# Exploratory Design of Mechanical Devices with Motion Constraints

Author version*

Robin Roussel
University College London
London, United Kingdom
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK
Grenoble, France
robin.roussel.15@ucl.ac.uk

Marie-Paule Cani
LIX, École Polytechnique
Palaiseau, France
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK
Grenoble, France

Jean-Claude Léon
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LJK
Grenoble, France

Niloy J. Mitra
University College London
London, United Kingdom

## ABSTRACT

Mechanical devices are ubiquitous in our daily lives, and the motion they are able to transmit is often a critical part of their function. While digital fabrication devices facilitate their realization, motion-driven mechanism design remains a challenging task. We take drawing machines as a case study in exploratory design. Devices such as the Spirograph can generate intricate patterns from an assembly of simple mechanical elements. Trying to control and customize these patterns, however, is particularly hard, especially when the number of parts increases. We propose a novel constrained exploration method that enables a user to easily explore feasible drawings by directly indicating pattern preferences at different levels of control. The user starts by selecting a target pattern with the help of construction lines and rough sketching, and then fine-tunes it by prescribing geometric features of interest directly on the drawing. The designed pattern can then be directly realized with an easy-to-fabricate drawing machine. The key technical challenge is to facilitate the exploration of the high dimensional configuration space of such fabricable machines. To this end, we propose a novel method that dynamically reparameterizes the local configuration space and allows the user to move continuously between pattern variations, while preserving user-specified feature constraints.

We tested our framework on several examples, conducted a user study, and fabricated a sample of the designed examples.

## KEYWORDS

computational design, design space, constrained exploration, drawing machines, mechanical motion, fabrication

## 1 INTRODUCTION

Toy of the Year in 1967, the Spirograph is a simple-to-use family of interlocking cogs and teethed rings allowing to draw a great variety of patterns. Although many other mechanical drawing tools preceded and followed it (see Fig. 1), this modest set of gears has marked a generation, and remains one of the most well-remembered today. As a product of the relationship between art and technology, drawing machines are still popular across artists [15], enthusiastic inventors [13], and makers [34]. The simplicity of the mechanical parts involved makes them easily fabricable with modern personal fabrication devices, which in turn open the door to a level of customization leading to new and fascinating patterns. Beyond this goal, the inverse problem of finding the machine tracing out a specific trajectory has numerous applications (see e.g. Coros et al. [8]).

Designing such machines, however, is particularly challenging. First, many mechanical devices transform an input rotation into a more complex cyclic output by combining oscillations of different periods and amplitudes. To produce a closed end-effector curve, the radii of mating gears (or equivalently, the number of teeth) need to have rational ratios. It is easy to enforce this constraint by restricting radii to natural numbers; the size of the pattern can still be controlled by a global scaling factor. The downside is that the design space becomes much more complex to explore: as the period is governed by modular arithmetic between radii, the visual output can radically change from one value to the next. Furthermore, the number of design parameters obviously increases with the number of parts. While this greatly enriches the space of possible curves, manually refining a design becomes difficult with as little as three continuous parameters. Indeed, nonlinearities make the influence of each control hard to grasp, and each one possibly influences the bounds of the others, making the space harder to explore.

In this paper, we propose a constraint-based exploration framework to design complex mechanical trajectories by interacting directly with the output pattern. In contrast to previous work [1], we focus on: (i) highly structured curves, which would be tedious to edit point by point, and (ii) allowing the continuous exploration of local design variations, rather than recomputing a new solution after each curve edit. Indeed, the latter has the disadvantage that modifications made in one place of the pattern may result in unexpected changes somewhere else. Our method, on the other hand, allows the user to define visual preferences and explore the resulting constrained subspace.

Our exploration workflow consists in a coarse-to-fine definition of visual preferences that progressively refine the choice of curves. First, as an entry point into the design space, the user draws a coarse

*For citations, please refer to the published version at https://doi.org/10.1016/j.cag.2018.05.023

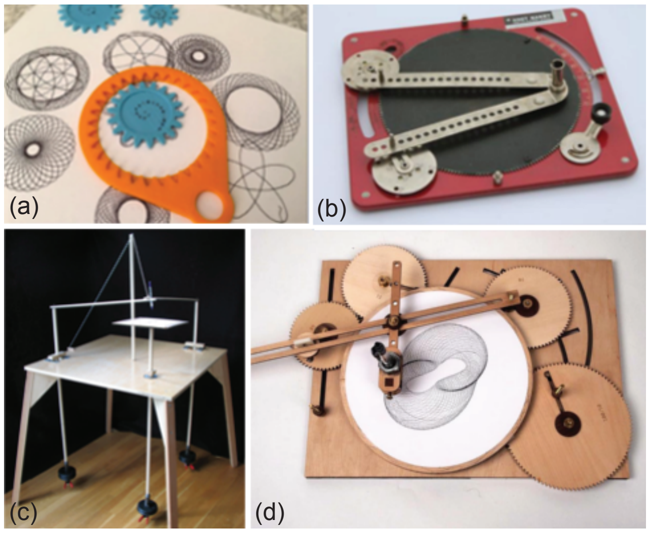

**Figure 1: Some examples of drawing machines: (a) Spirograph, (b) Hoot-Nanny, (c) Harmonograph, (d) Cycloid drawing machine.**

sketch that defines the global properties (e.g., order of rotational symmetry) and appearance of the desired pattern. After selecting an initial curve among suggestions proposed by the system, changes can be made via sliders within a domain that respects the feasibility constraints of the corresponding mechanism. When one slider is moved, the bounds of the others are automatically updated. As a key interaction, the user can define *visual preferences* directly on the drawing. These take the form of special points on the curve that can be constrained according to their geometric properties. The user can then explore local variations closest to these specifications via new handles that are automatically generated. Once the user is satisfied, the shape of the mechanical parts is automatically generated and exported for laser cutting fabrication (see Fig. 2).

Technically, we enable the above key interaction with a novel dynamic reparameterization method that locally samples the high dimensional configuration space of a given mechanism, measures closeness to the user-defined preferences, approximates the closest subspace, and exposes new parameters to navigate this subspace.

We evaluated the effectiveness of our design tool on several test scenarios, conducted a user study, and fabricated several physical prototypes able to draw patterns created by the users. Overall, we found that dynamic reparameterization allowed users to reliably make meaningful fine scale adjustments to their pattern designs.

This paper extends the previous conference work [28] by significantly extending the pattern retrieval step (including a novel technique to reduce the search space), providing new figures, a more detailed explanation of the fabrication process, and a virtual extension to a mechanical character to demonstrate the generality of our method.

## 2 RELATED WORK

Drawing machines have a long history in mathematics [19], art [12], and as toys. Before the computer era, they were the only way to accurately draw certain curves, with applications in architecture, astronomy, engineering, etc. [16]. While simulating such machines is nowadays relatively easy, the inverse problem of mapping an arbitrary end-effector trajectory to a reasonably simple mechanism remains a challenge. One of the most fundamental results in this regard is Kempe's universality theorem [21], which states that for any arbitrary algebraic plane curve, a linkage can be constructed that draws the curve. The constructive method proposed by Kempe, however, produces mechanisms with so many links that they are impossible to fabricate in practice. There have been many endeavors since then; to cite only one, Liu et al. [24] recently proposed a method to reproduce trigonometric plane curves with either Scotch yoke mechanisms or serial chains. Robotic arms and CNC machines have not made this problem obsolete, since there are situations where size is an issue (e.g. MEMS or nano applications), or electric power is unavailable or undesirable.

The following paragraphs discuss related advances in computational design and fabrication, with different application settings, both for inverse modeling and design exploration.

*Computational design from target motion.* In the context of automata design, researchers have investigated replicating target motion using an arrangement of mechanical parts in a classic instance of inverse problem setup. The general approach involves sampling configurations from a library of components to retrieve a local arrangement of parts, and then refining the shape parameters using a gradient-descent based optimization to fit the target motion. For example, Zhou et al. [39] and Coros et al. [8] design automaton characters, while Ceylan et al. [7] design automata to replicate mo-cap sequences.

In more interactive design settings, Umetani et al. [36] design paper airplanes based on their predicted flight dynamics; Thomaszewski et al. [33] use global optimization to design linkage-based characters; while Bächer et al. [1] develop a system to support interactive editing of fabricable linkages. Recently, Ion et al. [20] took a different approach by generating 3D-printable microstructures that are able to transmit movement through shearing of their constitutive cells.

The main goal of the above efforts is to either approximate a given motion, or indirectly edit a target automata or linkage-based kinematic chain. Formulating these problems as optimizations suggests that the user already knows what she wants, and simply wishes to obtain a specific motion as easily as possible. While some parts of our method appear similar (e.g. the rough pattern retrieval step), what we propose is a framework to support *exploration* and *progressive refinement*, which had not yet been achieved in this context. More formally, instead of trying to map a specific curve to a *point* in design space, our method maps a set of curve constraints to a design *subspace*. These approaches complete rather than oppose each other: even in the first case, there is interest in being able to explore local design alternatives, as it might reveal solutions that the user had not thought of.

*Guided exploration of valid designs.* In a broader context of design exploration, researchers have studied various properties in order to optimize a shape based on its intended usage. Examples include shape optimization based on stability considerations [27], or updating object shape for desired moment of inertia for object

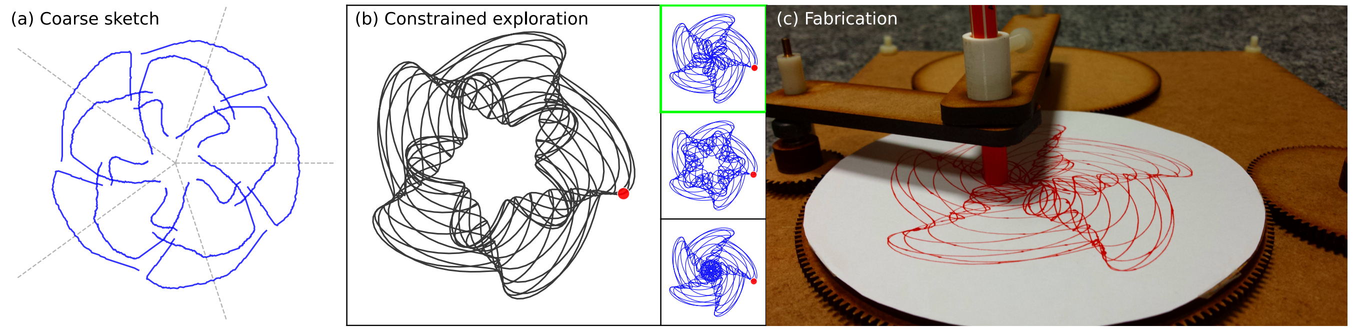

**Figure 2: Overview of our design workflow. The user first selects a mechanically feasible drawing by providing a rough sketch (a), and is then able to interactively explore local alternatives (b) by defining visual constraints directly on the pattern (here, the cusp position). The resulting machine is automatically exported to laser cutter profiles for fabrication. (c)**

spin [2], adaptively adjusting object parts for better reinforcement and strength [25, 38], designing hollow chambers for desired acoustic behavior [5], zero-waste furniture design [22], or modeling elastic behavior of foam microstructures for procedurally generating them for target material properties [26].

Closer to our concerns, Umetani et al. [35] proposed a furniture modeling system that actively guides the user to navigate valid regions of the design space; Bokeloh et al. [6] and Yumer et al. [37] developed modeling systems that preserve high-level structural and semantic relations in edited 3D models, while Koo et al. [23] proposed the use of functional specifications to map user prescriptions to constrained modeling for 'works-like' prototypes of furniture. We have been inspired by the work of Shugrina et al. [30], who precompute the domain defined by fabricability and functionality constraints to expose sliders with valid ranges to the user. In our work, however, additional constraints are defined by the user at runtime directly on the drawing, and new sliders are automatically generated to explore the resulting constrained space. Recently, Guerrero et al. [17] proposed efficient local approximations to enable exploration of pattern variations by dropping different constraints in the input patterns. However, the method is not suitable for variations that are additionally required to be physically realizable by drawing machines. As in these works, we aim to ease the exploration of the feasible space, but apply this to the problem of exploring the curve patterns produced by a mechanism, instead of exploring designs of the mechanism itself.

Additionally, in a classical constrained modeling setup, design from geometric constraints specifications has also been studied. Ivan Sutherland's Sketchpad [32] pioneered CAD software as the first graphical interface to support geometric constraints. Analyzing and solving such constraints have been tackled by Fudos et al. [14] in a general setting, and Sitharam et al. [31] in the context of mechanisms. Theoretically, even determining parameter bounds characterizing the (constrained) solution domain remains a known difficult topic [4]. Instead, in this work, we seek a general approximation strategy supporting interactive design space exploration for an intuitive workflow.

## 3 OVERVIEW

Mechanical drawing machines typically are arrangements of cogs and parts, with an end-effector that traces out intricate 2D patterns. Each such machine physically realizes an algebraic expression connecting the machine part parameters to the output drawing. This tight coupling between the parameters and the resultant pattern variations makes the designers' task of exploring the design space very challenging. Specifically, while on one hand modifying a single parameter may cause several simultaneous changes (e.g., twisting and scaling), on the other hand a single desired change often requires synchronous manipulation of multiple parameters. Our goal is to decorrelate these variations. Rather than trying to find the best possible separation (which tends to be subjective or context-dependent), our goal is to allow users to define their own visual constraints or invariants in the drawing space, so that other variations can be explored independently.

There are two main technical challenges to tackle: first, mechanisms are often described by a relatively high number of parameters (3-8 in our examples), both continuous and discrete, and whose valid domain is implicitly defined by a set of non-linear constraints; second, mapping invariants in the drawing to a corresponding parameter subspace cannot be done analytically in the general case, as the relation between parameter changes and drawing changes is very complex.

We address these challenges with a two-step workflow (see Fig. 2). The first step, described in Sec. 4, consists in selecting an appropriate machine by defining global pattern characteristics and providing a coarse sketch. This step notably allows to assign and fix all discrete parameters. During the second step, described in Sec. 5, local continuous variations can be explored while dynamically specifying visual invariants. Our key contribution is twofold: identify a set of recurring geometric regularities involving relevant feature points that can be tracked as the drawing changes (Sec. 5.1), and a novel local approximation method that allows to explore the subspace where such regularities appear (Sec. 5.2).

We evaluate our method in several ways (see Sec. 6). First, we demonstrate a number of cases where our invariants allow meaningful changes in the drawing (Sec. 6.1). Second, we validated the feasibility of our drawing machines by fabricating several prototypes (Sec. 6.2). Lastly, we conducted a user study to assess the

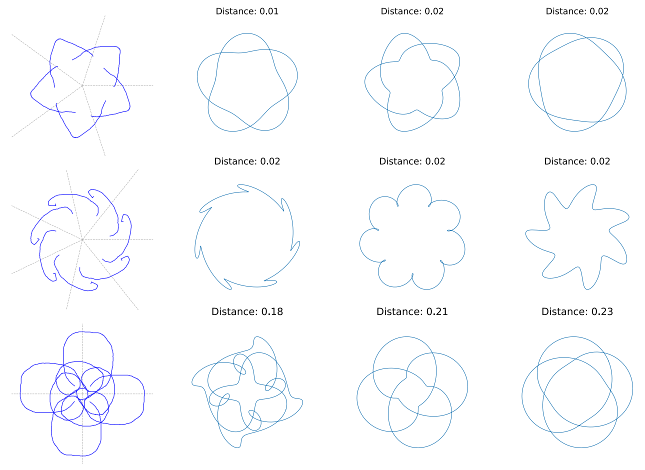

**Figure 3: Starting from a user sketch (left), we retrieve the best matching patterns in the database – lower value indicates a better candidate.**

ability of invariant-based parameterization to efficiently help navigating the configuration space (Sec 6.3).

Let us now define the notation used in the rest of the paper. Each type of machine is described by:

- A set of design parameters $\mathbf{p} = (\mathbf{p}^d, \mathbf{p}^c)$ (the concatenation of discrete and continuous parameters), evolving in a design space $D$ implicitly bounded by a system of algebraic constraints $\mathbf{C}(\mathbf{p})$ (typically nonlinear).
- A simulator that works out the trajectory $\mathcal{D}_\mathbf{p}(t),\ t \in [0, \mathcal{T}]$ (where $\mathcal{T}$ is the period) traced by the end point. It also provides a time series of the positions and orientations of each machine part for visual inspection.
- A representation $R_\mathbf{p}$ of the mechanism geometry at different levels of detail (coarse for visual checking, detailed for export and fabrication).

In this paper, 'configuration space' or 'parameter space' refers to $D$, while 'curve space' or 'drawing space' denotes the Euclidean 2D space of the trajectory.

## 4 PATTERN RETRIEVAL

The first step of the design workflow is an inverse problem: finding the parameter combination solution of

$$\min_{\mathbf{p} \in D'} d(\mathcal{D}_\mathbf{p}, \mathcal{S}) \tag{1}$$

where $\mathcal{S}$ is a spline fitted to the user's sketch (Section 4.1), $D'$ is a subspace of $D$ computed from the features of $\mathcal{S}$ (Section 4.2), and $d$ is a measure of dissimilarity between curves (Section 4.3). Since the patterns produced by drawing machines are most often abstract, intricate, and generally tedious to sketch precisely, the result may only coarsely match the user's intent. Our goal in this step is to find a value for the discrete parameters (i.e. the radii), so that the remaining continuous space can be explored.

Once $\mathcal{S}$ and $D'$ are computed, a set of $N_b$ best matching candidates $\mathcal{D}_\mathbf{p}^i$ is retrieved via brute force comparison of $\mathcal{S}$ to drawings sampled in $D'$. Redundant drawings are avoided (to a certain extent) by making sure that the radii are coprimes (i.e. they do not share a divisor other that 1). This is enforced by sampling values from the Farey sequence (without the first term), which can be computed with linear complexity in the number of terms [29]. Continuous parameters, on the other hand, are naively sampled (within feasible bounds) using grid search. The $N_b$ best matches are presented to the user, who can thus choose the best $\mathbf{p}$ (see Fig. 3).

### 4.1 Sketching and spline fitting

The user first draws a rough sketch. Construction lines can be used to pre-set the order of rotational symmetry: the pen strokes are then automatically symmetrized (see Fig. 4).

Our input is therefore a sequence of $N_s$ disconnected, noisy strokes (represented as polylines) that we want to turn into a smooth closed spline. Obviously, if $N_s = 1$ we can fit the spline directly. Otherwise, our goal is to obtain the shortest closed path that runs through all strokes. This amounts to a variant of the traveling salesman problem (TSP), where part of the path is already fixed. We start with the integer program formulation of Dantzig et al. [9], and make the following modification. Let the $2N_s$ ends of each stroke be the vertices $V$ of a complete undirected graph. The set of edges $E$ in this graph corresponds to the connections between each end. In order to force the connection between two ends of the same stroke, we define the coefficient of each edge as

$$c_{ij} = \begin{cases} -k & \text{if } i \text{ and } j \text{ belong to the same stroke,} \\ d_{ij} & \text{otherwise,} \end{cases} \tag{2}$$

where $d_{ij}$ is the Euclidean distance between the vertices $i$ and $j$, and k is a positive constant. For $k$ high enough, it is always more advantageous for the solver to connect the two ends of a given stroke together, while still having the freedom of choosing the connection order. We use the Gurobi solver [18] for this optimization, which relies on a linear-programming based branch-and-bound algorithm. We obtain an ordering of the strokes as well as the points within these strokes (given by the order of the ends). Lastly, we concatenate the strokes and fit a closed cubic B-spline $\mathcal{S}$ with a user-defined smoothing factor.

We note that while the TSP is NP-hard, in practice the number of strokes tends to be small; therefore, in all of our examples, the processing time of the entire step was around 0.01s.

### 4.2 Reducing the search space

To prevent the grid search from becoming prohibitively expensive, we prune the search space with a number of empirically determined rules. First of all, the degree or rotational symmetry, if there is such a symmetry, is always equal to the reduced radius $r_1$ of the leading gear $G_1$ ('reduced' meaning that all radii have been divided to become coprimes). Second, the frequency spectrum of the pattern is a useful source of information regarding the other radii. Indeed, each pattern is a result of the combination of periodic movements produced by the gears. Our strategy starts by computing the discrete Fourier transform (DFT) of a target curve to discover the dominant frequencies, before translating them in terms of gear radii.

First, we sample $N_f$ equally spaced parameter values in $[0, 1]$ and compute the corresponding points along the target spline $\mathcal{S}$. For convenience, we write these 2D points as complex numbers

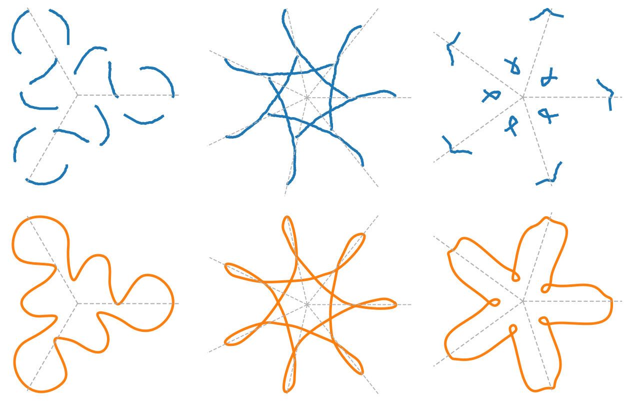
**Figure 4: Examples of splines fitted to input sketches. Each sketch required only 1-4 strokes that were automatically symmetrized.**

$x_n \in \mathbb{C}$, and compute the DFT $\mathbf{X} = \mathcal{F}\{\mathbf{x}\}$ with

$$X_k = \sum_{n=0}^{N_f-1} x_n \cdot e^{-\frac{2\pi i}{N_f}kn} \qquad \forall k \in [0\,..\,N_f-1]. \tag{3}$$

The DFT is, by definition, a sampling of the discrete-time Fourier transform (DTFT), which is itself a continuous function of frequency. Each value $X_k$ of the DFT corresponds to a frequency $f_k$ given by

$$f_k = \frac{f_s}{N_f}k \qquad \forall k \in \left[-\frac{N_f}{2}\,..\,\frac{N_f}{2}-1\right], \tag{4}$$

where $f_s$ is the signal sampling frequency. The different interval for $k$ compared to Eq. 3 is not a problem since $\mathbf{X}$ is periodic over this range. Moreover, by construction, our input curve spans exactly one period $\mathcal{T}$. Therefore, $f_s = N_f/\mathcal{T}$ and Eq. 4 becomes

$$f_k = \frac{k}{\mathcal{T}} \qquad \forall k \in \left[-\frac{N_f}{2}\,..\,\frac{N_f}{2}-1\right]. \tag{5}$$

Our goal is to find the equation of motion closest to the input curve; therefore, $\mathcal{T}$ is a priori unknown. As we will see, however, we do not need to determine its actual value.

The middle graph in Fig. 5 illustrates the $(f_k, |X_k|)$ mapping, also called frequency spectrum, for a given curve (where we arbitrarily set $\mathcal{T} = 1$). As we can see, for such a clean curve only a few frequencies dominate, symmetrically distributed around the fundamental frequency $\overline{f}_1$. Two of the simplest machines (Spirograph and Cycloid Drawing Machine) present a second dominant frequency $\overline{f}_2$, while the Hoot-Nanny even displays a third dominant frequency $\overline{f}_3$ (both $\overline{f}_2$ and $\overline{f}_3$ being accompanied by their harmonics). We postulate that the number of such frequencies increases by one for each new pair of mating gears in the system (unless of course they cancel each other out). In practice, however, $\overline{f}_3$ may be a multiple of $\overline{f}_2$, in which case it is indistinguishable from $\overline{f}_2$'s harmonics; this is a case where our analysis would only partially reduce the search space.

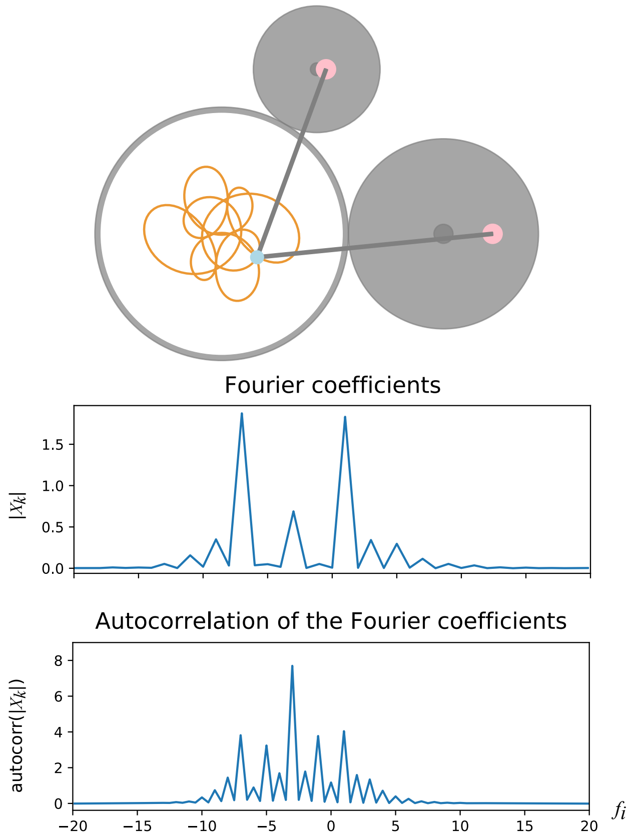

**Figure 5: Fourier analysis of a Hoot-Nanny drawing. The radii are $(4, 3, 2)$, and the highest peaks of Fourier spectrum happen at frequencies $(-3, 1, 3)$. One can easily check the validity of Eq. 6.**

To compute the radii of the $N_G$ gears from the frequencies $\overline{f}_i$, we have experimentally found that the radius of gear $G_i$ satisfied

$$r_i = \frac{\mathrm{lcm}(n_1, \ldots, n_{N_G})}{n_i} \qquad \text{with} \quad n_i = \begin{cases} \mathcal{T}|\overline{f}_i| & \text{if } i = 1, \\ \mathcal{T}|\overline{f}_i - \overline{f}_1| & \forall i \in [2\,..\,N_G], \end{cases} \tag{6}$$

where lcm() is the least common multiple, and $n_i$ is the number of rotations of $G_i$ over one period. From the frequency formula (Eq. 5), it is clear that $\mathcal{T}$ cancels out; therefore, we can simply set $\mathcal{T} = 1$ from the start, thus obtaining integer frequencies.

Eq. 6 can be verified in the case of the epitrochoid (i.e. the curve produced by a Spirograph with one gear rolling outside the other), whose Cartesian equation of motion is

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = r_R(q+1)\begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix} - d\begin{bmatrix} \cos((q+1)\,t) \\ \sin((q+1)\,t) \end{bmatrix} \qquad t \in [0, \mathcal{T}] \tag{7}$$

with $q = \frac{r_F}{r_R}$, where $r_F$ and $r_R$ respectively denote the radii of the fixed and rolling gears, and $d$ is the distance from the pen hole to the center of the gear. Since Eq. 7 is conveniently written as a

Fourier series, and $q + 1 > 1$ whatever the radii, it is clear that

$$\overline{f}_1 = \frac{1}{2\pi} \quad \text{and} \quad \overline{f}_2 = \frac{q+1}{2\pi}. \tag{8}$$

Taking the ratio,

$$\begin{aligned} \frac{\overline{f}_1}{\overline{f}_2} = \frac{1}{q+1} &\iff q = \frac{\overline{f}_2 - \overline{f}_1}{\overline{f}_1} \\ &\iff r_F \overline{f}_1 = r_R(\overline{f}_2 - \overline{f}_1). \end{aligned} \tag{9}$$

Since $r_F$ and $r_R$ are supposed coprimes,

$$r_F \overline{f}_1 = r_R(\overline{f}_2 - \overline{f}_1) = \mathrm{lcm}(\overline{f}_1, \overline{f}_2 - \overline{f}_1), \tag{10}$$

which is indeed a specific case of Eq. 6. The proof for other machines involves much more complex equations of motion, but we can intuitively understand why it still works: different linkages only affect the amplitude of the oscillations, and not their frequencies; the latter only depend on the radii of mating gears. Therefore, for instance, the frequency spectrum of a Hoot-Nanny can be seen as a sum of epitrochoids.

The challenge, then, is to determine the $\overline{f}_i$ from the spectrum of $\mathcal{S}$. First, we can safely set the $f_0$ peak (constant component) to 0, effectively centering the curve. Second, while $\overline{f}_1$ is the highest peak for the (Elliptic) Spirograph, this is not necessarily true for other machines (see Fig. 5). The spectrum, however, always tends to be symmetric; therefore, we perform an autocorrelation of the Fourier peaks (i.e. a discrete convolution of the peaks with themselves) to make the fundamental detection more robust. If the machine has only two gears, determining $\overline{f}_2$ is easy: it is the next maximal peak. The case of the Hoot-Nanny is more complex, as we saw that its frequency spectrum appears to be the sum of its two gear matings spectra considered separately. This has two consequences. First, $\overline{f}_2$ and $\overline{f}_3$ can only be distinguished if they are coprimes (which happens when the corresponding radii are also coprimes); otherwise their harmonics overlap. While all three radii are constrained to share no common divisor, any pair of them taken separately still can, which is why $\overline{f}_2$ and $\overline{f}_3$ are not necessarily coprimes. Second, their order in the spectrum remains ambiguous no matter what: the next highest peak after $\overline{f}_1$ can correspond to $r_2$ or $r_3$, while the Hoot-Nanny curves are sensitive to the order of $r_2$ and $r_3$.

From a practical point of view however, the goal of this step is only to reduce the search space, not to completely determine the radii. If the order of $r_2$ and $r_3$ is uncertain, we test both combinations; if the amplitude of a peak is too low, we leave the corresponding radius undetermined and sample it along the continuous parameters during grid search. Any pruning of the search space is good to take, since the complexity is combinatorial in the number of parameters. Using a sampling density of 4 values per dimension for all machines in our dataset, our experiments showed that curve retrieval was faster by an order of magnitude when at least one parameter value was fixed.

## 4.3 Curve dissimilarity measure

Let us consider two curves $C_A$ and $C_B$ represented as polylines with $N_A$ and $N_B$ vertices respectively. In the previous version of this method [28], curves were compared by normalizing and aligning

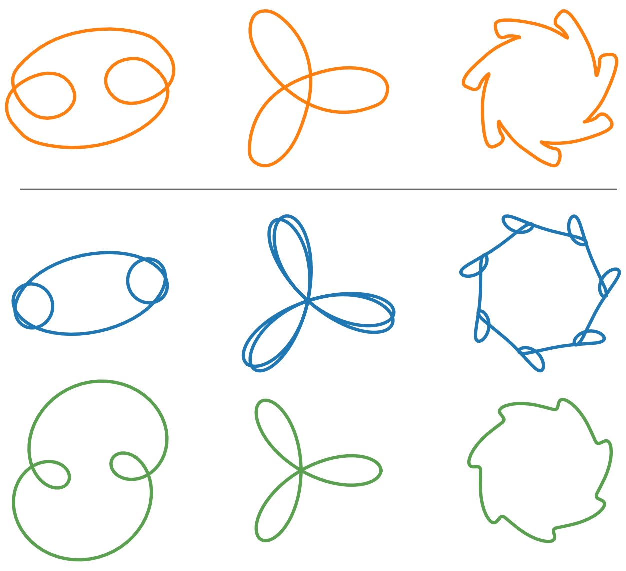

**Figure 6: Comparison of best matching curves for a given input spline (top row), respectively obtained with the distance field metric (middle) and the Procrustes distance (bottom).**

them, then computing their respective distance fields $F_A$ and $F_B$ defined as

$$\forall \mathbf{x} \in \mathbb{R}^2 \quad F_*(\mathbf{x}) = \inf_{\mathbf{x}' \in C_*} \|\mathbf{x}' - \mathbf{x}\|, \tag{11}$$

and finally computing the symmetric distance

$$d_F(C_A, C_B) = \max \left\{ \frac{1}{N_A} \sum_{i=1}^{N_A} F_B(\mathbf{x}_i^A), \frac{1}{N_B} \sum_{i=1}^{N_B} F_A(\mathbf{x}_i^B) \right\}. \tag{12}$$

In words, each side of the max tells us, on average, how close a point from one curve is to the other curve. Intuitively, $d_F$ quantifies how curves differ in terms of the 'density' of strokes in a region of space. From a practical point of view, this metric accepts a wide range of inputs, as each curve could be a collection of polylines, or even a binary image obtained from a picture. On the other hand, this metric is not as precise as e.g. Procrustes analysis [10] or the discrete Fréchet distance [11], two classic curve dissimilarity measures. The former requires $N_A = N_B$, but inherently normalizes and registers the curves with linear complexity in $N_A$ and $N_B$, while the latter has (sub-)quadratic complexity. Having the same number of vertices is not difficult in our new method, since we can resample $\mathcal{S}$ to match the size of $\mathcal{D}_\mathbf{p}$. Therefore, we adopt a symmetrized Procrustes distance as our new measure, defined as

$$d_P(C_A, C_B) = \min \left\{ \sum_{i=1}^{N} \|\mathbf{x}_i^A - \mathbf{x}_i^B\|^2, \sum_{i=1}^{N} \|\mathbf{x}_i^A - \mathbf{x}_{N-i+1}^B\|^2 \right\}, \tag{13}$$

where $N = N_A = N_B$. Here, one curve is compared to the other with its vertices taken in increasing, then decreasing order. This is because the regular Procrustes distance depends on the order of the sampling, and we do not know if $\mathcal{S}$ and $\mathcal{D}_\mathbf{p}$ are drawn in the same direction.

**Table 1: Sampling and computation times for the three examples of Fig.6. There were 12248 samples before pruning.**

| Input drawing | # samples | Sampling time (s) | Time per sample (s) Distance field | Time per sample (s) Procrustes distance |
|---|---|---|---|---|
| Loops | 376 | 0.85 | 0.0100 | 0.0006 |
| Trefoil | 360 | 1.04 | 0.0154 | 0.0007 |
| Whirl | 600 | 0.94 | 0.0225 | 0.0008 |
| | | **Average** | **0.0159** | **0.0007** |

Three comparisons between the distance field metric and the Procrustes distance are given Fig. 6. Both metrics were used to explore the same dataset of candidate curves and retrieve the best matching one. We observe that while the distance field metric is able to capture the general aspect of the curve, it is insensitive to other aspects such as the arc length because of its image based nature. Thus, for example, the trefoil knot in the middle is evaluated as similar despite looping twice. Furthermore, our timing measurements (see Table 1) show that the Procrustes distance between curves at a resolution of 1024 points is more than 20 times faster than the distance field metric between curves rasterized at a resolution of 512 by 512 pixels.

## 5 CONSTRAINED EXPLORATION

Once the the discrete parameters of the machine are fixed, the user can focus on fine tuning the continuous parameters. We note that an intuitive system should allow the user to edit different features of the drawing as independently as needed. This is not always possible: the smaller the number of degrees of freedom, the harder it is to prevent several changes from happening at the same time. For instance, a drawing machine with a single continuous parameter would not benefit from our system. Conversely, as the number of parameters increases, so does the extent to which modifications can be decorrelated. However, the exact combination of parameters that allows a constrained change is generally complex to determine, as it requires to either solve a system of non-linear equations, or to resort to manual trial-and-error. Hence, our goal is to efficiently identify, abstract, and expose the space of valid machine configurations subject to the specified constraints.

We allow the user to specify *visual preferences* as geometric properties that should stay fixed when a change is made. Note that this is different from handle-based deformation as the user indicates what *shouldn't change* during editing, rather than a specific target change. Then, our system computes a new parameter space that incorporates the previous machine-specific and global constraints with the new shape invariant(s). The resulting space can be explored via sliders, whose bounds are dynamically updated after each modification. The user can subsequently add more invariants, which further constrain the solution space until no remaining degree of freedom is left.

We first introduce the shape invariants that are supported and how they are dynamically computed and tracked (Section 5.1). Then, we propose a local reparameterization method that enables the user to intuitively explore the resulting invariant space in the form of desirable pattern variations (Section 5.2).

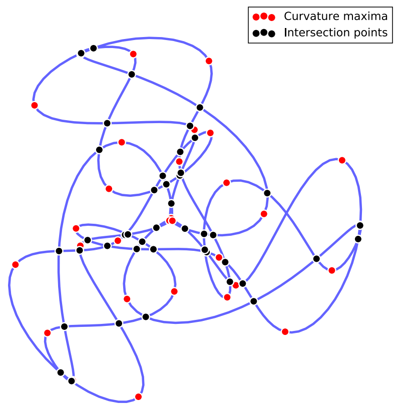

**Figure 7: Example of Points of Interest in a drawing.**

### 5.1 Pattern invariants

The curves generated by drawing machines are often highly structured and can be described at several levels of detail. If we attempt to decompose such a shape, the smallest discernible element is the point. However, not all points are perceptually equal: some have particular properties that make them stand out, such as intersection points and curvature maxima (see Fig. 7). We call them *Points of Interest* (PoI). Such points have generic attributes, such as Euclidean coordinates in curve space, and one (or two) associated time (or arclength) values. They also display properties which are specific to their type, such as the angle made by curve tangents at an intersection point, or the value of the curvature at the maximum (see Fig. 8).

Next, we define *Relations of Interest* (RoI), as relations that hold either between a PoI and an external object (e.g., a PoI lying on a geometric primitive), or between a group of PoIs (e.g., the distance between two PoIs). Any relation that can be expressed as an algebraic equation involving one or more features of one or more PoIs can be implemented in the system. While higher-level entities could also be considered, such as edges between PoIs, cells formed by edges, or even envelopes formed by sequences of PoIs, we currently only support PoIs and RoIs, as they are easier to compute and track in the parameter space. We note that the computation of the invariant subspace is agnostic of the nature of the features defined by the user.

*Selection and computation.* During the interactive session, the PoIs closest to the user's mouse are highlighted. Selecting one (or two) of them opens a menu allowing the user to choose a feature to freeze.

For generality, we compute the PoIs on a discretized curve output by the simulation, instead of solving for them analytically. Curvature maxima are straightforward to obtain, as discrete curvature on polyline vertices is easy to compute. Finding the self-intersections of a polygon, however, is more involved, as the naive algorithm (testing every pair of segments) has a complexity that is quadratic in the number of sides. This problem has been extensively studied and several methods (essentially sweep-line based) have been proposed. We use the Bentley-Ottmann algorithm, whose complexity is $O((n+k)\log(n))$, with $n$ line segments and $k$ crossings.

*Tracking.* Key to our approach, PoIs must be *tracked* as continuous parameter values change: in other words, when considering two patterns relatively close in the parameter space, we need to establish correspondences across the PoIs allowing us to quantify how much a specific PoI property has changed between two curves, and therefore, to build an invariant space.

Given two drawings $\mathcal{D}$ and $\mathcal{D}'$ and a reference PoI $\boldsymbol{\pi}_r^{\mathcal{D}}$ on $\mathcal{D}$, a naive criterion for such correspondence is to superimpose both drawings and take the closest PoI in $\mathcal{D}'$. In some configurations however, several PoIs can overlap each other, leading to ambiguities. This search can be made more robust by considering proximity in terms of the arc length (see Fig. 9):

$$\boldsymbol{\pi}_r^{\mathcal{D}'} := \arg\min \left\| \Lambda(\boldsymbol{\pi}_r^{\mathcal{D}}) - \Lambda(\boldsymbol{\pi}_i^{\mathcal{D}'}) \right\|, \tag{14}$$

where $\Lambda(\boldsymbol{\pi})$ gives the arc length (or pair of arc lengths) of $\boldsymbol{\pi}$ and $i$ indexes the PoIs in $\mathcal{D}'$. This is especially true in the case of drawing machines where the tracer needs to make a full turn before coming close again to the same area. Moreover, it should be noted that the matching PoI does not always exist: some intersections or curvature maxima are only present in a limited range of parameter values. Therefore, we define a distance threshold $\sigma_{PoI}$ between the reference PoI and its match, and discard curves for which this limit is exceeded. This threshold can also be used to make the search more efficient: indeed, candidate PoIs in other curves need only be computed in the circle of radius $\sigma_{PoI}$ centered at the reference PoI.

## 5.2 Exploring the invariant space

Once the desired pattern invariants have been selected by the user, the challenge is to explore the resulting constrained parameter space. In the general case, the invariant space is difficult to determine analytically. Therefore, we opt for a sample-based local linear approximation (see Fig. 10 for an illustration).

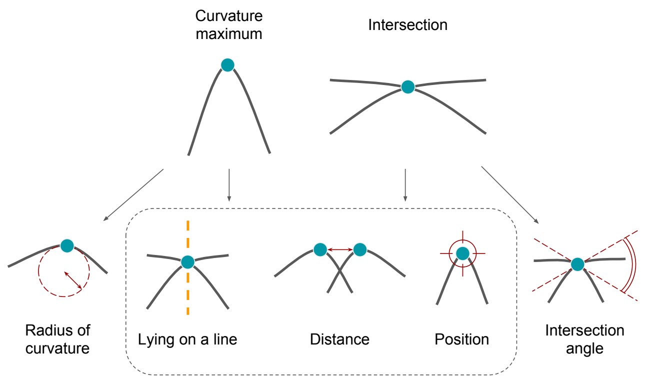

**Figure 8: Types of Points of Interest (PoI) and associated invariants supported by our system.**

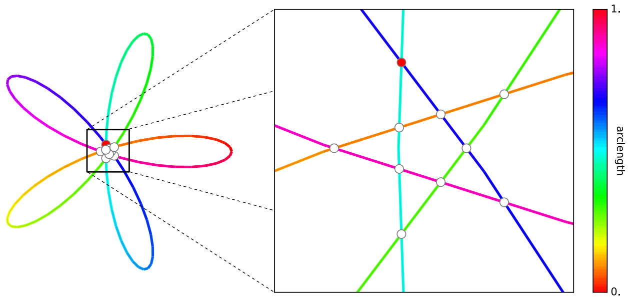

**Figure 9: Using unambiguous features to discriminate between Point of Interests. Although the red intersection point is close to the others, we can still differentiate it using the pair of arc lengths values at the crossing.**

In terms of interaction, our algorithm aims to provide the user with new sliders that allow interactive exploration. Since the approximation we use is only linear, regular re-projections and re-approximations of the invariant subspace are required. We perform them each time a slider is released after a move, which is preferable to continuous updates for two reasons: it ensures interactivity and allows *reversible* changes, ie. give the user the ability to come back continuously to a previous design while keeping the button pressed. We now describe our approach for sub-space approximation.

We consider an $n$-dimensional continuous parameter space implicitly bounded by several machine-intrinsic constraints, containing a point $\boldsymbol{p}_0$ associated with the initial drawing $\mathcal{D}_0$. For simplicity, we assume a single user-defined invariant expressed by

$$d_i^F \left( F(\boldsymbol{\pi}_r^{\mathcal{D}_0}), F(\boldsymbol{\pi}_r^{\mathcal{D}_i}) \right) = 0 \tag{15}$$

where $\mathcal{D}_i$ is the drawing associated to a neighboring point $\boldsymbol{p}_i$, $F$ is the feature of interest (real- or vector-valued), and $d_i^F$ is the Euclidean distance in the corresponding feature space.

First, we sample neighboring points $\boldsymbol{p}_i$, within the feasible continuous parameter domain, taking them on a grid whose resolution is adapted dimension-wise to the length of the feasible range. We instantiate the associated drawings $\mathcal{D}_i$, and track the corresponding PoI $\boldsymbol{\pi}_r^{\mathcal{D}_i}$. We define the *invariance score* as

$$S_i := \exp(-d_i^F). \tag{16}$$

We will use these scores as weights for the regression of the solution space. Before that, we filter the samples to keep only a fraction of the highest weights. We assume, given the locality of the neighborhood, that the resulting domain is convex and not disjoint.

Then, to perform the regression, we use a Weighted Principal Component Analysis (WPCA) centered on the starting point. Since the weights are our invariance scores, this algorithm provides a basis of vectors ordered by decreasing contribution to the invariant space. A local basis can therefore be taken as the first $m$ Principal Components, where $m$ is the dimensionality of the invariant subspace. It is important to note that $m$ cannot simply be deduced from the number of algebraic constraints, which are not necessarily independent. In other words, some constraints may be redundant, either between themselves or with the intrinsic constraints of the mechanism.

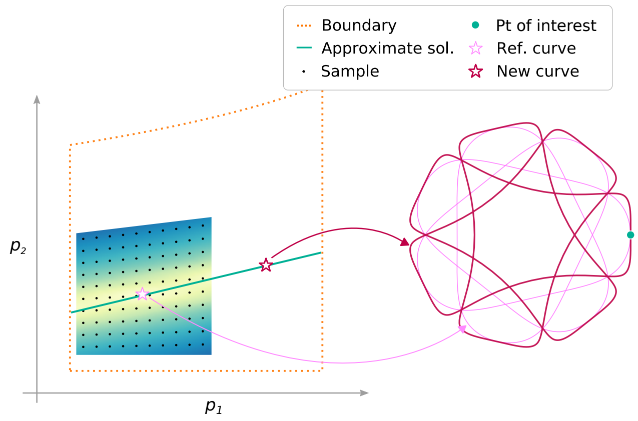

**Figure 10: Illustrating the invariant space with 2 continuous parameters. The user identifies a PoI directly on the curve and specifies its desired invariant (here: position). Our system then locally samples the parameter space, evaluates an invariance score (shown as a color map), and performs a linear regression on the sample values. Sweeping the regressed solution (cyan line) amounts to exploring desirable curves (e.g., maroon curve).**

In order to determine the dimensionality of the resultant space, we first make sure that it is not reduced to a singleton by checking the number of samples with a sufficiently high invariance score (superior to $\sigma_{inv} = 0.9$). If less than two points are found, we consider that the system is over-constrained and invite the user to remove one invariant. The WPCA gives us the proportion of variance explained by each Principal Component. Defining $v^1_{rel}$ as the highest relative variance in the set, we keep all components whose proportion is superior to $\sigma_{var} = 0.1 v^1_{rel}$. Each axis of the resulting subspace is mapped to a slider shown to the user. If no component is filtered out, we consider that all the invariants were redundant with the intrinsic constraints, and hence keep the original parameterization.

Next, we compute the bounds of the resultant solution space. Since the approximation is local, we do not need to allow too wide an amplitude around the starting position. Since each Principal Component is normalized, we put coarse bounds at −2 and 2. Even then, the intrinsic constraints may impose tighter bounds along some dimensions, which depend on the value of the other parameters; therefore, they need to be re-computed every time a slider is moved. We formulate this as a sequence of non-linear constrained optimization problems: for each parameter, with the other parameters held fixed, we successively find its minimal and maximal values. Please note that this optimization only uses the intrinsic constraints of the system, which do not require a simulation or the evaluation of PoIs (see supplementary document for details). Further, since we assumed that the local neighborhood was convex and connected, we expect a single range of possible values within the coarse bounds.

We are now ready to present the user with a set of sliders that can be moved while respecting the invariants. Once a slider is released, we update our model accordingly. First, we project the current position back onto the solution space, by finding the point closest to this position that maximizes the invariance score. Then, we re-compute a local approximation of the solution space, following the procedure that has just been described.

In addition, we make the system more intuitive to use by ensuring that the sliders have a temporally consistent *visual effect* on the drawing. Indeed, re-approximating the invariant subspace may typically result in the Principal Components flipping or rotating (see Fig. 11). Flipping can be easily resolved by comparing the old and new principal directions pairwise – since their order is preserved – and flipping them back if necessary. Rotation of principal directions, which typically happens when the spread is symmetrical (Fig. 11c), can be avoided by projecting the previous local basis onto the new one, and normalizing the resulting vectors. This ensures a consistent behavior of the sliders throughout the exploration.

## 6 RESULTS AND DISCUSSION

Our database of mechanisms contains four parametric models whose specifics are given in the supplementary document. Table 2 summarizes the main characteristics of these machines. While the Spirograph, the Cycloid Drawing Machine and the Hoot-Nanny are motivated by existing drawing machines, the elliptic Spirograph was designed by the authors to experiment with non-circular gears.

Various patterns designed with our system are shown in Figs. 2, 10, 12 and 13. Constrained exploration results are provided in Fig. 12 and in the accompanying videos, where we compare slider manipulation in the chosen design space to the corresponding changes happening to the base parameters. We further evaluated our method in two ways: first, we ensured that it produces mechanically functional machines by fabricating prototypes; second, we conducted a user study to assess the intuitiveness of our system compared to a forward simulator.

### 6.1 Constrained exploration results

We demonstrate examples of curve invariants for each row in Fig. 12. They were kept voluntarily simple to emphasize the effect of a given constraint (see the accompanying videos for more complex examples). Let us discuss each of these experiments.

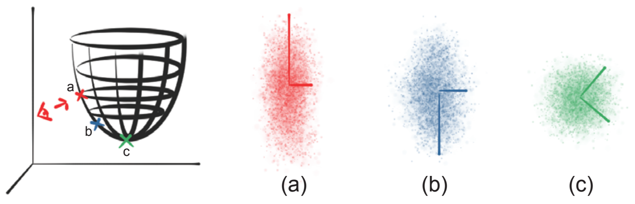

**Figure 11: Ensuring temporal consistency. Since the Principal Components may flip during a slider move from (a) to (b), or rotate when the user explores a near position (c), the axes of the new linear approximation are flipped or rotated when necessary, within the subspace of interest, to remain as consistent as possible with the original principal directions.**

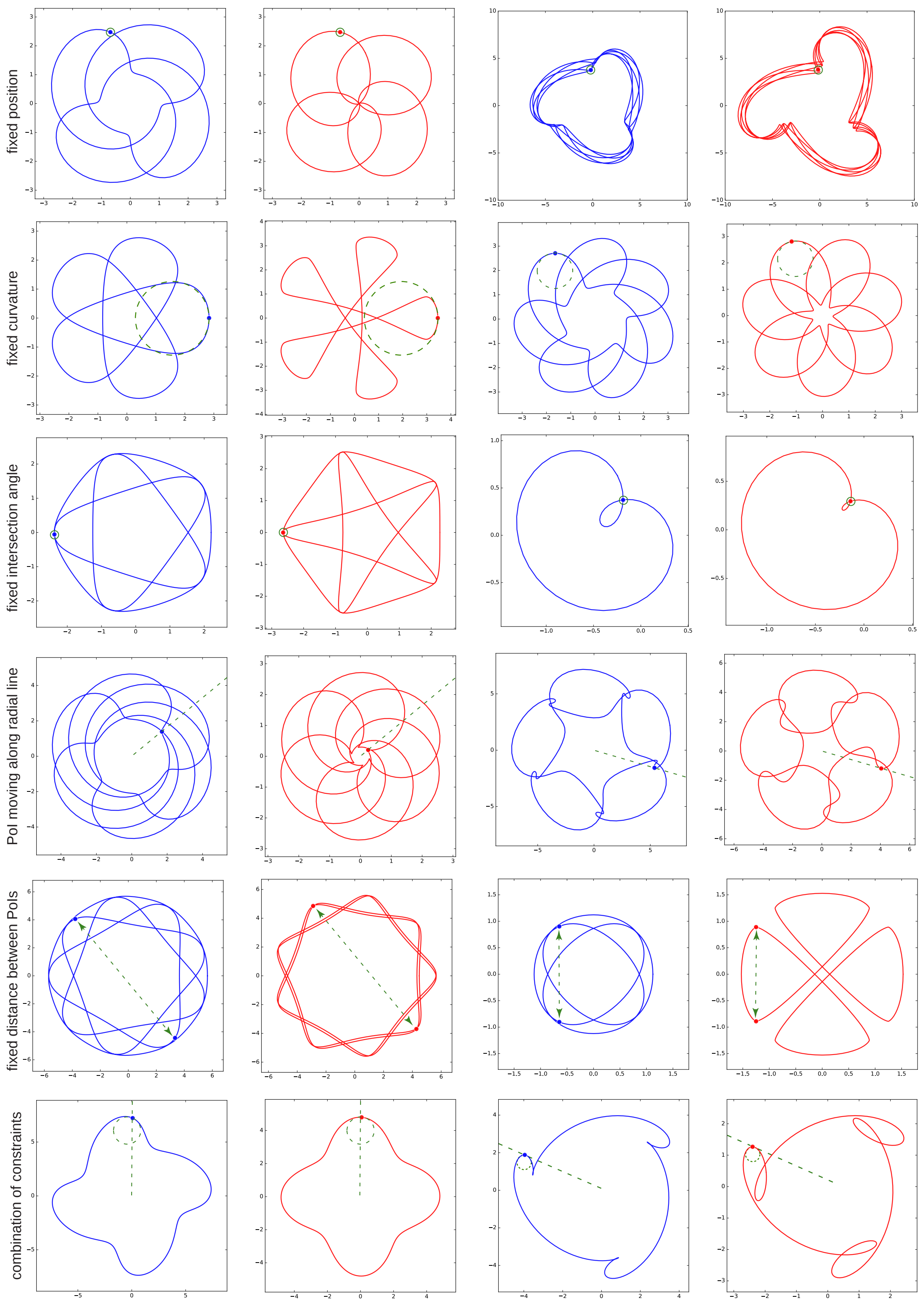

Figure 12: Examples of constrained variations obtained with our system (original curves in blue, modified in red). Our generated sliders allow significant visual changes to the curves, while respecting the visual constraints (one per row, two at the bottom).

**Table 2: Mechanisms implemented in our system.**

| Name | # exposed parameters (discrete+continuous) |
|---|---|
| Spirograph (S) | 2 + 1 |
| Elliptic Spirograph (ES) | 2 + 2 |
| Cycloid Drawing Machine (CDM) | 2 + 4 |
| Hoot-Nanny (HN) | 3 + 5 |

- *Fixed point.* The user fixed the location of the selected PoI. On the left (CDM), the interior boundary was pulled in, while keeping the external arc fixed. On the right (HN), the cusp point is fixed, while increasing the symmetric lobes.
- *Fixed curvature.* The user fixed the curvature at the selected PoI. In the left example (ES), the center was pulled in, while maintaining the PoI's curvature. In the right example (CDM), the central part was reduced and rotated, while maintaining the PoI's curvature.
- *Fixed intersection angle.* The user fixed the angle between tangents at the selected intersection point. In the left example (ES), the center was pulled in while preserving tangency between the curve segments (i.e., zero angle). In the right example (CDM), the loop size was changed, while keeping the inter-curve intersection angle (and symmetry).
- *Moving along radial line.* The user restricted the movement of the PoI along a radial line. On the left (CDM), the center was closed in while keeping the global orientation. On the right (HN), the central part was pulled in and the curvature at the cusp was changed, while keeping the original orientation.
- *Fixed distance between 2 PoIs.* The user fixed the distance between 2 selected PoIs. In the left example (ES), the external boundary size was maintained, while pulling the petals closer together. In the right example (ES), the size of the petals was held fixed, while pulling them apart.
- *Multiple specifications:* In these examples, multiple constraints were specified on selected PoIs. On the left (CDM), the asymmetry was changed while keeping the global orientation and curvature of petals. On the right (HN), the petals were made more ornamental while preserving their curvature and restricting movement along radial line.

## 6.2 Precise modeling and fabrication

We fabricated several examples of machines (see Fig. 13):

- the elliptic Spirograph, an easy to fabricate two-parts mechanism that we used to validate the first invariants;
- the Hoot-Nanny, which demonstrates our ability to manage devices with a wider range of parts and connectors.

Our general principle during the fabrication process was to laser-cut the precision-critical, horizontal parts, and to 3D-print the remaining custom connectors, which notably ensure the transmission of movement and support the different layers of flat components. While the vector files given to the laser cutter are automatically generated by a script, the 3D-printed components were designed by hand using CAD software, requiring to adjust tolerances to help the machine run smoothly.

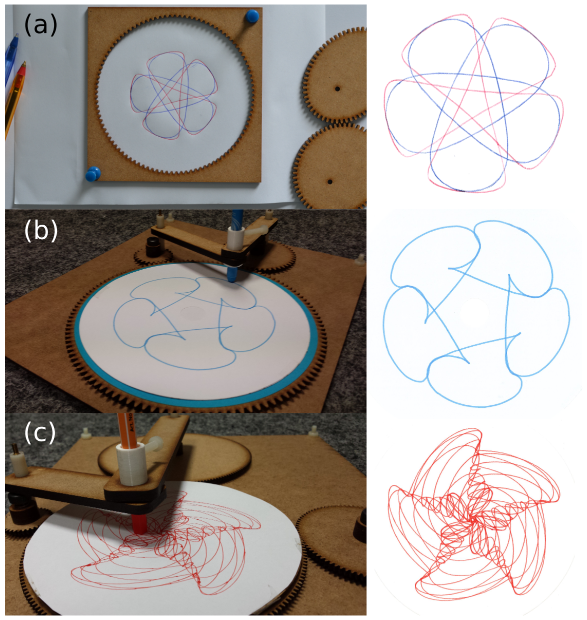

**Figure 13: Examples of fabricated prototypes: (a) Elliptic Spirograph with two curves drawn using a different elliptic gear; (b) and (c) Instances of the Hoot-Nanny. Drawn patterns on the right.**

One challenge encountered during fabrication was the design of gear profiles. Such profiles are usually not represented in CAD software, as they would unnecessarily make the geometric model more complex; moreover, these pieces are traditionally manufactured with normalized shaper cutters. Laser cutters, on the other hand, require a precise geometric model as input. Therefore, we implemented a procedural generation of involute gear profiles (which optimize the transmission of torques, see Fig. 14), for both circular and elliptical gears. The latter, which is less common, was derived from a method by Bair [3].

Pictures of some of the fabricated examples are given Fig. 2 and 13. Demonstration of their usage is given in the main supplementary video.

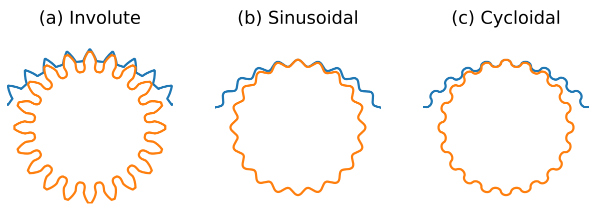

**Figure 14: Different possible gear profiles. Profile (a), although more complex to generate, maximizes torque transmission.**

## 6.3 User study

We conducted a user study with 8 participants to validate the efficiency of a mode of exploration based on visual constraints. We chose to focus on an important premise of our method – the fact that defining visual preferences can help navigating the configuration space more easily – rather than trying to evaluate the entire pipeline. This choice allowed to focus on the core contribution of constrained exploration, and made user sessions reasonably short in time and easier to compare.

We defined the following protocol. Each user session was divided into four pattern-editing tasks. In each of these tasks, the candidate was asked to transform an initial curve A into a target curve B, using sliders, in less than two minutes. The set of target patterns was the same for all users, while initial patterns were randomly generated for each new session. The editing operation had to be performed twice: once with the basic machine parameters (subtask 1), and once with parameters corresponding to a predefined visual invariant (subtask 2). The interface was kept minimal, has shown in Fig. 15 left. In order to focus solely on the efficiency of the parameterization, we designed both subtasks to be as close as possible interaction-wise. First, the same number of sliders was exposed each time (despite our method allowing to reduce this number), and the order in which the subtasks successively appeared was randomized. Second, the predefined invariant was *n*ot shown to the user. Lastly, we presented the re-projection and re-approximation process as a little "helper" which could be called by pressing the spacebar, triggering a change in the curve and in the behavior of the sliders. This "helper" had a negligible effect in the base case: a dummy waiting time was triggered (inferior to the time required by the true "helper"), and a tiny perturbation was added to the sliders. This managed to make both versions completely indistinguishable for all users. At the end of the session, candidates were presented with a table displaying their results (see Fig. 15 right). For each task, they were asked to rate the similarity with the target pattern between 0 and 5.

Results are given for two metrics (total time and perceived dissimilarity) in Fig. 16. With comparable times, candidates were in most cases able to reach a final result perceptually closer to the

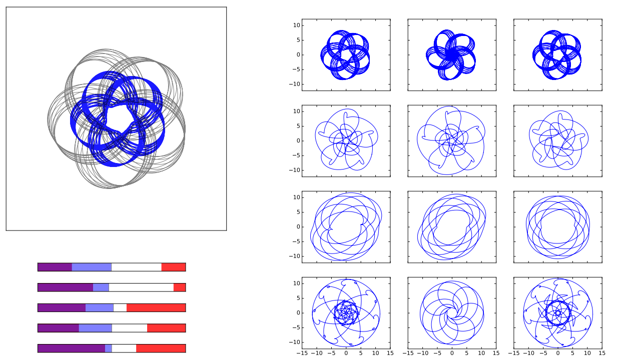

Figure 15: Left: interface for a subtask of the user study (target pattern in grey). Right: summary sheet presented to the user in order to rate the results (each column is respectively the target pattern, and results of subtask 1 and 2 in an arbitrary order).

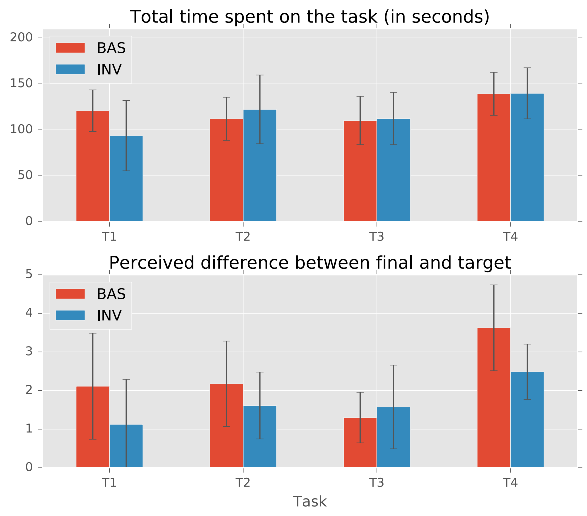

Figure 16: User study results (mean and standard deviation bars). "BAS" and "INV" respectively denote the base and invariant-space parameterizations.

target curve. The slightly higher times in our case can be attributed to the re-approximation step, which could take up to three times longer than the dummy step defined for the base case. This could, however, be reduced with a more efficient implementation. Moreover, additional time-independent metrics, namely the total number of slider moves and the total Euclidean distance travelled in the parameter space (given as supplementary material), demonstrate that our parameterization was more efficient.

Lastly, we note that this study only partially validates the efficiency of our method, as candidates were not allowed to choose their own invariants (which would have required a longer familiarization time). Therefore, the intuitiveness of the Points of Interests and associated invariants has not been assessed. Moreover, an editing task with a specific target does not exactly correspond to the exploration scenario we envisioned for this method; it is, however, easier to evaluate quantitatively.

## 6.4 Discussion

Our method presents several limitations, which open the way for future developments:

- The curve metric used for pattern retrieval, while efficient, does not necessarily reflect perceived proximity between drawings. A possible improvement could be to train a feature-based curve metric with a perceptual study, as done by Coros et al. [8], while being aware that perception tends to be application dependent, and may not be as general.
- Our naive grid-based local sampling method is combinatorial in the number of parameters, which allowed to keep interactive rates up to only six continuous parameters in our single-threaded Python implementation; we note, however, that computing samples and PoIs could be done in parallel, and that a subset of the most significant parameters can be preselected before applying our method.

- Bounded sliders are straightforward to implement, but they lack a clear meaning in terms of visual effect on the drawing. Possible improvements include adding intuitive visual clues beside each slider, or more advanced controls.
- Lastly, transforming an abstract mechanical model into a fabricable assembly remains a tedious task, as many physical aspects that are neglected (gear backlash, defaults in 3D-printed parts, frictions and instabilities) may end up impairing the final drawing quality. This notably explains why we were not able to reliably build machines with a higher number of gears: while our method is completely able to handle more complex devices virtually, the accumulation of physical errors made the resulting prototypes too unstable to produce drawings of satisfying quality.

The specific application domain presented in this paper, while interesting from an educational and artistic point of view, could be seen as limited in terms of practical value. The core concepts of our method, however, are not bound to drawing machines: they remain valid for more general mechanisms, as illustrated in Fig. 17. Following our previous work [28], we selected one of the elementary mechanisms from Coros et al.'s paper [8] and added it to our system. Connecting a 'leg' to the end-effector (animated with inverse kinematics), we can imagine a situation where a user wishes to have the foot kicking a ball at the maximum of curvature (therefore deciding to fix its position) while exploring the various trajectories taken by the foot to reach this position.

Furthermore, designing cyclic motions is also relevant in industrial settings such as assembly lines, where the available constraints on a point of interest could be extended to speed and force, with no change needed in the rest of the pipeline.

Lastly, this method should be applicable to a wider range of generative design systems with a set of continuous and discrete parameters as input, assuming access to a reasonably fast forward simulation (or procedural generation) leading to an output having measurable (and desirable) invariants.

## 7 CONCLUSION

We presented a framework for exploring and fabricating drawing machines. The user can directly select among different machines along with their parameter settings using high-level scribbles, and then refine the retrieved drawing pattern by specifying constraints on dynamically computed feature points.

The main idea is to locally sample the design space and regress to the subspace that best preserves user-specified constraints on Points of Interest in the drawing. We linearize the space using a weighted PCA and expose the desirable region of the design space to the user. The user can simply navigate the solution space using an intuitive slider interface. We tested our setup on several classical drawing machines, designed various patterns using it, and fabricated a few prototypes to demonstrate the effectiveness of the approach.

In the future, we would like to extend our framework in different ways. An important next step would be to support interactive topological changes to machine configurations and allowing users to seamlessly transition across such variations directly by sketching curves and indicating suitable invariants. Another interesting extension would be to support 3D space curve drawing machines which would be relevant for recently introduced 3D doodle pens. Finally, we plan to investigate how our dynamic reparameterization approach can be used in other contexts of design exploration where analytically solving for and characterizing valid solution spaces is too expensive and impractical.

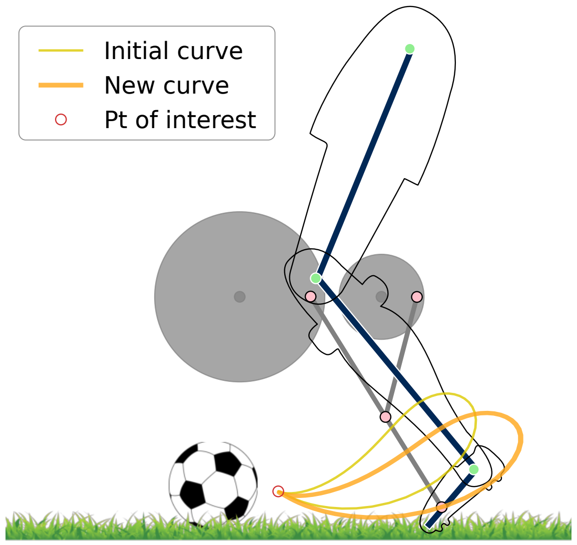

**Figure 17: Extension to a mechanical character: a leg kicking a soccer ball.**

## ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their valuable comments and suggestions. We also thank Aron Monszpart for proofreading the paper and Estelle Charleroy for helping with the video. This work was supported by the European Research Council Starting Grant SmartGeometry 335373 and Advanced Grant Expressive 291184, and gifts from Adobe. Prototypes were fabricated with the Equipex Amiqual4Home (ANR-11-EQPX-0002).

## REFERENCES

[1] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing Using Symbolic Kinematics. *ACM Trans. Graph.* 34, 4, Article 99 (July 2015), 8 pages. https://doi.org/10.1145/2766985
[2] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4 (2014), 1–96. https://doi.org/10.1145/2601097.2601157
[3] Biing W. Bair. 2002. Computerized tooth profile generation of elliptical gears manufactured by shaper cutters. *Journal of Materials Processing Technology* 122, 2-3 (2002), 139–147. https://doi.org/10.1016/S0924-0136(01)01242-0
[4] Hichem Barki, Lincong Fang, Dominique Michelucci, and Sebti Foufou. 2016. Re-parameterization reduces irreducible geometric constraint systems. *CAD Computer Aided Design* 70 (2016), 182–192. https://doi.org/10.1016/j.cad.2015.07.011
[5] Gaurav Bharaj, David I W Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational Design of Metallophone Contact Sounds. *ACM Trans. Graph.* 34, 6 (2015), 1–13. https://doi.org/10.1145/2816795.2818108
[6] Martin Bokeloh, Michael Wand, Hans-Peter Seidel, and Vladlen Koltun. 2012. An algebraic model for parameterized shape editing. *ACM Trans. Graph.* 31, 4 (2012), 1–10. https://doi.org/10.1145/2185520.2335429
[7] Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph.* 32, 6 (2013), 1–11. https://doi.org/10.1145/2508363.2508400

[8] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph.* 32, 4 (2013), 1–83. https://doi.org/10.1145/2461912.2461953

[9] G. Dantzig, R. Fulkerson, and S. Johnson. 1954. Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America* 2, 4 (1954), 393–410. https://doi.org/10.1287/opre.2.4.393

[10] Ian L Dryden and Kanti V Mardia. 1998. *Statistical shape analysis.* Vol. 4. Wiley Chichester.

[11] Thomas Eiter and Heikki Mannila. 1994. *Computing Discrete Frechet Distance.* Technical Report CD-TR 94/64. Information Systems Department, Technical University of Vienna.

[12] Max Ernst. 1942-7. Young Man Intrigued by the Flight of a Non-Euclidean Fly. (1942-7). Collection: Hamburger Kunsthalle, Hamburg/Pietzsche Collection.

[13] Joe Freedman. 2015. Cycloid Drawing Machine | leafpdx. http://leafpdx.bigcartel.com/product/cycloid-drawing-machine. (2015). Accessed: 2018-04-30.

[14] Ioannis Fudos and Cm Hoffmann. 1997. A graph-constructive approach to solving systems of geometric constraints. *ACM Trans. Graph.* 16, 2 (1997), 179–216. https://doi.org/10.1145/248210.248223

[15] James Nolan Gandy. 2016. James Nolan Gandy. http://www.jamesnolangandy.com/. (2016). Accessed: 2018-04-30.

[16] Pablo Garcia. 2016. DrawingMachines.org. https://drawingmachines.org/. (2016). Accessed: 2018-02-01.

[17] Paul Guerrero, Gilbert Bernstein, Wilmot Li, and Niloy J. Mitra. 2016. PATEX: Exploring Pattern Variations. *ACM Trans. Graph.* 35, 4, Article 48 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925950

[18] Inc. Gurobi Optimization. 2016. Gurobi Optimizer Reference Manual. https://www.gurobi.com. (2016).

[19] Leon M. Hall. 1992. Trochoids, Roses, and Thorns - Beyond the Spirograph. *College Mathematics Journal* 23, 1 (1992), 20–35.

[20] Alexandra Ion, Johannes Frohnhofen, Ludwig Wall, Robert Kovacs, Mirela Alistar, Jack Lindsay, Pedro Lopes, Hsiang-Ting Chen, and Patrick Baudisch. 2016. Metamaterial Mechanisms. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 529–539. https://doi.org/10.1145/2984511.2984540

[21] A. B. Kempe. 1875. On a General Method of describing Plane Curves of the nth degree by Linkwork. *Proceedings of the London Mathematical Society* s1-7, 1 (1875), 213–216. https://doi.org/10.1112/plms/s1-7.1.213

[22] B. Koo, J. Hergel, S. Lefebvre, and N. Mitra. 2016. Towards Zero-Waste Furniture Design. *IEEE Transactions on Visualization and Computer Graphics* 99 (2016). https://doi.org/10.1109/TVCG.2016.2633519

[23] Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. 2014. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.* 33, 6 (2014), 1–9. https://doi.org/10.1145/2661229.2661289

[24] Yang Liu and J. Michael McCarthy. 2017. Design of Mechanisms to Draw Trigonometric Plane Curves. *Journal of Computing and Information Science in Engineering* 9, 2 (2017). https://doi.org/10.1115/1.4035882

[25] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph.* 33, 4 (2014), 97. https://doi.org/10.1145/2601097.2601168

[26] Jonàs Martínez, Jérémie Dumas, and Sylvain Lefebvre. 2016. Procedural Voronoi Foams for Additive Manufacturing. *ACM Trans. Graph.* 35 (2016), 1 – 12. https://doi.org/10.1145/2897824.2925922

[27] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph.* 32, 4 (2013), 81:1–81:10. https://doi.org/10.1145/2461912.2461957

[28] Robin Roussel, Marie-Paule Cani, Jean-Claude Léon, and Niloy J. Mitra. 2017. SPIROU: Constrained Exploration for Mechanical Motion Design. In *Proceedings of the 1st Annual ACM Symposium on Computational Fabrication (SCF '17)*. ACM, New York, NY, USA, Article 7, 11 pages. https://doi.org/10.1145/3083157.3083158

[29] Norman Routledge. 2008. Computing Farey Series. *The Mathematical Gazette* 92, 523 (2008), 55–62.

[30] Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab Forms: Customizable Objects for fabrication with Validity and Geometry Caching. *ACM Trans. Graph.* 34, 4 (2015), 1–100. https://doi.org/10.1145/2766994

[31] Meera Sitharam and Menghan Wang. 2014. How the Beast really moves: Cayley analysis of mechanism realization spaces using CayMos. *CAD Computer Aided Design* 46, 1 (2014), 205–210. https://doi.org/10.1016/j.cad.2013.08.033

[32] Ivan E. Sutherland. 1963. Sketchpad: A Man-machine Graphical Communication System. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference (AFIPS '63 (Spring))*. ACM, New York, NY, USA, 329–346. https://doi.org/10.1145/1461551.1461591

[33] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-based Characters. *ACM Trans. Graph.* 33, 4, Article 64 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601143

[34] Valdis Torms. 2015. 3d printēts spirogrāfs. http://www.3domas.lv/2015/07/pastaisits-spirografs_1.html. (2015). Accessed: 2018-04-30.

[35] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.* 31, 4 (2012), 1–11. https://doi.org/10.1145/2185520.2335437

[36] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes. *ACM Trans. Graph.* 33, 4 (2014), 1–10. https://doi.org/10.1145/2601097.2601129

[37] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. 2015. Semantic Shape Editing Using Deformation Handles. *ACM Trans. Graph.* 34, 4, Article 86 (July 2015), 12 pages. https://doi.org/10.1145/2766908

[38] Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-case Structural Analysis. *ACM Trans. Graph.* 32, 4, Article 137 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461967

[39] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided Mechanical Toy Modeling. *ACM Trans. Graph.* 31, 6 (2012), 1–127. https://doi.org/10.1145/2366145.2366146