object for which a unique tangent plane is defined at each point of its surface.

We assume that we see the object from a *generic viewpoint*, meaning such that a slight change in the view direction would not lead to significant changes in our understanding of the shape. A viewpoint which is not generic is called accidental. This definition is quite subjective, since it refers to our ability to perceive the shape. To be more specific, in a generic viewpoint, it is usually assumed that:

- straight lines in 2D correspond to straight lines in 3D
- intersections in 2D are actual intersections in 3D

Note that while the generic viewpoint assumption is relevant in single-view based modeling methods, in multi-view studies it is common practice to work with at least one front and one side view of the object, which are often accidental views [RDI10, JHR+15].

If we choose one viewpoint as a set of rays in space (which covers both perspective or orthographic camera models), then the curve on the object dividing its visible parts from

its non-visible parts in this viewpoint is called the *rim*. The rim can also be defined as the locus of points where the visual direction grazes the surface of the object [KVD82], and correspond to points of brightness discontinuity in the viewer's eye.

The *contour* is then defined as the projection of the rim in the image plane. In practice, some parts of the rim may be occluded, and only visible parts of the contour are represented in a drawing. Note that this contour representation is subjective and may vary from one artist to another.

If we consider piecewise smooth objects, meaning objects made of smooth surface patches glued together, brightness discontinuities can also occur at patches intersections. We will refer to the lines representing these discontinuities as contour lines as well.

Contour lines in a drawing are usually classified in two categories [BT81]:

- *silhouettes*, or extremal boundaries, where the surface normal turns smoothly away from the viewing direction
- *borders*, or discontinuity boundaries, where the surface terminates or intersects

See an illustrating example in Figure 2.10. Note that in this example, we consider the object as made of two patches of smooth surfaces assembled together (one cylindrical, and one with a spherical-like part), and thus consider as borders the curves of intersection between these two pieces.

Silhouettes have a significant meaning towards the object's local geometry, and were widely studied in a goal of surface reconstruction [WBCG09, RDI10, BVS16]. To each point in the silhouette corresponds a point in the rim. At each point of the rim, both the viewing ray and the tangent vector to the rim lie in the tangent plane of the point [CB92]. In general, these two vectors along with the 2D silhouette curve determine uniquely the normal of the surface at this point. It is not the case when the viewing direction is colinear to the tangent of the rim, which in practice would result in a degenerated silhouette curve represented by a point in the 2D space.

Moreover, as proven by Koenderink et al. [Koe84], the curvature of the silhouette curves is directly related to the Gaussian curvature of the surface at those points. In particular, convexities of the silhouette correspond to convexities of the surface, and concavities of the silhouette to saddle-shape surfaces. In the case of developable surfaces, where the Gaussian curvature is zero everywhere (cf Section 2.1.1), the silhouettes from any viewpoint are always straight lines corresponding to planar rims.

These geometric properties of contour lines are a good basis to understand how to interpret 2D line drawings. However, there is still some ambiguity to find the 3D shape represented, and additionnal perceptual cues are needed to formulate geometric constraints on the shape.

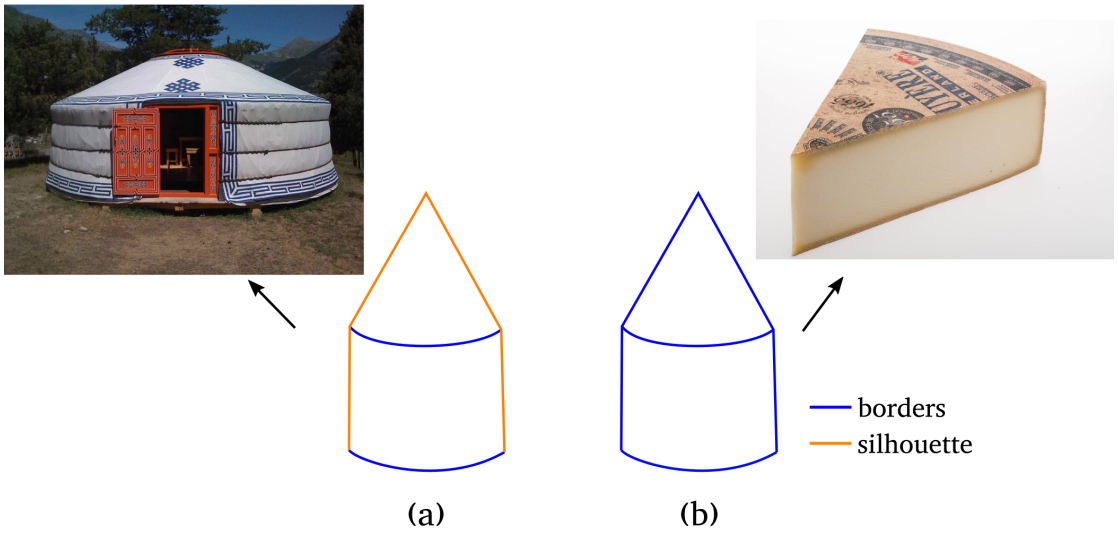


FIGURE 2.11: Ambiguity due to line classification. Different choices of classification would lead to the perception of different, but all plausible surfaces: a yurt (a) or a slice of cheese (b). Illustration reproduced from [BT81]

Interpretation of 3D shapes through contour lines Because silhouette and border curves in a drawing have different meanings for the 3D shape, interpretation of line drawings requires first to classify contours by those two categories. The problem of line classification is non-deterministic by nature: most of the time, any classification would be correct, and would lead to different surface, as in the example of Figure 2.11. However, some configurations are more likely to be perceived than others. There are two principal bases for classifying lines: local cues based on line junctions, and global cues based on similarities between curves. The description of those cues is explained in detail in [BT81], we only provide here an overview.

Line junctions are carrying a lot of information in the behavior at discontinuity and the topology of the shape. Standard junction classifiers are using catalogs of junction types and their possible interpretations [Cha79]. However, these classifiers may depend on subtle variations that may be difficult to distinguish in practice, cf Figure 2.11.

Global cues may help to get a more consistent classification. It seems that the human eye is sensitive to regularities such as parallelisms and symmetries. 2D lines that are perceived as parallel or symmetrical are therefore more likely to be of the same nature (parallel or symmetrical) in 3D.

Once the lines are classified, reconstruction of a surfaces involves different issues:

- compute a 3D representation of the boundaries,
- estimate normals for points inside the contour,
- infer occluded parts of the object.

Some further general cues may help to achieve reconstruction. For example, it is commonly stated that the 3D curve interpreted by the eye is the smoothest possible space curve that projects correctly in the drawing. Smoothness is here measured as uniformity of curvature, or planarity [BT81].

However, this knowledge about perception of contour line is generally not enough to recover a 3D shape, and we need additionnal information about the geometry of the object.

2.2.2 Sketch-based modeling

In this section, we will discuss some methods for modeling 3D shapes from sketches. This is a vaste topic, whose litterature was reviewed in several surveys [OSSJ09, KYZ13].

Several systems were proposed for interactive creation of 3D shapes from curves drawn in different planes/views [IIMT07, TZF04, BBS08, SWSJ06, NISA07, JC08], or by editing a 2D painting [ZDPSS02, BPCB08]. Sketch-based modeling was used for polygonal shapes, in particular in architecture [DTM96, JTC09], but also to model specific geometries such as characters [BCV+15, EBC+15], hair and helices [WBC07, FWTQ07, CCM14], or trees [WBCG09, OI03], to name a few. Recently, data-driven and template-based methods were proposed, using sketches to retrieve shapes in a database [XCF+13, XXM+13] or using neural networks to generate new shapes [HKYM16, NGDA+16].

We focus our study to some approaches that are more related to our subject. We first study 3D reconstruction performed through a global optimization based on geometric cues. We then discuss the importance and strength of exploiting symmetry for sketch-based modeling. The specific topic of sketch-based modeling of sewed objects and garments will be treated in Section 2.2.3.

Optimization-based reconstruction A popular approach to single-view reconstruction consists in complementing a re-projection error with various geometric constraints on the lines of the drawing. Such constraints are typically expressed as energy terms in an optimization that balances all concurrent goals.

One of the first of such algorithms was proposed by Lipson and Shpitalni [LS96], who focused on polyhedral shapes on which they impose parallelism, orthogonality, planarity and symmetry constraints. While planarity and parallelism act as regularity constraints, orthogonality and symmetry are enforcing the shape to be lifted in 3D. It is the balance between these two types of constraints that creates convincing 3D shapes such as the one in Figure 2.12.a.

This algorithm formed the basis of many subsequent systems, enabling rapid prototyping with finite-element simulation [TML09], in-context modeling of furniture [LSMI10], and

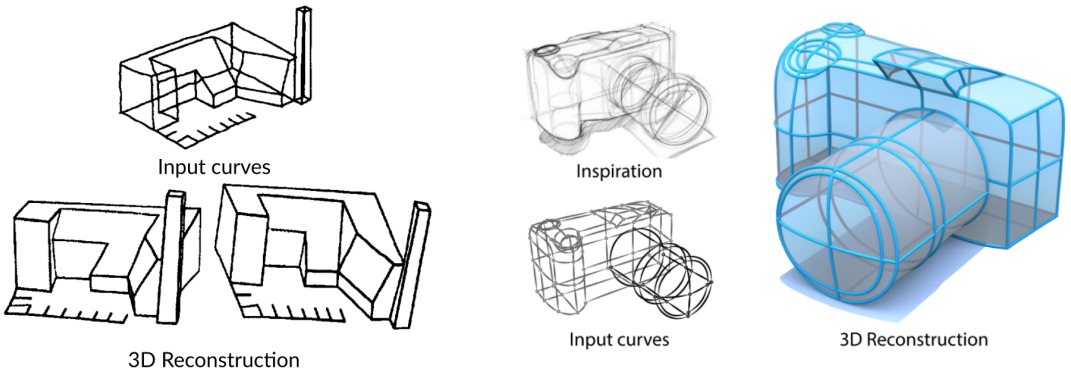


FIGURE 2.12: From polygonal shapes to smooth surfaces using cross sections. Orthogonality energy, that was necessary to lift polygonal shapes in 3D is used on the intersection of specific curves, called *cross sections* to give volume to smooth shapes from drawings. left: result from [LS96], right: result of [XCS+14]

3D reconstruction of complex models composed of multiple parts [YLT13]. However, the constraints used by these methods restrict them to objects dominated by flat, orthogonal faces.

Schmidt et al. [SKSK09] lift this restriction by reconstructing polyhedral shapes that serve as scaffolds to model curved surfaces, mimicking a traditional drawing technique used by professional designers. Similarly, Shao et al. [SBSS12] and Xu et al. [XCS+14] derive an orthogonality constraint from cross-section lines that designers draw to convey curvature directions on smooth surfaces. As we can see in the example of Figure 2.12.b, the resulting shapes show more complexity and smoothness.

Cross sections were also used in BendFields [IBB15] to esimate normals and curvature in a sketch, and shade the object accordingly. It inspired a new method for modeling free-form surfaces based on an annotated sketch called BendSketch [LPL+17]. In this approach, the sketch is annotated with curvature specifications, such as convex/concave principal curvature direction, sharp features, flat areas, and bending lines.

They compute principal curvature directions and values in the surface with an iterative optimization system which evaluates principal curvatures constrained by the annotations, and enforces them while preserving the positions of contours and silhouettes.

The Symmetry assumption Because many organic and manufactured shapes are made symmetrical, the symmetry assumption has been studied a lot in the field of sketch-based reconstruction. We only deal with the case of 3D mirror-symmetry, whether it is for the entire shape or parts of it. The assumption of mirror-symmetry for an object is a strong constraint on its global shape. The work made in [FMW02] demonstrates that a well-chosen single view of a mirror-symmetric object can provide as much information as two orthogonal views of the same object.

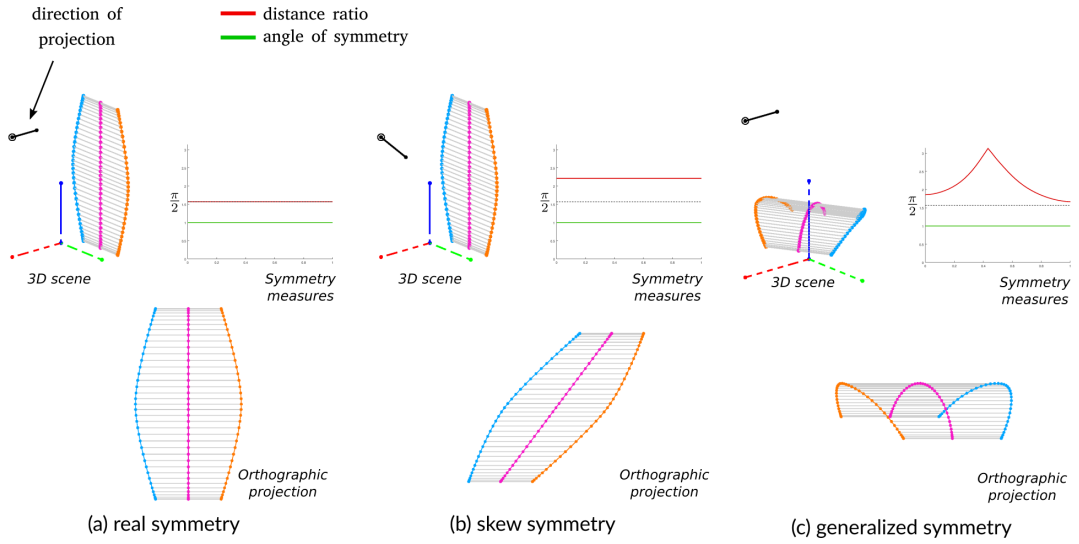


FIGURE 2.13: Example of different categories of symmetries.
3D symmetric curves (blue,orange) and their axis of symmetry (pink) are projected under orthographic projection (in the direction given by the black vector). We measure on the projection: the ratio between the distances of symmetrical points to their corresponding point in the axis (green), and the angle between lines of symmetry and the axis of symmetry (red).

Two points $p, q \in \mathbb{R}^3$ are said to be symmetrical with respect to a plane (O, n) if and only if:

$$\begin{cases} \overrightarrow{pq} \text{ is colinear to } n \\ \frac{1}{2}(p+q) \in (O, n) \end{cases} \tag{2.5}$$

Two curves C_1, C_2 are symmetrical with respect to the plane if they can be decomposed in pairs of symmetrical points (see Figure 2.13).

We call *lines of symmetry* the line segments joining each pair of symmetrical points, and *axis of symmetry* the curve made by the middle points of the lines of symmetry.

One of the issue is to detect such configuration within a line drawing, which represents a projection of the lines and points. We usually classify mirror-symmetry of lines in a 2D drawing in 3 categories [TNT89], which are illustrated in Figure 2.13:

- *real symmetry*: each symmetrical pair of point is at equidistance of and orthogonal to a common straight axis. With the assumption of generic viewpoint, this shapes are generally perceived as planar in 3D [UN93].
- *skewed symmetry*: each symmetrical pair of points is at equidistance of a common straight axis, and the angle made by lines of symmetry and the axis of symmetry is constant along the axis [LS96]. It is generally perceived as the projection of a 3D mirror-symmetry with a straight axis of symmetry.

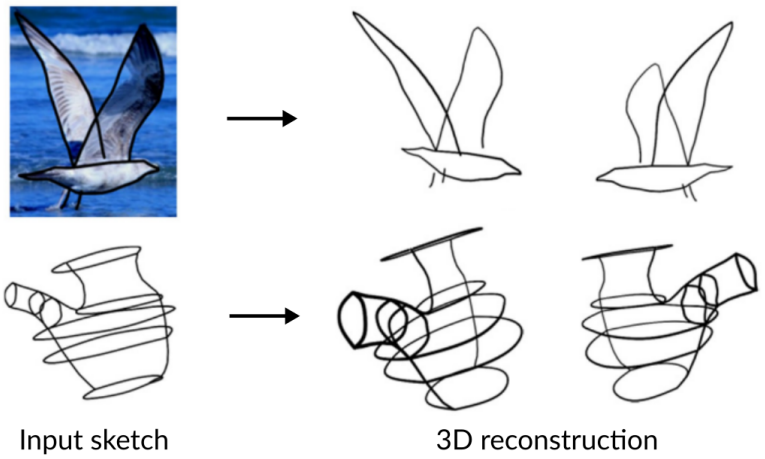


FIGURE 2.14: Example of results from [CSMS13]. Depth is recoverred in a 2D line drawing using the assumption of global mirror-symmetry.

- *generalized symmetry*: each symmetrical pair of points is at equidistance of a common smooth 2D curve. It is a generalization of the projection of two 3D symmetrical curves [TNT89].

Note that these definitions account for symmetries visible under orthographic projection.

Ulupinar et al. [UN93] defined *planar symmetry*, as a specific case of generalized symmetry, where all symmetrical pairs of points share the same tangent along their respective curve. In their approach, they propose a classification of surface patches using the symmetries implied by the projection of their borders. In particular, if the border is covered by exactly 2 symmetries: one skew symmetry with straight curves of symmetry, and a planar symmetry, the surface is perceived as a developable surface.

Recovering symmetry correlations in a single-view image is a problem in itself. One approach proposed reconstructing architecture models from a single image [JTC09] while basing the symmetry correspondances on user annotations. In particular, the user annotates vertices in the image representing a pyramid frustum which is used as a symmetrical primitive for the model.

The approach presented by Oztireli et al. [ÖUP+11] recovers the depth of a hand-drawn sketch requiring the user to annotate only a few points in the sketch, the ones that lie on the symmetry plane. These annotated points are used to infer the orientation of the plane. Their algorithm then performs the matching between the curves in the sketch, based on feature points computed in each curve. Cordier et al. [CSMS13] presented a method to compute the symmetry orientation without any user annotations. The orientation of the symmetry is chosen as the one that maximizes the amount of possible pairs of symmetrical curves, and the compactness of the resulting 3D shape. See an example of result in Figure 2.14. For both methods [ÖUP+11, CSMS13], the

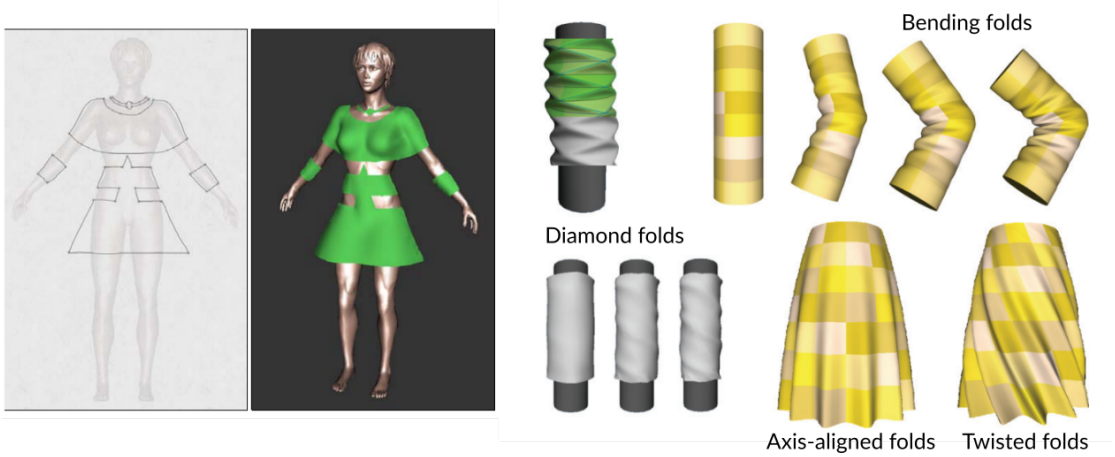


FIGURE 2.15: Illustration of the interface presented in [TCH04] (left), and of the different procedural folds models proposed in [DJW+06] (right).

reconstruction is completely determined by the symmetry assumption, without requiring other optimization.

Other methods use the symmetry assumption indirectly, proposing interfaces to model shapes composed of geometric primitives, such as generalized cylinders and cuboids [GIZ09, SAG+13, CZS+13]. Primitives are then transformed and optimized to fit the projected lines depicted in a drawing or a photo.

Our algorithm to reconstruct 3D sewed objects from a single sketch combines constraints of symmetry and developability of the object. We therefore consider now some relating works dealing with developable surface in the context of 2D input based modeling.

2.2.3 Modeling sewed objects from 2D inputs

In this section, we focus on approaches modelling developable objects, whether they are rigid sewed objects, such as fashion accessories, or more extensible ones, such as garments, for which the shape also depends on the character that is dressed. Garment modeling also accounts for fold modeling, which is a complex task for which sketch-based methods are of great interest.

Geometric modeling from sketches One of the first sketch-based modeling system for garments was proposed by Turquin et al. [TCH04]. It is based on border and silhouette curves drawn by the user on top of a view of the character to be dressed, see an illustration of the interface in Figure 2.15. This method allows the creation of basic garments, without folds, nor taking into account developpability of the surface.

This method was completed by Decaudin et al. [DJW+06], who added fold modeling, and a developability optimization. Their approach models different kinds of folds: axis-aligned folds due to gravity, twisting folds, and diamond folds due to cloth compression (see Figure 2.15). The latest is modelled by a procedural technique, called buckling mesh. These folds are not positionned using the sketch, their shape is modeled on a cylinder primitive and depends on a parameter representing the thickness of the cloth.

Fold modeling from sketches was also tackled in [TWB+07]. The user draws fold lines and annotates the radius at extremities, and the folds are modelled with a conic shape. Robson et al.[RMSC11] proposed a system using context-related constraints, providing more flexibility to sketch inaccuracies: transitions between tight and loose areas of the garments are guided by gravity, silhouettes of cloth hanging are made vertical, and border curves are made planar.

Such input was also used in the SecondSkin system [DPS15], to model layered surface patches dressing a virtual 3D character. The 2D input curves are drawn in various viewpoints of the character, and are directly interpreted as 3D boundary curves of the patches.

As for sewed objects, Plushie [MI07] was proposed as an interactive system to model 3D plush toy models by drawing their desired silhouette from different points of view. The system computes a set of 2D patterns for each patch added during edition and computes its 3D shape using a physics-based simulation.

Jung et al. [JHR+15] designed a fully geometric-based method to compute the mesh of a sewed object with folds from two sketches corresponding to orthogonal views of it. The sketches contain contour curves annotated as either silhouette, border or seams. A first rough surface is modeled using the contour curves and the surface is refined optimizing alternatively for developability of the surface and silhouette matching.

Recently an hybrid method was proposed to add realistic folds in sewed objects and garments [LSGV18]. Their system models 4 different categories of folds: hemline folds, uniformly gathered folds, pinched pleats, and knife pleats (illustrated in Figure 2.16). They use sketched path strokes to determine the location, direction and the type of folds wanted, and the 3D shape and 2D patterns are computed using a combination geometric and physics-based constraints.

Image-based and data-driven approaches Other approaches used pictures instead of sketches to guide the reconstruction of 3D objects. Some of them are using shape-from-shading techniques [ZCF+13], but most of them rely on garment databases. First data-driven methods were generating images of new garments from databases of images [SSP+14]. While available 3D garment databases became richer, many data-driven image to 3D garment methods arose.

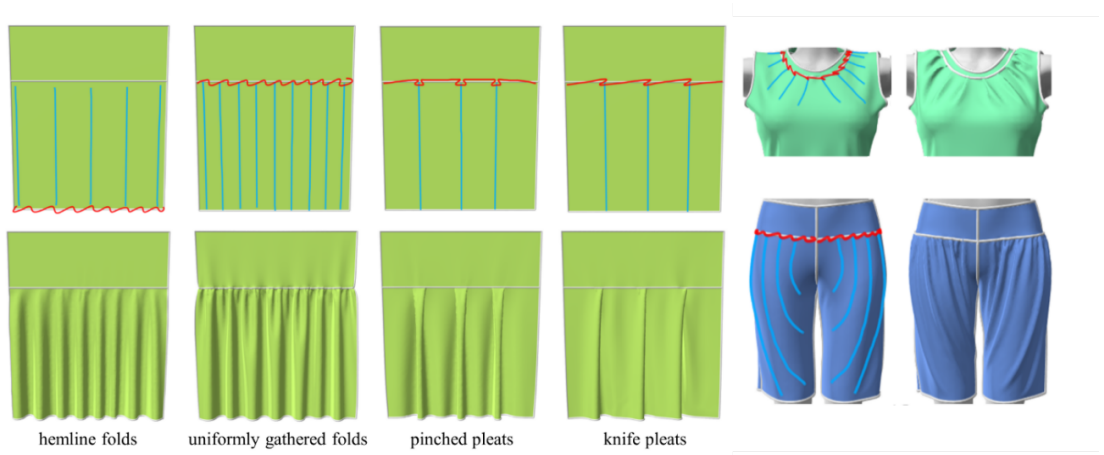


FIGURE 2.16: Different folds modelled in FoldSketch [LSGV18] (left), and examples of folds added in a 3D garment using their system (right). User annotations are represented with red and blue curves.

Image-based reconstruction is usually a three-step process: segmentation to distinguish body and garment in the image, research of 3D garment templates in the database, and deformation of the garment so that it fits the image. One of the main challenges is to find in the image the parameters leading to the right deformation.
Some methods are overcoming this challenge by using additionnal information about the image, using a depth camera [CZS+13], or a body mannequin of known 3D geometry [JHK15].

Machine learning methods were also proposed. Daněřek et al. [DDÖ+17] used a CNN trained to output garment vertex deformations so that a 3D mesh fits at best a picture. It is mostly used for deformation due to posing and motions, such as wrinkles.

Yang et al. [YAP+16] proposed a data-driven method that learns garment parameters such as fabric material, design pattern parameters, sizing and wrinkle density based on a picture. Template garments in the database are used as first reconstruction and the estimated parameters are used in an optimization combining statistical, geometrical and physical priors. A body model is also estimated from the input image, based on a human body model database. Their approach show convincing reconstruction and retargetting, as illustrated in Figure 2.17. The computational time is quite heavy, taking 4 to 6 hours to compute a garment from a picture.

Deep learning was also used to learn a latent space relating the spaces of 2D sketches, 3D meshes, and the combined space of body shape + garment pattern parameters [WCPM18]. One of their application is to generate 3D garments from drawings. Their approach requires a training step as pre-process over a large database of available 3D garments along with their 2D patterns, as well as various body shapes, and is for now restricted to shirts and kimonos.

FIGURE 2.17: Results of image based reconstruction and retargetting of garments using the system of Yang et al. [YAP+16].

2.3 Transferring styles and shapes

Data-driven and example based approaches for synthesis of 3D surfaces share realistic results. One of their strengths is the use of garment templates that represent realistic geometry. Instead of focussing on producing from scratch correct geometry and topology, the challenge becomes to fit the objective shape and style.
We address in this section the issue of style transfer, or how to use an example as a guide to modify an existing content. The first section introduces the challenges and different application of such method. The second section focuses on garment-related approaches, which are more of our concern.

2.3.1 Example-based transfer

Transfer by analogy Example-based style transfer was introduced first by Hertzmann et al. [HJO+01] presenting the concept of *Image analogies*. This method uses 3 inputs: an unfiltered and a filtered source image, respectively A, A', and an unfiltered target image B. The goal is to generate a new image B', which undergoes a similar transformation as $A \rightarrow A'$. See an example in the Figure 2.18.

One important challenge of this type of approach is to find correspondences between A and B to perform the transfer of deformation. For images, illumination information has been found to be really effective for matching pixels areas in a goal of style transfer [FJL+16]. The original paradigm inspired other methods in various domains.

FIGURE 2.18: Image analogies, by Hertzmann et al. [HJO+01]. Their algorithm generates a stylized version B' of the image B from a pair of initial/stylized images (A, A')

Curve analogies [HOCS02] performs transfer of curve styles and patterns. They propose a matching measure that is invariant under rigid transformations and allows to synthetize line patterns for curves with different orientation and direction.

Deformation transfer [SP04] was proposed to transfer the animation of a 3D mesh to a mesh of a different character with similar topology. The goal is to animate the target object in a similar way as the source object. They map vertices between the two meshes by computing a transformation on the source mesh that deforms it into the target mesh. Then, they choose closest vertices between the deformed source mesh and target mesh.

Style transfer was recently experimented on 2D line animation [DBB+17], to transfer the style of a motion from one animation to another. The difference with deformation transfer is that here one wants to keep the motion of the target animation, but only transfer the expressiveness of the source animation, meaning timing, spacing, and pose-centered deformation.

Transfer by analogy relies on the transformation $A \mapsto A'$ to generate B' from B. What happens if A is not available, but only A' and B to generate B' ? Intuitively, by looking at pictures of Figure 2.18 one can imagine what the result should be, because we perceive the different style of the two pictures. However, to automatize this process, the challenge is to define which features of A' define style and which features of B should not be altered by style transfer. Other methods were proposed following this idea, we present some of them in the next paragraph.

Feature based style transfer One popular topic in geometric modeling is transferring distributions and arrangements of elements. Among its various applications, we can note transfer of distribution of 2D patterns [HLT+09], and 3D objects [EVC+15]. In these approaches, the items that are arranged in the scene keep their geometry, but share statistical distribution with the source configuration. The distribution is transferred using Markov chains and Monte-Carlo like algorithms. A statistical version of the curve analogy algorithm was also proposed based on this principle [LA15].

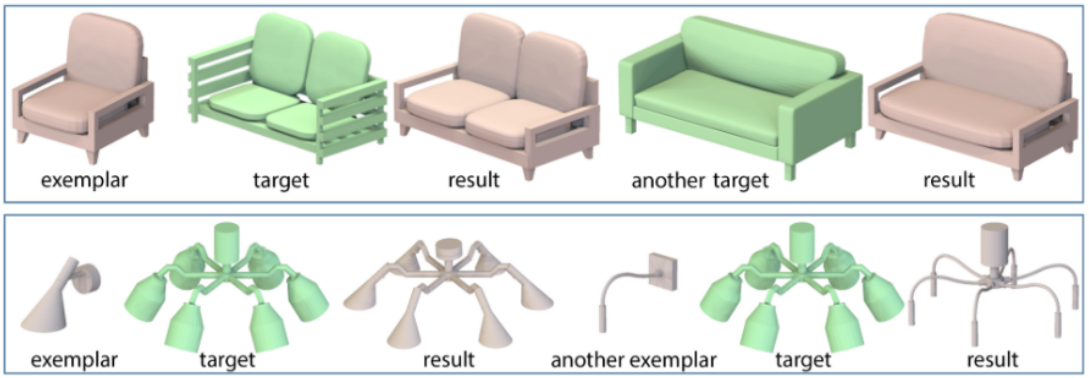


FIGURE 2.19: Style transfer used in the scope of furniture modeling [LKWS16]. While the functionnality of the target is preserved, style of the exemplar is transferred.

Definning and transferring style of 3D objects was discussed in the scope of furniture modelling. Liu et al [LHLF15] studied the perception of style in such objects, and concluded that while their functionality is strongly related to their gross shape and arrangement of their major parts, their style is perceivable in the fine geometric details of each part. These observations were used in an algorithm to transfer the style of a piece of furniture to another presenting different functionnality [LKWS16], allowing to create a set of detailed furnitures by only designing the details of one, see Figure 2.19.

2.3.2 Style transfer of garments

For garments, two types of style transfer can be found in literature. On one hand, the transfer of the geometry and distribution of geometric details to different shapes. On the other hand, transferring a whole garment from one person to another, giving rise to the question of garment style, and what makes two garments looking similar without sharing the same geometry.

Transferring wrinkles Transfer was used to generate quickly wrinkles in a dressed human animation [WCPM18]. A dataset of wrinkles is first synthesized using physics based animation on one sample pose of the character. This dataset is decomposed according to their location in the human body, and are transferred to a different pose in the animation. Their method allows a more computationnaly efficient process for animating a dressed character. Instead of generating the whole animation by launching physics-based simulation on a fine cloth mesh, this process only requires to run the simulation on the fine cloth mesh for one examplar frame of animation. The rest of the frames can be generated using a low definition mesh, which can be then refined by transferring the wrinkles from the mesh of the examplar frame.

FIGURE 2.20: Style preserving garment transfer [BSBC12]: the same garment is transferred to characters with different morphologies.

Transferring geometric details was also proposed for furnitures made of cloth (armchairs, pillows..) [BHS+17]: wrinkles, fabric patterns, wood grain, scratches and cloth seams. Their method is related to transfer by analogy since they use as input source a coarse mesh and a displacement map generating the high level source mesh. The input target coarse mesh is mapped to the input one using an important amount of geometric measures including, but not restricted to curvature, normals and height.

Transferring garments to different morphologies The last field of study we discuss is the problem of transferring garments from one character to another, which is often named *pattern grading*. Imagine we want to dress multiple characters with the same garment. One can easily understand why applying physics-based simulation to the same patterns on different character often leads to inadequate results. Scaling uniformly the patterns is in general not enough for the garment to keep the same style while fitting the new character. Pattern grading in cloth manufacturing abides by a multitude of rules varying from one garment style to another [MMY01].

Solutions based on skinning techniques were proposed by Wang et al. [WWY05]. While those techniques usually work well with tight areas in the garment, they often fail preserving the style of the garment in lose areas. They inspired alternative solutions, one adaptative, were skinning techniques are only applied in tight areas, while lose areas are scaled uniformly [CSMT03], and one interactive, were the user specifies constraints on the shape by drawing silhouette curves [MWJ12].

Brouet et al. [BSBC12] presented a fully automatic solution for style preserving pattern grading of virtual garments. They propose geometric criteria to define style in a garment, and other criteria for plausibilty. The method is then based on transferring the style criteria while maintaining plausibility in the generated garment. See examples of their results in Figure 2.20.

In Chapter 5, we present how we extracted these style criteria from fashion drawings, and applied them in a 3D garment modeling process (cf Section 5).

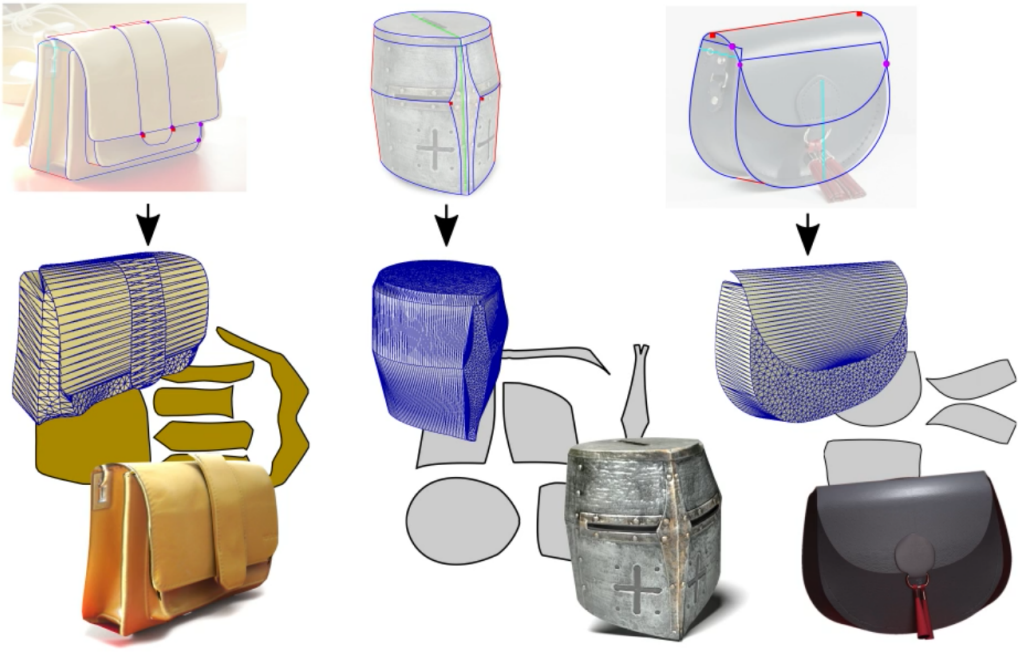
2.4 Conclusion

We saw the main challenges induced by single-view based reconstruction approaches. This ill-posed problem generally requires strong a priori knowledge on the geometry of the object we want to model, or annotations of features that are strongly significant to describe its shape. On the other hand, C^2 developable surfaces are highly constrained by their geometry. Many modeling approaches exploit the different mathematical formulations of this geometric constraint. We propose in Chapter 3 an approach to reconstruct the surface of a piecewise C^2 developable surface using a single annotated photo. Our approaches uses the strenght of both the C^2 developability and the symmetry assumption in an linear optimisation-based reconstruction system.

This C^2 developable constraint does not account for garment, with often exhibit some extensibility, and may contain folds. While many methods are trying to reconstruct a garment based on an image, or sketches drawn in a plane of the 3D space, we chose an approach inspired by style transfer methods to synthesize garments out of a 2D sketch. We first propose an approch that interprets, reconstructs and completes hidden parts of the folded boundary curve of a garment (Chapter 4). This method also exploits the paradigm of image analogies to transfer the reconstructed folded boundary curve to fold the boundary curve of another surface. Then, we propose a 2D-to-3D garment transfer, in which a 3D garment is synthesized using a single 2D stylized sketch (Chapter 5). The style of the output garment is guided by the input sketch, while suiting a character with possibly different sizen morphology, and pose.

Chapter 3

Modeling symmetric sewed objects using a single photo



This Chapter addresses the reconstruction of sewed objects from a single annotated photograph. Developable materials such as paper, cardboard, metal sheets, cloth or leather are extensively used in the design of industrial products, from Chinese lanterns, furniture, ship hulls and architecture to many fashion items. Despite multiple fields of application, modeling and editing developable surfaces in 3D is a complex problem, for which standard interactive modeling frameworks do not hold.

In this work we investigate a new way to create piece-wise developable surfaces, namely reverse engineering them from annotated photographs. The method presented in this chapter has been presented at jFiG 2016 in Grenoble, where it received a best paper

award. It was published in the journal Computers & Graphics 2017 and has been presented at SMI'2017 in Berkeley.

Single-view reconstruction of 3D shapes is an ill-posed problem, as a 2D picture can represent an infinite number of different 3D surfaces, as stated in Section 2.2.1. Given annotated contours, prior work managed to address this problem in a few specific cases by complementing the minimization of *re-projection error*, meaning the distance of the projection of the reconstructed lines to the input lines, with specific geometric constraints. These constraints include parallelism and orthogonality [LS96], exact mirror-symmetry [CSMS13], or orthogonality of cross-sections in the case of industrial design sketches [XCS+14] (see Section 2.2.2 for details). We contribute to this line of research by introducing a new constraint, tailored to developability conditions.

We have integrated our algorithm into a sketch-based modeling system where users annotate silhouettes and symmetries over a photograph of the object they wish to reconstruct. Drawing over a photograph allows inexperienced users to quickly model common objects. In contrast, drawing an imaginary developable surface without guidance would require much more expertise. This is especially the case for objects composed of surfaces which exhibit particular mathematical non intuitive cues.

Our contributions are:

1. An end-to-end system to reverse-engineer piecewise developable objects from a single annotated photograph;
2. A method to infer the 2D projection of rulings over a developable surface from its silhouette
3. A new linear energy term for encouraging the developability of a 3D surface reconstructed from the projection of its rulings

We validate our method by reconstructing a variety of piecewise-developable objects from photographs, such as furniture, fashion items, or tents.

Figure 3.1 illustrates the main steps of our method which can be summarized as follows.

Our system takes as input a photograph of a piecewise developable object, on which the user traces 2D Bézier curves (piecewise cubic C^1 Bézier splines) to delimit surface patches, along with other annotations. A full description of the annotations and hypothesis on the input is proposed in Section 3.1.

The first stage of our pipeline analyzes the 2D input to compute constraints to be applied onto the 3D interpretation of the curves, which is described in Section 3.2. To this end, we first identify surface patches by extracting the minimal cycles of the curve network. We then generate rulings inside each patch which is partly delimited by a silhouette

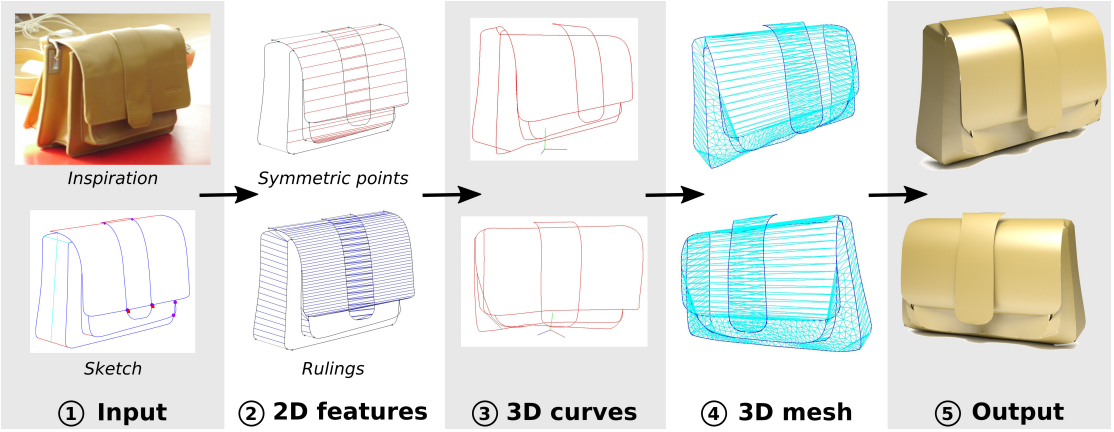


FIGURE 3.1: Overview of our approach. (1) Our system takes as input a photograph where the user has traced the object silhouette and the surface patch boundaries. We also ask users to annotate a global symmetry plane and to indicate symmetric curves. (2) We analyze these annotations to register symmetric curves and to propagate rulings from the silhouettes towards the interior of the surface patches. The detected symmetric points and rulings provide us with geometric constraints on the 3D curves, which we express as linear terms in an optimization. (3) Solving this optimization produces a 3D curve network, which we subsequently surface with developable patches. (4,5) These curves are used as boundaries to generate a piecewise developable mesh of the object

curve, by building on the fact that the silhouettes of a developable surface are straight lines aligned with rulings. Finally, we find point to point correspondences between each pair of symmetric curves.

The output of the analysis stage is a 2D set of rulings and a set of 2D pairs of symmetric points, see Figure 3.1-(2). The second stage of our approach aims at lifting the contour curves in 3D (Figure 3.1-(3)), which we present in Section 3.4. This step is achieved by constraining the control points of the contours in order to satisfy the detected symmetries while yielding a constant surface normal along rulings. We express these symmetry and developability constraints as linear terms in a function to optimize. This function is complemented by terms expressing the minimization of re-projection errors and foreshortening, enabling us to improve robustness to sketch inaccuracies and perspective distortions.

The last stage of our approach generates a developable triangle mesh for each surface patch, see Figure 3.1-(4). We simply generate two triangles for each pair of consecutive rulings for the patches partly delimited by silhouette segments. We resort to a more involved surface optimization process [BK04] for the other patches. See Section 3.5 for details.

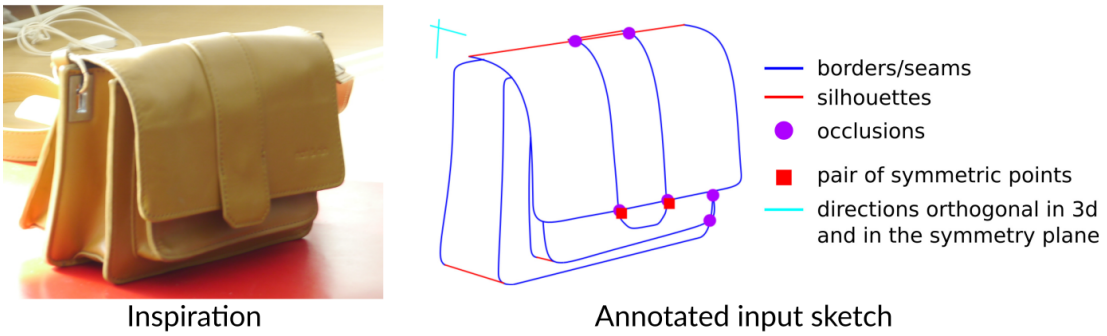


FIGURE 3.2: Illustration of the annotations provided by the user

3.1 Input image and annotations

Our method takes as input one single annotated photograph. In this section, we present the assumptions which we make on the image and the represented object, and we describe the type of expected annotations.

3.1.1 Hypothesis on the image

To perform our analysis, we need to make a few assumptions on the photograph and the object represented on it. We assume that the object represented is piecewise developable, meaning made of developable pieces, that we call *surface patches*, sewed together. Three additionnal hypothesis are made : the object is seen from a general viewpoint, the image represents an orthographic projection of the object, and the object is globally mirror-symmetrical.

Generic viewpoint As defined in Section 2.2.1, a viewpoint is generic if it is stable under small changes, meaning that our perception of the object does not change when the viewpoint changes a bit. It is a very common assumption in single-view based modeling [XCS+14, CSMS13], since it ensures that the view represents the global form of the object. Under those assumptions, we can assume that all projected lines that are straight in 2D corresponds to straight lines in 3D, and that intersections occuring in 2D are actual intersections in 3D. However, we observed that it is not true in some cases, and intersections in 2D can occur at points having different depths in 3D (see the purple points in the sketch of Figure 3.2). We therefore decide to interpret intersections in 2D as intersections in 3D by default and allow the user to annotate the occasionnal intersections in 2D that correspond to occlusions in 3D. One possible automation of this process could be to use line junction classifiers, as we presented in Section 2.2.1. But, as discussed by Barrow et al. [BT81], this type of method is not robust to ambiguous cases.

Orthographic projection The annotated contours are traced over a photo, so they technically represent a perspective projection of the object. We assume here that distortions due to perspective are low, and interpret the curves as if they represented an orthographic projection. This assumption greatly simplifies our symmetry interpretation. However, we account for this inaccuracy while lifting the contour curves in 3D by allowing their projection to vary from the image (cf Section 3.4).

Global mirror-symmetry As stated in Section 2.2.2, the hypothesis of mirror-symmetry has been proven efficient in the field on 3D reconstruction of line drawings [CSMS13, ÖUP+11]. We assume that our object is globally mirror-symmetric, meaning that there exist a plane in 3D for which each point on the surface of the object has a symmetric point also on the surface of the object. This hypothesis will be used during our curve reconstruction process in the form of an energy to be minimized (cf Section 3.4.1)

3.1.2 User annotations

As displayed in the example of Figure 3.2, our input contains a set of annotations, that we describe here. The annotations are used to depict the contours of the object and specifically the surface patches. As in True2Form [XCS+14], the input curves are represented with piecewise cubic Bézier splines, which is a representation commonly used in softwares like Inkscape[1] or Illustrator[2]. This representation will be kept for the 3D reconstruction step, because it guarantees piecewise C^1 continuity of the curves while using a limited number of control points (4 per Bézier curve).

Contour annotations The object is depicted in the input with contour curves, that match the brightness discontinuities of the image. We use color to distinguish between *silhouettes* (orange), as defined in Section 2.2.1, and *borders* (blue). Border curves correspond to surface patches boundaries in this case, and include seams which are located at patches junctions.

We use the theorem proven by Koenderink [Koe84] stating that in the case of developable surfaces, where the Gaussian curvature is zero everywhere, the silhouettes from any viewpoint are always straight lines corresponding to planar rims. Since we are working with developable surface patches, we assume that silhouettes should be represented with straight lines, and we regress a straight line out of each silhouette curve provided.

Symmetry annotations Further annotations help us to exploit symmetry. The user specifies a global symmetry plane by tracing the projection of two orthogonal vectors

[1] https://inkscape.org/
[2] https://www.adobe.com/ch_fr/products/illustrator.html

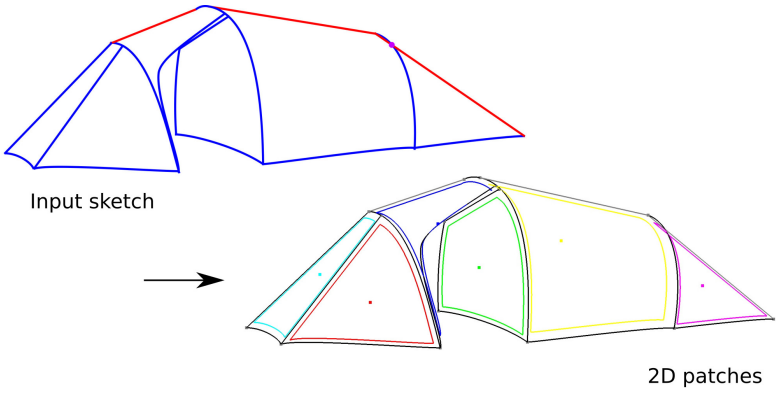


FIGURE 3.3: Extraction of 2D cycles. Top: input sketch with silhouettes in red and boundaries in blue. Bottom: Different colors are used for the different 2D patches.

in that plane (cyan lines), and two symmetric points (red squares), which form a third 2D vector that is the projection of a vector that is orthogonal to the symmetry plane. These three vectors combined allow us to compute the location and orientation of the symmetric plane, see Section 3.2 for details. Finally, the user indicates each pair of symmetric curves as well as self-symmetric curves. While automatic methods have been proposed to detect global symmetry automatically [CSMS13, ÖUP+11], we found that these annotations are easy to provide while greatly simplifying subsequent analysis.

3.2 2D curve analysis and rulings extraction

The first stage of our method analyses the input 2D curves and annotations to identify the surface patches, generates their rulings, and builds correspondences between symmetric curves.

3.2.1 Extracting 2D patches

The input of our method is a network of Bézier curves that represents the object's smooth silhouettes, sharp boundaries, or interior seams. We automatically detect intersections between the traced curves and resample them to obtain C^1 piecewise cubic curves with endpoints at intersections. We then extract the *minimal cycles* of this curve network, each cycle representing a surface patch, see Figure 3.3 for an example. A cycle is minimal if it does not contain any other cycle. We represent the curve network as a graph, where the nodes correspond to curve intersections and the edges to curve segments between the intersections. Each curve linking two intersection nodes n, n' is represented in the graph with two directed edges $e = (n, n')$ and $e' = (n', n)$. We use a standard algorithm to detect minimal cycles in the graph, the pseudocode is provided in Algorithm 1. It starts by chosing an arbitrary node n_0 and incident edge $e_0 = (n_0, n_1)$. The next edge

Algorithm 1 Algorithm to find minimal cycles in a planar graph.

function FINDMINIMALCYCLES(edges)
  edges_to_visit ← stack(edges ∪ {$(e_1, e_0)|(e_0, e_1) \in edges$})
  min_cycles ← {}
  visited_edges ← {}
  while edges_to_visit not empty **do**
    (e_0, e_1) ←edges_to_visit.pop()
    starting_node ← e_0
    cycle ← $\{e_0\}$
    while $e_1 \neq$ starting_node **do**
      cycle.append(e_1)
      incidental_edges ← $\{(f_0, f_1) \in$ edges_to_visit$|(f_0 == e_1)\}$
      if incidental_edges.empty() **then**
        break
      end if
      $(e_0, e_1) \leftarrow \underset{(f_0,f_1)\in\text{incidental\_edges}}{\arg\min} \left\{\alpha(\overrightarrow{e_0e_1}, \overrightarrow{f_0f_1})\right\}$
    end while
    if $e_1 \neq$ starting_node **then**
      continue
    end if
    min_cycles.append(cycle)
    visited_edges ← visited_edges ∪ edgesOf(cycle)
  end while
  return min_cycles
end function

$e_1 = (n_1, n_2)$ is then chosen as incident to n_1 and minimizing the angle between $\overrightarrow{n_0n_1}$ and $\overrightarrow{n_1n_2}$. The process continues until we reach the first node n_0 again, a first minimal cycle has been found. The algorithm then iterates starting with an edge that has not yet been selected in a minimal cycle, until there is no such edge anymore. The output of this algorithm, the list of minimal cycles, contains an extra cycle corresponding to the external boundary of the curve network, which we remove from the list of cycles.

3.2.2 Extracting symmetry

As stated in Section 3.1, we assume there exist a plane for which the object is globally mirror symmetric. We compute the location and orientation of this plane in 3D and match sampled vertices between symmetric curves. This 2D information will be then used in our algorithm in the form of a symmetry energy function, inspired by the symmetry-based reconstruction algorithm presented by Cordier et al. [CSMS13].

Computing the plane of symmetry We compute the plane of symmetry using the two 2D vectors u, v and the symmetric points s_0, s_1 provided by the user (see Figure 3.2). The symmetric points s_0, s_1 provide a projection of one point in the symmetry plane

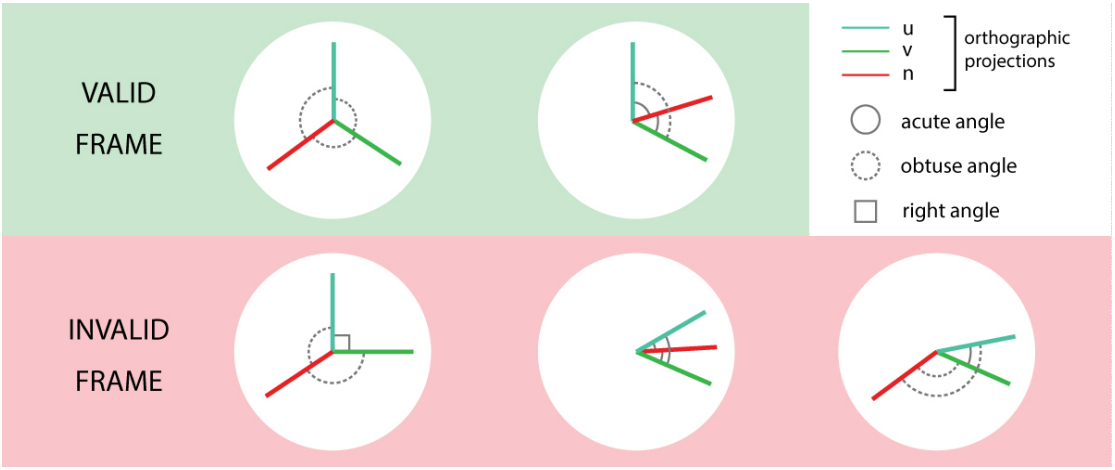


FIGURE 3.4: Examples of valid and invalid projections of orthogonal frames.

(the middle point $m = \frac{1}{2}(s_0 + s_1)$), and of the direction of symmetry $s = \overrightarrow{s_0 s_1}$. Our goal is to obtain the 3D direction of symmetry, meaning the 3D coordinates of s.

The vectors u, v are interpreted as the projection of two vectors that are orthogonal in 3D and lying in the symmetry plane. In 3D, the vectors u, v, s form an orthogonal frame, for which :

$$\begin{cases} \vec{u}.\vec{s} &= 0 \\ \vec{v}.\vec{s} &= 0 \\ \vec{u}.\vec{v} &= 0 \end{cases} \tag{3.1}$$

Since we are assuming an orthographic projection, the x, y coordinates of u, v and s are directly read in the sketch. We solve the system of Equation 3.1 for the unknown z coordinate of s, which is equivalent to solve :

$$\begin{cases} u_z v_z s_z^2 &= (\overline{u}.\overline{s})(\overline{v}.\overline{s}) \\ u_z v_z &= -\overline{u}.\overline{v} \end{cases} \tag{3.2}$$

Two conditions are necessary in order to get a solution for s_z. First the two vector u, v must not be orthogonal in 2D, i.e. $\overline{u}.\overline{v} \neq 0$. When the first condition is satisfied, the second condition is mathematically formulated as follows.

$$\frac{(\overline{u}.\overline{s})(\overline{v}.\overline{s})}{\overline{u}.\overline{v}} < 0 \tag{3.3}$$

It is a condition on the angles between each of the vectors in 2D. It occurs when either one of the angles, or all of them are obtuse. If the drawn u, v, s does not satisfy these conditions, which will be denoted as an invalid frame in 2D, we cannot compute the 3D coordinates of s. A non exhaustive, but representative list of valid, and invalid frames is displayed in Figure 3.4 : in practice invalid frames are unlikely to be drawn because they visually do not represent the projection of 3 orthogonal vectors.

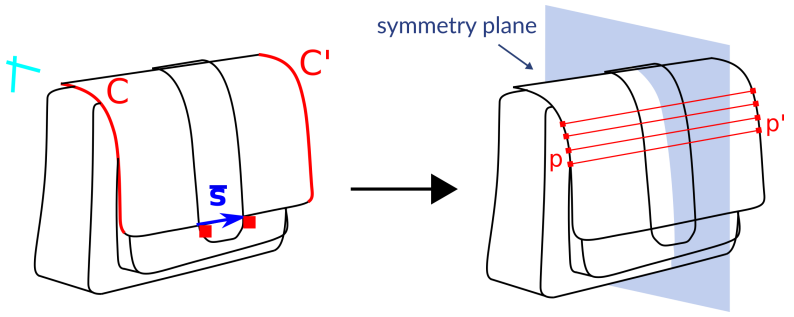


FIGURE 3.5: Symmetry correspondences computed for all pairs of symmetric curves according to the global mirror symmetry. The user-annotated pair of symmetric points (red squares) provides the direction of symmetry in 2D.

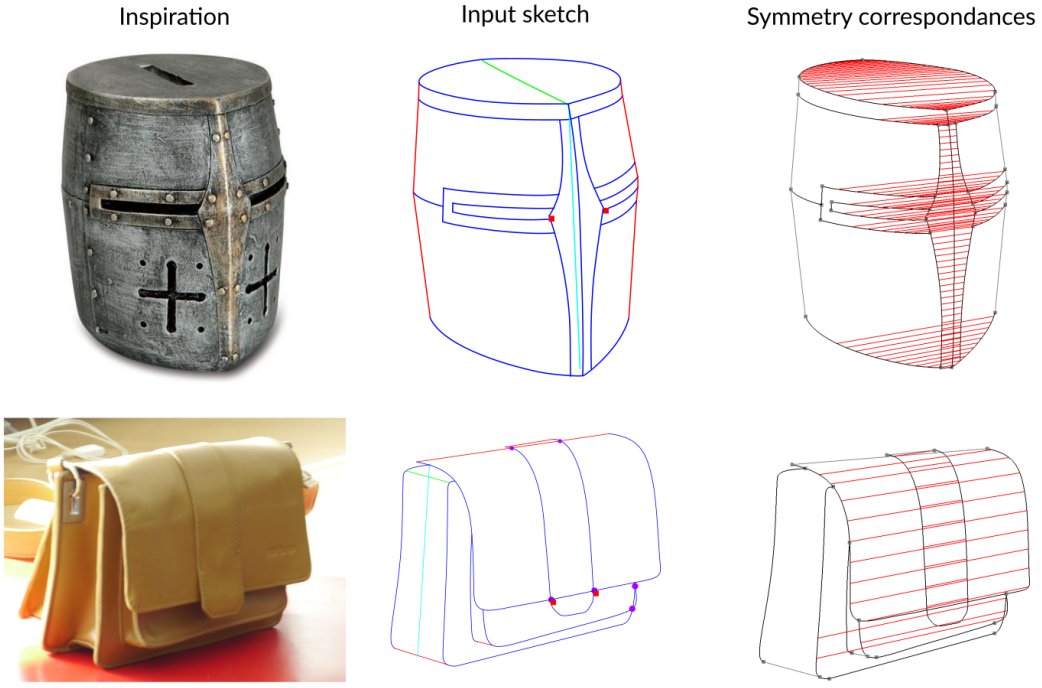


FIGURE 3.6: Some examples of symmetric correspondances, linked by red lines in the second column, extracted from sketches of real life objects (left).

Matching points in symmetric curves Next step of the 2D curve analysis stage consists in extracting point-wise correspondences between all symmetric curves in the network. Note that two symmetric curves may belong to different surface patches, for example C and C' in Figure 3.5. Under orthographic projection, the lines that join symmetric correspondences are all parallel to the projected normal of the symmetry plane.

In practice, we obtain this direction of symmetry from the two symmetric points annotated by the user (red squares in Figure 3.5, left). We then build correspondences between each pair of symmetric curves by sampling them curvilinearly and finding for

each sample of one curve the closest sample of the other curve in the direction of symmetry. See some examples of point correspondances in Figure 3.6

3.3 Extracting rulings in a patch

Additionnally to symmetry, we want our method to exploit the hypothesis of C^2 piecewise developability of the surface. We use the fact that those surfaces are ruled, and propose an algorithm to find the projection of some rulings of the surface in the input drawing.

As presented in Section 3.2.1, the minimal cycles computed in 2D are interpreted as the borders of the projection of 3D surface patches. Each patch is developable, and we present here a method to infer the projection of a set of rulings of the patches. For that, we use the silhouette lines that were drawn by the user (cf Section 3.3.1), and propagate their direction along the patches' borders (cf Section 3.3.2).

3.3.1 Silhouettes as projection of rulings

The goal of this section is to interpret the projected contours of a sewed object to find the projection of rulings of its surface. In this first part, we explain how the silhouette lines give a first set of those projected rulings.

Observations Let us make some observations on different views of one real sewed object, namely the hat displayed in Figure 3.7. We have drawn ontop of one part of the object a set of rulings of the corresponding surface patch. We can easily distinguish two cylindrical parts were rulings are parallel to each other, two conical parts, were rulings all connect at an apex point, and a planar part for which any set of rulings is valid. Recall that those 3 types of surfaces are developable (cf Section 2.1.1).

Let us now focus on the silhouettes of the object, while excluding the case of borders and seams. We observe two things : in every point of view displayed in Figure 3.7, all silhouettes are straight lines, and they correspond to rulings drawn in the hat, or to planar parts of the surface. The first remark directly echoes the theorem of Koenderink et al. [Koe84] stating that the only surfaces yielding planar rims and straight silhouette lines are Zero Gaussian curvature surfaces, i.e. developable surfaces (cf Section 2.2.1 for details). The observation that silhouettes of the hat correspond to rulings of the hat's surface is interesting for our purpose, because, if generalizable, it provides us of the direction of one ruling for each straight silhouette drawn in the image.

We now demonstrate how this observation can be generalized using the mathematical properties of C^2 developable surfaces (see Section 2.1.1 for a recall of these properties).

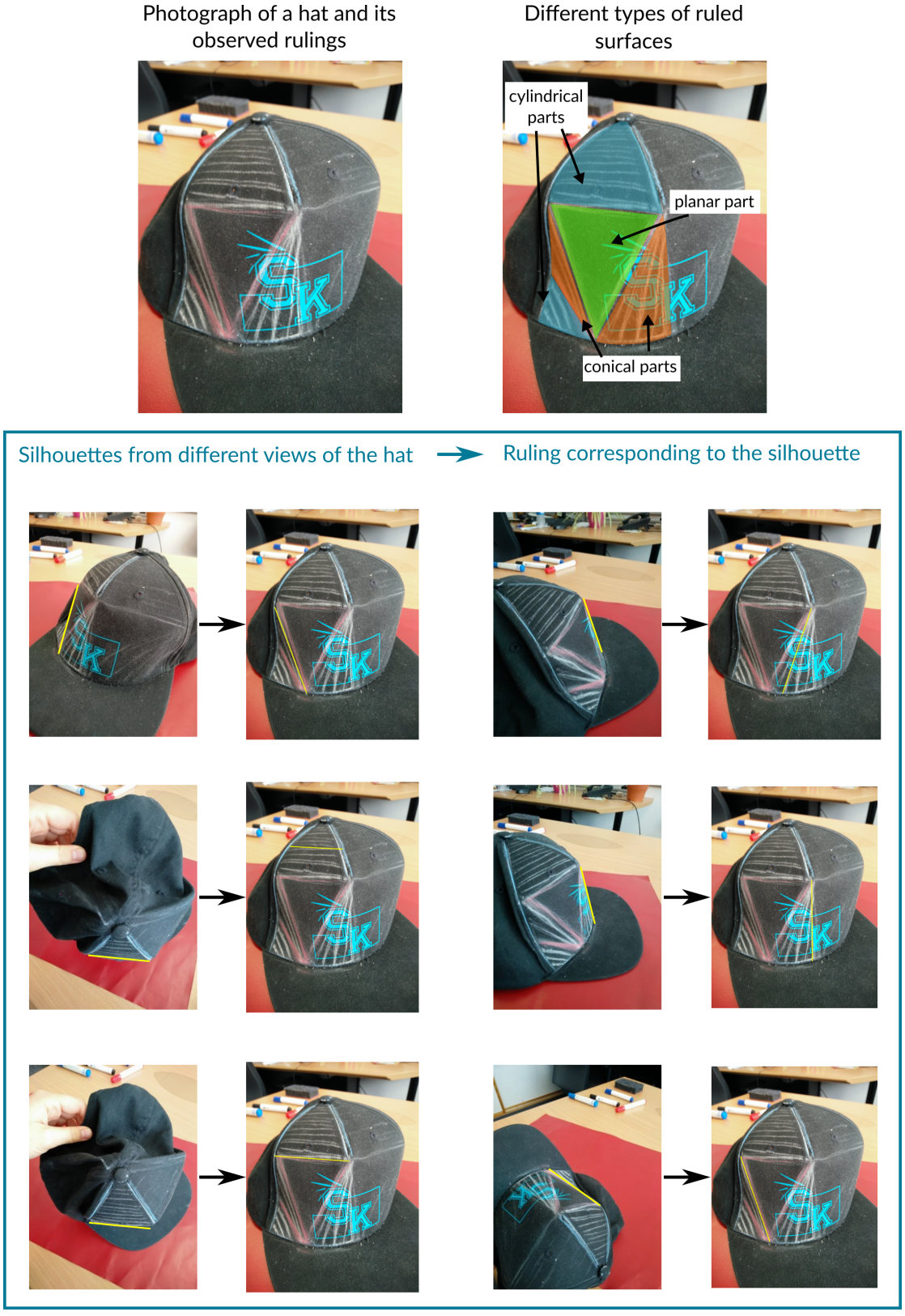


FIGURE 3.7: Different views of a same sewed object. We observe that all silhouette lines correspond to projection of rulings of the surface (yellow lines).

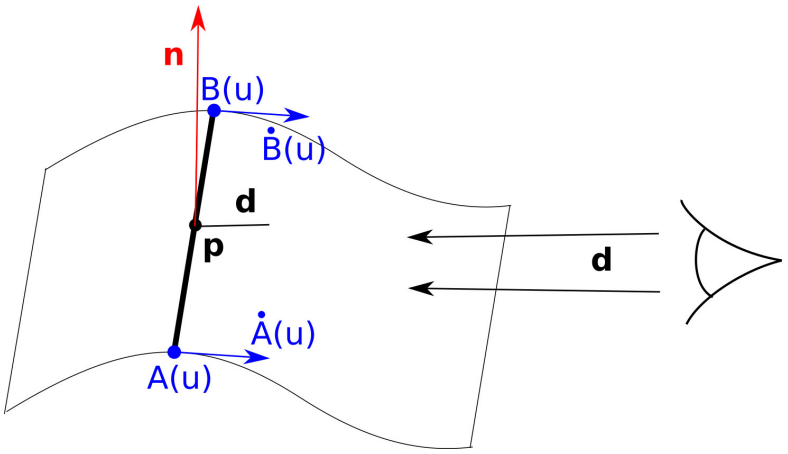


FIGURE 3.8: Developable surface and a ruling projecting onto a straight silhouette.

Generalization As stated before, C^2-continuous developable surfaces have the property that their 2D silhouettes are straight segments corresponding to planar rims. We propose a short derivation of this property. Let us first define a *silhouette* as the set of points of a surface S with normal n orthogonal to the view direction d [BT81]. Note that using this definition, silhouettes do not include borders (ie. boundaries of a trimmed surface). The 2D silhouette is obtained by projection of this set of points to the image plane.
Let us consider a point p on a silhouette of a C^2 developable surface (see Figure 3.8). By definition, a point of a developable surface belongs to some ruling with a constant normal vector along it. Moreover, as a point of a silhouette, the normal vector at p is orthogonal to the viewing direction d. Therefore, all points along the ruling have a normal vector orthogonal to d. This makes the ruling being a silhouette and the silhouette being a straight line. As a consequence, all silhouettes of a developable surface are necessarily rulings. Since the rulings are straight lines, their projections form 2D lines in the image plane.

To be fully complete, we show that the only configuration under which a developable surface can yield a non-straight silhouette is when the silhouette is confounded with the surface boundary, such as when a cylinder is viewed from a viewpoint aligned with its axis of revolution. In our context, we assume that the object is not photographed from such an accidental viewpoint.

Let us consider a ruled surface S parameterized by $S(u,v) = \alpha(u) + v\omega(u),\ (u,v) \in D \subset \mathbb{R}$. Recall that (Section 2.1.1) S is developable if and only if

$$\det\left(\alpha'(u), \omega(u), \omega'(u)\right) \tag{3.4}$$

We denote $n(u,v)$ the unit normal at parameter (u,v). For the sake of simplicity, we do not explicitly write the parameters (u,v) for the functions when the relation is true for

the entire surface. We also denote $S_u = \frac{\partial S}{\partial u}$, and similarly with S_v, and n_u. We further denote the mixed partial derivative $S_{uv} = \frac{\partial^2 S}{\partial u \partial v}$.

Note that for a given $\hat{u}$, $S(\hat{u}, v)$ is a ruling of the surface, and $S_v(\hat{u}, v)$ is a director vector of the ruling. Let us consider a point at parameter (u_0, v_0) on the silhouette satisfying therefore $n(u_0, v_0) \cdot d = 0$, where $n(u_0, v_0)$ is the unit normal at the parameters (u_0, v_0), and d is the constant view direction. The set of parameters corresponding to a silhouette in a neighborhood of (u_0, v_0) must satisfy $n(u_0 + \mathrm{d}u, v_0 + \mathrm{d}v) \cdot d = 0$, where $(\mathrm{d}u, \mathrm{d}v)$ are some infinitesimal displacements in the parametric space. As the normal is constant along the v direction, i.e. corresponds to the rulings, the parameter $\mathrm{d}v$ can be dropped. Thus any silhouette of S which is not a ruling should satisfy the relation

$$n_u(u_0, v_0) \cdot d = 0 \ . \tag{3.5}$$

We show in the following that Eq. (3.5) only holds if the direction d is aligned with the rulings, i.e. the rare case where the surface is viewed from its side.

First, we can state that satisfying Eq. (3.5) implies that the three vectors

$$(n(u_0, v_0), n_u(u_0, v_0), d) \text{ are forming an orthogonal frame.} \tag{3.6}$$

Indeed, the silhouette condition implies that $n(u_0, v_0)$ is orthogonal to d, Eq. (3.5) implies that $n_u(u_0, v_0)$ is orthogonal to d, and n is necessarily orthogonal to n_u as it is a unit vector. Second, we can state that for a developable surface, the three vectors

$$(n, n_u, S_v) \text{ define an orthogonal frame,} \tag{3.7}$$

for any parameters (u, v). By definition, n is orthogonal to S_v. And we can show in the following that n_u is orthogonal to S_v. The developability condition from (3.4) can be rewritten in term of surface derivatives as $\det(S_u, S_v, S_{uv}) = 0$. This determinant can further be expressed in term of scalar and vector product and rewritten as

$$S_{uv} \cdot (S_u \times S_v) = 0$$

implying that $S_{uv} \cdot n = 0$. Moreover, one can check that every smooth surface satisfies $S_{uv} \cdot n = -S_v \cdot n_u$, which leads to the expected conclusion that S_v is orthogonal to n_u.

Finally, comparing the two frames in (3.6) and (3.7) leads to the conclusion that d must be parallel to S_v, and therefore to the rulings of the surface. We conclude that silhouette of developable surface which are not only rulings of the surface only arises if the view direction is aligned with the rulings, and then be confounded with the boundary of the surface.

Note that the inverse is also true: let us assume that the surface is viewed from a direction d aligned with the rulings S_v. As property (3.7) is satisfied for any parameters (u, v), it implies that d is orthogonal to n, and thus is a silhouette.

Conclusion We showed that any silhouette of a developable which is not a boundary of the surface is necessarily a straight segment corresponding to a ruling of the surface. Any other silhouettes which corresponds to the side view of the surface, arising when the view direction is aligned with the rulings, are confounded with the surface boundary.

This property of developable surfaces is fundamental for our method, for two reasons. First, it makes sense to compute the 3D reconstruction of the silhouette lines, because they represent 3D straight lines lying on the surface of the object. Second, the presence of a silhouette in the annotated contour lines gives us the direction of one ruling of the patch it belongs in. The next section explains how to use this information in order to find other rulings in the patch.

3.3.2 Rulings propagation inside the patch

We address now the issue of finding the projection of rulings inside a developable patch using its projected contours. Finding rulings inside a patch corresponds to finding an appropriate matching between vertices of its boundaries. The rulings are then defined as the straight lines between the matched points.

In 3D the matching problem was discussed in various literature [PS07, Fre02] (cf Section 2.1.2). It is usually based on minimizing the *warp angle* between two vertices, defined as the angle between the normals of the surface at those vertices. In our case, we do not have 3D boundary curves, but only part of their projections. However, using the theorem discussed in the previous section, we can assume that the straight silhouette lines drawn by the user in the input sketch are providing a first set of valid projected rulings of the patch they belong to.

In this section, we investigate geometric cues that will help us to find a suitable matching of other vertices of the patch.

Observations We note that for cylindrical parts, all rulings are parallel in 3D and remain parallel under orthographic projection. We can recover these cylindrical rulings in the 2D image using an algorithm similar to the symmetric pair matching algorithm (see Section 3.2.2), i.e. by searching for vertices that are linked with colinear vectors. The silhouette line provides, in this case, the direction guiding the matching.

Remember that the other categories of developable surfaces are cones, for which all rulings meet at an apex point, tangential surfaces, for which all rulings are tangents

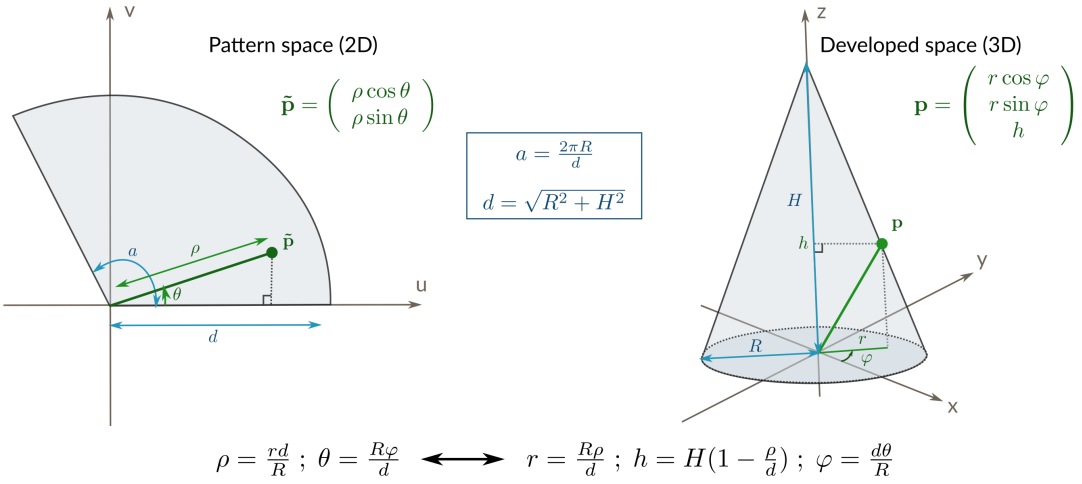


FIGURE 3.9: Transformation between a point p at the surface of a cone (right), and its corresponding point $\tilde{p}$ in the pattern space (left).

of a smooth curve, and planes for which any matching of vertices is a valid ruling. More generally, piecewise developable surfaces are unions of these 4 types of developable surfaces, as shown in the example of the hat in Figure 3.7.

As for cones, let us study the influence of the pattern-to-3D deformation on the tangent curves of the borders of a developable patch. Our protocol is the following, and illustrated in Figure 3.10. We trace two boundary 2D curves C_1, C_2 ontop of a virtual parametrized cone pattern. The pattern is then lifted in 3D with the cone transformation described in Figure 3.9. The position of the vertices in the cone pattern provide us directly which vertices are the extremities of rulings of the conical surface. We resample the curves according to rulings extremities, which gives us 2 sets of vertices $r^1_{i\,i\in\{1..R\}} \in C_1$ and $r^2_{i\,i\in\{1..R\}} \in C_2$ such that $\forall i \in \{1..R\}$, $[r^1_i r^2_i]$ is a ruling. For each pair $(i,j) \in \{1..R\}^2$, we compute :

- the angle between tangents at r^1_i and r^2_j in the pattern space
- the angle between tangents at r^1_i and r^2_j in the cone space
- the difference between those angle, in absolute value

Each of these measure, computed in degree, is displayed as a separate color map in the Figure 3.10.

For each configuration of pair of boundary curves, we consider 3 different cones.
In the first boundary configuration (Figure 3.10.top), tangents of the boundary curves on the pattern are equal for all points in the curves. In 3D however, we notice that they are only equal for vertices aligned on a ruling of the surface.
In the second case (Figure 3.10.middle), there is a parallel symmetry between the curves in the pattern, and it is aligned with the rulings, meaning extremities of rulings have

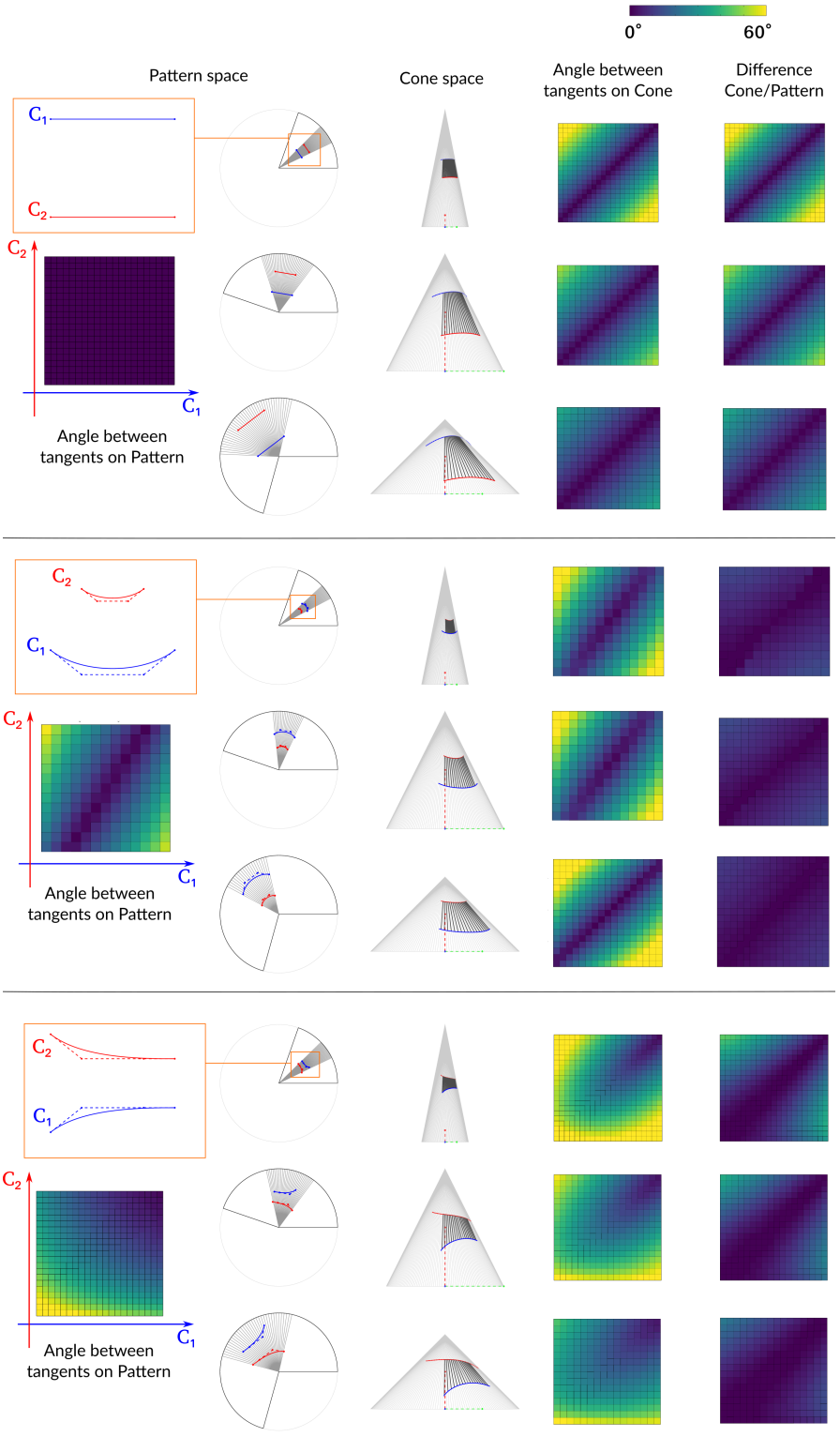


FIGURE 3.10: Experiment on the influence of boundary curves for conical patches. 3 different pairs of boundary curves presenting different geometric properties (1st col.), are drawn ontop of 3 different cone patterns (2nd col.). We study the map of angle between tangents of each curve in 2D (1st col.), and of the corresponding tangents in the curves on the cones (4th col.). We also display the difference between those maps (5th col.)

colinear tangents in the pattern. We notice that in 3D, the tangents at rulings extremities remain colinear.
In the third case (Figure 3.10.bottom), curves present a mirror symmetry, but do not share similar tangents at rulings extremities. We notice that the angle between tangents at rulings extremities is preserved in the pattern and the cone representation.

In conclusion, the cone transformation preserves the angle between tangents of pairs of vertices, if those vertices are extremities of a ruling. In particular, if the tangent at ruling's extremities are colinear in the pattern, they remain colinear in the cone. In this case, a way to find rulings within the 3D curves is to match vertices with colinear tangents. Colinearity is preserved under orthographic projection, so this method remains valid using the orthographic projection of those 3D curves.

Similarly to Ulupinar and Nevatia [UN93], we make the assumption that the surface patch is a straight generalized cone cut by two parallel planes, although we do not require these planes to be perpendicular to the cone axis. Under this assumption, the tangents of the two curves that intersect each ruling are parallel in 3D. We further assume an orthographic projection, so that these tangents are also supposed to be parallel in 2D. We propose an algorithm to match vertices within a pair of 2D curves that are the orthographic projection of the boundaries of a developable patch, using the extremities of a 2D straight silhouette as a first match.

Dynamic Time warping algorithm Our algorithm starts from the straight silhouette segment and propagates it along its adjacent curve segments, as illustrated in Figure 3.11.

Given the above assumptions, our goal is to form rulings by matching pairs of points along the two curves adjacent to the silhouette segment such that

- the rulings should intersect the curves at points with parallel tangents.
- adjacent rulings should be as parallel as possible.

While perfect parallelism of rulings is only true for straight generalized cylinders, the second objective acts as a regularization term in the presence of ambiguity of the first term, such as when the two curves we traverse contain straight segments.

We use dynamic time warping [KP01] to perform this point to point matching, which is an algorithm commonly used to compare time series, for example in the scope of gesture or speech recognition. This algorithm uses dynamic programming to minimize a cost function reflecting the quality of the point matches, see Algorithm 2 for the pseudo-code. Denoting C_i and C_j the two curves to be matched, and $\{C_i(k \in [0..N])\}$, $\{C_j(l \in$

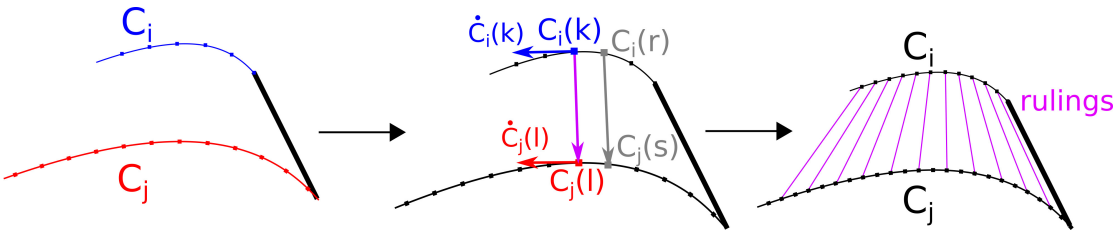


FIGURE 3.11: Our method generates 2D rulings over surface patches by propagating silhouette segments using the dynamic time warping algorithm.

Algorithm 2 Algorithm to match ruling vertices s_1, s_2 in two boundary curves by minimizing the angle between their tangents t_1, t_2

function DTWDISTANCE(s_1, s_2, t_1, t_2)
  $n_1 \leftarrow \text{length}(s_1)$
  $n_2 \leftarrow \text{length}(s_2)$
  dtw_map $\leftarrow$ double$[n_1 + 1][n_2 + 1]$
  prev_row $\leftarrow$ int$[n_1 + 1]$, prev_col $\leftarrow$ int$[n_2 + 1]$,
  ▷ Initialization
  for $i = 0..n_1$ **do**
    dtw_map $[i + 1][0] = +\infty$
  end for
  for $j = 0..n_2$ **do**
    dtw_map $[0][j + 1] = +\infty$
  end for
  ▷ Filling dtw map
  ▷ Notations : α : angle between two vectors, $r_{i,j} = \overrightarrow{s_1[i]s_2[j]}$
  for $i = 0..n_1$ **do**
    for $j = 0..n_2$ **do**
      $p_i, p_j = \underset{\substack{i' \in \{i-1,i\},\, j' \in \{j-1,j\} \\ (i',j') \neq (i,j)}}{\arg\min} \{\text{dtw\_map } [i'][j'] + \lambda\alpha(r_{i,j}, r_{i',j'})\}$
      dtw_map$[i][j] = \alpha(t_1[i], t_2[j]) + \text{dtw_map}[p_i][p_j] + \lambda\alpha(r_{i,j}, r_{p_i,p_j})$
      prev_row$[i] = p_i$, prev_col$[j] = p_j$
    end for
  end for
  ▷ Computing the best match
  path $\leftarrow \{\}$
  $i \leftarrow n_1 - 1, j \leftarrow n_2 - 1$
  while $i \geq 0$ & $j \geq 0$ **do**
    path.append$((i, j))$
    i = prev_row$[i + 1, j + 1]$-1
    j = prev_col$[i + 1, j + 1]$-1
  end while
  return dtw_map $[n_1, n_2]$, path
end function

$[0..M])\}$ successive point samples on these curves, we express the cost of matching sample $C_i(k)$ to $C_j(l)$ with the recursive expression

$$\gamma(k,l) = \alpha\big(\dot{C}_i(k), \dot{C}_j(l)\big) + \min\{\Gamma_{k,l}(k-1,l-1), \Gamma_{k,l}(k-1,l), \Gamma_{k,l}(k,l-1)\}$$

where

$$\Gamma_{k,l}(r,s) = \gamma(r,s) + \lambda\,\alpha\big(C_i(k) - C_j(l), C_i(r) - C_j(s)\big)$$

and $\dot{C}_i(k)$ denotes the tangent of curve C_i at sample k and α measures the angle between two vectors. The first term of the expression penalizes non-collinear tangents, while the recursive second term penalizes non-collinear successive rulings, see Figure 3.11.

Once we have computed all rulings using propagation along two contour curves, we reject those lying outside the patch.

We show in Figure 3.12 examples of rulings computed from sketchs of real-life objects.

3.4 3D contours optimization

Up to now, we have computed the 3D normal of the symmetry plane, a set of symmetrical points in the sketch (Section 3.2.2), and a set of rulings in the sketch (Section 3.3.2).

We are now ready to lift the 2D curves into 3D by using symmetry and developability constraints. Since the line drawings we target are traced over photographs, they may be distorted by drawing inaccuracy and weak perspective. Such distortions prevent a direct reconstruction using hard 2D positional and symmetry constraints, as done by Cordier et al. [CSMS13]. Instead, we formulate our reconstruction algorithm by defining a set of energy functions as soft constraints on the 3D coordinates of the Bézier control points, and compute a global optimal solution that can deviate from the input curves if necessary.

3.4.1 Energy formulation

Our energy function is composed of five different quadratic functions. The first three, namely projection accuracy, minimal variation and minimal foreshortening are *regularization* energies and were introduced by Xu et al. [XCS+14]. We will recall their definitions, and then introduce two new energy functions: *developability* is the key feature of our method as it enables to restore developable surface patches from the 2D sketches, while *symmetry* is necessary to recover the depth of the object. In contrast to the formulation by Xu et al. [XCS+14], all our functions are quadratic, which allows us to find a global minimum using a linear solver.

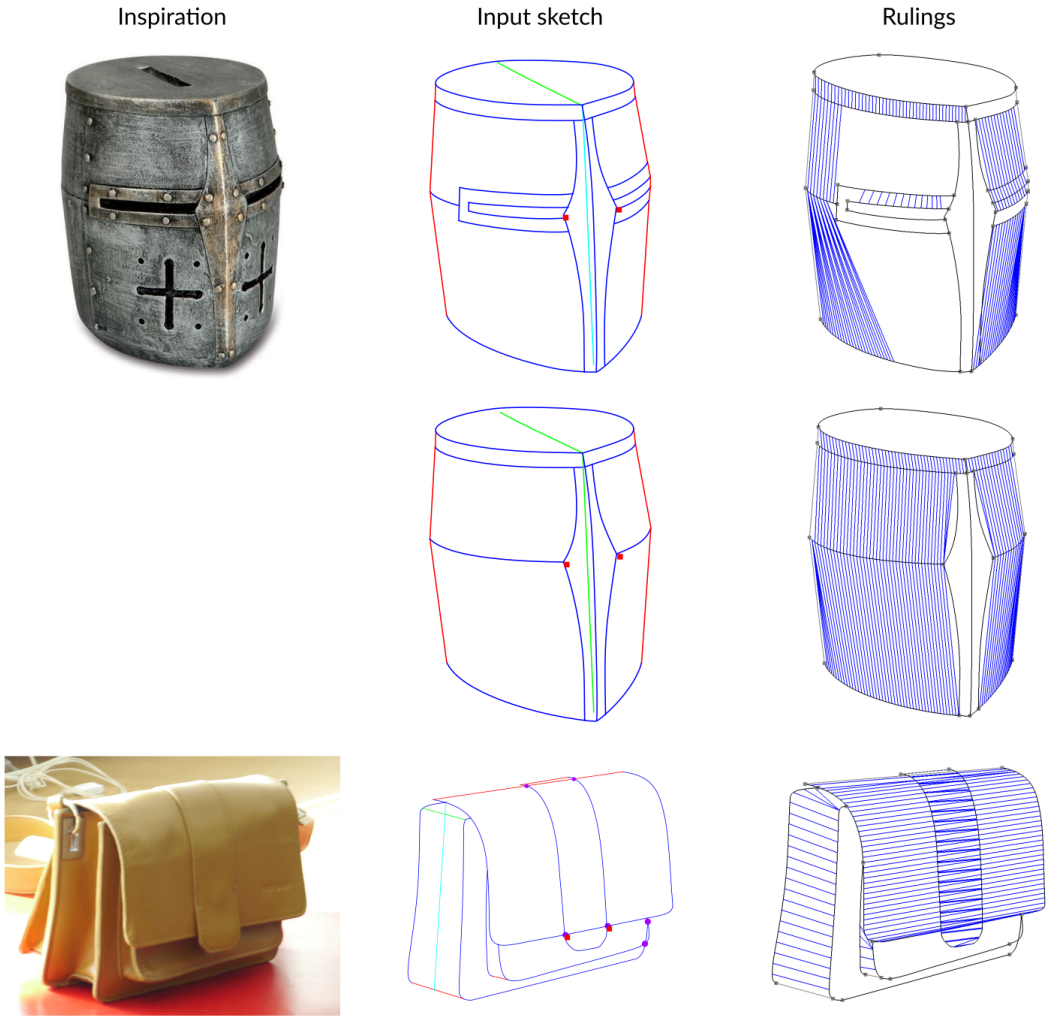


FIGURE 3.12: Example of rulings computed with our algorithm. We see that in the first version of the sketch of the helmet (1st row), the rulings computed are not accurate. If we correct the sketch to show the occluded geometry of the patch (2nd row), the result is better.

In the following, we denote B^k the k-th segment of a cubic Bézier curve, and $\{b_i^k\}_{i=0..3}$ its control points. Note that, since the curves are piecewise C^1-continuous, the three consecutive points $b_2^k, b_3^k = b_0^{k+1}, b_1^{k+1}$ are collinear. We differentiate a 2D control point, whose coordinates are provided by the user sketch, using an upper-bar notation $\bar{q}$, from the corresponding 3D control point denoted by q. The latter are the unknowns in our system.

Regularization energies These energies aim at regularizing the 3D shape of the reconstructed curves. They act by default when the points are not constrained otherwise, and help the propagation of the depth impulsed by symmetry.

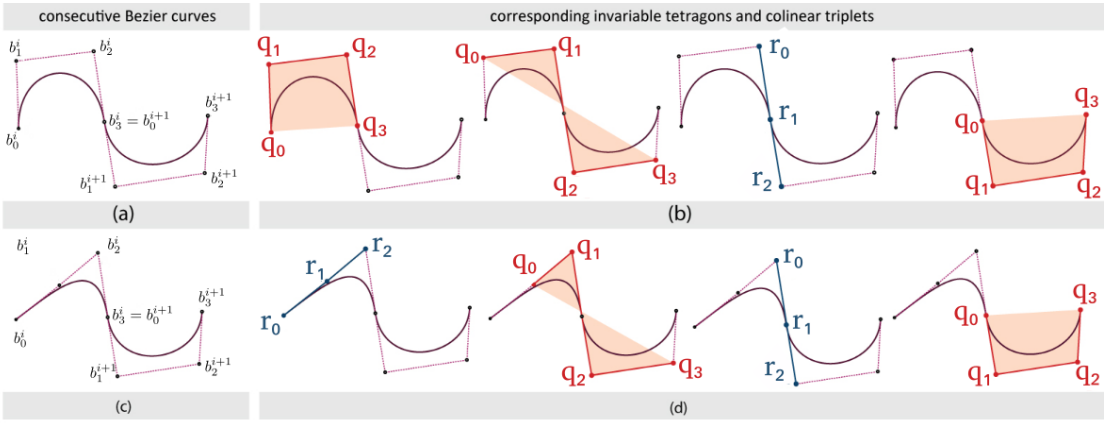
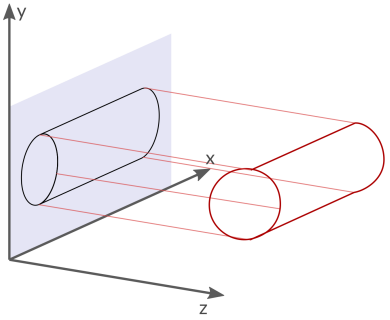


FIGURE 3.13: Invariable tetragons and colinear triplets of consecutive Bezier curves. In the general case (a), invariable tetragons are either made of the control points of a Bezier curve or made of the tangents of two successive Bezier curves (b). If one of the tetragons has three colinear points (c), we apply minimal variation on the colinear triplet made of the three colinear points (d)



Projection accuracy penalizes strong deviation of the orthogonal projection of the control points $q|_{z=0}$ from the ones given in the sketch

$$E_{\text{proj}} = \| q|_{z=0} - \bar{q} \|^2.$$

Minimal variation penalizes out of plane variations by favoring an affine relation between 4 successive non-collinear control points. These 4 points constitute either an entire Bézier segment, or cover two segments, such as b_1^k, b_2^k, b_1^{k+1}, b_2^{k+1} (cf Figure 3.13). Denoting the 4 points by $q_{i=0,\dots,3}$, the energy term on non-collinear points is defined as

$$E_{\text{minvar}} = \left\| \varphi_0 q_0 + \varphi_1 q_1 + \varphi_2 q_2 - q_3 \right\|^2,$$

where $\varphi_0, \varphi_1, \varphi_2$ are the barycentric coordinates of $\bar{q}_3$ with respect to $\bar{q}_0$, $\bar{q}_1$, $\bar{q}_2$. In addition, the minimal variation term also applies on all triplets of successive collinear points to encourage them to remain collinear in 3D, which is critical to maintain C^1-continuty between Bézier segments. Denoting such triplets by $r_{i=0,1,2}$, the energy term on collinear points is defined as

$$E_{\text{col}} = \left\| (1-\delta) r_0 + \delta r_2 - r_1 \right\|^2,$$

where $\delta, (1-\delta)$ are the barycentric coordinates of $\bar{r}_1$ with respect to $\bar{r}_0$ and $\bar{r}_2$. See a representative list of cases in Figure 3.13.

Foreshortening penalizes strong differences in the depth ($z-$coordinate) of two successive control points q_i, q_{i+1} of a Bézier curve

$$E_{\text{foreshort}} = (q_{z,i} - q_{z,i+1})^2.$$

Symmetry Minimizing this energy encourages the symmetry of two points with respect to a global symmetry plane defined by a unit normal $n = (n_x, n_y, n_z)$. Recall from Section 3.2.2 that we computed the symmetry plane, and therefore n, from user annotations.

Given two 2D symmetrical points $\bar{p}$ and $\bar{p}'$ with respect to the plane orthogonal to n, the 3D vector $t = p - p'$ and midpoint $m = p + p'$ can be computed using the formulas derived from Cordier et al. [CSPN11].

$$\begin{cases} t = & \left[(\bar{p}_x - \bar{p}'_x),\ (\bar{p}_y - \bar{p}'_y),\ \dfrac{-n_z}{n_y}(\bar{p}_y - \bar{p}'_y)\right]^T \\ m = & \dfrac{1}{2}\left[(\bar{p}_x + \bar{p}'_x),\ (\bar{p}_y + \bar{p}'_y),\right. \\ & \left.\dfrac{-1}{n_z}\left(n_x(\bar{p}_x + \bar{p}'_x) + n_y(\bar{p}_y + \bar{p}'_y)\right)\right]^T. \end{cases}$$

We then propose the following energy formulation for two symmetrical points p and p'

$$E_{\text{sym}} = \left\| \left(p - p'\right) - t \right\|^2 + \left\| \frac{1}{2}\left(p + p'\right) - m \right\|^2. \tag{3.8}$$

Note that the points p and p' are not Bézier control points, but sample points of the arc-length parameterized Bézier curves. However, each sample point can be expressed as a linear combination of the 4 unknown control points $\{q_i\}_{i=0}^3$ of the Bézier segment they belong to. The energy term E_{sym} is thus quadratic with respect to the variables q_i of our system.

Developability We recall that a developable surface is a ruled surface with constant tangent plane along each ruling (cf Definition 2.2). This is equivalent to say that the extremity points of each ruling along with their tangent vectors on the surface are coplanar. We approximate this property using the two points p_i and p'_j of curves C and C' joined by a ruling and their immediate neighbors p_{i+1} and p'_{j+1}. Given these 4 points, consistency of the tangent plane along each ruling $p_i p'_j$ is expressed using the energy functional

$$E_{\text{develop}} = \left\| \phi_0 p_i + \phi_1 p_{i+1} + \phi_2 p'_j - p'_{j+1} \right\|^2 \tag{3.9}$$

where ϕ_0, ϕ_1, ϕ_2 are the barycentric coordinates of $\bar{p}'_{j+1}$ with respect to $\bar{p}_i$, $\bar{p}_{i+1}$, $\bar{p}'_j$. Here again, note that each of these 4 points is an affine combination of the Bézier control points of the segments they belong to. E_{develop} is therefore a quadratic function of

the unknowns.

3.4.2 Optimization

Finally, the different energy terms are summed over the free variables and assembled together in a global quadratic energy function E

$$E = \omega_1 E_{\text{proj}} + \omega_2 E_{\text{minvar}} + \omega_3 E_{\text{col}} + \omega_4 E_{\text{foreshort}} + \omega_5 E_{\text{sym}} + \omega_6 E_{\text{develop}}$$

In practice, we chose $\omega_1 = \omega_3 = \omega_5 = \omega_6 = 1$ and $\omega_2 = 10^{-1}$, $\omega_4 = 10^{-8}$. The optimal solution can be efficiently computed as the solution of a linear system of equations on the unknown control points. In practice, we use an SVD decomposition in order to handle potential rank-deficiency of the associated matrix.
For a sketch described by N control points, the size of the matrix is generally of the order of $4N + 3R + 6S$, where R is the number of rulings, and S the number of pairs of symmetric points.

3.5 Developable surface generation

Our final step consists in generating a symmetrical surface bounded by the 3D contours and supported by the rulings we computed. Each surface patch is defined with a set of 3D boundary Bezier curves, and we find a triangulation that matches these boundary curves.

Patches symmetrization Even though our 3D contours optimization takes advantage of symmetric correspondences, it may result in 3D curves that are not perfectly symmetrical with respect to the global symmetry plane. Inaccuracies of the sketch, perspective weakness, and occluded parts may be the reason for imperfect symmetry. We therefore enforce perfect mirror symmetry in the network of curves and rulings as an extra step. Assuming that the object is seen from an informative 3/4 view, the positive and negative half space separated by the global symmetry plane can be respectively considered as the most reliable, and less reliable sides. Our approach consists in removing all curves and rulings belonging to the negative half space, and generating new ones using mirror symmetry from the one in the positive half space. In the specific case of a self-symmetrical patch, and therefore defined in both half spaces, some of the rulings may cross the symmetry plane. We delete such rulings and generate new ones by linking pairs of points, which are symmetrical with respect to the symmetry plane.

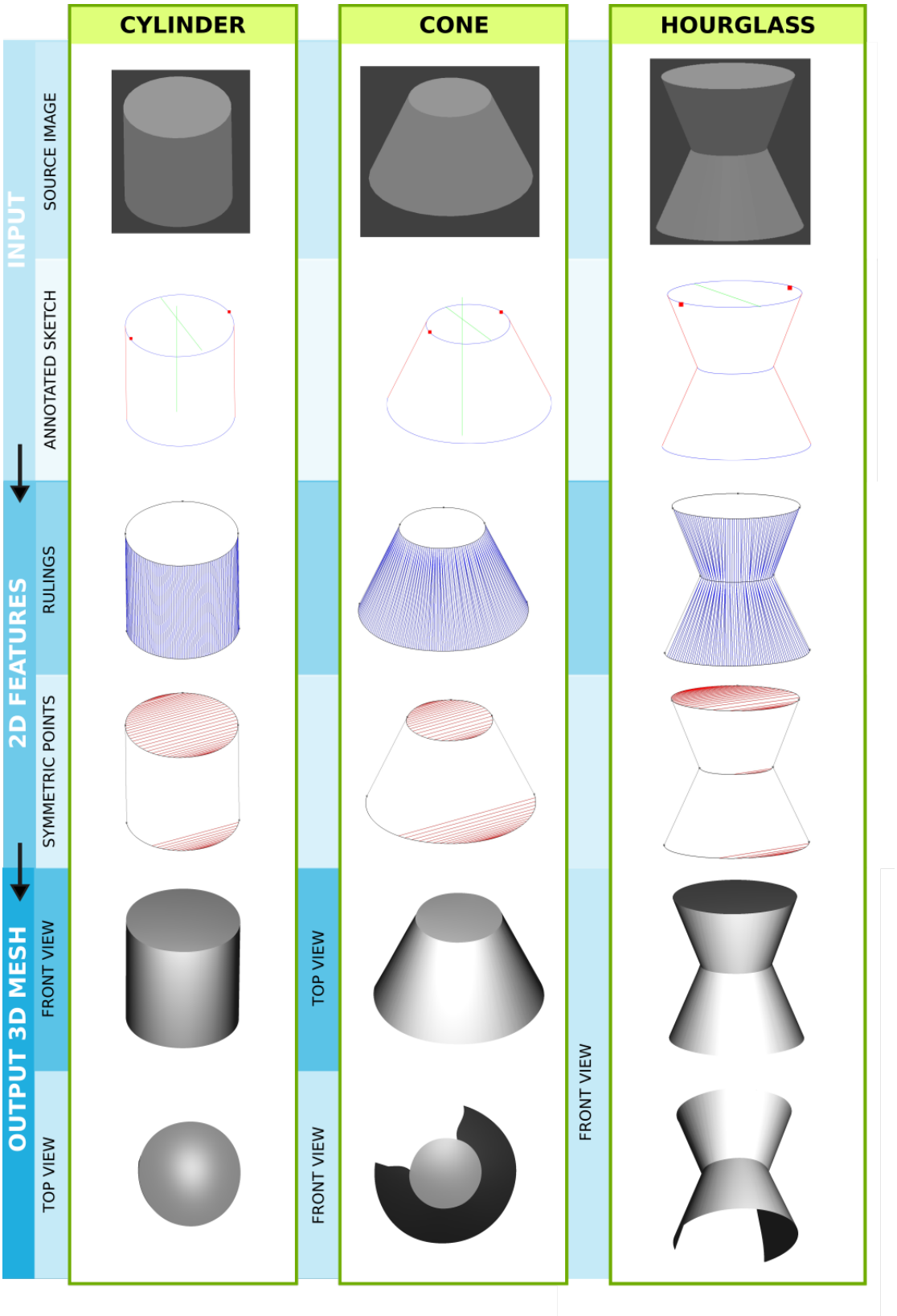


FIGURE 3.14: Results of our reconstruction algorithm on synthetic models.

Surface triangulation For each patch, we infer then infer a surface interpolating the previously computed boundary curves. If a patch contains rulings, then the corresponding 3D curves can be trivially associated to a mesh by triangulating between consecutive rulings, which does not require the introduction of interior points. If not, we generate a surface with minimal mean curvature interpolating the 3D border using the variational Laplacian approach [SHBS16, BK04]. In this case, the connectivity of this mesh is generated using a Delaunay triangulation of the 2D patch contours, where triangles are constrained to have a maximal area of 10% of the diagonal of the input image. The resulting mesh can then be improved using the developability optimization algorithm proposed by Wang and Tang [WT04]. Another method could also be to use the branch-and-bound algorithm proposed by Rose et al [RSW+07] to compute a developable surface mesh from 3D boundary curves.

As a last step, we generate a 2D pattern for each mesh, using standard parameterization algorithm minimizing stretch [SLMB05]. This works well in our case since all patches are close to developable.

3.6 Results and Validation

3.6.1 Results

We used our method to model a variety of synthetic and real-world objects.

Synthetic models For the 3 examples in Figure 3.14, we created ground-truth 3D objects representing piecewise developable surfaces (cones and cylinders). We rendered each object under an informative viewpoint using orthographic projection, and manually drew the annotations.

Rulings computed by our algorithm are displayed in blue in Figure 3.15. We can see that they are close to ground-truth rulings (displayed in orange). The 3D reconstructions also correspond well to the expected results, as shown by the top views that reveal near-perfect circular cross-sections. Note that since these models are rendered under perfect orthographic projection, we did not use the foreshortening energy, which acts as a regularization on perspective-distorted drawings.

Real-world objects We applied our method to photographs representing real-life objects. The results are displayed in Figure 3.17 and 3.18. Note that for some of the examples such as the side of the purse on the top, the sketch simplifies the geometry of the object, so that we get stronger developability constraints.
Our method finds plausible rulings for each of the examples. In particular, it succeeds in identifying cylindrical and conical parts, even when the curves linked by rulings do not

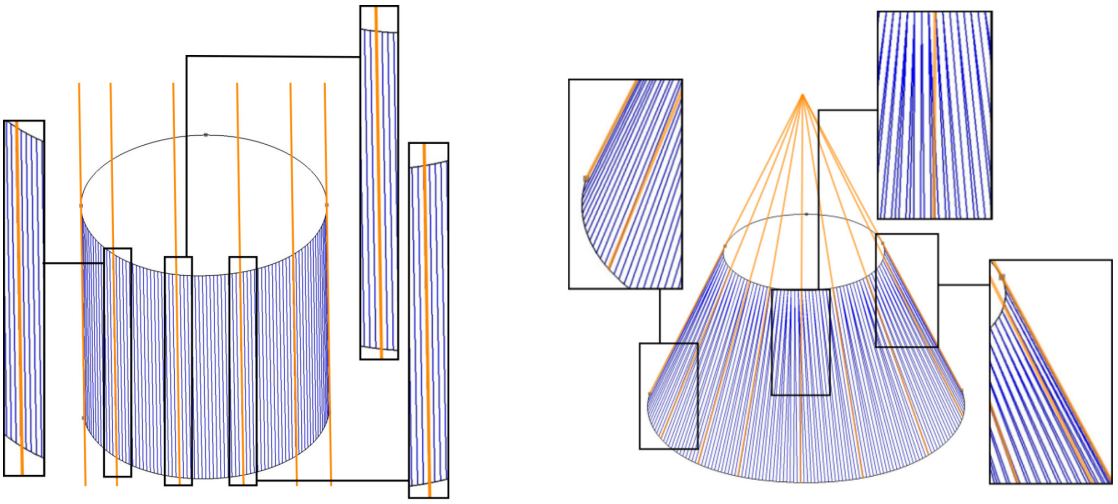


FIGURE 3.15: Evaluation of ruling propagation on two synthetic examples. Blue lines represent the rulings found by our algorithm, orange lines represent ground-truth rulings. Our rulings perfectly match ground-truth for the cylinder, and remain close to it for the truncated cone.

have the same length. As we only keep rulings that fully lie within the interior of their respective patch, our approach is also able to successfully recover partial cylindrical and conical parts within a patch. This is, for instance, the case on the curved patch of the tent on top of the door hole (see second row of Figure 3.17). This patch exhibits only 5 rulings despite a very long curved side on the left. Note that this feature also allows to handle partial occlusion associated to concavities (see for instance the top-left patch of the couch in the last row of Figure 3.18). For the specific case of a perfect cone, we added an extra annotation specifying the location of the apex, as for the right side of the tent. We also illustrate the detected symmetric correspondences, including pairs of symmetric curve (eg left side of the tent) and self-symmetric curves (eg top of the helmet).

Influence of symmetry constraint Symmetry is an important linear constraint in our method for two reasons: it provides the depth, i.e. the volume to the model, and allows to recover some occluded parts of the object from a single image. In Figure 3.16 we show the evolution of a result under an increasing number of symmetry annotations, thus allowing the user to iteratively refine the reconstructed 3D model until reaching a satisfying result.

3.6.2 Evaluation

Metrics As a mean of evaluation, we measure for each reconstructed model the developability of the patches containing rulings. A ruled surface is perfectly developable

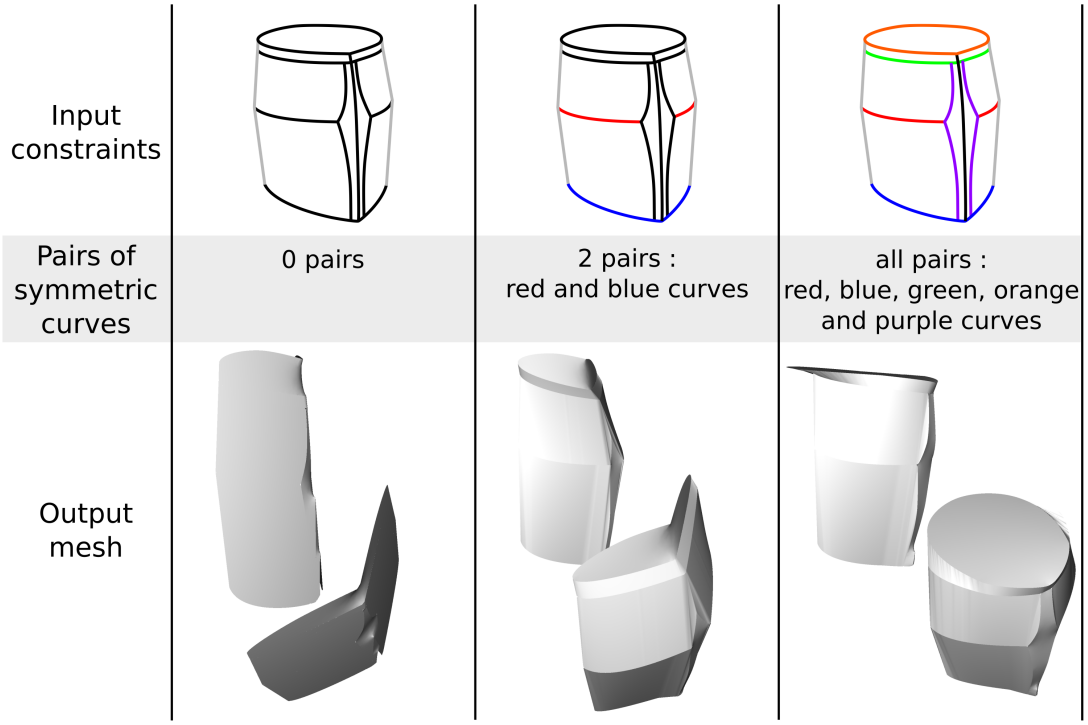


FIGURE 3.16: Evolution of the shape obtained while adding incrementally symmetry constraints to the input. The first row shows the symmetry features of the input sketch: each pair of symmetric curve has a given color, black curves correspond to non-symmetric curves, and gray lines to silhouettes. The second row displays two views of the 3D model generated by our method.

if its rulings have a constant tangent plane. We measure the developability error of a ruling by computing the angle between the normals at the ruling's extremities. We average this developability error over all detected rulings of a model. Table 3.1, last column, shows that this error varies between 6 and 16° for the objects in Figures 3.17 and 3.18. The computational time varies between 5 and 30 seconds depending on the number of control points in the curve network and the number of rulings (Table 3.1, second column). Such performances allow an interactive workflow where users can quickly visualize the reconstructed 3D shape and add missing annotations on the photo to improve it if necessary.

Analysis of the developability constraint The main novelty of our approach resides in the new developability constraint. We now evaluate its impact on 3D reconstructrion.

We compared our approach with a downgraded version where we removed the term representing developability by setting ω_6 to 0. Without this energy term, the average developability error on the resulting model increases. For example, for the model purse

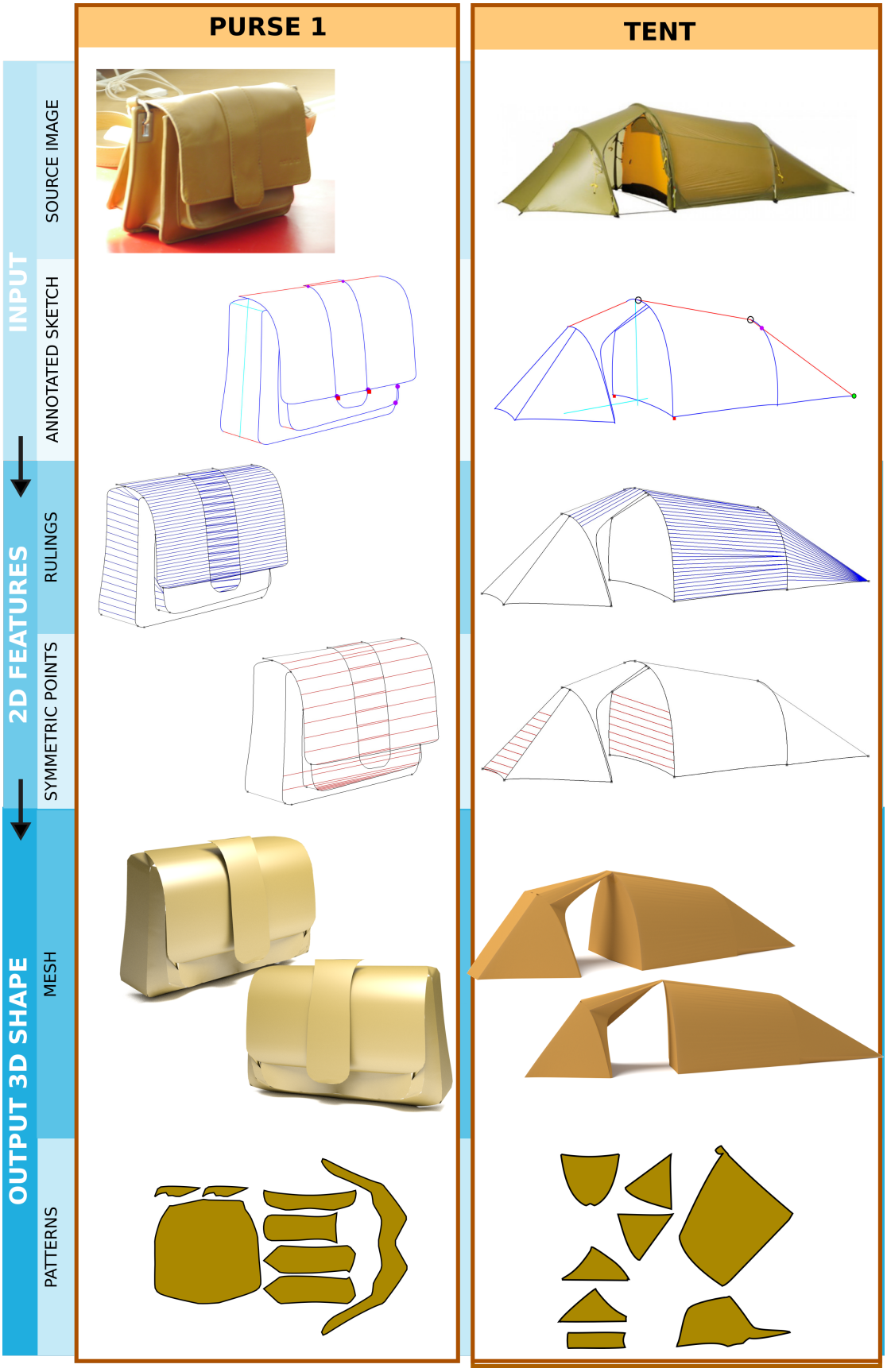


FIGURE 3.17: Results on real-world examples.

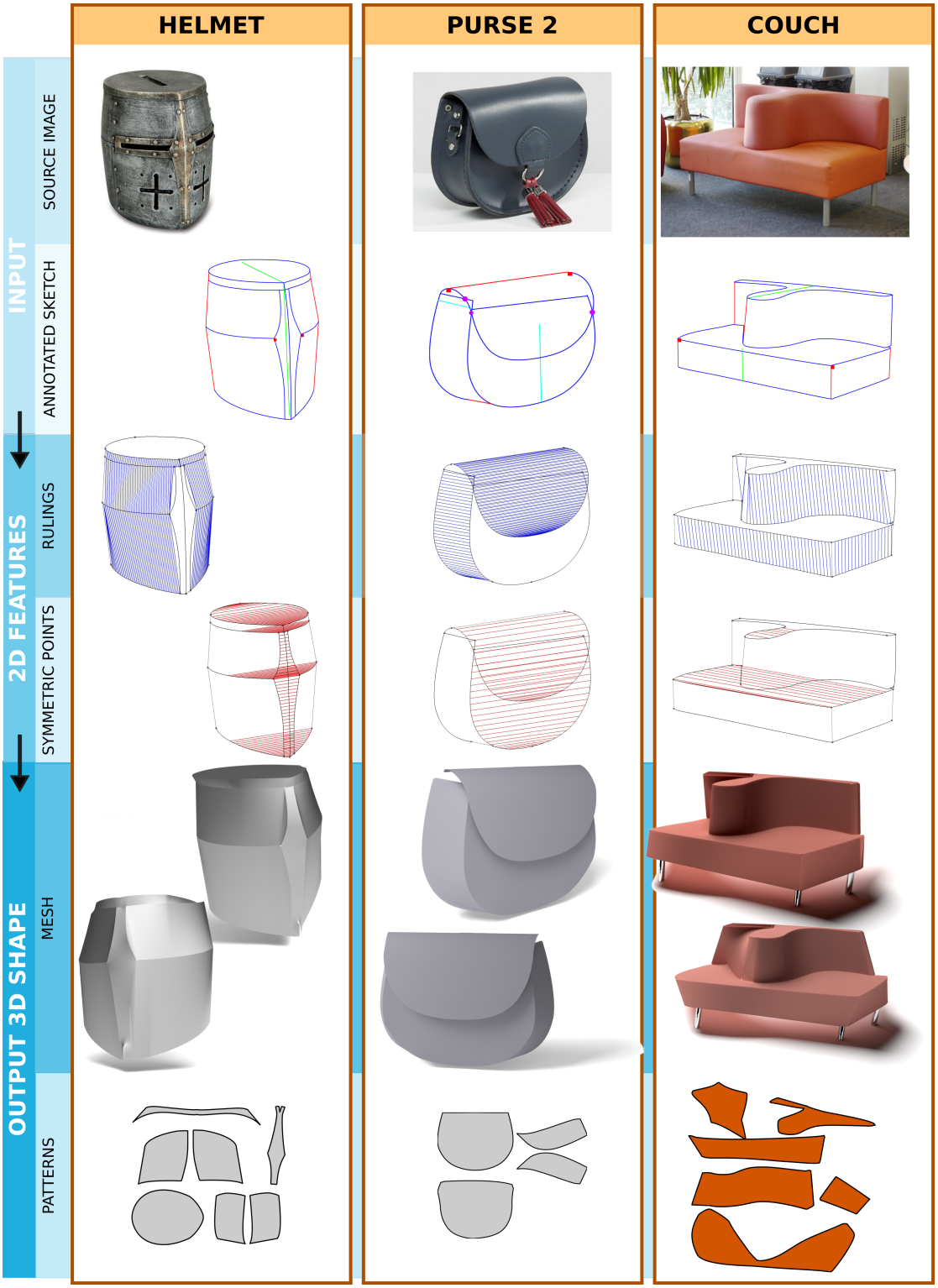


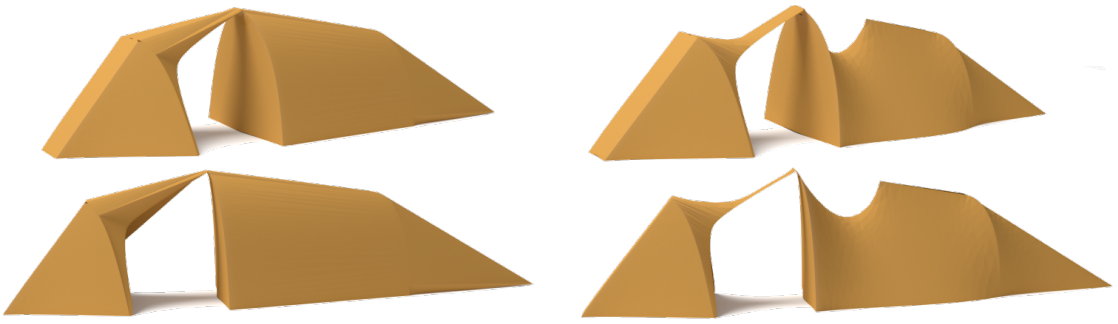
FIGURE 3.18: Results on real-world examples.

model	time	control points	rulings	pairs of sym. pts	patches (cycles)	avg error develop.
purse 1	28s	119	135	27	7	6.61°
helmet	30s	100	165	93	9	12.49°
purse 2	7.8s	58	112	43	3	9.65°
tent	4.7s	61	26	21	6	16.33°
couch	19s	100	104	18	6	13.84°

TABLE 3.1: Dimension, computational time and developability error of the examples in Figures 3.17 and 3.18

1, developability error goes from 6.61° with $\omega_6 = 1$ to 10.22° with $\omega_6 = 0$. We made a similar observation for all other tested models.

We can also note the positive influence of the rulings in the resulting surfaced model. For example, the surface tent model could not be correctly reconstructed using only minimal surfaces, as shown in Figure 3.19.



(a) surface reconstructed using rulings (b) surface reconstructed without rulings

FIGURE 3.19: Example of reconstructed surfaces with (a) and without (b) the use of 3D rulings.

Failure cases In Figure 3.20 we present two failure cases of our system. Both examples satisfy the assumptions of near orthographic view, global mirror-symmetry of the object and piecewise developability. Our system robustly computes a set of 3D curves and fits the patches.

However, the contours at the top of the cap only consist of patch boundaries and do not exhibit silhouettes. Thus, no rulings are computed by our approach, which results in non-developable surface. Moreover, the visor of the cap contains conical sections, but they do not satisfy our hypothesis of being cut by two parallel planes, thus the criteria of collinear tangent at rulings extremities presented in Section 3.3.2 does not hold. The rulings extraction is not guaranteed to work in these cases of developable patches, even though the computed rulings in this case remain plausible.

The bag example lacks symmetry input, which is an essential constraint to inflate the volume of the model. Similar to Figure 3.16-left, middle, the optimization will lead to a

flat model. One may overcome these special cases, by allowing the user to e.g increase symmetry annotations with some additionally sketched patch boundaries.

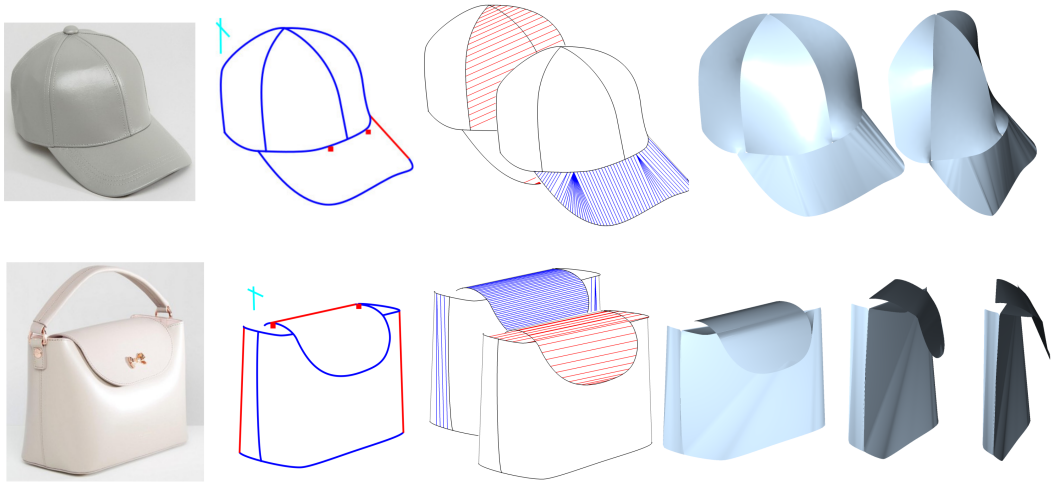


FIGURE 3.20: Examples of failure cases that do not exhibit sufficient symmetry and developability features. Note that our system is still able to provide a solution, but the result lacks volume. Note also we present the resulting meshes without the symmetrization step in these examples.

3.7 Conclusion and Discussion

We have presented the first method enabling the reverse-engineering of symmetric, developable products from a single photo. Our approach assumes a simple orthographic projection. While our least-squares optimization approach tolerates minor perspective distortions as demonstrated by our results, strong perspective effects can yield distorted surfaces. Accounting for perspective would require a more complex formulation of the re-projection error. In addition, perspective should also be taken into account when searching for symmetric correspondences. Figure 3.21 illustrates how annotating two parallel lines on a symmetric model could help account for vanishing points during 2D analysis.

Since our system only takes a single photograph as input, we cannot reconstruct all occluded parts of the model. Completing the model by simple back-facing symmetry may still lack realism, since some occluded parts, such as the back of the helmet in Figure 3.18 would not be completed as expected. An interesting direction for future research would be to integrate our developability constraints into a multi-view modeling system where the object would be captured from a few, complementary viewpoints.

In this chapter, we only account for the reconstruction of a smooth C^2 developable surface, which are very constrained by their rulings. In particular, such surface can not

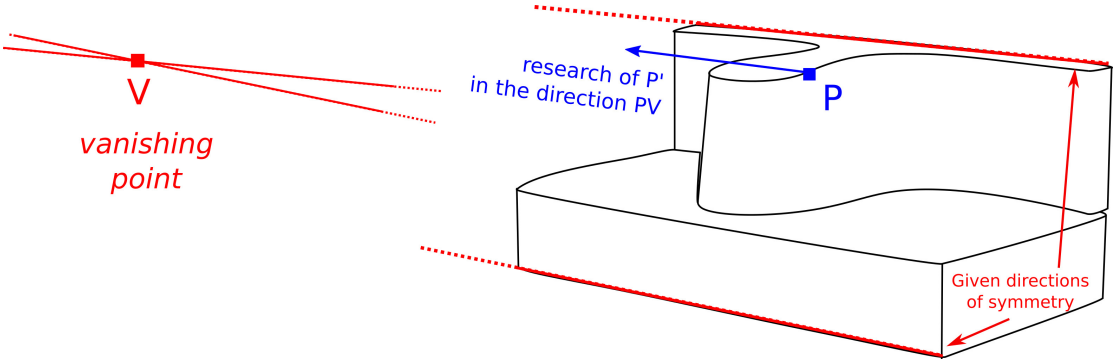


FIGURE 3.21: Finding the vanishing point of the image from two annotated red lines would improve the search for the point symmetric to P.

contain interior folds or creases. Designing developable surfaces containing interior curve and creases implies to account for non-C^2 developable surfaces. In this case, the ruling representation is not adapted anymore. In the next chapter, we focus on the design of folds using a 2D sketch. These folds will then be used in our 2D-to-3D garment modeling algorithm (Chapter 5).

Chapter 4

Sketch-based modeling of tubular folds

Folds are an important feature of garment design. Their geometry, determined by the material, the cloth's pattern shape, and the body of the dressed character, show great variety and is significant of the style of a garment.

There are different categories of folds: the ones issued from stretching or compression, and the ones caused by gravity (see examples in Figure 4.1). Some of them are sewed directly in the fabric, and others are the result of the dressed character's position, or location of the hanging points. In this chapter, we study tubular, or axis-aligned folds, and their representation in 2D fashion sketches. Tubular folds are folds that appear on a garment due to gravity, looseness of the fabric in the body and an excess of cloth in

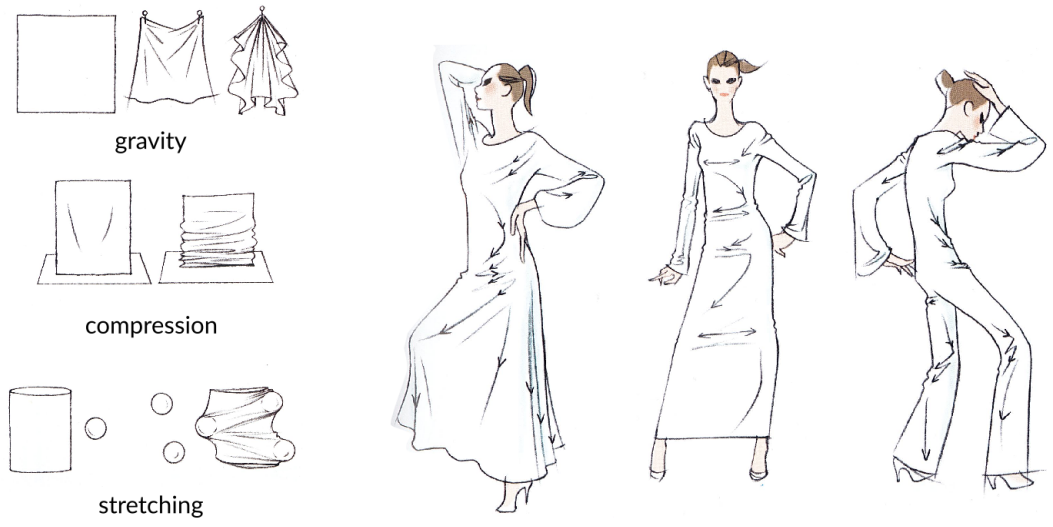


FIGURE 4.1: Illustration of different types of folds, from [Wat09]. Additionnally to the pattern cutting and chosen material, the folds' appearance and location on garments are conditionned by the pose of the character wearing it.

the loose border part of the garment. For example, you can see tubular folds appearing in the Figure 4.1-left character, but none of the middle character of the same figure. While, the garment looks the same, the position of the dressed character makes the garment tight in the second case, which discards the appearance of tubular folds. Other examples of tubular folds are illustrated in the Figure 4.2. We propose an approach to add tubular folds in a surface using a unique 2D sketch as guide.

Design of virtual folds and wrinkles has been the topic of several studies in Computer Graphics' litterature. While the traditional method to model folds using physics-based simulation is constrained by physics-based parameters related to the fabric's material, alternative geometry-based methods were proposed.

Decaudin et al. [DJW+06] presented parametric models for several categories of folds: axis-aligned, compression, bending and twisting folds. Their model, based on cardinal spline surfaces, is a good basis for modeling regular folds, but is not enough to represent the variety of existing folds. Sketch-based modeling approaches for folds were also proposed, some of them asking the user to indicate a radius and direction value for each of the folds [TWB+07], others using the sketched hemline to classify the fold's shape in one specific category, while the actual shape of the folds is obtained by physics-based simulation [LSGV18]. Closer to our work, Robson et al. [RMSC11] proposed a sketch-based modeling interface where the user draws the folded border directly on top of a view of the character to be dressed. Knowing the calibration of the virtual camera, and with the assumption that the border is planar, they invert the perspective matrix to find the depth of the curve and deform an existing mesh garment so that it folds accordingly.

We propose an approach to generate, from arbitrary sketches, a fold shape representation that can be transferred to any parametric surface. We show how our approach can be used for a large variety of tubular folds and how it adapts to different surfaces. After studying how such folds are sketched in fashion illustration, and how this representation relates to their 3D geometry (Section 4.1), we propose an algorithm to build a normalized fold distribution curve from an arbitrary sketch (Section 4.2). Finally, we present in Section 4.3 a method to use such fold curve to add tubular folds in an existing surface.

4.1 Representing tubular folds

Tubular folds can present various shapes, and different representations in sketches. In this section, we first describe most common 2D features depicting tubular folds in sketches (Sec. 4.1.1). We then study the 3D geometry of cloth with tubular folds, and discuss models of virtual camera that could link those two representations (Sec. 4.1.2).

FIGURE 4.2: Technical drawings (top row) and free illustrations (bottom row) of garments with different axis-aligned folds, from specialized litterature [NG09, Wat09].

4.1.1 Tubular folds in fashion illustration

Axis-aligned folds, or tubular folds, appear on cloth subject to the effect of gravity, and sustained by one or several hanging points. Hanging points can be seamed in the fabric, as the skirt on Figure 4.2b, or caused by the position of the character wearing the garment, such as the knee and right hand in the example of Figure 4.3. The representation of these folds varies subjectively from one artist to another (see Figure 4.2).

We focus here on line representations, and therefore will not account for shading and color information. The most common way of drawing the shape of axis-aligned folds is to draw inner silhouette lines, giving the direction of the folds along the fabric, and the hemline curve, which represents the projection of the free border of the surface on the image plane (see an example in Figure 4.3).

Inner silhouettes As discussed in Section 2.2.1, silhouette lines are the projection of the *rim*: a curve on the surface where the surface turns smoothly away from the

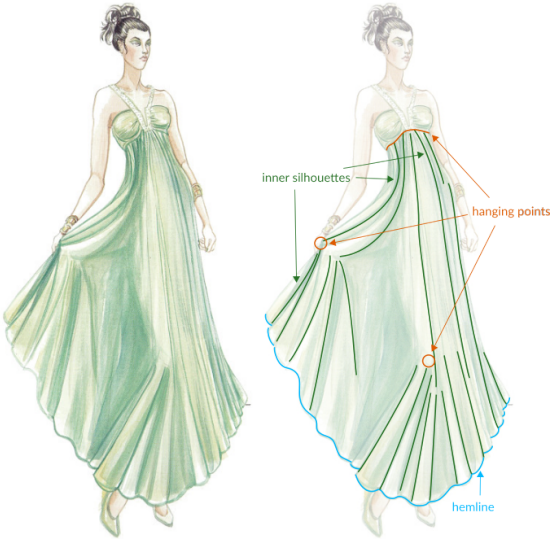


FIGURE 4.3: Example of sketch from [NG09] representing tubular folds, created by several hanging points. We can extract line information representing the folds' shape, such as the folded hemline and inner silhouettes.

viewer. In the context of tubular folds, as seen in Figure 4.4.(a), inner silhouette can be the projection of arbitrary parts of the folds in the surface. Inner silhouettes are commonly represented with shading effects or variation of texture, as in Figure 4.2.(e,f,g). They can more or less easily be distinguished depending on the depth of the fold (see Figure 4.4.(c,d)), or the curvature of the fold undulation (see Figure 4.2.(b,d)), which makes them ill suited to be represented as line drawings.

Inner silhouette are in the direction linking hanging points to free borders, and they can be used to extract location of the hanging points in the garment. For example, in Figure 4.3, many silhouette lines converge into the location of the hidden knee. Twisted folds could also be recognized using the direction of inner silhouettes.

Inner silhouette represent how the folds are propagating from a hanging point to a border. In some cases, the cloth is not folded at the location of the hanging point, and the undulation appears as we go closer to the free border, as in the example of Figure 4.4. In other cases, the folds are seamed directly on the fabric and are uniformly deforming the cloth from the hanging point to the free border, as in the example of Figure 4.2.

Exploiting the drawn inner silhouettes to build a normalized representation of tubular folds raises different challenges. First, while their direction and length depicts the way of propagation of the folds from one border to another, it highly depends on the pose and morphology of the character wearing it. Second, representation of inner silhouette is unstable from one sketch to another. They are not reliable enough to count nor to infer width of folds in a garment.

Hemline The second important feature for the representation tubular folds is the projection of the free border, which we call *hemline* (see an example in Figure 4.3).

While this curve displays significant information on the number, width and depth of the folds, its shape is also influenced by the shape of the corresponding cloth pattern, and the motion of the cloth (cf Figure 4.2.(a,e)). Exploiting the hemline to build a normalized representation of the fold distribution implies to decorrelate inflexions due to folds from the ones due to the pattern shape and cloth motion. Such decorrelation using only fashion sketches is an ill-posed problem, and would require additionnal information on either the pattern's shape, or an associated technical drawing. Therefore, in this work, we will assume that inflexion points in the hemline are only due to folds, and not to specific cuttings of the patterns.

Another challenge is that tubular folds often causes self occlusions in the garment which would lead to discontinuous incomplete representation of the hemline, in particular for deep folds, such as in Figure 4.2.(g). We discuss how we overcome this issue in Section 4.2.3.

4.1.2 Modeling folds in 3D

We have seen the important features representing tubular folds in a 2D sketch. We now discuss the representations of the underlying 3D shape that the sketch represents. After presenting our choice of 3D representation for tubular folds in a garment, we discuss how to interpret the shape of the 2D sketched hemline to model them.

Garment 3D representation In Figure 4.4 we display some examples of virtual garments with tubular folds obtained using a traditional modeling system with physics-based simulation. We observe that the free boundary curve contains undulations that are representative of both the global shape of the garment, and the shape of the tubular folds. In those garments, these undulations are the most visible in this part of the garment, while their are fading out as we march in the opposite direction of gravity.

From these observations, we chose to represent the folded surface in a two-layered model: a smooth primitive surface representing the cloth without folds and an additionnal fold perturbation curve. We aim at using the sketched hemline to perform reconstruction of folds, and transfer to different surfaces. We need a fold curve representation that is decorrelated to the shape of the smooth primitive, and that can be easily manipulated, in order to perform fold transfer.

As displayed in Figure 4.4, we observe that we can represent accurately various folded boundary curves using cubic Bézier splines with few control points. The fold spline is decomposed into Bezier segments representing individual folds, alternately concave and convex. This representation is still dependant of the shape of the smooth primitive. We

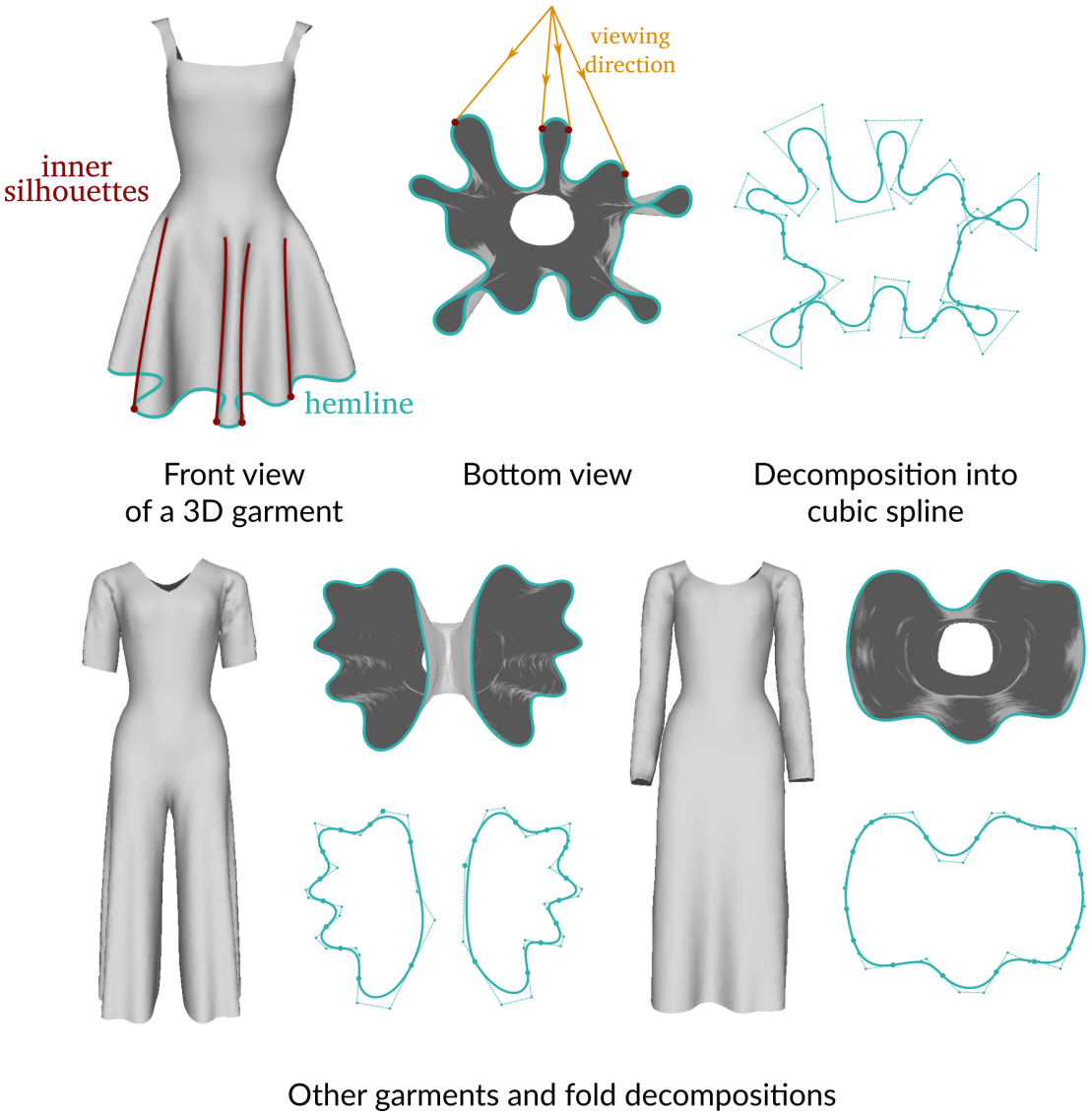


FIGURE 4.4: Some examples of 3D garments modelled using physics-based simulations. From the combination of inner silhouettes and hemline curve, we distinguish tubular folds on the garments. The undulations of the free border, visibles in bottom view, and representatives of the folds' shape, can be represented with cubic Bézier splines.

will denote as *directrix* curve the image in the sketch of the boundary of this smooth primitive. We look for a representation of the folds that is independent from this directrix curve.

Influence of perspective in sketched hemlines We first try a naïve approach to compute such directrix-independant fold representation. Here, we represent the directrix as a polynomial curve of degree 2. We obtain its shape by fitting the folded hemline into such polynomial curve. As discussed in previous section, we consider the folded hemline as a perturbation of this smooth directrix. We measure this perturbation along the directrix curve to obtain a directrix-independant representation of the folds.

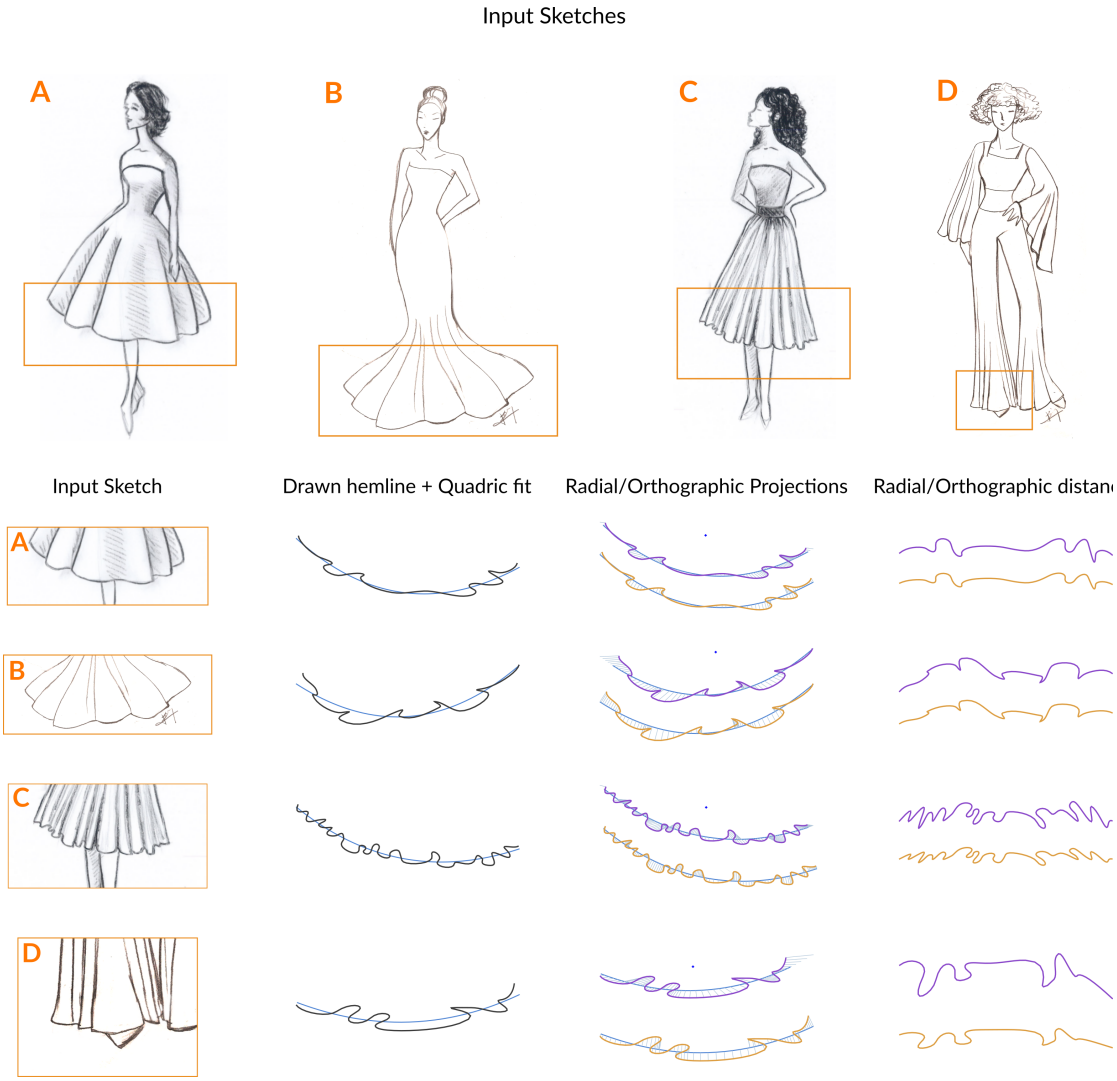
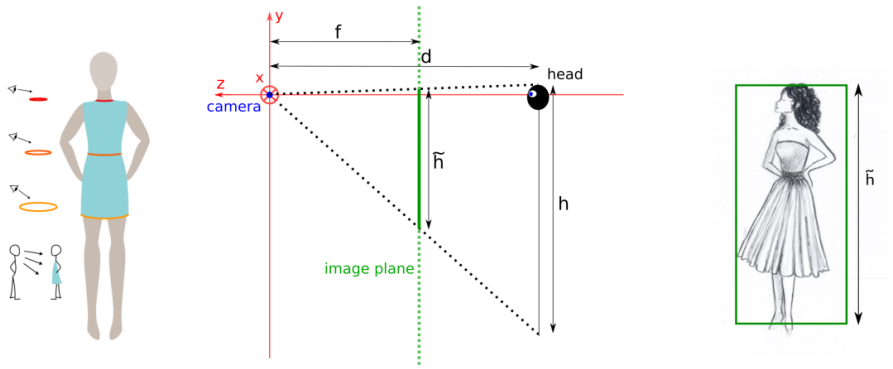


FIGURE 4.5: Study of folded hemline drawn on 2D sketches. For each hemline, we approximate a smooth directrix curve (gray). We then compute the radial and orthogonal distances from the hemline to this directrix (respectively purple and yellow).

We compute two distance measures on the sketched folded hemlines (illustrated in Figure 4.5):

- radial distance: in a radial direction with respect to the directrix center, computed as the middle point of the curve's extermities (in purple in Figure 4.5).
- orthographic distance: in the normal direction of the directix curve (in yellow in Figure 4.5).

We notice some distortions in these representations. In particular, the folds appearing at the extremities of the smooth curve seem horizontally squashed in compared to the ones in the center of the curve. We also notice that the radial distance model provides in general deeper folds than the orthographic one, in particular at curve extremities.



(a) Viewpoint in fashion illustration (b) Perspective model - side view (c) Example of sketch

FIGURE 4.6: Illustration of our perspective camera model, inspired by the classical viewpoint taken in fashion illustration (a). The model is based on a virtual camera looking at a character in the z-direction (b). We use this perspective model along with assumptions on the values of d and h to interpret a single 2D sketch (c).

To overcome these distortions, we need to account for the perspective effect in the sketch.

We propose a virtual camera model, based on fashion illustration specialized litterature [NG09, Wat09]. In this model, the camera is positionned at eye's height, and looking in an horizontal direction at the eye of the character (see illustration of the virtual camera model in Figure 4.6). We assume that the hemline is contained in a plane which is orthogonal to the image plane, meaning for which the $z-$component of the normal is 0. We fix the camera/character distance to $d = 250cm$, and assume the model's height is approximately $h = 170cm$.

We propose an approach to model folds on a parametric surface using a sketched hemline representing possibly irregular folds. Our algorithm uses this perspective model to interpret the sketched hemline and generate a normalized fold distribution curve (Section 4.2). The folded garment is then obtained by applying the fold distribution on the free border of the surface, and propagating them linearly to the hanging border (Section 4.3).

4.2 Sketch-based modeling of folded hemline

We work on fold distributions represented in different spaces, all summarized in Figure 4.7. The input sketch space (a) is a 2D space of curves, that is interpreted as the perspective projection of a dressed character. The boundary curve space (b) is a top view of the planar folded boundary of the garment, in local 2D coordinates. The flat fold distribution space (c) is a 2D shape-independant space representing the fold distribution. The surface space is the space of the 3D folded surface.

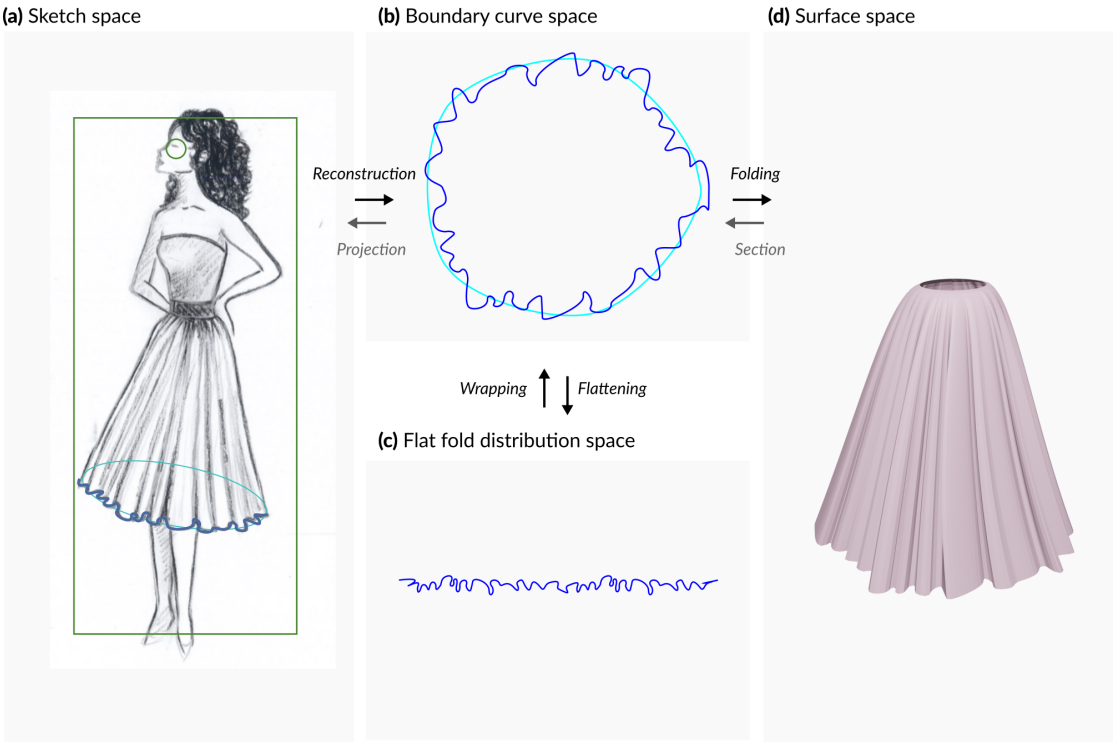


FIGURE 4.7: Overview of the different spaces representing tubular folds we manipulate in our method. Transitions from one space to another are written in black if described by an algorithm in this chapter, in gray if not: sketch-based reconstruction in Section 4.2.1, flattening in Section 4.2.2, wrapping in Section 4.3.2 and folding in Section 4.3.1

In this section, we present an algorithm to compute a fold distribution curve from an input sketch representing the projection of a folded hemline ((a) → (b) → (c) in the Figure 4.7).

4.2.1 Reconstruction algorithm

First, our algorithm uses the perspective model presented in Section 4.1.2 to reconstruct the depth of a folded hemline (from (a) to (b) in Figure 4.7). Additionnally to the projected folded hemline, our algorithm requires the user to draw the directrix as a 2D convex closed curve, representing the rough shape of a non-folded version of the garment's free border, along with the bounding box of the dressed character and the location of its eye (see Figure 4.6-c for an example). As Robson et al. [RMSC11], we assume that the hemline is planar, and lies in a plane $\mathcal{P}$ orthogonal to the image plane. The first step of the process is to estimate $\mathcal{P}$. Then, we apply a reverse-perspective computation to get the coordinates of the curve in this plane.

Plane estimation Let's denote as (Ω, n) the origin and normal of the $\mathcal{P}$ plane.

$$\Omega = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \qquad\qquad n = \begin{pmatrix} \sin\varphi \\ \cos\varphi \\ 0 \end{pmatrix}$$

Our algorithm starts by computing a 2D ellipse that fits at best the 2D convex curve annotated by the user, the *directrix*. It is supposed to belong in the border's plane. In 3D, the ellipse follows an equation of the form:

$$\forall t \in [0, 2\pi],\ \mathcal{E}(t) = \begin{pmatrix} a\cos\varphi\cos(t) + x_0 \\ -a\sin\varphi\cos(t) + y_0 \\ -b\sin(t) + z_0 \end{pmatrix} \tag{4.1}$$

Its image under our perspective transformation is:

$$\tilde{\mathcal{E}}(t) = \frac{f}{b\sin(t) - z_0} \begin{pmatrix} a\cos\varphi\cos(t) + x_0 \\ -a\sin\varphi\cos(t) + y_0 \end{pmatrix} \tag{4.2}$$

If we denote as A, B, C, D, F the algrebraic coefficients describing $\tilde{\mathcal{E}}$ as:

$$A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2 + 2D\tilde{x} + 2F\tilde{y} + 1 = 0 \tag{4.3}$$

Then we show that:

$$\begin{aligned} A &= \frac{a^2(z_0^2 - b^2)\sin^2\varphi + b^2y_0^2}{a^2f^2(x_0\sin\varphi + y_0\cos\varphi)^2} \\ B &= \frac{a^2(z_0^2 - b^2)\cos\varphi\sin\varphi - b^2x_0y_0}{a^2f^2(x_0\sin\varphi + y_0\cos\varphi)^2} \\ C &= \frac{a^2(z_0^2 - b^2)\cos^2\varphi + b^2x_0^2}{a^2f^2(x_0\sin\varphi + y_0\cos\varphi)^2} \\ D &= \frac{z_0\sin\varphi}{f(x_0\sin\varphi + y_0\cos\varphi)} \\ F &= \frac{z_0\cos\varphi}{f(x_0\sin\varphi + y_0\cos\varphi)} \end{aligned} \tag{4.4}$$

Let $c_\varphi = \cos\varphi$, $s_\varphi = \sin\varphi$.

$$\begin{aligned} &(a^2f^2(x_0s_\varphi + y_0c_\varphi)^2)(A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2) \\ &= a^2(z_0^2 - b^2)(\tilde{x}^2s_\varphi^2 + 2\tilde{x}\tilde{y}c_\varphi s_\varphi + \tilde{y}^2c_\varphi^2) + b^2(\tilde{x}^2y_0^2 - 2\tilde{x}\tilde{y}x_0y_0 + \tilde{y}^2x_0^2) \\ &= a^2(z_0^2 - b^2)(\tilde{x}c_\varphi + \tilde{y}s_\varphi)^2 + b^2(\tilde{x}y_0 - \tilde{y}x_0)^2 \end{aligned}$$

Replacing $\tilde{x}$ and $\tilde{y}$ with their corresponding expression from Eq. (4.2) leads to

$$\begin{aligned}
&(a^2f^2(x_0s_\varphi + y_0c_\varphi)^2)(A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2) \\
&= \frac{a^2f^2}{(b\sin(t) - z_0)^2}(z_0^2 - b^2\sin^2(t))(x_0s_\varphi + y_0c_\varphi)^2 \\
&\Rightarrow A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2 = \frac{z_0^2 - b^2\sin^2(t)}{(b\sin(t) - z_0)^2}
\end{aligned}$$

$$\begin{aligned}
&a^2(z_0^2 - b^2)(\tilde{x}s_\varphi + \tilde{y}c_\varphi)^2 \\
&= \frac{f^2a^2(z_0^2 - b^2)}{(b\sin(t) - z_0)^2}\left((ac_\varphi\cos(t) + x_0)s_\varphi + (-as_\varphi\cos(t) + y_0)c_\varphi\right)^2 \\
&= \frac{a^2f^2}{(b\sin(t) - z_0)^2}(z_0^2 - b^2)(x_0s_\varphi + y_0c_\varphi)^2
\end{aligned}$$

and

$$\begin{aligned}
&b^2(\tilde{x}y_0 - \tilde{y}x_0)^2 \\
&= \frac{f^2b^2}{(b\sin(t) - z_0)^2}((ac_\varphi\cos(t) + x_0)y_0 - (-as_\varphi\cos(t) + y_0)x_0)^2 \\
&= \frac{a^2f^2}{(b\sin(t) - z_0)^2}b^2\cos^2(t)(y_0c_\varphi + x_0s_\varphi)^2
\end{aligned}$$

Therefore

$$\begin{aligned}
&(a^2f^2(x_0s_\varphi + y_0c_\varphi)^2)(A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2) \\
&= \frac{a^2f^2}{(b\sin(t) - z_0)^2}((z_0^2 - b^2) + b^2\cos^2(t))(x_0s_\varphi + y_0c_\varphi)^2 \\
&= \frac{a^2f^2}{(b\sin(t) - z_0)^2}(z_0^2 - b^2\sin^2(t))(x_0s_\varphi + y_0c_\varphi)^2
\end{aligned}$$

and

$$A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2 = \frac{z_0^2 - b^2\sin^2(t)}{(b\sin(t) - z_0)^2}$$

On the other side,

$$\begin{aligned}
&f(x_0s_\varphi + y_0c_\varphi)(2D\tilde{x} + 2F\tilde{y}) = 2z_0(\tilde{x}s_\varphi + \tilde{y}c_\varphi) \\
&= \frac{2z_0f}{b\sin(t) - z_0}(x_0s_\varphi + y_0c_\varphi) \\
&\Rightarrow 2D\tilde{x} + 2F\tilde{y} = \frac{2z_0}{b\sin(t) - z_0}
\end{aligned}$$

Finally

$$\begin{aligned} &A\tilde{x}^2 + 2B\tilde{x}\tilde{y} + C\tilde{y}^2 + 2D\tilde{x} + 2F\tilde{y} + 1 \\ &= \frac{z_0^2 - b^2\sin^2(t)}{(b\sin(t) - z_0)^2} + \frac{2z_0}{b\sin(t) - z_0} + 1 = 0 \end{aligned}$$

We derivate these equations to obtain the 3D parameters of the plane in terms of the coefficients of the projected ellipse $\tilde{\mathcal{E}}$:

$$\tan\varphi = \frac{D}{F} \qquad , \qquad \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \frac{z_0}{(\tilde{x}_0 D + \tilde{y}_0 F)f} \begin{pmatrix} \tilde{x}_0 \\ \tilde{y}_0 \end{pmatrix} \tag{4.5}$$

where $\begin{pmatrix} \tilde{x}_0 \\ \tilde{y}_0 \end{pmatrix} = \frac{1}{B^2 - AC} \begin{pmatrix} CD - BF \\ AF - BD \end{pmatrix}$ is the origin of the projected ellipse $\tilde{\mathcal{E}}$.

Using the assumption that the origin of $\mathcal{E}$ lies in the image plane ($z_0 = -d$), we can compute the plane's normal and origin.

Reverse perspective Once the plane $\mathcal{P}$ is known, we can reconstruct the 3D coordinates of all points lying in it from their projection in the sketch. Let $\tilde{p} = (\tilde{x}, \tilde{y})$ image of a point $p = (x, y, z)$ on a plane (Ω, n). Then its 3D coordinates verify:

$$\begin{aligned} & (p - \Omega) \cdot n = 0 \\ \Rightarrow \quad & (\tilde{x}\ z/f\, n_x + \tilde{y}\ z/f\, n_y + z\, n_z) - \Omega \cdot n = 0 \\ \Rightarrow \quad & z = f\, \frac{\Omega \cdot n}{\tilde{x}\, n_x + \tilde{y}\, n_y + f\, n_z}\ . \end{aligned} \tag{4.6}$$

And we can compute x, y using the equation of perspective deformation:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \frac{f}{|z|} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.7}$$

We use this reverse perspective process to compute 3D coordinates of the 2D folded curve.

4.2.2 Flattening the fold curve

We now have a fold curve represented in the section plane space ((b) in Figure 4.7). The next step of our algorithm is to compute a flattened directrix-independent representation of this curve ((c) in Figure 4.7). Working in this flat space allows easier manipulation of the individual folds, independently of the curve they are wrapped around.

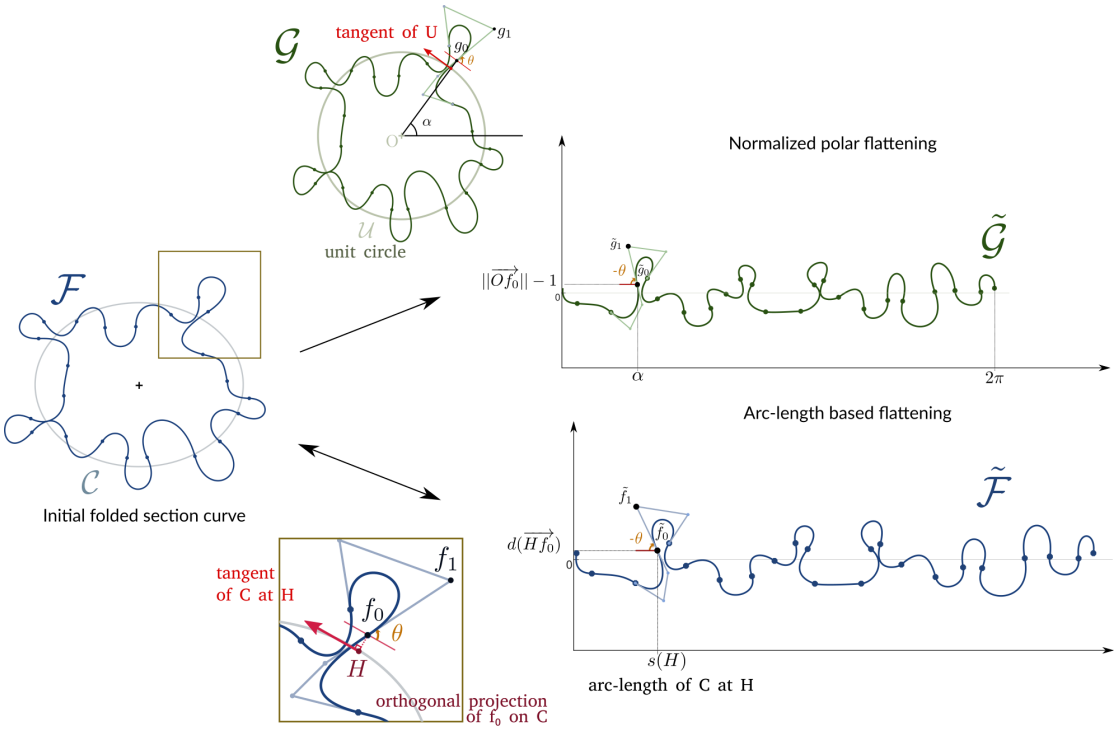


FIGURE 4.8: Overview of the two different flattening algorithms on the same initial fold curve $\mathcal{F}$ and its directrix $\mathcal{C}$. In the normalized polar flattening, $\mathcal{F}$ is first normalized into a fold curve $\mathcal{G}$ so that it is wrapped to a unit circle $\mathcal{U}$, and then mapped by polar angle. In the arc-length dependant version, $\mathcal{F}$ is directly mapped into a flat curve using the arclength of the orthogonal projections of $\mathcal{F}$'s junctions along $\mathcal{C}$.

This directrix-independent representation will be used to complete occluded parts of the hemline (Section 4.2.3), and to perform transfer on different boundary curves (Section 4.3.2).

We present two methods to compute a flattened representation, illustrated in Figure 4.8. The first one can only be applied for elliptic directrix curves, and provides a normalized representation of the folds. The second method can be applied to any smooth directrix curve and is independent the shape of the directrix while accounting for its dimension.

The depth extraction process described previously outputs a 3D planar folded curve wrapped around a directrix curve $\mathcal{C}$. The folded curve is represented as a cubic Bézier spline $\mathcal{F}$. This spline is composed of N cubic Bézier curves $\mathcal{F}_i$ such that:

$$\forall t \in [0,1],\ \mathcal{F}_i = \sum_{j=0}^{3} f_j^i B_j(t)$$

where B_j are the cubic Bernstein polynomials basis and f_j^i are the control points of $\mathcal{F}_i$.

Normalized polar representation In this first method, we assume the directrix is an ellipse. We first express this curve around a unit circle instead of the ellipse by applying the corresponding affine transformation to its control polygon (see Figure 4.8, top row). We denote the resulting curve $\mathcal{G}$ and its control points $\{g_j^i\}$. We detail the process of flattening for a pair of points (g_0^i, g_1^i) denoted (g_0, g_1) for simplification, the remaining control points of each Bézier segment being processed symmetrically.

Denoting $(\tilde{g}_0, \tilde{g}_1)$ the flat representation of (g_0, g_1), we set $\tilde{g}_0 = [\alpha, r-1]$ where (α, r) are the polar coordinates of g_0. Then, we measure θ as the signed angle between the tangent of $\mathcal{U}$ at g_0's projection, and the vector $\overrightarrow{g_0 g_1}$. We compute the flat version $\tilde{g}_1$ of g_1 as

$$\tilde{g}_1 = \tilde{g}_0 + ||\overrightarrow{g_0 g_1}|| \begin{pmatrix} \cos\theta \\ -\sin\theta \end{pmatrix}$$

We obtain $\tilde{\mathcal{G}}$ as a function of a generic angular parameter.

Arc-length dependant representation In this second method, we aim at a representation that preserves the information of the length of the directrix while being independent of its shape. In particular, we want a method able to perform on non-elliptic directrix curves.

As previously, we detail the process of flattening for a pair of points (f_0^i, f_1^i) denoted (f_0, f_1) for simplification, the remaining control points of each Bézier segment being processed symmetrically.

We compute the orthogonal projection H of f_0 on $\mathcal{C}$:

$$f_0 = H + d(\overrightarrow{H f_0})\mathbf{n}(H)$$

where $\mathbf{n}(H)$ is the unit normal vector of $\mathcal{C}$ at H, and $d(\overrightarrow{H f_0}) = \overrightarrow{H f_0} \cdot \mathbf{n}(H)$ the signed distance between H and f_0.
The flat representation of f_0 is then computed as $\tilde{f}_0 = [s(H), d]$ where is $s(H)$ the arclength of H along $\mathcal{C}$.

We compute $\tilde{f}_1$ using a similar method than for the normalized polar flattening, except that θ is defined as the angle between the tangent vector of $\mathcal{C}$ at H and $\overrightarrow{f_0 f_1}$ (see Figure 4.8 2nd row for an example).

In practice, after the reconstruction algorithm, we always obtain an elliptic directrix curve, and we can apply the first method to get the flat representation of folds extracted from a sketch. However, our ultimate goal is to apply this fold distribution to existing smooth surface, for which the boundary may be of different length. Depending on whether we want to apply the folds while preserving the same number of folds or preserving their shape, we may want to keep the information of length of the directrix curve (cf Section 4.3.2). In this case we can use the second method to compute directly

a dimension-dependant flat fold distribution, or use the result of the first method and uniformly scale it by the ratio $\frac{s}{2\pi}$ where s is the total arc-length of the ellipse. These two methods give similar results in most examples of fold reconstruction we worked with.

The second strategy has the advantage of being invertible, meaning that we can compute $\mathcal{F}$ on any, non restricted to elliptical, directrix $\mathcal{C}$ using only the flat representation $\tilde{\mathcal{F}}$. We use this reverse operation for fold application on existing surface sections (see Section 4.3.1).

4.2.3 Completing the occluded parts of the hemline

So far, we described an algorithm allowing to build a directrix-indepedent fold curve from a sketched hemline. However, the sketch is only a partial representation of the hemline. As stated previously, tubular folds often causes self occlusions in the garment that may not be represented in the input sketch. Additionnally, a whole side of the garment is hidden by the viewpoint, and we need to complete the fold curve to have a full representation of the boundary.

Self-occlusions of the hemline In some cases, the depth of the folds creates self-occlusion of the cloth, and the hemline is not fully visible, hence drawn as discontinuous, as in the examples of Figure 4.9. The goal of this section is to obtain a continuous fold distribution curve from such a discontinuous projected hemline.

A first method to overcome this issue would be to complete the discontinuous parts of the hemline itself. Occlusions occur either at T-junctions with inner silhouettes, or at silhouette points, where the surface smoothly turns away from the view direction. At T-junctions, we can expect the hemline to continue in the same direction. Such assumption is commonly used while trying to recover occluded lines in line drawings [CSPN11]. On inner silhouettes, the tangent of the hemline turns away from our view direction. While the human eye could in general easily infer the direction of the hidden part here (see Figure 4.9, green), automatize such completion is not trivial.

A second method would be to launch the reconstruction process on the discontinuous hemline, obtain a discontinuous flat fold curve $\mathcal{D}$, and try to complete the discontinuities directly on this curve. In flat fold space, we can build a first continuous curve with C^1 continuous completion (see examples in Figure 4.9,blue). The inferred occluded parts may be inconsistent with the rest of the fold curve. We propose an algorithm to compute a continuous fold curve which is consistent in visible and occluded parts.

We model S as a spline composed of N cubic Bézier curves S_i, with

$$S_i(t) = \sum_{j=0}^{3} b_j^i \, B_j(t) \ , \tag{4.8}$$

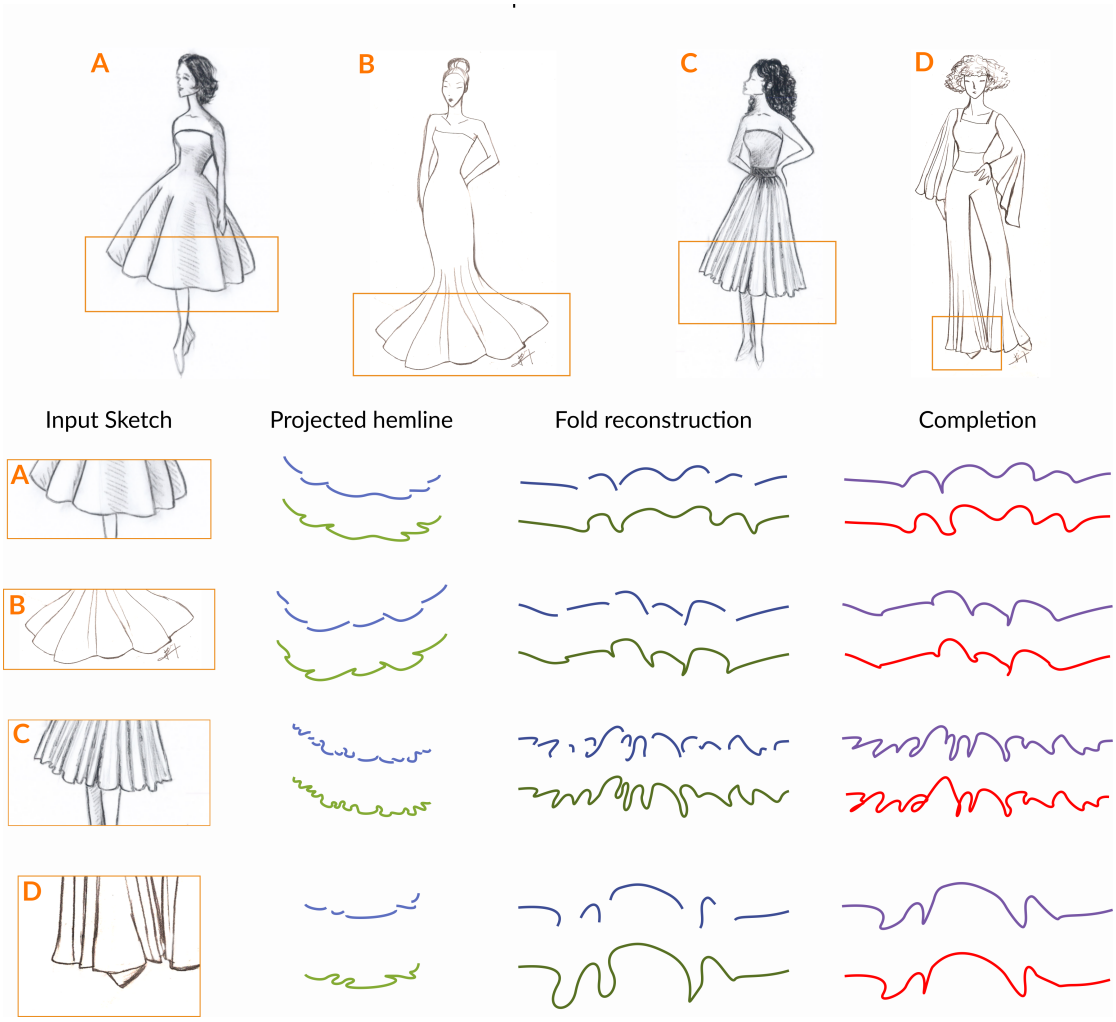


FIGURE 4.9: Examples of occluded projected hemlines. In all these sketches, the visible part of the hemline is a discontinuous curve (2nd col, blue), for which we can compute a normalized fold curve (3rd col,blue). We can apply on this discontinuous fold curve a simple C^1 completion (4th col,blue), and improve it using a consistency-preserving algorithm (4th col,red). We compare this result with a fold curve reconstructed from a continuous version of the hemline (2nd and 3rd col, green).

where B_j are the cubic Bernstein polynomials basis and b_j^i are the control points of S_i.

The general idea is to consider that non-visible curve portions should be similar to visible ones. We thus force S to approximate $\mathcal{D}$ in the visible region, while inferring non-visible part using similarity as well as continuity criteria. Note that S only approximates $\mathcal{D}$ in the visible part in order to remain robust to possibly imprecisely sketched folds depicted by the user, especially at grazing angle. Our method works as follows.

We start by identifying the parts of the spline that are represented in the discontinuous reconstruction $\mathcal{D}$.

We first generate an initial guess Bézier spline $\hat{S}$ interpolating $\mathcal{D}$ in the visible region, and completing the missing part using C^1 continuity (see Fig. 4.10a). To ease further

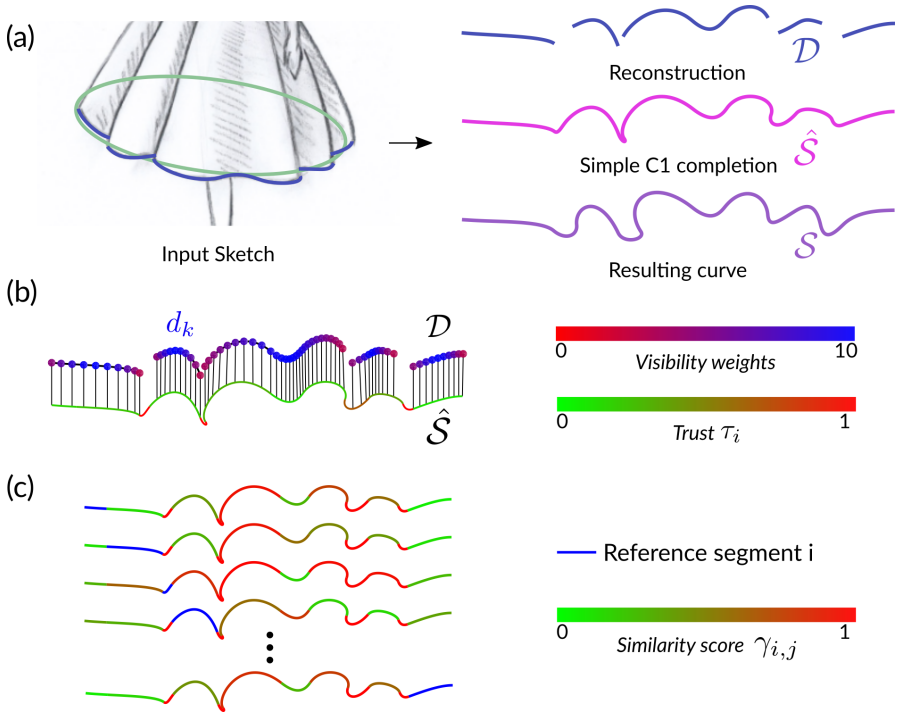


FIGURE 4.10: Illustration of the consistency-preserving completion algorithm. The discontinuous fold curve reconstructed is first completed using a simple C^1 completion process (pink curve in top row). Our algorithm matches at best the discontinuous fold curve in trustable areas of the continuous fold curve, while mimicking similar existing folds in occluded areas.

analysis, $\hat{S}$ is resampled such that individual junctions between *polynomial segments* occur at inflexion points of the spline (an example of such decomposition is illustrated in Fig. 4.10c) We also consider a set of uniformly distributed samples $\{d_k\}$, $k = 1, \ldots, K$, of $\mathcal{D}$ and associate to each sample its orthogonal projection $\hat{S}_i(t_k)$ belonging to the polynomial segment $\hat{S}_i$ at some parameter value t_k.

Secondly, we associate a *trust score* $\tau_i \in [0, 1] = t_k^{\max} - t_k^{\min}$ to each segment of S given by the size of the range of parameters $[t_k^{\min}, t_k^{\max}]$ on which samples $\{d_k\}$ are projected to $\hat{S}_i$. A fully visible segment will therefore be associated to the trust score 1, while a fully occluded one will be associated to 0 (see Fig. 4.10b).

Third, we define a *shape similarity score* $\gamma_{i,j}$ between two polynomial segments (i, j), as illustrated in Fig. 4.10c, in computing the maximal angle between corresponding tangent vectors over a discrete uniform sampling of the curve $\hat{S}$ at parameters $t_r = r/(R-1)$, with $R = 10$.

$$\gamma_{i,j} := \max_{t_r} \operatorname{acos}\left(\frac{\hat{S}_i'(t_r)}{\|\hat{S}_i'(t_r)\|} \cdot \frac{\hat{S}_j'(t_r)}{\|\hat{S}_j'(t_r)\|}\right) . \tag{4.9}$$

The map $Sim : i \to j$ associating a segment i to its most similar segment j is precomputed by taking into account the *shape similarity score* weighted with respect to the

trust score to favor the most visible segments

$$Sim(i) = \underset{\substack{j\in\{0,\dots,N-1\}\\ j\neq i}}{\arg\min}\ 1 - e^{-\gamma\,(i,j)^2/\tau_j^2}. \tag{4.10}$$

Note that we only consider a segment to be similar to another one if its associated value $1-e^{-\gamma\,(i,j)^2/\tau_j^2} > 0.5$, otherwise we set $Sim(i) = -1$ allowing to prefer completion based on other criteria such as smoothness in the next optimization.

The final curve S is globally computed by minimizing the following quadratic energy E_S, expressed with respect to the control points of the spline:

$$E_S = E_{visibility} + E_{similarity} + E_{distance} + E_{G^1}\ . \tag{4.11}$$

- $E_{visibility}$ enforces the curve to match the discontinuous projection

$$E_{visibility} = \sum_{k=0}^{K} \omega^k_{visibility}\, \|S_{i_k}(t_k) - d_k\|^2,$$

  where the visibility weights $\omega^k_{visibility}$ are defined using a Gaussian function centered on the middle index of each continuous part of $\mathcal{D}$, such that $\omega^k_{proj} = 1$ at discontinuities and $\omega^k_{proj} = 10$ at the center of continuous parts, see an example in Figure 4.10b. This energy component allows to locally limits clearly visible part of the curve to be deformed, while discontinuous part, often seen at grazing angles, may be more easily modified.

- $E_{similarity}$ encourages segments that are not clearly visible to locally reproduce the shape of their associated similar segment,

$$E_{similarity} = \sum_{i=0}^{N-1} \omega^i_{similarity} \sum_{r=0}^{R-1} \|S'_i(t_r) - S'_{Sim(i)}(t_r)\|^2,$$

  where $\omega^i_{similarity} = (1-\tau_i)$ if $\tau_j < \frac{1}{2}$ and $Sim(i) \geq 0$, and 0 otherwise.

- E_{G^1} enforces the tangent-continuity at segment junctions

$$E_{G^1} = \omega_{G^1} \sum_{i=0}^{N-2} \|(b^i_2 - b^i_3) - \alpha_i(b^{i+1}_0 - b^{i+1}_1)\|^2,$$

  where $\omega_{G^1} = 1$, $\alpha_i = \|\hat{b}^i_2 - \hat{b}^i_3\| / \|\hat{b}^{i+1}_0 - \hat{b}^{i+1}_1\|$ with $\hat{b}_i$ being the control points of the initial guess curve $\hat{S}$.

- $E_{distance} = \omega_{dist} \sum_{i=0}^{N-1} ||b^i_0 - \hat{b}^i_0||^2$ prevents the points from varying too much from their initial position. This criteria is only useful to initialize the general placement of the curve and we set a low weight $\omega_{dist} = 0.01$

We end up with a continuous curve representing a full version of the visible side of the garment's hemline.

We discuss in Section 4.4.2 the effects of this completion algorithm after applying folds on a surface (see Figure 4.16).

The Hidden side of the hemline The fold distribution reconstructed from the sketch only represents a partial representation of the folds in the garment. We need to extend it to represent the hidden side of the garment, not drawn in the sketch.

We first resample our fold distribution spline so that junction points are located at inflexion points of the curve, which was already the case if we went through the completion algorithm previously described. The curve is now a continuous set of alternatively concave and convex individual folds. We extend the curve by duplicating those individual folds, until the curve hits a target abscisse length L, while ending with a fold of opposite curvature sign than the first one. Then, we smooth out the curve so that it is G^1 continuous, accounting for the continuity between the point of abscisse L and the first point. In the case of polar fold distribution (see Section 4.2.2), the target abscisse length would be $L = 2\pi$, whereas in the case of arc-length based distribution, it is the total arc-length of the ellipse.

This method allows the extension with folds that are similar to the reconstructed ones. Another way of extending the fold distribution would be to use statistics-based methods, such as [HOCS02] or [LA15] to generate new folds following the same statistical distribution.

4.3 Transferring folds to existing geometries

The reconstruction algorithm described in previous section builds a flat fold distribution representing the visible parts of tubular folds in a sketch.

In this section, we describe how to use a flat fold curve to fold a 3D surface, using the section curve space as intermediate space ((c) → (b) → (d) in Figure 4.7). First, we explain how to do this in the case of reconstruction, meaning when the fold curve is already adapted to the surface we want to fold. Then, we study the case of transfer, in which we want to fold a surface of different shape.

4.3.1 Applying a fold curve to a smooth surface

We assume our garment patch contains two boundary curves between which the folds propagate. One of this boundary curve is the hemline, and we first present how to apply

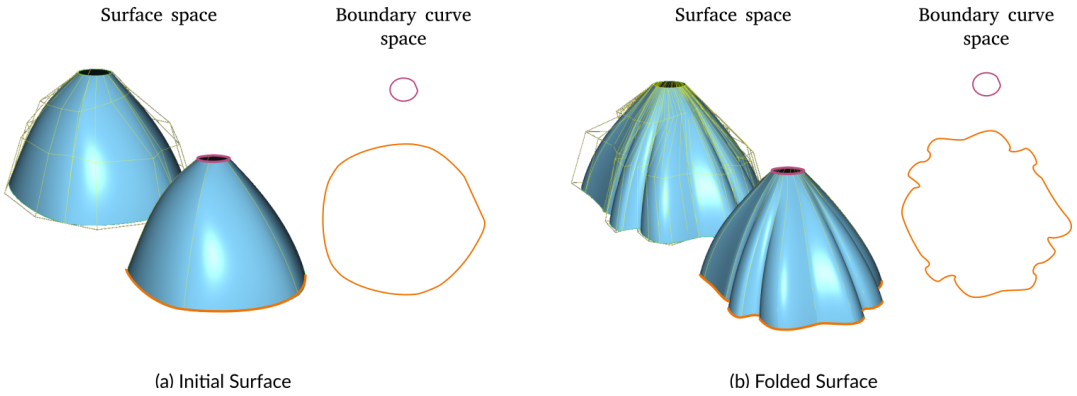


FIGURE 4.11: Example of folding a surface represented with tensor Bézier spline (a). We apply fold distribution on one of the section curve boundary (orange) which is a spline curve of the control polygon of the surface (yellow). The resulting surface (b) propagates smoothly the fold perturbation from one boundary curve to the other.

the fold distribution on it. Then we discuss on how to propagate this fold perturbation on the rest of the surface.

Folding a boundary curve As before, we assume that the boundary curve of the garment is planar, so we will compute its folded representation in 2D local coordinates. The idea is to apply the inverse of the arc-length based flattening method presented in Section 4.2.2.

Using the notations of that section (see also Figure 4.8), for each control point $\tilde{f}_0 = [s, \rho]$, we compute f_0 as

$$f_0 = H + \rho \mathbf{n}_{\mathcal{C}}(H)$$

where H is the point on $\mathcal{C}$ of arclength s and $\mathbf{n}_H$ is the unit normal vector of $\mathcal{C}$ at H.

We compute f_1 using the angle $\theta = -\arg(\overrightarrow{\tilde{f}_0 \tilde{f}_1})$:

$$f_1 = f_0 + ||\overrightarrow{\tilde{f}_0^i \tilde{f}_1^i}|| R_\theta \mathbf{t}_{\mathcal{C}}(H)$$

where R_θ is the 2D rotation matrix of angle θ, and $\mathbf{t}_{\mathcal{C}}(H)$ is the unit tangent vector of $\mathcal{C}$ at H.

Fold propagation We now have a planar folded curve representing the hemline of the folded surface. In this work, we focus on application on tensor Bézier spline surfaces for which the control polygon is aligned with the fold direction, meaning that the boundary curves of the patch are either section curves to be folded, or silhouette curves of the surface (see one example in Figure 4.11). We apply the fold distribution using the wrapping algorithm on one of the boundary curves, and redistribute the junction points of the other to obtain a smooth propagation of the folds.

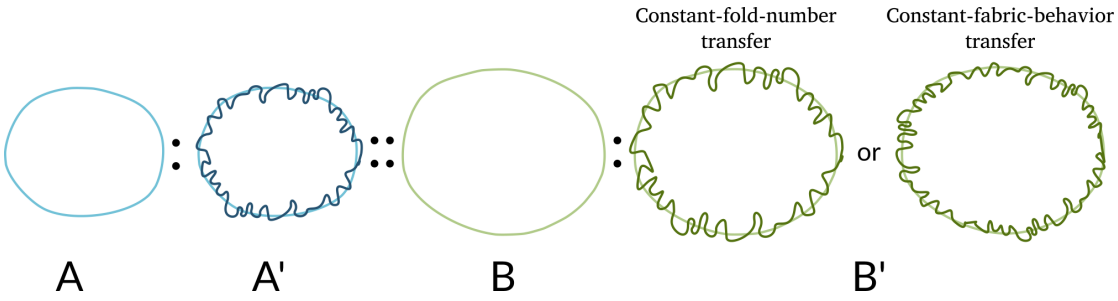


FIGURE 4.12: An example of fold analogy paradigm: we want to apply folds on a directrix B using the fold distribution described by its shape A' wrapped around a directrix A. Two solutions can be accepted, either if we want to preserve the number of folds or their shape from A'.

The choice of using parametric surface may look restrictive, especially for garment modeling which is more commonly done using meshes. However, we believe this fold representation could be used for other types of representations, in particular in meshes, using a system similar to FoldSketch [LSGV18] so that it can produce a wider diversity of folds. In Chapter 5, we present a sketch-based garment modeling system using the tensor Bézier spline surface for which this fold transfer algorithm is well suited.

4.3.2 Fold analogies

We now assume that we want to apply a fold curve issued from the reconstruction algorithm to a section curve with different shape. The process described in previous section will not work in most cases, in particular because the target section curve will not have the same arclength. This problem can be formulated similarly to the image analogy paradigm [HJO+01], by denoting A, A' as the source directrix and fold curves, and B, B' as the target directrix and output fold curve (see Figure 4.12).

There are different ways of computing B' depending on what aspects of the folds we want to preserve. We may want to preserve the number of folds, at the risk of changing their dimension. On the other hand, we may want to preserve the fold's scale, and adapt the number of folds, which is what would happen if B was made of the same fabric than A. We present here how to perform this transfer in both cases, the output of both methods is a flat fold distribution curve ((c) space in Figure 4.7) which then can be mapped to the curve section domain (cf Section 4.3.1).

Transfer strategies For the first method, called *constant-fold-number* strategy, we map A' to a flat representation using the polar flattening algorithm (see Section 4.2.2). We then have a normalized version of the folds, as if wrapped around a unit circle. Then, we can compute a folded section B' curve using a similar method than presented in the previous section but using the angular parameter instead of the arclength. As

a result, the number of folds remains constant, while their sizes scale with the curve length. This approach allows to visually give the appearance of a similar garment, as was done in [BSBC12].

Note that this method only works if B is a convex closed curve with no self-intersection. An alternative version would be to scale the normalized flat fold curve by $\frac{s}{2\pi}$ where s is the arclength of B, and use the inverse flattening algorithm presented in previous section to compute B'.

The second strategy, called *constant-fabric-behavior transfer*, seeks to adapt the number of folds with respect to the arc-length of the target section curve, while maintaining a constant size of folds. We use the arc-length dependant method (see Section 4.2.2) to get a flat fold distribution mapped with the arc-length of A. Then, the fold distribution can be extended or cut out in order to be sized with the length of B with a method similar to the completion of the hemline in Sec. 4.2.3. Then, B' is computed using the inverse flattening algorithm presented in previous section.

Comparison While those strategies can lead to very different results, we may wonder when to apply one or the other.

The second strategy is well-suited to transfer folds between surfaces that have comparable dimensions. For example, say we want to transfer folds from a sleeve to a dress. Then, we will probably expect the folds to keep the same size, but adapt the number of folds to the length of the dress' boundary curve.

Say now we want to transfer folds taken from a human sized garment model to a tiny character, say a smurf for example. Then, the constant fabric behavior transfer would probably lead to a garment with few folds, and a very different aspect than what it had on the human. The constant-fold-number transfer on the other hand, would give a visually similar appearance to the garment.

An third adaptative version would be to add an intermediate step in the constant-fabric-behavior transfer to uniformly scale the flat fold curve before cutting/extending it. The scaling factor is there computed as the ratio between the dimensions of the transfer spaces (for example ratio between the height of the characters wearing the garments). This third method would allow to properly transfer folds from a sleeve of a garment wore by a human to a dress wore by a smurf.

Some results are displayed in the Figures 4.17 and 4.18 and discussed in Section 4.4.2.

4.4 Results and Evaluation

In this section, we present and discuss some results we obtain using our algorithm. First, we propose some quantitative and qualitative evaluation of the reconstruction algorithm. Then, we show fold reconstruction and transfer on a few applicative examples.

4.4.1 Evaluation

We evaluate our reverse-perspective based reconstruction algorithm on synthetic examples. The 3D scenes were modeled and rendered on Blender.

Plane estimation We first study the algorithm that estimates a plane from the 2D sketch and hypothesis on the perspective model (cf Sec 4.2.1). We use a 3D model of an ellipse that is rendered with the camera set up described in Figure 4.6, for different values of a, b (semi-length axis of the ellipse), (x_0, y_0) and φ where $n = [\sin(\varphi), \cos(\varphi), 0]^T$.

The results are evaluated with the following measures:

$$E_\Omega = \frac{||\Omega_{GT} - \Omega||}{||\Omega_{GT}||} \qquad\qquad E_n = |\arccos(n_{GT} \cdot n)|$$

Results are displayed in the Figure 4.13. The first column shows how the hypothesis $d = 250$cm influences the plane estimation. We launched the plane estimation algorithm, described in Section 4.2.1, on renderings of the scene while translating the objects along the $z-$axis, changing the distance camera/model d. We can see that even for scenes where d is far from 250, the normal vector of the plane is still very close to the ground truth. The estimated plane origin goes away from the ground truth as d differs from the hypothesis. Most of this difference is explained by the inherent error on the z-component (displayed in discontinuous black line in Figure 4.13).

The second column shows the evolution of the error measures while adding a rotation to the ellipse. We can see that the error on the estimated plane origin remains very low, and the error on the estimated plane normal is under 20° for all study cases.

The third columns shows the influence of adding a non-zero $z-$component on the normal vector of the plane. We rotate this plane by an angle $\theta \in [0, \frac{\pi}{2}]$, and measure the error in the plane estimation. As previously, the error remains low for the location of the plane origin (under 10% in all cases). As for the estimated normal, the angle between ground truth and estimation is very close to θ in all cases, which is explained by the hypothesis taken as $\theta = 0$.

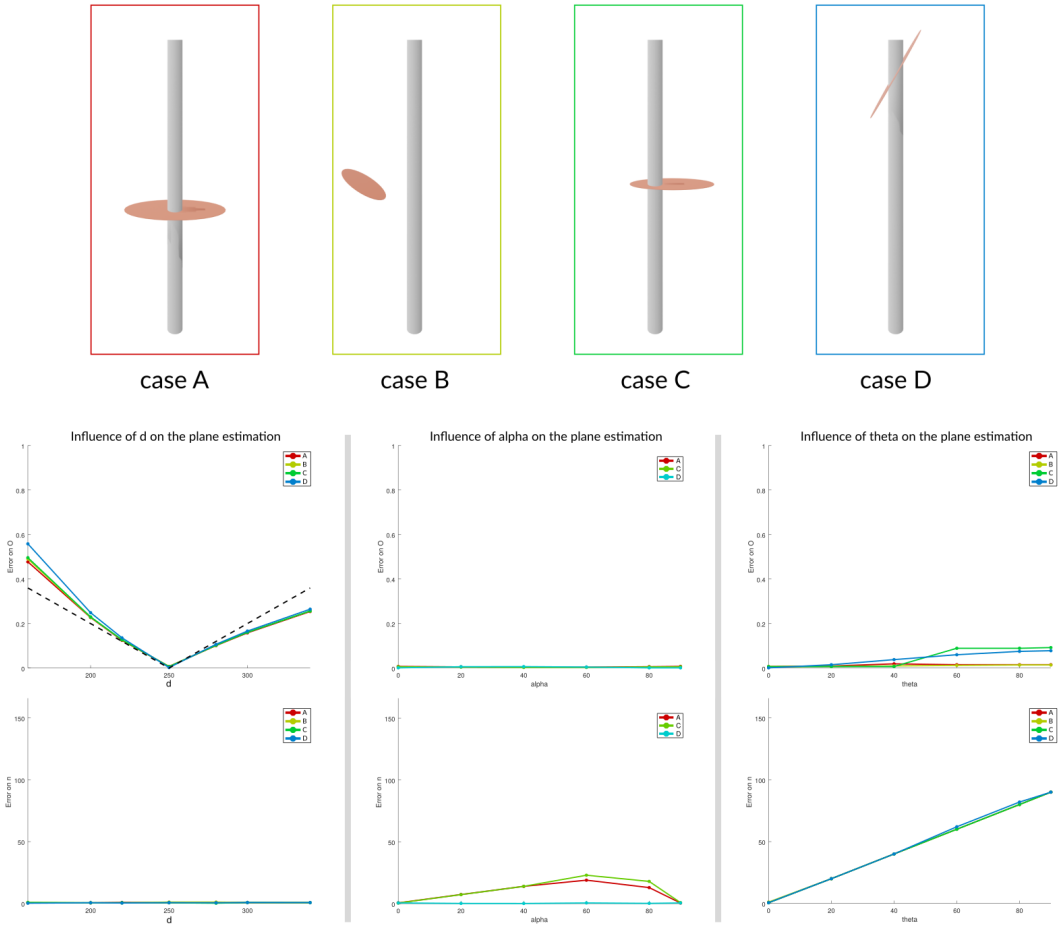


FIGURE 4.13: Error measure on plane estimation. First row shows the configuration of the 3D scenes for each study case. Each scene contains a straight cylinder, which represents the character, and a planar ellipse. Based on a renderer image of these scene, we estimate the plane containing the ellipse, and measure the error from the ground truth. Second row shows evolution of error measures on the normal and origin of the plane while deviating from the hypothesis. d represents the distance camera/model, α the rotation angle of the ellipse in its local representation, and *theta* the rotation angle of the normal plane around the y-axis.

Fold reconstruction In this second experiment, we use three different materials pre-set from the Blender software to compute different variations of folded-shapes. Each simulated mesh is rendered as an image, and the user draws on top of the resulting image the folded border (occluded border is also drawn) and the approximated ellipse.

Our algorithm generates a 2D folded section curve that we compare to the true trace of the 3D border curve projected orthogonally onto the horizontal plane. The results displayed in Fig. 4.14 show that the reconstructed hemlines are in general close to the ground truth, even with irregularly shaped folds, such as the silk model. One can note that the least accurate results are related to the case with few large folds. Indeed, in the latter case, the true 3D folded border curve deviates more from the planar hypothesis assumed during the reconstruction process.

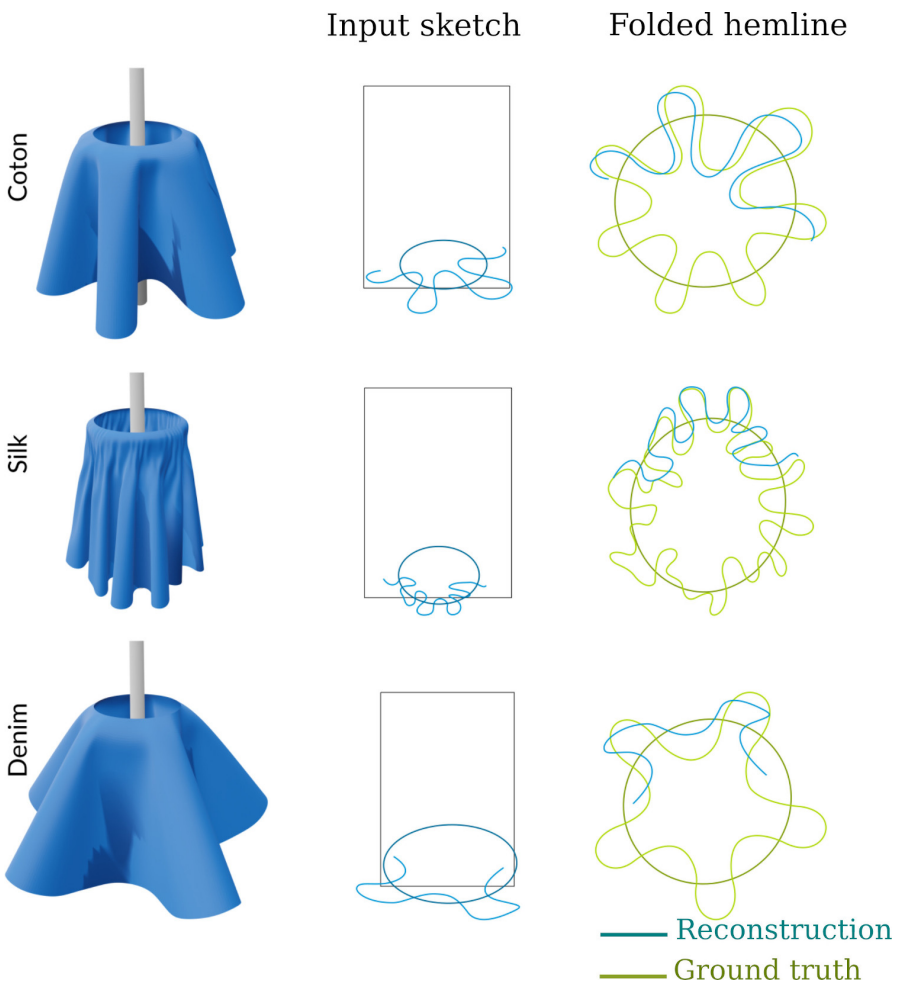


FIGURE 4.14: Reconstruction of fold curves generated from physics-based simulation. The result of the simulation is rendered on an image (1st col), ontop of which we drawn the projection of the folded hemline (2nd col). Our reconstruction algorithm provides an estimated folded hemline (4th col, blue), which is compared to the ground truth (3rd and 4th col, green).

4.4.2 Results

We now present some applicative results of our algorithm. We first focus on the completion algorithm presented in Section 4.2.3 on some examples of sketches. We then show results of reconstruction and transfer on some primitive surfaces.

Note that in some of the examples, we anticipate the following of this thesis and apply folds on a garment modeled with the system presented in Chapter 5.

In the Figure 4.15, we first present some examples of folds applied on surfaces ressembling the ones in the input sketch. We see that our method provides visually good results for folds with different shapes.

FIGURE 4.15: Some examples of fold reconstruction and application on surfaces. Initial surfaces were obtained using the method for garment modeling presented in Chapter 5.

Completion of discontinuous hemlines We then study the results of our completion algorithm presented in Section 4.2.3. In Figure 4.16, we present for each input sketch two surfaces corresponding to two different fold distribution curves: one is obtained by completing manually the occluded parts of the hemline on the input sketch (green curve), and the other by inputting a discontinuous hemline and using our automatic completion algorithm (red curves).

Our algorithm manages to add curvature in some hidden concave parts, such as in the sketches A and B of Figure 4.9. In those cases, the discontinuous hemline provides at least an example of trustable concave fold which is matched to the other with the nearest neighbor computation, see Figure 4.10 for an example.

If the discontinuous projected hemline does not show any concave part, and therefore no depth, as for the sketch D in Figure 4.9, then the completion is not better than the simple C^1 completion. Improvement of this algorithm could be done by using the unsigned curvature of the folds instead of their tangent direction so that concave parts can be matched to convex part with similar profiles, which are usually more trustable.

Note that our algorithm does not account for self-intersections of the fold curve that may appear (for example in sketch C), which could be solved in a post-process step.

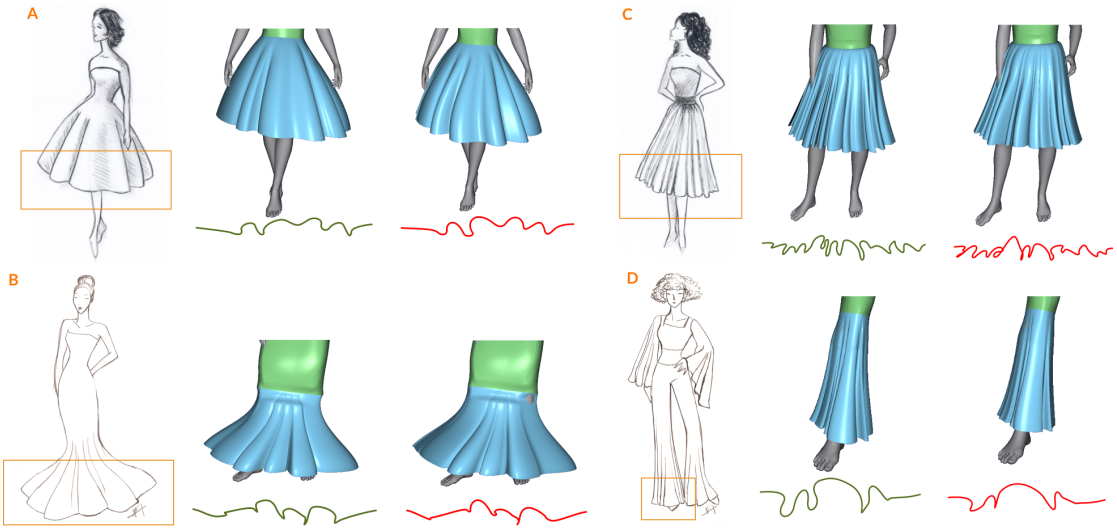


FIGURE 4.16: Comparison between folds reconstructed using our completion algorithm (red curve), and with human annotated completion (green curve) on several sketches.

In Figure 4.16, we apply the fold distributions obtained with our completion algorithm to a 3D surface, and compare it to the fold distribution obtained from a continuous hemline representation. We see that the differences in occluded parts of the fold distribution are not so visible in the 3D reconstruction, and the results obtained with our completion algorithm are convincing.

Transferring folds distributions Finally, we present the results of fold transfer on some examples.

In Figure 4.17, we see how the fold distribution adapts to surfaces having the same shape but different dimensions. If the dimension are comparable, like the woman and child, then we observe that the constant fabric distribution is a good solution to preserve the fold's shape. But on a really smaller character, like the starfish, the result seems to have very few folds and differs significantly from the intput sketch's visual aspect. One solution is the adaptative strategy, which provides good result in this case (see Section 4.3.2 for details).

The second experiments applies on the same initial surfaces a fold distribution extracted from a smaller curve, see Figure 4.18. We see here that we obtain very different results whether we take a constant-fold-number or a constant-fabric-behavior strategy. In the case of the woman and child, both seem acceptable, depending on the expected result. In the case of the starfish, we see that we need the adaptative strategy to get a visual aspect similar to the constant-fabric-behavior transfer. The character being significantly smaller than the standard size, the classic constant-fabric-behavior strategy gives fewer folds than we can expect.

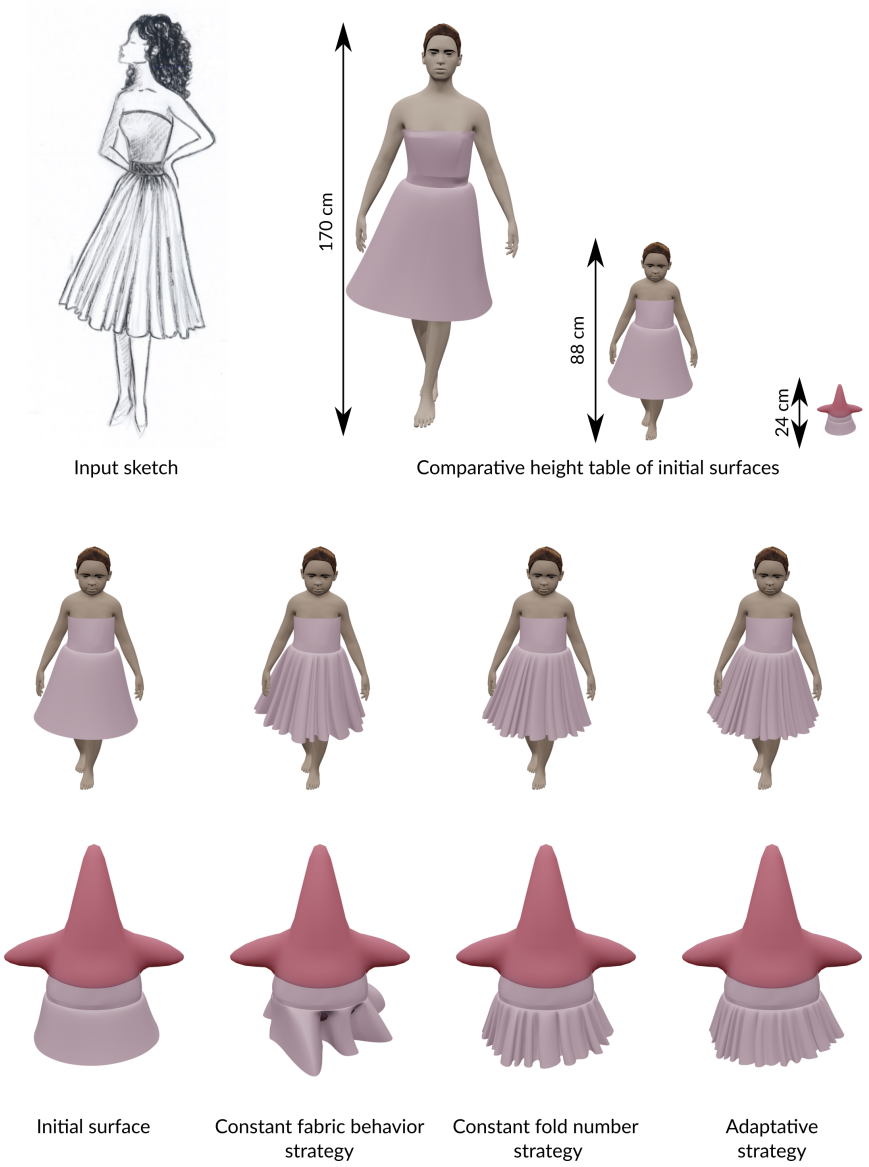


FIGURE 4.17: Some result of folding garment surfaces with different sizes: one is close to reconstruction (target character with the same size than the input sketch's model), the other one is adapted to a smaller target character, and the last one is adapted to a tiny character. Different folding strategies are performed on these surfaces. See Section 4.3.2 for details.

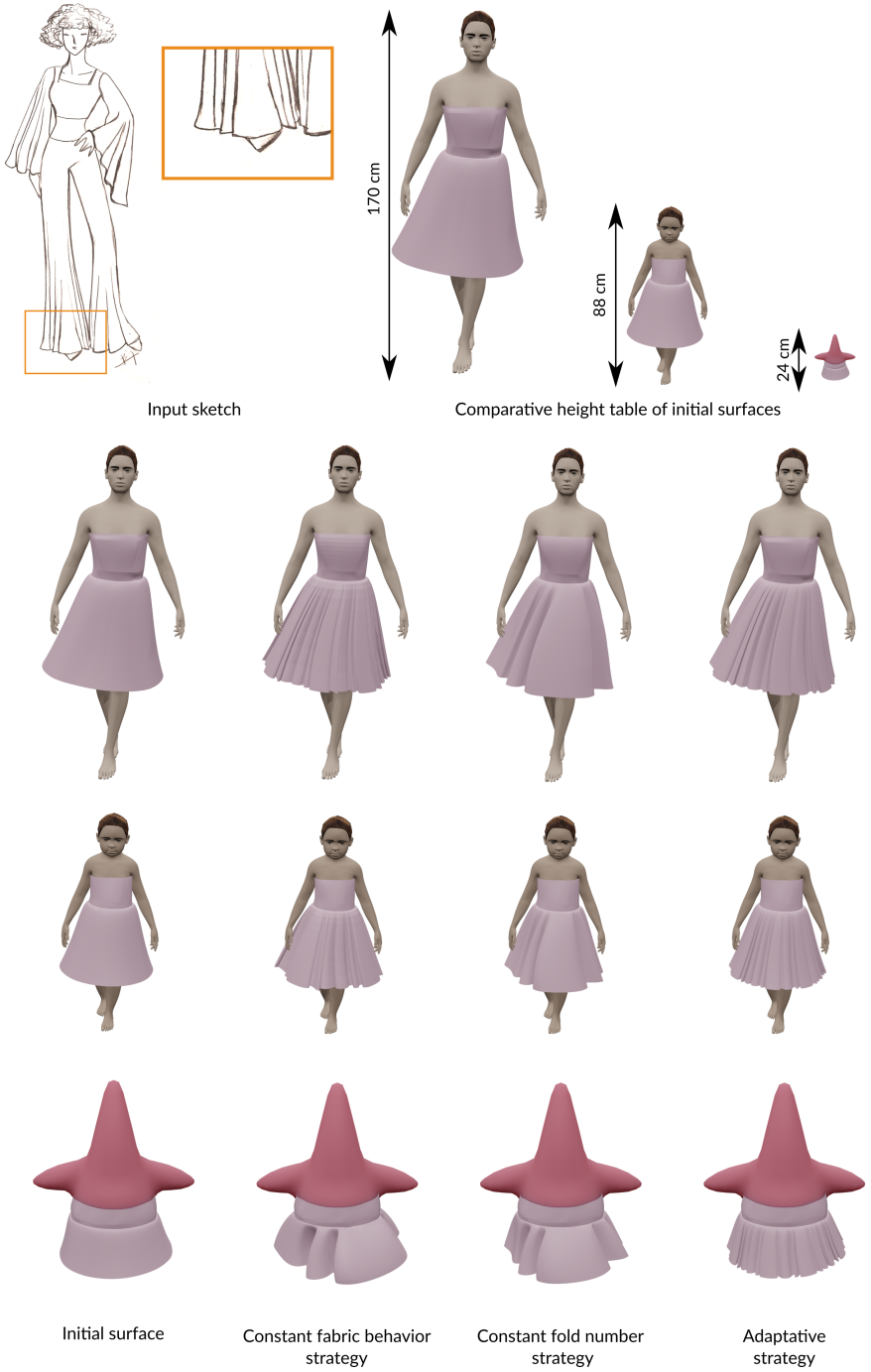


FIGURE 4.18: Some results of folding garment surfaces differing from the input sketch. Different folding strategy are applied on each of the initial surfaces.See Section 4.3.2 for details

4.5 Conclusion

We presented an approach to model and transfer axis-aligned tubular folds using one sketched folded boundary. Our approach exploits the perspective inherent to fashion sketches. We proposed to transfer folds to different boundary curves with two different strategies: one preserving the number of folds from the sketch, and another preserving their shape.

While we performed a smooth linear propagation of those folds from a free boundary curve to another on a surface, other methods could be investigated. For example, we could mimick seamed folds by squashing the fold curve in the tight boundary curve. Another idea would be to use inner silhouettes, as discussed in Section 4.1.1 to guide the direction of propagation, and model twisted folds.

In the next section, we exploit the rest of the 2D curves contained in a fashion sketch. We present an algorithm to synthesize a virtual garment on an arbitrary character using a single sketch.

Chapter 5

Dressing arbitrary characters using a single sketch

FIGURE 5.1: Our approach takes as input a single, annotated fashion drawing and synthesizes a 3D garment of similar style over any existing 3D character. The method handles deep folds and automatically adapts the garment to the morphology and pose of the target, allowing to easily dress human-looking to cartoon-style models.

The fabric's material, the shape of the cloth patterns, the dressed character's morphology, pose and motion: a multitude of parameters are conditionning the shape of a 3D garment. Designing an intuitive tool to create and model virtual garments has been an important challenge in the field of 3D graphics.

While traditional physics-based approaches [VCMT05] are providing realistic results, they rely on the design of 2D patterns, making it difficult for an artist to model a garment on an arbitrary character using a final visual aspect as target. On the other hand, example-based and data-driven approaches provide reconstruction systems using no parameters. They rely on important databases of garments [WCPM18] and manage to model garments from a picture at the cost of computing efficiency [YAP+16]. Alternative approaches try to use the expressiveness of sketches to help artists to model garments. They usually propose an interface on which the user draws silhouette, seams, folds and border lines ontop of a view of the character to be dressed [DJW+06, TWB+07,

FIGURE 5.2: Examples of fashion sketches from specialized litterature [NG09], with diverse levels of abstraction.

RMSC11]. Those methods may constrain the expressiveness of the artist, the input sketches being in practice, pretty far from the artistic variety of fashion illustration.

In this work, we address the issue of interpreting a fashion sketch to model a garment fitting an arbitrary target character. Many challenges arise from this topic. First, fashion sketches may be more or less expressive or precise, and we need to decide what features we want to exploit and how. Secondly, transferring the garment to an arbitrary character implies to account for different morphologies, geometries and different poses, which is challenging because all of this parameters are influencing the garment's shape. What makes two garments similar ? What are the features ones wants to extract from the sketch, and what is determined by the shape of the target character ? Brouet et al. [BSBC12] tackled the issue of defining style of a garment in the context of 3D-to-3D transfer. The similarity criteria they established serve us as basis for our 2D-to-3D transfer.

Our method takes as input a fashion sketch with a few user annotations, described in Section 5.1.2. In particular, we ask users to over-trace the free borders of loose parts, which convey folds; the silhouette of the garment, which conveys surface normals; and skeletal bones of the character, which convey the relative location of the garment with respect to the limbs and body of the character. It then works in four steps, illustrated in Figure 5.3. We first extract geometric information from the input annotations, and create an initial shape of the virtual garment over a target rigged character. This first surface represents a garment that matches the proportions, position and orientation of the garment depicted in the sketch (Section 5.2). We then deform this initial surface, so that its shape matches the shape in the drawing (Section 5.3). Garment patches are represented as parametric cubic Bezier patches, which brings a compact representation, parameterized along limbs direction.

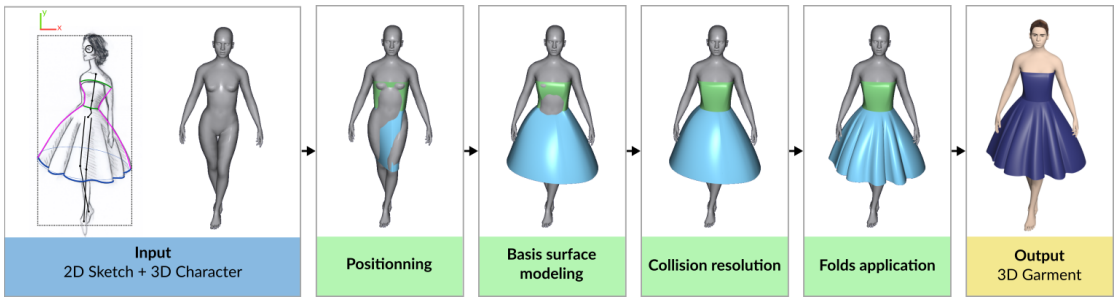


FIGURE 5.3: The four steps of our garment synthesis pipeline satisfying the style-transfer criteria : scale, fit, shape and folds.

Finally, the last step of our method consists in synthesizing folds over the garment. We achieve this goal by transferring the fold patterns extracted from the sketch using the algorithm presented in Chapter 4, anf by scaling them to fit the character size. Thanks to the parameterization of the patches, garment exhibiting deep folds along limbs direction, typically encountered in loose garments falling under gravity, can be easily represented.

We illustrate the versatility of our method by creating a variety of garments, including dresses, shirts and pants. More importantly, our results were obtained from sketches of various styles, from fashion sketches to cartoon sketches with extreme proportions. Similarly, we transferred the extracted garments to realistic as well as to stylized 3D characters.

The works presented in Chapter 4 and Chapter 5 has been submitted for publication in October 2019.

5.1 Garment representations

The goal of this first section is to propose a 2D and a 3D representation for garment modeling and transfer.

We need to understand what is represented in line drawing fashion sketches, and how it relates with the geometric cues defining the style of a garment (Section 5.1.1). We use this study to define the inputs of our modeling method and the hypothesis necessary to their interpretation (Section 5.1.2). We then relate this to a surface representation able to express and enforce the style criteria in the target character's space (Section 5.1.3).

5.1.1 Style in garment sketches

A seen in the previous chapter, there are many different types of garment sketches. Some are very realistic and explicit, and others can show more distortions, may be very expressive, or even abstract, like on the example of Figure 5.2(D). However, in most

cases, a sketch is sufficient to provide a good idea of the 3D shape of the garment it is representing.

Garment drawings A stylized sketch-based representation of a garment differs from the technical representation of sewed objects as presented in Chapter 3 in many ways. Let us highlight some of them.

Similarly to sewed objects, silhouette and border curves are usually well depicted in garment sketches: they represent in particular the boundary between the cloth and the rest of the drawing. However, the seams between patches, the dart lines and other fabrication indications may not be represented, at the profit on more aesthetic lines, such as fold's silhouettes and shadows. Moreover, many garments are not perfectly developable when fitting the body in doubly-curved surface areas, such as the shoulders. It seems rather unsuitable to search for a ruled surface representation in this case.

The body of the dressed character is another important feature of the garment sketch. Not only does it provide information on the positionning and orientation of the garment worn by the character, but it also affects the shape of the garment itself, in particular in tight areas of the garment, where the silhouette continuously follows the body's outline (for example the top parts of Figure 5.2(A,D)).
It is not common in fashion illustration to draw realistic human body proportions. Exagerration of proportion and expressiveness of the outline are usually left to the choice of the artist: the outline may not fully show the outline of the body, as in Figure 5.2(B), the proportions of the character may be exagerrated, as in Figure 5.2(A), or the pose depicted may be unrealistic, as in Figure5.2(C).

In most cases, even if the drawing is not completely explicit, one can easily infer the shape of the garment that is represented, and imagine how it would look on a given character. However, in some cases, it is very unlikely that the outline of the resulting garment would exactly fit the lines drawn in the sketch.
Thus, we may wonder what are the features which make us believe that the garment we imagine corresponds to the one depicted on the sketch. Or, formulated differently, what makes two garment looking similar in style while being different in geometry ?

The following paragraph presents the style criteria which turn out to be pertinent for our goal of style preserving garment transfer.

Style criteria The problem of defining style of garment was adressed by different authors, mostly in the context of 3D-to-3D garment transfer to different human morphologies. Brouet et al. [BSBC12] came up with a list of geometric criteria defining *garment style*. Among them are:

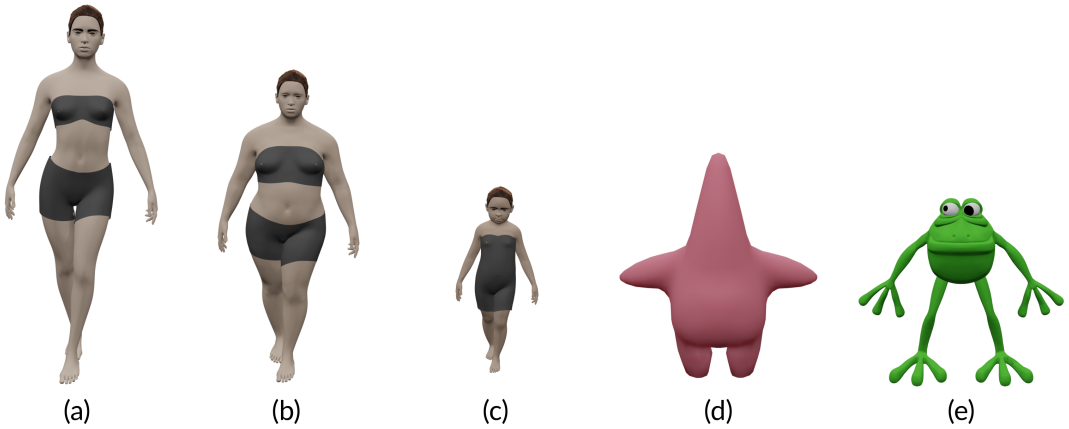


FIGURE 5.4: Examples of potential target characters having different geometries while sharing comparable topologies

(I) **Scale** (or proportionality), expressed through the *relative location* of the garment with respecto to the character's body and limbs.

(II) **Fit**, expressed through the preservation of the *tight regions* where the garment should fit the body.

(III) **Shape**, expressed using normals in loose areas.

We use these criteria to guide our style preserving garment transfer. Moreover, since we want to allow style transfer between radically different characters, we choose to consider the shape criteria at the scale of the *overall shape*, and add the following fourth's similarity criteria to capture details:

(IV) **Folds**, expressed as the shape and frequency of wrinkles along the hemline.

This choice allows characters of different sizes to have the same folds frequency, instead of the same number of folds around their dresses, so that it looks like the same fabric has been used.

Let now discuss how to relate these style criteria to 2D garment sketches. Preserving the relative location and proportion of a garment (I) implies to find correspondences between the bodies of the drawn source and the target 3D character. A first method could be to use the drawn outline of the dressed character's body in the sketch to compute geometry correspondances with the target body. This method seems rather unstable for two main reasons. First, as seen before, this outline is more or less precisely drawn in fashion sketches (see Figure 5.2(B)), and in many cases, most of it is hidden by the cloth itself (like the legs in Figure 5.2(D)). Second, we would be limited to transfers between characters of similar types, for instance human bodies. In order to account for characters with of different type, for example the starfish and frog of Figure 5.4(C,D), we

seek for a topology-related cloth positionning criteria. To this end, we rely on skeleton representations. Skeletons and medial axis are indeed very common features both for analysis of 2D shapes and for animation of 3D characters. Positionning a 2D skeleton ontop of a sketch representing a character seems like a reasonable task that a user can perform. As for 3D, rigging characters remains a non trivial but necessary task for animation. This task has been made simple with new interactive systems, like Mixamo[1], which allow to easily compute a basic rig of a 3D character.

The scale and fit criteria (II,III) imply to identify the parts of the garment that are tight to the character and the parts that are loose, in other terms, the parts for which the garment's silhouette should follow the body's, and the parts for which the silhouette should match the one in the sketch. Ideally, the sketch would contain the full outline of the character's body, and we could, as most sketch-based interfaces [TWB+07] do, measure distance to body all along the garment's silhouette. However, as discussed previously, this is not the case for most garment sketches, and sometimes estimating oneself the hidden parts of the character's body is not a trivial task (think of the legs of Figure 5.2(D) for example), and we do not need it. Indeed, the shape and fit criteria do not require any distances to body, but only the information of tight or loose areas in the garment. To put it differently, the position or geometry of the underlying limbs are not supposed to change the shape of the garment if it is loose, and are completely defining it if it is tight. We decide to read the sketched garment as a composition of tight and loosepatches. Because most of sketched garments can be interpreted as loose or tight, we leave the decomposition to the user.

The original shape (III) criterion presented in [BSBC12] was translated into preservation of the normals in the world's frame. This formulation allowed to perform transfer even in the case of strong differences in geometry, as for example transferring a loose dress from a woman of standard morphology to a pregnant woman body. This works well when the source and target characters share the same pose. However, a change of pose may imply a rotation of the global shape of the garment, and therefore of the normals. Thus, our insight is to adapt this criterion in order to express the normals in the skeleton's frame of the associated body or limb.

All normals of the garments are not directly readable in the sketch. Inferring normals of a surface using a line drawing sketch usually requires more information than contour curves, one may use, for example the projected cross-section curves of the surface [SBSS12]. On the other hand, we discussed in Section 2.2.1 the importance of the silhouette curves in the understanding of the shape and in particular its curvature. We saw that, if the surface is C^2 perfectly developable, then the silhouette is necessarily a straight line (cf Chapter 3). As said before, most garments are however not perfectly developable, and the silhouettes drawn in fashion illustration are not necessarily straight lines. In the case of quasi-developable surfaces containing creases, the silhouette curve

[1] https://www.mixamo.com/

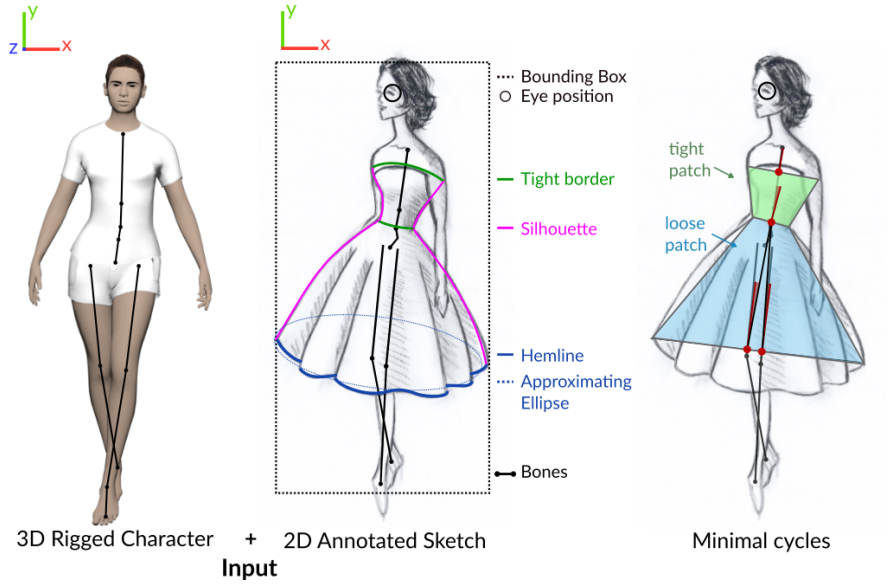


FIGURE 5.5: The input of our algorithm includes a 3D rigged character and a 2D sketch. The user is asked to annotate the sketch with virtual bones in correspondence with the bones of the 3D character.Borders and silhouettes of the garment are also over-sketched. They are used to compute a decomposition of the garment into patches and infer their relative location and orientation with respect to the 2D skeleton.

may correspond to a non planar rim [JHR+15]. However, if the garment contains no creases and the pose is not accidental, it seems reasonable to interpret the silhouette as the projection of a planar rim on the garment's surface, in a plane orthogonal to the viewing direction. Normals on this planar rim can then be directly computed from the normals of the sketched silhouette. In addition, to fulfill the fold criteria (IV) we also account for variations of these normals due to tubular folds and which are generally displayed on the sketch through undulating borders called hemlines (cf Chapter 4).

5.1.2 User Inputs

The input of our algorithm is two-fold, as illustrated in Figure 5.5. The 3D rigged character on which a 3D garment is to be synthesized, and a pre-existing drawing depicting the garment to be transferred. Even though much progress has been recently done in the vectorization and cleaning of paper drawings [FLB16] and in learning-based modeling from sketches [DAI+18, SBS19], automatic line drawing interpretation is still a challenge. In our implementation as in many similar works [LPL+17, JHR+15], we rather ask the user to annotate the sketch to ease interpretation. This is done as follows.

Garment contours are over-drawn using Bézier curves of different colors representing silhouette (pink), tight border (green), and loose border, also called hemline (blue). As for our method presented in Chapter 3, we chose cubic Bézier curve representation because they are transparent to the user, since they are a common tool in most commercial 2D

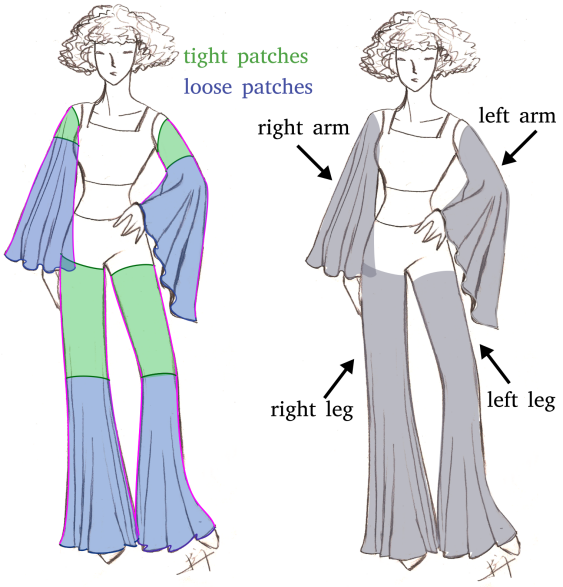


FIGURE 5.6: Example of input sketch containing 4 garments, corresponding to the 4 different connected graph implied by the drawn contour curves in the sketch. Each garment is decomposed into connected cycles of contour curves, which are associated to garment patches to be synthesized in the target 3D space. Those garment patches are identified as tight or loose depending on the annotation of the borders.

vector graphics softwares. To retrieve the local orientation of the different pieces of the garment, we also ask the user to depict the bones of the limbs wrapped by the garment (black segments in Figure 5.5, middle). These bones are manually matched to their counterparts in the 3D rigged character.

In the case of a loose border with folds, additionnal annotations are necessary to reconstruct the fold's shape: approximate directrix curve, to which an ellipse is automatically fitted (dotted green curve in Figure 5.5, middle), the bounding box of the character and the approximate location of the eyes (black circle), which will be used as a proxy for the height of the viewpoint from which sketched person is seen, see Chapter 4 for details.

As a precomputation step, we decompose the set of garment contours as a list of minimal cycles bounded by curves alternating between *silhouette-type* and *border-type* (see Figure 5.5-left). We compute a graph representation of the sketch were nodes are the intersection of the contour curves, and edges are connecting the intersections that belong in the same contour curve. The graph is decomposed into a set of connected subgraphs. Our algorithm creates a garment surface for each subset of contour curves corresponding to a connected subgraph. For example, in the sketch displayed in Figure 5.6, we can see 4 connected subgraphs of contour curves. Each of the associated garment surfaces is computed independently. In the following of this chapter, we will describe the process for only one of them.

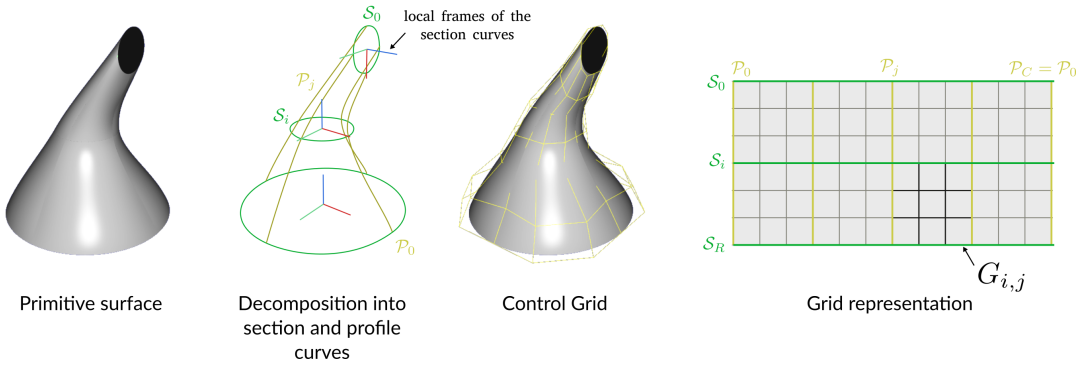


FIGURE 5.7: Example of garment primitive, defined by a set of section $\{\mathcal{S}_i\}$ and profile $\{\mathcal{P}_j\}$ spline curves. Each section curve is planar and associated to a local frame. The set of curve is topologically similar to a closed grid.

We compute the list of minimal cycles of the connected graph using the standard algorithm presented in Section 3.2.1. This list of minimal cycles decomposes the garment to be synthesized into *garment patches*. Each garment patch is automatically identified *loose*, when one of its border is loose, or *tight* otherwise (see an example in Figure 5.6).

In the following section, we describe the 3D representation we choose for each of these garment patches.

5.1.3 3D Primitive surface

We aim at a mathematical garment representation suited to adapt to an arbitrary target character, and to the complexity of the garment represented, while being able to access and deform silhouettes and borders efficiently, because these features are the main component of line drawing sketchs that depict the style criteria we want to transfer.
We observe that most garments can be decomposed into cloth pieces wrapped around one or several limbs of the character, that we call *garment patches*. We propose a representation able to model such patches. This excludes the parts of the cloth were these garment pieces meets, such as the chest, and the pelvis. Other representations can be considered to model such parts, and we discuss them in Section 5.4.3.

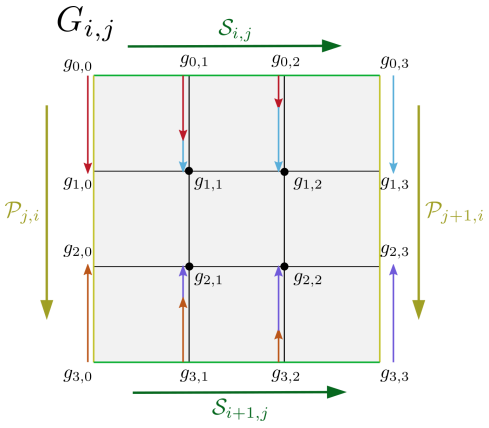
We propose a parametric garment primitive defined by two sets of 3D G^1-continuous cubic Bézier splines: $\{\mathcal{S}_i\}_{i\in\{0..R\}}$, called *sections*, and $\{\mathcal{P}_j\}_{j\in\{0..C\}}$, called *profiles* (see Figure 5.7). The network of Bézier splines forms a cylindrical grid of R rows and C columns of Bézier segments, such that:

- $\mathcal{P}_C = \mathcal{P}_0$
- $\forall i \in \{0..R\}$, the spline curve $\mathcal{S}_i$ contains C Bézier segments: $\{\mathcal{S}_{i,k}\}_{k=0,..,C-1}$
- $\forall j \in \{0..C\}$, the spline curve $\mathcal{P}_j$ contains R Bézier segments: $\{\mathcal{P}_{j,l}\}_{l=0,..,R-1}$

- $\forall j \in \{0..C\}, \forall i \in \{0..R\}, \mathcal{P}_{j,i}|_0 = \mathcal{S}_{i,j}|_0$

where $\mathcal{S}_{i,j}|_0$ (respectively $\mathcal{P}_{j,i}|_0$) represents the first control point of the Bézier segment $\mathcal{S}_{i,j}$ (respectively $\mathcal{P}_{j,i}$). We also add the constraint that the section splines $\mathcal{S}_i$ are planar in 3D, and we associate to each section spline a local frame in this plane. We denote as the *axis* of the primitive the 3D polyline made of the centers of these local frames.

We define a tensor product spline surface from this network of Bézier splines where each patch i, j is defined by 16 control points $G_{i,j} \in \mathcal{M}_{4,4}(\mathbb{R}^3)$ whose borders are composed of the control points of the Bézier segments $\{\mathcal{S}_{i,j}, \mathcal{S}_{i+1,j}, \mathcal{P}_{j,i}, \mathcal{P}_{j+1,i}\}$.



For the sake of readability, we note the coefficients of each $G_{i,j}$ as $(g_{k,l})_{k,l=0,1,2,3}$, where:

$$\begin{cases} \{g_{k,0}\} & \text{are the control points of } \mathcal{P}_{j,i} \\ \{g_{k,3}\} & \text{are the control points of } \mathcal{P}_{j+1,i} \\ \{g_{0,l}\} & \text{are the control points of } \mathcal{S}_{i,j} \\ \{g_{3,l}\} & \text{are the control points of } \mathcal{S}_{i+1,j} \end{cases}$$

The 4 inner control points are computed as:

$$\begin{cases} g_{1,1} &= g_{0,1} + \frac{2}{3}(g_{1,0} - g_{0,0}) + \frac{1}{3}(g_{1,3} - g_{0,3}) \\ g_{1,2} &= g_{0,2} + \frac{1}{3}(g_{1,0} - g_{0,0}) + \frac{2}{3}(g_{1,3} - g_{0,3}) \\ g_{2,1} &= g_{3,1} + \frac{2}{3}(g_{2,0} - g_{3,0}) + \frac{1}{3}(g_{2,3} - g_{3,3}) \\ g_{2,2} &= g_{3,2} + \frac{1}{3}(g_{2,0} - g_{3,0}) + \frac{2}{3}(g_{2,3} - g_{3,3}) \end{cases} \tag{5.1}$$

Then the bicubic surface patch is defined as:

$$\mathcal{G}_{i,j}(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M G_{i,j} M^T \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}^T, (u,v) \in [0,1] \times [0,1] \tag{5.2}$$

where $M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

For each cycle in the sketch, the corresponding garment patch will be synthesized using this primitive representation. Borders in the sketch are thus associated to the boundary section curves S_0, S_R of the primitive surfaces that corresponds to cycles containing the border curve.

In the following of this chapter, we describe the four steps of our algorithm to synthesize a garment on an arbitrary character using an input annotated sketch. Recall that we defined in Section 5.1.1 four criteria which depict the style of a garment. The goal of first step of our algorithm is to create for each garment patch in the sketch an initial

primitive surface that satisfies the scale (I) criteria. This means that the initial surface needs to transfer the relative location and orientation of the garment depicted in the 2D sketch on the target character.

5.2 Positioning a garment patch using a sketch

The first step of our method (see Figure 5.3) consists in inferring an initial garment patch for each cycle of contour curves in the sketch. This very basic initial surface should preserve the relative location and orientation drawn in the sketch, and proportions, with respect to the body. This positionning step not only enforces the scale (I) criteria for transfer, but will also allow us to formulate the shape and fit (II,III) criteria in the target character's space (see Section 5.3). We use the correspondances between the 2D and 3D skeleton to translate the location and orientation described by the sketch to the target character's space (Section 5.2.1), and then compute an initial surface for each garment patch (Section 5.2.2).

5.2.1 Proportion-preserving garment positioning

First, we want to position the boundaries of the garment patch in the target character's space. For each border of the patch (tight or loose), we compute a plane and corresponding local frame in 3D.

Transferring location We first determine the position of the garment in the 2D sketch by computing the relative location of a set of keypoints. Those keypoints are located at the intersections between the bones and the line joining the extremities of the border curves in the sketch. We denote as $\{\tilde{b}_i^k, t_i^k\}$ the set of keypoints associated to the border i, where $t_i^k \in [0,1]$ is the linear coordinate of the intersection point in the bone segment $\tilde{b}_i^k$. This method allows us to process all bones crossed by the border while not suffering from the bias in the shape of the border curve due to perspective (see for instance Figure 5.5-right, where the front of the skirt looks longer than the silhouettes due to a camera position located higher than the border).

Using the direct correspondance between 2D and 3D skeletons, each keypoint is associated to a position $o_i^k \in \mathbb{R}^3$ in the target character's space. The purpose of this section is to associate these positions to a local plane, of normal vector n_i^k. We will then average all planes relative to the same border curve to compute one plane for each boundary curve in the 3D space.

To simplify the study we first illustrate the problem on patches for which each border crosses only one bone $\tilde{b}_i$. In this case, each border i is associated to only one position

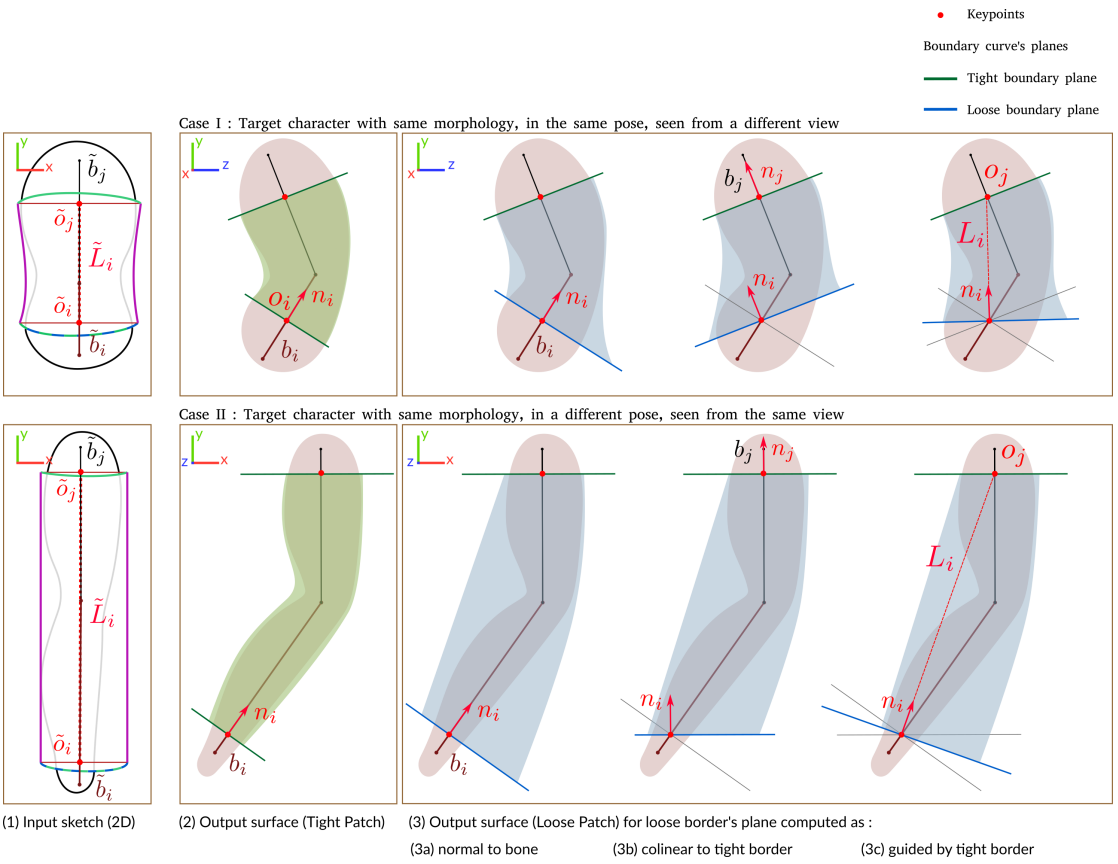


FIGURE 5.8: Illustration of the choice of boundary planes on two cases of input sketches with symmetric cycles (1). We first associate to each border i in the input sketch a position o_i in the 3D skeleton of the target character.
The second column (2) shows the result of our method to compute the boundary planes in the case of a tight patch, i.e. when both border curves in the sketch are annotated as tight : tight boundaries (green) lie on a plane orthogonal to the direction of the bone they are crossing.
The rest of the columns (3a,b,c) are showing different possible results of boundary plane when the patch is loose, i.e. if one of the border is annotated as loose (here the bottom ones for each input sketch). The first one (3a) uses the same process as for tight border : the plane containing the loose boundary is orthogonal to the bone it crosses. The second one (3b) computes the plane of the loose boundary as parallel to the one of the tight boundary of the patch. The last one (3c) uses as normal of the loose boundary plane the line L_i linking the position o_j in the 3D skeleton associated to the tight border of the cycle to the position o_i associated to its loose border.
For each method, we infer manually the silhouette of a surface bounded by those planes, and matching the normals of the silhouettes of the input curve in the case of loose patches, while fitting the body of the target character in the case of tight patch. In Section 5.2.1 we discuss the advantages and drawbacks of using each method.

in 3D $o_i = o_i^0$, and unknown normal vector $n_i = n_i^0$. The plane (o_i, n_i) will, in this case, be the plane of the boundary curve associated to the border.

Computing this normal in the 3D target character space using the 2D sketch raises several challenges. First, because the sketched border is a 2D projection of an hypothetic 3D boundary, it only provides partial information on the plane on which the boundary curve is lying. In particular, the depth ($z-$component) of the normal vector of the plane is not readable in the sketch. Even if we want to transfer the boundary's plane to a target character with similar morphology and pose than the one depicted in the sketch, we would need to adjust the plane so that the 3D representation remains plausible, see for example the first row example of Figure 5.8. Secondly, the target character may have a different position than the one in the sketch, as in the second row of Figure 5.8. Our transfer needs to account for these changes in position while being visually similar to the sketch.

Transferring orientation Let's first study how to compute this orientation in what we call symmetric inputs, meaning input cycles of contour curves where border lines are parallel and each border is crossing bones in an orthogonal direction. In Figure 5.8, we show two examples of symmetric inputs and two examples of possible transfer in the 3D space. In both examples, the goal is to find a plane in the 3D space corresponding to each border curve in the sketch.
For *tight borders*, we assume the body guides the orientation of the border. So we compute the normal vector n_i in the direction of the bone b_i corresponding to $\tilde{b}_i$ in the 3D skeleton of the target character (see Figure 5.8(2)). However, if the border is loose, this choice (Figure 5.8(3a)) is not providing a satisfying result. In particular, the example of the top row shows a loose border looking unnatural, and silhouettes with very different length, which was not the case in the input sketch.
In the case of *loose borders*, the orientation of the bones underneath the cloth should not guide directly the normal n_i of the plane containing the loose boundary. Intuitively, we expect the position of the hanging points of the cloth, meaning the tight boundary of the patch, to have an influence on n_i. One method would be to chose n_i colinear to the normal vector of the plane containing the tight boundary curve. As illustrated in Figure 5.8(3b), this would also lead to silhouette curves with significantly different lengths, and possibly break the scale criteria. In particular, the pants in the second row seem to have been shortened during transfer. Another method could be to define the normal as the direction of the line $\tilde{L}_i$ joining the position o_j associated to the tight boundary and the one associated to this loose boundary o_i (see Figure 5.8(3c)). The orientation of the planes of the target garment's boundaries orthogonal to this normal mimicks a pendular motion, and provides good results in most examples. We therefore use the lattest method to compute the orientation in the case of loose borders.

Let us now treat the case of asymmetrical input patches, i.e. inputs for which the borders are not necessarily parallel to each other, and the bones are not crossing orthogonally to the patch border. First, we compute an initial guess of n_i with the method used for symmetrical inputs : in the direction of the bone b_i if the boundary is tight, and in the direction of the line L_i joining the the locations o_i, o_j associated to each boundary of the patch if the boundary is loose. We then compute an angle α_i depicting the inclination of the border curve in the sketch, and rotate the initial normal vector around the $z-$axis in the 3D space to obtain n_i. The angle α_i is computed differently in the sketch depending on whether the border is annotated as loose or tight.

As discussed previously, the orientation of a tight border is only defined by the local orientation of the body. Therefore, we read on the input sketch the angle α_i between the normal direction of the border line and the direction of the bone crossing the border. For loose borders, the orientation does not depend on the bone's direction, but on the position of the hanging points. In this case, α_i is computed as the angle between $\tilde{L}_i$ and the border's direction.

We have associated to each border in the 2D sketch a plane for the corresponding boundary curve in the 3D space. Recall that we only studied the case on which each border crosses only one bone in the sketch. If a border crosses multiple bones, then we use the same process to compute one plane (o_i^k, n_i^k) for each bone crossed by the border i. The plane of the corresponding boundary curve is computed by averaging the origins o_i^k and normals n_i^k.

5.2.2 Initial garment surface

At this point, we have associated to each border in the 2D sketch a plane for the corresponding boundary curve in the 3D space. We use these planes to compute an initial garment surface which fits the target character in scale and orientation. We first compute one 3D boundary curve for each border in the sketch, and then use this pair of boundary curves to generate a surface patch.

Computing boundary curves The goal is to compute a 3D boundary curve of the initial garment surface for each border i so that it envelops the mannequin body within the plane we computed before. To this end, we compute for each border plane, the convex hull of the intersection points between the 3D plane and the mannequin's surface mesh.

We have a polyline representing the enveloppe of the character's body, and we want a spline fitting at best this polyline with the minimum possible number of control points. We use the Ramer-Douglas-Peucker algorithm [DP73] to select points in the polyline that are the most relevant to depict its shape. Then we compute a cubic Bézier spline for

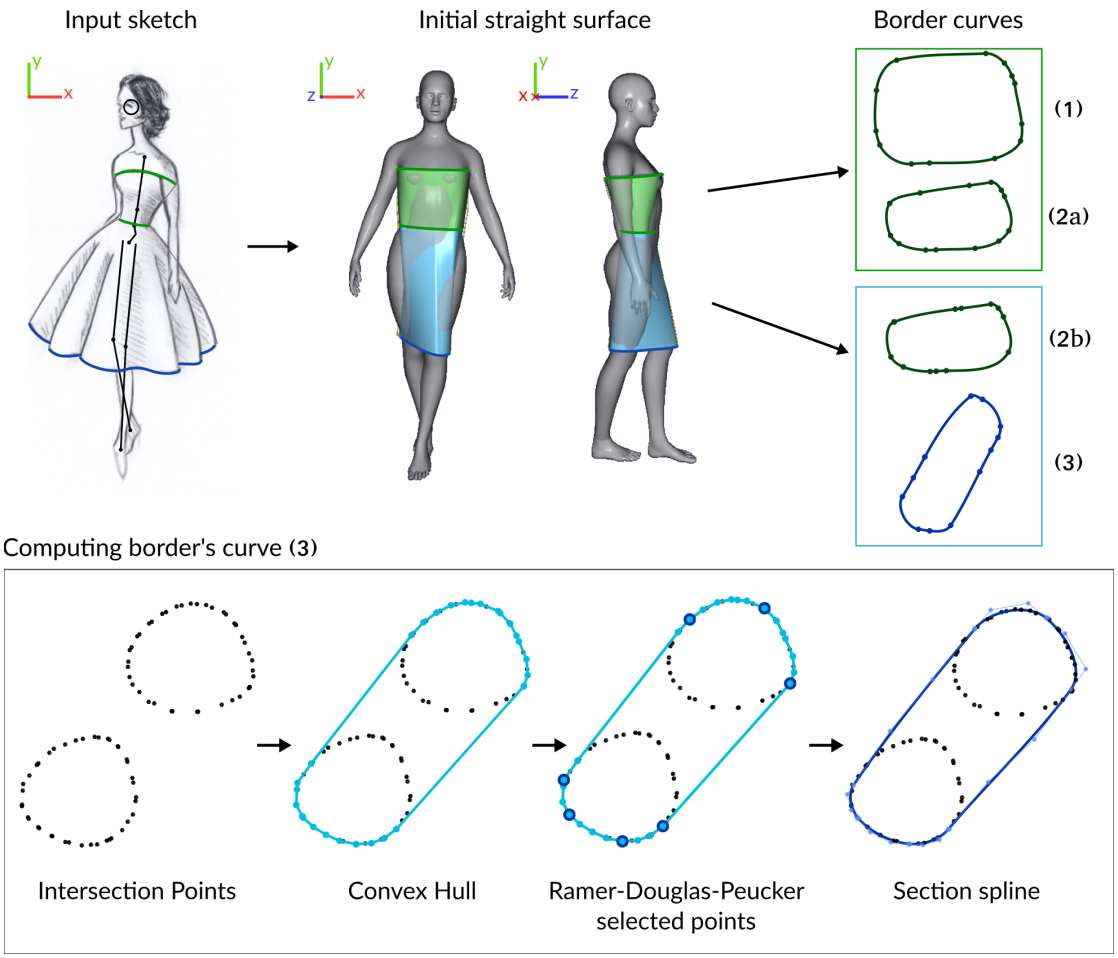


FIGURE 5.9: Example of initial garment surface a garment composed of one tight patch (green and one loose patch (blue). Each patch is represented in the target character's space with a primitive composed of two planar boundary curves (right). This curves are computed as a spline approximation of the convex hull of the set of points on the target's body lying in the border's plane.)

which the junction points are those selected points, while approximating in a least square sense the rest of the points in the polyline (see an example of result in Figure 5.9(3)).

Note that doing the intersection between the whole body mesh and the border's plane may capture vertices from unwanted limbs. For example, in the Figure 5.9, computing the intersection of the plane containing the top tight boundary curve (1) with the whole mesh representing the target character would capture vertices of the arms of the character. To overcome this issue, we use the skinning weights of the rigged character, and only select the vertices for which the weight is above a fixed threshold for the bones crossed by the border. In some parts of the body, the threshold may need to be adjusted to get a satisfying result, mainly in joint areas, such as in the chest.

Straight primitive surface We now have computed the two 3D planar surface upper and lower boundary curves, we create an initial surface. Following the primitive surface

representation introduced in Section 5.1.3, we define two planar section curves $\mathcal{S}_0, \mathcal{S}_1$ and their associated local frames, and connect them with $C+1$ profile curves $\{\mathcal{P}_j\}_{j\in\{0..C\}}$ as follows.
We set the origin of the local frame of the section curves to the barycenter of the associated planar boundary curve, and set the normal of the local frame to the normal of the plane on which the boundary lies. To obtain the section curves, we reparameterize the two planar boundary curves of the patch, such that their junctions share the same polar angle in their local frame. Profile curves $\mathcal{P}_j$ are then created as straight lines linking the C resulting junctions of the section curves. For further processing we define them as cubic Bézier curves with control points:

$$\{\mathcal{S}_{0,j}|_0,\ \frac{1}{3}(2\mathcal{S}_{0,j}|_0 + \mathcal{S}_{1,j}|_0),\ \frac{1}{3}(\mathcal{S}_{0,j}|_0 + 2\mathcal{S}_{1,j}|_0),\ \mathcal{S}_{1,j}|_0\},\ j = 0, .., C$$

See Figure 5.9 for a result. We obtain an initial garment surface, composed of 2 section curves $\mathcal{S}_0, \mathcal{S}_1$ and C straight profile curves $\mathcal{P}_j,\ j = 0, .., C$.

5.3 Transfer shape and fit

Our initial garment surface satisfies the criteria of scale (I) induced by the sketch, but not the fit (II), shape (III) and folds (IV). In this section, we show how we apply the fit and shape criteria on the initial surface using information extracted from the input sketch (Section 5.3.1). We then present an algorithm to solve potential collisions in the resulting surface (Section 5.3.2). Finally, we present our method to apply folds (IV) on the surface (Section 5.3.3).

5.3.1 Shape and fit

As discussed in Section 5.1.1, the shape and fit criteria express the fact that the transferred garment should fit the target body in tight areas while preserving normals of the source garment in loose areas. In our work, garments are decomposed in patches that are either tight or loose. Remember that we call a tight surface a surface where both boundary curves fit the body tightly, so must do the entire surface. We call a surface loose when one boundary is tight but the other one is loose. We describe in two separate processes how we enforce the fit (II) criteria in tight patches, and the shape (III) criteria in loose patches.

At this point, we have two section curves $\mathcal{S}_0, \mathcal{S}_1$, and $C+1$ profile curves $\mathcal{P}_j$, $j \in \{0..C\}$, that are straight lines, represented with Bézier curves. For each profile curve $\mathcal{P}_j$, we compute the plane fitting 4 points made of the endpoints of $\mathcal{P}_j$ and the origins of the local frames corresponding to each section curve, we call this plane *profile plane* associated to $\mathcal{P}_j$. Our goal in both cases is to deform the profile curves so that the

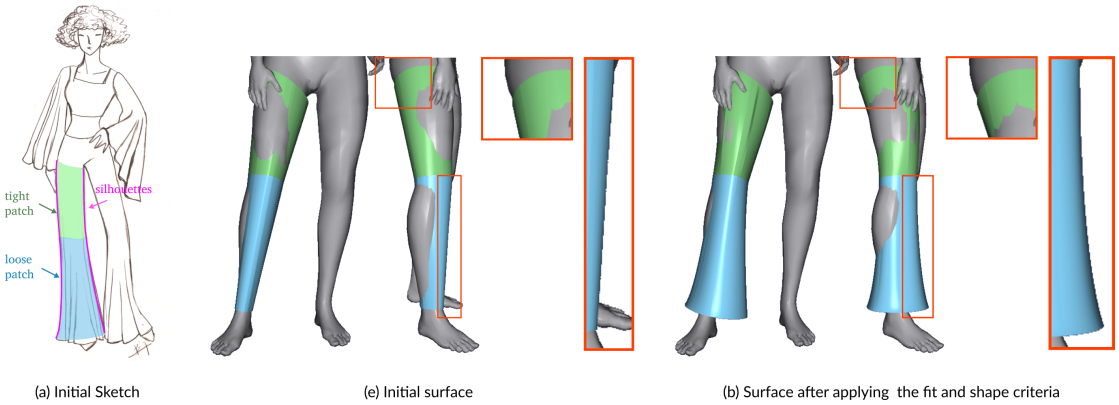
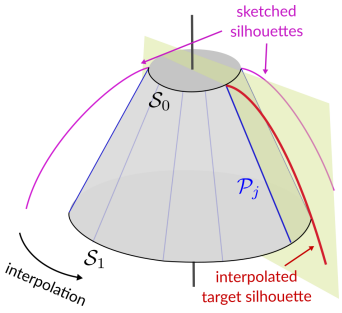


FIGURE 5.10: Illustration of our algorithm to transfer fit and shape from a sketch in a garment's surface. Depending on whether the garment patch is tight or loose, the profile curves are set to match either the underlying body's silhouette (green patch), or the sketched garment silhouette (blue patch).

surface either fits the character (tight garments) or mimicks the shape displayed in the sketch (loose garments).

In the case of tight garments, the shape is fully guided by the body on which it is transferred. Since both tight boundaries (section curves) already fit the body shape, we only want to modify the inner points of the Bézier profile curves, while keeping their endpoints unchanged. The two free control points for each profile curve define the tangent vectors of the curve endpoints. In order to adapt to the body shape, we compute the tangent vectors by intersecting the profile plane, and the tangent plane of the body mesh at the corresponding closest point (see an example of result in Figure 5.10,green patch).

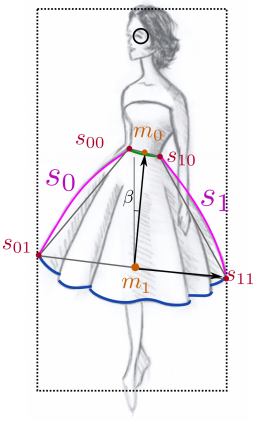
In loose patches, our goal is to transfer the global shape from the sketch to the primitive garment surface. Note that the tubular folds are integrated later. Recall the shape criterion (III) as formulated by Brouet et al. [BSBC12]: in loose areas, the style of the garment is expressed by the direction of its normals in the world's frame. As discussed in Section 5.1.1, in order to account for garment transfer to characters in different poses, we adapt the criterion to express the normals in the skeleton's frame of the associated body or limb.



Our goal is to compute new profile curves still fitting the character in the tight boundary of the loose patch, while having normals matching the ones displayed by the silhouette curves in the sketch. The main idea is to rotate the sketched silhouette curves around the primitive axis by keeping its anchor point (one of its endpoints) on the tight boundary curve section, and by deforming it slightly in order to interpolate the left and right sketched silhouette curves, which are not identical. To achieve this, we process as follows.

In the following, we assume that, among the two initial section curves $\mathcal{S}_0, \mathcal{S}_1$, the tight boundary curve is $\mathcal{S}_0$, while the loose one is $\mathcal{S}_1$. In this configuration, for each profile curve $\mathcal{P}_j$, the first endpoint of the curve $\mathcal{P}_j|_0$ is connected to the tight section boundary spline while the last endpoint $\mathcal{P}_j|_1$ of $\mathcal{P}_j$ is connected to the loose section boundary spline.

We denote as s_0, s_1 the two 2D silhouette curves drawn in the sketch, oriented such that their first endpoints are connected to the tight border, and their last endpoints are connected to the loose border in the sketch. As stated in Section 5.1.2, these curves are represented as cubic Bézier splines. We first compute the splines $\hat{s}_0, \hat{s}_1$ that represent s_0, s_1 in a local 2D frame induced by the garment's borders drawn in the sketch. Let us denote as s_{00}, s_{01} (respectively s_{10}, s_{11}) the endpoints of s_0 (respectively s_1), and β the angle between $Oy = [0; 1]$ and $\overrightarrow{m_0 m_1}$ where $m_0 = \frac{1}{2}(s_{00} + s_{10}), m_1 = \frac{1}{2}(s_{01} + s_{11})$. Then,



$$\begin{cases} \hat{s}_0 &= \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} R_{-\beta}(s_0 + \overrightarrow{s_{00} m_0}) \\ \hat{s}_1 &= R_{-\beta}(s_1 + \overrightarrow{s_{10} m_1}) \end{cases}$$

where $R_{-\beta}$ is the 2D rotation matrix of angle $-\beta$. In this local frame, the first endpoints of $\hat{s}_0$ and $\hat{s}_1$ have the same coordinate.

For each profile curve $\mathcal{P}_j, j \in \{0..C\}$, we compute the polar angle θ_j of the first endpoint of $\mathcal{P}_j$ in the local frame of the corresponding tight section curve. The goal is that the profile curve at angle 0 has the same visual aspect than s_0 and that the profile curve at angle π has the same visual aspect than s_1, while guaranteeing the continuity $\mathcal{P}_0 = \mathcal{P}_C$. Therefore, we compute a target 2D profile $\hat{\mathcal{P}}_j$ as

$$\hat{\mathcal{P}}_j = \begin{cases} (1 - \frac{2\theta_j}{\pi})\hat{s}_0 + \frac{2\theta_j}{\pi}\hat{s}_1 & \text{if } \theta_j \in [0, \pi] \\ (1 - \frac{\theta_j - \pi}{\pi})\hat{s}_0 + \frac{\theta_j - \pi}{\pi}\hat{s}_1 & \text{if } \theta_j \in [\pi, 2\pi] \end{cases}$$

We then compute a 3D position of this curve in the profile plane, such that its endpoint $.\hat{\mathcal{P}}_j|_0$ matches the endpoint of the profile curve $.\mathcal{P}_j|_0$. We proportionally scale this 3D curve so that its height matches the height of the initial profile curve in the direction of the primitive axis, to obtain our new profile curve.

This method ensures that

- the height of the garment patch does not vary much after transformation,
- the normals of the interpolated target silhouettes are preserved in the local frame induced by the profile plane,
- the endpoint $\mathcal{P}_j|_0$ connected to the tight boundary section curve $\mathcal{S}_0$ remains unchanged.

The endpoint of $\mathcal{P}_j$ connected to the loose boundary section curve $\mathcal{S}_1$ on the other hand, is generally changed by the process. We need to update the section curve $\mathcal{S}_1$, and its associated local frame to account for these changes, to keep the cylindrical grid form described in Section 5.1.3.

First, the endpoints $\mathcal{P}_j|_1$ need to lie on a plane, so we project them onto their closest fitting plane. We compute a new local frame from the section spline $\mathcal{S}_1$ using as normal n_1 the normal of the fitting plane, and as origin o_1 the barycenter of the endpoints $\mathcal{P}_j|_1$. In this plane, we compute a new closed section spline $\mathcal{S}_1$ made of C Bézier segments as follows. Junction points (endpoints of Bézier segments) are the endpoints $\mathcal{P}_j|_1$ of the new profile curves. The inner points of the Bézier curves, depicting the tangents of the curves at junction points are estimated as follows. For each three consecutive junction points j_0, j_1, j_2, we estimate the tangents at j_1 using a locally quadratic approximating curve of endpoints (j_0, j_2). We iterate this process to compute the tangent at all junction points (recall that the section curves are closed curves), and compute the corresponding Bézier control polygon.

5.3.2 Solving collisions with the body

At this step, the synthesized garment surface modeled as a Bézier surface matches the scale (I), fit (II) and shape (III) criteria with respect to the sketch, but it may be in collision with the body surface, stored as a triangular mesh. Collision handling between a meshed surface and a body is a well studied problem for garment modeling [BWK03, JYSL19]. Inter-penetration is typically solved in pushing penetrating vertices outside of the body volume along a vector normal to the closest body position [RMSC11, GRH+12].

Even though inspired by these approaches, our problem states differently, since we have to solve penetrations between a parametric surface and a mesh. We propose to interactively subdivide the Bézier patches where maximal penetrations occur, and to deform

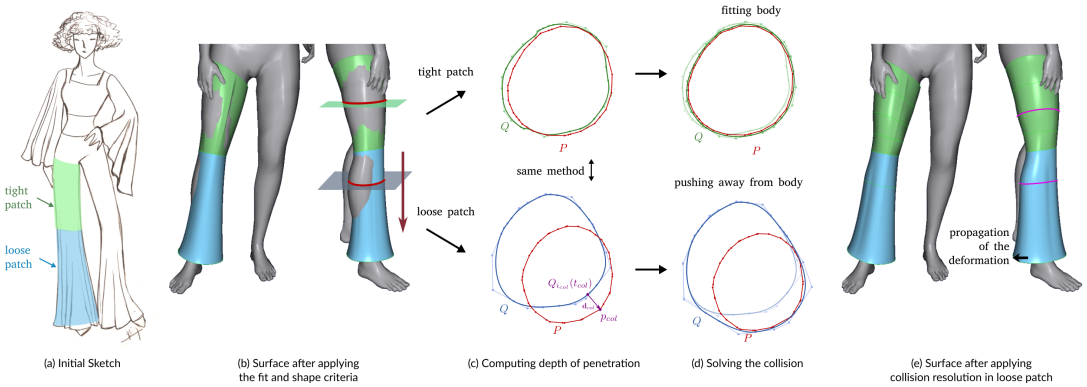


FIGURE 5.11: Overview of our algorithm to solve collisions between a garment patch and the target body mesh. We march along the axis of the primitive surface, while measuring a depth penetration distance. If the garment patch is loose, we stop at first local maximum of this distance, solve the collision by inserting a new section and pushing it outside of the body, and propagate the deformation of the surface on all the sections below. If the garment is tight, then we find the maximal distance, insert a section and displace it so that it fits locally the body.

the newly inserted control points in order to push the garment surface outside the body as illustrated in Figure 5.11.

Collisions are detected within planes by marching along the axis of the primitive surface. For each plane, we evaluate a collision distance by computing the cubic spline curve Q of intersection with the garment surface, and the polyline P of intersection with the character's body. Q is composed of C Bézier curves Q_i of control points denoted $q^i_j, j = 0, .., 3$.

$$\forall t \in [0,1], Q_i(t) = \sum_{j=0}^{3} q^i_j B^3_j(t)$$

We compute the collision distance as

$$d_{col} = \max_{p,i,t} \left\{ \|\overrightarrow{pQ_i(t)}\| \mid \overrightarrow{pQ_i(t)} \cdot Q'_i(t) = 0 \text{ and } Q_i(t) \text{ is interior to } P \right\}$$

.

We denote $(i_{col}, t_{col}, p_{col})$ the parameters of the deepest collision of Q in P : $d_{col} = \|\overrightarrow{p_{col}Q_{i_{col}}(t_{col})}\|$ (see an example in Figure 5.11).

In the case of loose garments, the action of gravity makes the fabric fall on the highest part of limbs first. We model this behavior by marching the planes along the axis from tight boundary to loose boundary, and stop when the first encountered penetration reaches its maximum.

The parametric garment patch is then subdivided by inserting Q as an extra section curve in the primitive surface, in order to gain sufficient degrees of freedom to allow its

collision solving deformation. The actual deformation of the Bézier curve is computed as a quadratic energy minimization problem, weighting between repulsion of the deepest colliding position and preservation of the current shape as measured by the preservation of curve derivatives.

Using the previous notations, we set $\tilde{p}_{col} = p_{col} + \varepsilon \frac{\mathbf{d}_{col}}{||\mathbf{d}_{col}||}$, where $\varepsilon = 10^{-1}$, as the target point the curve needs to reach, while keeping at best its initial shape. The spline is closed and C^0 continuous, so the control points of Q_i verify: $q_3^{i-1} = q_0^i$, and $q_0^i = q_3^{C-1}$. We also denote by $\hat{q}_j^i$ the initial coordinates of the control points. The collision is then solved by minimizing the following quadratic energy on the control points of all Bézier segments Q_i of the spline Q in the plane.

$$E_{collision} = E_{col} + E_{stretch} + E_{G^1} + E_{data} \ , \tag{5.3}$$

where

- $E_{col} = ||\tilde{p}_{col} - Q_{i_{col}}(t_{col})||^2$
  aims at pushing the spline outside of the body at the location computed as the point of deepest penetration
- $E_{stretch} = \sum_{i=0}^{C-1} \sum_{j=0}^{K-1} ||\hat{Q}_i'(\frac{j}{K}) - Q_i'(\frac{j}{K})||^2$
  to minimize variations in the derivative.
  We fix the number of tangents sampled in each Bezier curve $K = 4$, which is enough to depict the shape of the degree 3 curve.
- $E_{G^1} = \sum_{i=0}^{C-1} ||(q_2^i - q_3^i) - \alpha_i(q_0^{i+1} - q_1^{i+1})||^2$, with $\alpha_i = \frac{||\hat{q}_2^i - \hat{q}_3^i||}{||\hat{q}_0^{i+1} - \hat{q}_1^{i+1}||}$
  to enforce tangent continuity at junctions. Here, we extend the notations to account for the fact that the section curve is closed : $q_0^N := q_0^0$ and $q_1^N := q_1^0$.
- $E_{data} = \sum_{i=0}^{C-1} \omega_i ||q_0^i - \hat{q}_0^i||^2$ with $\omega_i = e^{-\frac{d_i^2}{2\sigma^2}}$
  where d_i is the projected distance of q_0^i to P if it lies inside P, and 0 otherwise, and $\sigma = 0.2 \ \max_{i \in \{0..N-1\}}\{d_i\}$, keeps the initial position of the control points that are far from p_{col}.

The energy $E_{collision}$ is minimized in the least square sense, and the process is iterated until the spline is completely outside of P. Once the collision is solved, we propagate the deformation of the surface in all following section curves $\mathcal{S}$ below it by applying the displacements $\delta_i := q_0^i - \hat{q}_0^i, i = 0, .., N-1$ to all junction points of $\mathcal{S}$. Then, the marching algorithm is pursued until reaching the lowest extremity of the axis.

We now explain how to deform the surface in the case of a tight garment patch. Contrary to loose garments, the geometry of tight garments can be considered as independent from the action of gravity. Therefore, we iteratively solve the deepest detected intersection

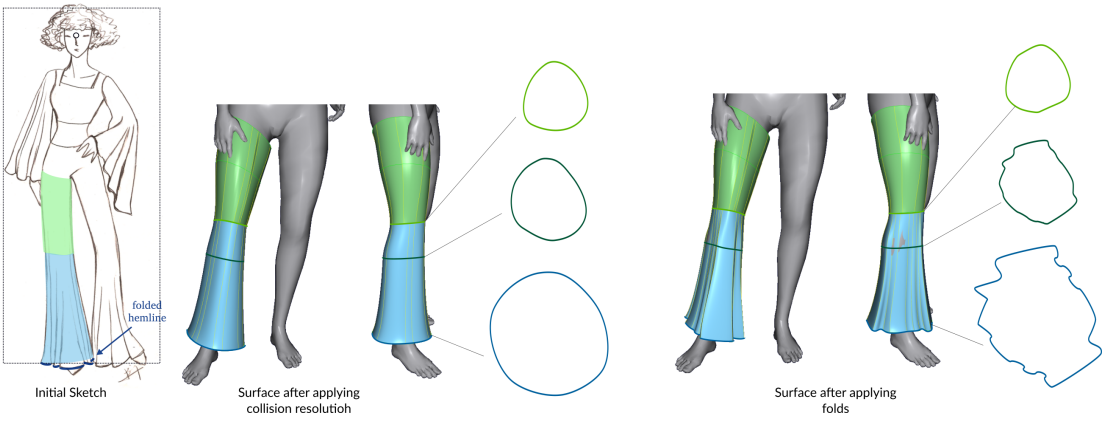


FIGURE 5.12: Example of surface folding using our method. The shape of tubular folds is extracted from the input sketch, and applied to the loose boundary of the garment surface. The folds are then propagated linearly from the loose to the tight boundary of the garment.

along the entire axis, instead of the first encountered local maximum. As for loose patches, we insert a new intermediate section curve in the plane of deepest collision. In the case of loose patches, our algorithm to deform the inserted section curve seeks to preserve at most the initial section curve's shape. However, for tight patches, we want the section to fit at best the body everywhere.

Our method to deform the inserted section curve in the case of a tight patch garment is similar to the algorithm used to compute the initial boundary curves of the surface (cf Section 5.2.2). We compute the convex hull of the polyline of intersections with the body, and recenter it around its barycenter. Instead of selecting points using the Ramer-Douglas-Peucker algorithm, we sample in this convex polyline a set of points having the same polar coordinates in this local frame than the junctions of the initial section curve in its local frame. We then approximate in a least square sense a spline having these points as junction points, and approximating the convex polyline in between. Our newly inserted section curve is now a convex closed spline fitting the character's body locally (see an example of result in Figure 5.11).

We now have a set of curved primitive patches, tight or loose, which are of the same global shape as the garment in the sketch. We have indeed transferred the garment to a target character by taking into account its different size and morphology, and possibly different pose.

5.3.3 Adding tubular folds

The final step of our method is the application of tubular folds. As discussed in Chapter 4, tubular folds are induced by the effect of gravity. They are usually visible in sketches with inner silhouettes, that are generally represented with shading effects, and

undulating loose border of the garment. We presented in the previous chapter a method to compute a shape-independent representation of tubular folds using the 2D sketched representation of this loose folded border (Section 4.2). We also presented a method to transfer this representation to an arbitrary spline curve, with two different strategy: one preserving the number of folds depicted in the input sketch, and another preserving the shape of the folds, and their frequency (Section 4.3).

In the case of loose patches, we can use this method to transfer the folds to the section curve $\mathcal{S}_R$ corresponding to the loose boundary of the patch. Each other section curve $\mathcal{S}_i,\ i \neq R$ of the surface is resampled to a unfolded section curve $\mathcal{S}_i^U$ so that junctions have the same polar coordinates than the corresponding ones of the loose border. We then compute a folded version $\mathcal{S}_i^F$ of $\mathcal{S}_i$, with a fold wrapping algorithm similar to the one presented in Section 4.2.2. The final section is computed as a linear interpolation :

$$\mathcal{S}_i = (1-\gamma_i)\mathcal{S}_i^U + \gamma_i \mathcal{S}_i^F, \text{ with } \gamma_i = \frac{\sum_{j=0}^{i-1} \|\overrightarrow{o_j o_{j+1}}\|}{\sum_{j=0}^{R-1} \|\overrightarrow{o_j o_{j+1}}\|}$$

where the o_j are the origins of the local frames associated to each section $\mathcal{S}_j$. This interpolation ensures a smooth transition between the loose boundary curve which is completely folded $\gamma_R = 1$, an the tight boundary curve which is not folded $\gamma_0 = 0$.

After the folding step, some collision with the body may appear, like in the Figure 5.11. Our collision algorithm is not well-suited to detect and solve properly these collisions, because the folded sections are no longer convex. In this case, we propose to triangulate the surface and solve collisions with an existing method for mesh collisions. In the following results, we corrected manually those collisions if they appeared.

5.4 Results and discussion

We proposed a method to synthesize virtual garments with folds on arbitrary target characters using a single annotated sketch. Our goal was to model a 3D garment whose style matches the one of the sketched 2D garment while suiting at best the target character, despite of the differences in morphology, geometry, or pose. We display in this section some garment synthesized with our method, and discuss the quality of the result, in terms of style and character fitting (Section 5.4.1). We then explain details of implementation, and computation efficiency of our algorithm (Section 5.4.2). Finally, we discuss on some failure cases and limitations of our method (Section 5.4.3).

We recall that each input is made of contour curves that are composing a list of minimal cycles. Our method synthesizes a garment for each connected set of cycles in the sketch. Therefore, one sketch may contains several garment pieces. Each garment is itself composed of connected garment patches. Each synthesized garment patch is associated to

a minimal cycle of contour curves in the 2D sketch, and identifies as tight or loose. See in Figure 5.6 the decomposition of an example of input sketch.

5.4.1 Results

We applied our style preserving garment transfer algorithm to many different stylized sketches and several 3D target characters, all displayed in Figure 5.4.

We see on Figure 5.1 different results of garment synthesis. On the left of the picture, we can see four results where the target character looks like the one in the input sketch, with similar morphology and pose. We can see that the proportions, fit, shape and folds criteria are well-transferred in these cases. On the left of the same figure, we see some results of synthesized garments where the target character differs from the one on the input sketch, because its morphology is different, such as the starfish, frog and child wearing garments drawn on adult human models, or because it is placed in a different pose, such as the child doing gymnastic moves, or superman pose.

In Figure 5.13, we display the full 3D view of some of the synthesized garments. We see how our method generates different types of garments : skirts, pants, sleeves, and collars, and different types of folds. Our garment primitive manages to create a shape that is consistently fitting in both the front and the part of the character that is hidden in the drawing. Folds are also continuously applied in a consistent way all along the boundary curve of the garment, regardless of the fact that only half of them were actually sketched in the input. This figure also shows the variety in the artistic style of the input sketches that our method can use : realistic (1), stylized (3), and even cartoon (4).

Figure 5.14 illustrates how our algorithm adapts to characters of different morphologies. Our method adapts the shape of the garment so that it adapts to human bodies with different body proportions (5a,b,c). Note for example how the height of the different patches of the pink dress adapts in function of the length of the torso and legs of the characters. Even if the target characters have different morphologies, the dress starts under the armpits and stops at the knees of each of the characters. Because our transfer is based on topological correspondances between the 2D and 3D characters, our algorithm also works with non-human target bodies, such as the starfish (d) and frog (e).

Figure 5.15 illustrates how our algorithm adapts to characters of different morphologies. We see that the style criteria which we formulated are well preserved during transfer, even if the 3D skeleton is in a different configuration than the one in the sketch. Note for example the transfer of the blue dress from a sketch on which both legs of the character are parallel to a 3D character who is standing on one leg only. As stated in Section 5.1.1, we decided to account for a shape criterion expressed in a frame that is local to the limbs of the characters. As a consequence, our method does not account for gravity, and can model garments such as the pink dress on the child doing a handstand (5f).

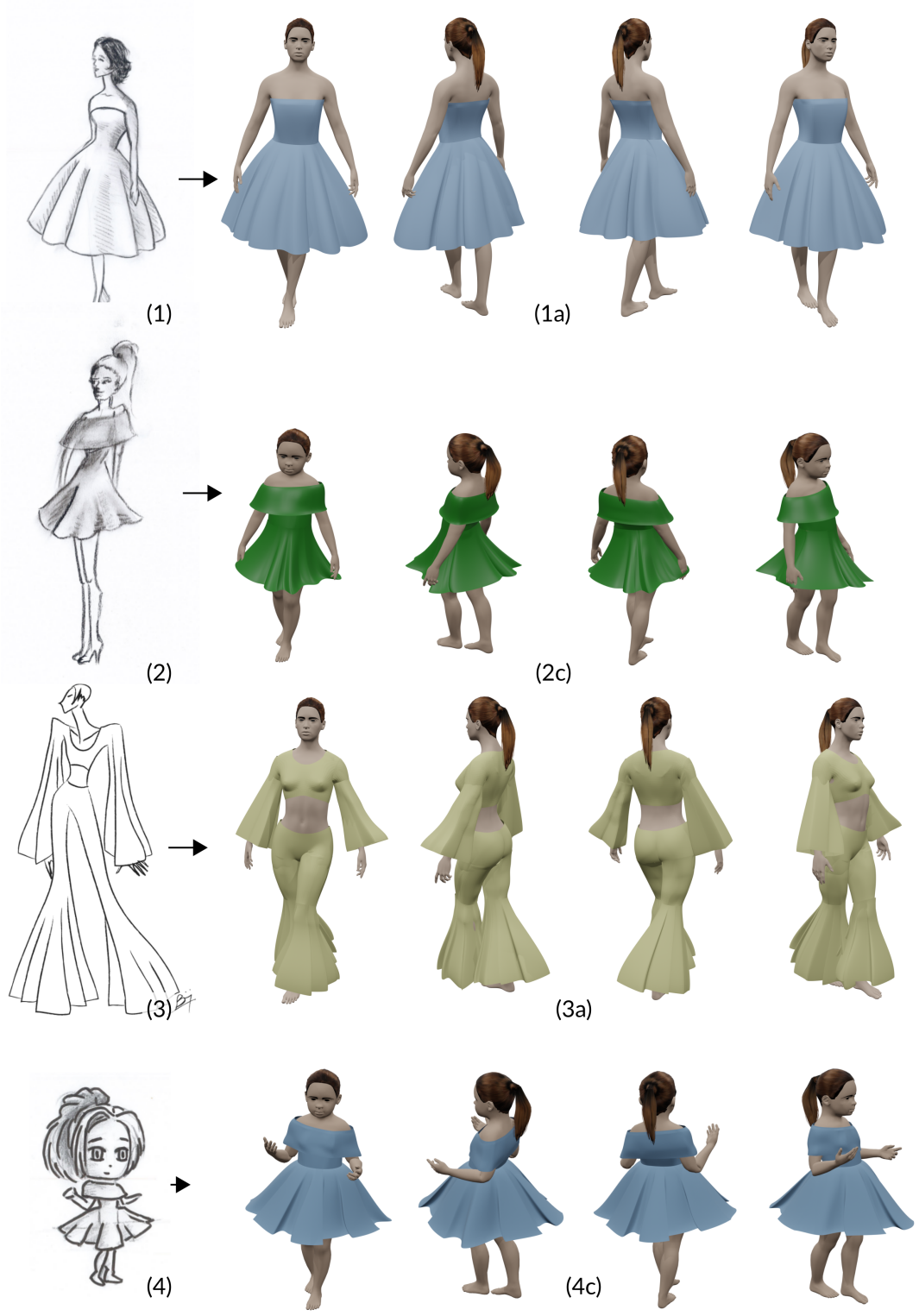


FIGURE 5.13: Example of garments generated with our method

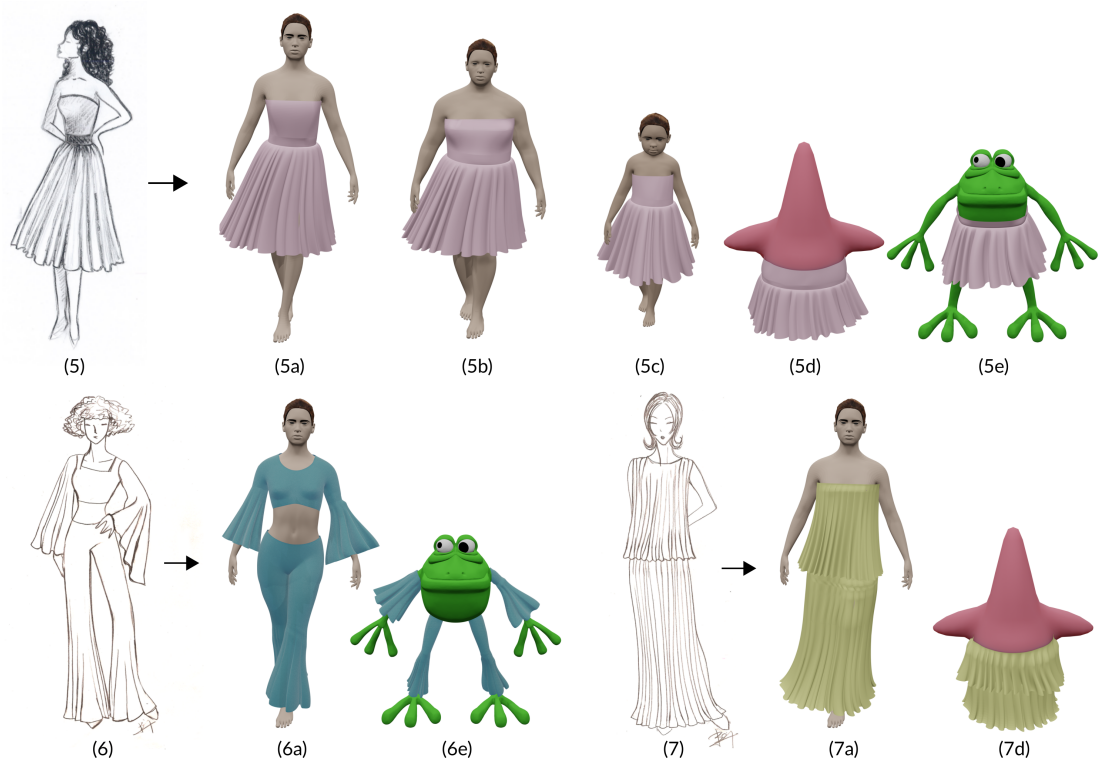


FIGURE 5.14: Adaptation of our modeling method to target characters with different morphologies and body geometries.

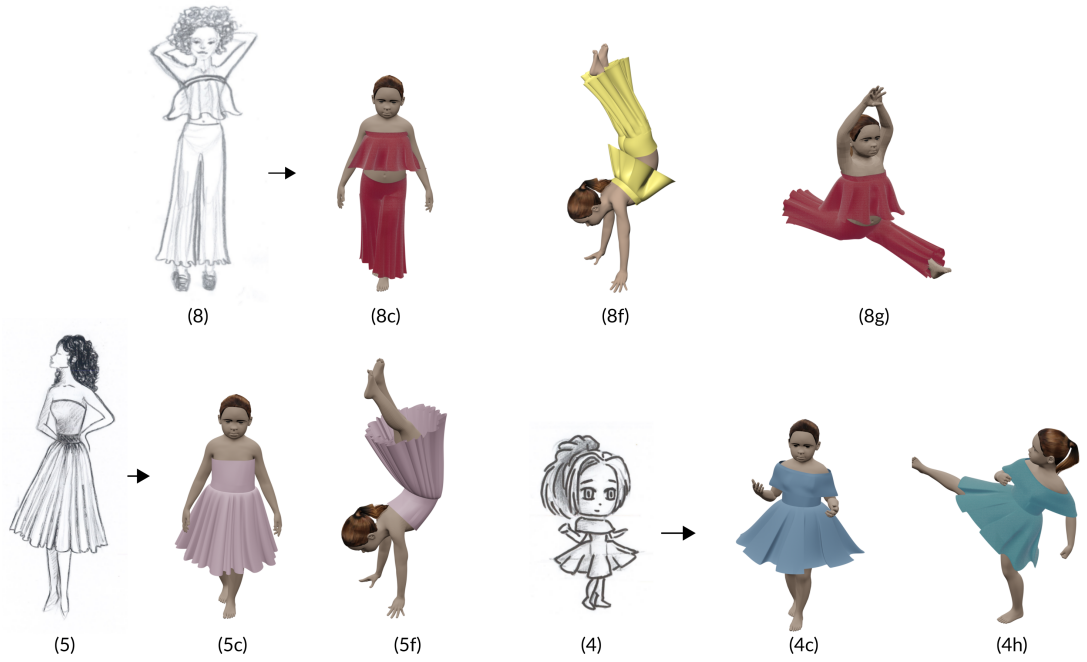


FIGURE 5.15: Adaptation of our modeling methos to target characters in different poses.

5.4.2 Technical details

We discuss here the implementation details and performance results. We implemented our algorithm in C++/OpenGL. It takes as an input an SVG file, which, in our case was created using the open-source software Inkscape. Some garment sketches (3,6,7) were made by a graphic artist, while the others (1,2,5,8) were drawn by the author, inspired by professional fashion illustrations found online. The human target characters (a,b,c) were created and rigged using the software MakeHuman, while the non humans character (d,e) were modeled and rigged from scratch by a graphic artist.

Most of the annotations we ask for the sketch can be added very quickly and intuitively: contour curves, bounding box, and location of the eye. Annotating for the 2D skeleton may be trickier, because the user needs to position each bone surrounded by the garment as a segment line. However, the time consumption of such task can be lightenned by providing a first 2D skeleton with similar topology than the one of the target character. The task of drawing bone segment becomes a task of positionning a 2D skeleton in the drawing, which is much easier.

	positionning	shapefit	collisions	folds	R	C	n
1a	0.7s	1.8s	8.4s	8.6s	2, **1**	13, **24**	50
2c (dress)	0.8s	1.9s	5.3s	5.8s	1, **1**	16, **18**	34
2c (top)	0.6s	0.8s	4.5s	N/A	**3**	**15**	45
3a (right leg)	0.5s	1.2s	4.1s	4.4s	3, **1**	10,**21**	51
3a (left leg)	0.5s	1.2s	6.7s	6.9s	5, **2**	11,**14**	83
3a (left arm)	0.5s	1s	3s	3.4s	1, **2**	8,**8**	24
3a (right arm)	0.5s	1.1s	2.4s	2.5s	2, **1**	9,**8**	26
5a	0.7s	2.4s	8.8s	9.6s	3,2,**2**	14,10,**52**	90
5e	0.8s	1.3s	2s	2.7s	2,2,**2**	12,12,**52**	76

TABLE 5.1: Cumulative computation time for some examples of garment transfer (see Figure 5.13) at each step of the algorithm (see Figure 5.3). The table also contains for each patch of the garment the number of rows R and columns C of the corresponding primitive in the final state of the algorithm. If the patch is loose, then these numbers are displayed in bold font. Finally, n is computed as the number of surface patch of the final spline surface representing the garment.

Execution time We display in Table 5.1 the computation times for some of the garments synthesized with our algorithm. The displayed times are cumulative, because each step depends on the former steps to be executed. We include the final meshing of the surface, which is done only once for each of the times presented.

We see that our algorithm synthesizes each garment in less than 10 seconds, and most of them were synthesized in less than 6 seconds. We also see that collision resolution is the most time consuming step of all, which takes more than half of the execution time

in all the examples. The algorithm for collision resolution computes many intersections between the body and intermediate planes along the primitive axis (cf Section 5.3.2). It was implemented in a naive way, and could be significantly optimized, for example by precomputing a distance map of the character's body (as done in [BSBC12]) or using acceleration methods which partition the space as pre-processing. The collision distances along the primitive axis could also be computed in parallel.

The effect of the collision resolution step on the execution time is particularly visible if we compare the pink dress transferred to the woman (5a) and the frog (5e). The dress synthesized on the frog was not subject to any collision while the one synthesized on the woman had to fit the complex geometry of her torso. In this case, we note that the difference in execution time is amplified by the fact that the woman mesh contains more faces than the frog one.

Complexity of the output models We also displayed in the Table 5.1 the model statistics, such as the number of rows R and columns C of the Bézier curves grid for each garment primitive synthesized.

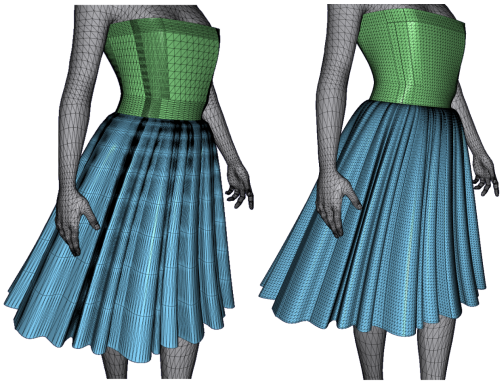
We see that the number of rows remains relatively low: the maximum is 5 rows for one patch, and most of the patches only have 1 or 2 rows. The number of rows is directly related to the collision resolution step, because it is the only step of the algorithm where section curves are inserted, and the primitive is subdivided. Tight borders are more subject to collisions with the body, and this is why they usually have more rows in their grid.

On the opposite, loose borders may contain folds, which is the main factor which increases the number of profile curves in the patch. The number of profile curves (and thus, columns in the Bézier grid) increases with the number of folds. For example, the garments (1a) and (5a) have roughly the same shape (see Figure 5.13 and 5.14). However, the dress (5a) contains significantly finer and more numerous folds. Thus, the loose patch containing the folds is composed of a significantly bigger number of columns.

We also computed the total number n of patches of bicubic surface in the garment (i.e. number of subgrids bounded by 4 Bézier curves in the control grid of the primitives), as the sum of the products RC of each primitive that the garment is composed of. This number depicts the complexity of the model, and is proportional to the total number of control points needed to represent the garment. We can see that the number of patches in each garment is situated between 24 and 90 patches. As stated before, the shape of the folds plays a significant role in the evolution of this number.

However, if the garment has fine folds, then the patches are also smaller. We computed the final garment mesh from this parametric representation with two different methods. The first one is based on the arclength of the Bézier curves that are the boundaries of the patch, for which all the edges of the mesh share the same length. The second one is

based on an uniform sampling of the Bézier curves, for which each patch is triangulated with the same number of vertices.



Uniform surface sampling Arc-length based sampling

The main advantage of the first method is that the number of faces is adapted to the complexity of the surface patches. For example, in the model of the pink dress, we see that there are much more faces in finely folded parts of the garment than in the rest. One of the drawbacks is that the parametrization is different in each patch, so the mesh is disconnected at the boundaries of the surface patch, and can not be animated. With the second method, we can have a consistent mesh for each garment. However, this method does not adapt the roughness of the mesh to the complexity of the patches.

5.4.3 Limitations

Junction areas The main limitation of our method is the fact that it can not compute garments in T-junction areas of the body, such as the chest or the pelvis. For those areas, a mesh-based representation would seem more appropriate, as used by other sketch-based modeling methods [TCH04, DJW+06, RMSC11]. These approaches generally use a 2D closed curve, drawn by the user in a plane of the 3D space and depicting the outline of the garment, to compute an initial 2D mesh, which is then lifted in 3D to match some geometric properties.

We found however two drawbacks to the use of such a method. First, we would need to compute such outline curve and place it in a plane of the 3D space. Our target character may be in different pose than the one depicted by the sketch, so finding a plane may be impossible (for example to compute the garment (5f) in Figure 5.15). Computing the outline itself might also be tricky. Because of the differences in morphology, the silhouette of the garment in tight areas may not correspond at all to the one displayed in the 2D sketch. Secondly, defining the resolution of the initial mesh before lifting it in 3D strongly constrains the type of fold we want to apply. As seen in Section 5.4.2, our parametric model allows the computation of a mesh adapted to the complexity of the shape and folds of the garment.

Another way to overcome this limitation would be to use a different type of parametric primitive, that accounts for junctions, for example T-splines [SCF+04].

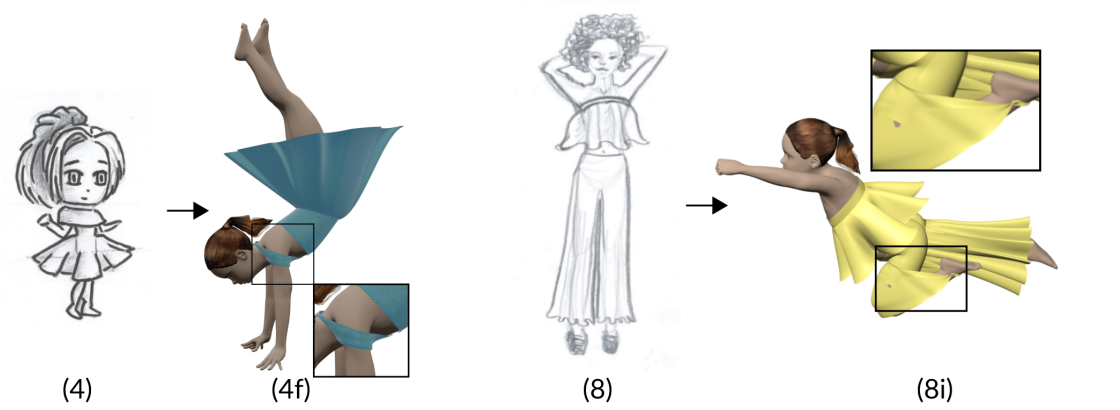


FIGURE 5.16: Some failure cases of our method.

Adaptation to extreme poses In some cases, the pose of the target character is not compatible with the position and orientation of the garment in the sketch.

We discuss in this section some limitation and failure cases of our method.

For example in the garment (4f) displayed in the Figure 5.16, the top part of the blue dress on the child doing the handstand does not give a satisfying result because the arms of the character are above her neck. The planarity imposed on the boundary curves is in conflict with this configuration, where we may expect the garment patch to cover the child's shoulder and back.

Another failure case happens, when the limbs of the characters are so much bended that the plane of the boundary curve includes vertices from another area of the limb, see for example the yellow pants on the child in superman's pose Figure 5.16(8i). In this case, the synthesized garment contains important self-intersections.

Fully tight garments Recall our algorithm to deform tight garment patches so that they fit the body of the target character: we first create the two boundary curves of the patches so that they fit the body locally (Section 5.2.2), then deform the profile curves of the patch so that they fit the tangent of the body at their extremities (Section 5.3.1), and then solve potential collisions that may appear with the body (Section 5.3.2).

In some cases, the resulting garment will not fit the target character everywhere, and ome locally loose surfaces may appear. One solution we found is to modify the sketch and subdivide the cycle to add a tight border. This way, the garment computed is made of a supplementary primitive surface for which the boundaries are tight to the body. Another solution would be to adapt our collision algorithm to also insert section curves in areas where the garment is loose. This solution would however slow down the algorithm by a lot.

FIGURE 5.17: Results of our modeling algorithm using some fashion sketches of various styles.

5.5 Conclusion

We presented a method to synthesize garment patches on an arbitrary character using a single independent sketch. Our approach transfers the style of the garment depicted by the sketch. It manages to properly extract and apply style features, even from non-realistic or imprecise sketches. Let us go back to the set of fashion sketches we showed in the beginning of this Chapter. We see in Figure 5.17 some of the results we could get using our algorithm on those sketches. Note that the folds of the black dress were transferred from the model of the sketch (5) (see Figure 5.14), because the folded hemline was not depicted well in the sketch A, and that we symmetrized the leg warmer in sketch B to obtain because the right leg was hidden in the sketch. As stated previously, some parts of the garments are not synthesized, such as the chest part of the dresses, and the cut and details of the shirt (b). However, despite the variety of style and abstraction of

the sketch, 2D curve information was enough to properly transfer the proportions and shape of most of the garments.

One of the application of our work could be to extract those style constraints and apply them on an existing surface, or to provide a first surface guess, which would be refined using inverse approaches [LCBD+18].

Chapter 6

Conclusion

This thesis presented methods to synthesize garments and sewed objects using 2D inputs. We presented two approaches for modeling 3D surfaces out of one single annotated sketch.

The first method proposed exploited for the first time the geometric properties of piecewise C^2 developable surfaces and global mirror-symmetry to reconstruct the 3D boundary curves of a sewed object represented in a photo. Our algorithm then uses these 3D curves to generate a surface mesh for each developable piece of the object, and output the corresponding 2D patterns.

The second method proposed a way to interpret a 2D fashion sketch representing a garment. In contrast to previous works, we are able to interpret stylized fashion sketches. In particular, we extracted style features from it and use those style features to synthesize garment surfaces dressing an arbitrary virtual character.

We also presented an algorithm to build a shape-independant representation for tubular folds using a single sketch of the folded boundary of a garment as input. We proposed a method to deal with discontinuous sketches of this boundary curve, and two different ways of transferring those folds on a different surface.

Our work has several limitations, which we discuss here.

Annotations All the methods are based on 2D input annotations. Most of them were realised using the 2D vector graphics software Inkscape[1]. Although we believe those annotations not to be significantly time-consuming, we may consider to automatically detect some of them using image segmentation methods. Detecting contour lines of an object seems to be a standard problem in Computer Vision, which has been improved greatly by the use of learning-based algorithms. Doing this on a fashion sketch may however be more challenging, because of the variety of

[1] https://inkscape.org/

possible shapes and ways of representing the garment. Methods to automatically position 2D skeleton using the silhouette of a character also exist. However, they usually require to know the corresponding 3D character [BVS16], or work only with photos of human bodies [CHS+18].

Developability While the issue of developability was tackled in our method to reconstruct piecewise C^2 developable surfaces, we did not account for it in our garment modeling approach. As stated in Chapter 5, since garments are not perfectly developable, it seems ill-suited to use a ruling-based approach in that method. One option would be to generate a mesh representation of our output parametric surface, and launch an optimization algorithm as the one presented by Wang et al. [WT04]. The optimization scheme could be extended in order to integrate the preservation of our style features. Another idea would be to exploit algorithms that optimize developpability in Bézier surfaces [GY06] and integrate them in the modeling pipeline, even though their non-linearity may decrease computational efficiency.

Animation Our approaches deal with the modeling of static shapes only. These garment shapes would require manual edits before being used in a cloth simulator, because they are composed of separate garment patches, and not representing the full shape of a garment. However, patches can be glued together as postprocess step. Another possible application of our algorithm would be to use it as input of an inverse modeling method such as [LCBD+18] which computes the physics parameters and patterns of a developable surface using a target 3D shape. One of the challenge of their problem is to find a suitable initial 3D surface, which our method is able to provide. The output of their algorithm is a ready-to-animate mesh, with physics parameter optimized so that the shape of the garment is stable in a simulation involving gravity.

However, this work also prospects lots of exciting new directions of research and potential future work

Extending folds variety We proposed a method to model a specific type of folds using a single sketch. Many other types of folds could be added using the same kind of methods. For example, we could be inspired by the work of Decaudin et al. [DJW+06] who proposed a parametric model for diamond folds. The parameters of this model, instead of being chosen by the user, could be guided by the sketch of those folds. Wrinkles could be also guided by independent fashion sketches. However, their representation in line drawings is usually limited to their position and direction in the garment. Finding their depth without asking the user to annotate it directly on the sketch as in [TWB+07], or without using physics as in FoldSketch [LSGV18] seems an ill-posed issue.

Sketch-based animation The analysis of garment style in sketches could also be useful to generate garment animations. If the 2D features expressed as style constraints in Chapter 5 could be extracted automatically from a photo or sketch, we can imagine to design an algorithm generating an animation of a garment from a video, or a set of keyframe sketches. The next level would be for the keyframe sketches to guide an animation while not depicting the same garment as the one we want to animate, in a similar way than the work done by Dvorovzvnak et al. [DBB+17].

Physics from sketches While studying fashion illustrations, we found out that designers put a great effort into conveying the aspect of different fabric materials. Using different shading techniques, texture and colors, we generally well perceive the material of the cloth represented in the sketch. An interesting future work would be to analyze these brightness and color information to extract the physics parameters of the fabric drawn in the sketch, to guide the modeling and potential animation of the garment.

Bibliography

[AKA+17] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George W Fitzmaurice. Experimental evaluation of sketching on surfaces in vr. In *CHI*, volume 17, pages 5643–5654, 2017.

[Aum03] Günter Aumann. A simple algorithm for designing developable bézier surfaces. *Computer Aided Geometric Design*, 20(8-9):601–619, 2003.

[BBS08] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 151–160. ACM, 2008.

[BCV+15] Mikhail Bessmeltsev, Will Chang, Nicholas Vining, Alla Sheffer, and Karan Singh. Modeling character canvases from cartoon drawings. *ACM Transactions on Graphics (TOG)*, 34(5):162, 2015.

[BGK+13] Floraine Berthouzoz, Akash Garg, Danny M Kaufman, Eitan Grinspun, and Maneesh Agrawala. Parsing sewing patterns into 3d garments. *Acm Transactions on Graphics (TOG)*, 32(4):85, 2013.

[BHS+17] Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. Learning detail transfer based on geometric features. In *Computer Graphics Forum*, volume 36, pages 361–373. Wiley Online Library, 2017.

[BK04] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, August 2004.

[BPCB08] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loïc Barthe. Matisse: Painting 2d regions for modeling free-form shapes. 2008.

[BSBC12] Remi Brouet, Alla Sheffer, Laurence Boissieux, and Marie-Paule Cani. Design preserving garment transfer. *ACM Transactions on Graphics*, 31(4):Article–No, 2012.

[BSK+16] Aric Bartle, Alla Sheffer, Vladimir G Kim, Danny M Kaufman, Nicholas Vining, and Floraine Berthouzoz. Physics-driven pattern adjustment for direct 3d garment editing. *ACM Trans. Graph.*, 35(4):50–1, 2016.

[BT81] H. Barrow and J. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17, 1981.

[BVS16] Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. Gesture3d: posing 3d characters via gesture drawings. *ACM Transactions on Graphics (TOG)*, 35(6):165, 2016.

[BWK03] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[Car76] M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[CB92] Roberto Cipolla and Andrew Blake. Surface shape from the deformation of apparent contours. *International journal of computer vision*, 9(2):83–112, 1992.

[CC04] Chih-Hsing Chu and Jang-Ting Chen. Geometric design of developable composite bézier surfaces. *Computer-Aided Design and Applications*, 1(1-4):531–539, 2004.

[CCM14] Nicolas Cherin, Frederic Cordier, and Mahmoud Melkemi. Modeling piecewise helix curves from 2d sketches. *Computer-Aided Design*, 46:258–262, 2014.

[Cha79] Indranil Chakravarty. A generalized line and junction labeling scheme with application to scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):202–205, 1979.

[CHS+18] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[CS02] Chih-Hsing Chu and Carlo H Séquin. Developable bézier patches: properties and design. *Computer-Aided Design*, 34(7):511–527, 2002.

[CSMS13] Frederic Cordier, Hyewon Seo, Mahmoud Melkemi, and Nickolas S Sapidis. Inferring mirror symmetric 3d shapes from sketches. *Computer-Aided Design*, 45(2):301–311, 2013.

[CSMT03] Frederic Cordier, Hyewon Seo, and Nadia Magnenat-Thalmann. Made-to-measure technologies for an online clothing store. *IEEE Computer graphics and applications*, 23(1):38–48, 2003.

[CSPN11] Frederic Cordier, Hyewon Seo, Jinho Park, and Jun Yong Noh. Sketching of mirror-symmetric shapes. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1650–1662, 2011.

[CZS+13] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6):195, 2013.

[DAI+18] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei Efros, and Adrien Bousseau. 3D sketching using multi-view deep volumetric prediction. *Proc. ACM on Computer Graphics and Interactive Techniques*, 1(1), 2018.

[DBB+17] Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sỳkora. Example-based expressive animation of 2d rigid bodies. *ACM Transactions on Graphics (TOG)*, 36(4):127, 2017.

[DDÖ+17] R Daněřek, Endri Dibra, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment: 3d garment shape estimation from a single image. In *Computer Graphics Forum*, volume 36, pages 269–280. Wiley Online Library, 2017.

[DJW+06] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum*, 25(3):625–634, 2006.

[DP73] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[DPS15] Chris De Paoli and Karan Singh. Secondskin: sketch-based construction of layered 3d models. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH'96*, pages 11–20, 1996.

[EBC+15] Even Entem, Loïc Barthe, Marie-Paule Cani, Frederic Cordier, and Michiel Van De Panne. Modeling 3d animals from a side-view sketch. *Computers & Graphics*, 46:221–230, 2015.

[EVC+15] Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. Worldbrush: Interactive example-based synthesis of procedural virtual worlds. *ACM Transactions on Graphics (TOG)*, 34(4):106, 2015.

[FJL+16] Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sỳkora. Stylit: illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics (TOG)*, 35(4):92, 2016.

[FLB16] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Fidelity vs. simplicity: A global approach to line drawing vectorization. *ACM Trans. Graph.*, 35(4), 2016.

[FMW02] Alexandre RJ François, Gérard G Medioni, and Roman Waupotitsch. Reconstructing mirror symmetric scenes from a single view using 2-view stereo geometry. In *Object recognition supported by user interaction for service robots*, volume 4, pages 12–16. IEEE, 2002.

[Fre02] William H Frey. Boundary triangulations approximating developable surfaces that interpolate a closed space curve. *International Journal of Foundations of Computer Science*, 13(02):285–302, 2002.

[FWTQ07] Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. Sketching hairstyles. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 31–36. ACM, 2007.

[GIZ09] Yotam Gingold, Takeo Igarashi, and Denis Zorin. Structured annotations for 2d-to-3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, page 148. ACM, 2009.

[GRH+12] Peng Guan, Loretta Reiss, David Hirshberg, Alexander Weiss, and Michael Black. Drape: Dressing any person. *ACM Trans. Graph.*, 31(4), 2012.

[GY06] Mo Guoliang and Zhao Yanan. Designing bézier surfaces minimizing the gaussian curvature. In *Proceedings of the International Conference on Robotics, Control and Manufacturing Technology*, pages 271–276, 2006.

[HJO+01] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[HKYM16] Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE transactions on visualization and computer graphics*, 23(8):2003–2013, 2016.

[HLT+09] Thomas Hurtut, P-E Landes, Joëlle Thollot, Yann Gousseau, Remy Drouillhet, and J-F Coeurjolly. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*, pages 51–60. ACM, 2009.

[HOCS02] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M Seitz. Curve analogies. In *Rendering Techniques*, pages 233–246, 2002.

[IBB15] Emmanuel Iarussi, David Bommes, and Adrien Bousseau. Bendfields: Regularized curvature fields from rough concept sketches. *ACM Transactions on Graphics (TOG)*, 34(3):24, 2015.

[IIMT07] Takeo Igarashi, Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Acm siggraph 2007 courses*, page 21. ACM, 2007.

[JC08] Pushkar Joshi and Nathan A Carr. Repoussé: Automatic inflation of 2d artwork. In *SBM*, pages 49–55. Citeseer, 2008.

[JHK15] Moon-Hwan Jeong, Dong-Hoon Han, and Hyeong-Seok Ko. Garment capture from a photograph. *Computer Animation and Virtual Worlds*, 26(3-4):291–300, 2015.

[JHR+15] Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. Sketching folds: Developable surfaces from non-planar silhouettes. *Acm Transactions on Graphics (TOG)*, 34(5):155, 2015.

[JTC09] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong. Symmetric architecture modeling with a single image. In *ACM Transactions on Graphics (TOG)*, volume 28, page 113. ACM, 2009.

[JYSL19] Liguo Jiang, Juntao Ye, Liming Sun, and Jituo Li. Transferring and fitting fixed-sized garments onto bodies of various dimensions and postures. *Computer-Aided Design*, 106:30 – 42, 2019.

[KFC+08] Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J Mitra, Alla Sheffer, and Helmut Pottmann. Curved folding. In *ACM Transactions on Graphics (TOG)*, volume 27, page 75. ACM, 2008.

[Koe84] Jan J Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984.

[KP01] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. SIAM, 2001.

[KVD82] Jan J Koenderink and Andrea J Van Doorn. The shape of smooth objects and the way contours end. *Perception*, 11(2):129–137, 1982.

[KYZ13] Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. A survey of 2d and 3d shape descriptors. In *2013 10th International Conference Computer Graphics, Imaging and Visualization*, pages 1–10. IEEE, 2013.

[LA15] Katrin Lang and Marc Alexa. The markov pen: online synthesis of free-hand drawing styles. In *Proceedings of the workshop on Non-Photorealistic Animation and Rendering*, pages 203–215. Eurographics Association, 2015.

[LCBD+18] Mickaël Ly, Romain Casati, Florence Bertails-Descoubes, Mélina Skouras, and Laurence Boissieux. Inverse elastic shell design with contact and friction. *ACM Trans. Graph.*, 37(6), 2018.

[LHLF15] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3d furniture models. *ACM Transactions on Graphics (TOG)*, 34(4):85, 2015.

[LKWS16] Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. Functionality preserving shape style transfer. *ACM Transactions on Graphics (TOG)*, 35(6):209, 2016.

[LPL+17] Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. Bendsketch: Modeling freeform surfaces through 2d sketching. *ACM Transactions on Graphics (TOG)*, 36(4):125, 2017.

[LPW+06] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 681–689. ACM, 2006.

[LS96] H Lipson and M Shpitalni. Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design*, 1996.

[LSGV18] Minchen Li, Alla Sheffer, Eitan Grinspun, and Nicholas Vining. Foldsketch: enriching garments with physically reproducible folds. *ACM Transactions on Graphics (TOG)*, 37(4):133, 2018.

[LSMI10] Manfred Lau, Greg Saul, Jun Mitani, and Takeo Igarashi. Modeling-in-context: user design of complementary objects with a single photo. In *Proc. Sketch-Based Interfaces and Modeling*, 2010.

[MI07] Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. *ACM Transactions on Graphics (TOG)*, 26(3):45, 2007.

[MMY01] Carolyn L Moore, Kathy K Mullet, and Margaret Prevatt Young. *Concepts of pattern grading: techniques for manual and computer grading*. Fairchild Books, 2001.

[MS11] James McCrae and Karan Singh. Neatening sketched strokes using piecewise french curves. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 141–148. ACM, 2011.

[MWJ12] Yuwei Meng, Charlie CL Wang, and Xiaogang Jin. Flexible shape control for automatic resizing of apparel products. *Computer-Aided Design*, 44(1):68–76, 2012.

[NG09] C.A. Nunnelly and M.C. Guyon. *Techniques d'illustration de mode*. Atout carré. Eyrolles, 2009.

[NGDA+16] Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, 35(4):130, 2016.

[NISA07] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. In *ACM transactions on graphics (TOG)*, volume 26, page 41. ACM, 2007.

[NPO13] Rahul Narain, Tobias Pfaff, and James F O'Brien. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG)*, 32(4):51, 2013.

[OI03] Makoto Okabe and Takeo Igarashi. 3d modeling of trees from freehand sketches. In *ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1. ACM, 2003.

[OSSJ09] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *CAG*, 33, 2009.

[ÖUP+11] A Cengiz Öztireli, Umut Uyumaz, Tiberiu Popa, Alla Sheffer, and Markus Gross. 3d modeling with a symmetric sketch. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 23–30. ACM, 2011.

[PS07] Francisco Pérez and José Antonio Suárez. Quasi-developable b-spline surfaces in ship hull design. *Computer-Aided Design*, 39(10):853–862, 2007.

[RCHT11] Damien Rohmer, Marie-Paule Cani, Stefanie Hahmann, and Boris Thibert. Folded paper geometry from 2d pattern and 3d contour. In *Eurographics 2011 (short paper)*, pages 21–24, 2011.

[RDI10] Alec Rivers, Frédo Durand, and Takeo Igarashi. *3D modeling with silhouettes*, volume 29. Acm, 2010.

[RHSH18] Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. Discrete geodesic nets for modeling developable surfaces. *ACM Transactions on Graphics (ToG)*, 37(2):16, 2018.

[RMSC11] Cody Robson, Ron Maharik, Alla Sheffer, and Nathan Carr. Context-aware garment modeling from sketches. *Computers & Graphics*, 35(3):604–613, 2011.

[RSW+07] Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. Developable surfaces from arbitrary sketched boundaries. In *SGP'07-5th Eurographics Symposium on Geometry Processing*, pages 163–172. Eurographics Association, 2007.

[SAG+13] Alex Shtof, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. Geosemantic snapping for sketch-based modeling. In *Computer graphics forum*, volume 32, pages 245–253. Wiley Online Library, 2013.

[SBS19] Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. Deep sketch-based modeling of man-made shapes. *ArXiv*, abs/1906.12337, 2019.

[SBSS12] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. Crossshade: shading concept sketches using cross-section curves. 2012.

[SCF+04] Thomas W Sederberg, David L Cardon, G Thomas Finnigan, Nicholas S North, Jianmin Zheng, and Tom Lyche. T-spline simplification and local refinement. In *ACM transactions on graphics (TOG)*, volume 23, pages 276–283. ACM, 2004.

[SGC18] Oded Stein, Eitan Grinspun, and Keenan Crane. Developability of triangle meshes. *ACM Transactions on Graphics (TOG)*, 37(4):77, 2018.

[SHBS16] Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Saguin-Sprynski. Surfacing curve networks with normal control. *Computers & Graphics*, 60:1 – 8, 2016.

[Sin99] Karan Singh. Interactive curve design using digital french curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 23–30. ACM, 1999.

[SKSK09] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. Analytic drawing of 3d scaffolds. In *ACM Transactions on Graphics (TOG)*, volume 28, page 149. ACM, 2009.

[SLMB05] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)*, 24(2):311–330, 2005.

[SP04] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[SRH+15] Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie CL Wang, and Jean-Francis Bloch. Nonsmooth developable geometry for interactively animating paper crumpling. *ACM Transactions on Graphics (TOG)*, 35(1):10, 2015.

[SSP+14] Masahiro Sekine, Kaoru Sugita, Frank Perbet, Björn Stenger, and Masashi Nishiyama. Virtual fitting by single-shot body shape estimation. In *Int. Conf. on 3D Body Scanning Technologies*, pages 406–413. Citeseer, 2014.

[SVWG12] Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Flexible developable surfaces. In *Computer Graphics Forum*, volume 31, pages 1567–1576. Wiley Online Library, 2012.

[SWSJ06] Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses*, page 14. ACM, 2006.

[TCH04] Emmanuel Turquin, Marie-Paule Cani, and John Hughes. Sketching garments for virtual characters. In John F. Hughes and Joaquim A. Jorge, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Grenoble, France, 2004. Eurographics.

[TML09] Chao Tian, Mark Masry, and Hod Lipson. Physical sketching: Reconstruction and analysis of 3D objects from freehand sketches. *Computer Aided Design*, 41(3):147–158, 2009.

[TNT89] Toshimitsu Tanaka, Seiichiro Naito, and Tokiichiro Takahashi. Generalized symmetry and its application to 3d shape generation. *The Visual Computer*, 5(1-2):83–94, 1989.

[TWB+07] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F Hughes. A sketch-based interface for clothing virtual characters. *IEEE Computer graphics and applications*, 27(1), 2007.

[TZF04] Chiew-Lan Tai, Hongxin Zhang, and Jacky Chun-Kin Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. In *Computer Graphics Forum*, volume 23, pages 71–83. Wiley Online Library, 2004.

[UKIG11] Nobuyuki Umetani, Danny M Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 90. ACM, 2011.

[UN93] Fatih Ulupinar and Ramakant Nevatia. Perception of 3-d surfaces from 2-d contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):3–18, 1993.

[VCMT05] Pascal Volino, Frederic Cordier, and Nadia Magnenat-Thalmann. From early virtual garment simulation to interactive fashion design. *Computer-aided design*, 37(6):593–608, 2005.

[Wat09] Naoki Watanabe. *Contemporary fashion illustration techniques*. Rockport Publishers, 2009.

[WBC07] Jamie Wither, Florence Bertails, and Marie-Paule Cani. Realistic hair from a sketch. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*, pages 33–42. IEEE, 2007.

[WBCG09] Jamie Wither, Frédéric Boudon, M-P Cani, and Christophe Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. In *Computer Graphics Forum*, volume 28, pages 541–550. Wiley Online Library, 2009.

[WCPM18] Tuanfeng Y Wang, Duygu Cylan, Jovan Popovic, and Niloy J Mitra. Learning a shared shape space for multimodal garment design. *arXiv preprint arXiv:1806.11335*, 2018.

[WT04] Charlie CL Wang and Kai Tang. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer*, 20(8-9):521–539, 2004.

[WWY05] Charlie CL Wang, Yu Wang, and Matthew MF Yuen. Design automation for customized apparel products. *Computer-aided design*, 37(7):675–691, 2005.

[XCF+13] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. Sketch2scene: sketch-based co-retrieval and co-placement of 3d models. *ACM Transactions on Graphics (TOG)*, 32(4):123, 2013.

[XCS+14] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. True2form: 3d curve networks from 2d sketches via selective regularization. 2014.

[XXM+13] Xiaohua Xie, Kai Xu, Niloy J Mitra, Daniel Cohen-Or, Wenyong Gong, Qi Su, and Baoquan Chen. Sketch-to-design: Context-based part assembly. In *Computer Graphics Forum*, volume 32, pages 233–245. Wiley Online Library, 2013.

[YAP+16] Shan Yang, Tanya Ambert, Zherong Pan, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Detailed garment recovery from a single-view image. *arXiv preprint arXiv:1608.01250*, 2016.

[YLT13] Linjie Yang, Jianzhuang Liu, and Xiaoou Tang. Complex 3d general object reconstruction from line drawings. In *IEEE International Conference on Computer Vision*, 2013.

[ZCF+13] Bin Zhou, Xiaowu Chen, Qiang Fu, Kan Guo, and Ping Tan. Garment modeling from a single image. In *Computer graphics forum*, volume 32, pages 85–91. Wiley Online Library, 2013.

[ZDPSS02] Li Zhang, Guillaume Dugas-Phocion, Jean-Sebastien Samson, and Steven M Seitz. Single-view modelling of free-form scenes. *The Journal of Visualization and Computer Animation*, 13(4):225–235, 2002.