Symposium MEIS2015:
Mathematical Progress in Expressive Image Synthesis

Editors: Hiroyuki Ochiai, Yoshinori Dobashi

MI Lecture Note Vol. 64: Kyushu University
Symposium MEIS2015:

Mathematical Progress in Expressive Image Synthesis

Editors:
Hiroyuki Ochiai
Yoshinori Dobashi
About MI Lecture Note Series

The Math-for-Industry (MI) Lecture Note Series is the successor to the COE Lecture Notes, which were published for the 21st COE Program “Development of Dynamic Mathematics with High Functionality,” sponsored by Japan’s Ministry of Education, Culture, Sports, Science and Technology (MEXT) from 2003 to 2007. The MI Lecture Note Series has published the notes of lectures organized under the following two programs: “Training Program for Ph.D. and New Master’s Degree in Mathematics as Required by Industry,” adopted as a Support Program for Improving Graduate School Education by MEXT from 2007 to 2009; and “Education-and-Research Hub for Mathematics-for-Industry,” adopted as a Global COE Program by MEXT from 2008 to 2012.

In accordance with the establishment of the Institute of Mathematics for Industry (IMI) in April 2011 and the authorization of IMI’s Joint Research Center for Advanced and Fundamental Mathematics-for-Industry as a MEXT Joint Usage / Research Center in April 2013, hereafter the MI Lecture Notes Series will publish lecture notes and proceedings by worldwide researchers of MI to contribute to the development of MI.

October 2014
Yasuhide Fukumoto
Director
Institute of Mathematics for Industry

Symposium MEIS2015:

Mathematical Progress in Expressive Image Synthesis

MI Lecture Note Vol.64, Institute of Mathematics for Industry, Kyushu University
ISSN 2188-1200
Editors: Hiroyuki Ochiai, Yoshinori Dobashi
Date of issue: 18 September 2015
Publisher:
Institute of Mathematics for Industry, Kyushu University
Graduate School of Mathematics, Kyushu University
Motooka 744, Nishi-ku, Fukuoka, 819-0395, JAPAN
Tel +81-(0)92-802-4402, Fax +81-(0)92-802-4405
URL http://www.imi.kyushu-u.ac.jp/

Printed by
Kijima Printing, Inc.
Shirogane 2-9-6, Chuo-ku, Fukuoka, 810-0012, Japan
TEL +81-(0)92-531-7102 FAX +81-(0)92-524-4411
Preface

Welcome to MEIS2015, the international symposium “Mathematical Progress in Expressive Image Synthesis”, held in Fukuoka, Japan, September 25-27, 2015. The international symposium MEIS2015 aims at giving a unique venue where various issues in CG application fields are discussed by mathematicians, CG researchers and practitioners. Through the previous conferences, MEIS2013 and MEIS2014, mathematicians as well as CG researchers have recognized that CG is a specific and practical activity derived from mathematical theories. Issues found in CG broaden the field of mathematics (and vice versa), and CG visualizes mathematical theories in an aesthetic way.

The present volume is the proceedings of MEIS2015. Several invited talks will attract and inspire prospective attendees who work in academia or industries having strong interests on digital media creations, scientific visualization and visual engineering. From the graphics community we have three outstanding invited speakers: Robert Bridson (Autodesk), Florence Bertails-Decoubes (INRIA), and Daniele Panozzo (ETH Zurich). In this year, we try to provoke interdisciplinary research projects through the peer-reviewed paper/poster presentations at the symposium. The topics includes geometry, curves and surfaces, fabrication, fluid, interpolation, illusion, texture, visualization, and rendering.

We are very much grateful to the Institute of Mathematics for Industry (IMI), Kyushu University for sponsoring this symposium. We would like to thank the Japan Science and Technology Agency (JST), Mathematics Program: Alliance for Breakthrough between Mathematics and Sciences (ABMS) on our five-year project “Mathematics for Computer Graphics” for continuous support. We also extend our thanks to Ayumi Kimura for her hard work on the conference arrangement and the production of the proceedings. Last but not least, we appreciate the hard work of the international program committee and the external reviewers in the tight schedule. Finally we wish to thank all contributing authors and attendees for their involvement. We hope all the participants enjoy this exciting event in Fukuoka.

Program Co-Chairs:
Yoshinori Dobashi
Hiroyuki Ochiai
Committee

Co-Chairs

Hiroyuki Ochiai, Kyushu University
Yoshinori Dobashi, Hokkaido University

International Program Committee

Ken Anjyo, OLM Digital, Inc.
Jean-Marie Aubry, Weta Digital
Bernd Bickel, Disney Research
Yasuhide Fukumoto, Kyushu University
Sunil Hadap, Adobe Research
Kei Iwasaki, Wakayama University
Shizuo Kaji, Yamaguchi University
Kenji Kajiwara, Kyushu University
Miyuki Koiso, Kyushu University
J.P. Lewis, Victoria University of Wellington / Weta Digital
Yoshihiro Mizoguchi, Kyushu University
Yoshiyuki Ninomiya, Kyushu University
Yutaka Ohtake, The University of Tokyo
Makoto Okabe, The University of Electro-Communications
Daisuke Tagami, Kyushu University

External Reviewers

Alexandre Derouet-Jourdan, OLM Digital, Inc.
Committee Co-Chairs
Hiroyuki Ochiai, Kyushu University
Yoshinori Dobashi, Hokkaido University

International Program Committee
Ken Anjyo, OLM Digital
Jean-Marie Aubry, Weta Digital
Bernd Bickel, Disney Research
Yasuhide Fukumoto, Kyushu University
Sunil Hadap, Adobe Research
Kei Iwasaki, Wakayama University
Yasuhide Kaji, Yamaguchi University
Kenji Kajiwara, Kyushu University
Miyuki Koiso, Kyushu University
J.P. Lewis, Victoria University of Wellington / Weta Digital
Yoshihiro Mizoguchi, Kyushu University
Yoshiyuki Ninomiya, Kyushu University
Yutaka Ohtake, The University of Tokyo
Makoto Okabe, University of Electro-Communications
Daisuke Tagami, Kyushu University

External Reviewers
Alexandre Derouet-Jourdan, OLM Digital

Computer graphics
Discrete differential geometry
Lie theory
Computational fluid dynamics
Graph theory
Function interpolation
Learning theory


Nishijin Plaza, Kyushu University, Japan
Contact: Institute of Mathematics for Industry, Kyushu University
TEL: +81-92-802-4402 E-mail: kyoto_ryou@imi.kyushu-u.ac.jp
Symposium Co-Chair:
Hiroyuki Ochiai, Kyushu University
Yoshinori Dobashi, Hokkaido University
http://mcg.imi.kyushu-u.ac.jp/meis2015/

 участие

免費

平成27年度 九州大学マス・フォア・インダストリ研究所 共同利用研究集会
「デジタル映像表現のための数理的手法－交流と創出－」
会場／九州大学 西新プラザ（福岡市早良区西新 2-16-23）
# Table of contents

- Geometry and Mechanics of fibers: Some numerical models* ............................................. 1  
  Florence Bertails-Descoubes (INRIA)

- Active Comicing for Freehand Drawing Animation .......................................................... 6  
  Tsukasa Fukusato (Waseda University) and Shigeo Morishima (Waseda Research Institute for Science and Engineering)

- Visual Media Culture Supported by Human Depth Illusion* .......................................... 16  
  Kokichi Sugihara (Meiji University/JST, CREST)

- Finite-dimensional RKHS for Solving Computer Graphics Problems .............................. 23  
  Ken Anjyo (OLM Digital, Inc./JST, CREST)

- Sparsity and Information Processing* ............................................................................. 31  
  Shiro Ikeda (The Institute of Statistical Mathematics)

- Tetrisation of triangular meshes and its application in shape blending ......................... 33  
  Shizuo Kaji (Yamaguchi University/JST, CREST)

- A construction method for discrete constant negative Gaussian curvature surfaces .................. 43  
  Shimpei Kobayashi (Hokkaido University)

- Revisiting vorticity: pushing fluid solvers to the next level* ......................................... 51  
  Robert Bridson (Autodesk)

- A Simple Method for Morphing Smoke ........................................................................... 54  
  Yoshinori Dobashi (Hokkaido University/JST, CREST), Tsubasa Tani (Hokkaido University), Syuhei Sato (UEI Research), Tsuyoshi Yamamoto (Hokkaido University)

- Wang Tiles Modeling of Wall Patterns ........................................................................... 61  
  Alexandre Derouet-Jourdan (OLM Digital, Inc./JST, CREST), Yoshihiro Mizoguchi (Kyushu University/JST, CREST), Marc Salvati (OLM Digital, Inc.)

- High resolution visualization library for Exa-scale supercomputer ................................... 71  
  Yoshitaka Wada (Kindai University), Kohei Murotani (University of Tokyo), Masao Ogino (Nagoya University), Hiroshi Kawai (Tokyo University of Science), Ryuji Shioya (Toyo University)

- Production Monte Carlo Rendering* .............................................................................. 80  
  Ryusuke Villemin (Pixar Animation Studios)

- A Multilayered Model for Artificial Intelligence of Game Character as Agent Architecture* .............................................................. 89  
  Youichiro Miyake (Digital Game Research Association JAPAN)
dNLS Flow on Discrete Space Curves .................................................................93
Sampei Hirose (Shibaura Institute of Technology), Jun-ichi Inoguchi (University of Tsukuba), Kenji Kajiwara (Kyushu University), Nozomu Matsuura (Fukuoka University) and Yasuhiro Ohta (Kobe University)

Aesthetic Design with Log-aesthetic Curves and Surfaces* ...........................................103
Kenjiro T. Miura (Shizuoka University) and R. U Gobithaasan (University Malaysia Terengganu)

Fabrication-Aware Geometry Processing* .................................................................113
Daniele Panozzo (ETH Zurich)

Drawing Curves* ..................................................................................................117
Toshio Oshima (Josai University)

Attractive plane curves in Differential Geometry* .................................................121
Jun-ichi Inoguchi (University of Tsukuba)

* Invited paper
Geometry and Mechanics of fibers:  
Some numerical models

Florence Bertails-Descoubes *

INRIA and Laboratoire Jean Kuntzmann (Grenoble University, CNRS)

(Joint work with Romain Casati, Gilles Daviet, and Alexandre Derouet-Jourdan)

Abstract  In this talk I will give an overview of our work on the simulation of fibers and entangled materials, such as hair, with a specific interest for virtual prototyping and computer graphics applications. I will first introduce a family of high-order, reduced models for discretizing Kirchhoff’s equations for thin elastic rods in a both faithful and robust way. Such models are particularly well-suited for simulating inextensible fibers subject to bending and twisting, and featuring an arbitrary curly resting geometry. Then I will show how such models can be coupled to frictional contact using the nonsmooth contact dynamics framework, and I will present a hybrid iterative solver suitable for robustly handling thousands packed fibers at reasonable frame rates. Finally, I will give some insights into the inverse modeling of fibers, consisting in taking an arbitrary curve geometry as input and inferring corresponding geometric and physical parameters of the simulator such that the input geometry corresponds to a stable configuration at equilibrium.

Keywords:  Physics-based simulation, thin elastic rod, frictional contact, hair simulation, inverse physics-based design

Introduction  Deformable slender structures such as hair fibers, cloth, ribbons, tree branches or leaves, are ubiquitous around us. They often feature an intricate natural shape, ranging from straight to curly, and are characterized by a complex motion involving strongly nonlinear deformations, such as buckling. Such complex shapes and motions greatly contribute to the visual richness of the real world. When multiple such structures are coupled together with contact and friction, the range of emerging phenomena is even more exacerbated, giving rise to stick-slip dynamical instabilities, entangling, or spontaneous collective behavior. Human hair, which is typically composed of 150,000 thin fibers, beautifully depicts such complex mechanical behaviors when fluttering in the wind.

*Florence.Descoubes@inria.fr
As the essence of Computer Graphics is to represent the visual appearance of the real world with the highest fidelity, it is important for practitioners to be able to capture all the relevant details of a given scene. In the case of a dynamical scene involving passive objects such as hair, cloth or natural phenomena, physics-based simulation has proven over the years to be a method of choice for capturing resulting visual effects. Unlike phenomenological methods which develop descriptive models for reproducing a given emerging phenomenon, physics-based methods provide generative models whose goal is to explain the physical causes of the phenomenon. From a set of initial conditions as well as a few physical parameters (e.g., the mass, stiffness, natural shape), a physics-based simulator may generate not just a single effect, but a wide range of emerging phenomena revealing the whole complexity of the underlying physics.

When designing a physics-based simulator for computer graphics, one has inevitably in mind the four following criteria:

**Realism:** Ingredients which are necessary to capture relevant visual effects should be identified, and translated numerically with as few quality loss as possible.

**Robustness:** The simulator should converge properly for a subsequent range of parameters.

**Efficiency:** The simulator should be fast enough for allowing complex scenes to be simulated in reasonable timings (in our case, a few days of computation for a given scene is an upper-bound).

**Control:** The simulator should provide the user with some handles to control the shape and motion of the object in an intuitive way.

Over the ten past years, we have been striving to develop some numerical models satisfying all four criteria at the same time. Our work has focused on the simulation of slender structures prone to contact and dry friction, and especially on the dynamics of thin fiber assemblies, with some direct applications in hair simulation and inverse physics-based design.

**High-order reduced models for simulating dynamic fibers** The first part of the talk will be devoted to the presentation of the numerical models that we have been developing for simulating the dynamics of flexible fibers, the so-called super-helix and super-clothoid models [1, 2, 4].

I will first introduce the mechanical equations for inextensible thin elastic rods, namely the Kirchoff equations, which take the form of second order partial differential equations subject to boundary conditions. Noting that curvatures and twist play a major role both in the geometric and dynamic description of this model, we have come up with a spatial rod discretization based on elements that are polynomial in such quantities.

Our first scheme relied on piecewise constant curvatures [1], and was then extended to piecewise linear curvatures [2, 4]. One major advantage of such curvature-based formulations is that the kinematics of the discretized rod remains, by construction, perfectly inextensible. Such intrinsic inextensibility thus removes the burden of adding subsequent (stiff) inextensibility constraints when solving the dynamics. The price to pay, however, is that the geometry of the rod is not readily available but has to be computed iteratively from the curvatures.

In the piecewise constant case [1], each element turns out to be a perfect circular helix (hence the super-helix name for the model), leading to a cheap and exact evaluation of all the terms of the discrete dynamic equations. For higher orders however, one unfortunately loses such a closed-form formula and a both accurate and efficient spatial integration scheme has to be designed.

In the piecewise linear case (where each element takes the form of a 3D clothoid), we were able to build an accurate integration scheme which proved to be orders of magnitude faster compared to
The key of our approach was to leverage the form of the solution as a power series expansion, while avoiding the pitfall of catastrophic cancellation through an adaptive integration strategy. With this tool in hand, we were able to demonstrate that the super-clothoid model could capture intricate shapes both robustly and efficiently, with better spatial accuracy and geometric fairness compared to state-of-the-art methods (see figure 1).

Figure 1: Many physical strands exhibit a smooth curled geometry with linear-like curvature profile, which is captured and deformed accurately thanks to our super-clothoid model [4]. From left to right and top to bottom, three examples of real strands whose shapes are synthesized and virtually deformed in real-time using a very low number of 3D clothoidal elements: a vine tendril (4 elements), a hair ringlet (2 elements), and a curled paper ribbon (1 single element). Left photograph courtesy of Jon Sullivan, pdphoto.org.

**Robust frictional contact model for fiber assemblies** The second part of the talk will be focused on the dynamic simulation of fiber assemblies, where individual fibers are coupled to each other through contact and friction.

I will first illustrate why capturing threshold effects in friction is key to realism (see figure 2), before introducing the nonsmooth Signorini-Coulomb friction model and its various formulations. We shall notably see what the numerical counter-parts are for each main formulation, and how each of them performs in terms of efficiency, robustness, and scalability [3].

Then I will explain how we managed to design a robust and scalable frictional contact solver by combining an iterative Gauss-Seidel strategy together with an extremely robust one-contact solver [5]. Our global solver proved to converge well in scenarios involving thousands fibers subject to tens thousands frictional contact points, and thus allowed us to enhance considerably the realism of hair simulations. Our method has been adopted by the special effects industry for simulating hair and fur accurately [10].
From geometry to physics: Inverse design of fiber assemblies

Finally, I will present some new important challenges regarding inverse physics-based design. While current simulators may succeed in reaching a good level of realism, they remain difficult to control in order to achieve a precise artistic goal. More precisely, to generate some desirable shapes and motions, one should be able to feed a simulator with the “right” parameters. Finding such parameters remains a very difficult task, which is often performed through a tedious trial and error process. To make this task fully automatic, we have started looking at inverse solutions in the case where a static shape is provided as input: the inverse model should be able to interpret automatically this shape as a stable equilibrium of the simulator, under gravity and other external forces such as contact and friction.

In the case of an isolated fiber, we have shown that inverting any of our super-model [1, 4] boils down to two decoupled problems that are both easy to solve [7, 8]: first, an equilibrium condition which appears to be linear in the natural shape of the fiber, thanks to the curvature-based parameterization of our fiber models; second, a sufficient stability condition that can be simply set by fixing a lower-bound for the ratio of stiffness over mass. Actually, the only remaining difficulty is to solve a merely geometric fitting problem – converting a curve as a piecewise helix or clothoid. In the case of helical fitting, we have already brought some efficient and robust solutions [7, 9].

In the presence of contact and friction, Coulomb sticking constraints have to be considered, which makes the overall inverse problem nonsmooth and ill-posed. We have shown that assuming known mass and stiffness and a simplified inverse model, it is possible to recover a plausible natural shape as well as frictional contact forces at play [6]. This work allowed us, for the first time, to animate in a plausible way a few hair geometries stemming from recent hair captures (see figure 3).

![Figure 3: Real curly wig (a) captured from [11], inverted by our method in [6] and physically animated (b) and trimmed (c).](image)

Conclusion

Throughout this long-term work on the numerical modeling of fibers and frictional contact, we have learned that systematically concentrating the efforts on the upstream modeling and formulation of problems often pays off: even for very complex problems, the resulting numerics may be greatly simplified and thus solved more easily and robustly. Keeping in mind this key lesson, we are starting to investigate the case of 2D slender structures (namely plates and shells), for which many exciting challenges remain open.
References


Active Comicing for Freehand Drawing Animation

Tsukasa Fukusato*  
Waseda University

Shigeo Morishima†  
Waseda Research Institute for Science and Engineering

Abstract  This paper presents Active Comicing, a prototype sketching system that provides enhanced frame interpolation capability for freehand drawing animation. In this system, the user draws several 2D freeform strokes interactively on multiple frames, and the system automatically constructs stroke-to-stroke interpolation frames. To compose a comprehensive and coherent least-distorting interpolation, we assume input stroke has ghost points, which are additional points defined on stroke edges, and define affine transformations. In addition, the system semi-automatically guides the template motion of each stroke. For example, if the user draws an arrow, the system assigns the stroke moves in the direction of the arrow. To assign template motion, we compute the stroke similarity between the user’s input and stroke information from a database. With this method, it is possible to generate stroke animation on each frame without stroke interpolation. By combining these techniques, the user can generate freehand animations easily and quickly.

Keywords:  As-Rigid-As-Possible Stroke Interpolation, Stroke Matching, Interactive Drawing

1 Introduction

2D freehand animation enables viewers to intuitively experience artistry and feeling. Among these techniques, GIF animation (e.g., LINE’s stamp and Twitter icon) has attracted worldwide attention in social networks. To present a worldview using 2D freehand animation, anime-like techniques such as flip books and motion comics are employed. However, the creation of 2D freehand animations has always been a time-consuming and skill-demanding process. Software such as Adobe Photoshop provides some assistance by creating animations from a small number of key frames and generating in-between frames automatically. For example, animation software might deform a shape using handles or transform simple geometric primitives. However, creating the numerous key-frames of animation, such as those in a flip book, requires significant skill and time.

Conversely, the cartoon animation industry tends to shift traditional hand-drawn techniques to a pipeline using parameterized 3D models. Although a 3D model technique reduces production costs, this approach comes at the expense of well-established cartoon animation values, such as character and expression. These models may diminish freedom, expressiveness, and the artist’s commitment to the characters. It is difficult to parameterize the freedoms of pencil and paper. In short, many amateur animators, including children, find it difficult to create freehand drawing animation.

*tsukasa@moegi.waseda.jp  
†shigeo@waseda.jp
Our goal is to create new shape from freehand drawing 2D shapes, sketched onto a drawing interface. In this paper, we present Active Comicing, a sketch-based interface that allows users to interpolate freehand drawing strokes without a skeleton (i.e., graffiti animation). Figure 1 shows a freehand stroke animation on a display-integrated tablet. Specifically, we have implemented a freehand stroke interpolation method, based on As-Rigid-As-Possible Stroke Interpolation, which does not require editing commands or special interaction modes. To address the vertex correspondence problem, in which correspondence between input strokes must be established, our system constructs a simple layer structure. To reduce the distortion of the in-between shapes, we compute the orientation information for stroke vertices (stroke triangulation). Moreover, this system allows users to edit the animation path of each stroke using a template motion database. As a result, we can easily create a simple 2D freehand drawing animation.

The reminder of this paper is organized as follows. Related works are reviewed in Section 2. We discuss the user interface in Section 3, and describe the main ideas underlying the proposed method’s algorithms in Section 4. In Section 5, we describe the implementation details of our prototype system. We conclude the paper and discuss limitations and future work in Section 6.

![Prototype drawing system with Active Comicing, on a display-integrated tablet.](image)

## 2 Related Works

Recently, 3D computer graphics researchers have proposed generating 3D models from 2D drawings. In particular, Teddy [1] inflates the stroke region surrounded by a silhouette. River et al. [2] propose a method to create 2.5D models, which are hand-drawn illustration style models. Although their method can automatically estimate the depth information of each polygon section, users must configure Z-ordering and create each section from scratch. It is difficult to parameterize freehand drawings created with pencil and paper.

Drawing is simple tool that reflects the artist’s creative sense. Pencil lines or brush coloring can express rich emotions or subtle charms. Live2D [4] can create rich animations based on standard linear interpolation (point-to-point interpolation) while keeping the original charms intact. However, it is necessary to determine the character pose on the frame or redraw some or all of the strokes in the model manually. In addition, significant time is required to create a mesh structure and edit mesh deformations. Cambell [5] and Baxter [6] propose intuitive approaches, whereby line drawings are interpolated in a pose-space with reduced dimensions. This method enables the subspace of a pose to be browsed. Unfortunately, it is limited to line drawings with the same number of lines, and may give unnatural results because curves are linearly interpolated.
While physically based simulations such as the mass-spring model [7] can also be used for this purpose, it tends to be slow and produces unstable shapes. Wang’s approach [8] enables image deformation based on meshless rigid shape matching [10]. Moreover, Sykora et al. [12] apply this approach to elasticity-inspired character registration. With this method, it is possible to register images undergoing large free-form deformations and appearance variations. Unfortunately, they cannot directly obtain pixel or sub-pixel precision, because they embed the image into a coarse lattice. Although this method can apply a multi-scale extension, increasing the number of squares makes the overall iterative process ineffective.

In shape interpolation research, As-Rigid-As-Possible Shape Interpolation approaches (ARAP) have been studied [13][14][15][16][17][18]. These methods enable the volume of the stroke’s interior to be maintained and produce more plausible animations by using triangle mesh structures. Furthermore, Baxter [19] has extended this method to examples-way rigid interpolation. However, this system uses polygonal boundary-based triangulation; that is, it focuses only on similarity shape morphing. In addition, it is difficult to edit the mapping of each stroke. Alexa [23] propose laplacian coordinate for shape interpolation; however, the morphing results have shrinkage. Sederberg [26] exploits intrinsic blending on a basis of interpolating the respective vertex angles and edge length. Moreover, Whited [27] develops BetweenIT, a technique for stroke interpolation from two key frames. This technique combines stroke motions constructed from logarithmic spiral vertex trajectories with stroke deformations based on curvature averaging and twisting warps. This system provides a context in which the user can guide the system in a natural manner to produce quality results efficiently. However, this system only focuses on tight in-betweens, which are drawn between two key frames that are very similar in shape.

Other approaches have processed more general shapes by considering deformations of a template model. For example, Igarashi’s Spatial Keyframing [28] animates 3D objects composed of skeletons. Moreover, applying motion capture data to a single character image based on a skeleton has been studied [29][30]. However, the range of deformation is limited with these approaches. In addition, these approaches do not specify how handles should be interpolated to achieve plausible interpolations. In contrast, Sumner’s [31] Mesh Inverse Kinematics system interpolates between multiple meshes. However, a non-linear inverse kinematics approach is not browsed directly.

To summarize, previous animation techniques and tools have restrictions on the types of input strokes that can be used for similarity stroke morphing, and it is necessary to redraw some or all of the strokes in the model. Therefore, we propose a method to interpolate freehand-drawing strokes. The proposed method is of great value, and can create a simple animation interactively.

Figure 2: System overview.
3 User Interface

ActiveComicing’s physical user interface utilizes traditional 2D input devices such as a standard mouse and pen tablet. Figure 2 shows an overview of our system. In drawing mode, a user draws several 2D freeform strokes interactively on some key frames. This system has various user drawing functions such as image (e.g., jpg image file) loading functions and key frame copy function. The user can also redraw some or all of the strokes on each frame. Moreover, using onion skinning, the user can make decisions on how to create key frames based on the previous key frames in the sequence.

A stroke consists of a sequence of points on the plane, which we call stoke vertices. The stroke vertices are interpolated using a centripetal Catmull-Rom spline. In editing mode, the user can transform the strokes to edit the \( xy \)-coordinate of each stroke vertex; this translation is performed by dragging the mouse. The system automatically assigns labeling numbers (or layer number) to the strokes on each key frame based on the stroke order. The input strokes on one frame spatially correspond to those on another frame based on the labeling number; stroke-to-stroke correspondences are defined. Moreover, the user can edit the layering order by dragging and dropping layers with the mouse.

The user can easily generate a freehand stroke animation as a GIF image, using the provided animation timeline in animation mode.

4 Algorithm

In this section, we introduce the algorithm that creates the stroke animation between frames. As-Rigid-As-Possible Stroke Interpolation is described in Section 4.1; template motion blending is discussed in Section 4.2.

4.1 Stroke Interpolation Method

To interpolate two frames, the corresponding strokes have to have the same number of stroke vertices. The source strokes are first resampled to the number of target stoke vertices \( n \) equidistantly. Let \( P = (\vec{p}_0, \cdots, \vec{p}_n) \) be the source stroke and \( Q = (\vec{q}_0, \cdots, \vec{q}_n) \) be the target stroke.

For 2D interpolation technique, Sederberg [25] proposes a solution to the vertex correspondence problem, and the vertex path problem is dealt with in Sederberg [26] method, which interpolates the edge lengths and the angles between consecutive edges of polygonal curves. To ensure these blended strokes are closed without local self-intersection, they set to an equality constraint of the end positions by tweaking the edge length only; however, the final morphing results are dependent on the computation order of dihedral angels and edge length. Moreover, they cannot add some constraints, and extend this method to an invariant interpolation under similarity transformation, i.e., rotation and scale. Most shape interpolation and deformation studies have focused on 2D or 3D triangle mesh because the affine transformation of each triangle polygon can be computed easily. However, this approaches focus only on triangle mesh structure and do not determine a vertex path for stroke interpolation. Baxter et al. [19] apply a Delaunay triangulation to 2D stroke vertices, and then deform the Delaunay triangles based on ARAP. Unfortunately, the Delaunay triangulation approach is focused only on a closed stroke, and is less intuitive for interpolations between closed strokes. Specifically, it is difficult to define an affine transformation based on the source and the target stroke vertices only.
Therefore, we assume that a each stroke vertex \( \vec{v}_{\text{i}} \) has a ghost vertex \( \vec{v}_{\text{i}}^g \in \mathbb{R}^2 \), which is placed on a certain distance along normal direction of each adjacent edge (as shown in Figure 3). Representing the ghost vertex is mainly inspired by Umetani’s ghost point approach [32] and Sumner’s surface tetrahedra [33]. Using the ghost vertex, we generate the triangles of the source and the target strokes in order to compute a unique affine transformation of each edge. For the corresponding triangle in each stroke shape, the system first computes the ghost vertex \( \vec{v}_{\text{i}}^g \) of each stroke vertex (\( R_{90} \) denotes rotation matrix by 90 degrees):

\[
\vec{v}_{\text{i}}^g = \vec{v}_{\text{i}} + R_{90} \left( \frac{\vec{v}_{\text{i+1}} - \vec{v}_{\text{i-1}}}{2} \right)
\]

(1)

As the result, the source and the target stroke can consist of a chain of triangles. Then, we focus on ARAP interpolation of local and global linear transformation. An affine mapping represented by matrix \( A \) transforms the source into the target triangle. The matrix \( A \) is parameterized by time \( t \in \mathbb{R} \) such that \( A(0.0) = I \) (identity matrix) and \( A(1.0) = A \). We next deal with the 2D interpolation of the entire input strokes (the source and the target triangles). To compute a global transformation \( B_i(t) \) based on local translation \( A_i(t) \), we use Kaji’s local error function using the polar decomposition and the exponential map [16] as follows:

\[
A_i(t) = R_{\delta} \cdot \exp(t \log S)
\]

(2)

\[
E_i^R(A_i(t), B_i(t)) = \min_{s, \delta \in \mathbb{R}} \| sR_{\delta}A_i(t) - B_i(t) \|_F^2
\]

(3)

where \( R_{\delta} \) is a rotation matrix, and \( s \) is scale value. This equation measures how different \( A_i(t) \) and \( B_i(t) \) are as affine transformations of \( i \)th triangle. With the local error functions for each triangle, we combine them into a single global error function. If a local affine transformation can be formed by reflections, we exclude an error value of local triangle distortion from the global error function. The global error function is a positive definite quadratic form. Instead, in-between parameter \( t \) requires the solution of a linear system of equations.

In addition, using the ghost vertex, we can unify the stroke’s global orientation into a counterclockwise orientation. We compute a sign area \( S \) as follows:

\[
S = \frac{1}{2} \sum_{i} (v_{ix}v_{(i+1)y} - v_{(i+1)x}v_{iy})
\]

(4)
This computation gives a positive signed area $S$ for a simple stroke (non-self-intersecting polygon) ($S > 0$) when the vertices are oriented counterclockwise around the polygon, and negative ($S < 0$) when oriented clockwise. However, this equation cannot be applied to complex strokes (self-intersecting polygon). It is necessary to split the complex strokes into several simple strokes.

For interpolating the line weight and RGBA color information of each stroke, we used standard linear interpolation. The system allows us to set the in-between parameter $t$ for linear and interactive interpolation of each stroke shape. The acceptability of the computation time depends on the shape and the desired application. Figure 4 illustrates the resulting transformations from a source to a target shape. For comparison, Figure 4(a) shows Sederberg’s method [26] of each vertex coordinate with $t = 0.5$, and Figure 4(b) shows laplacian morphing method [23]. Our transformation (ARAP with ghost vertex) is depicted in Figure 4(c). The results show that we have successfully reduced distortions in stroke shape transformations. In addition, we can incorporate some constraints into the global error function.

### 4.2 Template Motion Blending

In this section, we describe a method to animate strokes based on template effects, such as those in *Microsoft PowerPoint*. The template effects consist of affine transformations (e.g., translation matrix $T$, scaling matrix $S$, and rotation matrix $R$) and alpha blending. This system is formed with the origin at the centroid vertices of each stroke. By setting the stroke motion matrix (e.g., the animation path), we can interactively edit the results of the stroke animation.

In addition, we attempt to synthesize the template effects automatically. For example, when the user draws an *arrow* shape, our system assigns the stroke moves in the direction of the arrow. First, we assume that the user assigns the same effect to similar shape strokes. Therefore, we propose a technique to compute the similarity between strokes, and recommend optimum template effects based on the stroke database, which contains sets of stroke and template effect. We compute the degree of similarity between the input stroke $\vec{v}$ and the database stroke $\vec{d}$, and present template effects of highly similar strokes from the database. In image processing research, stroke similarity has been proposed. Ip et al. [34] propose affine-invariant stroke features based on stroke area and angle information. Because a single shape signature (in the form of a nineteenth-dimension histogram) records the stroke area and angle information, their similarity between stroke shapes can be computed efficiently using a signature difference. However, this approach, known as histogram
intersection, does not consider stroke’s global orientation, and has difficulty representing complex stroke shapes. Therefore, to compute stroke similarities, we use rigid shape matching [8] [10] based on the centroid position of the input stroke $v_{cm}$ and the database stroke $d_{cm}$. We define the quadratic error function between the stroke vertices $\vec{p}_i = \vec{v}_i - v_{cm}$ and $\vec{q}_i = \vec{d}_i - d_{cm}$ as follows:

$$E = \sum_i |\vec{p}_i - sR \cdot \vec{q}_i|^2$$

$$R = A_{pq}S^{-1} = A_{pq}(\sqrt{A_{pq}^T A_{pq}})^{-1}$$

$$A_{pq} = \sum_i \vec{p}_i \cdot \vec{q}_i^T$$

where $n$ is the number of stroke vertices, $s$ is the normalized value ($s = \sum_i |\vec{p}_i|/|\vec{q}_i|$). The optimal rotation $R$ is the rotation portion of $A_{pq} = RS$; we compute rotation matrix $R = A_{pq}S^{-1}$, where the symmetric portion is $S = \sqrt{A_{pq}^T A_{pq}}$. The output value $E$ provides the dissimilarity value because the value tends to be smaller if two signatures are more similar. However, it is essential to work to have the same number of vertices in a stroke. In this paper, the database strokes are resampled to the number of input stroke vertices during pre-processing.

To evaluate the performance of our similarity for stroke retrieval, we perform an experiment. The experiment is carried out to retrieve relevant strokes based on the users’ sketching of the desired stroke with a stroke database of 20 strokes. The evaluation of the approach is based on retrieval accuracy (precision rate). The stroke retrieval results are shown in Table 1. For comparison, we use Ip’s affine-invariant histogram approach [34]. These results show that our similarity can determine highly accurate animation template motions. Moreover, we add an editing function for relearning moving guidance. By adding $(k + 1)$’s editing data (a set of stroke and template motion) to the stroke databases, it is possible to obtain a more suitable optimum stroke motion.

<table>
<thead>
<tr>
<th>Number of strokes</th>
<th>Our Approach (%)</th>
<th>Ip et al. 2002 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>90.0</td>
<td>50.0</td>
</tr>
<tr>
<td>15</td>
<td>75.0</td>
<td>40.0</td>
</tr>
<tr>
<td>20</td>
<td>70.0</td>
<td>35.0</td>
</tr>
</tbody>
</table>

5 Implementation

Our prototype system is written using openFrameworks, an open source C++ toolkit. A 64bit Windows PC (Intel® CoreTM i7-3770 CPU@3.40GHz 8GB RAM; NVIDIA GeForce GT 620M 1GB) is used. By drawing freehand strokes on some frames (max number = 4), users can easily generate animation. Although all results are generated at over 30.0 fps, it is difficult to accurately measure the performance of each computation. Our results are presented in Figures 5 and 6.

The display-integrated tablet version of Active Comicing has been used to create different 2D freehand animations, mainly by computer graphics researchers and students. Our prototype system was also evaluated by users who provided individual feedback: One user stated that the hand drawing animation capabilities were more impressive and expressive than normal CG animation techniques. Other users commented that they wanted to upload information results to LINE or Twitter, and that the system could benefit from a more elastic function for editing stroke shapes. In the future, we
Figure 5: Selected examples of deformable stroke animation, 'muscle training,' produced by our technique. (a) $t = 0.0$ (b) $t = 0.25$ (c) $t = 0.5$ (d) $t = 0.75$ (e) $t = 1.0$.

Figure 6: Selected examples of deformable stroke animation, 'girl's looking back motion,' produced by our technique. (a) $t = 0.0$ (b) $t = 0.25$ (c) $t = 0.5$ (d) $t = 1.0$.

plan to include shape deformation functions such as physical simulation in the user interface to create richer animations.

6 Conclusions and Future works

We have presented a method to interpolate and animate freehand drawing strokes. The prototype system, Active Comicing, enables the easy creation of simple 2D GIF animations. Moreover, the results of the stroke animation can be edited according to user preferences by template effects. It is assumed that the stroke similarity technique could also be applied to character recognition to help users find and review required freehand information. We intend to apply the proposed approach to character recognition.

In future work, we plan to increase the number of key-frame, e.g., multi-stroke morphing, and focus on color interpolation, e.g., color model or gradient color. Moreover, we recognize that sampling technique based on stroke shape will help users to create richer animations more efficiently. Therefore, we intend to study these functions. Such functions are applicable to a wide range of situations in anime production.

Acknowledgements

This project was funded in part by grants from the Japanese Information-Technology Promotion Agency (IPA) and by Research Fellowship for Young Scientists of Japan Society for the Promotion of Science (JSPS).
References


[4] Live2D


Visual Media Culture Supported by Human Depth Illusion

Kokichi Sugihara∗
Meiji University/JST, CREST

Abstract There is great difference between the real world and its images; the real world is three-dimensional whereas the images are two-dimensional. In spite of this difference we can enjoy visual media without any special efforts. Why can we do this? This question can be partly answered by the study of depth illusion. It seems that the human brains try to recover the depth from images using strong performances for some special subclass of objects such as rectangular solids. This observation also suggests fragility of the visual media culture. We discuss this point using optical illusion raised by impossible objects and impossible motions.

Keywords: Depth perception, impossible object, impossible motion, optical illusion, visual media

1 Introduction

We see objects in the real world by two eyes, by which we can recognize the distances to the objects. This is based on a mathematical fact that each point on an object can be located in the three-dimensional space as the point of intersection of two rays of sight from our eyes to the target point. This visual function is called the binocular stereo [1, 3, 4].

When we see images such as photographs and movies, on the other hand, the visual data include information obtained through only one eye, because a camera has only a single lens center. There is no explicit information about the distance to the objects. Nevertheless, we usually perceive depth to the objects in the images. This visual function is sometimes called the monocular stereo [1, 7].

Of course we can see the images by our two eyes, but this gives just the distance to the images, but not to the objects contained in the images.

Nowadays we are surrounded by various visual media, and we are apt to forget the differences between seeing objects directly and seeing them through images. However, there is a great difference between the binocular stereo and the monocular stereo. The former is based on the mathematical principle, while the latter is not.

The monocular stereo is based on just guessing the depths, and consequently very fragile. We will see this fragility through depth illusion raised by impossible objects and impossible motions.

∗email kokichis@meiji.ac.jp
2 Degrees of Freedom in Object Reconstruction

If we are given an object, the viewpoint $E$ and the image plane $I$, the image of the object projected on the image plane with respect to the center of projection at $E$ is uniquely determined. However, even if we are given $E$, $I$ and the image, the original object is not unique. There is freedom in the reconstruction of the object from the image. This freedom can be mathematically specified in the following way [6].

We assume that the object is a polyhedron, that is, an object bounded by planar faces. Suppose that the viewpoint $E$ is at the origin of the $(x, y, z)$ coordinate system, and that the image is fixed on the plane $z = 1$. Assume that the object in the image contains $n$ vertices and $m$ faces. Let $v'_i = (x_i, y_i, 1)$ be the $i$-th vertex on the image. The original counterpart $v_i$ of this vertex should be on the ray of sight emanating at the origin and passing through $v'_i$. Hence, the original vertex can be expressed by $v_i = (x_i/t_i, y_i/t_i, 1/t_i)$ where $t_i$ is an unknown variable.

Let $f_j$ be the plane containing the $j$-th face of the object, and let

$$a_j x + b_j y + c_j z + 1 = 0$$  \hfill (1)

be the equation of $f_j$, where $a_j$, $b_j$ and $c_j$ are unknown variables.

Suppose that the vertex $v_i$ should be on the face $f_j$. Then we can substitute the coordinates of $v_i$ into eq. (1), and get

$$a_j x_i + b_j y_i + c_j + t_i = 0.$$  \hfill (2)

This is linear in the unknowns $t_i, a_j, b_j$ and $c_j$.

For each pair of a vertex and a face containing it, we get the equation similar to eq. (2), and hence we get a system of linear equations, which we denote by

$$A w = 0,$$  \hfill (3)

where $w = (t_1, \ldots, t_n, a_1, b_1, c_1, \ldots, a_m, b_m, c_m)$ is a vector of unknown variables and $A$ is a constant matrix.

This system of equation contains $n + 3m$ unknown variables, and hence there are

$$n + 3m - \text{rank}(A)$$  \hfill (4)

degrees of freedom in the choice of the object from the image.

3 Depth Illusion

The degrees of freedom specified by eq. (4) are at least 4 for a thick object. This freedom enables us to design unfamiliar objects that look familiar, and thus raise various optical illusion [6]. The followings are examples of such illusions.

3.1 Impossible objects

There is a class of pictures called “pictures of impossible objects” [5, 13]. This class of pictures gives us an impression of three-dimensional structure but at the same time give us an impression of impossibility. An example is the picture of endless loop of stairs shown in Fig. 1 which is famous in that it is used by Dutch artist M. C. Escher in his artwork “Ascending and Descending” (1960) [2].
The impression of impossibility seems to come from our knowledge that stairs consist of horizontal and vertical plates. However, the freedom specified in eq. (4) suggest that there are other objects, and actually we can construct a three-dimensional solid whose projection coincides with the picture. Fig. 1(b) shows an example of a solid that looks the same as Fig. 1(a) and Fig. 1(c) is another view of the same solid.

Another example is given in Fig. 2, where (a) shows a “picture of an impossible object”, (b) shows a solid, and (c) shows an another view of the solid.

These examples show that pictures of impossible objects are not necessarily impossible; some of them are realizable as actual three-dimensional solids. This phenomenon shows that the human brains sometimes cannot extract solids represented in the images.

### 3.2 Impossible motions

The freedom in the choice of objects represented in images also enables us to design solids that look ordinary, but motions inserted to the solids appear to be physically impossible [10].

An example is shown in Fig. 3. The image (a) appears to be a solid consisting of four slopes slanted down to the four directions from the highest center. However, if we place balls on any slopes, they roll toward the center as if they are defying the gravity. Actually the center is the lowest as shown in (b), and the balls just roll downhill obeying the gravity law [8].
Another example is shown in Fig. 4. As shown in (a), the solid consists of a pole and four perches which look horizontal and crossing in right angles. However, a flat ring hangs in such a way that it passes behind the pole but in front of all the four perches. The actual shape of the solid is as shown in (b); the four perches all extend toward behind the pole [9].

One interesting observation is that the impossible motions are perceived even if we are told the true shapes of the objects. This fact implies that the impossible motion illusion cannot be removed even if we know the truth.

3.3 Ambiguous cylinders

The third class of depth illusion is ambiguous cylinders, whose appearances change drastically when they are reflected in a mirror [11, 12]. An example is shown in Fig. 5(a). The roof of the garage appears to be round when it is seen directly, while it appears to be corrugated in a mirror. Fig. 5(b) shows the same solid seen from another angle. From this image, we can understand that the true shape of the roof is neither of the appearances.

Another example is given in Fig. 6. As seen in (a), the object appears to be a cylinder whose section
is a flower shape, while its mirror image is a butterfly shape. Fig. 6(b) is another view of the same cylinder.

Both of the objects are made as the surfaces swept by a line segment which moves in a space without changing the orientation. Let us call this type of surface as a cylindrical surface; the solid in Fig. 6 is a closed cylinder while the garage roof in Fig. 5 is an open cylinder. The length of the cylinder measured along the axis direction (i.e., the direction of the sweeping line segment) is the same at every point on the surface. The human brains seem to interpret the edge of the cylinder as the intersection of the cylinder and a plane perpendicular to the axis of the cylinder. This causes the ambiguous cylinder illusion.

Ambiguous cylinder can be made because of the freedom in the reconstruction of objects. Actually this freedom enables us to design solids whose projections onto two prespecified directions give desired pair of two-dimensional shapes [11].
4 Fragility of Visual Media

Depth illusions we have observed suggest that it is difficult to interpret images as three-dimensional objects. However, it seems that visual media in our daily lives such as photographs, television, movies and other videos are used on the assumption that the humans can perceive correct depths from those media. Therefore, we have to be careful about this gap.

It seems that visual media are created mainly for generating reality, but we can intentionally create visual media that give incorrect impression of the real world.

One common example is photographs used in the advertisement of real estates. Photographs attached in house advertisements usually give us impression that the rooms appear to be larger than the actual sizes. We might say this is also an illusion. However, this illusion is not serious in that what is different from reality is just an impression of the space size.

On the other hand, the optical illusions presented in Section 3 are more extreme, because the perceived shapes of objects are quite different from the real objects. For example, slopes are perceived opposite in “Magnet-Like Slopes”, and the appearances change in a mirror completely for ambiguous cylinders. These depth illusions suggest that visual media such as photographs and movies contain serious risk in communicating about shapes of objects. In particular this implies the following two risks.

First, we are apt to think that there is not so much difference between seeing objects directly and seeing them through images. However, there is great difference; the former is the binocular stereo while the latter is the monocular stereo. Therefore, shape information might be distorted through visual media even if it is not intentional.

Second and more seriously, visual media can be used to distort shape data intentionally, as we have seem in various types of depth illusions in Section 3. We can intentionally make viewers to perceive the orientations of slopes opposite to actual slopes, as we have seen in impossible motion illusion. We can also make two viewers who see objects from different viewpoints to perceive the same object as quite different from each other.

Creatures got a pair of eyes in the middle of Cambrian period, which is about 500 million years ago. Since then the stereo vision functions in our brains evolved in sufficiently long time. On the other hand, the visual media technology such as photographs and movies began only several hundreds years ago. This implies that if we shrink the time so that we got a pair of eyes one year ago, we got visual media technology only 30 second ago. Therefore, we have not yet experienced the visual media technology to evolve the monocular stereo function.

The serious illusions in depth perception through images might be the result of insufficient time to evolve the monocular stereo. So we have to be careful about this aspect of visual media technology.

5 Concluding Remarks

We have shown depth illusion phenomena such as impossible objects, impossible motions and ambiguous cylinders. They all suggest that the monocular stereo function in our brains is fragile, and sometimes makes serious mistakes in perceiving depth to the object surfaces and consequently the shapes of the objects. This might be due to the rapid development of visual media technology, which is too short for our brains to evolve the visual data processing for the monocular stereo.
We have to recognize this shortcoming of the visual media technology, and to be careful in avoiding errors in depth perception.

Acknowledgements

This work is partly supported by the Grant-in-Aid for Basic Scientific Research No. 24360039 and for Challenging Exploratory Research No. 15K12067.

References

[9] Impossible Motion 2, http://www.youtube.com/watch?v=1dISzq4IPc
Finite-dimensional RKHS for Solving Computer Graphics Problems

Ken Anjyo
OLM Digital, Inc. / JST CREST

Abstract We present a few computationally tractable models of finite-dimensional Reproducing Kernel Hilbert Spaces (RKHS) that give a theoretical foundation of the techniques we have developed to solve several problems in computer graphics. The problems we deal with in this paper are signal and geometry interpolation/extrapolation as well as solving an inverse problem in animation.

Keywords: computer graphics, interpolation, extrapolation, superresolution, Reproducing Kernel Hilbert Space

1 Introduction

This section briefly describes a general regularization problem on a Reproducing Kernel Hilbert Space (RKHS). In later sections, this general approach will give an alternative formulation to solve several problems that we have encountered in computer graphics, such as [6, 12, 1, 3, 8].

The RKHS itself is a real-valued function space, and is denoted by $H(\Omega)$ or by $H$ for short, throughout this article. Typically $\Omega$ then means a domain in $\mathbb{R}^k$ or simply a finite set of numbers $\{1, 2, \ldots, N\}$. In the case of $\Omega$ being the finite set, an element of $H(\Omega)$ simply means a vector in $\mathbb{R}^N$. The RKHS $H(\Omega)$ is associated with its kernel function $K$. The kernel function $K$ is a real-valued function defined on $\Omega \times \Omega$, and is a symmetric, positive semi-definite function. An RKHS is prescribed completely with the kernel $K$, which in particular defines the norm of this special Hilbert space.

Now we consider the following regularization problem in [13]:

$$\min_{f \in H} \left\{ \sum_{i=1}^{n} (L_i(f) - y_i)^2 + \alpha \|f\|_K^2 \right\}$$

(1)

where, for $1 \leq i \leq n$, a real number $y_i \in \mathbb{R}$ and a continuous linear functional $L_i : H \rightarrow \mathbb{R}$ are given, and $\| \cdot \|_K$ denotes the norm of $H$ induced by its kernel $K$. It was proved in [5] that there exists a minimizer for the regularization problem in (1). In solving this problem for our practical purposes, it is key to choose a good kernel function. This means to employ an appropriate RKHS, because, for a given symmetric, positive semi-definite function $K$, we can construct an RKHS associated with $K$.

*anjiyo@acm.org
As a typical case, where $\Omega = \mathbb{R}^k$ and $L_i(f) = f(x_i)$ for given $x_i \in \Omega$ ($1 \leq i \leq n$), we can derive several types of splines from the above RKHS formulation. For $k = 2$, for example, the thin plate spline is connected with the kernel $K$ such that: $K(x, y) = \phi(||x - y||)$, where $\phi(r) = r^2 \log r$ is called thin plate spline. More generally a rich class of radial basis functions (RBF) can also be derived from the RKHS formulation, employing the Green’s function for various differential operators.

Our focus in this article is on a finite-dimensional real-valued RKHS. We first deal with the case where $\Omega = \{1, 2, \ldots, N\}$. Then the kernel function simply means a symmetric, positive semi-definite matrix $A$ and the RKHS is therefore the image of $A$, which is a linear subspace in $\mathbb{R}^N$: $A(\mathbb{R}^N) := \{Ax \mid x \in \mathbb{R}^N\}$. In our context, such as for direct manipulation blendshapes for facial animation [3], the matrix $A$ is the covariance matrix of given prior data. We will show that the original formulation in [3] is reduced to solving the above regularization problem in the RKHS.

Another RKHS discussed in this article is a finite-dimensional function space, rather than a numerical vector space. This will be used to explain a superresolution technique which is derived from RBF regression in [8].

2 Mathematical background

In this section we briefly review the definitions and fundamental properties of RKHS (for reference see [4, 11] or [1]).

Let $\Omega$ be an abstract set, and $H(\Omega)$ ($H$, for short) be a Hilbert space consisting of the real-valued functions defined on $\Omega$, with the inner product $\langle \cdot, \cdot \rangle$.

**Definition 1** If the function $K : \Omega \times \Omega \to \mathbb{R}$ satisfies the following conditions\(^1\), $K$ is called a reproducing kernel of $H$:

1. For any fixed $y \in \Omega$, $K(x, y)$ belongs to $H$ as a function of $x$.
2. For any $f \in H$, we have $f(y) = \langle f(x), K(x, y) \rangle_x$.

**Definition 2** If Hilbert space $H$ has the kernel $K$ in Definition 1, then $H$ is referred to as a reproducing kernel Hilbert space (RKHS).

**Proposition 1.** For the reproducing kernel $K$, we have:

$$K(y, z) = \langle K(x, y), K(x, z) \rangle_x,$$

for any $y, z \in \Omega$.

Considering that the inner product in $H(\Omega)$ is a symmetric, bilinear form, we get the fundamental property of the kernel $K$ from the above proposition:

**Proposition 2.** Let $K : \Omega \times \Omega \to \mathbb{R}$ be the kernel function of RKHS $H(\Omega)$. Then $K$ satisfies the following properties:

1. $K$ is symmetric: $K(x, y) = K(y, x)$ for any $x, y \in \Omega$.

\(^1\)In condition 2, the inner product $\langle \cdot, \cdot \rangle_x$ means that we get the inner product value of the two functions with variable $x$. 

24
In this section we first describe the numerical vector space $A(\mathbb{R}^N)$ in section 1 as an RKHS. We will thereafter see how the regularization problem for $A(\mathbb{R}^N)$ in section 1 is solved for learning the prior data of facial animations in [3], where $A$ is the covariance matrix. In this section we set $H_A := A(\mathbb{R}^N) = \{ Ax \mid x \in \mathbb{R}^N \}$.

### 3 RKHS and Bayesian estimates

In this section we first describe the numerical vector space $A(\mathbb{R}^N)$ in section 1 as an RKHS. We will thereafter see how the regularization problem for $A(\mathbb{R}^N)$ in section 1 is solved for learning the prior data of facial animations in [3], where $A$ is the covariance matrix. In this section we set $H_A := A(\mathbb{R}^N) = \{ Ax \mid x \in \mathbb{R}^N \}$.

#### 3.1 $H_A$ as RKHS

We consider the case where $\Omega = \{1, 2, \ldots, N\}$. Any mapping: $\Omega \times \Omega \to \mathbb{R}$ can then be represented as a matrix. Let $A : \Omega \times \Omega \to \mathbb{R}$ be a mapping which is symmetric, positive semi-definite in the sense of Proposition 2 in section 2. This simply means that $A$ is an $N$-th order positive semi-definite symmetric matrix. Considering Theorem in section 2, we can make $H_A$ a computationally tractable RKHS with the given matrix $A$ as its kernel in the following way.

We first note that $H_A$ is endowed with an inner product $\langle , \rangle$, which is given by

$$\langle f, g \rangle = (x, Ay), \quad (3)$$

where $f = Ax, g = Ay \in H_A$, and $( , )$ denotes the usual inner product in $\mathbb{R}^N$. Actually $H_A$ is a finite-dimensional linear subspace of $\mathbb{R}^N$. Let us then check the well-definedness of the above inner product. Suppose that $f = Ax_1 = Ax_2$ and $g = Ay_1 = Ay_2 \in H_A$. We then have:

$$\langle x_1, Ay_1 \rangle = \langle x_1, Ay_2 \rangle = (A^T x_1, y_2)$$

$$= (Ax_1, y_2) = (Ax_2, y_2) = \langle x_2, Ay_2 \rangle.$$

This means that the mapping $( , ) : H_A \times H_A \to \mathbb{R}$ is well-defined by (3). It is easy to see that this mapping is symmetric, and bilinear. Further, since $A$ is positive semi-definite, it follows that $\langle f, f \rangle \geq 0$, for any $f \in H_A$. We now denote $\sqrt{\langle f, f \rangle}$ by $\| f \|_A$. Next we show that $\| f \|_A = 0$, if and only if $f = 0$. Denoting the rank of $A$ by $r$, we first note that $H_A$ is a linear space spanned by the eigenvectors $u_i$ of $A$, whose eigen values $\lambda_i$ are positive:

$$f \in H_A \iff f = \sum_{i=1}^r c_i u_i, \quad (4)$$

where $\lambda_j > 0$ for $1 \leq j \leq r$, and $\lambda_k = 0$ for $r < k \leq N$. We can then rewrite $f$ by introducing $\lambda_i$,

$$f = \sum_{i=1}^r c_i u_i = \sum_{i=1}^r \frac{c_i}{\lambda_i} Au_i = A(\sum_{i=1}^r \frac{c_i}{\lambda_i} u_i).$$

---

2 In section 4 we also consider RKHS for a symmetric, positive definite function.
According to (3), we thus have

$$\|f\|^2_A = (\sum_{i=1}^r \frac{c_i}{\lambda_i} u_i, A(\sum_{j=1}^r \frac{c_j}{\lambda_j} u_j))$$

$$= (\sum_{i=1}^r \frac{c_i}{\lambda_i} u_i, \sum_{j=1}^r c_j u_j) = \sum_{i=1}^r \frac{c_i^2}{\lambda_i}. \tag{5}$$

We note that equation (5) can be understood as the finite dimensional version of the Mercer’s theorem. So any coefficient $c_i$ must be 0 ($1 \leq i \leq r$), if $\|f\|_A = 0$. We have thus confirmed that the mapping $\langle \cdot, \cdot \rangle$ is the inner product of $H_A$, and that $\| \cdot \|_A$ is its norm.

Since $H_A$ is finite-dimensional, it is easy to see that $H_A$ is a Hilbert space with the inner product (3). Finally we see that the given matrix $A$ is the kernel of $H_A$. Let $e_i (1 \leq i \leq N)$ be the canonical basis in $\mathbb{R}^N$, such that $(e_i, e_j) = \delta_{ij}$ for $1 \leq i, j \leq N$. The $i$-th column vector of $A$, denoted also by $Ae_i$, which means that $A$ satisfies condition 1 in Definition 1. Next suppose that $f = Ax \in H_A$, denoted as $f = (f_1, \ldots, f_N)^T$. According to the definition of the inner product (3), it follows that

$$f_i = \langle Ax, e_i \rangle = \langle f, Ae_i \rangle = \langle f, A(\cdot, i) \rangle$$

This holds for $1 \leq i \leq N$, which means that $A$ satisfies condition 2 in Definition 1.

### 3.2 Regularization for direct manipulation blendshapes

Our blendshape facial model will be described with:

$$f = Bw + f_0 \tag{6}$$

where $f$ is a $3n$-dimensional vector containing the components of each of the $n$ vertices or control points on the face vectorized in some arbitrary order such as xzxyzxyz... The matrix $B \in \mathbb{R}^{3n \times m}$ contains the $m$ delta blendshape targets $b_1, b_2, \ldots, b_m$ in its columns using the same component ordering (see [9] for more details). $w \in \mathbb{R}^m$ are the blendshape weights, and $f_0$ is the neutral shape.

We then denote a vector consisting of the position-constrained vertices by $\bar{f}$ for $f$ in (6). From now on, for an arbitrary $3n$ dimensional vector $v, v$ means a vector consisting of the components of $v$ that have the same indices as those of the position-constrained vertices of $f$. More precisely, considering (6), let $\bar{f}_0 = Bw_0 + f_0$ and $f_n = Bw_n + f_0$, where $f_0$ is the face model before the constraints and $\bar{f}_n$ is the constrained result. In concept, we would like to solve the following problem:

$$\min_{\Delta f} \|L(\Delta f) - \Delta \bar{m}\|^2 \tag{7}$$

where $\Delta f = f_n - f_0$, $\Delta \bar{m}$ is the differences of the vertex positions that are constrained by the artist, and $L$ is the projection that is defined as the diagonal matrix taking 1 as the components corresponding to these constrained vertices. It can be recognized that this is a severely underconstrained problem – for example, the artist may constrain only one or several vertices (with one vertex corresponding to three diagonal elements in $L$), whereas most diagonal elements in $L$ may take 0. As a solution to the direct manipulation problem, what we want to find is the weight $w_n$, whereas the weight $w_0$ is known in advance. To make it, we first solve the regularization problem regarding (7) in the RKHS framework, where we find the minimizer $f$, rather than the weights. Once we find the solution to this regularization problem, we can easily get the weights, as is described in [7, 3].
3.2.1 Learned direct manipulation

A strong and flexible prior would be desired to regularize problem (7). Let us assume that the prior training data are given, such as the face capture data. We then select the RKHS prior developed in section 3.1, by choosing the kernel \( K \) as the covariance matrix \( A \) of the prior data:

\[
A = E[(x - e_0)(x - e_0)^T],
\]

where each \( x \) is a face vector of the training data, and \( e_0 \) means the mean shape of the face data. Since a representation of the unknown function \( f \) in terms of the eigenvectors of \( A \) was made in section 1, we will also explore this representation for the data-fitting term in equation (1). We will see that this choice leads to a particularly simple solution to the direct manipulation problem.

3.2.2 RKHS for data + prior algorithm

To develop this data+prior approach to the direct manipulation problem, we start with a principal component model. The PCA model will then be denoted

\[
f = Uc + e_0
\]

where \( U \) contains eigenvectors of the data covariance matrix \( A \). The vector \( c \) in (8) will be referred to as the coefficient vector of the eigenvectors (or the coefficients, for short) throughout this report. In our context, we may assume that \( U = (u_1, u_2, \ldots, u_r) \) is a \( 3n \times r \) matrix containing only the eigenvectors \( u_i \) whose eigenvalue \( \lambda_i \) are all positive for \( 1 \leq i \leq r \) (thus \( r \) would be much smaller than \( 3n \) in practice). In the following discussion we put \( U := (u_1, u_2, \ldots, u_r) \).

Now let us denote the expansions of \( f_a \) and \( f_b \) in terms of the eigenvectors of \( A \) as

\[
f_a = Uc_a + e_0, \quad f_b = Uc_b + e_0.
\]

Unlike [3], we do not assume that the coefficients \( c_a \) and \( c_b \) are zero-mean Gaussian. Setting \( \Delta f = f_a - f_b = U(c_a - c_b) = U\Delta c \), we therefore formulate the problem for unknown \( \Delta f \) (or \( \Delta c \)), rather than for \( c_a \).

The direct manipulation problem regarding (8) can thus be interpreted through RKHS formulation. Consider \( H_A = A(R^{3n}) \), where we wish to solve the following regularization problem:

\[
\min_{\Delta c \in R^r} \| L(\Delta f) - \Delta \mathbf{m} \|^2 + \beta \| \Delta f \|^2_A,
\]

where \( L(\Delta f) = U\Delta c \) in our context and \( \| \cdot \|_A \) denotes the RKHS norm (5) for \( H_A \). This is therefore equivalent to

\[
\min_{\Delta \mathbf{c} \in R^r} \| U\Delta \mathbf{c} - \Delta \mathbf{m} \|^2 + \beta \| \Delta \mathbf{c} \|^2_A
\]

where \( \| \cdot \| \) is the usual Euclidian norm and the second norm \( \| \cdot \|_A \) is defined for \( y \in R^r \) as

\[
\| y \|^2_A = y^T \Lambda^{-1} y,
\]

having \( \Lambda^{-1} \) as the \( r \times r \) diagonal matrix whose diagonal element is \( \lambda_i^{-1} \) for \( 1 \leq i \leq r \). According to section 1, the problem (9) is theoretically solvable and we can make it numerically regarding the least square problem (10), as shown in [3].

4 RKHS for superresolution

In [8] we discussed RBF (radial basis function) regression of an exponential type. With the known \( n \) training data points \( p_k \), we denote the data to be interpolated by a vector \( f_0 = (f_1, f_2, \ldots, f_n)^T \).
We then assume that the RBF regression at a location \( p \) has the form \( f(p) = \sum_{k=1}^{n} w_k G(\|p - p_k\|) \), where \( G() \) is a radial function of exponential type. So we want to decide the weights \( w = (w_1, w_2, \ldots, w_n) \) such that \( f(p_k) = f_k \) for \( 1 \leq k \leq n \). The \( n \times n \) matrix \( G_0 \), whose \((i, j)\) component given by \( G_{ij} \equiv G(\|p_i - p_j\|) \), is positive definite, symmetric, and therefore invertible. In matrix-vector notation, the regression can thus take the following form:

\[
    f(p) = r^T w = r^T G_0^{-1} f_0, \tag{12}
\]

where \( r = (G(\|p - p_1\|), G(\|p - p_2\|), \ldots, G(\|p - p_n\|))^T \).

The superresolution technique in [8] is motivated by the above expression (12). Having a stationary stochastic process in mind, we now consider \( r \) as a vector of cross-covariances indexed by the difference between the location \( p \) and the locations of the data points \( p_i \), instead of the \( r \) defined above in (12), \( r = (C(\|p - p_1\|), C(\|p - p_2\|), \ldots, C(\|p - p_n\|))^T \). We also replace \( G_0 \) by a covariance matrix \( C_0 \), whose \((i, j)\) component is given by \( C(\|p_i - p_j\|) \). Assuming that the covariance function \( C(\Delta) \) is known for all offsets \( \Delta \) and that \( C_0 \) is invertible, we thus have the following superresolution scheme:

\[
    f(p) = r^T C_0^{-1} f_0. \tag{13}
\]

This is an extension of the RBF regression (12) in that we can treat a wider class of the "kernel" functions (or the covariance functions) with (13). See [8] for numerical illustrations about this extension.

In this section we will give an RKHS framework for this scheme.

### 4.1 Finite dimensional function space having a reproducing kernel

Let \( H_n(\Omega) \) (\( H_n \), for short) be a finite-dimensional real-valued function space, having \( \{h_i(p)\}_{1 \leq i \leq n} \) as its basis:

\[
    f \in H_n \iff f(p) = \sum_{i=1}^{n} w_i h_i(p). \tag{14}
\]

where the coefficients \( \{w_i\}_{1 \leq i \leq n} \) are uniquely determined according to the choice of the basis.

Suppose that an \( n \times n \) positive definite symmetric matrix \( S \) is given, while denoting its \((i, j)\) component by \( s_{ij} \): \( S \equiv [s_{ij}] \). Then, for any element \( f(p) = \sum_{i=1}^{n} w_i h_i(p) \) in \( H_n \), we define the norm \( \| \cdot \|_{H_n} \) with:

\[
    \|f\|_{H_n}^2 = \sum_{i,j} s_{ij} w_i w_j \equiv (w, Sw) \tag{15}
\]

where we put \( w = (w_1, w_2, \ldots, w_n)^T \). We then note that putting \( \langle h_i, h_j \rangle_{H_n} := s_{ij} \) induces the inner product in \( H_n \), which is denoted by \( \langle \ , \ \rangle_{H_n} \). We thus get the following (see [11]):

**Proposition 3.** Denoting \( S^{-1} \) by \( [t_{ij}] \), let us define \( K : \Omega \times \Omega \to \mathbb{R} \) as

\[
    K(p, q) := \sum_{i,j} t_{ij} h_i(p) h_j(q). \tag{16}
\]

Then \( K \) is the reproducing kernel of \( H_n(\Omega) \).
Actually it is easy to see the above $K$ in (16) satisfies conditions 1 and 2 in the definition of the reproducing kernel (Definition 1 in section 2). For instance, we have, for any $f \in H_n$,

$$
\langle f, K(\cdot, q) \rangle_{H_n} = \langle \sum_i w_i h_i, \sum_k \left( \sum_j t_{kj} h_j(q) h_k \right) \rangle_{H_n} = \sum_i \sum_j \sum_k w_i t_{kj} \langle h_i, h_k \rangle_{H_n} h_j(q) = \sum_{i,j,k} w_i t_{ik} s_{kj} h_j(q) = \sum_{i,j} w_i \delta_{ji} h_j(q) = \sum_i w_i h_i(q) = f(q).
$$

This means that the above $K$ satisfies condition 2 in Definition 1.

Finally it should be noted that we can select an arbitrary positive definite symmetric matrix $S$ for the definition of $\| \cdot \|_{H_n}$ in (15) and therefore the kernel $K$ in (16), independent of the choice of the basis $\{h_i(p)\}_{1 \leq i \leq n}$.

### 4.2 The finite dimensional RKHS for superresolution

We have shown that $H_n(\Omega)$ is a finite dimensional RKHS. A very nice feature of this formulation is that, once we can specify the basis functions $\{h_i(p)\}_{1 \leq i \leq n}$ for a practical situation, the RKHS is computationally tractable, so that we can numerically solve the regularization problem in section 1.

We now go back to the superresolution scheme (13), where we consider $h_i(p) := C(p - p_i)$ for $1 \leq i \leq n$, having $\Omega = \mathbb{R}^k$. We may assume that $\{h_i(p)\}_{1 \leq i \leq n}$ is a linearly independent system with the covariance matrix $C_0$ being invertible. This assumption is actually quite reasonable for our practical situations in computer graphics (see [2, 8], for instance). We can therefore deal with a variety of regularization problems on $H_n$. We may then have $C_0$ as a choice of $S$ in (15).

On one hand, when we consider only the regression problem in this section, it is easily solved in $H_n$. This is because the condition

$$
f(p_i) = \sum_{j=1}^{n} w_j h_j(p_i) = f_i , \text{ for } 1 \leq i \leq n
$$

simply means that $f_0 = C_0 w$, which leads to

$$
f(p) = \sum_{j=1}^{n} w_j h_j(p) = \sum_{j=1}^{n} w_j C(p - p_j) = r^T w = r^T C_0^{-1} f_0.
$$

This is equal to the superresolution form (13).
5 Concluding remark

The general idea of formulating inverse problems as a sum of data and prior, or data and smoothness terms, has been independently discovered in several different fields. There are both probabilistic and deterministic formulations [10]. This paper describes a finite-dimensional RKHS framework that can encompass both probabilistic (section 3.2.2) and deterministic (section 4.2) formulations for regularizing an inverse problem.

The generality of this framework is illustrated by defining finite-dimensional RKHS formulations for several example problems in computer graphics: signal and geometry interpolation/extrapolation, and solving an inverse problem in animation. We are currently exploring further applications of RKHS, such as those for rendering and texturing problems.

Acknowledgement

The author would like to thank J.P. Lewis, Hiroyuki Ochiai and Ayumi Kimura for their valuable comments and warm words of encouragement.

References


Sparsity and Information Processing

Shiro Ikeda*
The Institute of Statistical Mathematics, Tokyo, Japan

Abstract Recently, many information processing methods utilizing the sparsity of the information source is studied. We have reported some results on this line of research. Here we pick up two results from our own works. One is an image reconstruction method for radio interferometry and the other is a motor command computation method for a two-joint arm.

Keywords: Sparsity, interferometry, motor control

1 Introduction

Since the proposal of LASSO [1] and Compressed Sensing [2], sparsity based methods are getting popular. We have proposed some related methods in different fields which include astronomy [3, 4], diffraction imaging [5], and neuroscience [6, 7]. Here, we show two of them.

2 Interferometry

First result is the image reconstruction method for radio interferometry in astronomy [4]. In radio astronomy, the VLBI (Very Long Vaseline Interferometry) is an important technology since it has a very high angular resolution. In VLBI, a signal from an astronomical radio source is recorded with multiple radio telescopes located in a remote locations on the earth. By combining the measurements of telescopes, an observation vector $y \in \mathbb{R}^M$ is computed. This vector corresponds to the Fourier transform of the image vector $x \in \mathbb{R}^N$. The relation between $y$ and $x$ is denoted as follows

$$y = Ax + n.$$

Here, $n$ is the noise vector and $A$ is a truncated Fourier matrix. The image reconstruction problem is to compute $x$ from $y$ knowing the matrix $A$. The problem is not straight-forward since $M < N$ generally holds and $x$ cannot be computed by the inverse Fourier transformation. The image reconstruction is an ill-posed problem.

When the astronomical object is compact, we can expect $x$ to be sparse, that is, a lot of components of $x$ are 0. Thus, we can use LASSO to compute $x$ from $y$. More specifically, $x$ is computed by solving the following optimization problem.

$$x_\lambda = \arg \min_x \left[ \|y - Ax\|^2 + \lambda \|x\|_1 \right] \quad \text{subject to} \quad x_i \geq 0 \quad (i = 1, \ldots, N).$$

*email shirc@ism.ac.jp. This work is partially supported by JSPS KAKENHI Grant Number 25120008.
We have demonstrated the algorithm with simulated and real observed data and the proposed image reconstruction method will be used for the Event-Horizon Telescope [8].

3 Motor Control

The previous work was a method to solve an inverse problem. We have also proposed a method to generate the motor command for a two-joint arm [6, 7].

We have prepared an anatomical plausible two-joint arm model and have assumed the motor command representation in the brain is sparse. This framework enables us to reconstruct smooth movements of the arm for reaching tasks. It is also possible to store some motor commands and design new movements by combining stored motor commands (Fig. 1).

![Figure 1](image.jpg)

Figure 1: Left: Design of the motor control. The motor commands from a grid to another grid were stored and for a new movements, they are combined. Right: A smooth trajectory of the movement of the arm is generated by the proposed method.

References

Tetrisation of triangular meshes and its application in shape blending

Shizuo KAJI
Yamaguchi University / JST CREST

Abstract We introduce a method to associate a tetrahedral structure to a triangular mesh in 3D space, which can then be used in shape deformation applications. Our method allows one to keep track of global geometry such as curvature while working locally on tetrahedra. We apply this method to give a shape blending algorithm based on the as-rigid-as-possible deformation scheme. An implementation and its MIT licensed source code is available online.

Keywords: shape blending, tetrahedral mesh, as-rigid-as-possible deformation

1 Introduction

In mesh editing applications, it is sometimes required to have a tetrahedral structure rather than a triangular mesh. For example, in the seminal paper [1] they used tetrahedral volume meshes to produce interpolation of shapes. However, in most computer graphic systems it is common to represent shapes by surface meshes. To convert a surface mesh to a volume mesh is a non-trivial task (see, for example, [15]) and the resulting volume mesh tends to have many extra internal vertices, which make applications inefficient.

Instead of considering volume meshes, one can “fatten” surface meshes. A common practice is to associate a tetrahedral structure to a triangular surface mesh by adding the normal vector for every triangle (see, for example, [17]). This simple trick has been widely used, and also has a nice theoretical interpretation in discrete differential geometry. However, it does not capture important geometric features of the mesh. For example, the relation between adjacent triangles is neglected.

The purpose of this paper is to introduce two more constructions to associate a tetrahedral structure to a triangular mesh, which we call tetrisation. Our method encodes inter-triangular properties such as the angle between adjacent triangles.

As an application, we give a shape blending algorithm based on [1] (Fig. 1). Given an arbitrary number of isomorphic surfaces, our algorithm produces inter/extrapolation of the shapes according to the weights given by the user. Roughly speaking, we define a “linear combination” of shapes

\[ w_1 Q_1 + w_2 Q_2 + \cdots + w_m Q_m, \]

where \( w_i \in \mathbb{R} \) are weights and \( Q_i \) are shapes. Note that our algorithm is highly non-linear although we described the procedure as taking the linear combination of shapes. We implemented the algorithm as the Autodesk Maya plugin. Its MIT licensed source code is available at [8].
Figure 1: Three shapes (yellow) are blended to produce variations (white). Not only interpolation but also extrapolation (with weights $> 1$ or $< 0$) is possible. The top-left shape is obtained by extrapolating the two yellow shapes in the top row.

2 Notation

First, we list the notation used in this paper. We assume all the transformations are represented by real matrices acting on real column vectors by the multiplication from the left.

- $\text{SO}(3)$: the group of 3D rotations. Its element is a $3 \times 3$ special orthogonal matrix.
- $\text{Sym}^+(3)$: the set of 3D shears. Its element is a $3 \times 3$ positive definite symmetric matrix.
- $\text{GL}(3)$: the group of 3D linear transformations (compositions of rotation, shear, and reflection). Its element is a $3 \times 3$ regular matrix.
- $\text{Aff}(3)$: the group of the 3D affine transformations (compositions of rotation, shear, reflection, and translation). Its element is a $4 \times 4$ homogeneous matrix.
- $\text{GL}^+(3), \text{Aff}^+(3)$: the subgroups of the reflection free (positive determinant) elements in the corresponding groups.
- $\hat{A} \in \text{GL}(3)$: the linear part ($3 \times 3$ upper-left corner) of $A \in \text{Aff}(3)$.
- $A^t$: the transpose of a matrix $A$.
- $|A|_F^2 = \text{tr}(A^t A)$: the squared Frobenius norm of a matrix $A$.
- $\#U$: the cardinality of a set $U$.

3 As-rigid-as-possible deformation framework

First, we recall the framework of the as-rigid-as-possible deformation technique (ARAP, for short) since it is the main application and motivation of our work. The ARAP was first introduced in [1] for interpolating two shapes and has provided one of the fundamental frameworks for shape deformation (see, for example, [3, 7, 9, 16, 17, 18]). The idea is to measure the global error of a deformed shape as the sum of the local error over the tetrahedra. The local error is, in turn, defined in terms of the matrix which maps an tetrahedron of the original shape to the corresponding one of the deformed shape. We will discuss it precisely in the following.

Definition 1. A tetrahedral structure is a pair $(V, T)$, where the vertex set $V$ consists of three dimensional vectors and the set of tetrahedra $T = \{ T_i | 1 \leq i \leq n \}$ consists of ordered tuples of four distinct vertices $T_i = (v_{i1}, v_{i2}, v_{i3}, v_{i4})$. Each vertex must be contained in at least one tetrahedron.
A tetrahedral structure is said to be non-degenerate when the vertices of each tetrahedron are not co-planar.

We emphasise that overlap between tetrahedra is allowed, and for this reason, we use the terminology “tetrahedral structure” rather than tetrahedral mesh.

Let us assume that we are given an original shape with a tetrahedral structure $Q = (V, T)$. By a deformation $Q'$ of $Q$, we mean a tetrahedral structure with a vertex set $V'$ of the same cardinality as $V$ and the same set of tetrahedra $T$. That is, only the vertex positions are modified.

We can pack the information of $V$ and $T$ into a set of $4 \times 4$-matrices:

$$
P_i := \begin{pmatrix}
v_{i1}(x) & v_{i2}(x) & v_{i3}(x) & v_{i4}(x) \\
v_{i1}(y) & v_{i2}(y) & v_{i3}(y) & v_{i4}(y) \\
v_{i1}(z) & v_{i2}(z) & v_{i3}(z) & v_{i4}(z)
\end{pmatrix}, \quad 1 \leq i \leq n
$$

where $(v_{ij}(x), v_{ij}(y), v_{ij}(z))^t \in \mathbb{R}^3$ is the vector representing the position of the vertex $v_{ij} \in V$.

Assume that the original tetrahedral structure $Q$ is non-degenerate so that all the $P_i$ are regular.

Then, we can define a series of affine transformations

$$
A_i := P_i P_i^{-1} \quad (1 \leq i \leq n)
$$

which maps the vertices $V$ of the original shape $Q$ to the ones $V'$ in the deformed shape $Q'$.

Now, suppose the local transformations $C_i \in \text{GL}^+(3)$ $(1 \leq i \leq n)$ are prescribed. It depends on applications how $C_i$ are given. They are usually calculated from the user’s input (see §5 or [1, 9]).

We compute the deformed shape $Q'$ such that $V'$ minimises an error function with respect to $C_i$.

We consider two different error functions. In [1], they introduced

$$
E_T(V', \{C_i\}) = \sum_{i=1}^{n} |\hat{A}_i - C_i|^2,
$$

where $\hat{A}_i$ is thought of as a linear function of $V'$ by (2). Many of ARAP based shape deformation applications including [2, 17, 18, 19] use this function. Note that this is translation invariant but not rotation invariant. Rotation invariance is sometimes preferable in shape deformation (see, for example, [13, 16] and Fig. 8). We propose a rotation and translation invariant error function defined by

$$
E_S(V', \{C_i\}) = \sum_{i=1}^{n} |S(\hat{A}_i) - S(C_i)|^2,
$$

where $S(X)$ for $X \in \text{GL}(3)$ is the shear factor of the polar decomposition of $X$ (see [14]). Intuitively, this error function measures how much each tetrahedron is distorted. Despite the simplicity and its invariance property, $E_S$ has not been considered in the literature as far as the author is aware.

We believe this error function gives a good alternative to $E_T$ in some applications (see Fig. 8).

Remark 2. We can assign a weight $W_i \in \mathbb{R}$ to each tetrahedron $T_i$ to specify its contribution to the error function. It is done simply by replacing the summation $\sum_{i=1}^{n}$ with the weighted one $\sum_{i=1}^{n} W_i$ in the definitions of the error functions. For notational simplicity, we omit them in this paper.

The deformed shape is computed as the minimiser of the error function. Note that there is some indeterminacy of the minimiser coming from the invariance. For example, any translation of a minimiser is also a minimiser. To obtain a unique minimiser, we can impose additional constraints; for $E_T$ fixing the position of the barycentre and for $E_S$ fixing the position of the barycentre and the orientation of some tetrahedra.
Computing the minimiser for $E_T$ is reduced to solving a sparse linear system (see [1, 18]). For $E_S$, the computation is not linear. An iterative way similar to [16] is given as follows:

1. Compute the minimiser of $E_T(V', \{C_i\})$ and set $\hat{A}_i$.
2. Compute the polar decomposition $\hat{A}_i = R_i S_i$.
3. Compute the minimiser of $E_T(V', \{R_i S(C_i)\})$ to update $\{\hat{A}_i\}$.
4. Repeat (2) and (3) until $\{\hat{A}_i\}$ converge.

4 Tetrisation

In computer graphics systems, shapes are usually represented by surface meshes. Therefore, we cannot directly apply the ARAP technique, which requires a tetrahedral structure. Here, we consider a method to obtain a tetrahedral structure from a given triangular mesh.

For a triangular mesh, we denote an element of the vertex set $V$ by a three dimensional vector and an element of the set of (face) triangles $F$ by an ordered tuple of three vertices $(v_1, v_2, v_3)$. For $(v_1, v_2, v_3) \in F$, we call the ordered tuples $v_1 v_2, v_2 v_3, v_3 v_1$ the oriented edges. A triangular mesh is said to be non-degenerate when the vertices of each triangle are not co-linear.

We associate to a triangular mesh a tetrahedral structure with which one can work in the ARAP framework.

**Definition 3.** Given a non-degenerate triangular mesh $(V, F)$. A **tetrisation** of $(V, F)$ is a tetrahedral structure which consists of the vertex set $\bar{V}$ and the set of tetrahedra $T$. We require $T$ to satisfy the following conditions:

1. $V \subset \bar{V}$. That is, $\bar{V}$ is obtained by adding ghost vertices to $V$.
2. Each triangle in $F$ has to be contained in at least one tetrahedron in $T$.
3. Each tetrahedron is non-degenerate, that is, the four vertices are not co-planar.

These conditions are exactly what are required in the ARAP framework.

We give three methods to produce tetrisation in the following. First, recall that the unit normal vector $n(T)$ of a triangle $T = (v_1, v_2, v_3)$ is computed by $\frac{(v_2 - v_1) \times (v_3 - v_1)}{|(v_2 - v_1) \times (v_3 - v_1)|}$, where the denominator $|(v_2 - v_1) \times (v_3 - v_1)|$ is twice the area $2\text{Area}(T)$ of $T$.

4.1 Face-normal tetrisation

We begin with a simplest method which has been commonly used in various applications. For each triangle $T = (v_1, v_2, v_3)$ in $F$, add the ghost vertex $v_0 = \frac{(v_1 + v_2 + v_3)}{3} + \frac{(v_2 - v_1) \times (v_3 - v_1)}{\sqrt{|(v_2 - v_1) \times (v_3 - v_1)|}}$ and form a tetrahedron $(v_0, v_1, v_2, v_3)$. The resulting tetrahedral structure has $\#T = \#F$ and $\#\bar{V} = \#V + \#T$.

A problem with this tetrisation when applied to the ARAP framework is that this does not capture the relation between adjacent triangles. For example, consider two triangles sharing an edge as in Fig. 2. The rotation invariant error function $E_S$ with $C_1 = C_2 = Id$ will be minimised with any angle between the two triangles. In other words, folds do not cause any penalty in the error function.
4.2 Edge-normal tetrisation

We assume each oriented edge appears only once among all the triangles. In other words, an unoriented edge should be contained at most two triangles with the opposite orientations. Also, we assume all the triangles have at least one shared edge, that is, there is no “lone” triangle. (We can remove this assumption by adding ghost vertices not only for shared edges but for all edges. However, this is inefficient and makes no sense.)

For each shared edge $v_1v_2$, denote by $T_1 = (v_1, v_2, v_3)$ and $T_2 = (v_1, v_4, v_2)$ the two triangles adjacent to it. Add a ghost vertex

$$v_0 = \frac{v_1 + v_2}{2} + |v_1 - v_2| \frac{n(T_1) + n(T_2)}{|n(T_1) + n(T_2)|}$$

and form two tetrahedra $(v_0, v_1, v_2, v_3)$ and $(v_0, v_1, v_4, v_2)$. The resulting tetrahedral structure has $\#T = 2 \cdot \#(\text{shared edges})$ and $\#V = \#V + \#(\text{shared edges})$.  

4.3 Vertex-normal tetrisation

We assume that every vertex has a neighbourhood homeomorphic to the plane or the half plane. In other words, the mesh is a manifold (with boundary). Also, we assume all the triangles have at least one shared vertex. (Again, we can remove this assumption as in the previous subsection.)

For each shared vertex $v$, denote by $T_1, T_2, \ldots, T_k$ the adjacent triangles. Add a ghost vertex

$$v_0 = v + \frac{\sum_{i=1}^{k} \text{Area}(T_i) \frac{n(T_1) + \cdots + n(T_k)}{|n(T_1) + \cdots + n(T_k)|}}{2}$$
and form \( k \) tetrahedra by adding \( v_0 \) to the triangles \( T_i \) (\( 1 \leq i \leq k \)). The resulting tetrahedral structure has \( \#T = 3\#F - \#(\text{unshared vertices}) \) and \( \#V = \#V + \#(\text{shared triangles}) \).

An advantage of this method is that it extends straightforwardly to general polyhedral meshes.

![Figure 4](image1.png)

Figure 4: Left: the original surface, Right: its vertex-normal tetrisation. Ghost vertex is marked with red circle. We omitted ghost vertices on the boundary vertices for simplicity.

The idea of this tetrisation is to encode the angle around internal vertices, which is neglected by the face-normal tetrisation.

## 5 Application: Shape blending

Finally, we discuss an application of our method to shape blending. Roughly speaking, shape blending produces a “linear combination” of shapes

\[
 w_1 Q_1 + w_2 Q_2 + \cdots + w_m Q_m, 
\]

where \( w_i \in \mathbb{R} \) are weights and \( Q_i \) are shapes. In particular, when the number of shapes is two, \( w_1 Q_1 + (1 - w_1) Q_2 \) for \( 0 \leq w_1 \leq 1 \) gives a morphing between them. A basic method is to simply take the linear combination of the coordinates of the vertices. This algorithm is very fast and widely used to produce variations of shapes, in particular, facial expressions (see, for example, [12]). However, since the geometry of shapes is disregarded, it does not always produce plausible outputs (Fig. 5).

![Figure 5](image2.png)

Figure 5: Interpolation between yellow shapes. Left: linear blend shape, Right: our method

In [18] they give an algorithm based on [1] to blend/morph two or more shapes while preserving the geometry of shapes. A 2D version is also discussed in [2]. Here, we propose a similar algorithm based on [1]. The main differences between the method which we present here and theirs are

1. [18] considers only the face-normal tetrisation
2. [18] uses the error function \( E_T \), which is not rotation invariant
3. [18] uses the polar decomposition ([14] and [5]) to interpolate affine transformations, while we use the Cartan decomposition based parametrisation given in [10].
We will explain the difference and see its visual consequences later.

The problem setting is as follows. We are given a shape represented as a non-degenerate triangular mesh \((V_0, \mathcal{F})\) and \(m\) its deformations \(V_j (1 \leq j \leq m)\). The system computes the deformation \(V(w_1, \ldots, w_m)\) by blending the given shapes according to the user specified weights \(\{w_j \in \mathbb{R} \mid 1 \leq j \leq m\}\). We insist that it interpolates the given shapes, i.e., \(V(0, \ldots, 0) = V_0\), and \(V(w_1, \ldots, w_m) = V_k\) when \(w_j = \begin{cases} 1 & (j = k) \\ 0 & (j \neq k) \end{cases}\). Notice we allow negative weights so that the system can extrapolate not only interpolate.

We describe a general scheme based on the ARAP. The first step is to assign a tetrahedral structure \((\hat{V}, T)\) to the triangular mesh \((V_0, \mathcal{F})\) by tetrisation. Then, by (2) we obtain a set of affine transformations

\[ A_{ji} := P_{ji} P_{0i}^{-1} \in \text{Aff}^+(3) \ (1 \leq j \leq m, 1 \leq i \leq n), \]

where \(P_{ji}\) are as in (1) created from \(V_j\). Notice that by virtue of the non-degeneracy condition of tetrisation, each \(A_{ji}\) has the positive determinant.

The second step is to compute the “blended local transformations” \(C_i \in \text{GL}^+(3)\) by blending \(A_{ji}\) according to the weights \(w_j\). Intuitively, \(C_i\) stipulates the local transformation for the tetrahedron \(T_i\).

To define \(C_i\), we can use any interpolation function \(\text{Blend}\) for \(\text{GL}^+(3)\) which satisfies the obvious requirement for interpolation and set

\[ C_i := \text{Blend}(w_1, \ldots, w_m, \hat{A}_{1i}, \hat{A}_{2i}, \ldots, \hat{A}_{mi}). \]

We give two such interpolation functions later.

The third and final step is to determine the positions of the vertices \(V(w_1, \ldots, w_m)\) as the minimiser of any error function in \(\S 3\). Since \(C_i\) are generally not compatible on shared faces, we need to “patch” them by finding the global piecewise linear transformation which minimises the error function.

Now, we recall two interpolation functions for \(\text{GL}^+(3)\). By the polar decomposition (see, for example, [14]), we obtain

\[ \hat{A}_{ki} = R_{ki} S_{ki} \]

where \(R_{ki} \in \text{SO}(3)\) is the rotation and \(S_{ki} \in \text{Sym}^+(3)\) is the shear. [18] suggests

\[ \text{Blend}_P(w_1, \ldots, w_m, \hat{A}_{1i}, \hat{A}_{2i}, \ldots, \hat{A}_{mi}) = \sum_{k=1}^{m} \exp(w_k \log(R_{ki})) \left( \sum_{k=1}^{m} w_k S_{ki} + \left(1 - \sum_{k=1}^{m} w_k\right) I \right), \]

where \(\log\) is the principal matrix logarithm and \(I\) is the identity matrix\(^1\). On the other hand, we suggest to use [10]

\[ \text{Blend}_C(w_1, \ldots, w_m, \hat{A}_{1i}, \hat{A}_{2i}, \ldots, \hat{A}_{mi}) = \sum_{k=1}^{m} \exp(w_k \log^c(R_{ki})) \sum_{k=1}^{m} \exp(w_k \log(S_{ki})), \]

where \(\log^c\) is the “continuous” logarithm such that it chooses the nearest branch of logarithm to the adjacent tetrahedra when \(i\) varies (see [10] for details). The indeterminacy of \(\log\) for \(\text{SO}(3)\) is in the rotation angle and \(\log^c\) chooses the angle continuously for adjacent tetrahedra. Note that [10] provides a direct and fast formula for \(\text{Blend}_C\) which does not require the polar decomposition.

They look similar but there are two significant differences; logarithm for \(\text{SO}(3)\) and blending for the shear part. The value of \(\text{Blend}_P\) can fall out of \(\text{GL}^+(3)\) due to the linear blending of the shear part, which causes distortion in the output (Fig. 6). The use of the continuous logarithm enables

\(^1\) The term involving \(I\) is for normalisation and it enforces \(\text{Blend}_P(0, \ldots, 0, \hat{A}_{1i}, \hat{A}_{2i}, \ldots, \hat{A}_{mi}) = I\).
the system to produce a smoother morph among shapes which performs large rotation in between (Fig. 7). Note in [1] which discusses morphing of two shapes, they suggest to use the quaternions and SLERP to interpolate the rotation part and the linear interpolation for the shear part. (With three or more shapes, one can use the linear blending of the quaternions for the rotation part as in [11].) However, this method shows similar deficiency as Blend$_P$.

![Figure 6: Interpolation/extrapolation of yellow shapes. Left: with Blend$_P$ function in [18], the extrapolated shape on the left is degenerate. Right: with our Blend$_C$ function, the extrapolated shape is non-degenerate.](image)

![Figure 7: Interpolation of yellow shapes. Left: with Blend$_P$ function in [18], some parts try to rotate inconsistently. Right: with our Blend$_C$ function, local rotations are appropriately handled to produce a smooth interpolation](image)

Fig. 8 visually compares different tetrisations in §4 and the error functions $E_T$ and $E_S$ in §3. We observe that $E_S$ produces more natural results than $E_T$ but much slower as we see in Table 1. With $E_S$, the face-normal tetrisation causes extra wrinkles compared to the edge-normal and the vertex-normal tetrisations. As far as we experimented, it depends on the character of shapes to be blended which tetrisation gives the best result. In general, with $E_T$ the output is more or less similar regardless of the choice of tetrisation. With $E_S$, the vertex-normal tetrisation seems to be a good choice.

Table 1 shows a timing comparison for different tetrisations and error functions. We blended two 3D models each with 26k triangles on a Macbook Air with 1.7Ghz Intel Core i7 and 8GB memory. Initialisation part involves the Cholesky decomposition of the space matrix necessary to solve the minimiser of the error functions. This is computed only once in the initialisation process. Note that the matrix is dependent on the tetrahedral structure but independent of the choice of the error function. Runtime part consists of finding the minimiser of the error functions and the computation of Blend functions.

<table>
<thead>
<tr>
<th></th>
<th>face $E_T$</th>
<th>edge $E_T$</th>
<th>vertex $E_T$</th>
<th>face $E_S$</th>
<th>edge $E_S$</th>
<th>vertex $E_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialisation (sec)</td>
<td>0.1976</td>
<td>0.3080</td>
<td>0.3556</td>
<td>0.1976</td>
<td>0.3080</td>
<td>0.3556</td>
</tr>
<tr>
<td>Runtime with Blend$_P$</td>
<td>45.45 fps</td>
<td>27.43 fps</td>
<td>30.86 fps</td>
<td>11.66 fps</td>
<td>4.132 fps</td>
<td>4.762 fps</td>
</tr>
<tr>
<td>Runtime with Blend$_C$</td>
<td>48.46 fps</td>
<td>29.31 fps</td>
<td>31.46 fps</td>
<td>12.19 fps</td>
<td>4.576 fps</td>
<td>5.037 fps</td>
</tr>
</tbody>
</table>

Table 1: Timing comparison
Figure 8: Top row from left to right: rest shape $V_0$ and its two deformations $V_1$ and $V_2$ to be blended with weights $w_1 = 1.0$ and $w_2 = 1.5$. Second row from left to right: results obtained by face-normal, edge-normal, and vertex-normal tetrisation with $E_T$. Third row: same as the second row but with $E_S$.

We implemented our algorithm as the Autodesk Maya plugin ([8]). In our system, the user can specify the weight for each shape by slider, or the ball controller which computes the weight by [4] from the configuration of the balls representing the shapes (Fig. 9).

Figure 9: Our Maya plugin

**Acknowledgement** This work was partially supported by the Core Research for Evolutional Science and Technology (CREST) Program titled “Mathematics for Computer Graphics” of the Japan Science and Technology Agency (JST), by KAKENHI Grant-in-Aid for Young Scientists (B) 26800043, and by JSPS Postdoctoral Fellowships for Research Abroad.
References


A construction method for discrete constant negative Gaussian curvature surfaces

Shimpei Kobayashi
Department of Mathematics, Hokkaido University, Sapporo, 060-0810, Japan
shimpei@math.sci.hokudai.ac.jp

Abstract This is a survey based on the author’s paper [9] about a construction method for discrete constant negative Gaussian curvature surfaces, the nonlinear d’Alembert formula. The heart of this formula is the Birkhoff decomposition and we give a simple algorithm for the Birkhoff decomposition.

Keywords: Discrete differential geometry, pseudospherical surface, loop groups, integrable systems

1 Introduction

The study of smooth constant negative Gaussian curvature surfaces (PS surfaces1 in this survey) is a classical subject of differential geometry. It is known that the Gauss-Codazzi equations (nonlinear partial differential equations) for a PS surface become a famous integrable system, sine-Gordon equation:

$$\partial_y \partial_x u - \sin u = 0.$$ 

One of the prominent features of integrable systems is that they can be obtained by compatibility conditions for certain linear partial differential equations, the so-called Lax pairs. Moreover, the Lax pair contains an additional parameter, the spectral parameter, and it is a fundamental tool to study the integrable systems. On a PS surface, the spectral parameter induces a family of PS surfaces, which will be called the associated family, and the Lax pair is a family of moving frames (Darboux frames) and it will be called the extended frame of a PS surface. The extended frame can be thought as an element of the set of maps from the unit circle $S^1$ in the complex plane into a Lie group, the loop group, see Appendix A.

In [10, 13], it was shown that loop group decompositions (Birkhoff decompositions, see Theorem A.1) of the extended frame $F$ of a PS surface induced a pair of 1-forms $(\xi_+, \xi_-)$, that is, $F = F_+ F_- = G_- G_+$ with $\xi_+ = F_+^{-1} dF_+$ and $\xi_- = G_-^{-1} dG_-$. Then it was proved that $\xi_+$ and $\xi_-$ depended only on $x$ and $y$, respectively. Conversely it was shown that solving the pair of ordinary differential equations $dF_+ = F_+ \xi_+$ and $dG_+ = G_+ \xi_-$ and using the loop group decomposition,

1A constant negative Gaussian curvature surface is sometimes called a pseudospherical surface, thus we use “PS” for the shortened name.
the extended frame could be recovered. This construction is called the \textit{nonlinear d’Alembert formula} for PS surfaces.

On the one hand a discrete analogue of smooth PS surfaces was defined in [1] and the nonlinear d’Alembert formula for discrete PS surfaces was recently shown in [9]. In this survey we show basic results for smooth/discrete PS surfaces and the nonlinear d’Alembert formula for smooth/discrete PS surfaces according to [5, 1, 9]. The heart of the formula is the Birkhoff decomposition and we give a simple algorithm (Lemma 3.1) for the Birkhoff decomposition in case of discrete PS surfaces in Section 3.

\textbf{Acknowledgement:} The author would like to thank an anonymous referee for helpful comments. The author is partially supported by Kakenhi 26400059.

\section{Preliminaries}

We briefly recall basic notation and results about smooth and discrete PS surfaces in the Euclidean three space $\mathbb{E}^3$, that is $\mathbb{R}^3$ with the standard inner product $\langle \cdot, \cdot \rangle$, see for examples [11, 1, 13, 5, 10]. Moreover, we recall the nonlinear d’Alembert formula for discrete PS surfaces [9].

\subsection{Pseudospherical surfaces}

We first identify $\mathbb{E}^3$ with the Lie algebra of the special unitary group $SU_2$, which will be denoted by $su_2$:

\[ t(x, y, z) \in \mathbb{E}^3 \longleftrightarrow \frac{i}{2} x \sigma_1 - \frac{i}{2} y \sigma_2 + \frac{i}{2} z \sigma_3 \in su_2, \tag{2.1} \]

where $\sigma_j$ ($j = 1, 2, 3$) are the Pauli matrices as follows:

\[ \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{and} \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \]

Note that the inner product of $\mathbb{E}^3$ can be computed as $\langle x, y \rangle = -2 \text{trace}(XY)$, where $x, y \in \mathbb{E}^3$ and $X, Y \in su_2$ are the corresponding matrices in (2.1). Let $f$ be a PS surface in $\mathbb{E}^3$ with Gaussian curvature $K = -1$. It is known that there exist the Chebyshev coordinates $(x, y) \in \mathbb{R}^2$ for $f$, that is, they are asymptotic coordinates normalized by $|f_x| = |f_y| = 1$. Here the subscripts $x$ and $y$ denote the $x$- and $y$-derivatives $\partial_x$ and $\partial_y$, respectively. Then the first and second fundamental forms for $f$ can be computed as

\[ I = dx^2 + 2 \cos u \, dxdy + dy^2, \quad II = 2 \sin u \, dxdy, \]

where $0 < u < \pi/2$ is the angle between two asymptotic lines. Let

\[ e_1 = \frac{1}{2} \sec(u/2)(f_x + f_y), \quad e_2 = \frac{1}{2} \csc(u/2)(f_x - f_y) \quad \text{and} \quad e_3 = e_1 \times e_3 \]

be the Darboux frame rotating on the tangent plane clockwise angle $u$. Note that it is easy to see that $\{e_1, e_2, e_3\}$ is an orthonormal basis of $\mathbb{E}^3$. Under the identification (2.1), $\{-\frac{i}{2} \sigma_1, -\frac{i}{2} \sigma_2, -\frac{i}{2} \sigma_3\}$ is an orthonormal basis of $\mathbb{E}^3$, and for given $F \in SU_2$ and $x \in su_2$, $\text{Ad}(F)(x)(:= FxF^{-1})$ denotes the rotation of $x$. Thus there exists a $\bar{F}$ taking values in SU2 such that

\[ e_1 = -\frac{i}{2} \bar{F} \sigma_1 \bar{F}^{-1}, \quad e_2 = -\frac{i}{2} \bar{F} \sigma_2 \bar{F}^{-1} \quad \text{and} \quad e_3 = -\frac{i}{2} \bar{F} \sigma_3 \bar{F}^{-1}. \tag{2.2} \]
Without loss of generality, at some base point \((x_*, y_*) \in \mathbb{R}^2\), we have \(\hat{F}(x_*, y_*) = \text{Id}\). Then there exists a family of frames \(F\) parametrized by \(\lambda \in \mathbb{R}_+ := \{ r \in \mathbb{R} \mid r > 0 \}\) satisfying the following system of partial differential equations, see [5] in detail:

\[
F_x = FU \quad \text{and} \quad F_y = FV, \tag{2.3}
\]

where

\[
U = i \left( -\frac{u_x}{\lambda} \right) \quad \text{and} \quad V = -i \left( \frac{\lambda^{-1} e^{-iu}}{u_x} \right). \tag{2.4}
\]

The parameter \(\lambda \in \mathbb{R}_+\) will be called the spectral parameter. We choose \(F\) such that \(F|_{\lambda=1} = \hat{F}\) and \(F|_{(x_*, y_*)} = \text{Id}\).

The compatibility condition of the system in (2.3), that is \(U_y - V_x + [V, U] = 0\), becomes a version of the sine-Gordon equation:

\[
u_{xy} - \sin u = 0. \tag{2.5}
\]

It turns out that the sine-Gordon equation is the Gauss-Codazzi equations for PS surfaces. Thus from the fundamental theorem of surface theory there exists a family of PS surfaces parametrized by the spectral parameter \(\lambda \in \mathbb{R}_+\). Then the family of frames \(F\) will be called the extended frame for \(f\).

From the extended frame \(F\), a family of PS surfaces \(f^\lambda, \ (\lambda \in \mathbb{R}_+)\) are given by the so-called Sym formula, [12]:

\[
f^\lambda = \lambda \left. \frac{\partial F}{\partial \lambda} F^{-1} \right|_{\lambda \in \mathbb{R}_+}. \tag{2.6}
\]

The immersion \(f^\lambda|_{\lambda=1}\) is the original PS surface \(f\) up to rigid motion. The one-parameter family \(\{f^\lambda\}_{\lambda \in \mathbb{R}_+}\) will be called the associated family of \(f\).

### 2.2 Nonlinear d’Alembert formula

Firstly, we note that the extended frame \(F\) of a PS surface \(f\) is an element of the loop group for \(\text{SU}_2\), that is, it is a set of smooth maps from \(S^1\) into \(\text{SU}_2\), see Appendix A for the definition. In fact the extended frame is defined on \(\mathbb{C}^\times = \mathbb{C} \setminus \{0\}\) and it can be thought as a map from \(S^1\) into \(\text{SU}_2\). Then it becomes a Banach Lie group with suitable topology, which is an infinite dimensional Lie group and thus it will be called the loop group. Then the Birkhoff decomposition of the loop group is fundamental, which will be now explained. The loop group of \(\text{SU}_2\) will be denoted by \(\Lambda \text{SU}_2\) and we consider two subgroups \(\Lambda^+ \text{SU}_2\) and \(\Lambda^- \text{SU}_2\) of \(\Lambda \text{SU}_2\) as sets of maps which can be extended inside the unit disk and outside the unit disk, respectively. In other words, maps \(F \in \Lambda \text{SU}_2, \ F_+ \in \Lambda^+ \text{SU}_2\) and \(F_- \in \Lambda^- \text{SU}_2\) have the following Fourier expansions:

\[
F = \sum_{j=-\infty}^{\infty} F_j \lambda^j, \quad F_+ = \sum_{j=0}^{\infty} F_+^j \lambda^j \quad \text{and} \quad F_- = \sum_{j=-\infty}^{0} F_-^j \lambda^j.
\]

Then we consider the following problem: for a given map \(F \in \Lambda \text{SU}_2\), does there exist \(F_\pm\) or \(G_\pm\) taking values in \(\Lambda^\pm \text{SU}_2\) such that

\[
F = F_+ F_- \quad \text{or} \quad F = G_- G_+
\]

holds? The Birkhoff decomposition theorem assures that this decomposition always holds in case of the loop group of \(\text{SU}_2\), see Theorem A.1 in detail. Using the Birkhoff decomposition theorem, we give a construction method for PS surfaces, the so-called the nonlinear d’Alembert formula.
From now on, for simplicity, we assume that the base point is \((x_*, y_*) = (0, 0)\) and the extended frame \(F\) at the base point is identity:

\[
F(0, 0, \lambda) = \text{Id}.
\]

The nonlinear d’Alembert formula for smooth PS surfaces is summarized as follows, [10, 13, 5].

**Theorem 2.1** ([13, 5]). Let \(F\) be the extended frame for a PS surface \(f\) in \(\mathbb{E}^3\). Moreover, let \(F = F_+ F_-\) and \(F = G_- G_+\) be the Birkhoff decompositions given in Theorem A.1, respectively. Then \(F_+\) and \(G_-\) do not depend on \(y\) and \(x\), respectively, and the Maurer-Cartan forms of \(F_+\) and \(G_-\) are given as follows:

\[
\begin{aligned}
\xi_+ &= F_+^{-1} dF_+ = \frac{i}{2} \lambda \begin{pmatrix} 0 & e^{-i\alpha(x)} \\ e^{i\alpha(x)} & 0 \end{pmatrix} dx, \\
\xi_- &= G_-^{-1} dG_- = -\frac{i}{2} \lambda^{-1} \begin{pmatrix} 0 & e^{i\beta(y)} \\ e^{-i\beta(y)} & 0 \end{pmatrix} dy,
\end{aligned}
\]

where, using the angle function \(u(x, y)\), \(\alpha\) and \(\beta\) are given by

\[
\alpha(x) = u(x, 0) - u(0, 0) \quad \text{and} \quad \beta(y) = u(0, y).
\]

Conversely, let \(\xi_{\pm}\) be a pair of 1-forms defined in (2.7) with functions \(\alpha(x)\) and \(\beta(y)\) satisfying \(\alpha(0) = 0\). Moreover, let \(F_+\) and \(G_-\) be solutions of the pair of following ordinary differential equations:

\[
\begin{aligned}
dF_+ &= F_+ \xi_+, \\
dG_- &= G_- \xi_-,
\end{aligned}
\]

with \(F_+(x = 0, \lambda) = G_-(y = 0, \lambda) = \text{Id}\). Moreover let \(D = \text{diag}(e^{-\frac{i}{2} \alpha}, e^{\frac{i}{2} \alpha})\) and decompose \((F_+ D)^{-1} G_-\) by the Birkhoff decomposition in Theorem A.1:

\[
(F_+ D)^{-1} G_- = V_- V_+^{-1},
\]

where \(V_- \in \Lambda_+ \text{SU}_2\) and \(V_+ \in \Lambda^+ \text{SU}_2\). Then \(F = G_- V_+ = F_+ D V_-\) is the extended frame of some PS surface in \(\mathbb{E}^3\).

**Definition 1.** The pair of 1-forms \((\xi_+, \xi_-)\) in (2.7) will be called the pair of normalized potentials.

**Definition 2.** In [5], it was shown that the extended frames of PS surfaces can be also constructed from the following pair of 1-forms:

\[
\eta^x = \sum_{j = -\infty}^1 \eta^x_j \lambda^j dx \quad \text{and} \quad \eta^y = \sum_{j = -\infty}^\infty \eta^y_j \lambda^j dy,
\]

where \(\eta^x_j\) and \(\eta^y_j\) take values in \(\text{su}_2\), and each entry of \(\eta^x_j\) (resp. \(\eta^y_j\)) is smooth on \(x\) (resp. \(y\)), and \(\det \eta^x_j \neq 0, \det \eta^y_{-1} \neq 0\). Moreover \(\eta^x_j\) and \(\eta^y_j\) are diagonal (resp. off-diagonal) if \(j\) is even (resp. odd). This pair of 1-forms \((\eta^x, \eta^y)\) is a generalization of the normalized potentials \((\xi_+, \xi_-)\) in (2.7) and will be called the pair of generalized potentials, see also [4].

### 2.3 Discrete pseudospherical surfaces

Discrete PS surfaces were first defined in [1]. Instead of the smooth coordinates \((x, y) \in \mathbb{R}^2\), we use the quadrilateral lattice \((n, m) \in \mathbb{Z}^2\), that is, all functions depend on the lattice \((n, m) \in \mathbb{Z}^2\). The subscripts 1 and 2 (resp. \(1\) and \(2\)) denote the forward (resp. backward) lattice points with respect to \(n\) and \(m\): For a function \(f(n, m)\) of the lattice \((n, m) \in \mathbb{Z}^2\), we define \(f_1, f_2, f_1, f_2\) by

\[
f_1 = f(n + 1, m), \quad f_1 = f(n - 1, m), \quad f_2 = f(n, m + 1) \quad \text{and} \quad f_2 = f(n, m - 1).
\]

Then a discrete PS surface \(f\) was defined by the following two conditions:
1. For each point $f$, there is a plane $P$ such that 
\[ f, f_1, f_2, f_2 \in P. \]

2. The length of the opposite edge of an elementary quadrilateral are equal:
\[
|f_1 - f| = |f_1 - f_2| = a(n) \neq 0, \quad |f_2 - f| = |f_1 - f_1| = b(m) \neq 0.
\]

Then the discrete extended frame $F$ of a discrete PS surface can be defined by the following partial difference system, see [2, Section 3.2] and [1]:
\[
F_1 = FU \quad \text{and} \quad F_2 = FV,
\]
where
\[
U = \frac{1}{\Delta_+} \left( e^{-\frac{i}{2}(u_1-u)} e^{\frac{i}{2}p\lambda} \right), \quad V = \frac{1}{\Delta_-} \left( -\frac{i}{2}qe^{-\frac{i}{2}(u_2+u)} \lambda^{-1} - \frac{i}{2}qe^{\frac{i}{2}(u_2+u)} \lambda^{-1} \right),
\]
with $\Delta_+ = \sqrt{1 + (p/2)^2 \lambda^2}$ and $\Delta_- = \sqrt{1 + (q/2)^2 \lambda^2}$. Here $u$ is a real function depending on both $n$ and $m$, and $p \neq 0$ and $q \neq 0$ are real functions depending only on $n$ and $m$, respectively:
\[ u = u(n,m), \quad p = p(n) \quad \text{and} \quad q = q(m). \]

The compatibility condition of the system in (2.9), that is $VU_2 = UV_1$, gives the so-called discrete sine-Gordon equation:
\[
\sin \left( \frac{u_{12} - u_1 - u_2 + u}{4} \right) = \frac{pq}{4} \sin \left( \frac{u_{12} + u_1 + u_2 + u}{4} \right).
\]

(2.11)

The equation (2.11) was first found by Hirota in [7] and also called the Hirota equation.

Remark 2.2. Strictly speaking, length of the edges for a discrete PS surface should be small (less than 1). If the length is big (greater than or equal to 1), then the compatibility condition $VU_2 = UV_1$ gives a discrete analogue of mKdV equation, see [8] in detail. To consider the discrete nonlinear d’Alembert formula, this restriction is fundamental and thus we assume the conditions in (2.13).

Then a discrete PS surface $f$ can be given by the so-called Sym formula, [1]:
\[
f^\lambda = \lambda \frac{\partial F}{\partial \lambda} F^{-1} \bigg|_{\lambda \in \mathbb{R}_+}.
\]

(2.12)

The family of frames $F$ defined by (2.9) with $F|_{(n_*, m_*)} = \text{Id}$ will be called the discrete extended frame for a discrete PS surface. It is easy to see that the map $f^\lambda$ defined in (2.12) has these properties and $f^\lambda$ gives a family of discrete PS surfaces, see [2, Theorem 3].

### 2.4 Nonlinear d’Alembert formula for discrete PS surfaces

In this subsection we assume that the base point is $(n_*, m_*) = (0, 0)$ and the discrete extended frame $F$ at the base point is identity:
\[ F(0, 0, \lambda) = \text{Id}. \]

Moreover, we also assume that the functions $p$ and $q$ in (2.10) satisfy the inequalities
\[ 0 < \left| \frac{p}{2} \right| < 1 \quad \text{and} \quad 0 < \left| \frac{q}{2} \right| < 1. \]

(2.13)

Then the discrete nonlinear d’Alembert formula can be summarized as follows.
Theorem 2.3 ([9]). Let $f$ be a discrete PS surface and $F$ the corresponding discrete extended frame. Decompose $F$ according to the Birkhoff decomposition in Theorem A.1:

$$F = F_+ F_- = G_- G_+,$$

where $F_+ \in \Lambda^+ SU_2$, $F_- \in \Lambda^- SU_2$, $G_- \in \Lambda_+ SU_2$ and $G_+ \in \Lambda^+ SU_2$. Then $F_+$ and $G_-$ do not depend on $m \in \mathbb{Z}$ and $n \in \mathbb{Z}$, respectively, and the discrete Maurer-Cartan forms of $F_+$ and $G_-$ are given as follows:

$$\left\{ \begin{array}{l} \xi_+ = F_+^{-1} (F_+) = \frac{1}{\Delta_+} \left( \begin{array}{cc} \frac{1}{2} p e^{i \alpha} & i \frac{1}{2} q e^{-i \lambda} \\ - \frac{1}{2} q e^{i \lambda} & 1 \end{array} \right), \\
\xi_- = G_-^{-1} (G_-) = \frac{1}{\Delta_-} \left( \begin{array}{cc} 1 & i \frac{1}{2} q e^{i \beta} \\ - i \frac{1}{2} q e^{-i \beta} & 1 \end{array} \right), \end{array} \right. \tag{2.14}$$

where $\Delta_+ = \sqrt{1 + (p/2)^2 \lambda^2}$ and $\Delta_- = \sqrt{1 + (q/2)^2 \lambda^{-2}}$, the functions $p$ and $q$ are given in (2.10), and $\alpha$ and $\beta$ are functions of $n \in \mathbb{Z}$ and $m \in \mathbb{Z}$, respectively. Moreover let $u(n, m)$ in (2.10), $\alpha(n)$ and $\beta(m)$ are given by

$$\left\{ \begin{array}{l} \alpha(n) = \frac{1}{2} u(n + 1, 0) + \frac{1}{2} u(n, 0) - u(0, 0), \\
\beta(m) = \frac{1}{2} u(0, m + 1) + \frac{1}{2} u(0, m). \end{array} \right. \tag{2.15}$$

Conversely, Let $\xi_\pm$ be a pair of matrices defined in (2.14) with arbitrary functions $\alpha = \alpha(n)$, $\beta = \beta(m)$ with $\alpha(0) = 0$ and $p = p(n)$, $q = q(m)$ satisfying the conditions (2.13). Moreover let $F_+ = F_+(n, \lambda)$ and $G_- = G_-(m, \lambda)$ be the solutions of the ordinary difference equations

$$(F_+) = F_+ \xi_+ \text{ and } (G_-) = G_- \xi_-,$$ \tag{2.16}

with $F_+(n = 0, \lambda) = G_-(m = 0, \lambda) = \text{Id}$ and set a matrix $D = \text{diag}(e^{i \frac{j}{2} k}, e^{-i \frac{j}{2} k}) \in U_1$, where $k(0) = 0$ and $k(n) = 2 \sum_{j=0}^{n-1} (-1)^{j+n} \alpha(j)$ for $n \geq 1$. Decompose $(F_+ D)^{-1} G_-$ by the Birkhoff decomposition in Theorem A.1:

$$(F_+ D)^{-1} G_- = V_- V_-^{-1}, \tag{2.17}$$

where $V_- \in \Lambda^- SU_2$, $V_+ \in \Lambda^+ SU_2$. Then $F = G_- V_+ = F_+ D V_-$ is the discrete extended frame of some discrete PS surface in $\mathbb{E}^3$. Moreover the solution $u = u(n, m)$ of the discrete sine-Gordon for the discrete PS surface satisfies the relations in (2.15).

Definition 3. The pair of matrices $(\xi_-, \xi_+)$ given in (2.14) will be called the pair of discrete normalized potentials.

Similar to the smooth case, we generalize the pair of discrete normalized potentials:

**Definition 4.** Let $(\xi_-, \xi_+)$ be a pair of discrete normalized potentials and let $\eta_m$ and $\eta_n$ be

$$\eta_n = P_-^{\xi_+} P_-^{\xi_-}, \quad \eta_m = P_+^{\xi_-} P_+^{\xi_+}. \tag{2.18}$$

Here we assume that $P^\xi_{l \rightarrow r} \ (l = l \text{ or } r)$ take values in $\Lambda^\pm SU_2$ and do not depend on $m$ and $n$, respectively, that is, $P^\xi_\star = P_\star(n, \lambda)$ and $P^\xi_\star = P_\star(m, \lambda)$. Thus the $\eta_n$ and $\eta_m$ do not depend on $m$ and $n$, respectively:

$$\eta_n = \eta_n(n, \lambda), \quad \eta_m = \eta_m(m, \lambda).$$

The pair $(\eta_n, \eta_m)$ given in (2.18) will be called the pair of discrete generalized potentials.

**Remark 2.4.** The pair of normalized potentials $(\xi_+, \xi_-)$ and the corresponding pair of discrete generalized potentials $(\eta_n, \eta_m)$ in (2.18) give in general different discrete PS surfaces.
3 Algorithm for Birkhoff decomposition

In this section, we give a simple algorithm performing the Birkhoff decomposition used in Theorem 2.3.

When one looks at the discrete extended frame $F$ defined in (2.9), $F_{+}$ and $G_{-}$ defined in (2.16), one notices that they are given by products of two types of matrices:

$$e_{+} = \frac{1}{\sqrt{1+|a|^{2}}} \left( \begin{array}{cc} e^{i\theta} & a\lambda \\ -\overline{a}\lambda & e^{-i\theta} \end{array} \right) \quad \text{and} \quad e_{-} = \frac{1}{\sqrt{1+|b|^{2}}} \left( \begin{array}{cc} e^{i\kappa} & b\lambda^{-1} \\ -\overline{b}\lambda^{-1} & e^{-i\kappa} \end{array} \right),$$  \hfill (3.1)

where $\theta, \kappa \in \mathbb{R}, a, b \in \mathbb{C}$ and $|a|, |b| < 1$. It is easy to see that $e_{\pm}$ take values in $\Lambda^{\pm}\text{SU}_2$, respectively. Two matrices $e_{+}$ and $e_{-}$ do not commute in general, however, the following lemma holds.

**Lemma 3.1.** Let $e_{\pm}$ be matrices in (3.1). Then there exist matrices $\tilde{e}_{\pm}$ which takes values in $\Lambda^{\pm}\text{SU}_2$ such that

$$e_{+}e_{-} = \tilde{e}_{-}\tilde{e}_{+}$$

holds. In particular $\tilde{e}_{\pm}$ can be explicitly computed as follows:

$$\tilde{e}_{+} = \frac{1}{\sqrt{1+|a|^{2}}} \left( \begin{array}{cc} e^{i\theta} & a\lambda \\ -\overline{a}\lambda & e^{-i\theta} \end{array} \right) \quad \text{and} \quad \tilde{e}_{-} = \frac{1}{\sqrt{1+|b|^{2}}} \left( \begin{array}{cc} e^{i\kappa} & b\lambda^{-1} \\ -\overline{b}\lambda^{-1} & e^{-i\kappa} \end{array} \right),$$

where $\tilde{a}, \tilde{b}, \tilde{\theta}$ and $\tilde{\kappa}$ are explicitly chosen by the following equations:

$$\tilde{a} = ae^{-i(\kappa+\tilde{\kappa})}, \quad \tilde{b} = be^{i(\theta+\tilde{\theta})} \quad \text{and} \quad \tilde{\theta} + \tilde{\kappa} = \theta + \kappa + 2 \arg(1 - abe^{-i(\theta+\kappa)}).$$

Note that $\tilde{e}_{\pm}$ are not unique and one can always choose $\tilde{\theta} = 0$ or $\tilde{\kappa} = 0$.

**Proof.** It is just a consequence of a direct computation of $e_{+}e_{-}$ and $\tilde{e}_{-}\tilde{e}_{+}$, respectively. \hfill \Box

Using Lemma 3.1 iteratively, we obtain the following algorithm for the Birkhoff decomposition.

**Theorem 3.2.** Let $F$ be the discrete extended frame of a PS surface. Moreover let $F_{+}, G_{-}$ and $D$ be the matrices defined in (2.16). Then the Birkhoff decompositions for $F$ and $(F_{+}D)^{-1}G_{-}$ can be explicitly computed.

A Loop groups and the Birkhoff decomposition

In this appendix we give a definition of the loop group of $\text{SU}_2$ and its subgroups $\Lambda^{\pm}\text{SU}_2$. Moreover the Birkhoff decomposition will be stated.

It is easy to see that $F$ defined in (2.3) together with the condition $F|_{(x_{+}, y_{+})} = \text{Id}$ is an element in the twisted $\text{SU}_2$-loop group:

$$\Lambda \text{SU}_2 := \left\{ g : \mathbb{R} \times \mathbb{C} \rightarrow \text{SL}_2 \mathbb{C} \mid g \text{ is smooth, } g(\lambda) = \left( \begin{array}{c} g(-\lambda)^{-1} \end{array} \right)^T \quad \text{and} \quad \sigma g(\lambda) = g(-\lambda) \right\},$$

where $\mathbb{R}^{\times} = \mathbb{R} \setminus \{0\}$, $\sigma X = \text{Ad}(\sigma_{\mathbb{R}})X = \sigma_{\mathbb{R}}X\sigma_{\mathbb{R}}^{-1}$, $(X \in \text{SL}_2 \mathbb{C})$ is an involution on $\text{SL}_2 \mathbb{C}$. In order to make the above group a Banach Lie group, we restrict the occurring matrix coefficients to the Wiener algebra $\mathcal{A} = \{ f(\lambda) = \sum f_{n} \lambda^{n} : S^{1} \rightarrow \mathbb{C} \mid \sum_{n \in \mathbb{Z}} |f_{n}| < \infty \}$, where we denote
the Fourier expansion of \( f \) on \( S^1 \) by \( f(\lambda) = \sum_{n \in \mathbb{Z}} f_n \lambda^n \). Then the Wiener algebra is a Banach algebra relative to the norm \( \|f\| = \sum |f_n| \) and the loop group \( \Lambda SU_2 \) is a Banach Lie group, [6].

Let \( \mathbb{D}^+ \) and \( \mathbb{D}^- \) be the interior of the unit disk in the complex plane and the union of the exterior of the unit disk in the complex plane and infinity, respectively. We first define two subgroups of \( \Lambda SU_2 \):

\[
\begin{align*}
\Lambda^+ SU_2 &= \{ g \in \Lambda SU_2 \mid g \text{ can be analytically extend to } \mathbb{D}^+ \}, \\
\Lambda^- SU_2 &= \{ g \in \Lambda SU_2 \mid g \text{ can be analytically extend to } \mathbb{D}^- \}.
\end{align*}
\]

(\text{A.2})

Then \( \Lambda^+ SU_2 \) and \( \Lambda^- SU_2 \) denote subgroups of \( \Lambda^+ SU_2 \) and \( \Lambda^- SU_2 \) normalized at \( \lambda = 0 \) and \( \lambda = \infty \), respectively:

\[
\begin{align*}
\Lambda^+ SU_2 &= \{ g \in \Lambda^+ SU_2 \mid g(\lambda = 0) = \text{Id} \} \quad \text{and} \quad \Lambda^- SU_2 = \{ g \in \Lambda^- SU_2 \mid g(\lambda = \infty) = \text{Id} \}.
\end{align*}
\]

The following decomposition theorem is fundamental.

**Theorem A.1** (Birkhoff decomposition, [6, 3]). The multiplication maps

\[
\begin{align*}
\Lambda^+ SU_2 \times \Lambda^- SU_2 &\to \Lambda SU_2 \quad \text{and} \quad \Lambda^- SU_2 \times \Lambda^+ SU_2 \to \Lambda SU_2
\end{align*}
\]

are diffeomorphisms onto \( \Lambda SU_2 \), respectively.

---

**References**


Revisiting vorticity: pushing fluid solvers to the next level

Robert Bridson∗
Autodesk

Abstract Although some of the earliest work in fluid simulation for computer graphics exploited vorticity (e.g. Yaeger et al.’s work on Jupiter for the film 2010 [4]), by and large practical work over the last decade or two has focused on velocity-pressure formulations. This talk looks at why vortex methods are worth coming back to, the troubles that have steered practitioners away from them, and how we might overcome them.

Keywords: Fluid simulation, physics-based animation, vortex methods

1 Background

From vast oceans to air puffing dust around a footfall, fluids have become a staple of visual effects work, and direct simulation of the underlying physics has emerged as the most attractive approach to generate detailed and natural-looking fluid animation. Current research in graphics on this field can loosely be divided into two parts: work extending the range of phenomena that can be achieved (such as new materials and new means of artist control), and work improving the quality and/or efficiency of standard solves. We focus on the latter in this talk, and mostly on improving smoke simulation in particular.

Measuring the “quality” of a simulation is in of itself a tough, open problem. Accurately solving chaotic, high Reynolds number flow is essentially infeasible, so errors are a given: the question is which sorts of error are acceptable? Backed by experience, I will argue that accurately tracking vorticity and vortex structures is a good goal for graphics.

Vorticity \( \vec{\omega} \) is the curl of velocity \( \vec{u} \),

\[
\vec{\omega} = \nabla \times \vec{u}
\]

which measures locally how the flow is rotating (as opposed to shearing). In rigidly rotation regions, vorticity is exactly twice the angular velocity of the region. Velocity can be reconstructed from vorticity via the Biot-Savart law, in a sense integrating to undo the differentiation, as long as boundaries are also known: vorticity can serve as the fundamental state of a solver just as well as velocity.

Why vorticity might be so important is best illustrated mathematically with the equations of motion for a constant density fluid in two dimensions with negligible viscosity. Recall the material derivative

\[
\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla
\]

tracks how a quantity attached to the fluid changes in time while it flows

∗robert.bridson@autodesk.com
Figure 1: Left: Brochu et al. tracked vorticity just on a triangle mesh dividing space between smoky and clear air, producing highly detailed results [1]. Right: Goldade et al. demonstrated a real-time smoke simulator using a very small number of vortex particles (in purple) to nonetheless capture detailed and lively fluid motion [3].

with the fluid. The usual velocity-pressures equations are

\[
\begin{align*}
\frac{D\vec{u}}{Dt} + \frac{1}{\rho} \nabla p &= 0, \\
\nabla \cdot \vec{u} &= 0,
\end{align*}
\]

which show velocity is always being changed by the pressure gradient, while pressure is computed so as to keep the velocity incompressible. On the other hand, the vorticity equation which describes exactly the same motion is this simple:

\[
\frac{D\vec{\omega}}{Dt} = 0.
\]

In other words, vorticity just moves with the flow without otherwise changing, which is both striking mathematically and far easier to numerically solve.

Even better, in many flow scenarios vorticity is highly concentrated in small structured regions, and basically is zero throughout most of the domain. This *sparsity* in the representation can also be exploited numerically by tracking vorticity with a small and sparse set of particles or other Lagrangian elements. Figure 1 shows recent smoke results using relatively lightweight vortex triangle meshes and particles.

The fly in the ointment, which I believe has steered people away from vortex methods in the past, is the reconstruction of velocity from vorticity and boundaries. Without boundaries, finding the velocity at a single point in space with the Biot-Savart law requires integrating vorticity with a kernel over the entire fluid domain; with boundaries additional integrals or PDEs are required to calculate their effect. Even just formulating solid and free surface boundary conditions in terms of vorticity can be very tricky indeed. On top of all this, in three dimensions there is an additional term in the vorticity equation, for “vortex stretching,” which can be difficult to stably approximate.

Velocity reconstruction needn’t be so difficult, however. While a purely Lagrangian approach using the Biot-Savart law may require complex algorithms such as the Fast Multipole Method to scale well, great results can be obtained with the Vortex-in-Cell (VIC) method [2], where vorticity is splatted to a background grid and velocity is reconstructed there via solving the Poisson problem. Even greater detail can be achieved with the Particle-Particle Particle-Mesh (PPPM) approach, without too much more complication for graphics [5].
While there is still much more to say about boundary conditions for vortex methods, recent work by Zhang et al. shows that it’s possible to augment a traditional velocity-pressure solver with a correction to track vorticity like a vortex method, while handling the boundary conditions directly with velocity and pressure [6]. Figure 2 demonstrates the improvement in quality this correction gives with only a small overhead.

There is still much left to do, especially in transferring these techniques to water simulations; I hope this talk will provide a useful view forward on next steps in research.

References


A Simple Method for Morphing Smoke

Yoshinori Dobashi* Hokkaido University/JST CREST
Tsubasa Tani† Hokkaido University
Syuhei Sato‡ UEI Research
Tsuyoshi Yamamoto§ Hokkaido University

Abstract Recently, computer graphics has been used in many applications, such as computer games and movies. In the field of computer graphics, physically based simulation of fluids is one of the most important research topics and it allows us to synthesize highly realistic images of fluid phenomena. However, one of the problems with the fluid simulation is the high computational cost, making its applicability limited. This paper proposes a method for efficiently synthesizing smoke animations by applying the idea of image morphing to the simulated density distributions of smoke. We precompute a set of dynamic density distributions by numerical fluid simulation and then create inbetween density distributions by deforming and blending the precomputed density distributions.

Keywords: Smoke, Fluid Simulation, Morphing

1 Introduction

Recently, we often see synthetic images created by computer graphics techniques in many applications, such as computer games and movies. Synthesizing realistic animations of fluid phenomena, such as smoke and water, has been an active research area in computer graphics and many methods have been proposed [2]. Most of the recent methods are based on the numerical analysis of the governing equations of fluid motion called Navier-Stokes equations. Since the numerical analysis of the equations is generally time-consuming, many acceleration methods have been proposed. However, creating realistic and high-resolution animation of fluids is still expensive. Moreover, the user has to adjust many parameters involved in those methods repeatedly until the desired visual effects are obtained. The expensive computational cost makes this adjustment process extremely tedious.

In this paper, we present a method for efficiently synthesizing realistic animations of smoke by using the idea of image morphing. We develop a method for morphing dynamic density distributions of smoke that are obtained by numerically solving the Navier-Stokes equations. In a preprocess, we simulate the smoke with different parameter settings by using the grid-based method [5]. Our method then synthesizes inbetween dynamic density distributions from the precomputed distributions of smoke. In order to achieve the morphing of smoke, we need the following three components:

- a method to determine corresponding points between different density distributions of smoke,
• a deformation method for the smoke distribution, and
• a blending method between smoke densities.

For the corresponding points, we ask the user to interactively specify a representative path of the flow for each of the precomputed smoke density distributions. We call the representative path a flow curve. Next, the grid storing the smoke densities is deformed by using the flow curve. The densities are then linearly interpolated. We apply our method to synthesize dynamic smoke blown by wind and demonstrate the efficiency of the method.

The overview of the paper is as follows. In Section 2, we discuss some of the related work. Section 3 describes an overview of our system. Section 4 briefly explains the numerical fluid analysis that are used in our preprocess for creating dynamic density distributions of smoke. Our morphing method is then explained in Section 5. In Section 6, some experimental results are shown. Finally, Section 7 concludes the paper with discussion on limitations and future work of our method.

2 Related Work

Stam [17] addressed the stability problem in solving the Navier-Stokes equations and made the fluid simulation practical. After this work, many methods have been proposed for simulating various fluid phenomena. Readers can find details of those methods in [2]. One of the problems with the fluid simulation, however, is the expensive computational cost. Many methods have therefore been proposed to reduce the cost [9, 3, 1, 4, 8]. However, high-resolution simulation is still costly and tuning simulation parameters with repeated simulations is time-consuming. Some researchers use 2D fluid simulations for efficiently creating high-resolution animations of explosions [13] or fire [6][14]. These methods can reduce the computational cost significantly compared to high-resolution 3D fluid simulations. However, these methods are limited to the simulation of fluids that can be roughly approximated by a set of 2D simulations.

Recently, several methods have been developed to create high-resolution results from a low-resolution simulation. Nielsen et al. [11, 10] proposed methods for simulating high-resolution fluids that resemble a reference low-resolution simulation result. Yuan et al. [20] proposed a method that regulates high-resolution fluid simulation with flow patterns extracted from a low-resolution fluid simulations. Although these methods can generate high-resolution results with desired fluid behavior, costly high-resolution simulations are required to produce final animations. By combining low-resolution fluid simulation with turbulent noise functions, high-resolution results can be synthesized without conducting any high-resolution fluid simulations [7, 16, 12]. These methods can add small scale details but the results are not very realistic when compared to the results obtained by fluid simulations.

Treuille et al. [18] and Wicke et al. [19] proposed data-driven methods for accelerating the fluid simulations. However, one problem with these methods is that the methods require significantly long precomputation time ranging from twenty to thirty hours.

3 Overview of Our Method

The input to our method is multiple dynamic density distributions of smoke that are simulated by numerically solving the NS equations. For the simulation of smoke, we use the method developed by Fedkiw et al [5]. The density distribution is then represented by using a grid and is stored at discrete time step \( t(= 1, 2, \cdots) \). Our goal is then to synthesize inbetween density distributions by
interpolating the precomputed dynamic density distributions. The simplest way to achieve this is to simply blend the precomputed density distributions but it does not work very well. It is obvious that the so-called ghosting artifact appears.

In order to address this problem, we borrow the idea from the image morphing; the precomputed density distributions are deformed first and then blended. Fig. 1 shows an overview of our method.

We first ask the user to specify the center line of the smoke or flow curve for each of the precomputed dynamic density distributions. These curves allow us to make corresponding points between the precomputed density distributions. Next, an inbetween curve is generated by linearly interpolating the corresponding points of the user-specified flow curves. The grid storing the precomputed density distribution is deformed such that the flow curve coincides with the inbetween curve. For the deformation of the grid, we use the method developed by Schaefer et al [15]. The deformed density distributions are then blended together to create the final density distribution that are visualized by using a volume rendering technique.

**4 Smoke Simulation**

We assume the inviscid, incompressible flow for creating the input density distributions of smoke. The motion of the smoke is then calculated by solving the following Navier-Stokes equations.

\[
\frac{\partial \mathbf{u}}{\partial t} = - (\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f},
\]

\[
\nabla \cdot \mathbf{u} = 0,
\]

where \( \mathbf{u} \) is the velocity of the fluid, \( \rho \) is the density, \( p \) is the fluid pressure, and \( \mathbf{f} \) represents any external forces such as gravity and wind. We use the GPU to accelerate the computation to solve the above equations numerically. We also use the vorticity confinement method [5] in order to add turbulent motions.

For simulating smoke, we simply advect the smoke densities along with the velocity field obtained by solving the above equations. The smoke density \( D \) is therefore computed by using the following
equation.
\[
\frac{\partial D}{\partial t} = -(u \cdot \nabla)D + D_s,
\]
where \(D_s\) is the density of the smoke source.

5  Morphing Method

This section describes the method for morphing the precomputed density distributions. There are two types of morphing in our method: spatial and temporal morphing. The spatial morphing deforms the grids so that the shapes of the precomputed smoke roughly match with each other. The temporal morphing is used to make the speed of the smoke motions become the same.

5.1 Spatial Morphing

Let us explain our method by assuming that we have two input density distributions \(D_a\) and \(D_b\) as shown in Fig. 1. Our system first asks the user to specify the flow curves by drawing spline curves on the screen. The system then generate a set of sample points on the curve. The inbetween flow curve is then computed by interpolating the corresponding sample points linearly according to the user-specified blending factor \(\alpha\). When \(p_i\) and \(q_i\) are the sample points on the two user-specified flow curves, sample point \(r_i\) on the inbetween flow curve is represented by:

\[
r_i = (1 - \alpha)p_i + \alpha q_i
\]

We then deform the two grids storing the precomputed density distributions so that \(p_i\) and \(q_i\) coincide with \(r_i\) by using the method proposed by Schaefer et al [15]. As shown in Fig. 2, our system first computes the two-dimensional deformation by using \(p_i\), \(q_i\), and \(r_i\). Then, this deformation is applied to every cross section of the grid perpendicular to the view direction. The density distribution of the morphed smoke, \(D_m\), at time step \(t(= 1, 2, \ldots)\) is obtained by using the following equation.

\[
D_m(t) = (1 - \alpha)F_a(D_a(t)) + \alpha F_b(D_b(t)),
\]

where \(F_a\) and \(F_b\) represent the deformation operators applied to \(D_a\) and \(D_b\), respectively. That is, \(F_a(D_a)\) and \(F_b(D_b)\) represent the deformed density distributions.
5.2 Temporal Morphing

By using the method described in the previous subsection, the shapes of the deformed smoke distributions become similar. However, when the speed of the smoke motion is different from that of the other smoke, the ghosting artifact will appear. That is, we perceive two different overlapping smoke distributions moving at different speeds. We address this problem by artificially modifying the speed of the smoke motion in the following way.

During the precomputation process, we record an average speed of the motion at the center position of the smoke distribution. The center position is specified by the user. In creating the final animation using our morphing method, the speed of the smoke motion is adjusted so that the recorded average speed becomes the same. For example, when we have two precomputed density distributions and their recorded average speeds are $v_a$ and $v_b$, respectively, we use the following equations instead of Eq. 5 to obtain the final density distribution at time step $t$:

$$D_m(t) = (1 - \alpha)F_a(D_a(t_a)) + \alpha F_b(D_b(t_b)),$$

$$t_a = t_a - 1 + \frac{v}{v_a},$$

$$t_b = t_b - 1 + \frac{v}{v_b},$$

$$v = (1 - \alpha)v_a + \alpha v_b.$$

Note that $t_a$ and $t_b$ are no longer integer values and we compute $D_a(t_a)$ and $D_b(t_b)$ by linearly interpolating the temporally neighboring precomputed density distributions.

6 Results

This section shows results obtained by using the proposed method. We used a desktop PC with an Intel Core i7-3770 (CPU) and an NVIDIA GeForce GTX 660 (GPU) to create the following examples. The GPU was used for the precomputation only and the morphing process was computed on the CPU.

Fig. 3 shows an experimental result created to demonstrate the validity of our method. We precomputed two dynamic density distributions with and without applying wind forces. The number of grid points is $100 \times 50 \times 100$. The number of time steps of the simulation was 180 and it took 327 seconds for the precomputation. We then compare the smoke animation created by using our morphing method (Fig. 3(a)) with the animation created by solving the NS equations by gradually increasing the wind speed (Fig. 3(b)). Although the small-scale details are different between these animations, overall motions are the same as shown in Fig. 3. Figs. 3(a) and (b) took 40 and 174 seconds, respectively. Although our method requires a long precomputation process, the computational cost for creating the smoke animation is approximately four times faster than that by solving the NS equations. In addition, once the precomputation has finished, smoke animations with different wind speeds can be efficiently generated by using our method.

In Fig. 4, we apply our method to the simulation of smoke rising from a steam locomotive. The grid size for the smoke simulation for the precomputation is $200 \times 100 \times 200$. The precomputation and the morphing process took 80 and 12 minutes, respectively. Our method can successfully synthesize realistic and high resolution smoke animation efficiently.
(a) smoke simulated by using our morphing method

(b) smoke simulated by solving NS equations

Figure 3: Comparison of smoke animations created by using our method and by solving NS equations.

Figure 4: Examples of smoke rising from a steam locomotive.

7 Conclusions

We have proposed the morphing method for smoke simulations. In the method, multiple dynamic density distributions of smoke are precomputed by numerical analysis of NS equations with different parameter settings. The smoke animations are efficiently generated by deforming and blending the precomputed density distributions. We demonstrated the validity and the efficiency of the proposed method by several examples. Our method can successfully generate realistic animations of smoke.

There are several issues to be addressed in the future. In our current method, the flow curves are specified by the user. We would like to develop a method that can automatically determine the flow curves. The corresponding points between the precomputed smoke density distributions are determined by using the flow curves only. It would be better to determine the corresponding points so that the differences between the density distributions are minimized. Accelerating our morphing process using the GPU is also an interesting and important topic for real time applications such as computer games.
References


Wang Tiles Modeling of Wall Patterns

Alexandre Derouet-Jourdan∗
OM Digital, Inc. / JST CREST

Yoshihiro Mizoguchi†
Kyushu University / JST CREST

Marc Salvati
OLM Digital, Inc.

Abstract Wall patterns are essential in the creation of textures for visually rich buildings. Particularly, irregular wall patterns give an organic and lively feeling to the building. In this paper, we introduce a modeling method for wall patterns using Wang tiles which are known for creating aperiodic tiling of the plane under certain conditions. We introduce a class of Wang tiles and prove that any rectangle with border constraints and bigger than a $2 \times 2$ rectangle can be tiled. We use this proof to derive a tiling algorithm that is in linear time. Finally, we give some results of our algorithm and compare the computation time with previous Wang tiling algorithms introduced in computer graphics [2, 7].

Keywords: texture synthesis, wall patterns, Wang tiles, tiling algorithms

1 Introduction

Designing interesting and visually rich buildings requires the creation of complex wall or ground textures. Particularly, it requires the creation of wall patterns. To give a lively feeling to its creation, the artist may need to add a stochastic aspect to the distribution of bricks, so that the wall appears less regular and more organic. An example of such a texture is given Figure 1.

The process of creating such patterns is tedious but can be achieved without a high level of human control. Basically, the artist proceeds by selecting the borders constraints, then fill the inside of the wall by adding rectangles and correcting the distribution until satisfaction. It is a long process that cannot be used to create very large texture. In this situation, in order to texture large building, the artist creates an auto-tileable pattern that is repeated on the building, introducing a visual periodicity.

In this paper, we propose to replace this tedious process by an algorithm that designs wall patterns automatically. For this, we model the intersection of bricks in the wall as Wang tiles. The algorithm is fast and can create large wall patterns and patterns that satisfy border constraints.

2 Related works

Tiling a rectangle (or a plane) means filling it with elementary shapes, the tiles, such that the interior of tiles do not overlap and every point of the rectangle belongs to at least one tile. Tiling is a very

∗alexandre.derouet-jourdan@olm.co.jp
†ym@imi.kyushu-u.ac.jp
powerful tool to create visually rich 2d structures. For more details on tiling and tiling in computer graphics, please consult [5].

2.1 Wang tiles

In this article, we are interested in a special class of tiles, Wang tiles [9]. A Wang tile is a unit square with colored edges. Tiling with Wang tiles requires that the tiles are placed on a regular grid, edges to edges. Tiles from the tile set can be used as many times as required, but two tiles can share an edge only when they have the same color on the shared edge. We give three examples of Wang tiles Figure 2.

The Wang tiles have the interesting property of creating irregular patterns. Although Wang conjectured that if a finite Wang tile set were tiling the plane, then it was possible to tile the plane with the same tile set but periodically, this conjecture was disproved by Robert Berger in [1] when he introduced a Wang tile set that could only tile the plane aperiodically. Since then, several other aperiodic tile sets have been introduced [3, 6]. In computer graphics, Wang tiles have been used to create non periodic texture or point distribution automatically [2, 7].
2.2 Tiling algorithms

In the computer graphics community, in the context of texture synthesis, two tiling algorithms have been elaborated to tile a rectangle with a given set of Wang tiles.

**Sequential tiling algorithm** The easiest approach to tile a rectangle with a set of Wang tiles is to tile sequentially, row by row (or column by column) the rectangle, choosing at random a tile that matches the tiles already inserted in the rectangle. This algorithm was introduced in [2] for textures and point distribution generation. It has one important requirement on the tiles, that is every color-combination for left edge and top edge must exist in the tiles.

Advantages of this algorithm is to be easy to implement and fast, since it operates in linear time. One major drawback is that it can be hard to satisfy boundary constraints with this algorithm, as they introduce strong requirements on the tile set – that our tile sets do not satisfy.

**Stochastic tiling algorithm** Another tiling algorithm for Wang tiles, the stochastic tiling algorithm, has been introduced in [7] to account for border constraints. The idea of the algorithm is very simple. First, the rectangle is filled with random Wang tiles, without accounting for edge color matching. Then the wall is randomly “corrected” until all edge colors match. More precisely, the algorithm minimize the number of errors (non matching edge colors) by picking at random a tile with non matching edges, then replacing it with a better tile, that is a tile with at most the same number of errors. By iterating this process, the algorithm converges to a distribution of tiles with all edge colors matching.

One big advantage of this algorithm is that it produces a tiling satisfying the border constraints. Although in practice this algorithm works well, it is not proven that it will always converge. It is also difficult to estimate the number of iterations until convergence. In a production context, it is very important to give some guarantee of convergence and some estimation of computation time.

2.3 Our contributions

In this paper, we propose to model the bricks intersections with Wang tiles. To this intent, we introduce a specific class of Wang tiles. Accounting for border constraints is important to create repeatable patterns or to stitch together different patterns. We prove that any rectangle with border constraints and bigger than a $2 \times 2$ square can be tiled with our Wang tiles as soon as the number of colors in the Wang tiles is greater than 2. This proof is inductive and provides a tiling algorithm that is fast (linear time) and accounts for border constraints.

3 Wang tiles for walls

In our approach, the Wang tiles are containing the corners of the bricks in our wall. In other terms, the Wang tiles are all the way 4 bricks can connect in the wall. We suppose that the edges of the bricks are axis-aligned and we add the constraints that

- for aesthetics reasons, the 4 bricks do not share a vertex - they do not make a cross,
- each tile is traversed with a straight line, either vertically or horizontally.
Figure 3: In this article, we use Wang tiles to model the brick corners. The Wang tiles colors, that is the colors of the edges of the Wang tiles, represent the position of the intersection of the bricks edges with the edges of the tile.

The Wang colors are then the position where the bricks intersect the tile edge, see Figure 3 for an example. For instance, in the results presented in section 4, we consider a color set $C$ containing five Wang edge colors, corresponding to bricks intersecting the tile edge at lengths 0.25, 0.45, 0.5, 0.55, 0.75.

Such a tile set has a very interesting property. It is indeed easy to show that given a surrounding with two or three tiles, it is possible to find a tile in the tile set that has no erroneous edges shared with this surrounding. In other words, if we fix up to three edges color, we can always find at least one tile in the tile set that matches those colors. This property does not hold for 4 edge colors constraints because for instance, a tile with all its edges colors equal does not exist because it would introduce a cross in the tiles.

### 3.1 Definitions & notations

Let $C$ be a finite set of colors. A **Wang tile** is a function $w : \{t, l, b, r\} \rightarrow C$. Values $w(t)$, $w(l)$, $w(b)$, and $w(r)$ denote colors of top, left, bottom and right of a tile $w$ respectively, as illustrated Figure 4. A Wang tile $w$ is said in the class $\mathcal{W}$, the class of brick corners Wang tiles, if $(w(t) \neq w(b)) \land (w(l) = w(r))$ or $((w(t) = w(b)) \land (w(l) \neq w(r)))$. 

Figure 4: Notations on one Wang tile as defined section 3.1.
We define the set $W = \{w \mid w \in W\}$ of all Wang tiles in class $W$. Let $P_{nm}$ be a rectangle with size $n \times m$, that is $P_{nm} = \{(i, j) \in \mathbb{N} \times \mathbb{N} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$. The set of edges is $E = \{e_{i,j}, e'_{i,j'} \mid 1 \leq i \leq n, 0 \leq j \leq m, 0 \leq i' \leq n, 1 \leq j' \leq m\}$.

A boundary coloring is a function $B_{nm} : \{e_{i,j}, e_{i,m}, e_{n,j}, e_{n,m}\} \rightarrow C$ and a tile $P_{nm}$ is a function $T : P_{nm} \rightarrow W$ and we denote $T(i, j)$ by $T_{i,j}$. Given a tiling $T$ and a tile $w \in W$, we denote $T_{i,j} = w$ the tiling obtained from $T$ by replacing $T_{i,j}$ with $w$. We call $S(T_{i,j}) = \{e_{i,j-1}, e_{i,j}, e'_{i-1,j}, e'_{i,j}\}$ surrounding edges of a tiling $T$ at a position $(i, j)$, see Figure 5. A validation map $c : E \rightarrow \{true, false\}$ is defined by

\[
\begin{align*}
    c(e_{i,j}) &= \begin{cases} 
true & (T_{i,1}(t) = B(e_{0,j}) \land (j = 0)) \text{ or } \ 
    (T_{i,j}(b) = T_{i,j+1}(t) \land (1 \leq j \leq m - 1)) \text{ or } \ 
    (T_{i,m}(b) = B(e_{i,n}) \land (j = m)), \ 
    
false & \text{otherwise,}
\end{cases} \\
    c(e'_{i,j}) &= \begin{cases} 
true & (T_{i,j}(l) = B(e'_{0,j}) \land (i = 0)) \text{ or } \ 
    (T_{i,j}(r) = T_{i+1,j}(l) \land (1 \leq i \leq n - 1)) \text{ or } \ 
    (T_{n,j}(r) = B(e'_{n,j}) \land (i = n)), \ 
    
false & \text{otherwise.}
\end{cases}
\end{align*}
\]

A tiling $T$ is valid if $c(e_{i,j}) = true$ and $c(e'_{i,j}) = true$ for all for edges $e_{i,j}$ and $e'_{i,j}$ in $E$. We call an edge $e$ erroneous if $c(e) = false$. We denote $e(T_{i,j})$ the number of erroneous edges in $S(T_{i,j})$. The total number of erroneous edges for a tiling $T$ is denoted by $E_T$.

![Figure 5: Notations for a tiling $T$ and its edges.](image)

**Definition 1** (Tileable). Let $n$ and $m$ be natural numbers. A rectangle $P_{nm}$ is tileable if there exists a valid tiling $T$ for any boundary coloring $B_{nm}$.

**Definition 2.** Let $n$ and $m$ be natural numbers and $n, m \geq 2$. A boundary coloring $B_{nm}$ is a torus boundary coloring if $B_{nm}(e_{0,j}) = B_{nm}(e_{i,j})$ and $B_{nm}(e_{i,0}) = B_{nm}(e_{i,m})$ for any $1 \leq i \leq n$ and $1 \leq j \leq m$. A valid tiling $T$ for a torus boundary is called a periodic tiling.
3.2 Satisfiability of border constraints

The stochastic tiling algorithm accounts for border constraints but requires an important computation time. The sequential tiling algorithm is faster but does not account for border constraints. In this section, we prove that the border constraints can always be satisfied with our Wang tiles when the rectangle bigger than a $2 \times 2$ square. The proof induces a new algorithm, similar to the sequential tiling algorithm and that accounts for border constraints.

**Proposition 3.** Let $C$ be a color set.

1. $P_{12}$ is not tileable (i.e. there exist boundary colorings that can not be satisfied).
2. $P_{22}$ is not tileable if $|C| = 2$.

**Proof.** Let $c_1, c_2 \in C$ and $c_1 \neq c_2$.

1. We show an example of a boundary coloring $B_{12}$ for which there is no valid tiling $T$. If $B_{12}(e_{10}) = c_1$, $B_{12}(e_{02}) = B_{12}(e_{11}) = B_{12}(e_{12}) = B_{12}(e_{10}) = B_{12}(e_{12}) = c_2$ then there is no valid tiling. If $T$ is a valid tiling, then $T_{11}(l) = c_1$ and $T_{12}(t) = T_{11}(r) = T_{12}(l) = T_{12}(b) = T_{12}(r) = c_2$. Since $T_{11}(b) = c_2$ and $T_{12}(t) \neq c_2$, we have $T$ is not a valid tiling.

2. We define a boundary coloring $B_{22}$ by $B_{22}(e_{10}) = B_{22}(e_{01}) = B_{22}(e_{12}) = B_{22}(e_{22}) = B_{22}(e_{20}) = c_1$ and $B_{22}(e_{02}) = B_{22}(e_{21}) = B_{22}(e_{22}) = c_2$. It is easy to check there is no valid tiling $T$ if $|C| = 2$.

**Lemma 4.** Let $C$ be a color set, $B$ a boundary coloring of $P_{12}$.

1. Let $B(e_{10}) = B(e_{12})$. There exists a valid tiling $T$ if and only if

   $$(B(e'_{01}) = B(e'_{11}) \iff B(e'_{02}) = B(e'_{12})).$$

2. Let $B(e_{10}) \neq B(e_{12})$. There exists a valid tiling $T$ if and only if

   $$(B(e'_{01}) = B(e'_{11}) \lor B(e'_{02}) = B(e'_{12})).$$

**Proof.** 1. Assume $B(e'_{01}) = B(e'_{11})$. If Eq (1) holds then $B(e'_{02}) = B(e'_{12})$ and $P_{12}$ admits a valid tiling. That is we choose tiles $T_{11}$ and $T_{12}$ with $T_{11}(l) = T_{11}(r) = B(e'_{01})$ and $T_{12}(l) = T_{12}(r) = B(e'_{02})$. We note that $T_{11}(b) = T_{12}(t) \neq B(e_{10})$ and $T_{11}(b) = T_{12}(t) \neq B(e_{12})$. Conversely, if $P_{12}$ admits a valid tiling then $T_{11}(l) = T_{11}(r)$, $T_{11}(t) \neq T_{11}(b)$, $T_{11}(t) = B(e_{10})$, $T_{12}(b) = B(e_{12})$ and $T_{12}(t) \neq T_{12}(b)$. So we have $B(e'_{02}) = T_{12}(l) = T_{12}(r) = B(e'_{12})$. That is Eq (1) holds.

   Assume $B(e'_{01}) \neq B(e'_{11})$. The Eq (1) means $B(e'_{02}) \neq B(e'_{12})$. It is easy to show that if $B(e'_{02}) \neq B(e'_{12})$ then there exists a valid tiling. And if $B(e'_{02}) = B(e'_{12})$ then there is no valid tiling.

2. Assume that Eq (2) holds. It is easy to show there exists a valid tiling. Conversely, assume there exists a valid tiling. Since $B(e_{10}) \neq B(e_{12})$ and $T_{11}(b) = T_{12}(t)$, we have $T_{11}(t) \neq T_{11}(b)$ or $T_{12}(t) \neq T_{12}(b)$. That is $B(e'_{01}) = T_{11}(l) = T_{11}(r) = B(e'_{11})$ or $B(e'_{11}) = T_{12}(l) = T_{12}(r) = B(e'_{12})$. 

**Proposition 5.** Let $C$ be a color set and $|C| \geq 3$. A rectangle $P_{nm}$ with a border coloring $B_{nm}$ is tileable for all natural numbers $n, m$ $(n, m \geq 2)$. 

66
Proof. (i) $P_{22}$ is tileable. We divide $P_{22}$ into two $P_{12}$ and the tileability is proved by matching a given boundary coloring to conditions of Lemma 4.

(ii) If $P_{nm}$ is tileable then $P_{n(m+1)}$ is tileable. Let $B_{n(m+1)}$ be a given boundary coloring for $P_{n(m+1)}$. We define a tiling $T$ of $P_{n(m+1)}$ as follows. We can define $T_{i(m+1)}(l) = B_{n(m+1)}(e_{i(m+1)})$, $T_{i(m+1)}(l) = T_{i(m+1)}(r)$ and $T_{i(m+1)}(b) = B_{n(m+1)}(e_{i(m+1)})$. Since each tile $T_{i(m+1)}$ has only two conditions, we can choose a Wang tile $T_{i(m+1)}$ satisfying conditions easily. Next we define $T_{n(m+1)}$ satisfying the following three conditions $T_{n(m+1)}(l) = T_{(n-1)(m+1)}(r)$, $T_{n(m+1)}(b) = B_{n(m+1)}(e_{n(m+1)})$ and $T_{n(m+1)}(r) = B_{n(m+1)}(e_{n(m+1)})$. Finally, we reduce the tiling problem of $P_{n(m+1)}$ to the problem of $P_{nm}$. Define a boundary coloring $B_{nm}$ for the problem of $P_{nm}$ by $B_{nm}(e_{i0}) = B_{n(m+1)}(e_{i0})$, $B_{nm}(e_{im}) = T_{n(m+1)}(l)$, $B_{nm}(e_{0j}) = B_{n(m+1)}(e_{0j})$ and $B_{nm}(e'_{nj}) = B_{n(m+1)}(e'_{nj})$, $(1 \leq i \leq n, 1 \leq j \leq m)$. Since we are assuming $P_{nm}$ is tileable, we have a tiling $T$ for $P_{nm}$ with boundary coloring $B_{nm}$ and finally we have a tiling $T$ for $P_{n(m+1)}$ with boundary coloring $B_{n(m+1)}$.

(iii) If $P_{nm}$ is tileable then $P_{(n+1)m}$ is tileable. It is similarly proved with the result of (ii).

(iv) $P_{nm}$ is tileable. It is proved inductively using the results of (i), (ii) and (iii). \qed

### 3.3 Boundary-constrained sequential tiling algorithm

The proof of Proposition 5 induces naturally a tiling algorithm for a rectangle $P_{nm}$ with a constrained boundary coloring $B_{nm}$. This algorithm, called boundary-constrained sequential (BCS) tiling algorithm consists in picking at random a boundary coloring $B_{nm}$ and then recursively tile the rectangle $P_{nm}$ until we have $(n = 2 \land m = 2)$. In practice, the recursivity is replaced by a sequential loop that visit the columns and then the rows to reduce the tiling to $P_{22}$. Then, the square $P_{22}$ is tiled using Lemma 4. This tiling algorithm has a computation time linear with respect to $nm$, the total number of tiles in the considered rectangle, and accounts for boundary constraints. Particularly, the BCS tiling algorithm accounts for torus boundary condition, which is useful in practice to create auto-tileable patterns that allows to texture a mesh with a low memory footprint.

### 4 Results

In the following, we evaluate the performance of the new algorithm and compare it with the two algorithms from previous work. In this section, the Wang tiles are built with a color set $C$ containing five Wang edge colors, corresponding to bricks intersecting the tile edge at lengths 0.25, 0.45, 0.5, 0.55, 0.75.

### 4.1 Performance

Figure 6, we display the result of running the combination of sequence and stochastic relaxation algorithm to create a tiling of $49 \times 30$ tiles. Using Wang tiles allows to create structures without visible repeated patterns. We added colors to the bricks in a post-process, for aesthetics reasons.

Table 1, we report the computation time of the three algorithm for the creation of a small size tiling ($49 \times 30$) and a big size one ($1000 \times 1000$). As we can see, the sequential algorithm is fast but does not create auto-tileable patterns since it can not account for border constraints. The
Figure 6: Example of wall pattern created by the boundary-constrained sequential tiling algorithm. This tiling contains $49 \times 30 = 1470$ tiles and was computed in 1ms. 1609 tiles were visited during this computation. The bricks colors are added in a post-process.

The stochastic algorithm accounts for the border constraints but is slower, as it visits more tiles during the computation (in this example, it visits 10 times more tiles). Using the combination of both algorithm allows to account for border constraints, while benefiting from the speed of the sequential algorithm, with only a limited overhead.

4.2 Limitations

Modeling the wall pattern structure using Wang tiles allows the creation of rich non-repeated patterns. The tiling algorithm are fast and it is possible to account for border constraints.

The next step to make it available in a production environment is to create shaders from those patterns, by adding a texture to the bricks and a proper gap between them. To this end, it is important to be able to use the tiling structure to have a direct control over the shapes of the bricks and the gaps between them. Moreover, artists often need control over the creative process. For instance, they may need to control the distribution of bricks, having bigger bricks in some region of the wall, or a regular pattern in another one. In terms of Wang tiles, it means that we need to add the possibility of using different Wang tiles for different region of the tiling. In that case, it becomes necessary to be able to create some transitions tiles between the tile sets to guarantee the tileability of the wall.

If the creation of wall patterns is automatic, it is still needed to create Wang tiles manually. It then becomes important to check that the new tiles can tile the rectangle and that the algorithms can be used to produce the tiling. In addition, while convergence of the stochastic algorithm is empirically verified, we do not have any proof of this claim. It is an important development as we need the stochastic algorithm to account for border constraints.
### Table 1: Results of the three algorithms for a rectangle of size $49 \times 30$ (middle column) and $2000 \times 2000$ (right column, given as an average on 20 runs). If the sequential tiling algorithm is fast, it does not produce a tileable pattern. The stochastic tiling algorithm produces such pattern but is slower. The boundary-constrained sequential tiling algorithm (BCS), with only a small overhead on the sequential tiling algorithm, create quickly tileable patterns.

<table>
<thead>
<tr>
<th></th>
<th>$49 \times 30$</th>
<th>$1000 \times 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequential</strong></td>
<td>Visited tiles</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>1470</td>
<td>1ms</td>
</tr>
<tr>
<td><strong>Stochastic</strong></td>
<td>Visited tiles</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>20324</td>
<td>30ms</td>
</tr>
<tr>
<td><strong>BCS</strong></td>
<td>Visited tiles</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>1470</td>
<td>1.4ms</td>
</tr>
</tbody>
</table>

5 Conclusion

In this article, we introduced the modeling method for wall patterns using Wang tiles. Such a modeling approach allows us to use tiling algorithm to create efficiently wall patterns usable for texturing.

As future works, besides those we discussed above, we would be interested in creating 3d walls, by using 3d Wang tiles, Wang cubes [4]. We would then like to investigate how such a tool could be useful for volume rendering or 3d textures. In 2d, we proved that any rectangle with a boundary constraint is tileable if it is bigger than a $2 \times 2$ square. We believe that this property remains true for more complex polygon when they contain a $2 \times 2$ square. We would like to study this assumption and prove it, formally and using a theorem prover such as Coq [8].
References


High resolution visualization library for Exa-scale supercomputer

Yoshitaka Wada∗
Department of Mechanical Engineering, Kindai University
Kohei Murotani†
Department of Systems Innovations, University of Tokyo
Masao Ogino‡
High Performance Division, Information Technology Center, Nagoya University
Hiroshi Kawai§
Department of Computer and Media Engineering, Tokyo University of Science, Suwa
Ryuji Shioya¶
Faculty of Information Sciences and Arts, Toyo University

Abstract This paper describes development of visualization library, which is named as LexADV VSCG, for very large scale on the next generation supercomputer. At this moment, the next generation supercomputer, which is called as exa-scale computer, is not designed clearly. We predict the exa-scale computer and research the possibility of new software which is optimized for the exa-scale computer. In this work, we show how to visualize and deal with ultra large scale FE and particle based data with very fine resolution. LexADV VSCG just provide only simple API for drawing and rendering triangles and lines including transparent and solid colors.

Keywords: Exa-scale computing, parallel finite element method, parallel particle-based method, ultra high resolution

1 Introduction

The large scale FE analysis would be bottleneck using supercomputer systems[1-4]. However a commodity computer system becomes popular and faster than ever and graphic accelerator becomes 20 times faster than 10 years before at moderate estimation. The Kei supercomputer can generate numerical computation result of over tera bytes. Even if using data compression technique, network bandwidth is not enough for data transmission through the Internet. In the next generation supercomputer, the data transmission problem will be obvious and we will not be able to handle post processing, which is an examination task using visualization software system after simulation, on outside computers. There are two ways to solve the problem. One is the data compression which
is reduced data set composed by coarse volume data or coarse facet data using parallel computer
environment[5-8]. The data set becomes much smaller than the original data and we can handle the
post processing with interactivity. The other is the direct visualization on the supercomputer which
is sequential post processing after numerical simulation[9-11]. However computer visualization as
a post processing in the finite element analysis provides important user experience. From a point of
this view, visualization on the computer could not provide experience well in the past. In addition,
the architecture of the next generation supercomputer is just determined. In the background, the au-
thors have developed scientific visualization library: LexADV VSCG[9,12-14] with high portability
for any computer environment. In this study, several results are shown and a design and an actual
implementation for the library through the visualized results are discussed.

2 LexADV VSCG library

2.1 Fundamental ideas of design and implementation

Fundamental ideas of LexADV VSCG are as follows[3].

- Simple implementation for any environment: independent from OS
- Multiple images to be one image using z-buffer: parallel processing
- No other special hardware and libraries: independent from specific environment issue
- Handling very fine resolution by well-designed software: key idea for ultra large scale data
- Impressive image by well-designed algorithm and implementation: key idea using very fine
resolution

These ideas are important keys for high reliability on the future supercomputer. As mentioned above,
we don’t know the actual next generation supercomputer very well. In order to ensure those ideas,
we have decided rendering processes conducted by software without any other software library.
LexADV VSCG is a simple library software which provides application programming interface for
scientific visualization. In short, LexADV VSCG is usually integrated into a simulation code and
doesn’t use any specific hardware accelerators and software libraries which are developed by other
research group and companies.

Image data structure has additional several bits to represent depth information and dot product by
surface normal and view direction. The structure possess extended z-buffer information. The advan-
tages of the extended z-buffer are suited for parallel computers on which the images are rendered
in parallel. The images are usually stored at the local node storage or stay in local memory. For
example, if a supercomputer has totally 50,000 cores, FE mesh also is subdivided into 50,000 parts.
View camera is fixed at 200 locations, 10,000,000 images are rendered at least. We should merge
all of these images into 200 images in this case. As a result, the merging process needs huge data
transmission between nodes. Merging tasks are sequentially processed as shown in Fig. 1. If the
tasks are processed for 16 times, $2^{16}$ images can be merged into one.

Visualization technique of LOD (Level Of Detail) is a practical idea to deal with a very fine FE mesh.
Simplicity of handling the visualized results is important. LOD requires several level of data set for
one analysis model. In case of massively parallel data, we have to manage hundreds of thousands of
data sets. Furthermore, very large scale FE data can be generated on only a supercomputer. If com-
putation from modeling to visualization is conducted on a supercomputer, computing cost would be
larger than we expected at first. In order to overcome these difficulties, we have decided to generate
$10^5 \times 10^5$ pixels image in parallel and we just merge hundreds of thousands of images into one on
the supercomputer.
2.2 Implementation and functions

LexADV_VSCG is implemented in C language which is compatible with C99 standard. Any kind of compiler on supercomputers usually doesn’t provide the latest standard, because reliability and performance are required to compilers. Since a reliability is strongly demanded in scientific visualization, LexADV_VSCG needs no other library except for standard C library defined in C99 standard. Functions in the library are shown in table 1. The library provides very fundamental functions to draw and render triangles with gradated colors in accordance with specified physical quantity, i.e. von Mises stress $\sigma$ and principal stresses $\sigma_1$, $\sigma_2$ and $\sigma_3$ and so on. One of the target applications of the library is particle based method, i.e. Moving Particle Semi-implicit (MPS) and Smoothed-Particle Hydrodynamics (SPH). A sphere is represented by a polyhedron which consists of 80 or 320 triangles. An example of particle visualization is shown in Fig. 2. Total numbers of polygons affect total time to get an image, however it is important for users to choose fine or coarse visualization result. In the very fine image, differences between fine and coarse polyhedron do not make large difference of both of images.

3 How to get interactivity using the library

In the development of LexADV_VSCG library, the objective is clearly limited to applications to very large scale fine mesh and huge number of particles. The library doesn’t provide such usual techniques for high quality rendering image, which are ray casting and glow shading. As a result of these limitation, the implementation goes back to fundamental techniques by drawing triangle facet and shading on the flat triangle. However, huge number of fine facet and particle which represented by triangle facets, the quality of the image rendered by the library is enough for scientific visualization with the supercomputer.

Interactivity of visualization is one of the most important issue in the scientific post processing. Regretfully the latest supercomputer doesn’t provide any interactive connection with user’s computer. All of programs are controlled by job controlling system on the supercomputer. LexADV_VSCG library is able to generate very fine image with $10^5 \times 10^5$ pixels resolution. The image has enough information with regards to fine FE mesh. Usual image viewer can easily handle the large image by...
Table 1: Implemented fundamental functions for scientific visualization

<table>
<thead>
<tr>
<th>Functions</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>draw_triangle, line</td>
<td>draw triangle line with same color and no depth info</td>
</tr>
<tr>
<td>draw_triangle, solid</td>
<td>draw solid triangle with same color and no depth info</td>
</tr>
<tr>
<td>draw_triangle, line, with_depth (z-buffer)</td>
<td>draw triangle line with color and have depth info</td>
</tr>
<tr>
<td>draw_triangle, solid, with_depth (z-buffer)</td>
<td>draw solid triangle with color and have depth info</td>
</tr>
<tr>
<td>draw_triangle, graduated, with_depth</td>
<td>draw solid triangle with graduated color and have depth info</td>
</tr>
<tr>
<td>draw_triangle, graduated, transparent</td>
<td>draw transparent triangle with graduated color and have no depth info</td>
</tr>
<tr>
<td>allocate_image</td>
<td>allocate image buffer for drawing</td>
</tr>
<tr>
<td>set_color_XXXX (setting default color)</td>
<td>generate colors for basic colors and color legend</td>
</tr>
<tr>
<td>vector_XXXX (vector operations)</td>
<td>2 and 3 dimensional vector operations</td>
</tr>
<tr>
<td>accumulate_image (image operations)</td>
<td>merge two or more z-buffer images into one image</td>
</tr>
</tbody>
</table>

Figure 2: Particle representation by a polyhedron: coarse polyhedron in left side and fine polyhedron in right side.
shrinking to appropriate size as an engineer needs. We require interactivity of rotation of an object too. The system provides multiple viewpoints image generation as shown in Fig. 3. Simple spherical polar coordinate system and the description rule of the file name including the coordinate system are defined as described in Fig. 4. The rule is easily extended to other coordinate system, i.e. Euler’s angle definition. There are two ways to describe rotation of an object in the library. First way

\[ K = \begin{pmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{pmatrix} \]  

(1)

\[ R = I + (\sin \theta)K + (1 - \cos \theta)K^2 \]  

(2)

where \( \theta \) is length of Rodrigues vector. Camera position is notated in spherical polar coordinate. The distance from the center of an object to camera is constant, which is 1. Amplitude \( \theta \) is an angle between z-axis and radius vector. Amplitude \( \varphi \) is an angle between x-axis and projected vector of radius vector to x-y plane. In actual implementation, the camera is fixed at \((0, 0, -1)\). In order to

Figure 3: Multiple view points for off-line interactive visualization.

\{(name)\}_{seq. num.}_{\theta in deg.}_{\varphi in deg.}_{time step}.png

• Name: arbitrary numbers and alphabets within 256 letters
• Sequential number: distinguishing analysis case by 8 digits
• \( \theta \) and \( \varphi \) in degree: angle in degree, i.e. 01050 = 10.50°
• Time step: time for dynamic analysis by 6 digits

Example:
MechanicalPart_00000001_01050_00050_000000.png

Figure 4: Notation rule for camera position in spherical surface.

directly describes rotation matrix. Second way is Rodrigues vector \((k_1, k_2, k_3)\). Rodrigues’ rotation formula is described as follows.
describe the notation, camera positions should be calculated by

\[
\begin{align*}
\begin{pmatrix}
  x \\
y \\
z
\end{pmatrix} &= R^{-1} \begin{pmatrix}
  0 \\
  0 \\
  -1
\end{pmatrix}
\end{align*}
\]

where \( x, y \) and \( z \) are camera position in orthogonal coordinate system; \( \theta \) and \( \varphi \) are camera position in spherical polar coordinate system.

4 Results in numerical examples

4.1 Application to finite element analysis in structural problem

Visualization techniques are well designed and developed using GPGPU hardware and those can present impressive artificial images. However, usual design process and analysis process doesn’t demand for such an impressive image but accurate and easy to understand. From a practical point of view, the library supports polygon with flat shading including alpha blending which represent transparency effect.

Figure 5 shows an example of high resolution FE analysis image, which is resized image from \( 10^5 \times 10^5 \) pixels. In the image, black lines with 1 pixel width are drawn for representing finite elements, however no element lines cannot be seen in Fig. 5 (a). The \( 10^5 \times 10^5 \) pixels image shows element lines as shown in Fig. 5 (b). Figure 5 (b) is magnified image from the \( 10^5 \times 10^5 \) pixels image. Gradated color is very smoothly distributed in spite of linear interpolation of gradated color. Engineers want to see finite elements even if a very large scale mesh. Since manipulations of 2-dimensional image is processed very fast, magnification and reduction processes are done for a short time.

4.2 Application to particle based method in fluid dynamics

Figure 6 shows another example of very high resolution analysis by explicit MPS method. The visualization is conducted using transparent object and solid colored particles. Figure 6(a) shows 16,000 \( \times \) 12,800 pixels image which is enough resolution to examine fluid flow velocity and vorticity. Figure 6(b) is magnified image of velocity from the original image. A large number of particles forms the 3-dimensional configuration of fluid surface in Fig. 6(b), since the sufficient resolution of images keeps shadow of each particle and automatically makes the surface formed.

5 Conclusions

We have developed library software to visualize scientific computation results on the next generation supercomputer. LexADV.VSCG generates very fine images on the parallel environment efficiently. Interactivity is one of the important issue for examination of analysis result. We have also proposed interactive operation way by simple representation of spherical polar coordinate system. In particular structural analysis engineers usually examine the result using interactive visualization software.
Figure 5: Visualized results for finite element model with $10^5 \times 10^5$ pixel resolution: (a) whole view, (b) magnified view.
Figure 6: Visualized results for MPS analysis with $30,000 \times 20,000$ pixels resolution: (a) whole view of velocity and vorticity, (b) magnified view of velocity field at the same time step.
Basic visualization techniques are already implemented in the library. Huge amount of data, cannot be downloaded through network, is computed by supercomputers. Automated process to obtain the best visualization results is required for the next generation visualization software. The library just provides fundamental functions and reliability. The library is now developed in progress and is evaluated for more modifications to ensure reliability.

Acknowledgment

This research was supported in the Basic Research Programs: Development of System Software Technologies for post-Peta Scale High Performance Computing by CREST, JST.

References

Production Monte Carlo Rendering

Ryusuke Villemin*
Pixar Animation Studios

Abstract  Production rendering changed radically in last few years. With ever increasing computing power, it is now possible to use raytracing and physically based lighting and shading in production. The tools evolved (surfaces are now using normalized brdfs, lighting is using sampled arealights) and rendering is becoming more and more a sampling problem. Even in real-time rendering, we begin to see the use of such algorithms (like sampled ambient occlusion). Monte Carlo integration is the most common method of solving the rendering equation. Although the basic version does not have a remarkable convergence behaviour, numerous techniques have been added to the toolbox of the shader writer to make it practical.

Keywords: Monte Carlo, Sampling, Statistical, Lighting, Shading, Raytracing

1 Introduction

Rendering is rapidly becoming a sampling problem. In the first part of this paper, we will first present and expose the problem, which has a very compact and clean form in the rendering equation. Then we will show some of the most common techniques used in production: importance sampling and multiple importance sampling. Finally, we will go through a few practical cases, direct lighting, and bsdf and volume sampling.

2 Rendering Equation

The rendering equation was first introduced by Kajiya [7], in the following form:

\[ L_o(x, \omega) = L_e(x, \omega) + L_r(x, \omega) \]  \hspace{1cm} (1)
\[ L_o(x, \omega) = L_e(x, \omega) + \int_{\Omega} f_r(x, \omega', \omega) L_i(x, \omega')(\omega'.n) d\omega' \]  \hspace{1cm} (2)

which can be reformulated as an area integral over all surfaces in the scene:

\[ L_o(x', x) = L_e(x', x) + \int_{S} f_r(x'', x', x) L_i(x'', x) V(x'', x') G(x'', x') dA'' \]  \hspace{1cm} (3)

*rvillemin@pixar.com
with

\[ V(x',x) = \begin{cases} 1, \text{ if } x' \text{ and } x \text{ are mutually visible}, & \text{0 otherwise} \end{cases} \quad (4) \]

\[ G(x',x) = \frac{(\omega'.n')(\omega.n)}{||x' - x||^2} \quad (5) \]

Depending on the sampling strategy used, one or the other formulation will be preferred.

Although this equation already presents a few interesting challenges, this is in fact a simplified version that does not account for volumetric effects or complex subsurface scattering. Readers interested in the full form should look at Glassner [2].

### Table 1: Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_o )</td>
<td>outgoing radiance</td>
</tr>
<tr>
<td>( L_i )</td>
<td>incoming radiance</td>
</tr>
<tr>
<td>( L_e )</td>
<td>emitted radiance</td>
</tr>
<tr>
<td>( f_r )</td>
<td>brdf</td>
</tr>
<tr>
<td>( x )</td>
<td>position on surface</td>
</tr>
<tr>
<td>( n )</td>
<td>normal at ( x )</td>
</tr>
<tr>
<td>( \omega )</td>
<td>outgoing direction</td>
</tr>
<tr>
<td>( \omega' )</td>
<td>incoming direction</td>
</tr>
<tr>
<td>( \Omega )</td>
<td>hemisphere to directions</td>
</tr>
</tbody>
</table>

3 Monte Carlo Sampling

3.1 Introduction

A PDF (probability density function) \( p \) is a function that describes the distribution of values a random variable \( x \) can take on a domain \( X \). It has 2 main properties:

\[ \forall x \in X, p(x) \geq 0 \quad (6) \]

\[ \int_X p(x) = 1 \quad (7) \]

Now let’s say we want to evaluate the following integral:

\[ F = \int_X f(x) \, dx \quad (8) \]

Even if we don’t have an analytical solution, if we can evaluate \( f \) for any point in \( X \), then we can compute \( F \) with Monte Carlo sampling. Suppose we have a random variable \( Y \) with a PDF \( p \) defined on \( X \), and a function \( g \) such as the expected value of \( g(Y) \) is \( F \), then:

\[ E[g(Y)] = \int_X g(y) p(y) \, dy \quad (9) \]

\[ = \int_X f(y) \, dy \quad (10) \]

\[ = F \quad (11) \]
The integration problem is now a mean estimation problem, if we take \( n \) samples, then the estimate will be:

\[
\hat{F} = \frac{1}{n} \sum_{i=0}^{n-1} g(y_i)
\]  

where \( y_i \) are random values from the distribution \( p \).

The estimate is unbiased:

\[
E[\hat{F}] = \frac{1}{n} \sum_{i=0}^{n-1} E[g(Y_i)]
\]  

\[
E[\hat{F}] = E[g(Y)]
\]  

\[
= \int_X f(x)dx
\]

And the variance is:

\[
E[\hat{F}] = \frac{1}{n} \sum_{i=0}^{n-1} V[g(Y_i)]
\]  

\[
E[\hat{F}] = V[g(Y)]
\]  

\[
= \int_X \left( g(x) - \int_X f(t)dt \right)^2 dx
\]

### 3.2 Importance Sampling

In the previous section, we showed a way to compute the integral:

\[
F = \int_X g(x)p(x)dx
\]  

Unfortunately this is not exactly the equation we are after, but a simple rewrite can get us back to the original problem:

\[
F = \int_X f(x)dx
\]  

\[
= \int_X \left( \frac{f(x)}{p(x)} \right) p(x)dy
\]

then the estimate is:

\[
\hat{F} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{f(y_i)}{p(y_i)}
\]
To have a good estimate, we can increase the number of samples $n$, but also try to reduce the variance of each $f/p$ ($p$ should match $f$). We still need a way to generate those $y_i$ samples based on $p$, a few techniques are available to us, the most interesting one and usable in practice, is the CDF inversion.

### 3.3 CDF Inversion

One question is: how do we distribute samples according to $p(y)$? For the random variable $Y$, the CDF (cumulative distribution function) is defined as:

$$P_Y(y) = P_{probability}(Y \leq y) = \int_{y_{min}}^{y} p(x)dx \quad (23)$$

If $Y$ is a scalar random variable with CDF $P_Y$, then the random variable $U$ defined by:

$$U = P_Y(Y) \quad (24)$$

is the uniform distribution. If we invert this equation:

$$Y = P_Y^{-1}(U) \quad (25)$$

then $Y$ is distributed with density $p$. This transformation is called the inverse CDF technique. This means that if you can compute the inverse of the integral of a PDF, then you can use the inverse CDF method. Other methods like rejection sampling are also available, but not very popular in practice, since there is no way to bound the computational cost, which is always a variable we try to minimize in rendering.

### 3.4 Multiple Importance Sampling

To use the CDF inversion method efficiently we need to find a PDF that will closely match the function $f$ we are trying to integrate. Unfortunately in real scenarios, it is usually not possible to find such functions, but instead we can find multiple PDFs, each of them roughly matching some part of $f$. We could let the user choose the best technique for each scenario but it’s not really practical. Therefore we use multiple importance sampling (MIS) to automatically combine PDFs. If we have $M$ techniques sampling $N$ samples, then:

$$\hat{F} = \frac{1}{N} \sum_{j=0}^{M-1} \sum_{i=0}^{N_j} \omega_j(y_i) \frac{f(y_i)}{p_j(y_i)} \quad (26)$$

the estimate will remain unbiased if the weights obey the following:

$$\sum_{j=0}^{M-1} \omega_j(y) = 1 \quad (27)$$
we could just take $\omega_j = 1/M$ but this would just give an average of all the techniques, Veach [11] instead proposed a very simple, yet efficient method called the balance heuristic:

$$\omega_i(y) = \frac{p_i(y)}{\sum_{j=0}^{M-1} p_j(y)}$$  \hspace{1cm} (28)

4 Applications

4.1 Direct Lighting

For direct lighting, we have the following equation which is a further simplified version of the general rendering equation:

$$L_o(x, \omega) = \int_{\Omega} f_r(x, \omega', \omega)L_{light}(x, \omega')(\omega'.n)d\omega'$$ \hspace{1cm} (29)

and in the three-point form:

$$L_o(x', x) = \int_{S} f_r(x'', x', x)L_{light}(x'', x)V(x'', x')G(x'', x')dA''$$ \hspace{1cm} (30)

$L_i$ in the integral is replaced by $L_{light}$, the energy emitted directly by light sources. The implicit recursion is gone and what’s left is just a 2D integral of a product of 3 functions $L_{light}$, $f_r$, and $V$. The visibility $V$ is scene dependent and usually there is no good way to compute it except to sample blindly; however we can use our knowledge of the material $f_r$ and the light source $L_{light}$ to guide our sampling. And by using multiple sampling, we don’t have to choose between the two.

Here is an example of a dome light, and a GGX specular brdf, using light importance sampling for infinite arealights [10], and microfacets based sampling [5] for the brdf:

![Figure 1: LightSampling, BrdfSampling, MIS(from left to right, equal time renders)](image)

When using only brdf sampling the reflections (Eq. 29) are clean but not the highlights, with light sampling only (Eq. 30), the highlights are nicely resolved but the rest is noisy. Because each sampling technique covers different parts of the integral, when using MIS we get the best result everywhere. We are effectively doubling the number of techniques, since we are shooting twice the number of samples, but even with this additional cost the result is more converged at equal render time (Fig. 1).
### 4.2 BSDF Sampling

In a pathtracer, to maintain good interactivity, we usually want only one path per iteration, i.e a camera ray hitting a surface will generate only one indirect ray to continue the path. However a material used in production will have multiple lobes, such as clearcoat, diffuse, specular, ..., so we need to choose one at every hit. We could just pick and evaluate one lobe, but one way to further reduce the variance is to use one-sample MIS [11]. The computation is the same as normal MIS, the only difference is that here we will pick one sampling strategy $j$ with probability $c_j$ to choose our unique sample.

$$F(y_i) = \frac{\omega_j f(y_i)}{c_j p_j(y_i)}$$  (31)

We will choose one lobe to pick the direction and still compute the whole contribution, and then use the balance heuristic on the PDF. For a brdf with a diffuse, specular and clearcoat:

$$f_r(x, \omega, \omega') = f_d(x, \omega, \omega') + f_s(x, \omega, \omega') + f_c(x, \omega, \omega')$$  (32)

In this example, we have 3 sampling strategies we can choose from, choosing $c_j$ proportional to each albedo, the final PDF $p$ will be independent of the chosen strategy:

$$\frac{1}{p(x)} = \begin{cases} \frac{1}{c_d p_d(x)} \frac{c_d p_d(x)}{c_d p_d(x) + c_s p_s(x) + c_c p_c(x)} & \text{when choosing diffuse} \\ \frac{1}{c_s p_s(x)} \frac{c_s p_s(x)}{c_d p_d(x) + c_s p_s(x) + c_c p_c(x)} & \text{when choosing specular} \\ \frac{1}{c_c p_c(x)} \frac{c_c p_c(x)}{c_d p_d(x) + c_s p_s(x) + c_c p_c(x)} & \text{when choosing clearcoat} \end{cases}$$  (33-35)

$$\frac{1}{p(x)} = \frac{1}{c_d p_d(x) + c_s p_s(x) + c_c p_c(x)}$$  (36)

The cost here is virtually the same as brdf evaluation and sampling represent a very small portion of an actual render (< 1%), for a cleaner result (Fig. 2).

### 4.3 Volume Sampling

Another interesting use case of one sample MIS is volume sampling. The simplest way to sample a volume is to use density sampling, i.e sample the transmittance. The contribution of a ray passing though a scattering volume is:

$$L_e(x', \omega) = \int_S \tau(t) \epsilon(x', \omega, t) dt$$  (37)

with the transmittance $\tau(t)$ defined as:

$$\tau(t) = \int_{t'=0}^t e^{-\sigma t'} dt'$$  (38)

if we use a PDF proportional to $\tau$, for a random number $\chi$, the CDF inversion gives us:

$$t = -\frac{1}{\sigma} \ln(1 - (1 - e^{-\sigma \cdot S}) \chi)$$  (39)

the problem here is that the extinction $\sigma$ is usually a color, so we need to choose one channel to do the sampling. We can pick the max, the min of some kind of average, but whatever choice we
make there are cases where the PDF does not match the extinction anymore and gives high variance results. The problem becomes bigger the more saturated extinction is. Here again, by doing MIS between channels we can get a better image.

When choosing the maximum extinction, the shallow scattering is not resolved enough and we get a noisy yellow scattering. If on the other side we choose to sample the minimum extinction, then in that case the deep scattering is not sampled correctly and we get blue fireflies. The MIS image shows a nice convergence throughout the thickness of the volume (Fig. 3).
5 Conclusion

At first glance it may seem that we only use very basic sampling techniques in production, whereas every year we see novel new algorithms being published. There are a few reasons for this. Contrary to architectural renders, time contraints are severe in movie production where we have to render animations, and even with large render farms, single frame renders need to stay typically within a few hours to few dozen hours. The other important limitation is that when working interactively, lighters get noisy unconverged images quickly but these should nonetheless give them a good estimate of the final result. That means the noise needs to be predictable, and this is one of the reasons we have yet to see good usage of MCMC techniques in production rendering.

One additional variable is that a large part of the scene construction is up to the artist, who can use procedural geometry and patterns on surfaces and volumes, and this introduces complexity and unpredictability.

Only a few algorithms can fulfill all the above contraints and MIS is one of them. It is still possible to sometimes find special cases that are important enough to get special treatment (for example, usage of control variates for diffuse and domelights [6]), but those are the exception. And even MIS is not without problems, because it’s just a combination of many sampling techniques, its cost increases with the number of strategies. We are still therefore looking to try new techniques, from the light integration point of view (better integration of difficult light paths [3] [4]), to better bsdf [1], hair [9], volume models, but also more fundamental sampling improvements [8]. Fortunately experimentation is now much easier since the whole rendering pipeline has moved to raytracing and physically based rendering.

References


A Multilayered Model for Artificial Intelligence of Game Character as Agent Architecture

Youichiro Miyake *
Digital Game Research Association JAPAN

Abstract As all mathematics have a beautiful structure, an inner mind model of Artificial Intelligence has a grand architecture. It consists of information flow and software modules. In this twenty years, an agent’s inner intelligence model has been researched and developed by many game AI programmers in game titles. A whole image of an agent’s intelligent model is explained.

Keywords: agent architecture, blackboard architecture, information flow, affordance, Umwelt

1 Grand design of an agent’s mind

A game character is living in a game world. In these twenty years, a game world becomes much larger and more complex drastically. AI of a game character (character’s mind) is required to be more intelligent, so it also becomes to have a strong intelligent structure in its mind. It is called “Agent Architecture”, which is originally developed in robotics. An agent architecture is a fundamental architecture which connects a game world and an agent’s mind and body(Figure.1). It consists of some modules which have a different function, such as a sensor module, knowledge-making (recognition) module, decision-making module, motion-making module, and memory module. A sensor module gathers and gets information of game world, knowledge-making module forms knowledge from the information which a sensor module delivers, decision-making module make a decision by using and joining some of knowledge together, and motion-making module makes a motion by following a decided decision.

2 Information flow

In the process in an agent architecture and a world, information flows through all the modules and the game world(Figure.1). The flow is called ”Information flow”. Information flow connects an agent and a game world. The information keeps changing and each module bakes temporary information on memory module. So some blocks of information are stacked on memory module with its format. These stacked memories are used by each module. Information flow has the information of the game world such as enemy’s position, change of a terrain and objects, and an event which happens in the game world . They consist of some symbols and float numbers. A form of data in information flow about an object, a terrain, or an event is called KR (Knowledge Representation).
Figure 1: Agent Architecture for game agent[1]. It shows a relation of an agent and a world. It separates a world and agent via sensors and effectors. A big circle means an information flow through a game world and agent inside, which connects a world and an agent itself. A small circle means an information flow inside agent architecture, which organizes inside data in agent architecture.

For example, an enemy's KR is a collection of a coordinate, a velocity vector, an enemy type (symbol), a weapon type (symbol), HP (numerical) and magic power (MP, numerical). An object KR is structured data including affordance (afforded actions to the object), a position, and an object type. In a game, there are many enemies and objects, so information flow has much information. They are transformed in agent architecture. In agent architecture, information flow is abstracted to information flow in higher level, and its information is written on the higher blackboard. This abstraction process is iterated, so information flow is repeatedly abstracted to higher level. By transforming information flow data to more abstract data, AI creates the abstract recognition of the world around itself. Blackboard architecture becomes multi-layered blackboard system combined by KS which processes abstraction.

3 Blackboard architecture in agent architecture

Blackboard architecture is simple architecture with one blackboard, several KS(Knowledge Source)s, and one arbiter(Figure.2, left image). The system is that some KSs write and read information on Blackboard, and their action timing is controlled by one arbiter. An agent architecture of a game character uses a blackboard architecture. Its design is to separate all processing modules (such as knowledge-making, decision-making, and motion-making module) from memory module(Figure.2, right image). A multilayered blackboard architecture consists of several blackboards and KSs(Figure.3). For example, four layered blackboard architecture is like this: On the lowest layer, concrete information is directly written by KS(sensor modules). On the second layer, abstract information is written by abstracting information of the first layer of the lowest layer by KS. On the third layer, more abstract information is written by abstracting the second layer information by
The lowest layer expresses a physical body. The body layer is connected with a game world with a physical body and sensors. The information flow on the lowest layer is most fundamental. On each layer, decision-making is processed, but each function of decision-making is different. For example, the top layer decides an abstract goal or task, but the lowest layer decides a physical body movement. Each layer has a different role for decision-making. There are two kinds of KS; one belongs each layer and the other abstract blackboard information to higher blackboard information. The lowest body layer has strong connection with the world, and the lowest blackboard shows the agent primitive recognition in physical layer. This is called Umwelt which equals a subjective world[4]. The environmental world depends on the body structure, its motions, and life-style. And an object in the world is recognized through multilayers, and each layer has a different abstract representation for the object(Figure.4).

References

http://id.nii.ac.jp/1004/00000517/


http://naimadgames.com/publications.html

Figure 3: Multilayered Blackboard architecture. It has four blackboards. The lowest blackboard represents a physical body, and it is strongly connected with the world via sensors and effectors. There are three higher layers which have abstracted information from the lower layer. Each layer has a decision-making module, but each module has a different scale and role.

Figure 4: Object Representation. Each layer has a different representation for an object in the world. Through multi-layer architecture, an object is recognized in a hierarchical structure.
Abstract The local induction equation, or the binormal flow on space curves is a well-known model of deformation of space curves as it describes the dynamics of vortex filaments, and the complex curvature is governed by the nonlinear Schrödinger equation (NLS). In this paper, we present its discrete analogue, namely, a model of deformation of discrete space curves by the discrete nonlinear Schrödinger equation (dNLS). We also present explicit formulas for both NLS and dNLS flows in terms of the $\tau$ function of the 2-component KP hierarchy.

Keywords: space curves, discrete space curves, nonlinear Schrödinger equation, discrete nonlinear Schrödinger equation, integrable systems, $\tau$ functions, solitons

1 Introduction

The local induction equation (LIE)

$$\frac{\partial \gamma}{\partial t} = \frac{\partial \gamma}{\partial x} \times \frac{\partial^2 \gamma}{\partial x^2},$$

(1)

is one of the most important models of deformation of space curves, where $\gamma(x, t) \in \mathbb{R}^3$ is a smooth space curve parametrized by the arc-length $x$ and $t$ is a deformation parameter. In a physical setting,
it describes the dynamics of vortex filaments driven by the self-induction in the inviscid fluid under
the local induction approximation [6].

It is well-known that if $\gamma$ obeys LIE, then the curvature and the torsion, or equivalently, the complex
curvature of $\gamma$ solves the nonlinear Schrödinger equation (NLS) which is one of the most typi-
cal equations in the integrable systems. To show this, we use the Frenet frame $\Phi = \Phi(x, t) = [T(x, t), N(x, t), B(x, t)] \in SO(3)$, where $T$, $N$, $B$ are the tangent, the normal, and the binormal
vectors defined by

$$T = \gamma', \quad N = \frac{\gamma''}{|\gamma'|}, \quad B = T \times N, \quad ' = \frac{\partial}{\partial x},$$

respectively. Note that it follows that $|T| = |\gamma'| = 1$ since $x$ is the arc-length. Then we have the
Frenet-Serret formula

$$\frac{\partial \Phi}{\partial x} = \Phi L, \quad L = \begin{bmatrix} 0 & -\kappa & 0 \\ \kappa & 0 & -\lambda \\ 0 & \lambda & 0 \end{bmatrix},$$

where $\kappa = |\gamma''|$ and $\lambda = -\langle B', N \rangle$ are the curvature and the torsion, respectively. In this setting,
LIE (1) is expressed as the deformation by the binormal flow

$$\frac{\partial \gamma}{\partial t} = \kappa B,$$

and the corresponding deformation equation of the Frenet frame is given by

$$\frac{\partial \Phi}{\partial t} = \Phi M, \quad M = \begin{bmatrix} 0 & -\kappa & 0 \\ \kappa & 0 & -\lambda \\ \kappa' & -\kappa \lambda & -\kappa'' + \lambda^2 \end{bmatrix}.$$  

The compatibility condition of the system of linear partial differential equations (3) and (5)

$$\frac{\partial L}{\partial t} = \frac{\partial M}{\partial x} - LM - ML$$

yields

$$\frac{\partial \kappa}{\partial t} = -2 \frac{\partial \kappa}{\partial x} \lambda - \kappa \frac{\partial \lambda}{\partial x} = \frac{\partial}{\partial x} \left( \frac{\kappa''}{\kappa} + \frac{\kappa^2}{2} - \lambda^2 \right).$$

Introducing the complex curvature $u = u(x, t) \in \mathbb{C}$ by the Hasimoto transformation[6]

$$u = \kappa e^{\sqrt{-\Lambda}}, \quad \Lambda = \int^x \lambda \, dx,$$

we see that $u$ satisfies NLS

$$\sqrt{-1} \frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{1}{2} |u|^2 u = 0.$$  

Also, one can show that this deformation is isoperimetric, namely $|\gamma'| = 1$ for all $t$.

Discretization of curves and their deformations preserving underlying integrable structures is an
important problem in the discrete differential geometry. For example, the isoperimetric deformation
of plane discrete curves described by the discrete mKdV equation (dmKdV) has been studied in
[12, 14, 15]. For discrete space curves, the deformations by the discrete sine-Gordon equation (dsG)
and dmKdV has been studied in [4, 13, 14], and the deformation by dNLS is formulated in [11, 18].

The dsG and dmKdV describe torsion-preserving isoperimetric and equidistant deformation of the
space discrete curves with constant torsion. However, formulation of discrete deformation of space
discrete curves with varying torsion is a difficult problem. The only example so far is presented by
Hoffmann [10, 11], where he has claimed that composition of certain two isoperimetric equidistant
deformations can be regarded as a discrete analogue of LIE. Also, it was used for numerical simulation of fluid flow [18, 19]. This formulation uses quaternions and its geometric meaning is clear, but description of the deformation parameters in terms of the complex curvature, thus the relation to dNLS are rather indirect.

In this paper, we present a formulation of the dNLS flow on space discrete curves from different approach; the deformation of curves is expressed in terms of the discrete Frenet frame with the coefficients given by the curvature and torsion of the curves explicitly. In this approach, dNLS arises as the equation governing the complex curvature of curves, which is the same as the case of smooth curves. Based on this formulation, we present explicit formulas for the NLS flow for smooth curves and the dNLS flow to discrete curves in terms of $\tau$ functions of the two-component KP hierarchy by applying the theory of integrable systems. We expect that our dNLS flow can be an alternative to Hoffmann’s formulation when it is used to simulate the dynamics of fluids. Also, explicit expression of the scheme and exact solutions may promote further development of theoretical studies of discrete dynamics of discrete curves from both mathematical and physical point of view.

2 Deformations of space curves

Let $\gamma_n \in \mathbb{R}^3$ be a discrete space curve with

$$|\gamma_{n+1} - \gamma_n| = \epsilon,$$

where $\epsilon$ is a constant. We introduce the discrete Frenet frame $\Phi_n = [T_n, N_n, B_n] \in SO(3)$ by

$$T_n = \frac{\gamma_{n+1} - \gamma_{n}}{\epsilon}, \quad B_n = \frac{T_{n-1} \times T_n}{|T_{n-1} \times T_n|}, \quad N_n = B_n \times T_n. \quad (11)$$

Then it follows that the discrete Frenet frame satisfies the discrete Frenet-Serret formula

$$\Phi_{n+1} = \Phi_n L_n, \quad L_n = R_1(-\nu_{n+1}) R_3(\kappa_{n+1}), \quad (12)$$

where

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \quad R_3(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (13)$$

and $\nu_n, \kappa_n$ are defined by

$$\langle T_{n-1}, T_n \rangle = \cos \kappa_n, \quad \langle B_n, B_{n-1} \rangle = \cos \nu_n, \quad \langle B_n, N_{n-1} \rangle = \sin \nu_n, \quad -\pi \leq \nu_n < \pi, \quad 0 < \kappa_n < \pi. \quad (14)$$

In order to formulate a “good” discrete deformation (discretization of time), we resort to the theory of discrete integrable systems to preserve integrable nature of the NLS flow (4). As a discrete analogue of NLS (9), we consider

$$\left(\sqrt{-1} \frac{\epsilon^2}{\delta} - 1 \right) u_{m+1}^m - \left(\sqrt{-1} \frac{\epsilon^2}{\delta} + 1 \right) u_m^m + (u_{n+1}^m + u_{n-1}^m)(1 + \epsilon^2 |u_m^m|^2) \Gamma_n^m = 0,$$

$$\frac{\Gamma_{n+1}^m}{\Gamma_n^m} = \frac{1 + \epsilon^2 |u_{n+1}^m|^2}{1 + \epsilon^2 |u_{n}^m|^2}, \quad (15)$$

which we refer to as the discrete nonlinear Schrödinger equation (dNLS) [1, 2, 7]. Here, $u_m^m \in \mathbb{C}$, $\Gamma_n^m \in \mathbb{R}$, $n$ is the space discrete variable which corresponds to the label of vertices of discrete curves, $m$ is the discrete variable corresponding to the step of deformation, $\epsilon$ and $\delta$ are constants which are
the lattice intervals of \( n \) and \( m \), respectively. Moreover, \( u^{m}_{n} \) is the complex discrete curvature defined by
\[
  u^{m}_{n} = \frac{1}{\epsilon} \tan \frac{\kappa^{m}_{n}}{2} e^{\sqrt{-1} \Lambda^{m}_{n}}, \quad \Lambda^{m}_{n} - \Lambda^{m}_{n-1} = -\nu^{m}_{n},
\]
We impose the boundary condition as
\[
  u^{m}_{n} \to 0 \ (n \to \pm \infty), \quad \Gamma^{m}_{n} \to \Gamma_{\pm \infty} \ \text{(const.)} \ (n \to \pm \infty). \tag{17}
\]
Then one of the main statements of this paper is given as follows:

**Theorem 1 (dNLS flow)**

For a fixed \( m \), let \( \gamma^{m}_{n} \in \mathbb{R}^{3} \) be a space discrete curve satisfying
\[
  |\gamma^{m+1}_{n+1} - \gamma^{m}_{n}| = \epsilon, \tag{18}
\]
and \( \Phi^{m}_{n} = [T^{m}_{n}, N^{m}_{n}, B^{m}_{n}] \in \text{SO}(3) \) be the discrete Frenet frame defined in (11) satisfying the discrete Frenet-Serret formula
\[
  \Phi^{m+1}_{n+1} = \Phi^{m}_{n} L^{m}_{n}, \quad L^{m}_{n} = R_{1}(\Lambda^{m+1}_{n}) R_{3}(\kappa^{m+1}_{n}). \tag{19}
\]
Let \( u^{m}_{n} \) be a complex discrete curvature of \( \gamma^{m}_{n} \). We determine \( u^{m+1}_{n} \) by dNLS (15) under the boundary condition (17) and put \( u^{m+1}_{n} = \frac{1}{\epsilon} \tan \frac{\kappa^{m+1}_{n}}{2} e^{\sqrt{-1} \Lambda^{m+1}_{n}} \). We define a new curve \( \gamma^{m+1}_{n} \in \mathbb{R}^{3} \) by
\[
  \frac{\gamma^{m+1}_{n+1} - \gamma^{m}_{n}}{\delta} = 2 \epsilon \left( P^{m}_{n} T^{m}_{n} + Q^{m}_{n} N^{m}_{n} + R^{m}_{n} B^{m}_{n} \right), \tag{20}
\]

\[
  P^{m}_{n} = \delta \left( -1 + \frac{\Gamma^{m}_{n}}{\cos^{2} \frac{\kappa^{m}_{n}}{2}} \right),
\]
\[
  Q^{m}_{n} = \delta \left[ \tan \frac{\kappa^{m}_{n}}{2} - \tan \frac{\kappa^{m+1}_{n}}{2} \frac{\cos(\Lambda^{m+1}_{n} - \Lambda^{m}_{n})}{\cos^{2} \frac{\kappa^{m}_{n}}{2}} \right],
\]
\[
  R^{m}_{n} = \epsilon^{2} \tan \frac{\kappa^{m}_{n}}{2} - \delta \tan \frac{\kappa^{m+1}_{n}}{2} \frac{\sin(\Lambda^{m+1}_{n} - \Lambda^{m}_{n})}{\cos^{2} \frac{\kappa^{m}_{n}}{2}}.
\]

Suppose that \( \Gamma_{\infty} = 1 \) or \( 1 + \frac{\epsilon^{4}}{2} \). Then, it follows that
\begin{enumerate}
  \item \( |\gamma^{m+1}_{n+1} - \gamma^{m+1}_{n}| = \epsilon \). Namely, \( \gamma^{m+1}_{n} \) is an isoperimetric deformation of \( \gamma^{m}_{n} \).
  \item \( u^{m+1}_{n} \) gives the complex discrete curvature of \( \gamma^{m+1}_{n} \).
\end{enumerate}

Figure 1: Numerical simulation of dNLS flow.
(1) The deformation (21) is not an equidistant deformation in contrast with the deformation described by dmKdV [13]. In fact, one can show that

$$\left| \frac{\gamma_{n}^{m+1} - \gamma_{n}^{m}}{\delta} \right| = \frac{4}{\epsilon^2} \left( -1 + \frac{\Gamma_{n}^{m}}{\cos^2 \frac{\kappa_{m}}{2}} \right).$$

Equation (22) also implies that the solution of dNLS (15) should satisfy the condition $\Gamma_{n}^{m} \geq \cos^2 \frac{\kappa_{m}}{2}$ in order to be consistent with the curve deformation. Note that this property does not contradict with Hoffmann's formulation where the deformation is constructed as composition of two isoperimetric and equidistant deformations.

(2) Continuous limit with respect to time can be simply taken as $t = m\delta$ and $\delta \to 0$. Then dNLS (15) and corresponding deformation equation (20) and (21) yields the semi-discrete NLS equation or the Ablowitz-Ladik equation [1, 2]

$$\sqrt{-1} \frac{du_{n}}{dt} + \frac{u_{n+1} - 2u_{n} + u_{n-1}}{\epsilon^2} = (u_{n+1} + u_{n-1})|u_{n}|^2 = 0,$$

and the deformation equation of space discrete curves [3, 9, 16]

$$\frac{d}{dt} \gamma_{n} = \frac{2}{\epsilon} \tan \frac{\kappa_{n}}{2} B_{n}.$$  

The dNLS flow (20) and (21) implies the deformation of Frenet frame as

$$\Phi_{n+1}^{m} = \Phi_{n}^{m} M_{n}^{m},$$

$$M_{n}^{m} = \frac{1}{\Gamma_{n+1}^{m}} \left[ \begin{array}{ccc}
|\alpha_{n}^{m}|^2 - |\beta_{n}^{m}|^2 & 2\Re (\alpha_{n}^{m} \beta_{n}^{m*}) & -2\Im (\alpha_{n}^{m} \beta_{n}^{m*}) \\
-2\Re (\alpha_{n}^{m} \beta_{n}^{m*}) & \Re (\alpha_{n}^{m} - \beta_{n}^{m}) & -\Im (\alpha_{n}^{m} + \beta_{n}^{m}) \\
-2\Im (\alpha_{n}^{m} \beta_{n}^{m*}) & \Im (\alpha_{n}^{m} - \beta_{n}^{m}) & \Re (\alpha_{n}^{m} + \beta_{n}^{m})
\end{array} \right] \in \text{SO}(3),$$

where $\alpha_{n}^{m}, \beta_{n}^{m} \in \mathbb{C}$ are given by

$$\alpha_{n}^{m} = \sqrt{-1} \frac{\delta}{\epsilon^2} \left[ \left( 1 - \sqrt{-1} \frac{\delta^2}{\Gamma_{n+1}^{m}} \right) - \left( 1 + \epsilon^2 u_{n+1}^{m+1} + u_{n+1}^{m+1*} \right) \Gamma_{n+1}^{m} \right] e^{\sqrt{-1} \left( \Lambda_{n}^{m+1} - \Lambda_{n}^{m} \right)},$$

$$\beta_{n}^{m} = \sqrt{-1} \frac{\delta}{\epsilon} \left( u_{n+1}^{m+1} - u_{n+1}^{m} \right) \Gamma_{n+1}^{m} e^{\sqrt{-1} \left( \Lambda_{n}^{m+1} + \Lambda_{n}^{m} \right)},$$

respectively. Here, * means the complex conjugate. The Frenet-Serret formula (19) and the deformation equation (25) can be transformed to the SU(2) version by the standard correspondence of SO(3) and SU(2) as

$$\phi_{n+1}^{m} = \phi_{n}^{m} L_{n};$$

$$L_{n} = \left[ \begin{array}{cc}
\cos \frac{\kappa_{n+1}^{m}}{2} e^{-\sqrt{-1} \nu_{n+1}^{m}} & -\sin \frac{\kappa_{n+1}^{m}}{2} e^{-\sqrt{-1} \nu_{n+1}^{m}} \\
\sin \frac{\kappa_{n+1}^{m}}{2} e^{\sqrt{-1} \nu_{n+1}^{m}} & \cos \frac{\kappa_{n+1}^{m}}{2} e^{\sqrt{-1} \nu_{n+1}^{m}}
\end{array} \right],$$

$$\phi_{n}^{m+1} = \phi_{n}^{m} M_{n}^{m};$$

$$M_{n}^{m} = \frac{1}{\sqrt{\Gamma_{n+1}^{m}}} \left[ \begin{array}{cc}
\alpha_{n}^{m} & \beta_{n}^{m*} \\
-\beta_{n}^{m} & \alpha_{n}^{m*}
\end{array} \right],$$

which is known as the Lax pair of dNLS [1, 2]. In fact, one can verify that the compatibility condition $L_{n}^{m} M_{n+1}^{m} = M_{n}^{m} L_{n+1}^{m}$ yields dNLS (15).

Outline of the proof of Theorem 1

The first statement may be verified directly in principle, by computing $\gamma_{n+1}^{m+1} - \gamma_{n}^{m+1}$ and its length from (20), (21) and the discrete Frenet-Serret formula (19) under the assumption that $u_{n+1}^{m+1}$ is determined by dNLS (15). However, this computation is hopelessly complicated to carry out. To make it
feasible, we change the Frenet frame to a different frame used in [6, 8], which we call the complex parallel frame in this paper. Let \( F_m = [T_m, U_m, U^{m+} ] \in U(3) \) be the complex parallel frame defined by
\[
U_n^m = \frac{e^{\sqrt{-1}A_n^m}}{\sqrt{2}} \left( N_n^m + \sqrt{-1}B_n^m \right).
\]
(28)

Note that it is related to the discrete Frenet frame \( \Phi_n^m \) as
\[
F_n^m = \Phi_n^m \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{\sqrt{-1}A_n^m} & 0 \\ 0 & 0 & e^{\sqrt{-1}A_n^m} \end{bmatrix}.
\]
(29)

Then the complex curvature \( u_n^m \) naturally arises in this framework; the discrete Frenet-Serret formula \( (19) \) and the deformation of the discrete curve are rewritten in terms of \( u_n^m \) as
\[
F_{n+1}^m = F_n^m X_n^m, X_n^m = \frac{1}{1 + \epsilon^2 |u_{n+1}^m|^2} \begin{bmatrix} 1 - \epsilon^2 |u_{n+1}^m|^2 & -\sqrt{2} \epsilon u_{n+1}^m & -\sqrt{2} \epsilon u_{n+1}^m \noalign{\hrule height 1pt} \sqrt{2} \epsilon u_{n+1}^m & -\epsilon u_{n+1}^m & -\epsilon u_{n+1}^m \noalign{\hrule height 1pt} \sqrt{2} \epsilon u_{n+1}^m & 1 & 1 \end{bmatrix},
\]
(30)

and
\[
\gamma_n^{m+1} = \gamma_n^m + \frac{2 \delta^2}{\epsilon^2} F_n^m \frac{\epsilon}{\sqrt{2}} \left\{ \left( 1 - \sqrt{-1} \frac{\epsilon^2}{\delta} \right) u_n^m - u_n^{m+1} \right\} + \frac{\epsilon}{\sqrt{2}} \left\{ (1 + \epsilon^2 |u_n^m|^2) \Gamma_n^m \right\}.
\]
(31)

respectively. The following lemma plays a crucial role in the proof:

**Lemma 3** Under the boundary condition \( (17) \), it holds that
\[
|\alpha_n^m|^2 + |\beta_n^m|^2 = \Gamma_n^{m+1},
\]
(32)

provided that \( \Gamma_{\pm\infty} = 1 \) or \( 1 + \frac{\epsilon^4}{\delta^2} \).

By using (32), we have after a long but straightforward calculations
\[
T_n^{m+1} = \frac{\gamma_n^{m+1} - \gamma_n^m}{\epsilon} = F_n^m \frac{1}{\Gamma_n^{m+1}} \begin{bmatrix} |\alpha_n^m|^2 - |\beta_n^m|^2 \noalign{\hrule height 1pt} -\sqrt{2} \alpha_n^m \beta_n^m e^{-\sqrt{-1}A_n^m} \noalign{\hrule height 1pt} -\sqrt{2} \alpha_n^m \beta_n^m e^{\sqrt{-1}A_n^m} \end{bmatrix},
\]
(33)

from which we obtain
\[
\frac{|\gamma_n^{m+1} - \gamma_n^m|}{\epsilon} = \left( |\alpha_n^m|^2 - |\beta_n^m|^2 \right)^2 + 4 |\alpha_n^m|^2 |\beta_n^m|^2 = \left( \frac{|\alpha_n^m|^2 + |\beta_n^m|^2}{\Gamma_n^{m+1}} \right)^2 = 1.
\]

This proves the first statement. The second statement is proved as follows. Starting from \( T_n^{m+1} \) (33), we have \( B_n^{m+1} \) and \( N_n^{m+1} \) in terms of \( F_n^m \) by using (11). Then we obtain an expression of \( \Phi_n^{m+1} = [T_n^{m+1}, N_n^{m+1}, B_n^{m+1} ] \) in terms of \( F_n^m \), which can be rewritten as \( F_n^{m+1} = F_n^m Y_n^m \) with a certain matrix \( Y_n^m \in U(3) \) by using (29). This can be also transformed to the deformation equation of discrete Frenet frame of the form \( \Phi_n^{m+1} = \Phi_n^m M_n^m \) with \( M_n^m \) given in (25). Finally one can check that \( \Phi_n^{m+1} \) satisfies the discrete Frenet-Serret formula \( (19) \) for \( \kappa_n^{m+1} \) and \( \nu_n^{m+1} \) determined from the complex curvature \( u_n^{m+1} \). This completes the proof of Theorem 1. \( \square \)
3 Explicit formulas

The formulation of NLS and dNLS flows in terms of the Frenet frame enables us to apply the theory of integrable systems. As an example, we here present explicit formulas of the NLS and dNLS flows in terms of the $\tau$ functions. For the case of plane curves, see [12]. These formulas are established based on the bilinear formalism in the theory of integrable systems by applying suitable reductions and imposing complex structure to $\tau$ functions of the 2-component KP hierarchy, but here we only show the results.

For any $N \in \mathbb{N}$, we first introduce the following three determinants, a $2N \times 2N$ determinant $\tau$, two $(2N + 1) \times (2N + 1)$ determinants $\sigma$ and $\rho$ as

$$
\tau = \begin{vmatrix}
    m_{11}^{(1)} & \cdots & m_{1N}^{(1)} & 1 & \emptyset \\
    \vdots & \ddots & \vdots & \vdots & \vdots \\
    m_{N1}^{(1)} & \cdots & m_{NN}^{(1)} & \emptyset & 1 \\
    -1 & \emptyset & \cdots & \cdots & \cdots \\
    \emptyset & -1 & \cdots & \cdots & \cdots
\end{vmatrix},
$$

(34)

$$
\sigma = \begin{vmatrix}
    m_{11}^{(1)} & \cdots & m_{1N}^{(1)} & 1 & \emptyset & \phi_1^{(1)} \\
    \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
    m_{N1}^{(1)} & \cdots & m_{NN}^{(1)} & \emptyset & 1 & \phi_N^{(1)} \\
    -1 & \emptyset & \cdots & \cdots & \cdots & 0 \\
    \emptyset & -1 & \cdots & \cdots & \cdots & 0 \\
0 & \cdots & 0 & \phi_1^{(2)} & \cdots & \phi_N^{(2)}
\end{vmatrix},
$$

(35)

$$
\rho = \begin{vmatrix}
    m_{11}^{(1)} & \cdots & m_{1N}^{(1)} & 1 & \emptyset & 0 \\
    \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
    m_{N1}^{(1)} & \cdots & m_{NN}^{(1)} & \emptyset & 1 & 0 \\
    -1 & \emptyset & \cdots & \cdots & \cdots & \psi_1^{(1)} \\
    \emptyset & -1 & \cdots & \cdots & \cdots & \psi_N^{(1)} \\
\psi_1^{(2)} & \cdots & \psi_N^{(2)} & 0 & \cdots & 0
\end{vmatrix},
$$

(36)

where $\emptyset$ is the empty block. Then the formulas for NLS flow on smooth curves and dNLS flow on discrete curves are obtained by choosing the entries of determinant as follows:
NLS flow on smooth curves

We choose the entries of determinants as

\[
\begin{align*}
\varphi_i^{(1)} &= p_i^n e^{\xi_i}, \\
\varphi_i^{(2)} &= -1, \\
\psi_i^{(1)} &= 1, \\
\psi_i^{(2)} &= -\left(\frac{1}{p_i}\right)^n e^{\xi_i}, \\
m_{ij}^{(1)} &= -\frac{\varphi_i^{(1)} \psi_j^{(2)}}{p_i + p_j}, \\
m_{ij}^{(2)} &= \frac{1}{p_i^{*} + p_j}, \\
\xi_i &= p_i x - \sqrt{-1} p_i t + \frac{1}{p_i} z + \xi_i^{(0)}, \\
p_i, \xi_i^{(0)} &\in \mathbb{C},
\end{align*}
\]

so that we write \(\tau = \tau_n(x, t; z), \sigma = \sigma_n(x, t; z), \rho = \rho_n(x, t; z)\). Here, \(n\) and \(z\) are regarded as auxiliary variables. Putting \(F = \tau_0, G = -\rho_0, h = -\sigma_2\), we have:

**Theorem 4 (Explicit formula for NLS flow)**

1. Let \(u = u(x, t) \in \mathbb{C}\) be

\[
u = 2 \frac{G}{F},
\]

Then \(u\) satisfies NLS (9).

2. Let \(\gamma = \gamma(x, t) \in \mathbb{R}^3\) be

\[
\gamma = \begin{bmatrix}
\frac{h + h^*}{F} \\
\frac{1}{\sqrt{-1}} \frac{h - h^*}{F} \\
2 \frac{\partial}{\partial z} (\log F) - x
\end{bmatrix}.
\]

Then \(\gamma\) satisfies the Frenet-Serret formula (3) and the deformation equation (4).

Figure 2: Interaction of loops of smooth curve by NLS flow obtained from Theorem 4.

---

1The \(N\)-soliton solution for the tangent vector has been constructed by using the bilinear formalism in [5].
dNLS flow on discrete curves

We choose the entries of determinants as

\[ \varphi_i^{(1)} = p_i^{-n} e^{t_i} (1 - a p_i)^{-m} (1 - c p_i)^{-r}, \quad \varphi_i^{(2)} = - \left( 1 - \frac{a}{p_i} \right)^m \left( 1 - \frac{1}{c p_i} \right)^r, \]

\[ \psi_i^{(1)} = (1 - a p_i)^{-m} \left( 1 - \frac{p_i^*}{c} \right)^{-s}, \quad \psi_i^{(2)} = -(p_i^*)^{-n} e^{s_i} \left( 1 - \frac{a}{p_i^*} \right)^m \left( 1 - \frac{c}{p_i} \right)^r, \]

\[ m_{ij}^{(1)} = - \varphi_i^{(1)} \psi_j^{(2)} / (p_i - p_j), \quad m_{ij}^{(2)} = - \psi_i^{(1)} \varphi_j^{(2)} / (p_i - p_j), \quad e^{s_i} = e^{\frac{1}{2} \frac{1 + c p_i}{1 - c p_i}}. \] (41)

so that we write \( \tau = \tau^m_n(r, s; z), \sigma = \sigma^m_n(r, s; z), \rho = \rho^m_n(r, s; z) \) with \( r, s \) and \( z \) being auxiliary variables. Putting

\[ F^m_n = \tau^m_n(0, 0; z), \quad G^m_n = \rho^m_n(0, 0; z), \quad h^m_n = c^{-n} \sigma^m_n(1, -1; z), \] (42)

we have:

**Theorem 5 (Explicit formula for dNLS flow)**

1. Let \( u^m_n \in \mathbb{C} \) be

\[ u^m_n = \frac{(-1)^m c^{-n - 2m}}{\epsilon} \frac{G^m_n}{F^m_n} \quad \Gamma^m_n = \frac{2a}{c + \frac{1}{c}} \frac{F^m_{n+1} F^m_n}{F^m_{n-1}}. \] (44)

Then \( u^m_n \) satisfies dNLS (15).

2. Let \( \gamma^m_n \in \mathbb{R}^3 \) be

\[ \gamma^m_n = \epsilon \left[ \begin{array}{c} (1)^m \frac{h^m_n + h^m_{n*}}{F^m_n} \\ (1)^m \frac{h^m_n - h^m_{n*}}{\sqrt{-1} F^m_n} \\ 2 \frac{\partial}{\partial z} (\log F^m_n) - n - 2m \end{array} \right]. \] (45)

Then \( \gamma^m_n \) satisfies the Frenet-Serret formula (19) and the deformation equation (20), (21).

Figure 3: Interaction of loops of discrete curve by dNLS flow obtained from Theorem 5.
References

Aesthetic Design with Log-aesthetic Curves and Surfaces

Kenjiro T. Miura* Shizuoka University
R. U. Gobithaasan† University Malaysia Terengganu

Abstract Bézier, B-Spline and NURBS are de facto flexible curves developed for various design intent. However, these curves possess complex curvature function complicating the process of aesthetic shape design. In order to shorten the process, we introduce the fundamental equations of aesthetic curves and surfaces. This paper elaborates on the technicalities of Log-Aesthetic (LA) curves and surfaces along with its practical application for industrial design. It is anticipated that the emerging LA curves and surfaces have good prospects to be the standards for designing aesthetic shapes.

Keywords: fair curves and surfaces, log-aesthetic curve and surface, general equations of aesthetic curves, logarithmic curvature graph

1 Introduction

Fairness metric is a conventional term used to describe the quality of curves and surfaces. According to Farin, a planar curve is fair if it has continuous curvature and consists of few monotonic curvature pieces [2]. In the past, researchers formulated high-quality curves and surfaces using polynomial representation i.e. Bézier, B-spline and NURBS. The underlying principle was to obtain suitable functionals in order to minimize the oscillation of a curvature. Numerous principles are elaborated as chapters in a book edited by Sapidis entitled “Designing Fair Curves and Surfaces” [1].

A well defined Cesáro equation can be employed to produce high quality curves and surfaces. This equation expresses intrinsic properties of curves and surfaces without the presence of traditional polynomials. In this paper, we follow suit to distinguish the terms “fair” and “aesthetic”. In brief, free-form curves and surfaces are constructed with specific formulations rather than polynomials or rationals to represent high quality shapes denoted as aesthetic curves and surfaces.

In recent years, researchers are perfecting the works on aesthetic curves contributing to exponential increase of findings and publications in this arena. However, research pertaining to the formulation of aesthetic surfaces are still at an early stage and offers bright opportunity for CAD community to contribute significantly. Figure 1 shows an Euler diagram depicting the classifications of fair and aesthetic curves. The members in the green circle comprises of the family which involves traditional curves. The orange circle encloses aesthetic curves including the log-aesthetic (LA) curves; which is one of the main focus of this paper. In this paper, we define aesthetic curves as non-polynomial
functions used for aesthetic shape designs. The intersection of these circles represent the curves expressed by traditional formulations as well as specific methods of construction.

Figure 1: Classifications of the researches on fair and aesthetic curves.

Recent advancement on the LA curve has been promising and it has now matured for industrial and graphical design practices. In 2009, Levien and Séquin [3] indicated that LA curves are the most promising curve for aesthetic design. Gobithaan and Miura [5] formulated the generalized log-aesthetic curve (GLAC) in a standard form by representing the gradient of the logarithmic curvature graph (LCG) as a function of its arc length. They also reported that the LCG gradient of Generalized Cornu spiral [6] can be written as a linear function of the LCG gradient [7]. In 2012, Miura et al. reformulated the LA curve using variational principles to obtain minimized functionals for free-form surface design [4]. In the same year, Ziatdnov et al. [8] showed that some LA curves can be expressed by incomplete Gamma functions which shortens the computation time up to 10 times. Recently Meek et al. [9] proved that an unique solution exists for $G^1$ interpolation by using an LA curve segment when $\alpha < 0$. In 2015, Miura et al.[10] proposed another type of aesthetic curve called polar-aesthetic curve for scissors blade design.

2 Log-aesthetic Curves

This section discusses about the details of LA curves and surfaces in a greater depth. We show that the formulation of LA curves has been perfected along with its fundamental properties. In the following sections we further illustrate that its applications for industrial design which proven to be promising. Albeit, the formulation and representation of aesthetic surfaces has been progressing and much efforts are anticipated for practical design. Next section is dedicated to dissect the foundation of LCG which led to the general equations of aesthetic curves.

2.1 Logarithmic Curvature Graphs

Harada’s formulation of the Logarithmic Distribution Diagram of Curvature (LDDC) is based on a quantitative approach which involves tedious curvature radius and its arc length frequency calculation. They highlighted that an aesthetic curve has a linear LDDC plot. However, the length frequency of the curve cannot be evaluated at certain positions on the curve or for arbitrary radius of curvature. Thus, it was not given much attention as a shape interrogation tool.

In 2003, Kanaya et al. [12] showed that for a given curve $C(t) = (x(t), y(t))$, the derivative of the arc length $s$ with respect to the logarithm of the radius of curvature $R = \log \rho$ is given by

$$
\frac{ds}{dR} = \frac{(x'y'' - x''y')(x'^2 + y'^2)^{\frac{3}{2}}}{3(x'x'' + y'y'')(x'y'' - x''y') - (x'^2 + y'^2)(x'y''' - x'''y')} \tag{1}
$$
where \( \dot{} \) denotes the derivative with respect to parameter \( t \). The quantitative LDDC approach is mathematically equivalent to equation (1) where horizontal and vertical coordinates represent \( R \) and \( \log |ds/dR| \), respectively. Thus, equation (1) is sufficient to analytically define the LDDC plot. However, it does not provide concrete conditions in order to approximate LDDC with a straight line or it does indicate the slope of the approximated line to represent aesthetic shapes. Furthermore, for a curve whose shape is obtained from its image data, only discrete data are available and the partial arc length \( s_j \) must be a finite value to calculate the length frequency.

Hence algebraic manipulation was carried out to reformulate the LHS of Eq. (1). Since the LDDC graph is expressed by \( \log |ds/d(\log \rho)| \) and both \( s \) and \( \rho \) are functions of the parameter \( t \), we can further simplify as follows

\[
\log \left| \frac{ds}{d(\log \rho)} \right| = \log \left| \frac{d}{dt} \left( \rho \frac{ds}{d\log \rho} \right) \right| = \log \rho \frac{ds}{d\log \rho}
\]

\[
= \log \rho + \log s_d - \log \left| \frac{d\rho}{ds} \right|
\]

(2)

where \( s_d = ds/dt \). Equation (2) is defined by the radius of curvature and its derivative. It describes the relationship between the radius of curvature and the derivative of the arc length more explicitly as compared to Eq. (1). This new analytical approach is denoted as Logarithmic Curvature Graph (LCG).

### 2.2 First & Second Fundamental Equations of Aesthetic Curves

In this section, we derive the equation of a curve that produces LCG strictly as a straight line. The curves obtained satisfying such a condition are defined as aesthetic curves and it is the fundamental equations of aesthetic curves [11].

If we assume that the LCG graph of a given curve is strictly expressed by a straight line, on the LHS of Eq. (2) there is a constant \( \alpha \) and

\[
\log \left| \frac{ds}{d\rho} \right| = \alpha \log \rho + C
\]

(3)

where \( C \) is a constant. By transforming Eq. (3), we obtain

\[
\frac{1}{\rho^{\alpha-1}} \frac{ds}{d\rho} = e^C = C_0
\]

(4)

Hence,

\[
\frac{ds}{d\rho} = C_0 \rho^{\alpha-1}
\]

(5)

If \( \alpha \neq 0 \),

\[
s = C_0 \frac{\alpha}{\alpha} \rho^\alpha + C_1
\]

(6)

In the above equation, \( C_1 \) is an integral constant. Therefore

\[
\rho^\alpha = C_2 s + C_3
\]

(7)

where \( C_2 = \alpha/C_0 \) and \( C_3 = -(C_1 \alpha)/C_0 \). Here we rename \( C_2 \) and \( C_3 \) to \( c_0 \) and \( c_1 \), respectively. Then

\[
\rho^\alpha = c_0 s + c_1
\]

(8)
The above equation indicates that the $\alpha$-th power of the radius of curvature $\rho$ is given by a linear function of the arc length $s$. The above equation is named as the first fundamental equation of aesthetic curves [11].

For the case of $\alpha = 0$,

$$s = C_0 \log \rho + C_1$$

(9)

Hence,

$$\rho = C_2 e^{C_3 s}$$

(10)

where $C_2 = e^{-C_1/C_0}$ and $C_3 = 1/C_0$. We rename $C_2$ and $C_3$ as $c_0$ and $c_1$, respectively. We get

$$\rho = c_0 e^{c_1 s}$$

(11)

The $\rho$ is given by an exponential function of $s$. The above equation is named as the second fundamental equation of aesthetic curves [11].

It is known that logarithmic spirals and clothoids are regarded as high-quality curves as explained in the next section. One of the principal characters of the logarithmic spiral is that its radius of curvature and arc length are proportional. Hence, the logarithmic spiral satisfies Eq. (8) and its $\alpha$ is equal to 1. Additionally, the main property of clothoids is that its radius of curvature is inversely proportional to its arc length. Thus, Eq. (8) is satisfied for the clothoid if $\alpha$ is given by $-1$.

In summary, the general equations of aesthetic curves expressed by Eq. (8) encompasses beautiful curves such as logarithmic spirals and clothoids. In fact, Nielsen’s spiral [13] is also expressed by Eq. (11). The curves which satisfy the first and second fundamental equations of aesthetic curves are denoted as Log-aesthetic (LA) curves, which was coined by Professor Carlo H. Séquin from University of California, Berkeley during the presentation of this work at the International CAD Conference & Exhibition 2007 at Honolulu, Hawaii.

### 2.3 Parametric Expression of LA Curve

In this section, we describe the parametric expressions of the fundamental equations for aesthetic curves as Eqs. (8) and (11). Let a curve $C(s)$ satisfies Eq. (8). Then

$$\rho(s) = (c_0 s + c_1)^{\alpha}$$

(12)

Since $s$ is the arc length, $|s_d| = 1$ (refer to, for example, [2]) and there exists $\theta(s)$ satisfying the following two equations:

$$\frac{dx}{ds} = \cos \theta, \quad \frac{dy}{ds} = \sin \theta$$

(13)

Since $\rho(s) = 1/(d\theta/ds)$,

$$\frac{d\theta}{ds} = (c_0 s + c_1)^{-\frac{1}{\alpha}}$$

(14)

Hence, if $\alpha \neq 1$,

$$\theta(s) = \frac{\alpha(c_0 s + c_1)^{\frac{\alpha-1}{\alpha}} + c_2}{(\alpha - 1)c_0}$$

(15)
If the starting point of the curve is given by \( P_0 = C(0) \), thus LA curve can be represented in a complex plane as follows

\[
C(s) = P_0 + e^{ie_2} \int_0^s e^{i\alpha(c_0u + c_1) \frac{\alpha - 1}{\alpha}} \, du
\]  

(16)

The above expression can be regarded as an extension of the clothoid curve where power of \( e \) in its definition is changed from 2 to \( \alpha + 1 \) and its LCG gradient can be specified to be equal to any value except for 0.

Similarly, the second fundamental equation of aesthetic curves are expressed by Eq. (11),

\[
\frac{d\theta}{ds} = \frac{1}{c_0} e^{-c_1 s} \quad \text{(17)}
\]

\[
\theta = -\frac{1}{c_0 c_1} e^{-c_1 s} + c_2 \quad \text{(18)}
\]

Therefore the curve is given by

\[
C(s) = P_0 + e^{ie_2} \int_0^s e^{-c_1 s} e^{-c_1 s} \, ds
\]  

(19)

Figure 2 shows LA curves with various \( \alpha \) values.

![Figure 2: LA curves whose LCG gradients are given by various \( \alpha \), top: curves with negative \( \alpha \), bottom: those with positive \( \alpha \) and two negative values \( -2 \) and \( -1 \) for comparisons.](image)

### 3 Log-aesthetic Spline

In this section, we describe a method to simultaneously specify endpoints, tangent vectors and its curvatures (\( G^2 \) Hermite data) using an LA spline which consists three LA segments. The idea was obtained from Lan et al.’s work [14] where they solve the \( G^2 \) Hermite interpolation problem using triple clothoids. Miura et al. [15] used the shape parameter \( \alpha \) as an additional parameter to make the end curvature as 0. Since \( \alpha \) is related to impressions of the shape [16], hence \( \alpha \) is fixed as a constant to produce \( G^1 \) Hermite interpolation using a single LA curve segment and a \( C^3 \) continuous compound-rhythm LA curve is connected with two LA segments.

\( \alpha \) is usually regarded as a parameter which can be controlled by designers. Thus, we do not use it to satisfy any geometric constraints to shape LA curves. Note that the degree of freedom (DoF) of the LA spline with three segments is similar to triple clothoids. It is a common practice to fix the value of \( \alpha \) to design aesthetic shapes using LA curves. An LA spline consists of three LA curve segments with different \( \alpha \) values which are joined with \( G^2 \) continuity [17].
3.1 Initial Value Estimation

We need initial values for parameters to define an LA spline. To obtain these initial values, we estimate curvatures at the two joints of the LA curve segments. We may use a Bézier curve of degree 5 for the estimation of the initial values to shape an LA spline. Let the total length of the Bézier curve as \( h \). In general, Bézier curves are not uniquely determined by endpoints, tangent directions and its curvatures; these conditions do not necessarily yield a suitable curve for the initial value estimation. Hence we use an objective function which is modified to be independent from the total length \( h \) as proposed by Miura et al. [17]:

\[
J_{LAC} = \sum_{i=1}^{3} \int_{S_{i-1}h}^{S_{i}h} \sqrt{1 + \alpha_i^2 \rho_i^2 \rho_i^2} \, ds \tag{20}
\]

where \( S_0 = 0, S_1 = 0.25, S_2 = 0.75 \) and \( S_3 = 1 \). The above function is minimized to generate an appropriate initial Bézier curve. Fig. 3(a) shows a Bézier curve of degree 5 and its initial control points in green and those after optimization in blue for \( \alpha_i = -0.5, i = 1, 2, 3 \).

Upon using the Bézier curve after optimization, an LA spline curve with three segments shown in red is determined. To be precise, the LA spline curve does not strictly minimize the objective function in Eq. (20), but note that the shapes of the Bézier curve and LA spline are almost in a similar shape. In this example, parameters calculated from the input of Bézier curve are not appropriate, hence the numerical calculation diverges because the total length \( h \) becomes negative. If we use the Bézier curve after optimization, we may obtain these values without calculation failure and generate an LA spline successfully. Since an optimization process is necessary only for the initial value estimation, we propose a simpler method as shown in Fig. 3(b). Let the lengths between the first and the second control points and the fifth and sixth control points be \( l_0 \) and \( l_3 \), respectively. Furthermore let parameters to determine the positions of the \( 3^{rd} \) and \( 4^{th} \) be \( l_1 \) and \( l_2 \), respectively. We change these parameters independently in the range of \( 0.05 \leq l_i / h \leq 0.5 \) for \( i = 0, 1, \cdots \) by 0.05 where \( h \) is the total arc length of the input Bézier curve. We may now obtain parameter values easily which minimizes the objective function in Eq.(20).

Figure 4 depicts Bézier curve of degree 5 (black) and an LA spline (red) generated using the proposed method. The Bézier curve is generated and deformed by built-in commands of a commercial CAD system to satisfy \( G^2 \) Hermite data posed at its endpoints. The LA spline is generated with similar \( G^2 \) Hermite data to achieve \( G^2 \) continuity at the joints. These curves are also shown separately for visual clarity. The curvature plot of these curves is drawn in blue. It is visually clear that the curvature of the Bézier curve varies drastically in order to satisfy \( G^2 \) continuity at its endpoints whereas the LA spline joins gradually.

Figure 5 shows a practical design of a car using LA splines. Figure 5(a) shows iso-parametric lines of the free-form surface generated using LA splines and its corresponding zebra maps. Figure 5(b) depicts the CAD model with a special lighting condition and 5(c) are photos of its mock-up.
manufactured based on the CAD model. To note, the roof of the car is designed by a LA spline curve with three segments and its zebra maps indicates the surface is of high quality. Based on our experience on various aesthetic shape design, LA splines can be used to satisfy any $G^2$ Hermite data.

Figure 5: A car model designed by using LA spline and its mock-up.

4 Variational Formulation of Aesthetic Shapes

In this section, we show on approximating aesthetic shapes with regard to the functional which LA curve minimizes [19]. Then, we extend the functional to formulate log-aesthetic surfaces [4].

4.1 Variational Formulation of LA Curves [19]

If we substitute $\rho^\alpha$ with $\sigma$ in Eq. (8), the equation is then given by

$$\sigma = cs + d$$  \hspace{1cm} (21)
The above equation indicates that LA curves are the representation of a straight line in the $s - \sigma$ plane where the horizontal and vertical axes are the arc length $s$ and $\sigma = \rho^\alpha$ respectively which connects two given points $(s_1, \beta_1)$ and $(s_2, \beta_2)$. In this case, the following objective functional is minimized.

$$J_{LAC} = \int_{s_1}^{s_2} \sqrt{1 + \sigma^2} ds = \int_{s_1}^{s_2} \sqrt{1 + \alpha^2 \rho^{2\alpha - 2} \rho^2} ds$$  \hfill (22)

### 4.2 Variational Formulation of LA Surfaces [4]

Here, we apply the variational principle to the surface formulation. We let the curvature of the curve $\kappa$ and the arc length $s$ correspond to the Gaussian curvature $K$ and the surface area $S$, respectively. From Eq. (22), when $\alpha = -1$, $\kappa_s = -\rho_s / \rho^2$ and we obtain the following equation.

$$J_{LAC} = \int_{s_1}^{s_2} \sqrt{1 + \kappa^2} ds$$  \hfill (23)

By reparameterizing the above equation with $s = s(t)$, it becomes

$$J_{LAC} = \int_{t_1}^{t_2} \sqrt{x_t^2 + y_t^2 + \kappa_t^2} dt = \int_{t_1}^{t_2} \sqrt{\lambda_C + \kappa_t^2} dt$$  \hfill (24)

where $s_1 = s(t_1)$, $s_2 = s(t_2)$, and $\lambda_C = \sqrt{x_t^2 + y_t^2}$. Note that $ds/dt = \lambda_C$.

By extending Eq. (24) into the surface, we define the objective functional for the surface $J_{LAS}$ as follows:

$$J_{LAS} = \int_{u_1}^{u_2} \int_{v_1}^{v_2} \sqrt{\det(I) + K_u^2 + K_v^2} du dv$$  \hfill (25)

where $I$ is a matrix expressed with the first fundamental form by

$$I = \begin{bmatrix} E & F \\ F & G \end{bmatrix}$$  \hfill (26)

where $E = S_u \cdot S_u$, $F = S_u \cdot S_v$ and $G = S_v \cdot S_v$. Note that the area of the surface $S$ is given by

$$S = \int_{u_1}^{u_2} \int_{v_1}^{v_2} \sqrt{\det(I)} du dv$$  \hfill (27)

We assume local parameterization $(s, t)$ around a point on the surface so that the tangent vectors with respect to the parameters are orthogonal to each other, their directions are the same as the principal direction, and the norm of each tangent vector is equal to 1. With this parameterization, $I$ becomes the $2 \times 2$ unit matrix. By performing integration around the point $S(s_1, t_1)$, Eq. (24) can be rewritten as

$$\Delta J_{LAS} = \int_{s_1}^{s_1 + \Delta s} \int_{t_1}^{t_1 + \Delta t} \sqrt{1 + K_u^2 + K_v^2} ds dt$$  \hfill (28)

According to the general principle of variational principle, to minimize the following functional

$$J = \int_{s_1}^{s_2} \int_{t_1}^{t_2} g(K, K_s, K_t, s, t) ds dt$$  \hfill (29)
the following equation should be satisfied.

\[
\frac{\partial g}{\partial K} - \frac{\partial}{\partial s} \frac{\partial g}{\partial K_s} - \frac{\partial}{\partial t} \frac{\partial g}{\partial K_t} = 0 \tag{30}
\]

Note that \( g = \sqrt{1 + K_s^2 + K_t^2} \) does not explicitly depend on \( K \). Eq. (30) yields

\[
(1 + K_s^2)K_{ss} - 2K_s K_t K_{st} + (1 + K_t^2)K_{tt} = 0 \tag{31}
\]

The above equation is called the minimal surface or Lagrange’s equation and the surface \( S(s, t) = (s, t, K(s, t)) \) is given by a minimal surface. Therefore, in a case where the Gaussian curvature on the boundary is specified, the Gaussian curvature should be given by a minimal surface interpolating the boundary values. The above discussion assumes local isometric parameterization whereby global isometric parameterization does not exist in general. It is not possible to deal with the case where the functional is defined globally as in Eq. (25). In such cases, some optimization technique should be adopted to minimize the functional to generate a desired surface.

According to Bernstein’s theorem [18], if the boundary of the surface is located infinitely far, the minimal surface is given by a plane. Therefore, the Gaussian curvature is given by

\[
K(s, t) = c_0 s + c_1 t + c_2 \tag{32}
\]

where \( c_0, c_1, \) and \( c_2 \) are constants.

For further extension, we may use the mean curvature \( H \) instead of the Gaussian curvature \( K \). In this paper we do not elaborate the effects of the power \( \alpha \). To take into account the effects of the power, we may use \( \kappa_{\alpha \text{max}} \kappa_{\beta \text{min}} \) where \( \kappa_{\text{max}} \) and \( \kappa_{\text{min}} \) are the maximum and minimum normal curvatures, respectively. For example, an objective functional may be defined by

\[
J_{\text{LAS}} = \int_{u_1}^{u_2} \int_{v_1}^{v_2} \sqrt{\det(I)} + \{(\kappa_{\alpha \text{max}} \kappa_{\beta \text{min}})_u\}^2 + \{(\kappa_{\alpha \text{max}} \kappa_{\beta \text{max}})_v\}^2 dudv
\]

These extensions are proposed as topics for future research.

5 Conclusions

This paper reviews on fair curves and surfaces leading towards the formalization of aesthetic curves and surfaces. LA curves depict promising properties for practical applications and we hope it will be used as one of the standard curves for industrial and graphical design in the near future. Much effort for theoretical as well as practical researches are anticipated in order to define and utilize LA surface for various design feats.

Acknowledgements

This work was supported in part by JSPS Grant-in-Aid for Scientific Research (B) Grant Number 25289021, JSPS Grant-in-Aid for Challenging Exploratory Research Grant Number 26630038, JST RISTEX Service Science, Solutions and Foundation Integrated Research Program and FRGS grant (FRGS: 59265) provided by University Malaysia Terengganu and Ministry of Education Malaysia.
References

Fabrication-Aware Geometry Processing

Daniele Panozzo*
ETH Zurich

Abstract The advent of commodity 3D manufacturing is increasing the demand for advanced design tools that make the designed shapes physically realizable. In this talk, I will present fabrication-aware algorithms to solve classical geometry processing problems such as intersection-free mesh deformation, surface parametrization and semi-regular meshing. In particular, I will focus on the interactive design of tangent vector fields, whose applications include the conversion of raw point clouds into coarse control grids, the design of planar tessellations and the design of self-supporting masonry buildings. To conclude the talk, I will give an overview of recent techniques that use mesh parametrization algorithms to apply a texture map to real-world objects using hydrographic printing.

Keywords: Geometry Processing, Digital Fabrication, Self-Supporting Surfaces, Hydrographic Printing, Quadrilateral Meshing, Appearance-Mimicking Surfaces.

1 Extended Abstract

The advent of commodity 3D printing is revolutionizing the way people think about designing and prototyping: a designer can now hold in her hands a 3D object hours after its design is complete, drastically reducing costs and enabling quick iterations over many designs. Additive manufacturing enables new applications that were impossible with traditional production processes.

However, the majority of software tools and algorithms currently used to create, manipulate and process digital geometry are not fabrication-aware: they model the shape as an abstract entity that often does not satisfy practical requirements such as stability, robustness or lack of self-intersections. This leads to a large gap between the digital design and the physical fabrication, which is the major obstacle preventing digital fabrication to become mainstream and to deeply change our working habits, in a way similar to the introduction of inkjet and laser printers. In this talk, I will present a series of works that strive to fill this gap, providing computational design tools that rely on numerical optimization to create fabrication-ready designs, which can be directly fabricated using digital fabrication technologies.

Avoiding Self-Intersections: A 3D model must be free of self-intersections to be suitable for fabrication. Technically, this means that no elements should intersect (in the case of thin layers such as cloth) and that no volumetric element has negative volume (in case of volumetric representations). This requirement is surprisingly difficult to enforce, especially during design [1], and it is often omitted in favor of simplicity and speed. While this is not a critical problem for models used in movies or games (the overlaps will often not be visible), it becomes mandatory to solve before models can be
Figure 1: Preventing self-intersections during modeling leads to more intuitive results that are ready to be 3D printed.

A 3D printer cannot print infinitesimally thin layers, and it thus needs to fill the interior of the shape, which is not defined and impossible to compute in presence of self-intersections.

Avoiding self-intersections during deformation is particularly simple if each point is restricted to move on a ray pointing the origin [2]. While this reduces considerably the deformation space, this special deformation is ideal to automatically create appearance mimicking surfaces starting from triangle meshes (Figure 2). The produced surfaces are, by construction, free of self-intersection and ready to be fabricated via 3D printing.

Figure 2: A collection of appearance-mimicking surfaces.

**Planar panelization:** Quadrilateral meshing algorithms are gaining popularity in the graphics community to convert high resolution triangle meshes into coarse control grids for Catmull-Clark subdivisions. They can also be applied to design planar tessellations, which are ideal for glass and steel construction, due to the low cost of producing flat glass panels. Starting from a planar tessellation, a building can be constructed by replacing each face with a flat glass panel, that is much less expensive to manufacture. Mathematically, the edges of a quadrilateral mesh with flat faces define
Planar quadrilateral meshes are used in architectural geometry to design free-form glass and steel structures. The designer specifies a set of alignment constraints (left), the constraints are interpolated in a conjugate direction field (middle) that is automatically converted into a mesh with planar faces (right).

A conjugate field [3], which can be designed with a simple and efficient algorithm [4] that allows architects to interactively experiment with different planar tessellations by simply specifying a set of desired alignment constraints.

**Free-Form Masonry Structures:** The automatic creation of quadrilateral meshes can be used to design and tessellate of free-form masonry structures [5, 6]. These structures are composed of unsupported stone blocks and they stand thanks to their special geometry where all blocks are in static equilibrium. The block pattern used is a quadrilateral mesh, where an edge every two is removed to create a staggering effect that increases the interlocking between the pieces, simplifying the construction and improving the structural properties of the masonry building.

Figure 4: An input surface is automatically transformed into a masonry 3D model. The equilibrium of the surface is represented by two planar graphs that encode the directions and magnitudes of all forces. The generated blocks are 3D-printed and assembled into a physical model of the surface that stands in compression without using glue or reinforcements.

**Hydrographics Printing:** In the digital world, assigning arbitrary colors to an object is a simple operation thanks to texture mapping. However, in the real world, the same basic function of applying colors onto an object is far from trivial. One can specify colors during the fabrication process using a color 3D printer, but this does not apply to already existing objects. Paint and decals can be used during post-fabrication, but they are challenging to apply on complex shapes. I will introduce a method to enable texture mapping of physical objects, that is, to allow one to map an arbitrary color image onto a three-dimensional object [7]. The approach builds upon hydrographics, a technique to transfer pigments printed on a sheet of polymer onto curved surfaces.
Figure 5: We start with a real-world object and a digital 3D model of this object. Using off-the-shelf 3D modeling software, we define a color texture on the digital model. Our algorithm then automatically generates a flat image that we print on a polymer film. We use hydrographics (water transfer printing) to apply this texture onto the real-world object. Our approach compensates for the deformation that happens during the transfer process, so that the final result looks like what we specified on the 3D model.

References


Drawing Curves

Toshio Oshima*
Josai University

Abstract A method drawing curves is proposed. A program drawing graphs of mathematical functions using this method is realized in a computer algebra and outputs the graphs in a source file of \( \text{\LaTeX} \) and then transforms it into a PDF file.

Keywords: Bézier curve, spline curve, computer algebra, Risa/Asir, \( \text{\LaTeX} \), TikZ, 3D graph

1 Introduction

Since the last year I have a class of calculus in my university and show graphs of functions such as \( f(x, y) = x^2 - y^2 \). I have been developing a library os_muldif.rr [1] of a computer algebra Risa/Asir [3] to realize my research explained in [2] and then I added some functions in the library for such educational purpose including calculus, linear algebra and elementary number theory. The library is an open source and can be equally executed by a personal computer with any one of the operating systems Windows, Mac and UNIX.

In fact, a function in the library executes the following procedure to get the graphs.

\[
\text{Risa/Asir} \xrightarrow{\text{output}} \text{TiKZ or XY-pic in a \LaTeX\ file} \xrightarrow{\text{\LaTeX}} \text{DVI file} \xrightarrow{\text{dvipdfm}} \text{PDF file} \quad (1)
\]

2 Curves

Consider a curve

\[
C : [a, b] \ni t \mapsto \gamma(t) = (x(t), y(t)) \in \mathbb{R}^2.
\]

We choose points in \( [a, b] \), namely, \( P_j = \gamma(t_j) \in C \) with \( a = t_0 < t_1 < t_2 < \cdots < t_N = b \) and draw a certain curve \( C' \) starting from \( P_0 \), exactly passing through \( P_1, \ldots, P_{N-1} \) in this order and ending at \( P_N \). We request the following conditions.

- \( C' \) is determined only by \( \{P_0, P_1, \ldots, P_N\} \).
- \( C' \) is a good approximation of \( C \) and it is free from its final resolution in drawing.
- Smaller size of data (i.e. the number \( N \)) and an output in a popular format are desirable.
- The curve can be described in a usual \( \text{\LaTeX} \) source file.

One of the way to realize it is to connect the points by Bézier curves and use TikZ and/or XY-pic which are in a package of a \( \text{\LaTeX} \) system (cf. (1)).

*t-oshima@josai.ac.jp
2.1 Smooth curves

A Bézier curve of degree \( n \) is

\[
[0, 1] \ni t \mapsto P(t) = P(B_0, \ldots, B_n; t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1 - t)^{n-i} B_i
\]  

(3)
determined by \( (n + 1) \) points \( B_0, \ldots, B_n \).

Note that \( P(B, B'; t) \) is the point internally dividing the line segment \( BB' \) by \( t : 1 - t \). Since \( P(B_0, \ldots, B_n; t) = P(P(B_0, B_1; t), P(B_1, B_2; t), \ldots, P(B_{n-1}, B_n; t); t) \), the point \( P(t) \) is geometrically described. For example, the cubic Bézier curve is

\[
P(t) = P(B_0, B_1, B_2, B_3; t) = P\left(P\left((B_0, B_1; t), (B_1, B_2; t), (B_2, B_3; t); t\right)\right)
\]

The curve starts from \( B_0 \) to the direction \( \overrightarrow{B_0B_1} \) and ends at \( B_3 \) to the direction \( \overrightarrow{B_2B_3} \). It does not necessarily pass through \( B_1 \) nor \( B_2 \).

Consider a curve \( C \) passing through \( P_0, P_1, P_2, P_3 \) in this order. We simulate the curve segment of \( C \) connecting \( P_1 \) to \( P_2 \) by the cubic Bézier curve \( P(P_1, Q, R, P_2; t) \) with the control points \( Q, R \) defined as above. Then the number \( c \) is determined by

\[
c = \frac{4P_1P_2}{3(P_0P_2 + P_1P_3)} \frac{1}{1 + \sqrt{1 - \frac{(P_0P_2 + P_1P_3)^2}{(P_0P_2)^2 + (P_1P_3)^2}}} \tag{4}
\]

so that the curve simulates the arc if \( P_0, \ldots, P_3 \) are on a circle with a center \( O \) and moreover \( P_0P_1 = P_1P_2 = P_2P_3 \). In this case, the error \( \frac{\text{arc}}{\text{arc'}} - 1 \) is less than \( \frac{1}{640} \) (resp. \( \frac{1}{3600} \)) if \( \angle P_1OP_2 \leq 120^\circ \) (resp. \( \leq 90^\circ \)). Note that a Bézier curve never coincides with an exact arc. For a closed curve \( C \) passing through points \( R_0, R_1, \ldots, R_N = R_0 \) in this order we draw a curve segment between \( R_i \) and \( R_{i+1} \) by putting \( P_i = R_{i+1} \) for \( i = 0, 1, 2 \) and 3 as in the above and \( R_{\nu+\pm N} = R_{\nu} \) (\( \nu = 1, \ldots, N \)). Then the resulting curve \( C' \) we draw is a smooth closed curve (of class \( C^1 \)) which simulates \( C \).

When the number \( c \) is fixed to be \( \frac{1}{n} \) in the above, the corresponding curve is known as the (uniform) Catmull-Rom spline curve. This curve is invariant under affine transformations and our curve is invariant under conformal affine transformations.

The following first example is the curve drawn by the three points \((\cos t, \sin t)\) with \( t = 0, \frac{2\pi}{3}, \frac{4\pi}{3} \) indicated by \( \bullet \). The other 6 points calculated by using (4) are indicated by \( \times \). In the final PDF file the positions of these 9 points are only written and the real rendering of the Bézier curve is done by a viewer of the file and therefore the size of the PDF file is small. The second example is the (uniform/centripetal) Catmull-Rom spline curve passing through these three points.
The other examples below are the Lissajous curve \( \gamma(t) = (\sin 2t, \sin 3t) \) drawn by the points corresponding to \( t = \frac{2\pi j}{N} \) for \( j = 0, \ldots, N \).

\[
\begin{align*}
N = 3 & \quad (3 \text{ points}) & \text{Catmull-Rom spline} & \quad N = 24, N = 96 & \quad N = 12 & \quad (9 \text{ points})
\end{align*}
\]

If the points \( P_j = \gamma(t_j) \) are not suitably chosen, the resulting curve drawn by the points may be not good. Even in this case our curve is better than the corresponding Catmull-Rom spline curve as in the following example.

Taking the points on the curve \( \gamma(t) = (t, t^2) \) corresponding to \( t = -2, -1, 0, 0.2, 1, 2 \), we draw curve for \( -1 \leq t \leq 1 \) by these 6 points.

\[
y = x^2 \quad (-1 \leq x \leq 1)
\]

piecewise linear  
Catmull-Rom spline  
Our curve

### 2.2 Singularities

We consider a curve \( \gamma(t) (t \in [a, b]) \) which has singular points or discontinuous points. We assume that the curve is a finite union of smooth curves but we do not know the singular points of the curve.

First we choose points \( P_j = \gamma(t_j) \) with \( t_0 = a < t_1 < \cdots < t_N = b \) on the curve. We put \( t_j = \frac{(N-j)a+jb}{N} \) in most cases (or as default)\(^1\). Then we recursively choose more points on the curve up to certain times if necessary as in the following way.

- If the number \( 1 - \frac{P_{j-1}P_jP_{j+1}}{P_{j-1}P_jP_{j+1}} \) exceeds a given threshold value, we add the points \( \gamma\left(\frac{t_j-t_{j-1}}{2}\right) \) and \( \gamma\left(\frac{t_{j+1}-t_j}{2}\right) \).
- If the length \( P_{j-1}P_j \) exceeds a given threshold value, we add the point \( \gamma\left(\frac{t_j+t_{j+1}}{2}\right) \).

If the length \( P_{j-1}P_j \) still exceeds a given threshold value after this procedure, we cut our curve between two points \( P_{j-1} \) and \( P_j \).

We show examples:

\[
y = |2 \sin x| - \left[|2 \sin x|\right] \quad (0 \leq x \leq 5)
\]

continuous at \( x = \frac{\pi}{6}, \frac{5\pi}{6}, \frac{7\pi}{6} \)
not smooth at \( x = \pi \)

\([t] : \text{the largest integer which does not exceed } t.\)

\(^1\)Moreover if the curve is defined outside \([a, b]\), we use the points \( P_{-1} \) and \( P_{N+1} \) to define Bézier curves.
It takes less than a second to get the following graph of Riemann’s zeta function in a PDF file.

\[ y = \left| \zeta\left(\frac{1}{2} + x\sqrt{-1}\right) \right| \]

3 3D graphs

Our main purpose is to draw graphs of surfaces defined by \( z = f(x, y) \) with mathematical functions \( f(x, y) \). Using our method (1), we draw curves on a surface defined by the condition that \( x \) is constant or \( y \) is constant. It takes \( 10 \sim 30 \) seconds to get a required PDF file after a command in Risa/Asir if \( f(x, y) \) is a simple rational function. We can use TikZ and XY-pic. In contrast to XY-pic the source text in TikZ is more readable, easy to be edited and has stronger abilities such that it supports coloring and filling region by a pattern but is takes a little longer time to be transformed into a PDF file. Hence our library supports both of them.

We give two examples:

- \( z = |\sin z| \) \quad \( (z = x + yi, -\pi \leq x \leq \pi, -1 \leq y \leq 1) \)  
  angle \((50^\circ, 15^\circ)\)  
  ratio \(1 : 3 : 3\)  
  \( (x, y) \approx (0.144, 0.144) \)

- \( z = \frac{x^2}{x^2 + y^2} \) \quad \( (-1 \leq x \leq 1, -1 \leq y \leq 1) \)  
  angle \((70^\circ, 20^\circ)\)  
  \( (x, y) \approx (0.12, 0.12) \)  
  This function is discontinuous at \((x, y) = (0, 0)\)

References


Attractive plane curves in Differential Geometry

Jun-ichi Inoguchi *
Institute of Mathematics
University of Tsukuba

Abstract This is a survey talk on plane curves from differential geometric point of view and applications of plane curves to computer aided designs.

Keywords: transformation group, log-aesthetic curve, similarity geometry, similarity curvature

1 Introduction

Plane curves are fundamental objects in differential geometry. In industrial shape design or computer aided geometric design, curves of particular property have been used as parts of figures. In particular, plane curves with monotonous curvature function have been paid much attention. In this talk, we discuss plane curves with monotonous curvature function via similarity geometry.

2 Euclidean geometry

A plane curve $p(t) = (x(t), y(t))$ is said to be regular if its derivative $\dot{p}(t) = dp/dt$ never vanishes. A regular plane curve can be reparametrised by the arc length parameter $s$. With respect to the arclength parameter, the derivative $p'(s) := dp/ds$ is a unit vector field along the curve. Moreover the vector field $T(s) := p'(s)$ and $N(s) = R(\pi/2)T(s)$ constitutes an orthonormal frame field $\{T(s), N(s)\}$ along the curve. Here $R(\theta)$ denotes the rotation matrix of rotation angle $\theta$. The orthonormal frame field is regarded as an orthogonal matrix valued function $F(s) = (T(s) \text{ } N(s))$. The matrix valued function $F(s)$ takes value in the rotation group SO(2) and called the Frenet frame of the curve. The rotation group SO(2) is a group of all real 2 by 2 orthogonal matrices of determinant 1.

For an arclength parametrised curve $p(s)$, its Frenet frame satisfies the Frenet equation:

$$F'(s) = F(s) \begin{pmatrix} 0 & -\kappa(s) \\ \kappa(s) & 0 \end{pmatrix}.$$ 

The function $\kappa(s)$ is called the curvature of $p(s)$.

It is well known that arclength parametrised curves are determined by the curvature function uniquely up to orientation preserving congruence transformations (Euclidean motions). Note that the set of all Euclidean motions forms a group (called the Euclidean motion group and denoted by SE(2)).

*inoguchi@math.tsukuba.ac.jp
In Euclidean plane geometry, there are some well known attractive curves which are characterised by the curvature function.

**Example 1 (Logarithmic spirals)** A logarithmic spiral is a curve parametrised as
\[
p(t) = a(e^{bt} \cos t, e^{bt} \sin t).
\]
Here \(a\) and \(b\) are positive constants. This logarithmic spiral has curvature \(\kappa(s) = 1/(bs + a\sqrt{1 + b^2})\) with \(s = a\sqrt{1 + b^2}(e^{bt} - 1)/b\). Thus logarithmic spirals are characterised as arclength parametrized curves whose curvature radius \(\rho(s) = 1/\kappa(s)\) is a linear function of the arclength parameter.

**Example 2 (Clothoids)** The clothoid (also called the Cornu spiral) is defined as
\[
p(s) = \left(\int_0^s \cos \frac{ks^2}{2} \, ds, \int_0^s \sin \frac{ks^2}{2} \, ds\right)
\]
by using the Fresnel integral. Here \(k\) is a positive constant. Clothoids have been used in highway design for many years. The clothoid has curvature \(\kappa(s) = ks\).

## 3 Transformation groups

From industrial point of view, we may replace the Euclidean motion group \(SE(2)\) by other transformation groups strictly larger than \(SE(2)\) (according to the purposes or interests).

**Definition 1** Let \(G\) be a Lie group acting transitively on \(\mathbb{R}^2\). The pair \((G, \mathbb{R}^2)\) is called a plane geometry with transformation group \(G\).

The Euclidean plane geometry is formulated as \((SE(2), \mathbb{R}^2)\).

**Example 3 (Affine geometry)** Let us consider the Lie group \(A(2)\) of all affine transformations on \(\mathbb{R}^2\). Affine transformations play fundamental roles in computer graphics. The plane geometry with transformation group \(A(2)\) is called the affine plane geometry. As is well known, every affine transformation on \(\mathbb{R}^2\) has the form:
\[
p \mapsto Ap + b,
\]
where \(A\) is a real nonsingular matrix and \(b\) is a vector. See also [2].

**Example 4 (Equiaffine geometry)** An affine transformation: \(p \mapsto Ap + b\) is said to be equiaffine if \(\det A = 1\). The Lie group of all equiaffine transformations is called the equiaffine transformation group. The plane geometry with equiaffine transformation group is called the equiaffine plane geometry.

Take a regular curve \(p(t)\). Assume that \(p(t)\) is nondegenerate, that is, \(\det(\dot{p}(t) \, \ddot{p}(t)) \neq 0\). Then there exists a parameter \(u\) (called the equiaffine parameter) such that
\[
F^{\text{eqa}}(u) = \begin{pmatrix} \frac{dp}{du}(u) & \frac{d^2p}{du^2}(u) \\ 0 & 1 \end{pmatrix}
\]
has determinant 1. Namely \(F^{\text{eqa}}\) takes value in the special linear group \(SL_2\mathbb{R}\). Here \(SL_2\mathbb{R}\) is the Lie group of all real 2 by 2 matrices with determinant 1. The matrix valued function \(F^{\text{eqa}}\) is called the equiaffine frame and satisfies the equiaffine Frenet equation:
\[
\frac{d}{du} F^{\text{eqa}} = F^{\text{eqa}} \begin{pmatrix} 0 & -\kappa^{\text{eqa}}(u) \\ 1 & 0 \end{pmatrix}.
\]
The function $\kappa^{SA}(u)$ is called the equiaffine curvature of $p(u)$.

Analogous to Euclidean plane geometry, equiaffine parametrised curves are determined uniquely by equiaffine curvature up to orientation preserving equiaffine transformations.

Consider a parabola $p(t) = (t, t^2/2)$. Although parabolas are fundamental examples of plane curves in Euclidean plane geometry, their curvature is not simple. In fact, one can check that $\kappa(t) = (1 + t^2)^{-3/2}$. On the other hand in equiaffine geometry, parabolas are characterised as equiaffine curves with vanishing equiaffine curvature.

### 4 Similarity geometry

In this talk, we concentrate our attention to plane curves under similarity transformations. The plane geometry under similarity transformations is called the similarity plane geometry.

Two circles $C_1$ and $C_2$ are congruent, that is, equivalent in Euclidean plane geometry if and only if they are related by congruent transformations. Thus $C_1$ is congruent to $C_2$ if and only if the radii of two circles coincide. On the other hand in similarity plane geometry, any two circles are equivalent. Any two parabolas are mutually equivalent in similarity plane geometry.

As is well known, logarithmic spiral is a self-similar curve. It is an interesting and natural question to characterise logarithmic spiral in terms of similarity plane geometry.

In similarity geometry, any regular plane curve with non-vanishing Euclidean curvature can be reparametrised by the turning angle function $\theta = \theta(s)$. Note that $d\theta/ds = \kappa(s)$. Analogues to Euclidean plane geometry, $p(\theta)$ admits a matrix valued function $F^S(\theta)$ which takes value in the matrix group $\text{CO}^+(2) = \{rA | r > 0, A \in \text{SO}(2)\}$. The similarity frame $F^S(\theta)$ satisfies the similarity Frenet equation

$$\frac{d}{d\theta} F^S(\theta) = F^S(\theta) \begin{pmatrix} -S(\theta) & -1 \\ 1 & -S(\theta) \end{pmatrix}.$$  

The function $S(\theta)$ is called the Similarity curvature. One can see that $p(\theta)$ has vanishing similarity curvature if and only if $p(\theta)$ is a circle. Regular plane curves with non-vanishing Euclidean curvature with non-zero constant similarity curvature are equivalent to logarithmic spirals.

### 5 Log aesthetic curves

Harada [1] considered logarithmic distribution diagram of curvature (LDDC) for plane curves. They discovered that attractive plane curves drawn by car designers have approximately linear LDDC. The LDDC of an arclength parametrised curve $p(s)$ is

$$(X(s), Y(s)) = (\log \rho(s), \log |ds/d\log \rho(s)|).$$

Miura [3] has given the following mathematical formulation of log-aesthetic curve: A log-aesthetic curve is an arclength parametrised curve $p(s)$ whose curvature radius $\rho(s)$ is given by

$$\rho(s)^\alpha = as + b,$$  

for some constants $a$ and $b$. In case $\alpha = 0$, a log-aesthetic curve is defined by the equation:

$$\rho(s) = \exp(as + b)$$
for some constants $a$ and $b$. Note that the class of log-aesthetic curves contains log-spirals ($\alpha = 1$), clothoids ($\alpha = -1$) and Nielsen spiral ($\alpha = 0$). We notice that the $Y$-coordinate of the LDDC is rewritten as $Y = X - \log |S(\theta)|$. This fact motivates us to study log-aesthetic curves in terms of similarity geometry. We have the following fundamental result.

**Theorem 1 (cf. [4])** Log-aesthetic curves are characterised as plane curves whose reciprocal similarity curvature $1/S(\theta)$ is a linear function of $\theta$.

Moreover one can see that the similarity curvature $S(\theta)$ of log-aesthetic curves satisfies the ordinary differential equation:

$$\frac{dS}{d\theta} = -(1 - \alpha)S^2.$$

Based on this fundamental fact, Sato and Shimizu [4] obtained similarity geometric formulation of certain generalisation of the class of log-aesthetic curves (called the $\rho$-shift generalised log-aesthetic curves). The work [4] motivates us to study attractive curves used in computer graphics or computer aided geometric design in terms of plane geometry under transformation groups *strictly larger* than Euclidean motion groups.

![Figure 1: log-aesthetic curve with $\alpha = -1/4$](image)

**References**


MI レクチャーノートシリーズ刊行にあたり

本レクチャーノートシリーズは、文部科学省 21 世紀 COE プログラム「機能数理学の構築と展開」(H.15-19 年度)において作成した COE Lecture Notes の続刊であり、文部科学省大学院教育改革支援プログラム「産業界が求める数学博士と新修士養成」(H19-21 年度)および、同グローバル COE プログラム「マス・フォア・インダストリ教育研究拠点」(H.20-24 年度)において行われた講義の講義録として出版されてきた。平成 23 年 4 月のマス・フォア・インダストリ研究所(IMI)設立と平成 25 年 4 月の IMI の文部科学省共同利用・共同研究拠点として「産業数学の先進的・基礎的共同研究拠点」の認定を受け、今後、レクチャーノートは、マス・フォア・インダストリに関わる国内外の研究者による講義の講義録、会議録等として出版し、マス・フォア・インダストリの本格的な展開に資するものとする。

平成 26 年 10 月
マス・フォア・インダストリ研究所
所長 福本康秀

Symposium MEIS2015:
Mathematical Progress in Expressive
Image Synthesis

発行 2015年9月18日
編集 落合啓之，土橋宜典
発行 九州大学マス・フォア・インダストリ研究所
九州大学大学院数理学府
〒819-0395 福岡市西区元岡744
九州大学数理・IMI 事務室
TEL 092-802-4402 FAX 092-802-4405
URL http://www.imi.kyushu-u.ac.jp/

印刷 城島印刷株式会社
〒810-0012 福岡市中央区白金 2 丁目 9 番 6 号
TEL 092-531-7102 FAX 092-524-4411
<table>
<thead>
<tr>
<th>Issue</th>
<th>Author/Editor</th>
<th>Title</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE Lecture Note</td>
<td>M.J.Shai HARAN</td>
<td>Arithmetical Investigations - Representation theory, Orthogonal polynomials and Quantum interpolations - 174pages</td>
<td>August 22, 2006</td>
</tr>
<tr>
<td>COE Lecture Note Vol.4</td>
<td>宮田 健治</td>
<td>反要素有限要素法による磁界解析 - 機能数理学特別講義 - 21pages</td>
<td>May 15, 2007</td>
</tr>
<tr>
<td>COE Lecture Note Vol.5</td>
<td>Francois APERY</td>
<td>Univariate Elimination Subresultants - Bezout formula, Laurent series and vanishing conditions - 89pages</td>
<td>September 25, 2007</td>
</tr>
<tr>
<td>COE Lecture Note Vol.7</td>
<td>若山 正人, 中尾 充宏</td>
<td>九州大学産業技術数理研究センター キックオフミーティング - 138pages</td>
<td>October 15, 2007</td>
</tr>
<tr>
<td>COE Lecture Note Vol.10</td>
<td>Jacques FARAUT</td>
<td>Infinite Dimensional Spherical Analysis - 74pages</td>
<td>March 14, 2008</td>
</tr>
<tr>
<td>COE Lecture Note Vol.12</td>
<td>Faculty of Mathematics, Kyushu University</td>
<td>Consortium “MATH for INDUSTRY” First Forum - 87pages</td>
<td>September 16, 2008</td>
</tr>
<tr>
<td>Issue</td>
<td>Author／Editor</td>
<td>Title</td>
<td>Published</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>COE Lecture Note Vol.15</td>
<td>Faculty of Mathematics, Kyushu University</td>
<td>International Workshop on Verified Computations and Related Topics</td>
<td>February 23, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.16</td>
<td>Alexander Samokhin</td>
<td>Volume Integral Equation Method in Problems of Mathematical Physics</td>
<td>February 24, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.17</td>
<td>矢崎 徹, 及川 正行, 椎原 健司, 辻 英一, 福本 康秀</td>
<td>非線形波動の数理と物理 66pages</td>
<td>February 27, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.18</td>
<td>Tim Hoffmann</td>
<td>Discrete Differential Geometry of Curves and Surfaces 75pages</td>
<td>April 21, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.19</td>
<td>Ichiro Suzuki</td>
<td>The Pattern Formation Problem for Autonomous Mobile Robots — Special Lecture in Functional Mathematics — 23pages</td>
<td>April 30, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.20</td>
<td>Yasuhide Fukumoto, Yasunori Maekawa</td>
<td>Math-for-Industry Tutorial; Spectral theories of non-Hermitian operators and their application 184pages</td>
<td>June 19, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.21</td>
<td>Faculty of Mathematics, Kyushu University</td>
<td>Forum “Math-for-Industry” Casimir Force, Casimir Operators and the Riemann Hypothesis 95pages</td>
<td>November 9, 2009</td>
</tr>
<tr>
<td>COE Lecture Note Vol.23</td>
<td>荒川 恒男, 金子 昌信</td>
<td>多重ゼータ値入門 111pages</td>
<td>February 15, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.24</td>
<td>Fulton B.Gonzalez</td>
<td>Notes on Integral Geometry and Harmonic Analysis 125pages</td>
<td>March 12, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.25</td>
<td>Wayne Rossman</td>
<td>Discrete Constant Mean Curvature Surfaces via Conserved Quantities 130pages</td>
<td>May 31, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.26</td>
<td>Mihai Ciucu</td>
<td>Perfect Matchings and Applications 66pages</td>
<td>July 2, 2010</td>
</tr>
<tr>
<td>Issue</td>
<td>Author / Editor</td>
<td>Title</td>
<td>Published</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>COE Lecture Note Vol.27</td>
<td>九州大学大学院 数理学研究院</td>
<td>Forum “Math-for-Industry” and Study Group Workshop Information security, visualization, and inverse problems, on the basis of optimization techniques 100pages</td>
<td>October 21, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.28</td>
<td>ANDREAS LANGER</td>
<td>MODULAR FORMS, ELLIPTIC AND MODULAR CURVES LECTURES AT KYUSHU UNIVERSITY 2010 62pages</td>
<td>November 26, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.29</td>
<td>若山 邦成 原田 昌晃 横山 悠一</td>
<td>Magma で広がる数学の世界 157pages</td>
<td>December 27, 2010</td>
</tr>
<tr>
<td>COE Lecture Note Vol.30</td>
<td>木田 凌 松井 卓 廣島 文生</td>
<td>Mathematical Quantum Field Theory and Renormalization Theory 201pages</td>
<td>January 31, 2011</td>
</tr>
<tr>
<td>COE Lecture Note Vol.31</td>
<td>若山 正人 福本 康秀 高木 剛 山本 昌宏</td>
<td>Study Group Workshop 2010 Lecture &amp; Report 128pages</td>
<td>February 8, 2011</td>
</tr>
<tr>
<td>COE Lecture Note Vol.34</td>
<td>Adrian Muntean Vladimír Chalupecký</td>
<td>Homogenization Method and Multiscale Modeling 72pages</td>
<td>October 28, 2011</td>
</tr>
<tr>
<td>COE Lecture Note Vol.35</td>
<td>横山 俊一 夫 記恵 林 卓也</td>
<td>計算機代数システムの進展 210pages</td>
<td>November 30, 2011</td>
</tr>
<tr>
<td>Issue</td>
<td>Author／Editor</td>
<td>Title</td>
<td>Published</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>COE Lecture Note Vol.40</td>
<td>長谷川顕一, 太田泰広, 勲 三郎, 椎原 魁司, 松浦 望</td>
<td>離散可積分系・離散微分幾何チュートリアル 2012</td>
<td>March 15, 2012</td>
</tr>
<tr>
<td>COE Lecture Note Vol.43</td>
<td>Institute of Mathematics for Industry, Kyushu University</td>
<td>Combinatorics and Numerical Analysis Joint Workshop</td>
<td>December 27, 2012</td>
</tr>
<tr>
<td>COE Lecture Note Vol.44</td>
<td>荻原 学</td>
<td>モダン符号理論からポストモダン符号理論への展望</td>
<td>January 30, 2013</td>
</tr>
<tr>
<td>COE Lecture Note Vol.46</td>
<td>西井 豪映, 脇伸一郎, 岡田啓三, 落合啓之, 小松深幸, 須藤新悟, 白井朝之</td>
<td>科学・技術の研究課題への数学アプローチー 数学モデリングの基礎と展開ー 325pages</td>
<td>February 28, 2013</td>
</tr>
<tr>
<td>COE Lecture Note Vol.47</td>
<td>SOO TECK LEE</td>
<td>BRANCHING RULES AND BRANCHING ALGEBRAS FOR THE COMPLEX CLASSICAL GROUPS 40pages</td>
<td>March 8, 2013</td>
</tr>
<tr>
<td>COE Lecture Note Vol.48</td>
<td>大住 佳寛, 平坂 貢, 谷口哲至, 畑部 修</td>
<td>博多ワークショップ「組み合わせとその応用」 124pages</td>
<td>March 28, 2013</td>
</tr>
<tr>
<td>Issue</td>
<td>Author／Editor</td>
<td>Title</td>
<td>Published</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>COE License Vol.49</td>
<td>照井 諭 小原 功徳 濱田 龍義 横山 俊一 穴井 宏和 橋田 博史</td>
<td>マス・フォア・インダストリ研究所 共同利用研究集会Ⅱ 数式処理研究と産学連携の新たな発展 137pages</td>
<td>August 9, 2013</td>
</tr>
<tr>
<td>MI License Vol.53</td>
<td>四方 義啓 櫻井 哲一 安田 貴徳 Xavier Dahan</td>
<td>平成25年度 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 安全・安心社会基盤構築のための数理科学 ～サイバー社会の信頼性確保のための数理学～ 158pages</td>
<td>December 26, 2013</td>
</tr>
<tr>
<td>MI License Vol.54</td>
<td>Takashi Takiguchi Hiroshi Fujiwara</td>
<td>Inverse problems for practice, the present and the future 93pages</td>
<td>January 30, 2014</td>
</tr>
</tbody>
</table>
シリーズ既刊

<table>
<thead>
<tr>
<th>Issue</th>
<th>Author／Editor</th>
<th>Title</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI Lecture Note Vol.59</td>
<td>西井 龍映</td>
<td>Study Group Workshop 2014 数学協働プログラム</td>
<td>November 14, 2014</td>
</tr>
<tr>
<td></td>
<td>岡田 勘三</td>
<td>Abstract, Lecture &amp; Report</td>
<td></td>
</tr>
<tr>
<td></td>
<td>梶原 健司</td>
<td>196pages</td>
<td></td>
</tr>
<tr>
<td></td>
<td>髙木 剛</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>若山 正人</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>山本 昌宏</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MI Lecture Note Vol.60</td>
<td>西浦 博</td>
<td>平成 26 年度九州大学 IMI 共同利用研究・研究集会 (I)</td>
<td>November 28, 2014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>感染症数理モデルの実用化と産業及び政策での活用のための新たな展開</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>120pages</td>
<td></td>
</tr>
<tr>
<td>MI Lecture Note Vol.61</td>
<td>溝口 佳寛</td>
<td>研究集会</td>
<td>February 26, 2015</td>
</tr>
<tr>
<td></td>
<td>Jacques Garrigue</td>
<td>高信頼な理論と実装のための定理証明および定理証明器</td>
<td></td>
</tr>
<tr>
<td></td>
<td>萩原 学</td>
<td>Theorem proving and provers for reliable theory and implementations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reynald Affeldt</td>
<td>(TPP2014) 138pages</td>
<td></td>
</tr>
<tr>
<td>MI Lecture Note Vol.62</td>
<td>白井 朋之</td>
<td>Workshop on “β-transformation and related topics”</td>
<td>March 10, 2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59pages</td>
<td></td>
</tr>
<tr>
<td>MI Lecture Note Vol.63</td>
<td>白井 朋之</td>
<td>Workshop on “Probabilistic models with determinantal structure”</td>
<td>August 20, 2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>107pages</td>
<td></td>
</tr>
</tbody>
</table>
Workshop on "B-transformation and related topics"

Editor: Tomoyuki Shirai