



**HAL**  
open science

# Multi-touch gesture interactions and deformable geometry for 3D edition on touch screen

Remi Brouet

► **To cite this version:**

Remi Brouet. Multi-touch gesture interactions and deformable geometry for 3D edition on touch screen. Graphics [cs.GR]. Université Grenoble Alpes, 2015. English. NNT : 2015GREAM073 . tel-01677433

**HAL Id: tel-01677433**

**<https://theses.hal.science/tel-01677433>**

Submitted on 8 Jan 2018

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

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

## THÈSE

Pour obtenir le grade de

### DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques-Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

**Rémi Brouet**

Thèse dirigée par **Renaud Blanch**  
et codirigée par **Marie-Paule Cani**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG)**  
du **Laboratoire Jean Kuntzmann (LJK)**  
et de l'école doctorale **EDMSTII**

# Interactions gestuelles multi-point et géométrie déformables pour l'édition 3D sur écran tactile

Thèse soutenue publiquement le **XXXX 2014**,  
devant le jury composé de :

**George-Pierre Bonneau**

Professeur, Grenoble Université, Président

**Joaquim Jorge**

Professeur, Université de Lisbonne, Rapporteur

**Laurent Grisoni**

Professeur, Université Lille 1, Rapporteur

**Marchin Hachet**

Professeur à l'Inria Bordeaux, Examineur

**Renaud Blanc**

Maitre de Conférence, Grenoble Université, Directeur de thèse

**Marie-Paule Cani**

Professeur, Grenoble Université, Directeur de thèse





# RÉSUMÉ

## **Interactions gestuelles multi-point et géométrie déformables pour l'édition 3D sur écran tactile**

Malgré les progrès en capture d'objets réels et en génération procédurale, la création de contenus pour les mondes virtuels ne peut se faire sans interaction humaine. Cette thèse propose d'exploiter les nouvelles technologies tactiles (écrans "multi-touch") pour offrir une interaction 2D simple et intuitive afin de naviguer dans un environnement virtuel, et d'y manipuler, positionner et déformer des objets 3D.

En premier lieu, nous étudions les possibilités et les limitations gestuelles de la main et des doigts lors d'une interaction sur écran tactile afin de découvrir quels gestes semblent les plus adaptés à l'édition des environnements et des objets 3D. En particulier, nous évaluons le nombre de degrés de liberté efficaces d'une main humaine lorsque son geste est contraint à une surface plane. Nous proposons également une nouvelle méthode d'analyse gestuelle par phases permettant d'identifier en temps réel les mouvements clés de la main et des doigts. Ces résultats, combinés à plusieurs études utilisateur spécifiques, débouchent sur l'identification d'un patron pour les interactions gestuelles de base incluant non seulement navigation (placement de caméra), mais aussi placement, rotation et mise à l'échelle des objets. Ce patron est étendu dans un second temps aux déformations complexes (ajout et suppression de matière ainsi que courbure ou torsion des objets, avec contrôle de la localité). Tout ceci nous permet de proposer et d'évaluer une interface d'édition des mondes 3D permettant une interaction tactile naturelle, pour laquelle le choix du mode (navigation, positionnement ou déformation) et des tâches correspondantes est automatiquement géré par le système en fonction du geste et de son contexte (sans menu ni boutons). Enfin, nous étendons cette interface pour y intégrer des déformations plus complexes à travers le transfert de vêtements d'un personnage à un autre, qui est étendu pour permettre la déformation interactive du vêtement lorsque le personnage qui le porte est déformé par interaction tactile.



# ABSTRACT

## **Multi-touch gesture interactions and deformable geometry for 3D edition on touch screen**

Despite the advances made in the fields of existing objects capture and of procedural generation, creation of content for virtual worlds can not be perform without human interaction. This thesis suggests to exploit new touch devices ("multi-touch" screens) to obtain an easy, intuitive 2D interaction in order to navigate inside a virtual environment, to manipulate, position and deform 3D objects.

First, we study the possibilities and limitations of the hand and finger gestures while interacting on a touch screen in order to discover which gestures are the most adapted to edit 3D scene and environment. In particular, we evaluate the effective number of degrees of freedom of the human hand when constrained on a planar surface. Meanwhile, we develop a new gesture analysis method using phases to identify key motion of the hand and fingers in real time. These results, combined to several specific user-studies, lead to a gestural design pattern which handle not only navigation (camera positioning), but also object positioning, rotation and global scaling. Then, this pattern is extended to complex deformation (such as adding and deleting material, bending or twisting part of objects, using local control). Using these results, we are able to propose and evaluate a 3D world editing interface that handle a natural touch interaction, in which mode selection (i.e. navigation, object positioning or object deformation) and task selections is automatically processed by the system, relying on the gesture and the interaction context (without any menu or button). Finally, we extend this interface to integrate more complex deformations, adapting the garment transfer from a character to any other in order to process interactive deformation of the garment while the wearing character is deformed.



# REMERCIEMENTS

TODO





# CONTENTS

<b>Contents</b>	<b>9</b>
<b>1 Introduction</b>	<b>15</b>
<b>2 State Of The Art</b>	<b>19</b>
2.1 Shape Representations . . . . .	20
2.1.1 Solid Primitives . . . . .	20
2.1.2 Meshes . . . . .	20
2.1.3 Implicit Surfaces . . . . .	21
2.2 Shape Edition . . . . .	21
2.2.1 Model-Based Methods . . . . .	22
2.2.2 Space Deformations . . . . .	22
2.2.3 Surface-Based Deformations . . . . .	23
2.2.4 Sketch-based modeling . . . . .	23
2.3 Rise of Multi-Touch Devices . . . . .	24
2.3.1 Devices Combination . . . . .	25
2.4 Modal Interactions . . . . .	26
2.4.1 Standard Menus . . . . .	26
2.4.2 Specific Menus for Multi-Touch Screen . . . . .	27
2.4.3 Contextual Interactions . . . . .	28
2.4.4 Gestural-Based Interaction . . . . .	29
2.4.5 Discussion . . . . .	29
2.5 Gestural-Based Interactions . . . . .	29
2.5.1 Extension of the standard 2D method: Rotate-Scale-Translate (RST) . . . . .	30
2.5.2 Other Multi-Touch Gestural-Based Interactions . . . . .	31
2.5.3 On-Air Gestural-Based Interactions . . . . .	32
2.5.4 Discussion . . . . .	32
2.6 Hand Behavior on Touch Screen and Its Limitation . . . . .	33
2.6.1 Finger Dependencies . . . . .	33

2.6.2	Hand Analysis	34
2.6.3	Bimanual Interactions	35
<b>3</b>	<b>Understanding Hand Degrees of Freedom on a Surface</b>	<b>37</b>
3.1	Introduction	38
3.2	Hand Motion Analysis: Phase-based Registration	38
3.2.1	Specific Hand Parameterization	38
3.2.2	Phase Registration	40
3.3	User-Study	40
3.3.1	Experiment	41
3.3.2	Apparatus	41
3.3.3	Participants	41
3.4	Results: Global Hand Motion Analysis	41
3.5	Results: Finger Motion Analysis and Dependencies	43
3.6	Conclusion	44
<b>4</b>	<b>Understanding Hand Gestures on a Surface</b>	<b>45</b>
4.1	Introduction	46
4.2	User-Study	46
4.2.1	Experiment	46
4.2.2	Apparatus	47
4.2.3	Participants	47
4.3	Interpreting Stored Gestures	47
4.3.1	Hand Phase Analysis	47
4.4	Results of the User-Study and Discussion	48
4.4.1	Hands/Fingers uses	48
4.4.2	Scaling Interferences	49
4.4.3	Modes Disambiguation	49
4.4.4	Group Selection	50
4.4.5	Navigation: Zoom Task vs. Depth Translation Task	50
4.4.6	Combining Interactions	50
4.4.7	Phase Analysis and Noticeable Gestural Design Pattern	51
4.5	Conclusion	52
<b>5</b>	<b>Understanding Deformation on a Surface</b>	<b>55</b>
5.1	Introduction	56
5.2	Interaction Principles for Deformation Tasks	56
5.2.1	Delimitating Region of Interest	56
5.2.2	Disambiguating Planar/Non-Planar Deformation Tasks	57
5.3	User-Study	57
5.3.1	Experiment 1	57
5.3.2	Experiment 2	58
5.3.3	Same Studied Set	58
5.3.4	Apparatus	58
5.3.5	Participants	58
5.4	Results	59
5.4.1	Use of Support Finger (P1)	59
5.4.2	Placement of Fingers (P2)	60

---

5.4.3	Distinguishing Planar/Non-Planar Tasks (P3 and P4)	61
5.4.4	Additional Results	62
5.5	Discussion	62
<b>6</b>	<b>Free-Form Editing of 3D Shapes</b>	<b>65</b>
6.1	Introduction	66
6.2	On-Line Phase-Based Analysis	66
6.2.1	Thumb Registration	67
6.2.2	Hand Registration	67
6.3	Mode/Task Selection	67
6.3.1	Mode Selection: Contextual Selection	68
6.3.2	Task Selection: Global Phase Analysis	68
6.4	Results and Discussion	69
6.5	Conclusion	71
<b>7</b>	<b>From Gestures to Skeleton-Based Implicit Surface Deformation</b>	<b>73</b>
7.1	Introduction	74
7.2	Scene and Objects Details	74
7.2.1	Scene Contents and Hierarchy	74
7.2.2	Structural Skeleton	75
7.3	Region of Interest Selection	77
7.3.1	Object Selection and Center of Transformation	77
7.3.2	Vertex Selection on Skeleton	78
7.4	Task Details	79
7.4.1	Gestural and Visual Consistency	79
7.4.2	Gestural and Visual Regularities	79
7.4.3	Deformation Details	80
7.5	Conclusion	82
<b>8</b>	<b>Design Preserving Garment Transfer</b>	<b>85</b>
8.1	Introduction	86
8.2	Grading Criteria	86
8.2.1	Proportion and Scale	87
8.2.2	Shape	87
8.2.3	Fit	87
8.2.4	Balance	88
8.2.5	Manufacturability	88
8.2.6	Collision Avoidance	88
8.3	Method Overview	89
8.4	Proportional Scaling	89
8.4.1	Selecting Reference Points	90
8.4.2	Proportional Scaling Using Offset Vectors	91
8.5	Shape Preserving Garment Transfer	91
8.5.1	As-2D-as-possible Deformation	91
8.5.2	Accounting for Relative Location and Fits Constraints	92
8.5.3	Handling Collisions	93
8.5.4	Extension to Layers of Garments	95
8.5.5	Pattern Extraction	95

8.6	Results and Discussion . . . . .	96
8.6.1	Validation . . . . .	97
8.6.2	Comparison . . . . .	97
8.6.3	Statistics . . . . .	98
8.6.4	Limitations . . . . .	99
8.7	Conclusion . . . . .	99
<b>9</b>	<b>Towards Interactive Garment Deformation</b>	<b>101</b>
9.1	Introduction . . . . .	101
9.2	Symmetric Skeleton . . . . .	101
9.3	Adaptation of the Garment Transfer Method . . . . .	103
9.4	First Results . . . . .	103
9.5	Discussion and Conclusion . . . . .	104
<b>10</b>	<b>Conclusion</b>	<b>107</b>
	<b>Bibliography</b>	<b>109</b>





## CHAPTER

# 1

# INTRODUCTION

From everyday life to professional business, and from digital entertainment to numerical simulation, the number of fields that require the production of virtual 3D content is widely increasing. For instance, more and more movies require 3D animation creation, some museum currently present a virtual visit and life simulation in augmented reality, professional designers show of their new project using a virtual model of their prototype. In every of these examples, 3D modeling skills are required.

Yet, the edition of 3D content is still a complex task. Even the case of a simple object 3D manipulation (such as looking around an object) or edition (such as scaling it) requires learning to use professional software, usually with ca consistent documentation, an overwhelming number of commands gathered behind menus or shortcut, and asking for parameters that require some mathematical background. Therefore, only professional designers or long-time passionate practitioners succeed in creating virtual 3D worlds.

This observation is even more true for complex tasks. Even though a large number of mathematical tools were developed to edit virtual content (including tools that may not be possible in the physical world, such as extrusion), these tools are often invented, developed and used only by few people. However, as developing efficient and quick-learned interaction techniques is usually not a prerogative, newcomers lack skills to efficiently use these tools.

The goal of this thesis is to propose a simplification of the 3D modeling pipeline enabling beginners to handle it. To reach this goal, we rely on two main criteria:

- User interactions should be as close as possible to the "natural" interaction (i.e., learning interaction set should be as fast as possible).
- 3D manipulation and edition tools should not require a specific mathematical background from users to be used efficiently.

Although our first criterion depends on "natural" interaction, there is no consensual definition of "natural" in the HCI field [WW11]. Indeed, many tasks can be performed with the same gesture (for instance, translating or scrolling an image might be mapped to one finger



translation gesture). In this case, which task is the most appropriate for this specific gesture? Conversely, one task might be handled by several gestures (for instance, zoom in/out might rely on one hand two fingers pinch gesture or two hands one finger pinch gesture). In that case, which gesture is more "natural" to perform this specific task? For these reasons, we are more concerned about gestures that are quickly learned and that are easily reproducible by the average user. For instance, the pinch gesture on touch screen of a mobile phone seems "natural", in the sense that nowadays, many users use it to zoom in/out an image. Though, the learning of this gesture was a cultural learning since the rise of touch devices.

To not overload the user comprehension of the interface, the second criterion deals with the mathematical issue. In standard software, designers consume a lot of time to edit the parameters of complex tools, which required a specific mathematical background to understand tools process. Therefore, graphical content should be high level graphical model that provide the right handle to convey user intents, while their mathematical complexity should be kept transparent to the user.

This leads to research at the interface between the Human-Computer Interaction (HCI) field and the Computer Graphic (CG) field in order to obtain a complete 3D creation interface.

## Multi-Touch Devices

Even if the first multi-touch device was created in 1982 [Meh82], these devices caught up only recently, with the development of tactile mobile phones and tablets. Simultaneously, the rise of these technologies induced many researches which already impacted the manipulation of both 2D and 3D virtual content.

We choose to consider the use of multi-touch devices in this work, because they bring several desirable properties, relating them to the way shapes are created or manipulated in the physical world: in traditional shape design, 3D objects are drawn on a sheet of paper (with fingers, pencils, or brushes), while the designer may rotate the paper to change the viewpoint. With multi-touch screens, 3D objects can be drawn onto a surface (with fingers, digital pencil or brush), while moving and rotating the 2D/3D scene. Similarly, placing objects (with respect of to others) in the physical world is generally done on a planar support, such as a table. Therefore, a multitouch table seems appropriate to design and place 3D objects.

In this work, we focus on manipulating the virtual world by interacting on the screen with fingers. In contrast, recent researches focussed on modeling 3D objects above the devices (without contact). However, as designers may spend several hours to edit objects, we believe that some support for arms and hands on a table should reduce fatigue. We also believe that contact interactions allow for more precise gestures than gestures in the air.

## CONTRIBUTIONS

The aim of this work is to simplify the 3D modeling pipeline. To reach this goal, we choose to study the whole modeling pipeline, from the raw input analysis (analysing gestures) to the edition tasks themselves. The contributions of this work include:

- **The understanding of hand behavior on tabletop:** every 3D editing system is based on the use and comprehension of the interactive tools (in our case, hands on a surface).

Therefore, our first work was to better understand hand gesture on tabletop, as the number of effective degrees of freedom (DoF), or hand decomposition of fundamental behavior (see chapter 3). Based on user-studies, the goal was to develop a set of hand gestures that ensure feasibility and effectiveness for designers, beginners or not. We successfully extracted a new on-line analysis method for hand gestures on tabletops, based on phase analysis.

- **The discovering of a gestural design pattern:** second step of our work was to define a gestural design pattern: a pattern that defines the map between analyzed gestures and possible tasks of the interface. Though, finding this specific pattern required two steps: we first investigate the whole set of possible tasks, according to three main modes: navigation (camera positioning), object positioning and object deformation (see chapter 4). The former mode (navigation) is a requirement of all 3D visualization softwares (and not only 3D modeling). However, results were not consistent enough to , especially because of deformation tasks. For this reason, an entire reflection and times were attributed to deformation tasks alone (see chapter 5).
- **The development of the interface:** considering the combination of all three modes, we develop a specific interface. Believing that the interaction would be similar, whatever the chosen tools to manipulate/deform object, we mainly focus on the interaction point of view (see chapter 6). To develop this interface, we rely on investigating principles from user-studies. A chapter is dedicated to choosen tools for the interface (see chapter 7).
- **The deformation of a specific complex model: garment** obviously, editing object does not only rely on elementary deformations. More complex tasks have to be considered and implemented with other methods, such as the garment deformation while a character body shape is deformed. A first step was performed by developing a new method that automatically transfer garment from a character to another one, preserving tailorship criteria (see chapter 8). Then, this method was extended to process the garment deformation while characters were deformed (see chapter 9).



CHAPTER

— 2 —

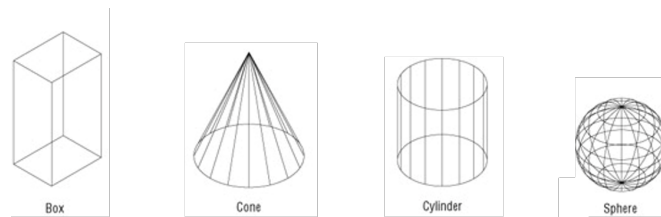
STATE OF THE ART

## 2.1 SHAPE REPRESENTATIONS

The requirement of virtual 3D content is included in an increasing number of field. As a start, let us give a brief overview of 3D shape representation. This section gives the minimal background for understanding the following sections and chapters, which include 3D shape manipulation (positioning or global scaling) and edition (from sketch-based modeling or deformation methods).

This section briefly describes thress different methods to represent 3D virutal shapes: solid primitives, meshes and implicit surfaces.

### 2.1.1 Solid Primitives

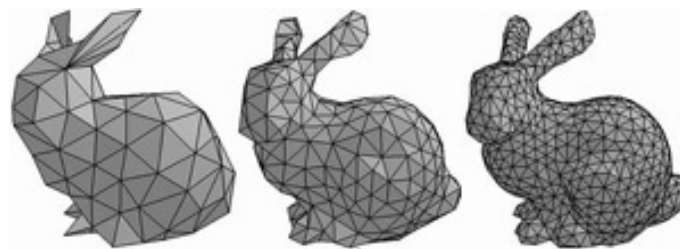


**Figure 2.1:** *Examples of Solid Primitives*

Especially developed for engineers to manufacture real pieces, solid primitives were the early work on digital shape design 2.1. The idea was to express shapes in elementary solid primitives (for instance spheres, cylinders, cubes), defined by their mathematical equations. Following a specific tree of successive operations (for instance union or intersection), the primitive were combined to construct the solid content: *Constructive Solid Geometry* (CSG) [Ric73]. Historically, the order of operations given by the tree CSG tree describes the steps order to manufacture object.

Although CSG is sufficient to design exact regular shapes, it is clearly insufficient and limited to produce any other object. Therefore, other approaches to render and construct free-form object were developed.

### 2.1.2 Meshes

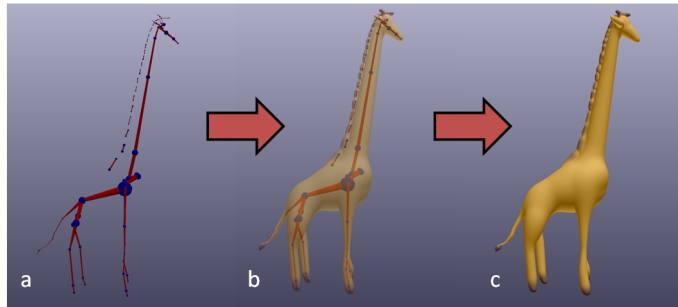


**Figure 2.2:** *Three examples of meshes, from the less to the most refined one*

Commonly, graphical 3D contents relies on polygon meshes 2.2: it is a collection of vertices, edges and faces that form the skin-like surface of an object. More than esaily reproduce any simple object (cube), meshes are able to approximate any complex model; an approximation that might tend to the real surface, depending on the surface refinement.

Since the introduction of Bézier surfaces, continuous surfaces (given by parametric surfaces) could be easily converted to mesh, depending on the desired resolution [Béz, Béz78]. More, parametric surfaces, such as Bézier, B-Spline or NURBS surfaces, rely on a limited number of control points to edit the surface either globally or locally [BBB87, Pie91]. This enable efficient computer visualization and control flexibility.

### 2.1.3 Implicit Surfaces



**Figure 2.3:** *Skeleton-based implicit surface: from a set of vertices and bones, the system generates a surface*

While most 3D modeling is based on basic primitives and parametric surfaces, smooth and deformable objects are still difficult to represent. Another approach to modelize object surfaces is implicit surfaces: the surface relies on an iso-surfaces through the field. Similar to spline surfaces, the first implicit surfaces (called blobs, metaballs or soft objects) also used control points: each point generates a 3D scalar field decreasing with the distance [Bli82, NHK\*85, WMW86]. These fields were summed into a global scalar field.

Thanks to the global scalar field, implicit surfaces do not only generate the surface of the shape, but also characterize the inside/outside of any object, offering many advantages: they correctly define closed surfaces, without any self-intersections, the visualization of the surface could use specific algorithm which rely on this in/out function, such as ray tracing or marching cube techniques.

Since the use of blobs, the techniques were extended in several means: surfaces generated by primitives are no more limited to points but also use more complex implicit primitives, similar to a skeleton(fig. 2.3) [AC02], other blending operators to combine generated scalar field were developed, requiring the use of a constructive tree, similar to CSG techniques [WGG99]. To avoid the generation of bulges at primitives junction, scalar fields depend on analytical convolution [BS91, She99].

In this thesis, we will be using implicit surfaces generated by skeletons for modelling the solid shapes we manipulate

## 2.2 SHAPE EDITION

Although many methods exist to visualize virtual 3D contents, designers require specific techniques to create and animate them. To understand the development of digital modeling techniques, let us look at used techniques in real world first: creating and editing free-form shapes is commonly based on sculpting or sketching.

The former method, sculpting, is characterized by the progressive transformation of a physical 3D model, whatever the used material. Techniques to transform the sculpture clearly depends on the choosen material: commonly, carving tools are used to remove material. However, with soft material such as clay or dough, the user is able to extend the model by adding new pieces, or to globally deform the sculpture. For instance, to keep the right proportions, artist might first sculpted a human figure in a symmetrical resting pose before deforming it to the desired pose. Note that such deformation preserves the model's volume.

Obviously, sculpting techniques are constrained by the phisiscal law. For instance, the sculpture might dry or crack under the medium pressure, or too thin part in the model would break the object.

On the other hand, the latter method, sketching, is more popular anf more familiar by the average people than sculpting to communicate about shapes. Drawer only requires a support surface (such as a paper), a medium to draw (such as a pencil), and ideally an eraser to undo-redo strokes. Therefore, sketching is less limited by the physcial world. However, unlike sculpting, sketching relies on the perception sense: drawing represents a 2D interpretation of the desired shape, in a specific view.

Similar to sculpting, sketching remains a progressive process: designer often starts from the largest features of the draw (such as circle or constructive lines). Then, the shape is progressively refines, features line oversketched in order to incorporate more and more finer details. If desired, the designer might render a more realistic draw by adding shading or color on it.

Although digital shapes are less constrained than real sculpting or sketching, the general process remains similar: the digital artist first start by creating largest features of the shapes, before progressively refining it to the wanted one. Once the digital shapes exactly represents the user's idea, again, the user might progressively animate it. This section describes several existing techniques that deforms digital shapes.

### 2.2.1 Model-Based Methods

The first approach to deform digital shapes relies on varying parameters of the model: for instance, moving the control points of a Bézier/spline curve, or editing the skeleton of an object.

In the first generation of modeling systems, artitsts used this approach as the only avaiable method. However, manipulating complex models using this method requires time, effort. More, such methods are not intuitive for designers: designers should have a perfect understanding of the shapre representation in order to optimally process the model-based deformations.

For instance, the control the designer gets from the skeleton is often indirect, as the skeleton is not located on the surface to deform. A direct approach anyway would not resolve the issue: editing a low level representation such as the mesh itself, is a complex task that might not keep the structural shape of the original design.

### 2.2.2 Space Deformations

Introduced in 1984 with Barr's deformations [Bar84], space deformations is also called free-form modeling or warping. Space deformation methods are characterized by a function from  $\mathbb{R}^n$  to  $\mathbb{R}^n$ .

Simplest methods to globally transform the whole space are linear transformations. using matrices to define the transformation. Simplest transformations are affine transformations: translation, rotation, scale or shear [Bar84, Bla95]. Though, main issue of these transformations is their lacks of control. A solution was to extend and generalize such transformation to bent axis [CR94, CBS96].

Another approach to deform 3D shapes are lattices space deformations. A lattice is a grid or a control box with variable resolution. According to this grid, a mapping in space between original location of the grid's point and their location when moved is easy to produce. Then, the transformation relies on this mapping to warp and distort objects inside the grid.

While first attempt (Basic FFD) to deform space field used cubic grid [SP86], many approaches extend the FFD method to any shape for the grid [Coq90, MJ96, MT97]. However, numerical computation of local coordinates is obviously more complex, and cells connections inside complex grid become tedious.

### 2.2.3 Surface-Based Deformations

Because space-based deformation is defined in space, an inherent issue is that such methods do not take account of the geometry of the model. For instance, animating fingers of a hand requires techniques that should move each finger independently, whereas space-based deformation might be too close in space to be independently deformed.

To focus on this issue, surface-based deformations computes the deformation on the surface itself. By defining several target positions for a number of points located on the surface, the user is able to automatically define the deformation, without editing parameters by hand afterwards. Also called *variational modeling*, such methods depend on optimization techniques: they automatically move the control points or the skeleton to fit the user's constraints and the structural shape intent [WW92, HHK92, Zor05].

Recent surface-based methods are based on a differential representation of the surface, solving large, sparse linear or non linear system of equations to obtain results [SCOL\*04, ZRKS05]. Mainly, these methods are interested in preserving small details in terms of shape and relative position and orientation. Some extensions were proposed to generate constant volume surface-based deformations [ZHS\*05, HSL\*06].

### 2.2.4 Sketch-based modeling

Most of the previously presented methods are hardly predictable for the designer, especially without any insight of either the shape representations or the mathematical model used for the tool. An alternative approach for deformations are sketching system, or sketch-based modeling. Sketch-based modeling systems allows user to construct a 3D shape via sketching. First, user draw the 2D border line of the desired shape. Then the system process the reconstruction of the 3D shape automatically.

First attempt to construct 3D models was the generation of mesh models using sketching. The prototype implementation is Igarashi's work Teddy [IMT07]: once the designer draws the desired model silhouettes, the system generates a plausible 3D mesh. Then designer can iterates the process to draw more and more details to the 3D object.



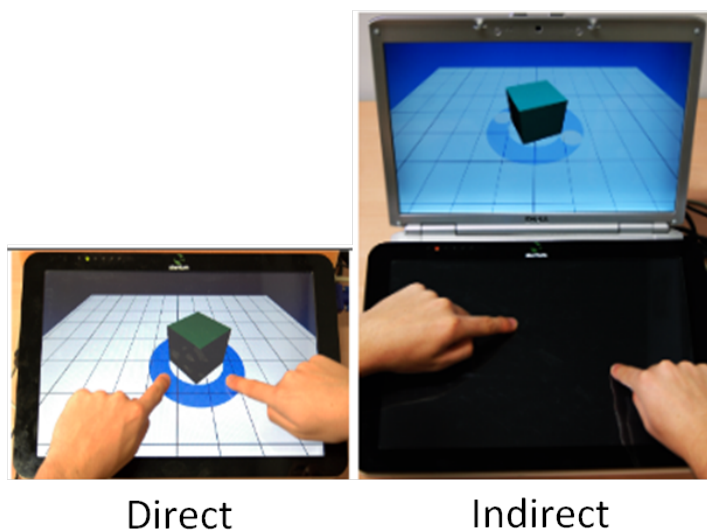
Other attempts to construct 3D models using sketch-based modeling are the reconstruction of implicit models. This approach commonly computes a 3D field function in space from the user sketching, and extract an iso-surface of the field to obtain a globally smooth 3D representation of the user intent. According to this approach, adding new parts to the model is simplified while the shape smoothness is preserved.

Generating implicit surfaces using sketching mainly relies on two distinct approaches: a first variational approach that resolve an equation to compute a surface interpolating the sketch [TO02, KHR02]. Although this method generates globally smooth surfaces, the optimization step consumes times and it is difficult to extract control points from the surface. the second approach is to deduce a skeleton from the sketch, which would generates the surface using skeleton-based implicit surfaces [ABCG05]. This approach offers more flexibility for users to further edit the 3D shape, and enables systems to fit the contour without any costly optimization steps. More, recent study successfully generates globally smooth surfaces that respect the draw topology; even with holes [BPCB08].

### 2.3 RISE OF MULTI-TOUCH DEVICES

With the expansion of graphics tablets and tactile mobile phones, multi-touch devices have grown exceptionally fast.

Multi-touch tables bring a number of features that is well-adapted for 3D design. First, designer may directly manipulate the desired object, and may perform any desired actions such as positioning them, looking around them, sketching them and so on. Then, due to the previous features, interacting on a surface is similar to one of the oldest way to design 3D shapes (which is sketching 2D projections of the shape on a support such as a sheet of paper). Lastly, direct touch manipulation enables the arm to get some support to rest, involving less fatigue than mid-air gestures, and enables the hand to get some haptic feedback during interaction without any additional gadget.



**Figure 2.4:** An example of direct/indirect touch interaction (extracted from [KH11])

A current debate about touch interactions is the differences between direct and indirect interactions. Because of the size of the interacting finger(s), designer may occlude relevant details that he/she manipulates during direct interactions and therefore the precision decreases. Conversely, during indirect interactions, the manipulation is slower and less immersive for the user. Many papers contribute to this debate [KH11]. In this thesis, we focus on direct-touch only, as direct interactions may manipulate 3D content quicker.

### 2.3.1 Devices Combination

Although the hand is a marvelous tool that permits many actions to design 3D shapes, many kind of objects production require the use of additional tools.

Traditional method to sketch 3D shapes on a sheet of paper often requires a pencil. Similar to designers drawing on paper, multi-touch table might be used with a pen or stylus to draw and manipulate 3D contents, which is the favorite additional device combined to multi-touch table [BBS08]. It may indeed complement tabletops to perform specific actions by itself, or altogether [BFW\*08, HYP\*10]. Although several results presented in this thesis focus on hand-only manipulations, some developed methods were thought by combining hand and pen manipulations (for instance, the combination of sketching using a stylus and the deformation using hands).

Even though pen or stylus might be the best additional devices to complete the virtual designer toolbox, several researches also combined other tangible objects to interact with the multi-touch table. Tangible user interfaces (TUI) usually expand the interaction vocabulary by exploiting the adequate tangible objects for a specific tasks [FIB95]. This domain claims that tangible object benefits are: intuitiveness, motor memory and learnability [IU97, RMB\*98, KHT06, HHC\*09].

On the other hand, one of the main issues of multi-touch tables is the dimension gap: multi-touch screens are providing 2D input data, while, in our case, the virtual manipulated space consists in a 3D output. To resolve this issue, other devices could be combined to the multi-touch screen. While some devices seem eccentric [LSS06], the additional devices are commonly related to the stereoscopic perception by the eyes.

However, stereoscopic glasses only permit users to watch the scene in 3D. To successfully manipulate the object above the surface, an additional gadget is required. In many cases, a specific finger detection devices is added and caught by a camera [SPS01, DACJ12b]. To obtain gestures from the complete motion of a hand, a specific glove might also be used [BIF05]. However, the generated fatigue by mid-air motion combined to the additional device weight is not negligible, reducing the precision during modeling process.

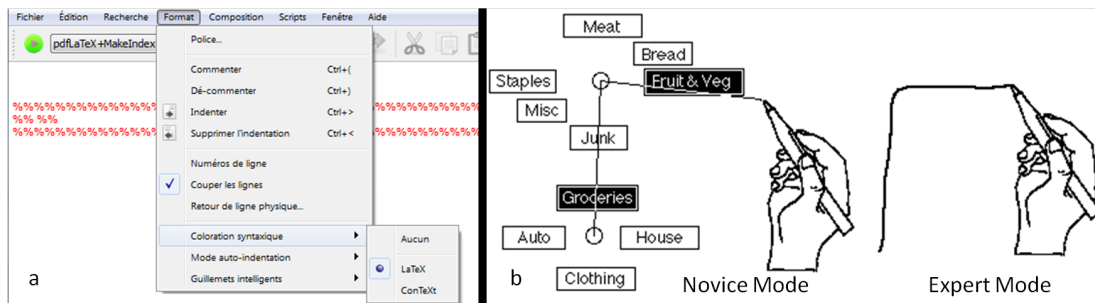
As editing 3D contents usually required a specific selection of the required tools or actions, we first detail related works on modal interactions. One crucial approach inside contextual interactions are gesture-based interactions. Therefore, a specific section is dedicated to them. To handle these gestures, a first step is to analyze raw inputs in order to fully understand the behavior of the hand on a surface. Approach to analyze hand motion is described in a specific section.

## 2.4 MODAL INTERACTIONS

3D editing interfaces provide an overwhelming number of tasks (i.e., actions that is processed by the system), a number that is gradually increasing with research. Because of that large number of possible tasks, specific interaction tools need to be provided to ease task selection. To simplify the task selection, tasks are commonly gathered under a specific mode. For instance, translating or rotating object tasks belong to object positioning mode, whereas extrusion or bending object belong to object editing mode. Although interfaces limit user action to the set of tasks in the current mode, mode/task selection is still compulsory.

This section defines the different existing tools to handle mode/task selection - from the most standard one (linear menus) to the most specific for touch system - and discusses their effectiveness when used with touch screens.

### 2.4.1 Standard Menus



**Figure 2.5:** Examples of 2 standard menus. a) Linear menu, b) Marking menu

Menus remain the most standard approach to provide an explicit mode selection: items are gathered together under significant labels, previsualisation facilitates navigation, and such menus may possibly own an exhaustive set of reachable modes/tasks. The most common kind of menus are linear menus (fig. 2.5.a).

However, linear menus are ill-adapted to touch-screens: to ease selections, big items, occluding a large part of the screen, would be required [RLG09]. Therefore, few multi-touch models interact with linear menus. One exception relies on finger-count interaction [BML12]: the selected menu or submenu corresponds to the number of interacting fingers. Such model is inadequate for tear-down menu, with a large number of possible mode.

The common approach on tabletop, combining gestural interaction and circular menu, is based on marking menus (fig. 2.5.b) [KSB93, KB94]. Marking menus are menus that appear around the interacting point, in a circular shape, such as the Maya<sup>1</sup>'s hotbox. The selection is chosen by a specific gesture (performed with the mouse/finger/pen/etc). Marking menus defined two modes: novice mode, in which users previsualized items, and expert mode, in which users only performed gesture (without feedback).

A main property of marking menus: they offer an ideal transition from novice to expert mode, as they performed the same gesture in both cases. Similar to linear menus, a mark-

<sup>1</sup><http://www.autodesk.fr/products/maya/overview>

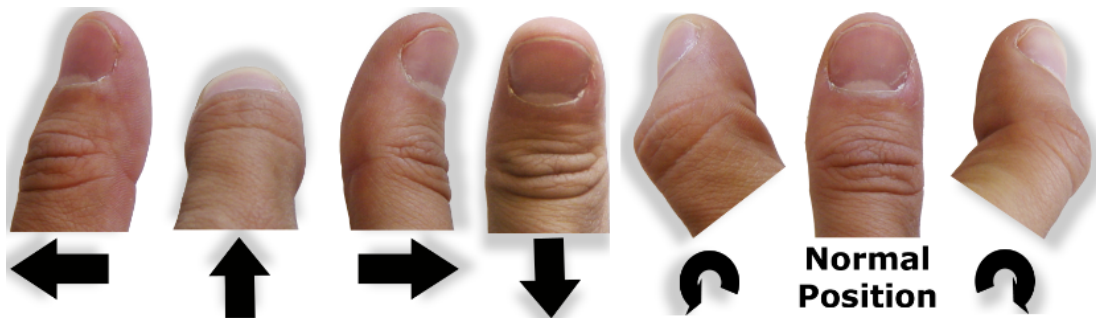
ing menu should be hierarchically classified, and facilitate navigation with previsualization support in novice mode.

However, with a lack of equivalent for the right-mouse button, users usually have to wait a temporal delay on a specific localisation to open the desired menu. About space occlusion, marking menus require more horizontal space than linear menus, occluding large part of the screen [BLN07]. More, performance tends to decrease with depth or for gestures along diagonal axes.

An extension of marking menus, resolving the last problem, are multi-stroke menus [ZB04]: relying on a temporal delay, users have to draw a set of simple inflexion marks instead of a single compound mark. Submenus are superposing over the last one, requiring less physical space, at the cost of previsualization.

### 2.4.2 Specific Menus for Multi-Touch Screen

As previously noted, common menus are ill-adapted to touch-screens: usually occluding large part of the screen, interfering with the scene edition. Therefore, tabletop interfaces require specific contextual interactions. Research on this field evolve in two different paths, according to the screen width.



**Figure 2.6:** *A new type of gesture: rolling movement (extracted from [RLG09])*

With the rise of small touch screens on mobile phone and tablets, mobile devices require dedicated contextual interactions. Despite the large panel of public software that breaks the current action in order to open the menu, many useful tools have been developed in recent research. Based on marking menus, but bounding the menu to a semi circular layout, ArchMenus and ThumbMenus dodge the issue of finger occlusion [HL07]. The RollMark menu is an interesting selecting tool relying on the finger roll motion (fig. 2.6) [RLG09]. This new type of gesture, with a specific signature, makes it possible to be unambiguously recognized on touch screens. However, rolling movement are neither a natural gesture (nearly no novice user would propose such gesture), nor easy to perform when more than one finger is interacting. More recently, Francone et al. proposed wavelet menus, in which menu partially appeared inside the circular width, in a wavelike shape [FBNL09].

Although small screen interaction research proposed interesting interaction methods, they need to focus their research on one hand, few fingers interactions. In our case, in which 3D content has to be manipulated and edited, users require large enough 3D scene visualization. Hence, a larger screen is recommended, and the methods previously described are too much

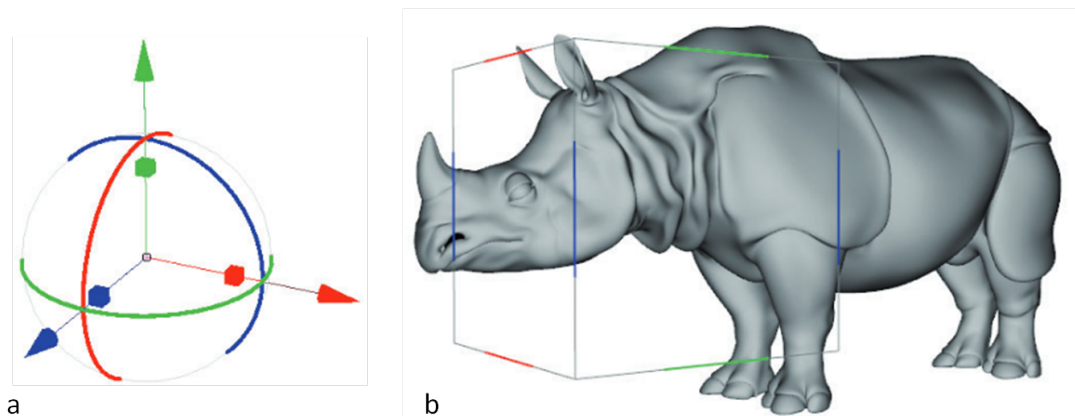
constrained to be adequate.

With a larger screen, the whole hand can be completely located on the screen. Therefore, techniques to activate menu, other than time delay, have been developed. For instance, Bailly et al. register the heel of the hand in order to open menus [BDLN08]. In asymmetric bimanual interactions, a role of the non-dominant hand might be the mode selection one [DACJ12b].

Mostly, multi-touch interfaces interact with mode selection by only manipulating with up to three fingers [WMW09, HHC\*09, MCG10, CDH11]. Considering this restriction as an issue (the upper bound DoF for a hand is ten or higher), Bailly et al. developed MultiTouch Menu, a specific mode selection tools that required the whole hand [BDLN08].

### 2.4.3 Contextual Interactions

Closer to our interest, some mode/task selection tools specifically relies on the context, e.g. the objects presents nearby the interaction in order to select the correct action. For instance, selecting/unselecting a file in a file explorer only relies on the interaction localization (i.e. on/outside the object). Contrary to menus, designer has to learn that a specific area may infer a different action than another area. To become efficient, such direct selection tools should not be too extravagant or too difficult in order to be learned quickly. In compensation, user gains time by directly performing the actions. This is especially true for repetitive actions.



**Figure 2.7:** Widgets that handle the 9 elementary DoF selection: a) standard 3D manipulation widget, b) tBox (extracted from [CDH11])

On the one hand, some interfaces that provide direct selection tools create specific visual feedbacks. A common technique is to manage a specific 3D graphical widgets. According to the interaction location on the widget, a specific task is selected. For instance, the standard 3D transformation widget gathers 3 arrows for translations [Bie87], and may also contains 3 circles for rotations and 3 squares for one-axis scaling (fig. 2.7.a). However, traditional widget might be difficult to interact with, specially when the widget occlude some relevant parts for the selection. To resolve this issue, Cohé et al. developed tBox (fig. 2.7.b), a specific 3D graphical widget that handle the 9 elementary DoF (translations, rotations and scaling) [CDH11].

Graphical feedbacks other than widgets might be used in order to specify the desired tool: a complete region might be dedicated to several specific tasks. For instance, corners or borders are frequently used in such way. To resolve the issue of manipulation the third dimension on a 2D screen, Yu et al. dedicates borders to perform translation along depth-axis or constrained

rotations, and corners to zoom in/out on the screen [YSI\*10].

On the other hand, interfaces might not have specific feedbacks. In such case, users have to distinguish areas themselves. To do so, these areas should be obviously discernible (for instance: on the object vs. outside the object). In many 3D editing softwares, the target of select/deselect tasks are often specified by the interaction localization itself. To handle the depth axis translation, Martinet et al. assigns the third dimension to the second hand that have to be outside the object [MCG10]. For instance, the RnT method manipulates 3 DoF in 2D space [KCST05], and up to 5 DoF in 3D space by only interacting with 1 finger [HCC07]. This method only involves the position of the interacting finger, compared to the center position of the manipulated object.

#### 2.4.4 Gestural-Based Interaction

Due to the rise of multi-touch or depth-based sensor devices, such as kinect<sup>2</sup> or leap<sup>3</sup> devices, a new method to distinguish tasks was also developed: gestural-based selection. Once a specific gesture is detected (compared to a predefined set of gestures), the interface assign to this gesture a specific task to perform. As this selection method is one basis of this thesis, the next section is dedicated to gestural-based interactions (cf. 2.5).

#### 2.4.5 Discussion

Selecting modes/tasks is a complex issue, which mainly involves the interacting device(s). While mouse/keyboard-based interaction may rely on shortcut to quickly switch between modes (for instance, ctrl, shift), this method would be less effective for touch system, especially when user manipulates with both hand.

To resolve the issue, an interesting option would be to combine the two last presented methods: contextual interactions and gestural-based interactions. This combination would drastically increase the number of reachable tasks by direct-manipulation only. More precisely, given that gestures are more reproducible on screen (meaning that users can perform the same gesture anywhere on the screen), the mode could be chosen using a contextual approach, while the specific task could be selected using a gestural method. This is the methodology we explore in the contribution part of this thesis.

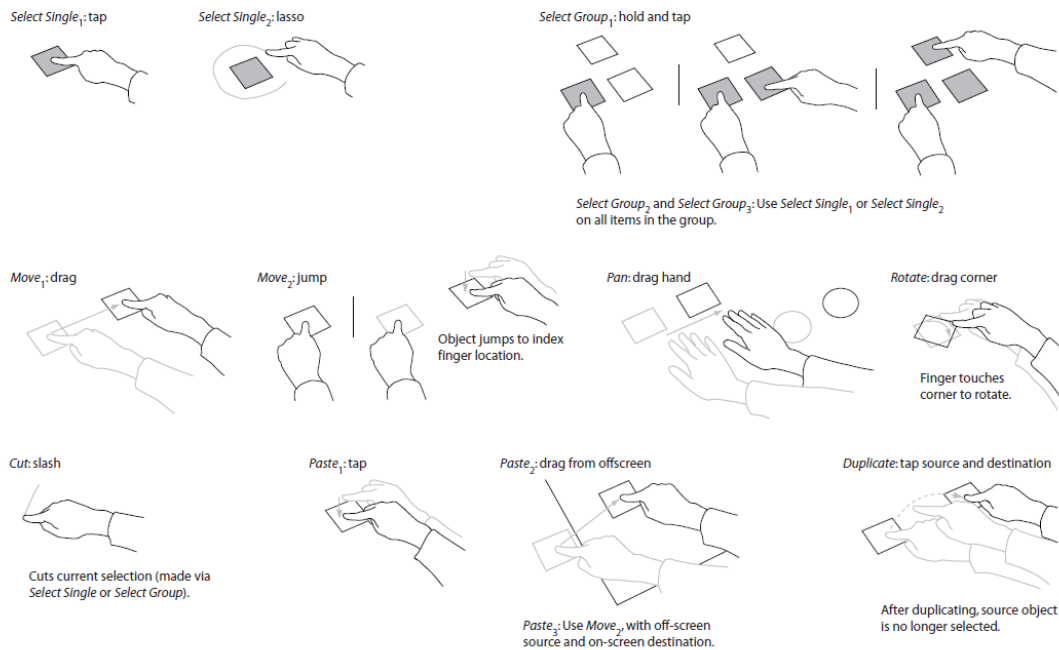
## 2.5 GESTURAL-BASED INTERACTIONS

A gesture is a form of non-verbal communication, in which body actions express particular message. Interpreting gestures is a tedious task, mainly because of the fact that gesture meaning involves cultural background. For instance, while nodding usually indicates agreement, in some country (such as in Greece) a single nod indicates a refusal. Thanks to Cadoz's studies, gestures could be classified into three main groups, depending on the gesture functions [Cad94]: epistemic gestures (gestures to learn through tactile or haptic experiences), ergotic gestures (the capacity to work and manipulate the physical worlds, to create artifacts), and semiotic gestures (gestures to communicate information and results).

Mathematically speaking, a gesture (on tabletop) is a set of continuous interaction positions associated with a specific meaning such as tapping, encircling, etc (fig. 2.8). Despite

<sup>2</sup><http://www.xbox.com/en-CA/Kinect>

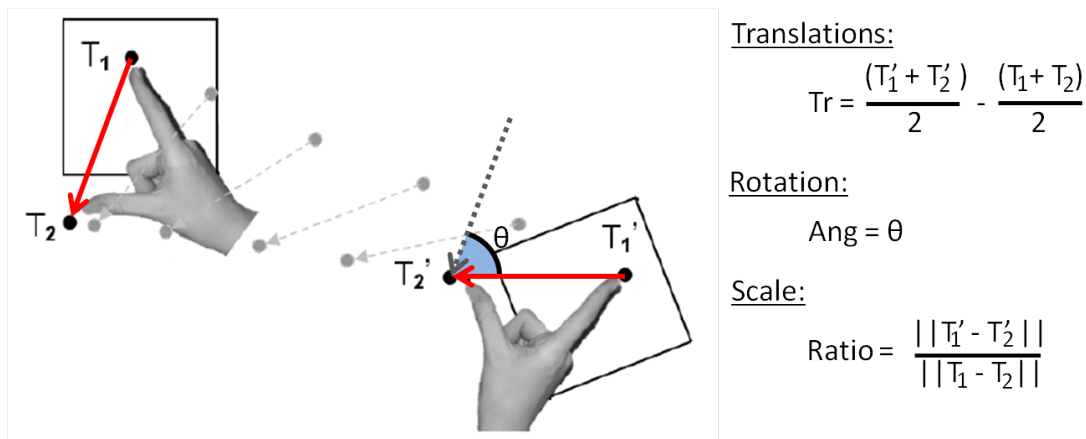
<sup>3</sup><https://www.leapmotion.com/>



**Figure 2.8:** A subset of possible user-defined gestures from Wobbrock's paper [WMW09]

this simple definition, interpreting gestures with a computer is still a complex task that limits the interface to a small subset of understandable gestures. In this section, we review several different methods using gestures, from their raw inputs translation to their association to tasks.

### 2.5.1 Extension of the standard 2D method: Rotate-Scale-Translate (RST)



**Figure 2.9:** RST methods processed on a square (partially extracted from [HCC07])

The Rotate-Scale-Translate (RST) interactions become a standard to manipulate contents in 2D space [KFBB97]. From two interacting points, the RST method calculates meaningful parameters from the performed gesture: translation, rotation and scale (fig. 2.9). Precisely, rotation is relying on angular data, scale on the length ratio and translation is linked to the

barycenter translation itself. Therefore, as every parameter is calculated with purely geometrical data, this method is quick to process.

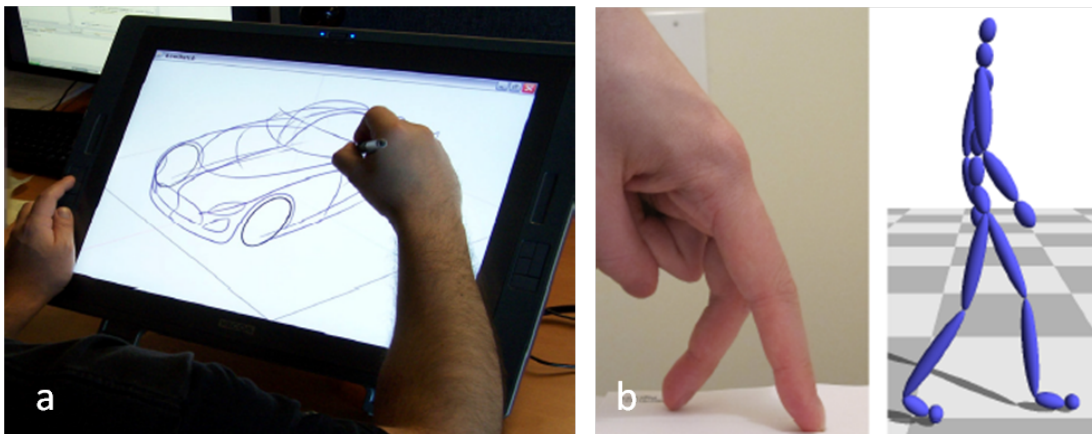
Once raw inputs are translated, the parameters are linked to their equivalent tasks (translation to object-translation task, rotation to object-rotation task and scale to object-scaling task). Hence, the second strong point of this transformation technique relies on the input-output dimensions match: 4 DoF for input (2 fingers on a 2D plane), and 4 DoF for output (2 translations, 1 rotation, 1 scaling).

As the RST interaction methods is efficient on 2D space, many recent researches desire to extend it into 3D spaces (to control at least the elementary DoF, which are translations, rotations and (sometimes) scaling). Close to the 2D space transformations, Knoedel and Hachet studied the RST technique by blocking 3D object on a planar ground [KH11]. In visualization, 2D RST techniques is combined with border interactions to manipulate the 9 elementary DoF [YSI\*10]. However, directly manipulating a higher number of DoF is more complex.

An interesting observation: in several cases, a second hand is required to manipulate a higher number of DoF. To clearly dissociate the DoF (skipping scaling part), Hancock et al. add a third finger to RST, and compared it to extension of RnT ones [HCC07]. In this paper, they observe that interaction dissociation is more efficient than others. Reisman et al. associated the RST methods with a second hand, manipulating the 3D scene like an implicit edited cube [RDH09]. In Martinet's work, the translation along depth axis is handled by an outsider interactive point (therefore, a finger from a second hand) [MCG10].

### 2.5.2 Other Multi-Touch Gestural-Based Interactions

In most cases, manipulating contents required the interaction to be located on the edited object. For instance, to manipulate 6 elementary DoF (translations and rotations) of an object, Liu et al. constrained themselves to two-finger gestures on the object [LAFT12]. But this is not always true: Au et al. preferred to select the manipulated axis by referring the axis direction and not their positions [ATF12].



**Figure 2.10:** Example of gestures: a) sketching gestures (ergotic), b) walking gestures (semi-otic) (extracted from [BBS08, LS12])

The easiest way to communicate with hand is mimicking. The first ever one: sketching on a electronic table is simply a clone of the traditional drawing gestures (fig. 2.10.a), even using



sometimes the same tools [IMT07, BBS08]. But, thanks to sketching modeling techniques, sketching are no more limited to the projection plane model (further details are given in the dedicated sections).

But, registered gestures that mimic tasks do not only stop on sketching. For instance, the natural gesture to express a walking man/woman is by mimicking it with two fingers (fig. 2.10.b), which could be registered by multi-touch systems [KGMQ08, LS12].

### 2.5.3 On-Air Gestural-Based Interactions

Different kind of gestural-based interfaces are gestures performed "on-air". In recent research, two main approaches exist: pure on-air gestures, and on-air combined to touch gestures.

Pure on-air gestures do not rely on multi-touch systems, but usually required depth captor devices. However, these gestures are not the one we are interested for. To obtain further details about on-air gestural-based,

Closer to our interest, a second approach consists in combining multi-touch gestural devices with on-air gestural devices. A main issue for basic touch screen is that the equivalent of mouseover event does not exist (for instance, when a user go through a menu item with the mouse icon, a visual feedback, caused by the mouseover event, appears and shows to the user that he/she is effectively above the item). Basic touch screens, as they directly interact with the system, cannot detect the presence of the finger above the different contents. To resolve this issue, some studies present alternative methods to accept the selection [SVC\*11]. New touch sensor technology (for instance, Floating Touch<sup>4</sup>) permits touch screen to detect the finger above the screen (up to 20mm in this case).

To effectively manipulate above the screen (at least higher than touch sensor technology can), an additional device is required, such as a grapping glove [BIF05]. Combining sketching interface on the screen, and a tracking device, De Araùjo et al. developed a system that effectively edit 3D contents on and above screens [DACJ12b]. Then, this work was extended in order to recognize more gestures, thanks to depth sensor camera [MFA\*14].

### 2.5.4 Discussion

Mouse and keyboard interactions offer a large panel of possibilities for computer users (as a proof, the overwhelming number of proposed software, either for professional or for individual. The benefits of touch interactions, specially for one finger interactions, are not always clear [FWSB07].

The multi-touch interaction strength resides in the large panel of possible gestures. Where mouse kind of gestural interactions are limited (for instance, click, double click and drag), hand gesturing is a robust phenomenon, found across cultures, ages and tasks, that even permits deaf people to communicate [GM99]. Therefore, communicating to an interface with gestures seems a natural approach that everyone is able to.

However, touch screens currently own two main issues to be perfectly efficient. Firstly, performing gestures on a touch screen constrain the hand motion to a subset of *plane* gestures. Without additional devices, an upside-down gesture is ineffective and imprecise. Secondly,

---

<sup>4</sup><http://developer.sonymobile.com/knowledge-base/technologies/floating-touch/>

both users and computers have to learn the possible set of gestures. To efficiently perform the desired tasks, users have to learn the recognized gestures from the interface. For computers, analyzing hand gestures - and classify these gestures into a set of predefined gestures are required. Although on-air gestures detected by depth-sensor camera do not own the first issue (the hand motion is not limited to plane gestures), the analysis of hand motion is much more complex, which limits the predefined set of known gestures from the interface. A third issue might exist, depending on the interfaces: gestural interaction features are not self-revealing, which means that there is no graphical elements that suggest and reminds the user how to perform the gestures.

Though hand possible gestures correspond to an impressive number of gestures, thanks to Cadoz's studies, gestures could be classified into three main groups [Cad94]: epistemic gestures (gestures to learn through tactile or haptic experiences), ergotic gestures (the capacity to work and manipulate the physical worlds, to create artifacts), and semiotic gestures (gestures to communicate information and results).

## 2.6 HAND BEHAVIOR ON TOUCH SCREEN AND ITS LIMITATION

To fully understand the hand behavior on multi-touch table, raw inputs have to be analyzed. Hand gesture analysis is a broad topic connected to many research fields: from biological to device interaction issues, a large panel of fields requires the comprehension of hand motions.

Although the DoF upper bound of the hand is 27 (4 in each finger, 5 for the thumb, and 6 for wrist rotations and translations) [ES03], it is well known that finger motions are interdependent, with a lower effective number of DoF for the hand. Despite someone may train their hand to achieve better precision (and therefore a higher effective number of DoF, such as musician, magician and so on), our interest is limited to the average people, without any specific everyday training.

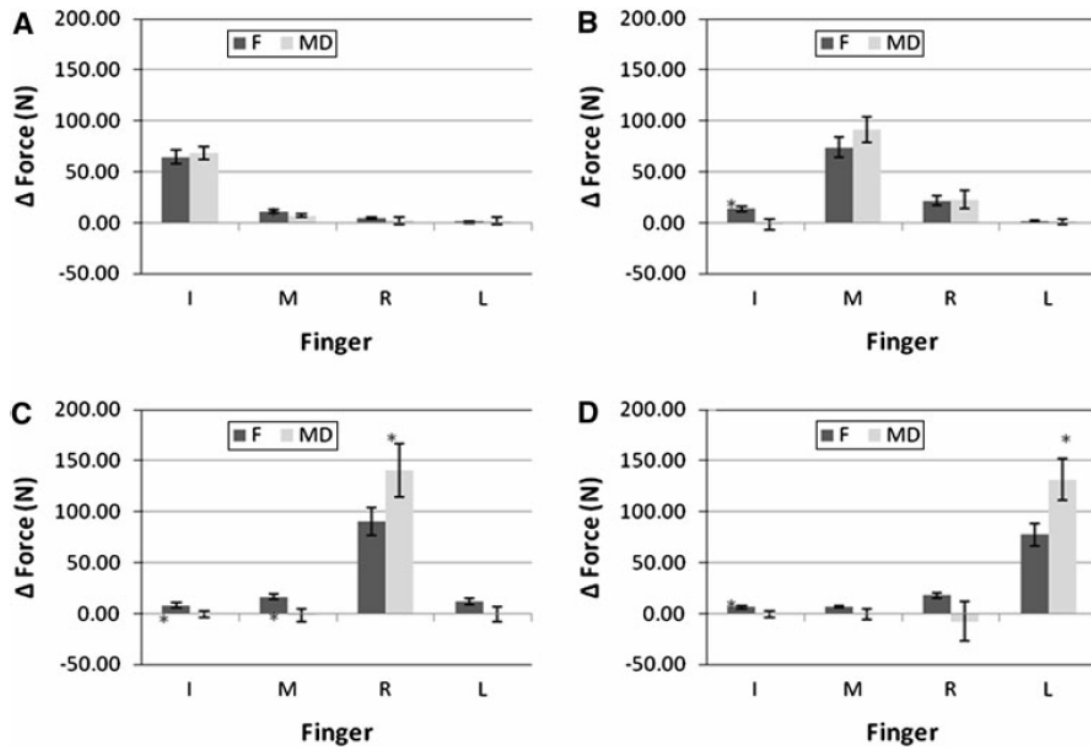
### 2.6.1 Finger Dependencies

Biologically speaking, hand and fingers are linked together by tendons, nerves and so on [Zan]. Three main nerves compose the hand, in which only two lead into fingers: median nerve and ulnar nerve. Curiously, both nerves lead into the ring finger. In hand simulation paper, such as in ElKoura's paper, they required a full knowledge of the hand coordination to perfectly simulate it [ES03].

Neuroscientists take part of the fingers dependencies by decomposing their motion into principal components, and observe that it can be usually decomposed into two main principal components [SFS98].

To have a better comprehension of finger dependencies, force production created during finger interactions have been mechanically analyzed. For instance, a force produced by a subset of fingers induces force production from the remaining fingers [ZLL98, ZLL00]. This is even true when the force is produced either voluntary or involuntary [MZL11].

The figure 2.11 illustrates several results of induce force productions. A quick observation shows that the dependencies are stronger for the interacting finger neighbors, and the index finger engender less dependencies than others.



**Figure 2.11:** Example of force production induced by the interaction of the A) Index (I), B) Middle (M), C) Ring (R) and D) Little (L) fingers (extracted from [MZL11])

Knowing the finger dependencies, interfaces can classify and analyze the whole hand motion. Therefore, the next section details the hand analysis state-of-the-art.

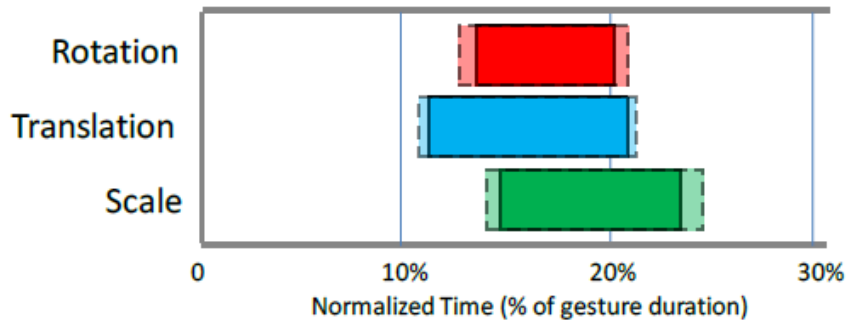
### 2.6.2 Hand Analysis

The main goal to analyze hand motion is to transform the interacting hand/finger raw inputs into readable inputs for the interface. Traditionally, multi-touch raw inputs are transformed into three main parameters: translation, rotation and scaling.

A first approach to analyze hand gestures is geometrical approach. Relying on finger translations, angles or stretching, the interface decomposed the gesture into three main parameters: translation, rotation and scaling. The Rotate-Scale-Translate (RST) method becomes the *de facto* standard method for 2D transformation, and depend on a geometrical approach [HVW\*06]. This analysis generates a specific motion/gesture, that is compared to a predefined set of gestures. More details are given in the section dedicated to gesture-based.

A geometrical decomposition of the hand motion brings many features. For instance, parameters are quick to calculate, and the object motion is perfectly following the touch interactions. However, such analysis is noise-sensitive (specially from hand tremor), and are more difficult to perform for more than 2 fingers.

Another approach to analyze hand gesture is a temporal analysis approach. At first, this approach was performed in a off-line methods, in order to discover an order of manipulation between the different phases. To learn about the temporal distribution of the interactions, Wang



**Figure 2.12:** From Nacenta's work [NBBW09], duration of the periods of maximum activity. Dotted lined define the confidence intervals.

et al. normalized the input signals, and calculate the contiguous area of the most significant one [WMSB98]. With a similar method, Mason and Nacenta also analyzed the input signals with a temporal approach [MB07, NBBW09]. Nacenta et al. extended the techniques to add the scaling phases analysis, obtaining the following manipulation order: translation, rotation and scaling. In both Wang and Nacenta works, the rotation phase is often contained within the translation phase (fig. 2.12).

However, to our knowledge, this temporal analysis approach has never been adapted to process on-line raw finger inputs into gestural inputs. In this thesis, we proposed a new method to analyze hand motion on surface relying on all phases by a phase-based hand analysis. For comparison, Nacenta et al. focused their temporal analysis on only one phase for each analyzed interaction.

Other approach to analyze hand motion exist, such as motion capture, or depth captor devices, but as this thesis focus on multi-touch system only, no further details are given here.

### 2.6.3 Bimanual Interactions

As this thesis focusses on large touch screens, *bimanual interactions* is a key concept. Indeed, while only one hand is sufficient for performing tasks on small tactile screens (the second hand being used for hanging the mobile phone itself) larger screens own larger space in which two hands fit and might interact.

Bimanual interactions are divided in two, depending on the symmetric/asymmetric roles of the hands [Gui87]. In further words, in symmetrical bimanual interaction cases, both hands own a similar role. On the opposite, asymmetric bimanual interactions attribute different functions to each hand. In the latter case, a mean to register DH from NDH has to be performed.

Both cases have proven their advantages and flaws, according to the desired performing tasks. On the first hand, symmetrical bimanual interaction has proven to be adequate to process exclusive spatial tasks - such as docking for instance [BH00]. Describing shapes with hands reveal to be more efficient by also performing symmetrical interactions [LKC05, LBS\*11].

On the other hand, asymmetric models were initially mimicking existing physical asymmetric tasks, such as digital tape drawing [BFKB99, GBK\*02]. Suggested by Guiard [Gui87], NDH should be used to manipulate objects or view, while DH, which is more precise than the NDH, should be used to edit the object or scene. This is the case in Balakrishnan's paper: while

the NDH controls the virtual camera, defining a frame of reference, the DH edit the object in the defined frame [BK99].

When multi-touch table is combined with another device, such as a pen, the hand role is clearly defined and asymmetric. In the IloveSketch system, users may control the virtual camera while the other hand is sketching with the pen [BBS08]. For De Araùjo et al., the DH manipulates and edits the 3D contents with the specific device, while the NDH, once registered, interacts with the possible contextual interactions [DACJ12b]. More generally, the device itself creates the role specifications. Here are other examples that proposed specific commands, depending on the interacting devices (mainly pen devices): [BFW\*08, HYP\*10, LI10, LMAJ11, BIF05, DACJ\*12a].

Both unimanual and bimanual gestures are handled by our developed interfaces: each hand owns its specific role, a support role (which defines the transformed/untransformed area), and an editing role (which effectively transformed the scene). Therefore, asymmetrical model is predominant. However, a main issue remains: the dominant/non-dominant hand registration. Our system does not rely on any additional device to capture hands, but relies on starting positions and distance parameters.

CHAPTER

— 3 —

UNDERSTANDING HAND DEGREES  
OF FREEDOM ON A SURFACE

### 3.1 INTRODUCTION

The interactions used to manipulate, create or edit 2D/3D contents need to control simultaneously a large number of DoF. For instance, the classical docking task (i.e. defining the position and orientation of an object) requires the control of 6 DoF in 3D space. With the recent rise of tabletop devices, especially multi-touch devices, the number of DoF that can be simultaneously controlled on a tabletop device is high: since each fingertip specifies a 2D position on the screen, the use of a single hand theoretically allows the control of  $5 \text{ fingers} \times 2 \text{ DoF} = 10 \text{ DoF}$

This value of 10 DoF is clearly an upper bound of the effective number of DoF that a user can simultaneously manipulate with a single hand. Several evidences show that the actual number is much lower. In practice, due to finger dependencies, nearly no multi-touch interaction method uses the positions of the five fingers of a hand to control 10 parameters (see chapter 2).

In this chapter, the main goal is to evaluate an effective upper bound of the number on DoF that can be simultaneously controlled by a hand on a multi-touch device. To aim that goal, we first propose a new method for analyzing hand motion on a table. The originality of our solution, based on phase decomposition is to use temporal information in contrast with previous, purely geometrical methods. Once the hand analysis method defined, we performed an experiment that confirms and refines what our common sense, as well as what the current corpus of current multi-touch interaction techniques tell us: the number of DoF of the hand on a surface is between 4 and 6.

### 3.2 HAND MOTION ANALYSIS: PHASE-BASED REGISTRATION

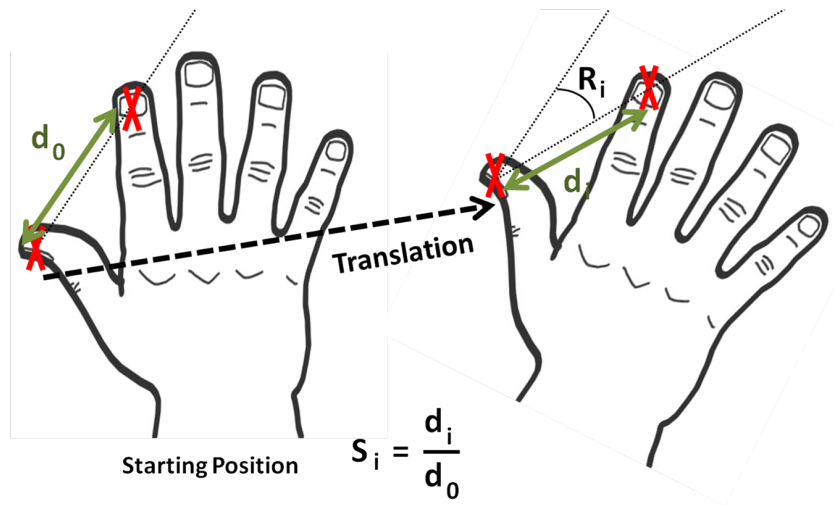
RST technique (introduced in section 2.5.1) is the *de facto* standard method to interact with 2D system. However, RST technique is limited to two interactive points - two-fingered gestures - which restrains user to perform more complex gesture. More, as a purely geometrical hand analysis, RST technique is prone to noise produced by hand tremors.

In this section, we proposed a new approach to analyze hand gesture and decompose it into relevant parameters. To successfully develop the method, we first have to define a specific parameterization of the hand based on a polar coordinate system centered on the thumb interaction point (section 3.2.1). Afterwards, section 3.2.2 details the algorithm to decompose gesture into either global parameters (which characterize the whole hand motion) or local parameters (which characterize a single finger motion). For global parameters, we desire to extend the RST technique to the whole hand motion.

#### 3.2.1 Specific Hand Parameterization

A first requirement to analyze and decompose gestures is to register the local frame of the hand. Therefore, we define the hand local frame as follow:

- the origin of the local frame is centered on the thumb position
- the reference axis is given by the thumb/forefinger direction in the starting position of the hand
- the second axis is simply given by the cross product with the screen normal to complete the frame.



**Figure 3.1:** *The chosen hand parameterization*

Hence, the hand position is given by the local frame (2 DoF for the origin position), and by the position of each finger in the frame (0 DoF for the thumb, 2 DoF - distance and angle - for remaining fingers). Figure 3.1 illustrates the selected hand parameterization.

More precisely, the position of each finger ( $i$ ) in the local frame ( $t$ ) is parameterized by a couple  $(R_i^t, S_i^t)$  - for rotation and scale (fig. 3.1).  $R_i^t$  is the angle defined by the finger of the local frame (i.e. the angle between the thumb - reference axis and the thumb - finger axis at the current position).  $S_i^t$  is the ratio between the current distance to the thumb of the finger, and its distance to the thumb at the starting position.

Although the hand analysis is unchanged by the choice of the second finger for the reference axis (only an offset is added to the  $R_i$  parameters), the reason to prefer the forefinger is twofolds. First, in everyday tasks, the two most used fingers of the hand are the thumb and the forefinger. Second, at the starting position, every  $R_i$  owns the same angular sign, which is more practical to identify the interacting hand (i.e., left hand or right hand).

With these definitions, a simple translation of the hand keeps the couples  $(R_i^t, S_i^t)$  unchanged (only the origin of the local frame, i.e. the thumb, changes). Meanwhile, a rotation of the hand changes all the  $R_i^t$  by the same amount, but has no impact on  $S_i^t$  parameters. In contrast, a pinch gesture does only impact the  $S_i^t$  parameters, decreasing them from 1 (i.e. the finger is exactly at the same distance of the thumb than while resting in the starting position) to a smaller value.

Although this parameterization locally quantifies the finger motion, the global gesture of the hand has not been processed yet. The global parts consist of:

- **Position:** the hand Translation (T), quantified by the position of the origin of the local frame (i.e. of the thumb)
- **Orientation:** the hand Rotation (R), defined as a weighted barycenter of the  $R_i^t$ .
- **Scaling:** the hand Scaling (S), defined as weighted barycenter of the  $S_i^t$ , similar to R.

The weights are chosen to reduce the impact of a finger that is far from the others (i.e. to provide a kind of continuous median value), e.g., for R:



$$R = \frac{\sum_i w_i^t R_i^t}{\sum_i w_i^t} \quad \text{with} \quad w_i^t = 1 + \sum_{j,k \neq i} (R_j^t - R_k^t) \quad (3.1)$$

For comparison, processing R and S with only two fingers exactly generates the same values than the geometrical RST method. Then, we focus on the second technical issue of the RST, i.e., the noise impact on parameters.

### 3.2.2 Phase Registration

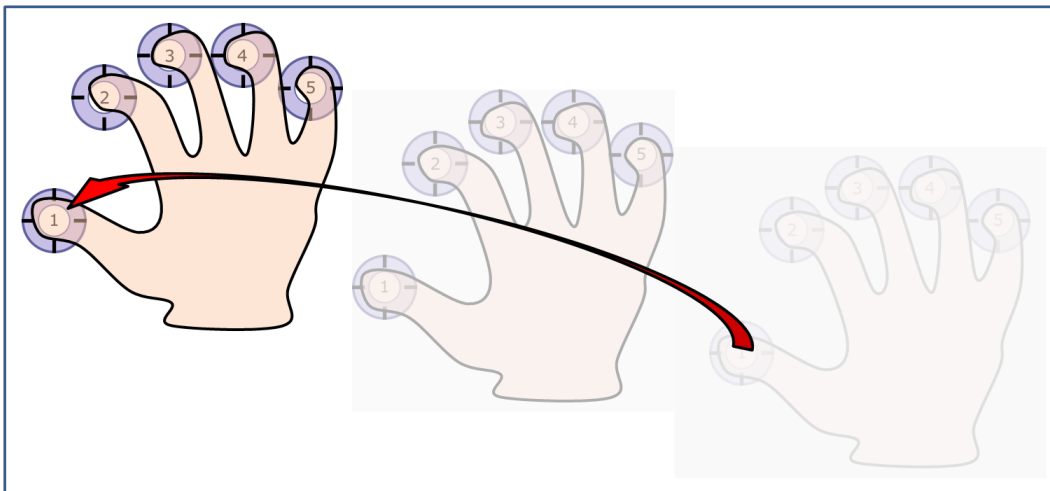
Once input data are pre-processed, a parameter variation can be either relevant or not. For instance, hand tremors are only producing noise, and should not be accounted as hand motion. To resolve this issue, we draw our inspiration from Nacenta work: gestures were analyzed off-line with a temporal process to discover the manipulation order [NBBW09]. In our context, hand gestures are decomposed into **phases** (i.e., the time interval in which a specific gesture, such as hand global translation, is registered), which characterise for the system whether the analyzed signal is significant ( $phase = 1$ ) or not ( $phase = 0$ ).

More precisely, a significant variation occurs for a variable when their first derivative is above a threshold. To dodge any noise effect that would impact the phase result, the signal is first filtered. An example of signals and resulting phases is illustrated in figure 3.3.

The main issue about processing phases is the choice of thresholds: in our case, thresholds are obtained experimentally. Fortunately, only the user speed impact this choice. The choice of thresholds does not involve the screen dimension. As a proof, nearly all parameters are angular or length ratio data - they are mainly dependant of the local frame. For thumb translation signals, as signals are derivated before comparison, only the local user speed is relevant.

At first, this phase registration is performed off-line, for each stored hand motion. Section 6.2 is dedicated to the on-line phase registration.

## 3.3 USER-STUDY



**Figure 3.2:** Experiment: each finger has to be moved from a starting point to an ending point

To get a better understanding of possible hand motions, specifically when fingertips are constrained to remain on a table, we ran a first experiment that does not involve any 3D task. Since our goal was to estimate the number of DoF a user is able to simultaneously control with a single hand, participants were asked to use their dominant hand to perform gestures.

### 3.3.1 Experiment

Participants were asked to perform several hand gestures on a surface. The gesture is specified by a starting position and an ending position (fig. 3.2). Those positions consist of five circles, each circle (resp. labeled with 1, 2 and so on), representing the position of a finger (resp. the thumb, the forefinger and so on). Once a finger is correctly positioned, the corresponding circle turns green. Once all fingers are correctly positioned, the circle vanish, and the ending position appears. Then, the participant has to move his/her fingers to match the ending position, while keeping fingers in contact with the surface. He/she can take as much time as desired to perform each gesture.

The experiment was composed of thirty-seven trials, each of them designed with various complexites: the simpler ones only involve movement of the whole hand, while the more complex ones involve the combinations of both hand movements and individual uncorrelated finger movements. Our set of gestures was designed by testing in a preliminary study a comprehensive combination of elementary movements, discarding those that were too difficult to perform.

For the first ten trials, an animation between the starting and the ending position was shown to the user prior the trial, whereas no path was suggested for other trials. The participants were not asked to follow the suggestion, and its presence had no noticeable effect on the results we report.

### 3.3.2 Apparatus

This experiment was conducted on a 22" 3M multi-touch display <sup>1</sup> (473 × 296 mm, 1680 × 1050 pixels, 60 Hz, 90 DPI). The software was based on the Qt <sup>2</sup> and Ogre3D <sup>3</sup> libraries.

### 3.3.3 Participants

31 participants, composed of 8 women and 23 men, were tested. Average age was 30 (min. 22, max 49). All participants had normal or corrected to normal vision. For left-handed participants, the experiment was mirrored. Participant's background was variable, and not only computer scientist background. Participants' experience with 3D applications, and tactile devices was variable, but this was not an issue, as the goal of the experiment was to get some understanding of fundamental physical behavior.

## 3.4 RESULTS: GLOBAL HAND MOTION ANALYSIS

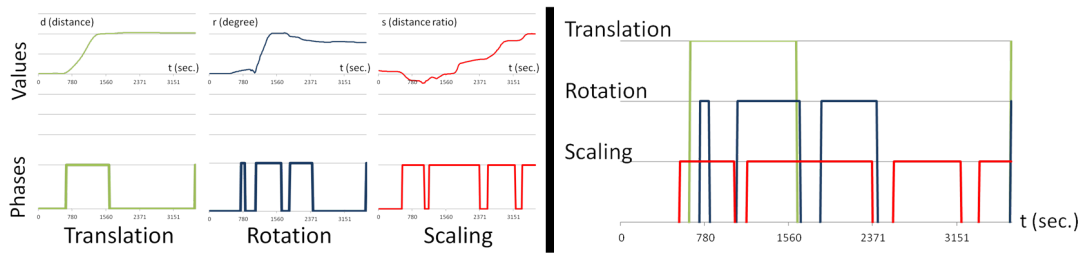
For each trial of the user-study, the hand motion is stored and decomposed in the hand parameterization defined above. Then, the signals of the Translation (T), Rotation (R) and Scaling (S)

---

<sup>1</sup><http://www.3m.com/>

<sup>2</sup><http://qt-project.org/>

<sup>3</sup><http://www.ogre3d.org/>



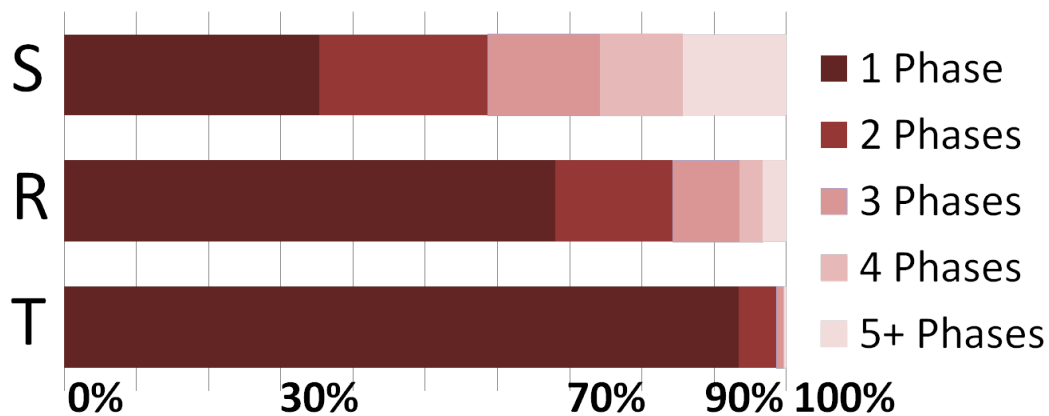
**Figure 3.3:** Left: Global variations (top) and corresponding phases (bottom), during a gesture: variations of translation, rotation and scaling. Right: Phase superposition for the same gesture.

of the global hand motion (fig. 3.3, left top), is analyzed using our phase registration method (fig. 3.3, left bottom).

The pattern formed by this example is typical of what can be observed: there is a single phase for translation, while rotation and scaling are achieved during several phases - typically less phases are required for R than for S. The phases start roughly at the same time, but end in this order: first Translation, then Rotation, and finally Scaling. This manipulation order is similar to the one observed by Nacenta et al. [NBBW09], since what they call "period of maximum activity" are the second phase for R and the second or third phase for S.

To further validate this order of manipulation, we can observe the number of needed phases to validate the user-study trials. Figure 3.4 summarizes those results: for more than 93% of the cases, users need a single translation phase to correctly position their hand; while a correct rotation is achieved within a single phase for 68% of the trials and a correct scale for only 35% of the trials.

From these observations, we believe that hand gestures can be decomposed into sub-parts with variable degrees of stability: from the most stable motion (global translation) to the less stable one (one finger motion). More, for a specific motion, the impact of less stable motions is usually negligible.

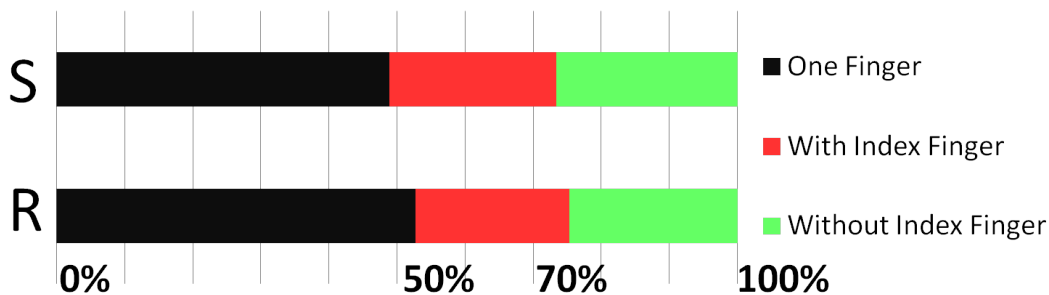


**Figure 3.4:** Percentage of tasks where 1, 2, 3, 4, 5 or more phases are required among all tasks and participants for Translation (T), Rotation (R), Scaling (S)

To illustrate this observation (fig. 3.3, right): first, global translation is the easiest to get right (1 phase only), without any interference afterwards. Conversely, global translation might induce interferences on global rotation (first rotation phase), before that the major rotation motion is performed (second rotation phase). As noted by Wang or Nacenta, these phases are performed simultaneously [WMSB98, NBBW09]. However, due to this simultaneity, sometimes rotation has to be corrected (third phase). Similar reasoning can be done on scaling phases.

### 3.5 RESULTS: FINGER MOTION ANALYSIS AND DEPENDENCIES

The local part of a gesture is the part of individual finger movements that is not explained by the global T, R and S described above. A first look at the data shows that local parts are mainly movements performed by the middle, ring and little fingers. For individual finger, the analyzed couple is  $(R_i^t, S_i^t)$ , which represents the Rotation and Scaling part. To get a better understanding of those motions, our analysis is focused on the trials in which users had to perform movements involving a subset of those fingers (middle, ring and/or little).

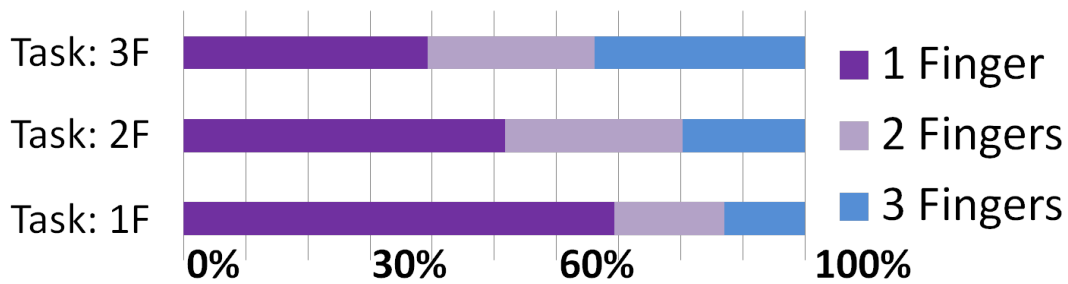


**Figure 3.5:** Average percentage of time spent for moving a single finger, or more than other fingers, including/excluding the index finger for tasks involving the motion of one or more fingers among the last three fingers only

For those tasks, fig. 3.5 illustrates the proportion of time spent moving a single finger ( $\approx 50\%$  of the time), the time spent moving more than one finger, including or excluding the index finger ( $\approx 25\%$  each). We can note that these proportions are roughly similar when the participants are asked to perform a rotation task - to control  $R_i^t$  - versus to perform a scaling task - to control  $S_i^t$ .

This illustration shows how much controlling the three last fingers simultaneously and independently is difficult. Moreover, it is also interesting to note that the movement of one (at least) of the last three fingers involves the motion of the index finger despite that the index finger was not supposed to move in those tasks. The interdependence between these fingers is consistent with the study conducted by Martin et al. [MZL11].

To further investigate the interdependencies among the last three fingers, we split the trials into three groups, depending on the number of fingers the users had to move among the middle, ring and little fingers.



**Figure 3.6:** Average percentage of time spent for moving 1, 2 or 3 fingers among the last three fingers, when the user was asked to move 1, 2, or 3 of them (1F, 2F, 3F)

Fig. 3.6 illustrates for each group (vertically: 1F(inger), 2F, 3F), the relative time spent moving 1, 2, or 3 of those fingers. This leads to an interesting observation: even when participants were asked to move a single finger (1F), they spent more than 30% of their time moving two or more fingers. Conversely, when participants had to perform the same motion for the last three fingers (3F), only one third of the time was used to move the fingers together, while nearly 40% of the time, fingers were moved individually.

This investigation confirms that the three last fingers can hardly be used to control something independently, even independently from the index finger, even if they are used together as a whole.

### 3.6 CONCLUSION

Theoretically, multi-touch devices offer the possibility of manipulating 3D scenes while simultaneously controlling many DoF: up to 20 actually, if both hands are considered. However, this upper bound is never reached, even for trained participants (such as musician).

On the one hand, global motion of the hand provides 4 DoF (2 translations, 1 rotation and 1 scaling), with a specific manipulation order linked to the different degrees of stability. Although more stable motions inferred to less stable ones, global gestures are often easy and quick to perform. In reverse, local motion of the hand is often difficult to perform, especially for untrained users. Because of the interferences between fingers and to their restricted motion when moved in contact with a plane, complex gestures involving all fingers are often unstable, and the time spent to perform such gestures would be prohibitive for an interactive use.

Therefore, it is difficult for users to efficiently control the 10 DoF of the hand. As shown by our study, this upper bound should be decreased to 4 DoF if only stable, global motion is consider, and to 6 DoF when the motion of a maximum of two independent fingers is to be accounted for. To further validate these results, a second user-study, about gestures, is performed and described in the next chapter.

CHAPTER

4

UNDERSTANDING HAND  
GESTURES ON A SURFACE

## 4.1 INTRODUCTION

In the previous chapter, we discovered that the upper bound of using 10 DoF for the hand is difficult to reach, especially for average users. Our next goal is to identify an effective mapping between a set of gestures and 3D editing manipulations. During our investigations, we focus on three modes of possible manipulation tasks:

- Navigation mode: the user point of view (or camera) is transformed
- Object Positioning mode: an object (or a set of objects) is transformed
- Object Deformation mode: an object is locally deformed.

Through the experiment described in chapter 3, the hand motion can be decomposed using temporal (i.e. phase) analysis into global and local motions. Effective number of DoF for the hand motion on a surface was evaluated to 4 to 6 DoF. Note that 4 DoF exactly corresponds to the global part of the motion. Therefore, in this chapter, we give new evidences that this decomposition is among the best decompositions for the hand.

In this chapter, our main challenge is to discover the most natural mapping between 2D gestures and 3D manipulation for navigation and object positioning modes. As object deformation mode requires more attention than other modes, a complete chapter is dedicated to its studies (chapter 5). A first step is to discover which interactions are the most efficient to exploit the hand motion on tabletops. We first show that interactions with 3D content on tabletops is not natural (in the sense that no consensus among participants on how nontrivial 3D edition should be performed through 2D gestures). Yet, we are able to characterize a set of principles to design 3D interactions on tabletops. One of the challenges is to discover if implicit information included in the interaction could be used to automatically switch between interaction modes, rather than having to provide explicit widgets for mode selection.

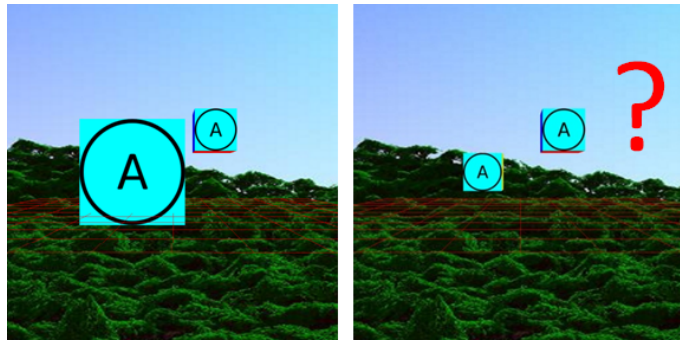
## 4.2 USER-STUDY

To get a better understanding of spontaneous hand gestures on a multitouch table, in order to define a possible mapping between 2D gestures and 3D tasks, we ran a second experiment, which involves fake 3D tasks.

### 4.2.1 Experiment

Participants were asked to first observe an animation of the desired task on the left part of the screen (fig. 4.1.left). Then, they had to suggest a gesture on the right part of the screen to control this specific task (fig. 4.1.right). Except some visual feedback given in the 3D scene (such as a grid), no help was given to participants. Their gestures were recorded to analyze them off-line.

The experiment was composed of twenty trials, divided into three classes: eleven navigation tasks, nine object positioning tasks. The scene was composed of two cubes, a red grid and a background picture (fig. 4.1).



**Figure 4.1:** *Experiment: an example of user-study trials. An animation is shown on one screen (left), while users perform gesture on second screen (right)*

### 4.2.2 Apparatus

This experiment was conducted on a 22" 3M multi-touch display <sup>1</sup> (473×296 mm, 1680×1050 pixels, 60 Hz, 90 DPI). The software was based on the Qt <sup>2</sup> and Ogre3D <sup>3</sup> libraries.

### 4.2.3 Participants

31 participants, composed of 8 women and 23 men, were tested. Average age was 30 (min. 22, max 49). All participants had normal or corrected to normal vision. For left-handed participants, the experiment was mirrored. Participant's background was variable, and not only computer scientist background. Participants' experience with 3D applications, and tactile devices was variable, but this was not an issue, as the goal of the experiment was to get some understanding of fundamental physical behavior.

## 4.3 INTERPRETING STORED GESTURES

Each performed gesture on each trial was stored and analyzed off-line. Results are shown in table 4.1.

### 4.3.1 Hand Phase Analysis

Participant's gestures are analyzed using the phase-based method presented in section 3.2.2. We however needed to slightly adapt:

- First, because participants sometimes used both hands, we had to perform two distinct phase analysis (one for each hand).
- Second, we had to adapt the hand parameterization to the number of fingers in the contact with the screen, as all interactions did not always involve the five fingers. A main issue of this adaptation is that the thumb might not always be used. According to the previous experiment in chapter 3, the thumb is usually the most stable finger (this was our reason for using it as origin of the local frame). Therefore, we assumed that the thumb to be the

---

<sup>1</sup><http://www.3m.com/>

<sup>2</sup><http://qt-project.org/>

<sup>3</sup><http://www.ogre3d.org/>



Tasks	Generality		2 <sup>nd</sup> Hand Data			Phase Analysis			Best Gesture 1 <sup>st</sup> /2 <sup>nd</sup>
	Dist. to Obj.	Avg. #Fingers	% Tasks	2 <sup>nd</sup> Hand #Fingers	Type	#T. Phase	#R. Phase	#S. Phase	
<b>Navigation</b>									
Translation /xy	1.5	2.5	03	1.0	Sym.	1.2	0.1	2.0	Tr./-
Translation /z	1.9	2.9	46	1.7	Sym.	1.5	0.1	1.0	Tr./Sym.
Rotation /xy	2.3	2.8	52	1.4	Sup.	1.1	0.3	1.5	Tr./Sup.
Zoom	1.6	3.5	54	1.8	Sym.	1.5	0.1	1.2	Tr./Sym.
Zoom to Object	1.2	3.0	40	1.6	Sym.	1.5	0.2	1.1	Tr./Sym.
<b>Object Positioning</b>									
Translation /xy	0.2	1.2	00	-	-	1.0	0.1	0.3	Tr./-
Translation /z	0.5	2.1	24	1.3	Sup.	1.2	0.1	0.6	Tr./-
Rotation /z	0.5	2.3	00	-	-	0.7	1.1	1.9	Rot2./-
Rotation /xy	0.7	2.3	57	1.1	Sup.	0.9	0.1	0.9	Tr./Sup.
Scaling	0.4	3.0	19	2.5	Sym.	1.4	0.4	1.0	Sca2./-
<b>Object Selection</b>									
Selection	0.5	1.0	-	-	-	-	-	-	*cf 4.4.4

**Table 4.1:** Statistical results of the experiment

one that was moving the less (this assumption can be wrong for translation gestures, but this is not an issue: since all fingers are being moved the same way in this case).

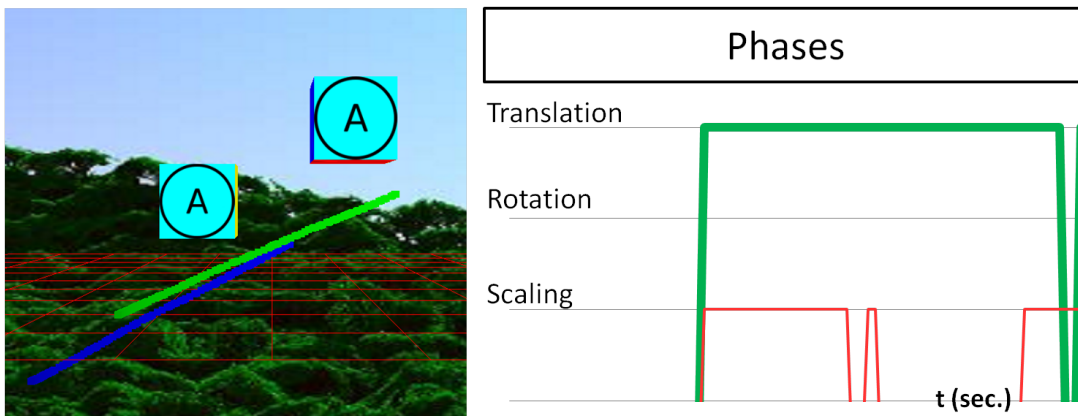
Remaining fingers follow the process described in section 3.2.2.

## 4.4 RESULTS OF THE USER-STUDY AND DISCUSSION

### 4.4.1 Hands/Fingers uses

To deeper investigate the efficient DoF a hand can control, we first observe that only three participants used more than 3 fingers a hand (up to 6 DoF a hand). Those cases mostly involved navigation tasks. In further details, when participants involved more than 3 fingers to manipulate 3D content, the principal phase of their interaction corresponds to a translation phase (i.e., the most stable global motion). On average, fewer fingers by hands are interacting to handle objects than to navigate (table 4.1). The difference between numbers can be explained by the use of the second hand. Further explanations are developed afterwards.

However, many participants interacted with both hands. From our observations, the NDH had two main functions: a support function (Sup.) (e.g., frequently indicating the parts of the scene that should not move, by keeping a still hand on them); or a symmetric function (Sym.) (e.g., doing symmetric gestures with both hands for scaling). The support function is most frequently used, specifically on object manipulation tasks, where it is used to maintain some objects or some part of the object of interest in place.



**Figure 4.2:** Performed gesture during a navigation translation task (left) and its phases decomposition (right)

#### 4.4.2 Scaling Interferences

While resolving trials, participants performed scaling phases during their interactions (tab. 4.1, "#S phase" column). Yet, in many cases, the DoF involved by scaling was meaningless.

Fig. 4.2 illustrates a typical example: during a navigation task (a translation in the (x, y) plane, left part of the image), performed gestures were analyzed (right part of the image). In this illustration more than 90% of the motion was identified as translation phase (as expected), while short scaling phases occurred in parallel.

As stressed in the experiment described in chapter 3, the stable and useful part of scaling motions commonly takes place once translation and rotation phases of a motion have ended. Therefore, scaling phases should not be taken account when they occur concurrently to other phases, and gestures in fig. 4.2 should be interpreted as a bare translation.

#### 4.4.3 Modes Disambiguation

Obviously, the vast majority of users (87%) performed ambiguous gestures, i.e., used similar gestures for at least two different tasks. This leads us to look for ways to disambiguate gestures.

A first clue for disambiguation is the location of the fingers at the start of the gestures (as in contextual mode selection, cf. 2.4.3): the first finger is hardly put on or around an object during navigation tasks (distance > 1, table 4.1), for which interaction takes place the background image. Furthermore, the grid (in red in fig. 4.2, left) is sometimes manipulated to indirectly perform navigation tasks, such as panning along the depth axis. Conversely, object positioning typically start in or nearby the object (distance < 1). This criterions enables us to distinguish navigation from object manipulation tasks.

A second clue for disambiguation is the number of fingers used. The average number of fingers involved to navigate is typically 3, while this number decreased to 2 for object positioning. Though, the NDH gives the most relevant number of fingers: 1 finger used for navigation, no finger for object positioning. In a large proportion, the NDH fingers reached the border of the screen for navigation tasks when it has a support function.

Therefore, the selection modes could be automatically handled during user interaction by mixing these two criteria: first, the context could tell the interface to which object (or the whole scene) the interaction is to be applied, then a finger-count method [BML12] would give the selected interface mode.

#### 4.4.4 Group Selection

Another issue investigated is how the same transformation could be applied to a set of objects. For this reason, some trials were asking participants to simultaneously perform the same transformation on two objects.

Commonly, the same gesture was performed on both object ( $\approx 75\%$  of users), each object involving one hand. But this interaction technique is not a valid option: this technique does not scale to more than two objects, and cannot be applied to gestures involving both hands.

Instead of simultaneously/sequentially manipulating the different 3D objects, fewer participants ( $\approx 20\%$ ) preferred to first select the object by clicking (or double clicking) before manipulation. Only two users performed a "lasso" gesture to select objects before any manipulation. After the object selection, the gesture was performed either on one of the object, or near the barycenter of the group. This leads us to conclude that a specific visual effect, such as a widget, should be created to represent the selected set.

#### 4.4.5 Navigation: Zoom Task vs. Depth Translation Task

During the experiment, depth axis translation and zooming tasks were asked for in two consecutive trials. Although a background image was added to the 3D scene in order to distinguish between these two gestures, all users but two asked for the difference. Once answered, they usually succeeded to understand the transformation shown.

Moreover, we can also notice that, although they did know the difference (as they asked for it), half of the participants still performed the same gestures for both tasks. Our conclusion is that one of these two transformations might be useless in 3D editing interfaces: zooming in/out a scene is often an artefact effect that is unreachable without electronic devices such as cameras.

#### 4.4.6 Combining Interactions

Tasks	% Sequential motions	% Concurrent motions
Translation + z/Rotation	58%	42%
Translation + xy/Rotation	79%	21%
Translation + Scaling	61%	39%






**Table 4.2:** *User preferences about separating (left) or not (right) the different motions*

Some trials consisted in combining elementary motions - for instance, an object translation task with a rotation task. Not to influence participants, leaving them free to invent their own interaction mode, only a before/after screenshot was shown in these cases. Analyzing data using our phase analysis method enabled us to easily distinguish whether users prefer to perform

each "elementary" motion sequentially, or simultaneously. Results are gathered on table 4.2.

In two third of the cases, participants preferred to decompose gestures into "elementary" ones, which is consistent with Martinet et al. work [MCG12]. In more details, performing a translation and a depth axis rotation (i.e., z/Rotation) are mainly decomposed into translation and then rotation phases. The higher number of participants simultaneously performing these two phases (42%) is consistent to Wang and Nacenta work [WMSB98, NBBW09], as these two phases slightly interfere with each other. Conversely, when a translation need to be coupled with a rotation along another axis (xy/Rotation), the phase analysis mainly identified two translation phases, where the second one corresponded to the second hand gesture (a trackball like rotation, cf. 4.4.7).

#### 4.4.7 Phase Analysis and Noticeable Gestural Design Pattern

Gestures	Translation	Rotation 1	Rotation2	Scaling 1	Scaling 2*
Phase:	Translation	Rotation	Translation + Rotation	Scaling	Translation. + Scaling
Type:					

**Table 4.3:** Five typical gestures for one hand interaction, identified through our experiment. Due to scaling interferences while translating (cf. 4.4.2), scaling 2 is more difficult to identify.

As previously observed, a majority of users performed ambiguous gestures. Therefore, the interface to be designed will require disambiguation between modes. Clues to handle this mode selection are twofolds: contextual disambiguation (such as finger localization on objects versus on the background), and the number of interacting fingers. Conversely, we note that some interactions can be linked together, enabling us to identify typical gestures, and a gestural pattern within each mode.

Once phases are analyzed, global hand motions on a surface can be classified into 9 gestures. However, by taking scaling interferences into account, this set of gestures can easily be reduced to 6: no motion, translation, pure rotation, pure scaling, translation and rotation, translation and scaling (this last gesture has to be kept for only one case: pinch gestures). The relevant gestures are illustrated in fig. 4.3. In further words, when all three phases of the global decomposition are detected, the associate gesture is only Rotation 2 gesture (and not RST kind of transformation).

Although the local part of phase decomposition was analyzed, no relevant information was extracted from it. On the other hand, all the parameters inferred by the global phase analysis were used. This observation is an additional proof that the effective number of DoF for hand motion is at least 4 DoF.

All these observations enabled us to associate each task of the user-study to the corresponding typical gestures for both hands (tab. 4.1, last column). A noticeable pattern from these

results associations, as summarized in table 4.4. Incidentally, we observe a strong caesura between transformations within the screen plane, and others (3D transformations):

- Interactions that transform the scene/object on the 2D screen plane mainly use one-handed gestures (e.g., translation/extrusion along (x, y) axis tasks are performed by one hand translation gesture). Scaling interactions can be gathered into two possible gestures, which both represent a shrink gesture, either performed by one or two hands.
- Interactions that require depth axis motions need more attention. For instance, rotation tasks are usually performed with two hands: one hand is keeping the object in place (support), while the second hand is "pushing" the object, like in the trackball technique [CMS88].

Unfortunately, translation along depth axis interactions are outsiders: no consistent gesture could be identified to perform this task.

Using such a gestural pattern for 3D multi-touch interfaces would bring a real advantage: since the average user *naturally* performs these gesture, an interface using such a pattern would accelerate its learning. More, reproducing a similar pattern to any additional mode reduce the time spent to learn gestures.

Action	Translation / xy	Translation / z	Rotation / z	Rotation / xy	Scaling / Zoom
Gestures (Phases)	Translation	?	Rotation	Translation + Support	1 or 2 handed Shrink gesture.

**Table 4.4:** Table grouping a set of actions and the gesture associated by participants. Note that this pattern was identified in two different modes: navigation and object positioning.

## 4.5 CONCLUSION

Through this experiment, users were free to invent 2D gestures to interact with 3D content. Quite interestingly, they tend to interact using the global hand motions only (global translation, rotation or scaling), and to use more than the necessary fingers, even if two or three fingers would be sufficient. For more complex interaction gestures, users naturally limit themselves from one to three fingers per hand, but still use global hand motion. Therefore, global phase analysis need to be used for mapping gestures and tasks: gestures are easily classified. Even more, a design gestural pattern for 3D content manipulation emerged, which is reproduced inside each tested mode, and could be extended to any other 3D content transformation mode. Though, due to interferences with other phases, scaling phases should be identified only when they occur alone.

As noted in our experiment, the location where the gesture starts is often meaningful: users typically use it to select the object to which the action is applied. In addition to control object selection, the hand location at the start of the gesture is a relevant method of automatically selecting between navigation (if gesture starts outside any objects) and object positioning. However, as a limitation arises for crowded scene (i.e., when the user does not have enough room to start a gesture without overlapping some objects), this scheme needs some adaptation.

Before discussing an implementation of an interactive system for navigation and object positioning tasks, which will be done in chapter 6, we perform a similar experiment, fully dedicated to 3D deformation tasks on tabletop. Chapter 5 details this experiment and proposed an extension of the gestural pattern we just identified to include deformation tasks.



CHAPTER

— 5 —

UNDERSTANDING DEFORMATION  
ON A SURFACE



## 5.1 INTRODUCTION

3D shape editing systems are essential for a large number of applications. Although many techniques were developed to simply deform object on tabletop (such as in [IMH05]), from the interaction point of view, recent research contributions simplify the 3D camera/objects positioning interactions. However, free-form editing interactions of 3D objects on tabletop still lacks of simplification, especially when mixed with other tasks such as navigation or object positioning.

Previous investigations already bring some propositions to mix deformation interactions with other interactions: a gestural design pattern, involving the global motion and the number of interacting hands, is previously recommended. However, this pattern is first not fulfilled, and second lacks robustness. Indeed, when the size of the deformed part is larger than the hand size, tasks disambiguation cannot be based on hand registration only.

Therefore, further investigation to fulfill our knowledge about deformation tasks on touch screens is required. Based on a relevant set of free-form deformation tasks (for 3D shapes), we report on a user-study, enabling us to discover fundamental user behavior defined by four principles, when performing standard shape deformation tasks on tabletop (such as bending, twisting, adding or deleting material).

## 5.2 INTERACTION PRINCIPLES FOR DEFORMATION TASKS

This section details four principles about user' interactions to select and deform part of an object. The focus point we study deals with the region of interest delimitation (i.e., which part is transformed or not), and the interaction disambiguation between deformation tasks, especially between planar and non-planar deformation tasks.

Planar tasks correspond to tasks that map the screen plane to itself when editing 3D object (e.g. x-/y-translation, rotation around depth-axis, or global scaling). In contrast, non-planar tasks correspond to all other 3D editing tasks. (for instance, twisting tasks). Because of the dimension gap issue (between input/output dimensions), the interaction classifications as one of these categories are useful.

### 5.2.1 Delimitating Region of Interest

A first issue before deforming a shape is to specify the modified part of the object (vs. the unmodified part). To handle this delimitation, two main approaches is relevant: the deformed region could either be pre-selected (for instance, by circling it) or kept blocked under some support fingers throughout the deformations.

Despite the former approach frees hands, it also involves an extra selection gesture that users will need to learn. We believe that this choice is less natural, especially in direct manipulation, as it would require more learning time. Therefore, our first studied principles is:

**P1: The use of support finger(s) is compulsory for delimitating the deformed vs. undeformed parts of the object. These support finger(s) are applied first, and kept still during the deformation.**

Although we believe in the use of support fingers, this technique also brings an additional issue: designers can only disambiguate regions with a limited number of interacting fingers. Consequently, they have to smartly locate these support fingers.

Hopefully, humans are conditioned to intuitively imagine a virtual skeleton inside any 3D shape. The latter is similar to the medial axis of the shape [LCLJ11], defined as the locus of the centers of maximal balls inside the shape. Using this notion of skeleton, one can easily delimitate any region of interest within an object using very few points. This leads us to our next principles:

**P2: Interacting fingers on the object begin their interaction on the medial axis (or very close to it).**

### 5.2.2 Disambiguating Planar/Non-Planar Deformation Tasks

On tabletop, two kinds of 3D deformations should be distinguished: planar deformations versus non-planar deformations. We already discussed about a method (cf. 4.4.7), especially for docking (i.e. object positioning) and scaling: the distinction between 2D/3D transformations relies on the number of detected hands - one-(resp. two-) handed interactions were mapped to planar (resp. non-planar) interactions. However, when only a tabletop is used, identifying the number of hands from a small set of interacting fingers can be difficult. More, this disambiguation is less true for deformation, especially when the size of the object part to be deformed is larger than the user hand size.

We rather rely on the following remark: during any deformation, one hand typically remains on the deformed object, to control its position (similar to support fingers). In case of planar deformations, all interacting fingers (moving include) is located on the object, as starting and ending points are both on the same screen plane. In non-planar cases, this is not true: controlling the deformation from the end-point of a part of the object might be impossible (the point could be occluded by the object itself). This leads us to P3 and P4:

**P3: For planar deformations, all interacting fingers are kept inside the manipulated object.**

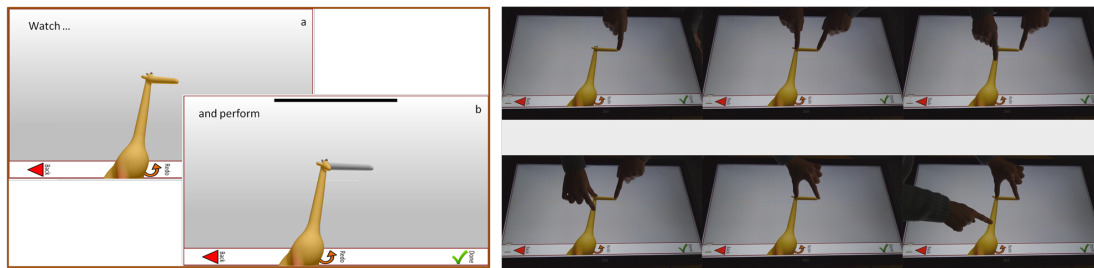
**P4: For non-planar deformations, some interacting fingers have to be outside the manipulated object.**

## 5.3 USER-STUDY

Once principles written, we developed a user-study to verify them. Divided into two experiments, this user-study focus on deformation tasks only, and is defined as follow:

### 5.3.1 Experiment 1

The first experiment as a *watch-then-perform* experiment: participants first observe an animation of the desired deformation, and then, they perform a gesture of their choice to handle the deformation (fig. 5.1, left). While performing, the desired result is shown in gray.



**Figure 5.1:** *Experiment 1 (left): participants watch an animation (a); and then they performed a gesture which would handle the animation (b). Experiment 2 (right): participants ranked videos from the most to the less logical ones.*

### 5.3.2 Experiment 2

The second experiment was composed of videos illustrating six different gestures performed synchronously by an actual user (fig. 5.1, right). Once videos observed, participant has to rank the six videos from the most appropriate gesture to the less appropriate one - by eventually eliminating the unmeaning gestures. Meanwhile, they have to explain the reason for their choices.

### 5.3.3 Same Studied Set

The studied set was exactly the same in both experiments, divided into four categories. It was composed of five adding, three deleting, five bending and two twisting deformation tasks.

Using the same of deformation in both experiments is relevant, especially in this order. Indeed, the first one aimed at getting a first gesture from participants, which we expect to be closest from a "natural" one. In this order, the participant is not biased by suggested options. In reverse, the second experiment allowed participants to take time to think about the different possibilities, and about the deformation they would expect from these gestures. Moreover, in some cases, participants had no idea of which gesture to perform, thus the second experiment gave the possibility to choose what is the most logical. More, it allows us to answer the following questions: do participants confirm their first move (are they consistent to the gesture they performed when ranking videos)? Were they sure of their choice while performing gesture in the first experiment?

### 5.3.4 Apparatus

This user-study was conducted on a 32" 3M multi-touch display<sup>1</sup> (698 × 393 mm, 1920 × 1080 pixels, 120 Hz, 70 DPI). The software was based on the Qt<sup>2</sup> and Ogre3D<sup>3</sup> libraries.

### 5.3.5 Participants

30 participants, composed of 11 womens and 19 men, were tested. Average age was 32 (min. 23, max 54). All participants had normal or corrected to normal vision. Participant's background was variable, and not only computer scientist background. Participants' experience

<sup>1</sup><http://www.3m.com/>

<sup>2</sup><http://qt-project.org/>

<sup>3</sup><http://www.ogre3d.org/>

Deformation Tasks	# Support Finger			# Moving finger			# Outside Finger	Preferred Phase	Preferred Video	Dist. (Pixels) Med. Axis
	#0	#1	#+	#0	#1	#+				
<b>Adding</b>										
Along the object	8	21	1	0	30	0	~0	Scaling	2f Scaling	6.58
Object Extrema	18	11	1	0	30	0	0.08	Translat.	2f Scaling	6.02
Ortho. to object	13	13	4	0	29	1	0.28	Scaling	2f Scaling	16.96
<b>Deleting</b>										
Along the object	5	24	1	0	29	1	~0	Scaling	2f Scaling	9.03
Object Extrema	1	27	2	0	29	1	0.18	Scaling	2f Scaling	6.70
<b>Bending</b>										
Pure Rotation	0	26	4	0	29	1	~0	Rotation	2f Rotation	5.33
Bending curve	3	20	7	0	17	13	~0	Rotation	2f Rotation	7.66
<b>Twisting</b>										
Twist along XY Axis	0	15	15	0	13	17	0.72	sup. + rot.	2f+2f Support + Rotation	8.19
Twist along depth axis	3	16	11	0	16	14	0.64	sup. + rot.	1f+2f Support + Rotation	*

**Table 5.1:** *User-study results*

with 3D applications, and tactile devices was variable, but this was not an issue, as the goal of the experiment was to get some understanding of how they behaved in front of tasks that do not exist yet on such device.

## 5.4 RESULTS

To further investigate our knowledge about possible interaction(s) for deformations on tabletop, performed gesture on each trials of the first experiment is stored and analyzed off-line. Results are summarized in table 5.1.

### 5.4.1 Use of Support Finger (P1)

Our first studied principle was the use (or not) of support fingers to delimit the modified parts of objects. Let us analyze results concerning this point (summarized in "*# Support Finger*" columns in table 5.1).

First, no participant specified the deformed region beforehand with a specific gesture (such as circling). They either defined the region of interest with support fingers (#1 and #+ columns), or considered the situation constrained enough to implicitly defined regions.

For bending or twisting cases, nearly all participants ( $\approx 95\%$  of trials) maintained as least one support finger on the non-deformed part of the object. The main reason that users claimed was to visually define the bounds between modified/unmodified regions of the object. Therefore, support fingers have a specific meaning, such as the pivot position of the rotation. For more complex tasks (for instance, twisting or bending curve deformations), more participants

even use an extra support finger to concretely define and keep the unmodified region in place.

However, this principle P1 was not always verified for the adding/deleting deformation tasks. Only six participants ( $\approx 20\%$  of users) made use of support finger(s) in all cases, claiming that those manipulations would lead to under-constrained problems without it. Interestingly, other participants gave three main reasons for not using support fingers for these tasks.

The first reason, given by half participants, involves the deformation localization: when designer has to deform one extremity of the model, the deformation is blocked by the larger part of the model the extremity is connected to. Therefore, there is no need of any support finger. This observation explains the less number of participants handling adding material deformation along an ending branch with support finger(s).

The second reason (20% of participants) for not using support finger involves tasks: to distinguish adding from deleting deformation tasks. From users' point of view, deformation tasks need an end point to specify until where deformation is performed, defined by the support finger(s). Similar to bending and twisting, support fingers have a specific (and geometrical) meaning. This reason explains the higher number of participants that need support fingers for deleting deformations.

The last reason (10% of participants) is the idea that implicit constraints were enforced using the nature of the object (because of its semantic, its color, its size and so on).

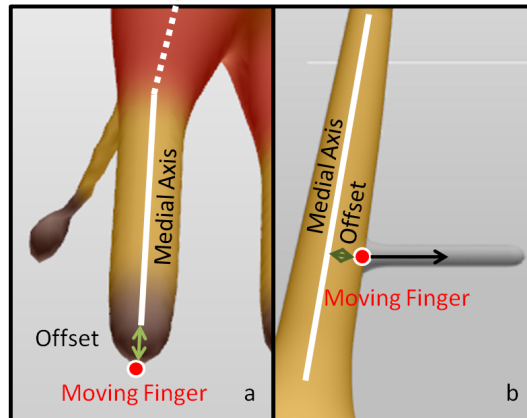
Quite surprisingly, participants' mind changed a lot during the second experiment. Indeed, for these tasks, 93% of the participants defined the scaling gesture with one support finger (named "2f Scaling" in table 5.1) as the best gesture, sharing the opinion about the under-constrained problems. Conversely, most participants disliked and even rejected the use of extra support fingers as shown in some video examples, claiming that meaningless support fingers should be avoided to simplify the interface.

#### 5.4.2 Placement of Fingers (P2)

As illustrated in table 5.1, the number of interacting fingers to control deformations is not high (usually two or three fingers are enough). Therefore, the placement of these few fingers is extremely relevant for understanding the interaction. Again, the two experiments give distinct results.

From experiment 1, the average distance (in pixels) between interacting fingers and the skeleton of the object is summarized in the last column of the table 5.1. Despite the medial axis was never shown during trials, users typically located their fingers on it (with an average 6-7 pixels distance ( $\approx 2$ -2.5mm), for objects of about 60 pixels wide ( $\approx 20$ mm)).

Two exceptions are noticeable. For the third deleting deformation case (fig. 5.2.a), users were asked to erase a whole part of an extremity of the object (one hoof of the girafe). As users wished to grab the point where deleting deformation started (i.e., around the object surface), while the skeleton stops before the extremity, obviously, this trial artificially increases the distance to the medial axis, and should be disregarded. For adding material perpendicular to the object case (fig. 5.2.b), the deformation requires an additional parameter: the thickness of the newly created branch. We observed that the smaller the amount of the material is, the closer to the border the moving finger starts. This observation validates the interaction described in Delamé et al. [DLCB11] works (in their work, the thickness of added material for a 2D shape



**Figure 5.2:** The two exceptions when users does not place the moving finger on the medial axis: a) deleting deformation case; b) adding deformation case

involved the distance to the border).

Despite the first experiment validates principle P2, this is more questionable once second experiment is considered. For support finger(s), the preferred finger positions were always on the medial axis. However, the position of moving fingers seemed to be less relevant: while most participants (93%) rejected planar deformations with fingers starting outside the object, half participants considered all other positions on the relevant part of the object (on the medial axis or not) as similar.

### 5.4.3 Distinguishing Planar/Non-Planar Tasks (P3 and P4)

P3 principle was easily verified through experiments: for planar deformations, nearly all participants put all fingers on the object (or on its border) to interact (see table 5.1, "# Outside Finger" column). Moreover, only two of them did not reject gestures with moving fingers starting outside the objects during experiment 2.

In contrast, P4 is more complex. Participants were puzzled by non-planar deformations, and had difficulties to define a reliable gesture for them, such that several participants could not come up with a gesture at all. This is obviously due to the dimension gap issue between 2D inputs and 3D outputs. To resolve this issue, most of participants (73%) relied on an increasing number of interacting fingers. For instance, to twist the giraffe neck, support fingers were often located around the bottom of the neck, while up to five fingers were interacting and twisting the head around its axis. Usually, one (or more) finger was starting the interaction outside the object.




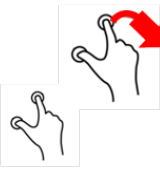
Hopefully, the second experiment helped users to make up their mind about non-planar deformations. Although users were surprised by gestures with fingers outside the object, in most cases (63%), participants picked them out as the best gestures. They noted that such gestures referred to an abstract representation of the deformation: while support fingers define the geometrical specifications for the deformation (such as the twisting axis), the gesture of the moving finger(s) involve the deformation.

### 5.4.4 Additional Results

As we expected, the number of interacting hands is not such a relevant information to manipulate object. This was validated by the fact that most participants classified two-fingers gestures performed with either one or two hands as equivalent. even though two-handed gestures enable wider amplitudes.

From this observation, and to further analyze stored gestures, we performed a phase-based analysis on them, with one adaptation: for two-fingers gestures (whatever the number of interacting hands), we performed an analysis as if the whole motion was considered as performed by one hand only. Results are summarized in table 5.1, "Preferred Phase" column. Similarly, performed gestures on videos were analyzed, and the preferred one by participants were summarized in "Preferred Video" column.

Once phase-based analyzed, categories that we defined for the studied set are relevant: the user gestures are similar in each of these categories, the only exception being adding material at an extremity (cf. 5.4.1). This analysis enables us to identify a gestural pattern for deformation tasks (fig. 5.3). Considering that a support finger is necessary (due to P1), this gestural design pattern is roughly consistent with previously defined pattern (cf. 4.4.7).

	<b>Opening Scaling</b>	<b>Closing Scaling</b>	<b>Rotate</b>	<b>2Hand Rotate</b>
<b>Mapped Gestures</b>				
<b>Deformation Tasks</b>	Local Adding	Local Deleting	Local Rotation	Local Twist

**Figure 5.3:** Gestural design pattern for deformation tasks

## 5.5 DISCUSSION

This user-study brings us many interesting features to deform 3D objects on tactile screens. The principles investigation leads us to develop a quick-learning interface, that would efficiently handle elementary deformation tools, such as adding/deleting material, bending or twisting.

An interesting first observation is given by P2 principle: even though P2 is close to the fundamental behavior (users unconsciously select the medial axis), the interpretation of gestures from users are more based on the geometrical properties of the object, rather than on the location of the interacting fingers.

On the one hand, support fingers do not only delimitate regions of interest (which regions are deformed or not), they also define geometrical specifications. For instance, one support finger defines the rotation center for bending tasks. Similarly, two support fingers define the axis around which the twist manipulation is performed.

For adding/deleting tasks, the support finger function seems to be more a starting/ending point of the performed deformation. For these tasks however, the meaning of support finger is not always clear - especially for object extremities: the adding (resp. deleting) material task could start (resp. finish) to where the finger first (resp. finally) interact with the object. But, this distinction would lead for users to spend more time learning gestures, and these users would also hardly understand why such distinction is kept.

On the other hand, the other fingers have to clearly define the transformation (or at least control the deformation parameters), either by a direct manipulation (for planar deformations) or by an abstract representation of the deformation (for non-planar deformations). For instance, bending part of the object starts and finishes with the finger (which is bounded to the surface representation), whereas, angular parameters of the twisting operation is controlled by the fingers outside the object.

Another interesting observation induces by the user-study is about hand roles. When there are more than one meaningful parts of the object to specify with support fingers (for non-planar deformations), participants always interacted with two hands for these trials. Conversely, planar deformations might be deformed with either one or two hands. However, when the gesture is considered as a whole, the motion is similar to one-hand two-finger gestures. This is consistent with our previous observation to distinguish planar and non-planar deformations (cf. 4.4.7), with little adaptation for two-fingers-only interactions.

The main goal of this user-study, combined with the two previous ones, was to fulfill our understanding of hand gestures on a surface while manipulating a virtual 3D scene. From this comprehension, we are able to propose an interface (described in the next chapter) that successfully combines the three main modes for editing 3D space - navigation, object positioning and object deformation - with important features:

- An easy-to-use interface, by dodging difficult gestures for the average user (even beginners).
- A quick-learning interface, thanks to an easy-to-learn mode selection and gesture similarities between modes.
- A contextual switch between modes, which reduces the time needed for editing, especially for recurring tasks.





CHAPTER

6

FREE-FORM EDITING OF 3D  
SHAPES

## 6.1 INTRODUCTION

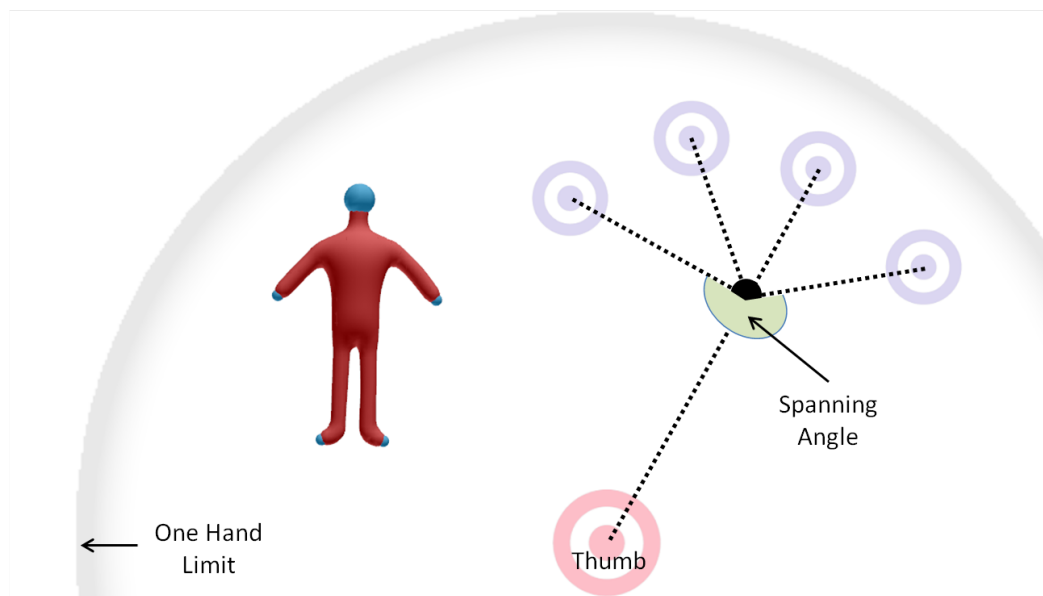
Previous user-studies yields to a clear classification of elementary task to edit 3D scenes, which would not be effective without three main nodes: navigation, object positioning and object deformation. For instance, a designer cannot deform an object without changing viewpoint since the part to be deformed might be occluded.

Therefore, this chapter describes a developed interface that follows three main goals: an easy-to-use interface, a quick learning interface, which would also efficiently handle the mode selection.

In this chapter, our main interest to edit 3D object is the interaction point of view. Therefore, we desire to develop an interface that may manipulate as-generic-as possible objects. Despite navigation and object positioning modes involve no specific property for object, the second studied principle for deformation referred to the medial axis (where designer intuitively locate their fingers, especially support fingers). Therefore, manipulated objects are restricted to any object that owns a skeleton for animation (which is close to medial axis) - for instance, any surfaces that might be deformed by a skinning method. In our case, we chose to represent 3D objects using skeleton-based implicit surfaces (precisely, the scale-invariant integral surfaces introduced by Zanny et al. [ZBQC13]).

To successfully develop this interface, we first describe how raw inputs are analyzed through an on-line phase-based analysis. Then, we deal with the interaction core: the algorithm to select modes and tasks. Finally, some results of this interface are discussed.

## 6.2 ON-LINE PHASE-BASED ANALYSIS



**Figure 6.1:** A typical screenshot of the interface once fingers are put on the screen (Barycenter circle and dot lines are not shown to the user).

To edit 3D scenes from multi-touch inputs, the interface first need to identify and classify hand gestures. As discovered by previous user-studies, the average user fundamentally performs global motion gestures to handle tasks. Therefore, we limit the hand decomposition to global parameters: Translations (T), Rotation (R), or Scaling (S). In the next sections, we describe how we extend the phase-based analysis into an on-line method.

### 6.2.1 Thumb Registration

The introduced phase based analysis requires a specific parameterization, where the thumb is the origin of the frame. Therefore, the thumbs need an efficient method to be quickly registered. Previously, the thumb (of each hand) was identified as the most stable finger (i.e. the one moving the slowest). However, this method implied off-line analysis, which is not possible.

To get a fast registration of the thumb, we rely on Au et al. method [?]: by comparing the relative spanning angles (i.e., the sum of the two angles of each side of the connecting line between the finger and the fingers' barycenter) between interacting fingers (fig. 6.1). The thumb spanning angle is usually the biggest one. Additionally, this technique is also able to detect left hand from right hand.

Limitations of this technique are twice: firstly, the number of interacting fingers (for one hand) should be at least three fingers. If less, we consider as thumb the first interacting finger. Secondly, the hand shape should be close to a "natural" pose. But this is not an issue: thumb registration is only required once, when all interacting fingers are put on the screen and before any motion. Therefore, the hand shape is commonly in a "natural" pose. More, once detected, the thumb is shown with a bigger, red circle (fig. 6.1).

### 6.2.2 Hand Registration

Thumb registration is not the only issue this on-line phase-based analysis has to resolve. To successfully analyze gestures on tabletop, the interface first need to distinguish the user's hands, without any additional device.

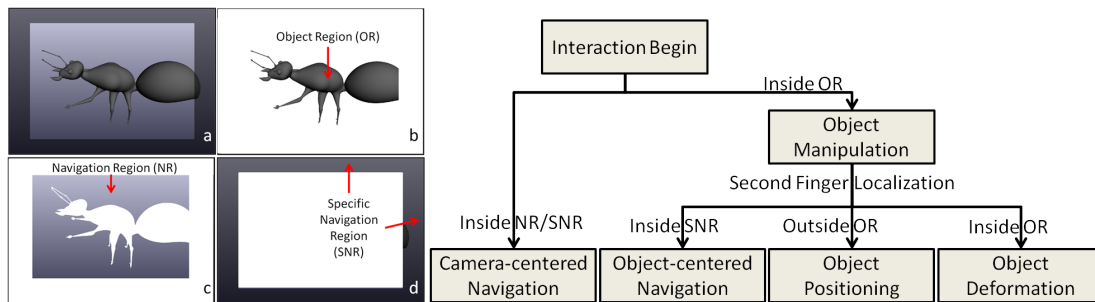
Once mode is selected (cf. 6.3.1), our method relies on distance: as long as only one hand is registered the distance between any new finger and the finger registered as a thumb is considered in order to associate the new finger to either the same, or a different hand. A visual feedback is shown to help user (fig. 6.1). When both hands are already registered, every new finger is linked to the nearest hand (i.e., whose thumb finger on screen is the nearest).

## 6.3 MODE/TASK SELECTION

A main issue the interface has to resolve is to combine navigation, object manipulation and object deformation in a quick-learning interface. At every time, users want to easily change viewpoint or viewing direction. Users might either look at the 3D scene globally (as they naturally do when moving or turning their head in the real world), or detail a specific object, by typically turning around for instance [MMCR13].

To handle task selection, the interface performs their gesture analysis in two steps: first, the interface selects the current mode the user desire to manipulate. This mode selection only relies on the context of interacting fingers. Then, in each mode, the selected task relies on the phase analysis outputs - in other words, on the typical gestures the user performs.

### 6.3.1 Mode Selection: Contextual Selection



**Figure 6.2:** (Left) Screen regions in 3D scene: a) user view, b) object region, c) navigation region, d) specific navigation region. (Right) Mode Selection hierarchy, depending on the first two interacting fingers

The mode selection (navigation, object positioning or object deformation) only relies on the context of interacting fingers, especially the first two interacting fingers (fig. 6.2, right). To handle the selection, three main regions have been defined (fig. 6.2, left): Object Region (OR), Navigation Region (NR) and Specific Navigation Region (SNR). The SNR is a specific dedicated region for navigation, mainly required when the screen is fulfilled, when NR is not reachable.

Any gesture beginning inside NR or SNR is mapped to camera-centered navigation. In contrast, any gesture beginning inside OR obviously has to manipulate object. Further mode selection is required in the latter case, which relies on the second interacting finger. When put on the SNR, the user can turn around the object, or zoom in/out it (Object-centered Navigation). If the finger is put outside the dedicated OR, the user can position the object (Object Positioning). Finally, Object Deformation mode is selected when both fingers is inside the same OR.

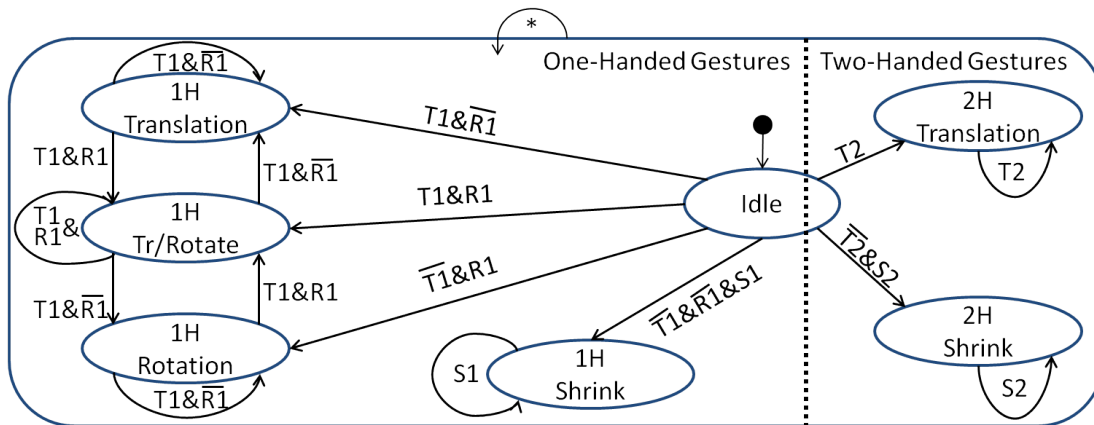
These region definitions and their mapping to modes are consistent with the user-studies performed. Indeed, starting finger localizations is a first clue to disambiguate (camera-centered) navigation from object manipulation, and also describes that for navigations, fingers often reached the border - or SNR (cf. 4.4.3). For object deformations, user has to define multi-touch interaction part on the object, with at least one support finger (cf. 5.4), inferring the object deformation way of selection.

### 6.3.2 Task Selection: Global Phase Analysis

Modes only gathered a set of tasks that user might use. Therefore, a second step is compulsory to select tasks. This is handled by interpreting hand gestures.

This interpretation relies on a generic state-machine that is similar to all reachable modes (6.3). Here, the states correspond to generic gestures (for instance, '2H Translation' defines a translation of the second hand, while the first hand stands as support), while events generated by the on-line phase analysis (T1 for first hand translation phase, R2 for second hand rotation phase, and so on, cf. 6.2) trigger the transitions between states.

Conform to previous user-studies, gestures that handle translation and rotation are grouped (cf. 4.4.6), while shrink gestures are isolated (due to scaling interferences, cf. 4.4.2). Similarly,



**Figure 6.3:** *Generic state-machine. Event are phase analysis outputs ( $T1$ ,  $R1$ ,  $S1$  for first hand,  $T2$ ,  $R2$ ,  $S2$  for second hand), while states are reachable gestures*

one-handed vs. two-handed gestures are also isolated.

The remaining step is the mapping between interpreted gestures and tasks. Fig. 6.5 and fig. 6.6 illustrate this mapping for each mode.

## 6.4 RESULTS AND DISCUSSION



**Figure 6.4:** *SeaScene created by only use Navigation and Positioning tasks*

Due to the mode/task selection, the interface can easily process any touch interaction. Snapshots of our interacting modeling system are depicted in fig. 6.5 and in fig. 6.6 and in fig. 6.4. The SeaScene was created by only using the navigation and positioning modes (without deformation), involving 20 different objects with 7 manipulation DOF each (3 translation, 3 rotation

and 1 global scaling). Our designer created the scene in less than 6 minutes using our interface, and noted that it would have taken her around half an hour with a standard 3D software.

Apart from the paper authors, our system was used by three users (two beginners and one professional computer artist) to perform a short comparison. With a few try-outs, all users understood how to interact with the system, confirming our choice of automatically selecting modes using contextual disambiguation.

Tasks	Beginners	Pro	Pro (Maya)
<b>Navigation</b> (4 Tasks)	1.6s	1.6s	1.8s
<b>Positioning</b> (5 Tasks)	3.2s	2.9s	3.5s
<b>Deformation</b>			
Adding Material	1.7s	1.7s	1.7s*
Bending	1.8s	1.7s	2.0s*
Twisting	2.8s	1.8s	2.5s *
<b>Combination</b>	20s	19s	25s

**Table 6.1:** *Time spent by external users to perform several tasks*

To further validate our 3D modeling system, we asked the professional artist to perform a set of tasks using her favorite commercial 3D modeling system (Maya<sup>1</sup>). Meanwhile, these three users were asked to perform the same set of task with our system. All users had first used our interface for nearly 20mn before this experiment. Then, the time they spent to perform tasks was recorded.

Although this test is only preliminary, results detailed in tab. 6.1 are promising: running time for simple manipulations are close to those of the professional artist, and even quicker for difficult manipulations (for instance, combining several simple tasks or deformation tasks). Note that deformation manipulation required some model set-up with standard softwares (such as tuning skinning parameters).

On the other hand, the developed interface only relies on the global part of the hand motion decomposition. From the user-studies feedback, if untrained, users prefer to rely on hand global motions. Therefore, to accelerate the learning of the interface, no difficult gesture implying local hand motion is proposed.

With the hand phase analysis, the interface recognizes three main phases each hand. For one-handed gestures, the interface is similar to any interfaces based on the RST method (cf. 2.5.1): every planar transformation (or 2D transformation) is handled by the interacting hand, performing a translation under a translation phase, a rotation under a rotation phase and a scaling under a scaling phase. The extension of the RST technique to 3D space transformations is handled by a second interacting hand.

Even though the second hand is dedicated to non-planar manipulations, the decomposition of the hand motion is also based on our temporal RST method: phase-based analysis. While one hand strictly defines region of interest and geometrical object properties (using only support

<sup>1</sup><http://www.autodesk.fr/products/maya/overview>















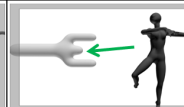
Mode Gesture	Starting 3D Scene	Camera-centered Navigation	Object-centered Navigation	Object Positioning	Object Deformation
1 Hand Translation 		 Scene Translation	Impossible with One-handed Gestures	 Object Translation	 Object Translation*
1 Hand Rotation 		 Scene Rotation		 Object Rotation	 Object Bending
1 Hand Shrink 		 Scene Depth Translation		 Object Scaling	 Adding/Deleting Material

Figure 6.5: Mapping between one-handed gestures and planar tasks











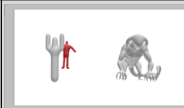
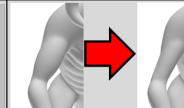
Mode Gesture	Starting 3D Scene	Camera-centered Navigation	Object-centered Navigation	Object Positioning	Object Deformation
2 Hand Translation 		 Camera-centered Rotation	 Object-centered Rotation	 Object Rotation	 Twist
2 Hand Shrink 		 Scene-centered Zoom	 Object-centered Zoom	 Object Depth Translation	 Local Growth

Figure 6.6: Mapping between one-handed gestures and non-planar deformation tasks

fingers), the second hand transform the object with an abstract representation of the desired transformation (such as pushing the object to rotate/twist it).

During user-studies, the "2-Hand Rotation" gesture was nearly never a relevant gesture for studied tasks. The exception was the twist deformation. But, to simplify the learning of the interface (no additional interaction to learn), we decided to select the second best choice from participants' opinion for this manipulation. Anyway, this unused gesture would be a relevant choice to extend the interface to any new elementary tasks: both user and interface already know the gesture.

## 6.5 CONCLUSION

This chapter describes a develop interface that successfully fills goals we fixed:



- An easy-to-use interface: the interface only relies on gestures that everyone can use (hand global motions). Even untrained, even beginners can handle such gestures (contrary to local motions).
- A quick-learning interface: by reproducing the set of gestures in every reachable modes, the user only have to learn the set of interactions one, and to combine it with the learning of mode selection.
- Mode Selection: thanks to a contextual switch, users reduces the time needed for (recurring) tasks, and so the time for modeling.

To develop this interface, we mainly rely on investigating principles from three user-studies, in order to fully understand the hand/user behavior to manipulate 3D shape on touch screens. One of the main principles is the dimension gap resolution: planar manipulations are handled by one-handed gestures, while non-planar gestures are handled by two-handed gestures. In both case, we rely on an extension of the RST techniques to register the hand motion.

In this chapter, we never describe the chosen method to manipulate (especially deform) object. Indeed, we do believe that the interaction would be exactly the same, whatever the chosen methods. The specification of the interface from the graphical field point of view is described in a later chapter.

CHAPTER

7

FROM GESTURES TO  
SKELETON-BASED IMPLICIT  
SURFACE DEFORMATION

## 7.1 INTRODUCTION

Previous chapters mainly investigated the principles to propose simple interaction for editing 3D objects, claiming that the proposed gestural design pattern can be used independently of the underlying graphical model. For instance, as the proposed method extends the RST technique, which is a standard method for many 2D interfaces using touch interaction, the pattern may be easily adapted to any docking interfaces using the RST method.

However, the proposed interface would not be complete without the details of how each edition task is performed on an actual graphical model. To further reach the goal of an easy-to-use interface, we focus on two important features to implement those tasks: consistency and regularities. The former one characterizes the motion of an object under a specific interacting finger: the selected object (or part of the object) has to remain under the finger. The latter, regularities, defines the tasks similarities while performing similar gestures.

This section details technical specificities of the interface. The first section 7.2 defines the different contents that compose a 3D scene in the interface, and the skeleton-based implicit surface chosen as underlying graphical model. Then the developed methods to select scene components and relevant geometric data are detailed in section 7.3. Efficiently selecting the component and relevant data for the handled task is a first challenge. While a majority of tasks only requires few data (such as the intersection point between the object surface and the finger), several tasks - especially from deformation mode - require more precise informations. Finally, the last section 7.4 precises the methods to perform the various existing tasks.

## 7.2 SCENE AND OBJECTS DETAILS

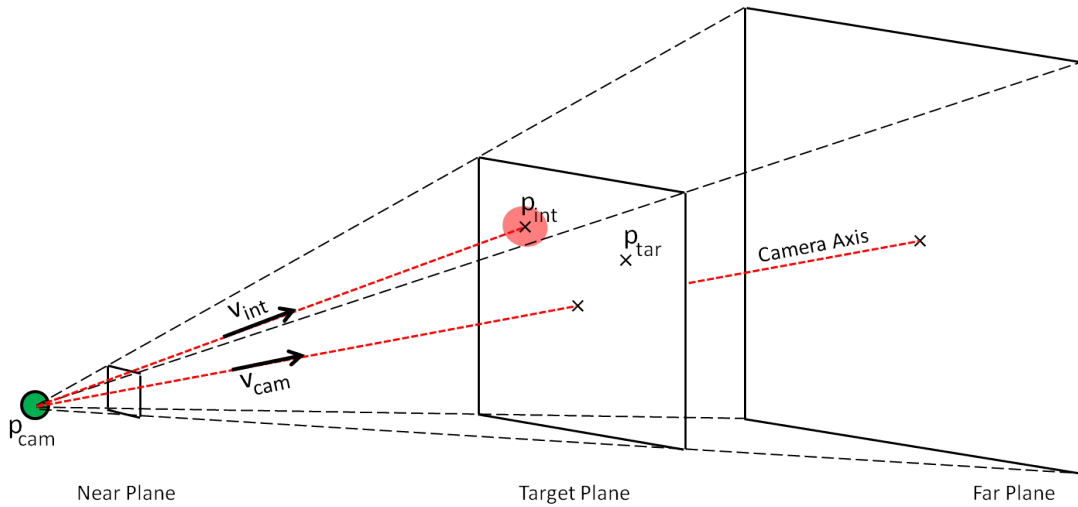
### 7.2.1 Scene Contents and Hierarchy

The scene represents the virtual space that users are able to interact with. Formally, the scene we handle is composed of components common in 3D graphics: lights, camera and objects. Although lights do not have specific features - they are omnidirectional spots - this section further details the other components.

**Camera:** as we are only interested in manipulating and editing 3D scene and objects, the only camera we have to care about is the main camera: the user point of view on the virtual space. Similar to the eye vision, camera vision shows the scene with a perspective projection vision (fig. 7.1): an object appears smaller with the distance to the camera.

In addition to camera attributes (the *camera position*  $p_{cam}$ , the normalized *camera direction*  $v_{cam}$ , the near/far plane and so on), we also define a relevant 3D point: the *camera target point*  $p_{tar}$ . This additional point is required to define several tasks. For instance, the rotation tasks have to turn around a specific center, which is defined by the camera target point. Note that the target point might not belong to the *camera axis* (characterized by the point  $p_{cam}$  and the vector  $v_{cam}$ ). Then, the target plane is defined using both the target point  $p_{tar}$  and the camera axis or direction  $v_{cam}$ .

Using the camera attributes, every interaction point (from mouse or touch interaction) defines an half straight line beginning at the camera position. Therefore, for each touch-interaction point, the interface can process relevant information using previously defined camera data (fig. 7.1): the *interaction ray*  $R_{int}$ , characterized by the camera position  $p_{cam}$  and the normalized *interaction vector*  $v_{int}$ . Then, the *interaction 3D point*  $p_{int}$  which belongs to the same



**Figure 7.1:** Camera in perspective projection view and relevant attributes for interaction

plane than the target point is defined as follow:

$$p_{int} = p_{cam} + k v_{int} \text{ with } k = \frac{(p_{tar} - p_{cam})v_{cam}}{\langle v_{cam}, v_{int} \rangle} \quad (7.1)$$

**Objects:** as they can represent any 3D shapes from a basic ball to a complex dragon, objects are the main elements of a 3D scene. Several methods to modelize them exist (section 2.1), but the proposed interface handles two representations: generic meshes and skeleton-based implicit surfaces.

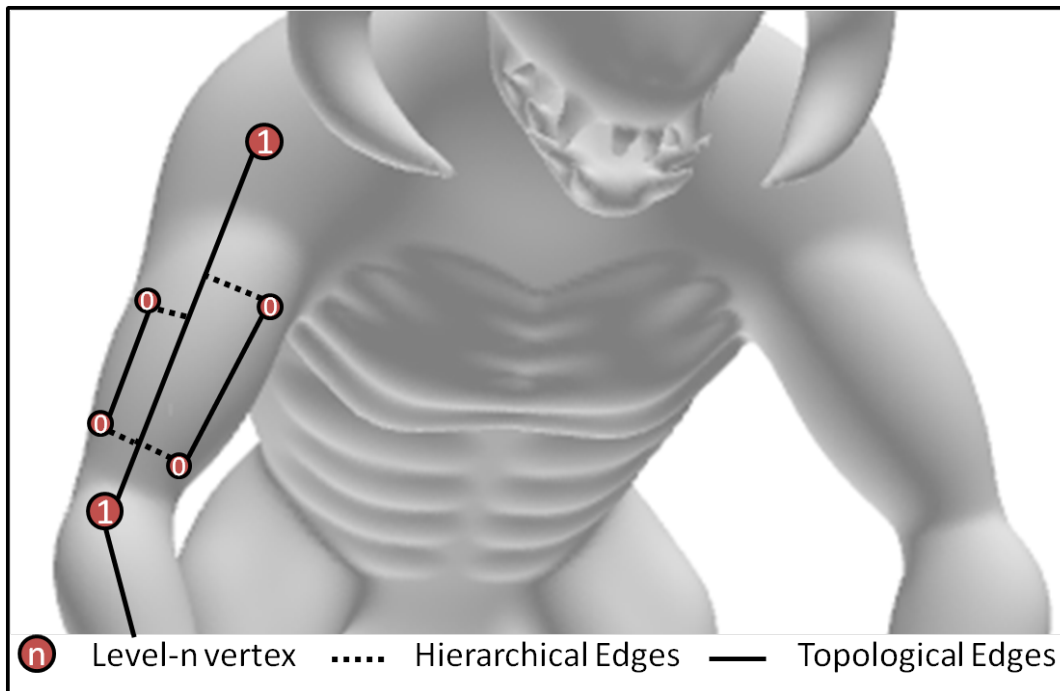
The former model, meshes, mainly defines scene elements that users can not directly deform, such as garments for a specific character. In the two last chapters of this thesis, we describe how garment meshes are deformed while users deform the associated character. If the mesh is not composed of triangle-only faces, we first have to triangulize it. In contrast, skeleton-based implicit surfaces are the core elements that users are able to deform, those surfaces are also represented by a mesh generated using the marching cube method [Blo88]. Reasons to choose skeleton-based implicit surfaces is that skeleton elements that generate the surface are commonly located around the medial axis which seems natural to interact with (section 5.4.2). Yet, note that any skinning-based methods that are able to deform the object may be used as graphical model, modulo the adaptation of the details presented below that are specific to the skeleton-based implicit surfaces [MTLT88, KCŽ008, VBG\*13].

While deforming object, users mainly interact with the implicit skeleton, although they do not see it. Therefore, the next section details the hierarchical structure of the skeleton.

### 7.2.2 Structural Skeleton

Although unseen, the object that we focus on are skeleton-based implicit surfaces, which use a skeleton to be modelized and deformed. In further words, models consist in two parts: the surface that represents and approximates the shape (also called skin or mesh), and a set of interconnected bones (called skeleton or rig). Skeleton elements are twofolds: vertices which represent the joints (similar to a kneecap), and edges which rigidly links vertices (similar to bones). Commonly, skeleton relies on a hierarchical set of bones, in order to be efficiently

animate it. Though, the hierarchical structure used in this work is not standard.



**Figure 7.2:** Skeleton example for the monster's arm, using both topological and hierarchical edges.

The links between vertices belongs to two different categories: hierarchical edges and topological edges. This skeleton representation is similar to Delame's works about skeleton [1]. The former edges (hierarchical) define the links between different levels of detail. The latter edges (topological) characterizes the core links in a given detail level. For instance, an arm would be first design with two straight topological edges, while muscle shape and other details are linked to these straight edges using hierarchical edges (fig. 7.2).

An integer is associated with each vertex to define the different levels of detail: the smaller the integer is, the finer details the vertex represents. The rules to handle edges are that:

- topological edges can only link two vertices with the same integer value; and
- hierarchical edges can only link a vertex of level  $n$  to a point on a level  $(n+1)$  edge (possibly a vertex).

The reason to use such a structural skeleton is to simplify the skeleton that users interact with. While interacting, user implicitly selects vertices on the skeleton. By enforcing the interaction on vertices belonging to the same level, (usually the highest possible one), the transformation has a smooth visual feedback. Moreover, all vertices with smaller integer exactly follow the motion of their parents, preserving the model details. In next section, we consider that manipulated skeleton always belongs to the same level of detail, and that interactions always occurs at the highest level of details (the core skeleton).

## 7.3 REGION OF INTEREST SELECTION

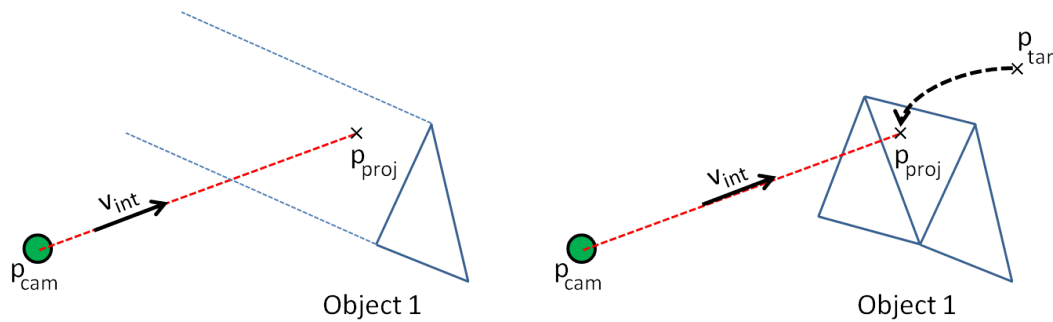
The very first step to manipulate and edit an object is to select a region of interest that users intent to edit. However, the information needed to actually perform the deformation depends on the specificities of their deformation. For instance, a pure translation of the camera only requires the camera position and the translation vector. In contrast, a local bending of the skeleton requires more information, such as the manipulated skeleton bones.

In fact, modes defined in section 6.3.1 gather not only tasks under a specific thematic, but also tasks which requirements are similar. Precisions about mode requirements are given in section 7.4.

This section details the different methods used by the interface to select attributes (section 7.3.1, or relevant vertices on the skeleton (section 7.3.2).

### 7.3.1 Object Selection and Center of Transformation

For the majority of tasks (especially for tasks belonging to navigation and object positioning modes), it is enough to select the object of interest and a specific target point as the center of all manipulations.



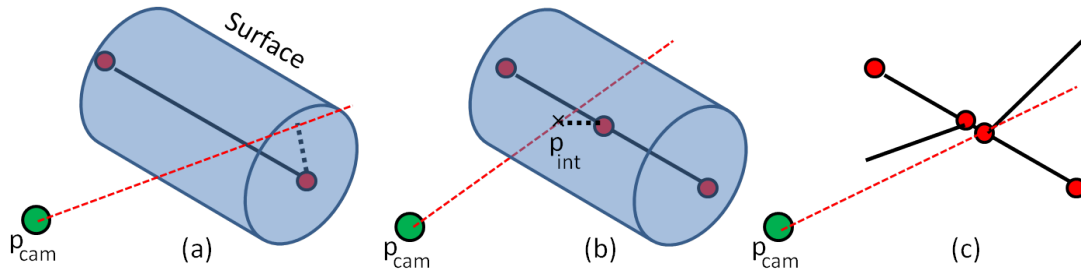
**Figure 7.3:** To discover whether an object is selected or not, we project the camera position  $p_{cam}$  along the vector  $v_{int}$  onto the plane defined by the triangle. left) When the projected point  $p_{proj}$  is outside all triangles, no object is selected. right) In contrast, if  $p_{proj}$  is inside one or more triangle, we store the nearest one and select the associated object.

To select a specific object, we rely on a greedy algorithm that looks for the intersection point between the nearest object and the interaction ray (fig. 7.3). More precisely, for each (non-degenerated) triangle defining an object, we first project  $p_{cam}$  onto the plane defined by the triangle, using the interaction vector  $v_{int}$ . Then, if the projected point  $p_{proj}$  belongs to the triangle area, we store the point and the associated object. If no point was detected using this method, no object is selected. If some objects are intersected, we choose the object with the nearest point from the camera position. In the latter case, the target point  $p_{tar}$  is set to the chosen point, which become the *center of transformation* (fig. 7.3, right).

Although the algorithm is greedy, it is performed only once at the beginning of the manipulation. Then, the goal is to keep the selected object under the interaction point during the whole manipulation (section 7.4). More, several tasks allowed by the interface, such as rotation, require that the new position of the target point to be the interacting point, which block us to use an approximate method.

### 7.3.2 Vertex Selection on Skeleton

Although storing the *center of transformation* in  $p_{tar}$  is enough for global object positioning, more refined object transformations require interaction with the skeleton itself. In our context, object deformation can only be handled by changing the skeleton. Therefore, the interface has to support the skeleton selection.



**Figure 7.4:** a) First method to select vertices: the closest vertex from the interaction axis is selected. b) Second method to select vertices: we first select the closest edges and project the point to create a new vertex in the skeleton. c) An example of a bad embranchment creation.

In contrast with the object selection method, the vertex selection for the skeleton consists of two methods:

- **Closest Vertex Method:** this method selects the closest vertex on the selected object's skeleton from the interaction ray (fig. 7.4, a). This method is fast but selects a point that can be distant from  $p_{int}$ . It can be used for transformation (not-support fingers) only require an approximation of the skeleton part to interact with (section 5.4.2), (e.g. adding material along the skeleton).
- **Closest Edge Method:** depending on the performed task, the interface may require a more precise point on the skeleton (fig. 7.4, b). The distance between the point resulting from the closest vector method and the interacting point is computed (the distance is performed in the 2D screen plane, in pixels). That point is selected if the distance is under a threshold. If not, the interaction point  $p_{int}$  is projected into each edge of the skeleton, and returned the closest one.

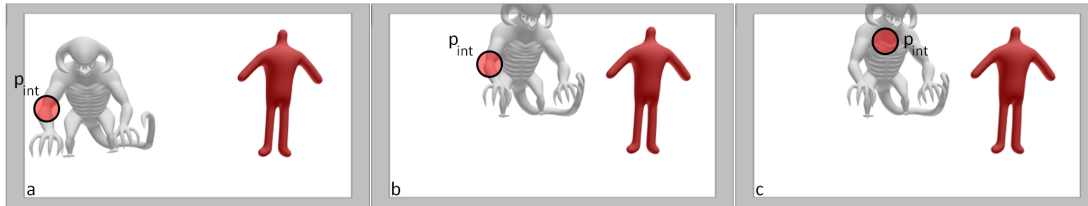
Reasons to first approximate by the closest vertex point are twofolds: first, users interact with fingers which sizes are not negligible. Therefore,  $p_{int}$  already approximate a region of interest. Due to perspective view, distance become larger and larger with the object distance from the camera position. That is the reason for processing distance in the 2D screen plane, which represents a more accurate approximation. Second, modifications of the skeleton structure engender many technical issues and could break the structure organization. For instance, fig. 7.4, (c) illustrates a case that the newly created vertex break the embranchment in two part, which is commonly undesired. For this reason, we prefer to first rely on the original skeleton structure before breaking it. Using the approximation preserves the original structure and the embranchments.

To enter the deformation mode, the user has to interact with two points on the object (section 6.3.1). Therefore, the interface registers two relevant vertices of the skeleton. The next step is to compute the path (named *chain*) between these two vertices. As the graph is an undirected unweighted graph, the interface performs a breadth first search algorithm to return the shortest path.

## 7.4 TASK DETAILS

A key goal is that users quickly learn the possible interactions provided by the interface. A first approach to ease interface learning is to ensure external consistencies, mainly by the choices for the mapping between user gestures and state-machine transitions (section 6.3.2). A second approach is to rely on internal consistencies and regularities while users perform gestures.

### 7.4.1 Gestural and Visual Consistency



**Figure 7.5:** Example of consistent/unconsistent translation. a) Starting 3D Scene; b) Consistent translation: the object exactly follows the finger movement and keeps the same point under the interaction point  $p_{int}$ ; c) Unconsistent translation: the selected point on the object is no more under the interaction point  $p_{int}$ .

Points defined with interacting fingers must be kept under the fingers (fig. 7.5). This is especially true for one-handed interactions, which are direct manipulations. For instance, a translated object have to remains under its interacting finger. Even though two-handed interaction represents non-direct manipulation, this property still remains for support fingers.

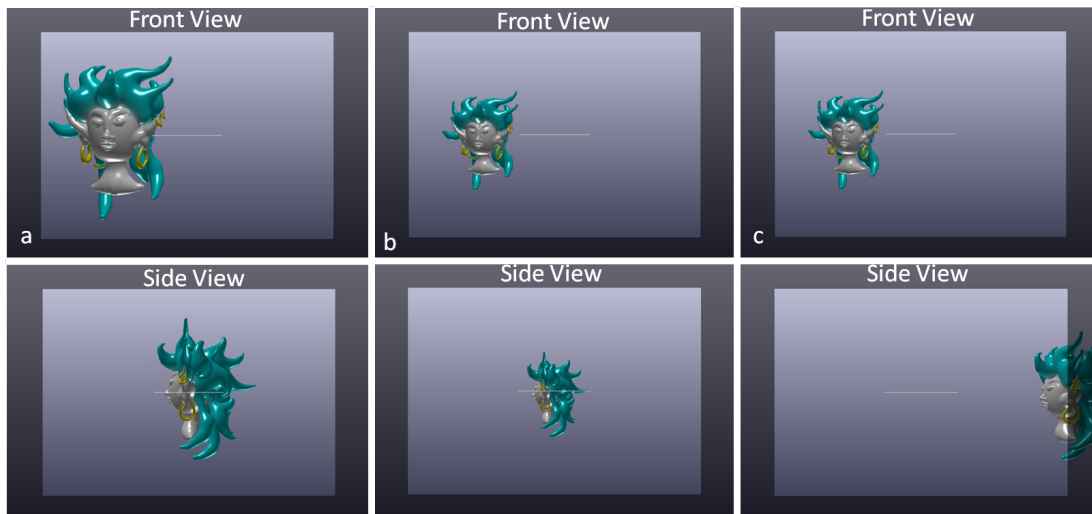
For tasks in navigation (resp. object positioning) mode, the interface relies on the *center of transformation* (i.e.  $p_{tar}$ ) to efficiently ensure this feature. For *1-Hand Translation* tasks,  $p_{tar}$  only has to follow the projected point  $p_{int}$  in the target plane. In other words, first the scene (resp. selected object) is translated by the vector  $p_{int} - p_{tar}$ , then  $p_{tar}$  is set to its new position  $p_{int}$ . For *2-Hand Rotation* tasks, the process is quite easy in Camera-centered Navigation: the camera turns around the axis defined by  $v_{cam}$  using the angular parameter given by the phase-based analysis (section 6.2). However, the process is a little more complex in Object-centered Navigation (resp. Positioning), as the interface has to relocate the camera (resp.  $p_{tar}$ ) afterwards to exactly observe the point under the finger at the same 2D screen position. To do so,  $p_{tar}$  position is decomposed in the old camera frame ( $v_{cam}$ ,  $up_{cam}$ ,  $right_{cam}$ , where  $up_{cam}$  and  $right_{cam}$  are the upper axis and right axis in the camera frame). Then the camera (resp.  $p_{tar}$ ) is translated using the same decomposition in the new camera frame. Similar techniques are performed for *1-Hand Shrink* and for support fingers on two-handed interactions.

For tasks in deformation mode, the techniques are similar to the translation, except that they are applied to the two selected vertices positions. Once fully positioned, the process runs through the skeleton graph to perform the desired deformation (section 7.4.3).

### 7.4.2 Gestural and Visual Regularities

Gestural and visual consistency are not the only features we desire to ensure for the interface. Another feature is that similar gestures should results in similar visual effects. These regularities are particularly important to characterize tasks performed with two hands.





**Figure 7.6:** Two object positioning tasks: a) Initial 3D Scene, b) Scene after scaling the object, c) Scene after translating it in depth. The transformation is performed in the front view. From this point of view, the visual effect is the same in order to preserve regularities.

For instance, within a given interaction, scaling/pinch gestures result in the same visual effect regardless the number of used hands (fig. 7.6). In further details, given the scale ratio parameter from the phase-based analysis (section 6.2), a scaling task globally scales the object by the ratio itself, while depth axis translation task translates the whole object by the vector  $(\frac{1}{s} - 1)(p_{int} - p_{cam})$ . In both case, the user performs a scaling gesture, either with one-handed gesture or with two-handed gesture. Note that the same couple of gestures results to be exactly the same transformation in Camera-centered navigation. This regularities is induced by the users' difficulties to distinguish zoom transformation from depth axis translation (section 4.4.5).

In two different modes, the visual effects of similar gestures are generally similar for the object of interest (or local part of interest), although different for the rest of the scene. Note that one-handed manipulations (in every mode) do not require special cases to preserve regularities: they are already preserved by the consistency itself. For two-handed manipulations, the regularity feature is particularly true between Object-centered Navigation and Object Positioning modes. Within deformation mode however, the two selected vertices in the skeleton graph (section 7.3.2) characterize an axis around which deformations are performed. Therefore, the same level of regularity would have been observed if the center of transformation had been replaced by the axis itself.

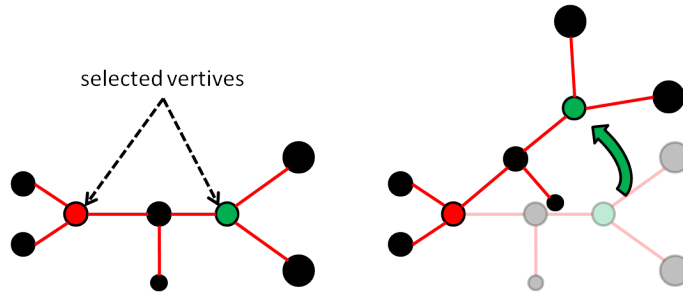
### 7.4.3 Deformation Details

While previous features, consistencies and regularities, are enough to define all tasks in navigation and positioning modes, deformations represent more complex tasks that require further computations. Therefore, this section presents the algorithms used in order to perform deformation tasks.

- **Local Growth:** although gesture to perform local growth deformation is one of the most complex one in the interface, computing the deformation is quite easy: the weight of

each vertex belonging to the selected chain is scaled according to the ratio given by the phase-based analysis.

- **Bending:** bending deformation is also quite easy to compute (fig. 7.7): the first vertex selected (in red) defines the pivot for the rotation (and so remains static), while the second vertex (in green) defines a rotation axis and a angular parameter by its old and new position. Then, using a depth first search algorithm on the branch characterized by the chain, all vertices rotate using the same rotation data (pivot, axis and angular).



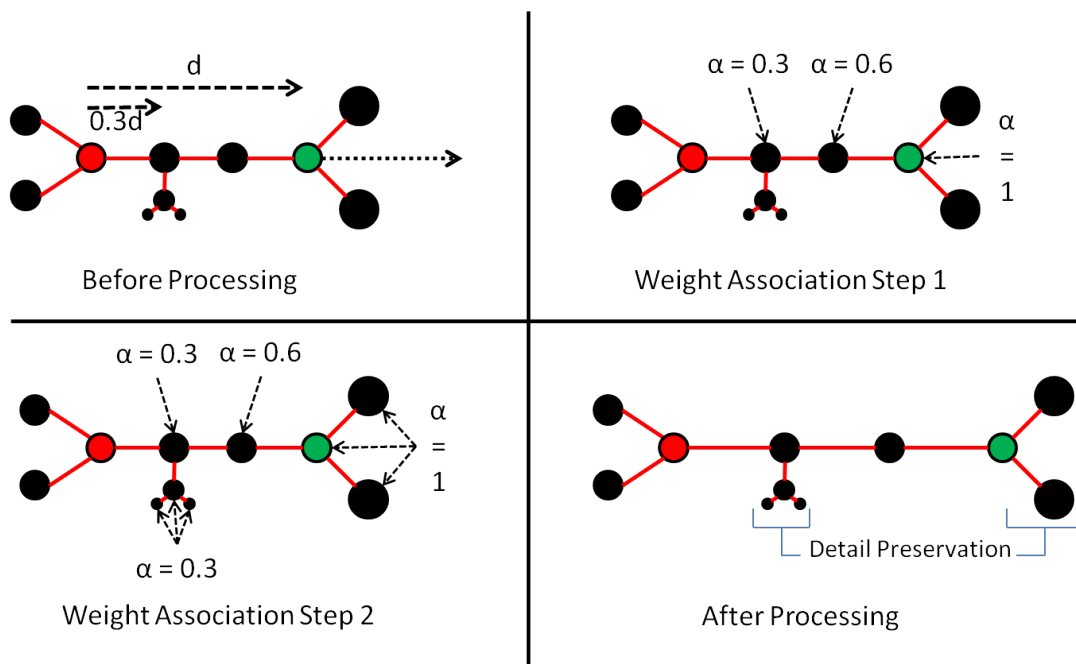
**Figure 7.7:** Example of a bending example. In red the selected vertex that remains unmoved, and in green the vertices following the second interacting finger.

In our context, the bending deformation is a rigid rotation of the whole branch defined by the two vertices. Despite other deformation techniques exist to smoothly bend the selected chain, rigid rotation deformation brings advantages: first, rigid rotation is one of the most common deformation, as it is very similar to real bone rotation. Second, technically speaking, rigid rotation do not depend on the number of vertices of the selected chain. Therefore, the obtain results is predictable, whereas smooth bending is harder to foresee. To complement a smooth bending algorithm, one could use a similar technique than the adding/deleting material deformation.

- **Adding/Deleting Material:** although adding (resp. deleting) material gathers two different tasks (adding material along the object vs. adding a whole new branch) performing them relies on the same algorithm. Indeed, once the whole new branch is created, the deformation consists in adding material along the new branch.

The process to add material is processed as follow (fig. 7.8):

- The process starts from an initial skeleton position in which two vertices were selected with fingers (fig. 7.8, a). The first one (in red) corresponds to the finger which remains static; The second one (in green) corresponds to the moving finger.
- Each vertex on the skeleton is associated to a specific weight value. First (fig. 7.8, b), each vertex in the selected chain is associated with the ratio  $\alpha$  between distance along the skeleton from the static vertex (in red) to the processed vertex/moving selected vertex. Note that the static vertex always has a ratio of 0 while the moving one ratio is 1. Second (fig. 7.8, c), the value is propagated along all derivated branches. In other words, remaining vertices are associated to the same value than the closest parents in the chain.



**Figure 7.8:** Process to perform adding material tasks: a) Starting skeleton positions; b) first step of weight association: each vertex in the main chain is associated to a weight; c) second step of weight association: remaining vertices are linked to the value of the closest parent in the chain. d) Final result after process.

- Finally, the process performs the task (fig. 7.8, d): each vertex is translated by a vector  $\alpha v_{add}$  in which  $v_{add}$  characterizes the translation of the moving selected vertex (the green one).

The reason to associate the weight value of the closest parent in the chain is to preserve details under the same relative location and the same size. Details - represented by the derivated branch from the selected chain - are usually created under a specific global scale, and shrinking it under a specific direction globally changes the shape of the details.

- **Twisting:** the algorithm used to handle twist deformations is similar to the adding/deleting material deformation, except than the associated values globally scale the rotation along the axis defined by the selected vertices. For this deformation, the two selected vertices remain static, while the third moving finger defines the maximum angular parameter (when  $\alpha = 1.$ ) - i.e., the distance performed by the finger is proportional to the angular value.

## 7.5 CONCLUSION

Although we mainly focused on the interaction point of view in this work, resolving technical issues brings interesting ideas to keep the interface logical and therefore logical to use. From the definition and the representation of contents in the 3D world, to the development of manipulation algorithm, which mainly rely on two principles (consistency and regularity), the interface achieves the main goal of this work: to propose an easy-to-use, quick to learn inter-

face by simplification of each step of the 3D modeling pipeline. Indeed, the interface contains the fundamental basis to analyze raw input and to create and modelize 3D shapes.

Even though users freely create 3D shapes, deformation tasks remain elementary tasks to modelize. The next chapters propose a mean to extend the interface to more complex deformations through an example: we first develop a new method to automatically transfer garments from a model to another. Then, the method is adapted to automatically deform garments while user deform the character.



# DESIGN PRESERVING GARMENT TRANSFER



**Figure 8.1:** *Design preserving garment transfer:* (left) transfer of a multi-layer outfit from a woman to a young girl; automatically graded patterns (bottom) shown to scale. The zoomed-in source and target patterns for the back highlight the subtle changes in shape. (right) Dancer outfit transferred from the female in the center to a variety of other characters.

## 8.1 INTRODUCTION

Modeling of complex real or virtual garments is a highly time consuming and knowledge intensive task. For instance, it took a professional computer artist three hours to generate the woman's outfit in fig. 8.1, left; a process that included creating 2D patterns, placing them on the mannequin, running cloth simulation and then iteratively adjusting patterns and parameters, repeating the process until the desired effect was achieved. If we wished to design the same woman while we adapt her proportions, this tedious process would need to be repeated practically from scratch. The issue is the same if we wished to design a similar looking outfit for a young girl who was very different proportions (fig. 8.1, left). The issue is the same if we wished to design the same woman while we deform her shape. Consequently, dressing a larger number of virtual characters in similarly designed outfits (fig. 8.1, right) is currently prohibitively time consuming.

In real-life, professional pattern makers adapt garments to people with different proportions and body-types through a complex process known as grading or pattern grading. Manual grading aims to preserve the design or look of the garment while adjusting it to fit the proportions of the new wearer [MMY01]. This process requires significant specialized expertise and relies on a set of precomputed rule tables for standard body sizes. Applying a similar process to virtual garments can be especially challenging since the characters to be dressed often have non-standard body proportions for which no rule-tables exist. While skinning-type garment transfer techniques [CSMT03, WWY05] adjust the garment to fit the target character's proportions reasonably well, they often fail to preserve the original design, requiring manual editing to recover it [MWJ12].

Instead, we first focus on automatic design preserving garment grading or transfer, introducing a method that produces garments that fit the proportions of the new wearer while retaining the original design as much as possible. As emphasized in the literature, garment transfer is an ill-posed problem. Formalizing what makes a garment look the same on characters of different body shapes, although the input and output garment surfaces obviously differ, is in itself a challenge. Therefore, our first contribution, presented in section 8.2, is to analyze professional pattern-grading criteria and reexpress them in terms of geometric requirements for garment-mesh transfer. We then introduce a garment transfer algorithm that satisfies these requirements, as our main contribution. We formulate garment transfer as a constrained optimization problem which optimizes design preservation subject to a number of geometric constraints arising from proportionality, plausibility and other considerations elaborated upon in section 8.2. To efficiently generate the graded garments we solve this problem using an iterative process, where each iteration consists of solving a quadratic minimization problem. In the next chapters, we details how this transfer techniques is adapted to deform garments while deforming characters.

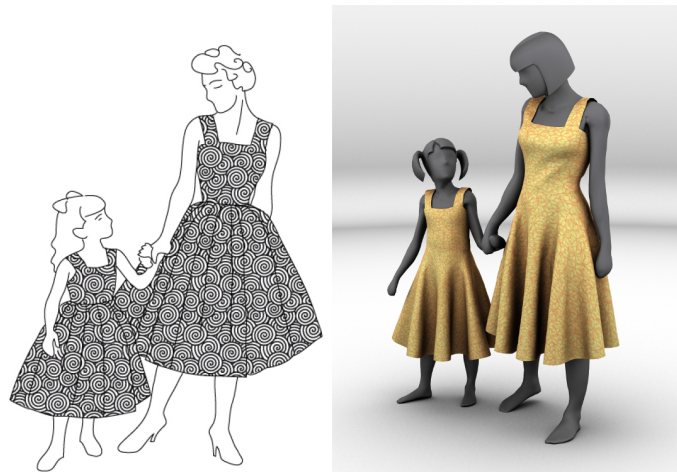
## 8.2 GRADING CRITERIA

Before developing a garment transfer method, we must know what users expect from a transferred garment method [MMY01]. A common statement of these transfer, or grading, requirement is that "the purpose of grading is to proportionally increase or decrease the size of a pattern, while maintaining shape, fit, balance, and scale of style details." In addition to the criteria above, we must consider two extra plausibility criteria that are implicit in real garment grading. The following discuss the geometric formulation of these criteria.

### 8.2.1 Proportion and Scale

The proportionality, or proportional increase/decrease of size, requires the resized garment to preserve the relative location of garment features with respect to the character's limbs and body. For instance: a knee-length skirt needs to remain knee length independent of a change in a character's height, a dress waistline needs to remain on the waist, and a side zipper needs to stay on the side. The proportional scaling of details reinforces proportionality, requiring not only to preserve feature location but to scale those appropriately. From a geometry processing viewpoint, proportionality and scale combined can be paraphrased as a single *relative location* requirement, where we associate a reference location on the body with each point on the garment and preserve the direction from one to the other during transfer, preventing the garment from sliding along or twisting around the body.

### 8.2.2 Shape



**Figure 8.2:** Mother-Daughter dresses: (left) traditional grading, (right) our variations

The shape preservation criterion requires grading to preserve the overall look or design of the input garment. E.g., a straight skirt should remain straight, boot-cut jeans should retain the boot cut and so on. The mother-daughter dresses (fig. 8.2, left) provide an illustration of shape preserving manual grading which successfully handles large differences in proportions. From our observations of professionally graded garments, confirmed in a discussion with a fashion designer, expert graders allow for significant changes in the geometry of the garments to accommodate changes in proportions but aim to preserve the slope or tangent plane orientations across the garment surface whenever possible. Thus for instance, the hem width of bell-bottom jeans designed for a taller person is increased proportionally to height to maintain the bell slope. Consequently, we restate shape preservation in geometric terms as *preservation of garment surface normals*.

### 8.2.3 Fit

The fit criterion reflects preservation of local distance between the garment and the wearer. As discussed above, the distance to the body in loose areas is often expected to change significantly to preserve shape; hence, this criterion is largely enforced in tight regions. Even for these regions fit preservation is often a design choice, a designer may selectively relax fit in some

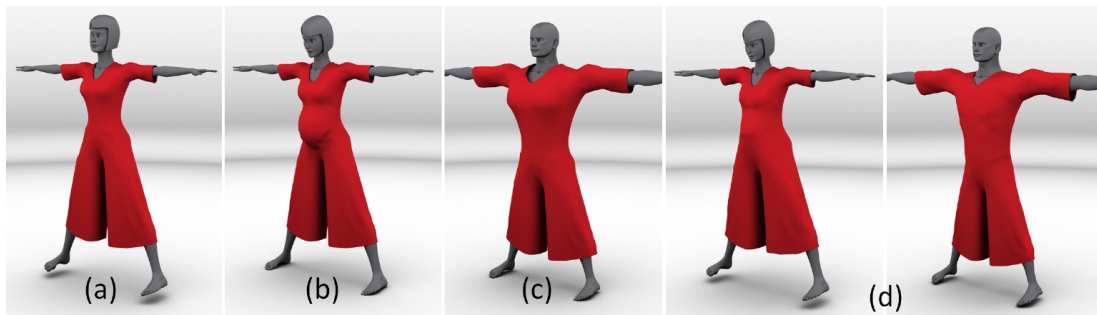


locations to more closely adhere to the source garment shape. A typical example is transfer of a skirt or pants to a fuller person, where designers typically loosen the garment on the lower half of the belly to achieve a visually skinnier look [Fre11]. Geometrically, fit preservation can be expressed as forcing the garment points that are very close to the source character's body to remain close to the target character after the transfer.

### 8.2.4 Balance

Lastly, the term balance refers to the hang of the garment around the body which is largely controlled by the orientation of the patterns with respect to the weave directions of the selected fabric. As such, it is orthogonal to the geometric considerations involved in transferring a 3D virtual garment, where one can always assume that the patterns remain correctly oriented.

### 8.2.5 Manufacturability



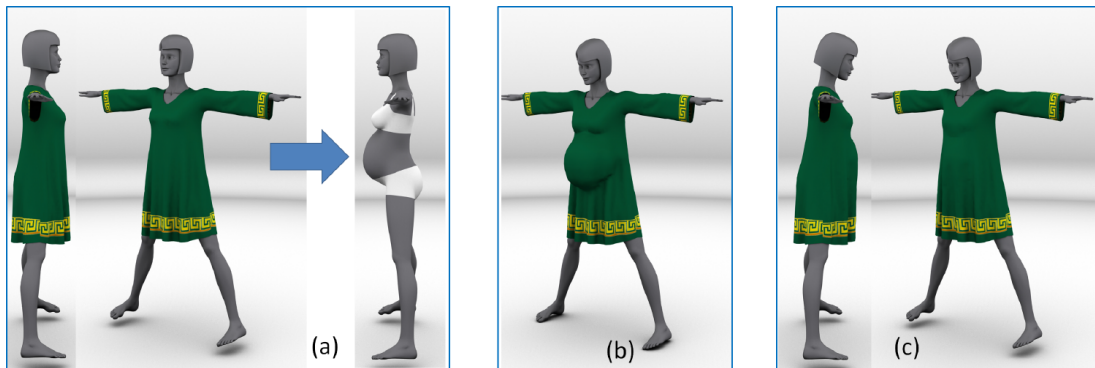
**Figure 8.3:** *The impact of fit settings: a) source; b) enforcing fit across the tight regions on a pregnant female leads to undesirable increase in curvature and focuses attention on the belly; c) with no fitting enforced the garment transferred to a male looks implausible. d) Our results for both methods with fit enforced selectively.*

Real garments need to be manufactured from planar fabric panels. As real-world fabric has a limited stretchability, the Gaussian curvature of the transferred 3D garment has to be bounded in the interior of each panel. The shape preservation metric previously discussed implicitly keeps the curvature low, since for a surface with *a priori* low Gaussian curvature, preserving normals implicitly preserves curvature. However, manufacturability needs to be explicitly considered when processing regions where fit is enforced. In these areas, the garment follows the shape of the body and therefore can be affected by increase in body surface curvature (see fig. 8.3, a,b). To preserve manufacturability professional graders either selectively relax the fit thus reducing curvature, or introduce complex changes to patterns, such as multiple darts and gussets, to maintain the fit. In this paper, we opt for the former option, selectively relaxing fit to minimize curvature increase.

### 8.2.6 Collision Avoidance

Real garments clearly cannot intersect the wearer's body or underlying clothing layers. Thus when transferring a garment to a larger person, designers increase pattern size to preserve shape and avoid fabric stretching or worse tear.

### 8.3 METHOD OVERVIEW



**Figure 8.4:** *Algorithm Overview: given the dressed source character and a target one (a), we first proportionally scale the 3D garment to obtain a transferred version that satisfied proportionally and fit (b). Then, an iterative optimization process combines the source garment normals and the positions generated by the scaling to obtain a design preserving transferred garment (c).*

Our garment transfer method operates directly on 3D garments and finds a satisfactory balance between the criteria listed above: relative location, shape, fit, manufacturability and collision avoidance.

The input to our method is a dressed source character model, possibly with the associated 2D clothing patterns, and a target character, all represented by manifold triangular meshes. We assume that the characters are posed in a similar pose, and that a dense correspondence or cross-parameterization is predefined between them. We also assume that both characters are rigged, i.e., have an animation skeleton. There are a large number of cross-parameterization methods, especially when it comes to rigged models, which can be easily used to satisfy these requirements [SPR06, CCLH06].

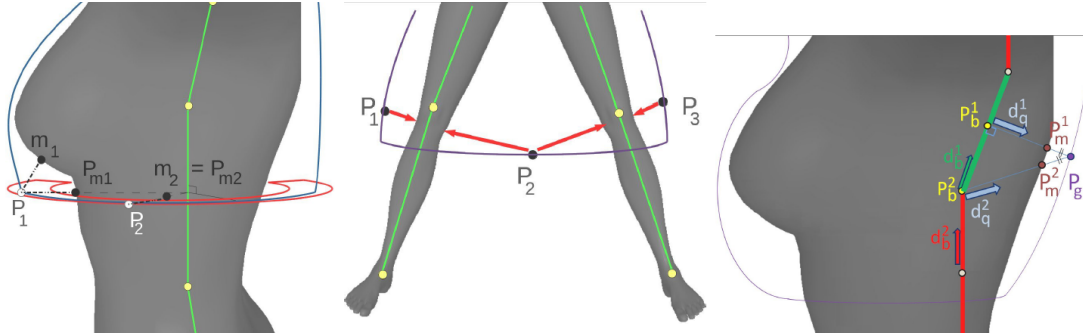
We break the transfer process into two steps (fig. 8.4): a skinning-like initialization that generates a proportionally scaled reference garment (see section 8.4) and a global design preserving optimization (see section 8.5). To efficiently preserve the shape of the source garment, we restate normal preservation in terms of per-triangle transformation gradient. We introduce a transformation gradient based formulation which balances normal, relative location, and fit preservation, and can be solved using an efficient iterative solver (see section 8.5.1 and 8.5.2). In sections 8.5.3 and 8.5.4, we show how to combine this minimization with collision avoidance and multi-layer garment processing, leading to the final iterative algorithm that generates the desired transferred garment.

### 8.4 PROPORTIONAL SCALING

The first step of our method generates a proportionally scaled version of the input garment which satisfies the relative location and fit criteria (fig 8.4, b). This scaled version is then used as a starting point for the normal preserving optimization which generates the target transferred garment. Our proportional scaling method can be seen as an extension of smooth skinning, where the combination of constant positions in several skeleton frames is replaced by a combination of constant *offset vectors* from a set of adequately chosen *reference points* on the

character models. Using skin-based offsets allows our method to cope with changes in body shape, such as arm thickness, in addition to handling changes in skeletal proportions.

#### 8.4.1 Selecting Reference Points



**Figure 8.5:** (left) Selecting closest points  $m_1$  and  $m_2$  as reference points for  $p_1$  and  $p_2$  is suboptimal, as they can move apart on the target character, causing an undesirable vertical displacement. Selecting the orthogonal direction will map both points to a circle around the same bone point, preserving the relative height. (center) Association to multiple reference points is necessary to correctly account for regions roughly equidistant to multiple bones. (right) Association must depend on angle as well as distance to nearest intersection, giving preference to  $p_m^1$  over  $p_m^2$ .

To perform the proportional scaling we first associate each garment vertex  $p_g$  with pairs  $(p_b, p_m)$  of relevant bone and character skin points. We call these pairs *reference points*. Note that using the closest skin point as a reference is not necessarily the best choice (fig. 8.5, left), as it can cause points to slide along the body on the target character. Assigning references orthogonally to the local bone directions reduces such sliding, resulting in better relative location preservation, most notable along garment boundaries. Therefore, the reference point pairs we select correspond to the local minima of a function expressing distance to the character model in a direction orthogonal to the local bone (fig. 8.5), computed as follows.

Given a garment vertex  $p_g$ , we first compute its closest points  $p_b$  on each bone of the animation skeleton ( $p_b$  is possibly an extremity). For each bone, we define  $p_m$  as the input character point closest to  $p_g$  along the segment  $[p_b, p_g]$ . Such a point necessarily exists since  $p_b$  is inside, while  $p_g$  outside the character model. To select one or more reference points we consider both the distance from  $p_g$  to each of the intersections  $p_m$  and the angle between the bone axis  $v_b$  and the vector  $p_g - p_b$  as we prefer these two vectors to be orthogonal. The pairs  $(p_b, p_m)$  we select as reference for  $p_g$  are those that minimize a combination of these two metrics:

$$\|p_m, p_g\| e^{\frac{\langle v_b, v \rangle^2}{\sigma^2}} \quad (8.1)$$

where we set  $\sigma = 0.1$ . In most cases, this function has a clear single minimum (fig. 8.5, right), which we select as the reference. When two identical or very close minima exist, we assign both as references. This is typically the case in areas in-between the legs of the character or under the arms (fig. 8.5, center). Finally, each point  $p_g$  is associated with an offset  $o = \|p_g - p_m\|$  for each reference pair  $(p_b, p_m)$ .

### 8.4.2 Proportional Scaling Using Offset Vectors

We use the computed reference points and offsets to obtain the proportionally scaled locations  $\hat{p}_g$  for each garment point with respect to the target character. First, for each reference pair  $(p_b, p_m)$ , on the source character, we use the cross-parameterization to obtain the corresponding pair of points  $(\hat{p}_b, \hat{p}_m)$  on the target. Each point  $p_g$  has one, or more, reference pairs. If it has a single reference pair  $(p_b, p_m)$ , we set  $\hat{p}_g = \hat{p}_m + o * v_{bm}$ , where  $v_{bm}$  is a unit vector in the direction of  $(\hat{p}_b, \hat{p}_m)$ . If there are several reference pairs for the garment vertex  $p_g$ , we use a weighted sum of the positions dictated by each reference, using non-linear weights to get a sharp transition,  $w_1 = \frac{\arctan(5(t-0.5)+\frac{\pi}{2})}{\pi}$  and  $w_2 = 1 - w_1$ , where  $t$  is the coordinate of the projection of the garment vertex  $p_g$  on the segment between the two relevant bone reference points.

Figure 8.4,b illustrates the output of the proportional scaling step. Note that garment borders and other features keep their relative location with respect to the character’s body and limbs, while the use of skin based offset accounts for changes in body shape between the input and output character, preserving fit. At the same time, the results clearly do not account for shape preservation, necessitating the main step of the algorithm described next.

## 8.5 SHAPE PRESERVING GARMENT TRANSFER

### 8.5.1 As-2D-as-possible Deformation

Preservation of garment shape requires the normals of the garment mesh triangles to remain constant or near-constant during transfer. In general, expressing normal preservation as a function of vertex positions leads to a non-linear, hard to control optimization. Hence, existing deformation and resizing methods, e.g. tend to penalize both out-of-plane rotations and 2D deformations [BWKS11]. To penalize all rotations, Kraevoy et al. use a volumetric formulation with global scaling constraints [KSSCO08]. Their approach can not be extended to our setup as we require a much finer level of control. To efficiently penalize rotations, while allowing for 2D deformations we express normal preservation via transformation gradients. This formulation allows for a solution mechanism based on iterative linear minimization of a quadratic functional. Contrary to previous gradient based methods, which aim to either preserve known transformation gradients [SP04] or to search for as-rigid-as-possible ones [IMH05, SA07], we search for as-2D-as-possible transformation gradients.

Following Sumner et al. [SP04], we label the vertices of each source garment triangle  $t$  as  $p_1, p_2$  and  $p_3$  and add a virtual vertex  $p_4$  computed by offsetting  $p_1$  by the triangle normal. We then define the  $3 \times 3$  matrix representing the local triangle frame as  $P^t = (p_4 - p_1, p_4 - p_2, p_4 - p_3)$ . The gradient of the triangle transformation from the source to the target garment can then be written as  $\tilde{P}^t(P^t)^{-1}$  where  $\tilde{P}^t$  is the local frame after the deformation. Normal preservation for all the mesh triangles is thus expressed as the minimization of:

$$E_{shape} = \sum_t \|\tilde{P}^t(P^t)^{-1} - T^t\|_F^2 \quad (8.2)$$

where  $\|\cdot\|_F$  is the Frobenius norm for matrices, and the unknown target triangle gradients  $T^t$  are constrained to be 2D-only transformations defined in the plane of the source triangle, i.e., applying  $T^t$  to  $t$  preserves the triangle’s normal.

**Solver:** The numerical challenge we face is to enforce the 2D-only constraints on  $T_t$  while solving for both  $\tilde{p}$ , the target vertex positions, and  $T^t$ , the target gradients, subject to the additional constraints listed in section 8.5.2. To compute a solution we use an iterated least-squares process which alternates between updating  $T^t$  while keeping  $\tilde{p}$  fixed, and vice versa. We start by setting  $\tilde{p}$  to the positions  $\hat{p}$  of the vertices after the proportional scaling based initialization (section 8.4). We then iterate the two subsequent steps.

**Updating  $T^t$ :** To update  $T^t$  given the current value of  $\tilde{p}$ , one option is to perform QR decomposition of the gradient  $\tilde{P}^t(P^t)^{-1}$  and use the non-rotational component as the new 2D-only transformation gradients. However, such decomposition discards the in-plane rotations, which in our case, we want to allow. Instead we compute  $T^t$  as follows: we project each triangle to the corresponding plane using the reference normal  $n^t$  on the source garment:

$$p_i^t = \tilde{p}_i^t - \langle \tilde{p}_i^t, n^t \rangle n^t \quad (8.3)$$

obtaining a local frame  $P^{tt} = (p_4' - p_1', p_4' - p_2', p_4' - p_3')$ . Typically,  $\|\tilde{P}^t(P^t)^{-1} - P^{tt}(P^t)^{-1}\| < \|\tilde{P}^t(P^t)^{-1} - T^t\|$ . If this is the case, we set  $T^t = P^{tt}(P^t)^{-1}$ , otherwise  $T^t$  is unchanged. Consequently the update step is guaranteed to never increase the value of the optimized functional. Since the same is true for the vertex solve, the optimization is guaranteed to converge.

**Vertex Solve:** With  $T^t$  fixed, the solution for  $\tilde{p}$  minimizing the quadratic equation 8.2 reduces to a simple linear system, which we solve in combination with the constraints detailed below. Since the positions do not change significantly between iterations, we use an iterative CG solver with the previous positions as an initial guess.

### 8.5.2 Accounting for Relative Location and Fits Constraints

To generate a properly transferred garment, the shape preservation term described by equation 8.2 needs to be combined with terms enforcing preservation of relative locations and fit. Both of these are by construction satisfied by the vertex positions  $\hat{p}$  of the proportionally scaled garment computed by the initialization step (section 8.4).

To introduce these considerations into the shape-preserving formulation, we consider the local frame(s) defined by the relationship between each vertex  $\hat{p}$  on the proportionally scaled garment and its associated reference skeleton point(s)  $p_b$ . Each frame is formed by the directions  $d_b$  of the corresponding skeleton bone, the unit vector  $d_q$  collinear to  $(p_b, \hat{p})$ , and their cross product  $d_t$  (by construction  $d_q$  is typically orthogonal to  $d_b$ ).

To enforce **relative location preservation**, we use the following term:

$$E_{rl} = \sum_{\tilde{p}} \alpha_p (\langle \tilde{P} - \hat{p}, d_b \rangle^2 + \langle \tilde{P} - \hat{p}, d_t \rangle^2) \quad (8.4)$$

The two components of the sum aim at preventing the garment from sliding along or twisting around the skeleton respectively. For interior vertices in loose regions we use a low  $\alpha_p = 0.5$  prioritizing shape preservation at the expense of minor inaccuracy in relative locations. Along garment boundaries and seams where any twisting or displacement would be very noticeable, as well as in the tight regions, we set  $\alpha_p = 1000$ . As relative location does not dictate any constraints on the distance between the character's body and the garment, the component of  $\tilde{p} - \hat{p}$  aligned with  $d_q$  is unconstrained.

To enforce **fit in tight regions**, specified as described below, the relative location term is augmented with a fitting term, explicitly constraining this distance:

$$E_{fit} = \beta \sum_{t \in F} \sum_{\tilde{p} \in t} \langle \tilde{p} - \hat{p}, d_q \rangle^2 \quad (8.5)$$

where the set  $F$  contains the triangles in the regions deemed tight, and  $\beta = 1000$ . The triangles in  $F$  are also removed from the shape energy term  $E_{shape}$  as in this case, there is no reason to preserve their normals.

Using the two terms above, the energy functional to be minimized in the iterative process described in section 8.5.1 is set to:

$$E = E_{shape} + E_{rl} + E_{fit} \quad (8.6)$$

Given the DoF provided by varying the distance to the character's body, we typically can simultaneously preserve both relative location and shape in the loose regions pretty well.

**Selecting the Tight Regions:** As stressed in section 8.2, in real life grading the selection of the region where fit should be preserved is a design choice [Fre11]. The simplest selection would be based on a distance threshold. However, the choice is problematic when the body curvature grows significantly between the source and target characters - e.g. from a regular female to a pregnant one (fig. 8.3, b). In this case, preserving tightness can increase the garment's Gaussian curvature making manufacturing a challenge and creating an unappealing look. Instead, we opt for the selective relaxation technique described in section 8.2, relaxing fit in areas where the normal to the source garment points downward or sideways. We thus assign to  $F$  triangles with all vertices located closer than a distance threshold from the model, and whose normals  $n^t$  on the source garment are pointing upward. This strategy leads to consistently believable transferred garments, such as the overalls in fig. 8.3, d. Note, that completely relaxing the fit across the model can lead to implausible results (fig. 8.3, c), with the male character looking like he wears a bra. The reason is that lacking physical context the method cannot distinguish between upward pointing garment areas which are a product of garment-body interaction, as is the case of the overall, and those formed by design (e.g. through bra type seaming). Our tightness setup assumes the former scenario, as it is a more likely one.

### 8.5.3 Handling Collisions

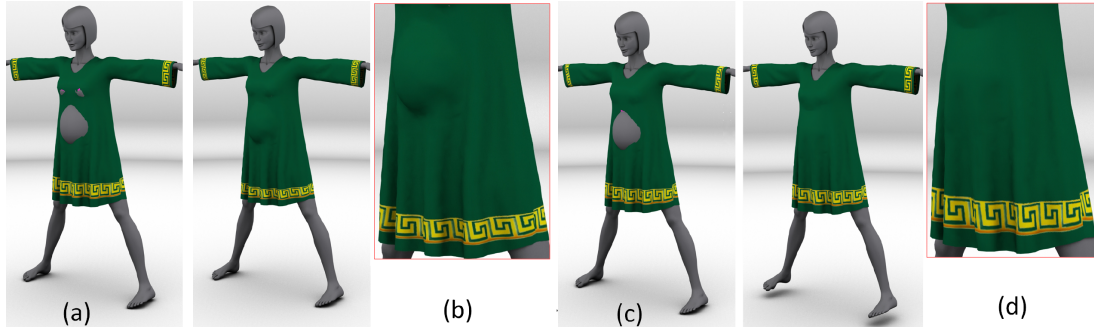
The transfer process as described so far does not explicitly prevent garment interpenetrations with the target character or self-intersections. Self-intersections are very rare and can be easily resolved by any standard post process that moves vertices apart without any changes to the overall garment design. In contrast, body-collisions typically occur when transferring a garment to targets with more protruding parts than the source (fig. 8.6, a) and indicate a need to add more fabric to accommodate the change in body shape. Note that collisions can only occur in the loose regions of the garment, as the fit constraint prevents interpenetrations in tight regions. To add more fabric and prevent collisions during transfer we take advantage of the local frames ( $d_b, d_q, d_t$ ) associated with each vertex (section 8.5.2). Using these frames collision avoidance can be paraphrased as:

$$\langle \tilde{p} - p_m, d_q \rangle \geq \epsilon \quad (8.7)$$

In other words, the garment has to be farther away from the bone than the skin of the model. Combining this constraint with strong preservation of relative location, one can theoretically

avoid all intersections with the body. The practical challenge is in enforcing such inequalities efficiently.

**Solution Mechanism:** To implement efficient collision avoidance, we opt for an active set approach especially tailored for our needs. Standard active set methods [NW06] replace inequalities by strategically selected equality constraints, using an iterative addition and removal process. In our case, we leverage knowledge about the problem setup at hand, to develop an efficient solution method where only strictly necessary equalities are added and consequently never removed. This modification makes the solution both faster and robust.



**Figure 8.6: Collision resolution order:** *a) unconstrained solution with step one collisions highlighted; b) converting all collisions into equality constraints at once leads to undesirable artifacts on the belly (see inset zoom). c) Incremental solution within one iteration resolves all collisions on the chest (step two collisions highlighted), d) and in a few iterations generates better transfer results with all collisions resolved.*

We first optimize the energy functional in equation 8.6 with no collision constraints, and then iteratively introduce collision-resolving equality constraints and repeat the optimization as long as collisions persist. Specifically, we observe that in our setup resolving collisions higher-up along the body often resolve those below, while the opposite practically never happens. In general, once we move vertices away from the body, shape preservation pulls those below away as well, while vertices higher up are held in place due to fit constraints typically enforced in the tight regions on the shoulders. This is likely due to garment interaction under gravity. Knowing this, we use a top to bottom equality enforcement strategy. We obtain best results by adding constraints very gradually, two by two (to account for symmetries often present in garment models). After each collision processing step, we repeat the optimization (equation 8.6). This iterative process leads to better results than introducing the constraints all at once, see fig. 8.6, b versus fig. 8.4, c.

The actual collision detection can be done very efficiently by intersecting the segment(s)  $\tilde{p} - p_b$  (where  $p_b$  is the reference bone point associated with  $p$ ) with the mesh representing the target character and performing an inside/outside test by comparing skeleton-garment vs skeleton-character distances. While in theory the garment can intersect other areas on the skin, this had never happened in practice. In the case of penetration, we recompute  $\hat{p}$  by projecting the vertex a small distance outside the model along the current segment  $\tilde{p} - p_b$  and add the highest triangle adjacent to the vertex to  $F$  (fig. 8.6, c), treating the triangle as tight fitting for further computations.

Even with the very gradual introduction of equalities, the collisions are typically resolved in just a few iterations (tab. 8.1).

### 8.5.4 Extension to Layers of Garments



**Figure 8.7:** A multi-layer flamenco dress transferred from a woman to a girl. The undistorted texture highlights our method’s ability to automatically generate appropriate patterns.

Real-life outfits often consist of multiple layers either fully separate, such as a dress with an under-skirt (fig. 8.1), or stitched together, such as a dress with multiple ruffle layers (fig. 8.7), or a jacket with outside pockets. Our transfer method easily extends to this case. We incrementally process each fabric layer, starting with the layer closest to the body and proceeding to the outer one. At each stage, we treat the already processed layers as part of the current character model. The outermost surface of this model is then used to compute the proportionally scaled version of the next source layer, used to initialize the subsequent optimization process.

### 8.5.5 Pattern Extraction



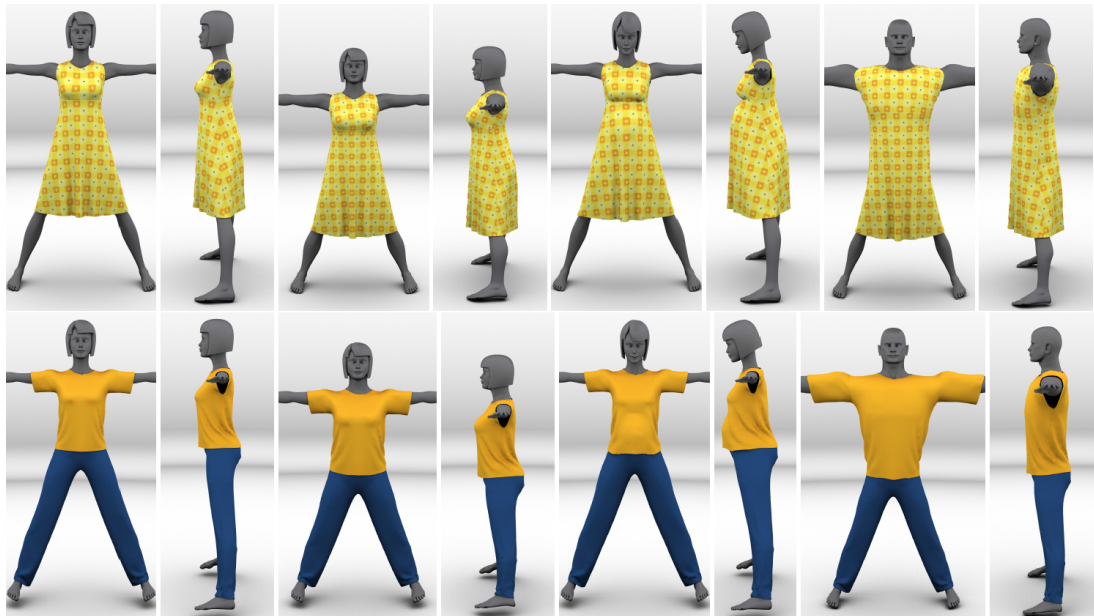
**Figure 8.8:** Validation of a real dress: we transferred the woman’s dress on the left to a Numina doll model and used the patterns to make an outfit for the real doll (right). The virtual source and real graded patterns for the front and back.

To obtain patterns for the transferred garment, we cut it along the seams indicated on the source and use ABF++ parameterization method [SLMB05] to parameterize the resulting charts. We found that this approach provides an optimal time-quality trade-off. It performs



equally well to more time consuming stretch minimization techniques. At the same time it creates better, more symmetric, patterns than faster, linear methods in cases where the source and consequently target garments are not perfectly developable [SPR06]. We automatically discard darts that are no longer necessary, i.e. ones effectively closed by the parameterization. The  $L^2$ -stretch between the garments and the patterns we computed was in range of 1.0002 to 1.003 (ideal value is 1), indicating negligible distortion. Fig. 8.1 and 8.8 show patterns created using our approach. Patterns created this way are well suited for texturing or physical simulation. However they tend to have somewhat wavy outlines, sub-optimal for actual manufacturing. To regularize those, one can apply standard methods which approximate nearly straight outline segments by straight lines, and nearly circular ones by perfect arcs [MZL\*09].

## 8.6 RESULTS AND DISCUSSION



**Figure 8.9:** Transfer of garments to progressively more different models (front/side views): a stockier shorter female, a pregnant female, and an exaggerated male character. The texture on the transferred dresses was generated using the automatically computed patterns. Discontinuities (e.g. the line under the chest) reflect seam locations.

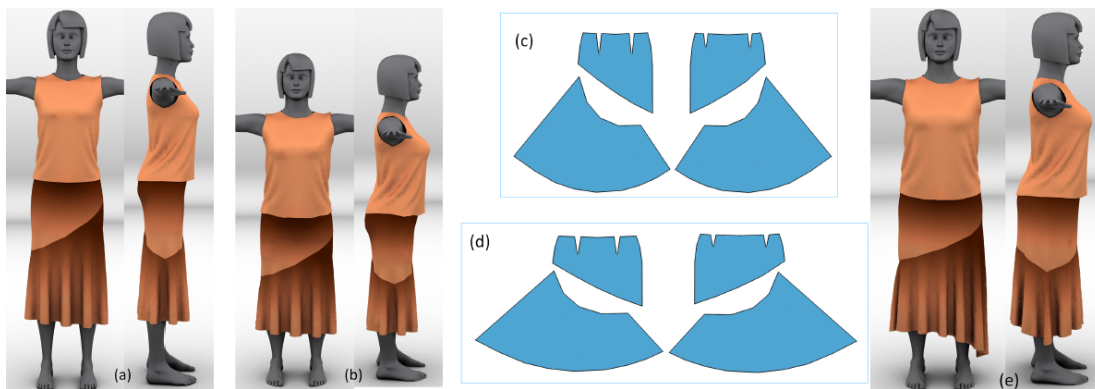
Throughout the paper we demonstrate the results of our method on a variety of models. We show results both in the canonical pose in which the transfer was processed (e.g. fig. 8.4 and 8.95, 12) and after simulation in alternative poses (e.g. fig. 8.1 and associated video). The latter demonstrate that, consistent with user expectations, the transfer process preserves design similarity between the source and target garments in general and not just in a particular pose. Fig. 8.1, left and 8.7 show believable transfer of complex layered outfits between models with significant differences in proportions and body shape. Fig. 8.1, right further reinforces the method’s robustness to large changes in proportions and shape and highlights the suitability of our method for transferring garments to large crowds of highly diverse, often unrealistic, virtual characters. Fig. 8.9 shows transfer of typical real-life garments to increasingly different

characters, creating results consistent with viewer expectation. The texture on the transferred models in fig. 8.2, 8.7 and others highlights preservation of manufacturability during transfer.

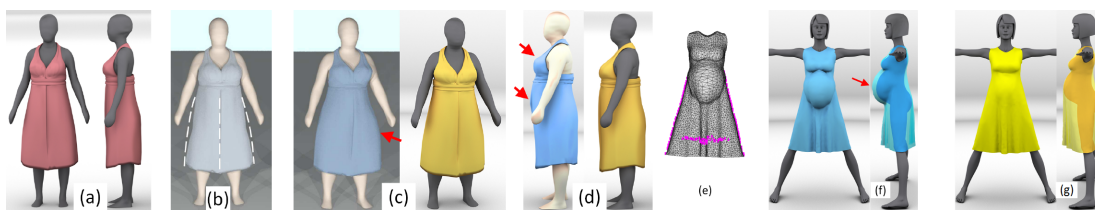
### 8.6.1 Validation

In addition to visual inspection by non-experts and amateur tailors, we had the results evaluated by a fashion designer, Nurit Perla<sup>1</sup>, who concluded that our outputs satisfy all the criteria used in traditional grading. We further validated the approach by making a dress for an actual doll (fig. 8.8). We transferred the source dress to a 3D model of the doll, which was very different proportions from the source character, and then used the resulting patterns to make the actual dress. The resulting dress maintains the design and style of the source while nicely fitting the doll’s dimensions, showcasing our method’s applicability for transferring real-life garments.

### 8.6.2 Comparison



**Figure 8.10:** Automatic transfer of a skirt from a tall model (a) to a short and stocky one (b). Scaling the skirt patterns (c) naively according to difference in height and hip circumference (d) results in uneven length and baggy skirt (e).



**Figure 8.11:** A dress a) is transferred to a fatter model using skinning [WWY05]; b) shows visible artefacts along silhouettes. cd) On this dress our automatic method (yellow) generates slightly better results than the user assisted approach of Meng et al. [MWJ12](blue) (arrows point to undesirable artefacts). When transferring a dress (fig. 8.9, top) to a more different model, such as a pregnant one, ef) using the indicated user-specified profiles, Meng et al. [MWJ12] generate sever artefacts resulting in a dress which is not physically plausible (floating around the belly). g) In contrast, our automatically generated output (yellow) reflects the input design.

<sup>1</sup>[www.talia-designer-clothes.com](http://www.talia-designer-clothes.com)

Existing transfer methods can be split into three groups: skinning based [WWY05, WHT07], axis-aligned scaling [CSMT03], and user-assisted [MWJ12]. Fig. 8.10 provides a comparison between axis-aligned scaling combined with physical simulation and our result. While the scaled skirt is baggy and has an uneven hem length, ours faithfully reflects the source design. Fig. 8.11 provides a comparison of our result to Wang’s et Meng’s works [WWY05, MWJ12]. The models were graciously provided by the authors. As clearly demonstrated by fig. 8.11, b skinning-like approaches create undesirable artifacts, avoided by our method. For similarly proportioned source and target models such as the dress in fig. 8.11, abcd, the output garments of Meng et al. and ours are of comparable quality. Note however that Meng et al. use manually traced profile curves, with a subset of those shown in the figure, to generate the output dress. Our transferred dress is generated automatically. However, close examination reveals artifacts on the dress created by the user-assisted technique, not present in our outputs. The differences are drastically more pronounced when a dress is transferred from a non-pregnant to pregnant model (fig. 8.11, ef). The result of Meng et al. retains the “skinning” effect around the belly, leading to an unnaturally looking and physically implausible result. In contrast, as demonstrated throughout the paper on this and other inputs our method can handle extreme changes in body shape and proportions, while preserving source design.

### 8.6.3 Statistics

Model	# $\Delta$	runtime	init	# iter.	# coll.
Short woman dress	8178	61s	15s	19	15
Pregnant w. dress		189s		64	57
Man dress		99s		34	18
Pregnant woman Tunic	9102	70s	16s	23	13
Man over-all	11631	181s	19s	46	34
Pregnant w. over-all		112s		28	18
Short w. t-shirt/trousers	6282/7563	14s/ 36s	10s/10s	3/12	0/9
Pregnant w. t-shirt/trousers		76s/137s		39/57	34/38
Man t-shirt/trousers		32s/32s		10/10	7/9
Girl 60s dress	11864	86s	8s	20	13
Doll 60s dress		171s		43	38
Girl layers	6274/5979	11s/10s	9s/ 9s	5/6	0/2
	5322	16s	9s	12	4
Girl flamenco	18560	76s	11s	12	5
Girl dancer top/bottom	4879/8809	89s/17s	5s/3s	38/10	32/6
Male dancer top/bottom		40s/23s		14/15	8/8
Tall dancer top/bottom		37s/21s		13/13	7/7
Fat dancer top/bottom		199s/49s		81/36	67/23

**Table 8.1:** Results statistics (left to right): model, size in triangles (shown once per source input), transfer time, reference point computation (section 8.4.1, done once per source model), number of iterations overall, and number of resolved collisions. For the flamenco dress, the number for the three layers are combined.

Table 8.1 summarizes the statistics of for our models. For typical virtual garments with five to twelve thousand triangles, our transfer process takes one to two minutes to generate the

target outfit on a standard desktop PC (Intel Core 2 Duo T6400 2GHz). The runtime largely depends on the number of collisions that need resolution.

#### 8.6.4 Limitations

The main limitation of our method is its sensitivity to the settings of the tight region tolerance, as unfortunately different input virtual models have different minimal offsets between the garment and the character, depending on the original modeling setup. Accordingly, the tightness tolerance needs to be adjusted depending on the model source. The output quality is also dependent on the quality of the cross-parameterization between the source and target characters. Lastly, our method is limited to manifold meshes; we did not find this to be a restriction, as most virtual garments are represented as such.

### 8.7 CONCLUSION

By enabling an automating garment transfer, our work greatly simplifies the generation of dressed characters, currently one of the more tedious and difficult task for computer artists. It enables transfer of both virtual and real-life garment between characters with very different body shape and proportions, automatically adapting source patterns to fit the transferred garment. Our technique lets non-expert users to adapt pre-existing wardrobes to their characters of choice, removing the tailoring knowledge barrier. It is directly applicable to clothing crowds of arbitrary characters, as it requires no per-output tweaks. The two key components of our approach are a geometric formulation and subsequent efficient optimization, of the set of criteria used in manual pattern-grading.

Although this method fully transfers garments from a character to another one, it requires some adaptation to deform garments during interactive character deformation. This extension of the method is described in the next chapter.



## CHAPTER

# 9

# TOWARDS INTERACTIVE GARMENT DEFORMATION

## 9.1 INTRODUCTION

Creating garment for a new character is a tedious task. Although the method presented in chapter 8 simplify the transfer process enabling to generate similar garments for different characters, the method has to be processed off-line, from one source character to another one.

In this chapter, we propose to adapt the method to compute the garment deformation while a user interactively deforms the character. To reach this goal, we first consider that the manipulated shape needs to be symmetric. Then, we proposed several adaptations to process garment deformation in real time on character models using skeleton-based implicit surfaces. This results into interactive garment deformation, while the character is being deformed using multitouch interaction.

## 9.2 SYMMETRIC SKELETON

To dress a character, the designer relies on 3D virtual model in the canonical pose. In this pose, the 3D shape owns a geometrical symmetry plane, which simplify garment creation. To bring this feature into the developed interface (chapter 7), we only have to symmetrize the skeleton defining an implicit character model. Indeed, the generated implicit surface will then keep the same symemetry properties.

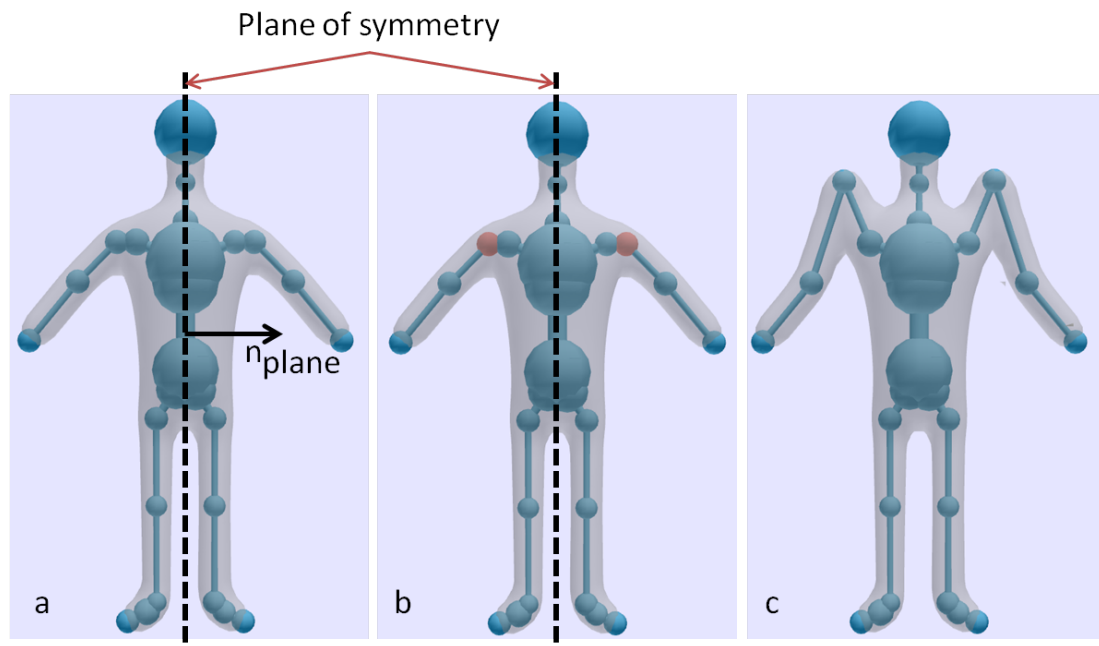
Though, there are two main concepts to symmetrize the model: geometrical symmetry and topological symmetry.

The former concept is the most standard notion of symmetry: a symmetric shape can be divided into two (or more) pieces in a way that there is a transformation that moves these pieces of the object on one another without changing the overall shape. In a human shape case, this

geometrical symmetry is a reflexional symmetry, as there is a symmetry plane dividing the shape in two mirror parts.

The latter concept - topological symmetry - is closer to the biological field. Indeed, nearly all animals, including humans, are bilateral animals, which means that the body can be divided into two parts: the left and the right halves. For instance, the left (resp. right) hand, arm, foot, leg, eye and many other subparts belong to this concept.

To transfer garments with the method proposed in chapter 8, the models (source and target) should be in the same symmetric poses. Else, the garment manufacturability and plausibility might not be preserved while transferred. For this reason, we limit the interface to geometrical symmetric shapes while deforming garments. However, an interesting research topic would be to extend the method to deform garments under topological symmetry.



**Figure 9.1:** a) for each symmetric model, a plane of symmetry is defined by its normal vector. b) then the user selects each pair of vertices defining the skeleton to enforce the symmetry. c) When a vertex is moved, its mirrored vertex also moves.

The method to handle geometrical symmetry is simple (fig. 9.1): for each symmetric object, the interface first stores the plane of symmetry. The plane is defined by its normal vector  $\mathbf{n}_{plane}$  following the equation :  $n_{plane,x}x + n_{plane,y}y + n_{plane,z}z = d_{plane}$ . Then, each vertex (or joint) in the skeleton graph is linked to another one. For vertices in the plane of symmetry, the vertex is linked to itself.

When a vertex  $v_1$  of the skeleton graph is moved, its associated vertex  $v_2$  also move, following the equation:

$$v_2 = v_1 - 2 \cdot \langle (v_1 - p_{plane}), n_{plane} \rangle \frac{n_{plane}}{\|n_{plane}\|^2} \quad (9.1)$$

where  $p_{plane}$  corresponds to any point in the plane of symmetry. If the vertex is linked to itself, a vertex  $v$  is only projected to the plane:

$$v = v - \frac{\langle v - p_{plane}, n_{plane} \rangle}{\|n_{plane}\|^2} n_{plane} \quad (9.2)$$

### 9.3 ADAPTATION OF THE GARMENT TRANSFER METHOD

The garment transfer algorithm presented in chapter 8 is a fully automatic method to transfer garments from a source character to a target one. The goal in this section is to adapt the algorithm in order to deform the garment while the character is being deformed.

A major issue to resolve is to adapt the transfer algorithm to implicit surfaces. In the previous method, character's models were defined using mesh representations and the transfer algorithm relied on a pre-computed isomorphism between the source and the target model.

The steps that we have to adapt are twofolds: the first transfer step and the collision process step. Indeed, only these two steps depend on the character's model, and therefore on the character mesh.

The former step (first transfer step) corresponds to the proportional scaling step (section 8.4). In the transfer method, the algorithm links to each garment vertex points from the object surface (skin) and from the skeleton. In our case however, the object surface is generated from the skeleton. Therefore, we only link the garment vertices to the skeleton points.

More precisely, each vertex is linked to a bone using the equation 8.1. Then the algorithm stores relevant data from the selected bone to replace the lacking data: local orientation in the bone frame, distance from the bone to the surface (defined by the bone's implicit weight at the location of the garment vertices). When the algorithm retargets the garment vertices, we simply exchange surface information by those ones.

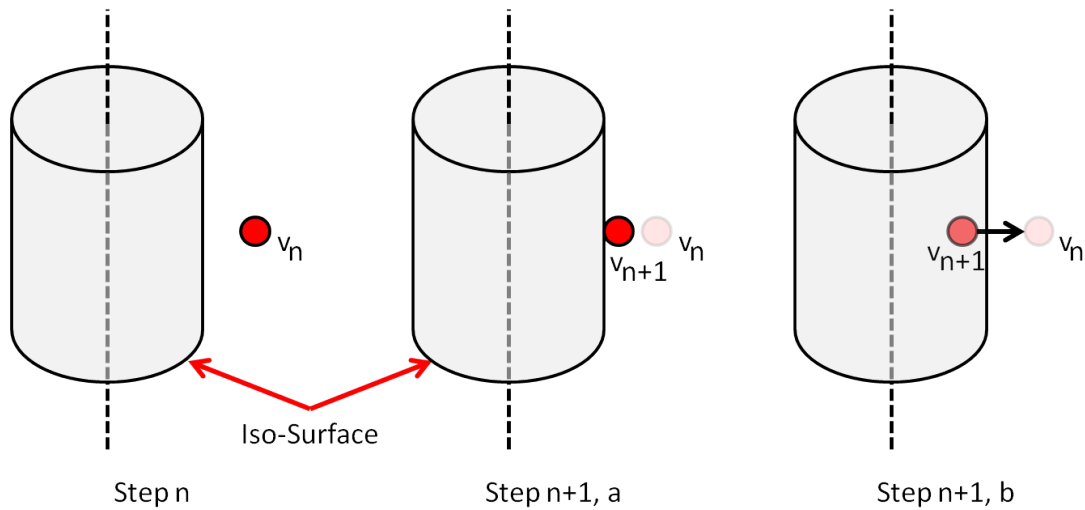
The second step to adapt (collision process step) is easier to resolve (fig. 9.2). We first rely on the field function of the implicit surface to know whether the garment vertices are inside the model or not. Then for all vertices inside the model, we push them back to their previous position, which was outside the model. Reason to push back at the previous position is to keep some loose for further process. Similar to the initial garment transfer method, we register the two highest vertices as fit (i.e. similar to the top to bottom collision process 8.5.3).

### 9.4 FIRST RESULTS

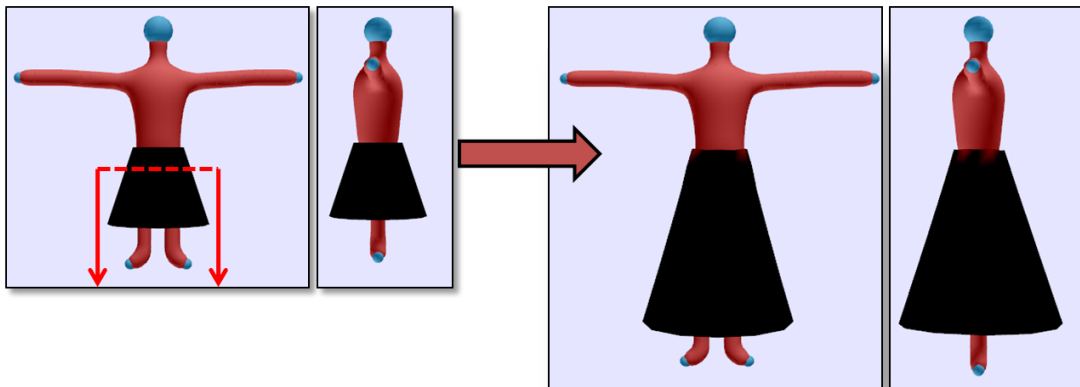
This section presents our first results on interactive garment deformation using on multitouch interface. Note that this method is still in development and requires further investigation. However, the first results are promising.

To validate our method, we first choose to deform a cone-shaped garment, enabling us to check that a cone remains a cone. Indeed, similar to garments, cones belong to developable surfaces. More, as the presented transfer techniques in section 8 relies on tailorship criteria, especially shape criteria, the resulting garment after transfer has to be a cone defined by nearly the same angle (Note that the feeling of watching longer legs in the left image is due to the perspective point of view of the camera). As illustrated in fig. 9.3, once legs are symmetrically elongated, the resulting garments is the expected cone.





**Figure 9.2:** *Adapted Collision process:* at step  $n$ , vertex  $v_n$  is outside the iso-surface of the object. at step  $n + 1$ , we check if the vertex is outside (case a), or inside (case b) the object surface by checking the field value at the vertex position. In case b, the vertex is pushed back to its previous position.

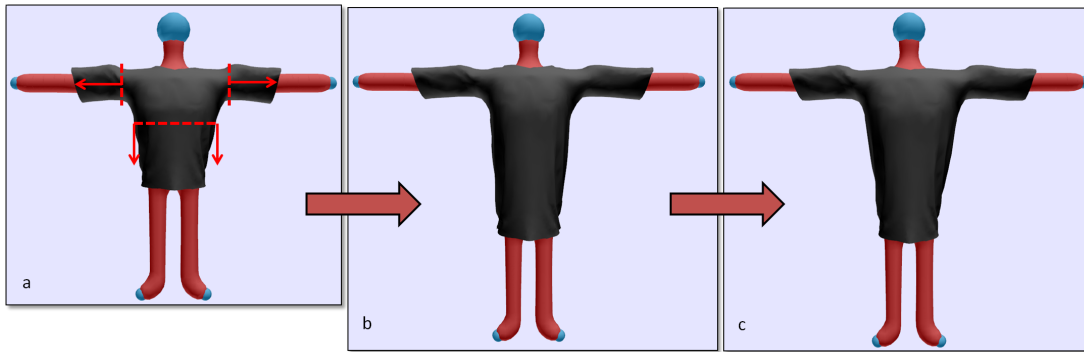


**Figure 9.3:** *Cone deformation example:* (left) From a character wearing a cone-shaped garment, the user elongates the leg from the dot line. (right) The resulting garment after deformation is still a cone.

Fig. 9.4 illustrates a more complex garment to deform and shows the adapted steps of the techniques. The middle image represents the garment after the proportional scaling step. As expected, the resulting intermediate garment still preserves proportion and scale criteria, without any collision with the character. Then, after a few iterations to restore the garment shape, using the adapted collision process, we obtain a result that fits tailorship criteria (right image).

## 9.5 DISCUSSION AND CONCLUSION

The presented technique still requires some improvements to be fully usable within our interface.



**Figure 9.4:** *TShirt Deformation example: (a) The starting pose of the character wearing a Tshirt. The elongated pieces are shown by the dot lines and arrows. (b) Garment after the proportional scaling step. (c) Deformed garment after a few iterations*

First main issue is that the presented technique is not performed in real-time yet. For instance, the Tshirt deformation required a few seconds to be processed. The step that consumes a lot of times is obviously the iterations, which alternate shape restoration iterations and collision processing. Although there is room to optimize the code itself, the more collisions are detected, and the more time the process consumes.

A clue to resolve the issue would be to update the fit regions more carefully. Currently, the process stores only one or two upper vertices that are in collision with the character after each shape restoration iteration. Using some better selection process, which still use a top-down process, would limit the number of collision detection and therefore accelerate the whole computation.

A more challenging limitation is that the process is ill-adapted when the user performs deformations other than elongations. Similar to the transfer method, character's pose should remains nearly the same (close to the canonical pose). This method enables all morphology changes (elongation or local growth), but not changes of poses. In contrast, handling garment deformation while the object is rotated is still complex and would require further investigations.

Although the garment deformation method should be improved in order to become interactive, the first results are promising. Indeed, the tailorship criteria that characterized the garment transfer method are preserved while deforming garments, and the time spend to deform will become shorter and shorter with the rise of new technologies and code optimisations. The main point to focus on is to extend the current technique to the whole set of possible deformations, while preserving the whole set of tailorship criteria.



## CHAPTER

# 10

## CONCLUSION

The main point of this thesis was the study of touch interactions in order to manipulate and deform 3D scenes and objects. This study is the basis to develop a 3D editing interface that would be easy-to use and easy-to learn, even for beginners.

To begin the investigation, in chapter 3, we first studied hand behavior on tabletop, to understand the hand limitations when constrained to remain on a surface. To successfully analyze the hand motion, we extended the standard RST technique to the whole hand, based on a temporal decomposition of gestures. Then we ran a user-study and analyzed the hand behavior of participants using this new hand analysis method. This led us to two main decompositions of hand motion: global hand gestures, and local hand gestures (or finger motions). While global hand gestures are the most stable gestures to perform, and therefore quite natural for the users of touch interfaces, local hand gestures are commonly difficult to perform (especially without training), due to the inter-dependence between fingers. This led us to conclude that the effective degrees of freedom for the hand on a surface is between 4 (global hand gestures degrees of freedom) and 6 (in the case of 3 finger interactions).

To further investigate natural interaction on tabletop, we focused the study on three modeling modes: navigation, object positioning and object deformation. In chapter 4, we mainly investigated navigation and object positioning modes through another user-study. In order to deeply understand the participants' hand behavior while performing 3D scene editing, we used the phase-based analysis method to analyze the motion and to decompose them into three main global phases: translation, rotation and scaling phases. In chapter 5, we focused the investigation on deformation tasks only. According to these two user-studies, we discovered several useful principles in order to create an interface that fulfills the goals we aimed:

- **Mode Selection:** contextual information under the two first interaction fingers is extremely relevant to inform the interface about the desired mode
- **Task Selection:** while mode selection only relies on the interaction context, the task itself involves the performed gesture (such as a rotation gesture to perform rotations).

- **Gestural Design Pattern:** using a similar pattern on each mode bring many important features for users, such as accelerating the learning of the interface.
- **Planar/non-planar tasks:** similar to the dimensional gap due to the 2D input / 3D output, users commonly distinguish planar and non-planar tasks. This distinction also induces the way of interacting with the 3D environment: one-handed gestures (or gestures assimilated to them with the two hands doing the same thing) for planar gestures and two-handed gestures for non-planar gestures.

The next step was to develop an interface that satisfies these goals while handling mode/task selection efficiently. In chapter 6, we focused on the interaction point of view, claiming that similar interaction approaches can efficiently handle any method to manipulate and deform objects. The proposed interface is easy-to use, fast to handle and accelerates recurrent tasks. In chapter 7, we describes the resolution of technical issues for developing the resulting interface, based on visuals and gesture similarities.

The next two chapters (8 and 9) presented a way to extend the interface to complex deformation through a specific example: garment deformation while the user is deforming a character's model. The chapter 8 detailed a new method to automatically transfer garments from a source morphology to any target one, based on real tailorship criteria. The chapter 9 detailed how the transfer method can be adapted to deform garments while one is deforming a model.

This thesis is representative of the current research trends between the Human-Computer interaction (HCI) and Computer Graphics (CG) field. While the computer graphics field brings more and more mathematical tools to create, manipulate, deform 3D objects, the developed methods usually lack the designer point of view to be used efficiently. Therefore, professional designers are disturbed with recent technologies and hardly use them before remanufacturing the techniques, which consumes times. Developing mathematical algorithms using user-friendly interactions is a key feature to accelerare the transition between recent technologies and professional products. In this thesis, we first gathered the users expectations to develop the interaction and the deformation methods. This was the case even for the garment transfer for which we asked professional designers to define the main criteria to be maintained. This general methodology, namely conducting preliminary user-studies, and then developing "responsive models" designed to respond in the expected way under user design gestures, is a general concept which we expect to be broadly reused in the future of our field.

# BIBLIOGRAPHY

- [ABCG05] ALEXE A., BARTHE L., CANI M. P., GAILDRAT V.: A sketch-based modelling system using convolution surfaces. *Rapport de recherche IRIT-2005-17-R, IRIT, Université Paul Sabatier, Toulouse* (2005).
- [AC02] ANGELIDIS A., CANI M.-P.: Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Proceedings of the seventh ACM symposium on Solid modeling and applications* (2002), ACM, pp. 45–52.
- [ATF12] AU O. K.-C., TAI C.-L., FU H.: Multitouch gestures for constrained transformation of 3d objects. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 651–660.
- [Bar84] BARR A. H.: Global and local deformations of solid primitives. In *ACM Siggraph Computer Graphics* (1984), vol. 18, ACM, pp. 21–30.
- [BBB87] BARTELS R. H., BEATTY J. C., BARSKY B. A.: *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1987.
- [BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: 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* (2008), ACM, pp. 151–160.
- [BDLN08] BAILLY G., DEMEURE A., LECOLINET E., NIGAY L.: Multitouch menu (mtm). In *Proceedings of the 20th International Conference of the Association Francophone d’Interaction Homme-Machine* (2008), ACM, pp. 165–168.
- [Béz] BÉZIER P.: Numerical control: Mathematics and applications. 1972. *AR Forrest (trans.) Wiley, London*.
- [Béz78] BÉZIER P.: General distortion of an ensemble of biparametric surfaces. *Computer-Aided Design* 10, 2 (1978), 116–120.

- [BFKB99] BALAKRISHNAN R., FITZMAURICE G., KURTENBACH G., BUXTON W.: Digital tape drawing. In *Proceedings of the 12th annual ACM symposium on User interface software and technology* (1999), ACM, pp. 161–169.
- [BFW\*08] BRANDL P., FORLINES C., WIGDOR D., HALLER M., SHEN C.: Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proceedings of the working conference on Advanced visual interfaces* (2008), ACM, pp. 154–161.
- [BH00] BALAKRISHNAN R., HINCKLEY K.: Symmetric bimanual interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (2000), ACM, pp. 33–40.
- [Bie87] BIER E. A.: Skitters and jacks: interactive 3d positioning tools. In *Proceedings of the 1986 workshop on Interactive 3D graphics* (1987), ACM, pp. 183–196.
- [BIF05] BENKO H., ISHAK E. W., FEINER S.: Cross-dimensional gestural interaction techniques for hybrid immersive environments. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE* (2005), IEEE, pp. 209–216.
- [BK99] BALAKRISHNAN R., KURTENBACH G.: Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (1999), ACM, pp. 56–62.
- [Bla95] BLANC C.: Generic implementation of axial deformation techniques. *Graphics Gems 5* (1995), p249–256.
- [Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)* 1, 3 (1982), 235–256.
- [BLN07] BAILLY G., LECOLINET E., NIGAY L.: Wave menus: improving the novice mode of hierarchical marking menus. In *Human-Computer Interaction—INTERACT 2007*. Springer, 2007, pp. 475–488.
- [Blo88] BLOOMENTHAL J.: Polygonization of implicit surfaces. *Computer Aided Geometric Design* 5, 4 (1988), 341–355.
- [BML12] BAILLY G., MÜLLER J., LECOLINET E.: Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *IJHCS* 70, 10 (2012), 673–689.
- [BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L.: Matisse: Painting 2d regions for modeling free-form shapes. In *Proceedings of the Fifth Eurographics conference on Sketch-Based Interfaces and Modeling* (2008), Eurographics Association, pp. 57–64.
- [BS91] BLOOMENTHAL J., SHOEMAKE K.: Convolution surfaces. In *ACM SIGGRAPH Computer Graphics* (1991), vol. 25, ACM, pp. 251–256.
- [BWKS11] BOKELOH M., WAND M., KOLTUN V., SEIDEL H.-P.: Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics (TOG)* 30, 6 (2011), 123.

- [Cad94] CADOZ C.: Le geste canal de communication homme/machine: la communication 'instrumentale'. *Technique et science informatiques* 13, 1 (1994), 31–61.
- [CBS96] CRESPIAN B., BLANC C., SCHLICK C.: Implicit sweep objects. In *Computer Graphics Forum* (1996), vol. 15, Wiley Online Library, pp. 165–174.
- [CCLH06] CHANG Y.-T., CHEN B.-Y., LUO W.-C., HUANG J.-B.: Skeleton-driven animation transfer based on consistent volume parameterization. In *Advances in Computer Graphics*. Springer, 2006, pp. 78–89.
- [CDH11] COHÉ A., DECLÉ F., HACHET M.: tbox: A 3d transformation widget designed for touch-screens. In *Proc. CHI'11* (2011), pp. 3005–3008.
- [CMS88] CHEN M., MOUNTFORD S. J., SELLEN A.: A study in interactive 3-d rotation using 2-d control devices. In *ACM SIGGRAPH Computer Graphics* (1988), vol. 22, ACM, pp. 121–129.
- [Coq90] COQUILLART S.: *Extended free-form deformation: a sculpturing tool for 3D geometric modeling*, vol. 24. ACM, 1990.
- [CR94] CHANG Y.-K., ROCKWOOD A. P.: A generalized de casteljau approach to 3d free-form deformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 257–260.
- [CSMT03] CORDIER F., SEO H., MAGNENAT-THALMANN N.: Made-to-measure technologies for an online clothing store. *IEEE Computer graphics and applications* 23, 1 (2003), 38–48.
- [DACJ\*12a] DE ARAÚJO B., CASIEZ G., JORGE J., HACHET M., ET AL.: Modeling on and above a stereoscopic multitouch display. In *3DCHI-The 3rd Dimension of CHI* (2012), pp. 79–86.
- [DACJ12b] DE ARAÚJO B. R., CASIEZ G., JORGE J. A.: Mockup builder: direct 3d modeling on and above the surface in a continuous interaction space. In *Proceedings of Graphics Interface 2012* (2012), Canadian Information Processing Society, pp. 173–180.
- [DLCB11] DELAMÉ T., LÉON J.-C., CANI M.-P., BLANCHÉ R.: Gesture-based design of 2d contours: an alternative to sketching? In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2011), ACM, pp. 63–70.
- [ES03] ELKOURA G., SINGH K.: Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), Eurographics Association, pp. 110–119.
- [FBNL09] FRANCONI J., BAILLY G., NIGAY L., LECOLINET E.: Wavelet menu: une adaptation des marking menus pour les dispositifs mobiles. In *Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine* (2009), ACM, pp. 367–370.



- [FIB95] FITZMAURICE G. W., ISHII H., BUXTON W. A.: Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1995), ACM Press/Addison-Wesley Publishing Co., pp. 442–449.
- [Fre11] FREETH M.: *Maintaining design aesthetics: case studies investigating grading for body shape variation: the translation of garment designs to fit fuller figured women: an essay presented in partial fulfilment of the requirements for the degree of Master of Design, Massey University, Wellington, New Zealand.* PhD thesis, 2011.
- [FWSB07] FORLINES C., WIGDOR D., SHEN C., BALAKRISHNAN R.: Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), ACM, pp. 647–656.
- [GBK\*02] GROSSMAN T., BALAKRISHNAN R., KURTENBACH G., FITZMAURICE G., KHAN A., BUXTON B.: Creating principal 3d curves with digital tape drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2002), ACM, pp. 121–128.
- [GM99] GOLDIN-MEADOW S.: The role of gesture in communication and thinking. *Trends in cognitive sciences* 3, 11 (1999), 419–429.
- [Gui87] GUIARD Y.: Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of motor behavior* 19, 4 (1987), 486–517.
- [HCC07] HANCOCK M., CARPENDALE S., COCKBURN A.: Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques. In *Proc. CHI'07* (2007), pp. 1147–1156.
- [HHC\*09] HANCOCK M., HILLIGES O., COLLINS C., BAUR D., CARPENDALE S.: Exploring tangible and direct touch interfaces for manipulating 2D and 3D information on a digital table. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (2009), ACM, pp. 77–84.
- [HHK92] HSU W. M., HUGHES J. F., KAUFMAN H.: Direct manipulation of free-form deformations. In *ACM Siggraph Computer Graphics* (1992), vol. 26, ACM, pp. 177–184.
- [HL07] HUOT S., LECOLINET E.: Archmenu et thumbmenu: contrôler son dispositif mobile «sur le pouce». In *Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine* (2007), ACM, pp. 107–110.
- [HSL\*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 1126–1134.
- [HVW\*06] HANCOCK M. S., VERNIER F. D., WIGDOR D., CARPENDALE S., SHEN C.: Rotation and translation mechanisms for tabletop interaction. In *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on* (2006), IEEE, pp. 8–pp.

- [HYP\*10] HINCKLEY K., YATANI K., PAHUD M., CODDINGTON N., RODENHOUSE J., WILSON A., BENKO H., BUXTON B.: Pen+ touch= new tools. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology* (2010), ACM, pp. 27–36.
- [IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 1134–1141.
- [IMT07] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Acm siggraph 2007 courses* (2007), ACM, p. 21.
- [IU97] ISHII H., ULLMER B.: Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (1997), ACM, pp. 234–241.
- [KB94] KURTENBACH G., BUXTON W.: User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1994), ACM, pp. 258–264.
- [KCST05] KRUGER R., CARPENDALE S., SCOTT S. D., TANG A.: Fluid integration of rotation and translation. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM, pp. 601–610.
- [KCŽO08] KAVAN L., COLLINS S., ŽÁRA J., O’SULLIVAN C.: Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)* 27, 4 (2008), 105.
- [KFBB97] KURTENBACH G., FITZMAURICE G., BAUDEL T., BUXTON B.: The design of a gui paradigm based on tablets, two-hands, and transparency. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (1997), ACM, pp. 35–42.
- [KGMQ08] KIM J.-S., GRAČANIN D., MATKOVIĆ K., QUEK F.: Finger walking in place (fwip): A traveling technique in virtual environments. In *Smart Graphics* (2008), Springer, pp. 58–69.
- [KH11] KNOEDEL S., HACHET M.: Multi-touch rst in 2d and 3d spaces: Studying the impact of directness on user performance. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on* (2011), IEEE, pp. 75–78.
- [KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 585–594.
- [KHT06] KLEMMER S. R., HARTMANN B., TAKAYAMA L.: How bodies matter: five themes for interaction design. In *Proceedings of the 6th conference on Designing Interactive systems* (2006), ACM, pp. 140–149.
- [KSB93] KURTENBACH G. P., SELLEN A. J., BUXTON W. A.: An empirical evaluation of some articulatory and cognitive aspects of marking menus. *Human-Computer Interaction* 8, 1 (1993), 1–23.

- [KSSCO08] KRAEVOY V., SHEFFER A., SHAMIR A., COHEN-OR D.: Non-homogeneous resizing of complex models. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 111.
- [LAFT12] LIU J., AU O. K.-C., FU H., TAI C.-L.: Two-finger gestures for 6DOF manipulation of 3d objects. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 2047–2055.
- [LBS\*11] LEAL A., BOWMAN D., SCHAEFER L., QUEK F., STILES C. K.: 3d sketching using interactive fabric for tangible and bimanual input. In *Proceedings of Graphics Interface 2011* (2011), Canadian Human-Computer Communications Society, pp. 49–56.
- [LCLJ11] LIU L., CHAMBERS E. W., LETSCHER D., JU T.: Extended grassfire transform on medial axes of 2d shapes. *Computer-Aided Design* 43, 11 (2011), 1496–1505.
- [LI10] LEE J., ISHII H.: Beyond: collapsible tools and gestures for computational design. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems* (2010), ACM, pp. 3931–3936.
- [LKC05] LATULIPE C., KAPLAN C. S., CLARKE C. L.: Bimanual and unimanual image alignment: an evaluation of mouse-based techniques. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (2005), ACM, pp. 123–131.
- [LMAJ11] LOPES P., MENDES D., ARAÚJO B., JORGE J. A.: Combining bimanual manipulation and pen-based input for 3d modelling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2011), ACM, pp. 15–22.
- [LS12] LOCKWOOD N., SINGH K.: Fingerwalking: motion editing with contact-based hand performance. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation* (2012), Eurographics Association, pp. 43–52.
- [LSS06] LAPIDES P., SHARLIN E., SOUSA M. C., STREIT L.: The 3D tractus: A three-dimensional drawing board. In *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on* (2006), IEEE, pp. 8–pp.
- [MB07] MASON A., BRYDEN P.: Coordination and concurrency in bimanual rotation tasks when moving away from and toward the body. *Experimental Brain Research* 183, 4 (2007), 541–556.
- [MCG10] MARTINET A., CASIEZ G., GRISONI L.: The design and evaluation of 3D positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on* (2010), IEEE, pp. 115–118.
- [MCG12] MARTINET A., CASIEZ G., GRISONI L.: Integrality and separability of multitouch interaction techniques in 3d manipulation tasks. *Visualization and Computer Graphics, IEEE Transactions on* 18, 3 (2012), 369–380.

- [Meh82] MEHTA N.: A flexible machine interface. *MA Sc. Thesis* (1982).
- [MFA\*14] MENDES D., FONSECA F., ARAUJO B., FERREIRA A., JORGE J.: Mid-air interactions above stereoscopic interactive tables. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on* (2014), IEEE, pp. 3–10.
- [MJ96] MACCRACKEN R., JOY K. I.: Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 181–188.
- [MMCR13] MARCHAL D., MOERMAN C., CASIEZ G., ROUSSEL N.: Designing intuitive multi-touch 3d navigation techniques. In *Human-Computer Interaction—INTERACT 2013*. Springer, 2013, pp. 19–36.
- [MMY01] MOORE C. L., MULLET K. K., YOUNG M. P.: *Concepts of pattern grading: techniques for manual and computer grading*. Fairchild Books, 2001.
- [MT97] MOCCOZET L., THALMANN N. M.: Dirichlet free-form deformations and their application to hand simulation. In *Computer Animation'97* (1997), IEEE, pp. 93–102.
- [MTLT88] MAGNENAT-THALMANN N., LAPERRIRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface'88* (1988), Citeseer.
- [MWJ12] MENG Y., WANG C. C., JIN X.: Flexible shape control for automatic resizing of apparel products. *Computer-Aided Design* 44, 1 (2012), 68–76.
- [MZL\*09] MEHRA R., ZHOU Q., LONG J., SHEFFER A., GOOCH A., MITRA N. J.: Abstraction of man-made shapes. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 137.
- [MZL11] MARTIN J., ZATSIORSKY V., LATASH M.: Multi-finger interaction during involuntary and voluntary single finger force changes. *Experimental brain research* 208, 3 (2011), 423–435.
- [NBBW09] NACENTA M. A., BAUDISCH P., BENKO H., WILSON A.: Separability of spatial manipulations in multi-touch interfaces. In *Proceedings of Graphics Interface 2009* (2009), Canadian Information Processing Society, pp. 175–182.
- [NHK\*85] NISHIMURA H., HIRAI M., KAWAI T., KAWATA T., SHIRAKAWA I., OMURA K.: Object modeling by distribution function and a method of image generation. *The Transactions of the Institute of Electronics and Communication Engineers of Japan* 68, Part 4 (1985), 718–725.
- [NW06] NOCEDAL J., WRIGHT S. J.: *Numerical optimization* 2nd.
- [Pie91] PIEGL L.: On nurbs: a survey. *IEEE Computer Graphics and Applications* 11, 1 (1991), 55–71.
- [RDH09] REISMAN J. L., DAVIDSON P. L., HAN J. Y.: A screen-space formulation for 2D and 3D direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (2009), ACM, pp. 69–78.

- [Ric73] RICCI A.: A constructive geometry for computer graphics. *The Computer Journal* 16, 2 (1973), 157–160.
- [RLG09] ROUDAUT A., LECOLINET E., GUIARD Y.: Microrolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of the 27th international conference on Human factors in computing systems* (2009), ACM, pp. 927–936.
- [RMB\*98] RESNICK M., MARTIN F., BERG R., BOROVYOY R., COLELLA V., KRAMER K., SILVERMAN B.: Digital manipulatives: new toys to think with. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1998), ACM Press/Addison-Wesley Publishing Co., pp. 281–287.
- [SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing* (2007), vol. 4.
- [SCOL\*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 175–184.
- [SFS98] SANTELLO M., FLANDERS M., SOECHTING J. F.: Postural hand synergies for tool use. *The Journal of Neuroscience* 18, 23 (1998), 10105–10115.
- [She99] SHERSTYUK A.: Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer* 15, 4 (1999), 171–182.
- [SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)* 24, 2 (2005), 311–330.
- [SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics* (1986), vol. 20, ACM, pp. 151–160.
- [SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 399–405.
- [SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision* 2, 2 (2006), 105–171.
- [SPS01] SCHKOLNE S., PRUETT M., SCHRÖDER P.: Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2001), ACM, pp. 261–268.
- [SVC\*11] SCODITTI A., VINCENT T., COUTAZ J., BLANCH R., MANDRAN N.: Touchover: decoupling positioning from selection on touch-based handheld devices. In *23rd French Speaking Conference on Human-Computer Interaction* (2011), ACM, p. 6.
- [TO02] TURK G., O’BRIEN J. F.: Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)* 21, 4 (2002), 855–873.

- [VBG\*13] VAILLANT R., BARTHE L., GUENNEBAUD G., CANI M.-P., ROHMER D., WYVILL B., GOURMEL O., PAULIN M., ET AL.: Implicit skinning: real-time skin deformation with contact modeling. *ACM Transactions on Graphics* 32, 4 (2013).
- [WGG99] WYVILL B., GUY A., GALIN E.: Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum* (1999), vol. 18, Wiley Online Library, pp. 149–158.
- [WHT07] WANG C. C., HUI K.-C., TONG K.-M.: Volume parameterization for design automation of customized free-form products. *Automation Science and Engineering, IEEE Transactions on* 4, 1 (2007), 11–21.
- [WMSB98] WANG Y., MACKENZIE C. L., SUMMERS V. A., BOOTH K. S.: The structure of object transportation and orientation in human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1998), ACM Press/Addison-Wesley Publishing Co., pp. 312–319.
- [WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The visual computer* 2, 4 (1986), 227–234.
- [WMW09] WOBROCK J. O., MORRIS M. R., WILSON A. D.: User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2009), ACM, pp. 1083–1092.
- [WW92] WELCH W., WITKIN A.: Variational surface modeling. In *ACM SIGGRAPH computer graphics* (1992), vol. 26, ACM, pp. 157–166.
- [WW11] WIGDOR D., WIXON D.: *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.
- [WWY05] WANG C. C., WANG Y., YUEN M. M.: Design automation for customized apparel products. *Computer-Aided Design* 37, 7 (2005), 675–691.
- [YSI\*10] YU L., SVETACHOV P., ISENBERG P., EVERTS M. H., ISENBERG T.: FI3D: Direct-touch interaction for the exploration of 3D scientific visualization spaces. *Visualization and Computer Graphics, IEEE Transactions on* 16, 6 (2010), 1613–1622.
- [Zan] ZANCOLLI E.: *Structural and dynamic bases of hand surgery*, 1979.
- [ZB04] ZHAO S., BALAKRISHNAN R.: Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (2004), ACM, pp. 33–42.
- [ZBQC13] ZANNI C., BERNHARDT A., QUIBLIER M., CANI M.-P.: Scale-invariant integral surfaces. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 219–232.
- [ZHS\*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large mesh deformation using the volumetric graph laplacian. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 496–503.

- [ZLL98] ZATSIORSKY V. M., LI Z.-M., LATASH M. L.: Coordinated force production in multi-finger tasks: finger interaction and neural network modeling. *Biological cybernetics* 79, 2 (1998), 139–150.
- [ZLL00] ZATSIORSKY V. M., LI Z.-M., LATASH M. L.: Enslaving effects in multi-finger force production. *Experimental Brain Research* 131, 2 (2000), 187–195.
- [Zor05] ZORIN D.: Curvature-based energy for simulation and variational modeling. In *Shape Modeling and Applications, 2005 International Conference (2005)*, IEEE, pp. 196–204.
- [ZRKS05] ZAYER R., RÖSSL C., KARNI Z., SEIDEL H.-P.: Harmonic guidance for surface deformation. In *Computer Graphics Forum (2005)*, vol. 24, Wiley Online Library, pp. 601–609.