SPIROU: Constrained Exploration for Mechanical Motion Design

Robin Roussel, Marie-Paule Cani, Jean-Claude Léon, Niloy J Mitra Mitra

SPIROU: Constrained Exploration for Mechanical Motion Design

Robin Roussel
University College London
Grenoble Alpes University
INRIA
robin.roussel.15@ucl.ac.uk

Marie-Paule Cani
Grenoble Alpes University
INRIA
marie-paule.cani@inria.fr

Jean-Claude Léon
Grenoble Alpes University
INRIA
jean-claude.leon@ense3.grenoble-inp.fr

Niloy J. Mitra
University College London
n.mitra@cs.ucl.ac.uk

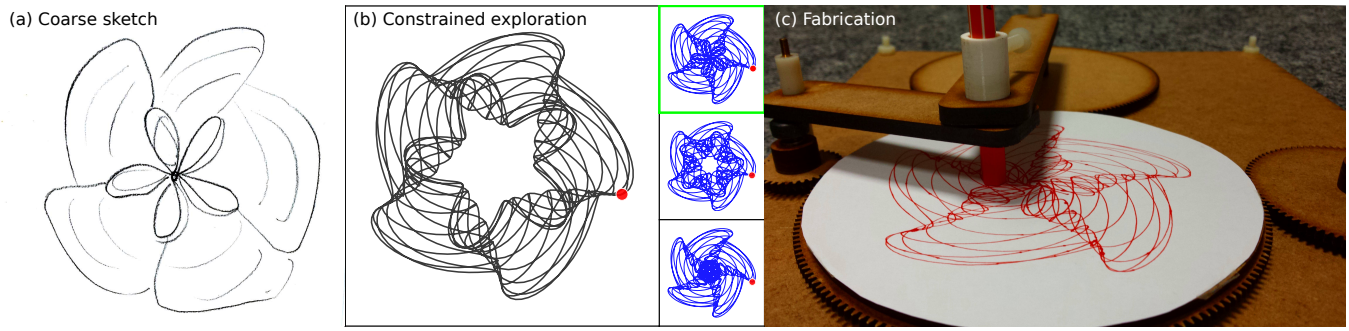

Figure 1: A drawing machine designed with our system. The user first selects a mechanically feasible drawing by providing a rough sketch (a), and is then able to interactively explore local alternatives (b) by defining visual constraints directly on the pattern (here, the cusp position).

ABSTRACT

Mechanisms are ubiquitous in our daily lives, and the motion they are able to transmit is often a critical part of their function. While fabrication from a virtual model can be done relatively easily in a fab lab, creating or customizing a model according to functional specifications remains a challenging task. We focus on a fascinating application: drawing machines. Devices such as the popular Spirograph can easily generate intricate patterns from an assembly of simple mechanical elements. Designing such machines, however, is made particularly tedious by the complex influence each configuration parameter has on the final drawing. 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 allow the user to effectively explore the high dimensional configuration space of such fabricable machines. To this end, we propose a novel method that dynamically reparameterizes the local configuration space in order to allow the user to explore drawing 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.

CCS CONCEPTS

•**Computing methodologies** → **Computer graphics;** *Shape modeling;*

KEYWORDS

design space, exploration, constraints, drawing machines, mechanical motion, fabrication

ACM Reference format:
Robin Roussel, Marie-Paule Cani, Jean-Claude Léon, and Niloy J. Mitra. 2017. SPIROU: Constrained Exploration for Mechanical Motion Design. In *Proceedings of SCF '17, Cambridge, MA, USA, June 12-13, 2017,* 11 pages.
DOI: http://dx.doi.org/10.1145/3083157.3083158

1 INTRODUCTION

Toy of the Year in 1967, the Spirograph is a simple-to-use family of interlocking cogs and teethed rings that allows users to draw an impressive variety of patterns. Although many other mechanical drawing tools preceded and followed it (see Fig. 2), this modest set of shapes has marked a generation, and remains one of the most

SCF '17, Cambridge, MA, USA

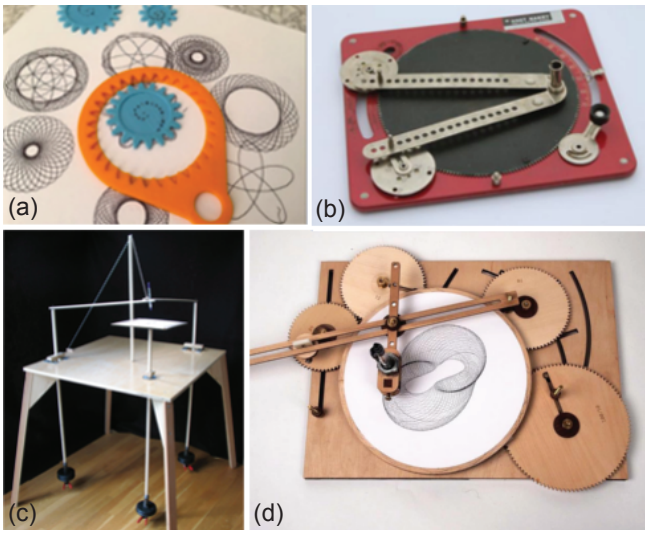

Figure 2: Some examples of drawing machines: (a) Spirograph, (b) Hoot-Nanny, (c) Harmonograph, (d) Cycloid drawing machine.

well-remembered today. As a product of the relationship between art and technology, such drawing devices are still popular across children, enthusiastic inventors, and makers. The simplicity of the mechanical parts involved makes it easy to fabricate with intricate designs being possible using even a pair of gears. More significantly, new personal fabrication devices such as laser cutters or 3D printers open the door to customized drawing machines, leading in turn to new and fascinating patterns.

Designing such machines, however, is particularly challenging. First, many drawing devices transform an input rotational movement into a more complex cyclic output by combining oscillations of different periods. As the result is governed by modular arithmetic between the periods, it is perceptually highly sensitive to small variations in some shape parameters, making the model chaotic if not handled properly. In particular, a naive realization can easily result in curves that may not even close in a finite number of cycles. Second, as the number of parts increases, so does the number of shape parameters. While this greatly enriches the space of possible curves, the increase in the number of controls makes it very difficult to manually refine a design, as machine parameters are correlated and each one has a complex influence on the pattern drawn.

In this paper, we propose a constraint-based exploration framework to design drawing machines while allowing users to directly interact with the end result – the drawing. Our goal is to allow an intuitive customization of highly structured curves, while ensuring that they can be physically realized. Therefore, in contrast to previous work [Bächer et al. 2015], we focus on easing the exploration of local design alternatives, rather than on computing a specific mechanism from an input end-effector trajectory.

Our exploration workflow consists in a coarse-to-fine definition of visual preferences that progressively refine the choice of drawings. First, as an entry point into the design space, the user draws a coarse sketch that suggests the global properties of the desired pattern (e.g., order of rotational symmetry and coarse shape features). After selecting an initial drawing among the suggestions proposed by the system, changes can be made via sliders within a domain that respects the feasibility constraints of the corresponding mechanism. As a key interaction, the user can define visual preferences directly on the drawing, and explore local variations that respect 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 to a format usable for laser cutting (see Figure 1).

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, approximates the subspace respecting user-defined geometric constraints, 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.

2 RELATED WORK

Drawing machines have a long history in recreational art, mathematics, and more broadly in the form of toys (for a historical overview, see https://drawingmachines.org). Their popularity stems from the simplicity of their constructions that can surprisingly produce a vast array of interesting and non-trivial patterns. While forward simulating such machines is relatively straightforward, the inverse problem of *designing* and *exploring* such machines based on target specifications has not yet been investigated. Here, we discuss related advances in computational design in different application settings, both for inverse modeling and also for 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 the configuration space (of part connections and parameters) to retrieve a local arrangement of parts, and then refine them using a gradient-descent based optimization to fit to a target specified motion. For example, Zhou et al. [2012] and Coros et al. [2013] design automaton characters, while Ceylan et al. [2013] design automata to replicate specified motion sequences.

In more interactive design settings, Umetani et al. [2014] design paper airplanes based on their predicted flight dynamics; Thomaszewski et al. [2014] use global optimization to design linkage-based characters; while Bächer et al. [2015] develop a system to support interactive editing of fabricable linkages. Recently, Ion et al. [2016] 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 directly author a target automata or linkage-based kinematic chain. We develop the first framework to support constrained exploration of mechanisms. Different from the above works, we

allow users to edit the current design (i.e., pattern drawn by the machine) by directly interacting with its feature points, thus allowing them to specify target properties and relations, instead of explicitly specifying the final pattern.

Additionally, in a classical constrained modeling setup, design from geometric constraints specifications has also been studied. Analyzing and solving such constraints have been tackled by Fudos et al. [1997] in a general setting, and Sitharam et al. [2014] in the context of mechanisms. Theoretically, even determining parameter bounds characterizing the (constrained) solution domain remains a known difficult topic [Barki et al. 2016]. Instead, in this work, we seek a general approximation strategy that support such interactive design space exploration for an intuitive workflow.

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 [Prévost et al. 2013], or updating object shape for desired moment of inertia for object spin [Bächer et al. 2014], adaptively adjusting object parts for better reinforcement and strength [Lu et al. 2014; Zhou et al. 2013], designing hollow chambers for desired acoustic behavior [Bharaj et al. 2015], zero-waste furniture design [Koo et al. 2016], or modeling elastic behavior of foam microstructures for procedurally generating them for target material properties [Martínez et al. 2016].

Closer to our concerns, Umetani et al. [2012] proposed a furniture modeling system that actively guides the user to navigate valid regions of the design space; Bokeloh et al. [2012] and Yumer et al. [2015] developed modeling systems that preserve high-level structural and semantic relations in edited 3D models, while Koo et al. [2014] 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. [2015], 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. [2016] 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 new and complex problem of patterns traced out by drawing machines, based on an end effector's motion.

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. 3). The first step, described in Sec. 4, consists in selecting an appropriate drawing 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 ability of invariant-based parameterization to efficiently help navigating the configuration space (Sec 6.3).

Let us now define the terminology used in the rest of the paper. A mechanical drawing machine comprises of:

- A *design space* made up of a set of (discrete or continuous) parameters, implicitly bounded by a system of algebraic constraints (typically nonlinear).
- A *simulator* that works out the pattern by tracing the machine over time. Optionally it outputs a time series of the positions and orientations of each component, which is useful for visual inspection. Please note that the simulator should be able to automatically determine how long the simulation should run until the drawing is completed.
- A *representation* of the associated 3D physical mechanism, possibly with different levels of detail (coarse for visual checking, and detailed for file export and fabrication).

Note that we have two levels of representation: the *mechanism model*, characterized by measurable dimensions, and the *pattern* it traces out, which is visible to the user. In this paper, we will use 'configuration space' or 'parameter space' when referring to the set of possible combinations of input values that translate into layout specifications for the mechanism; and use 'curve space' or 'drawing space' for the 2D space in which the drawing is realized.

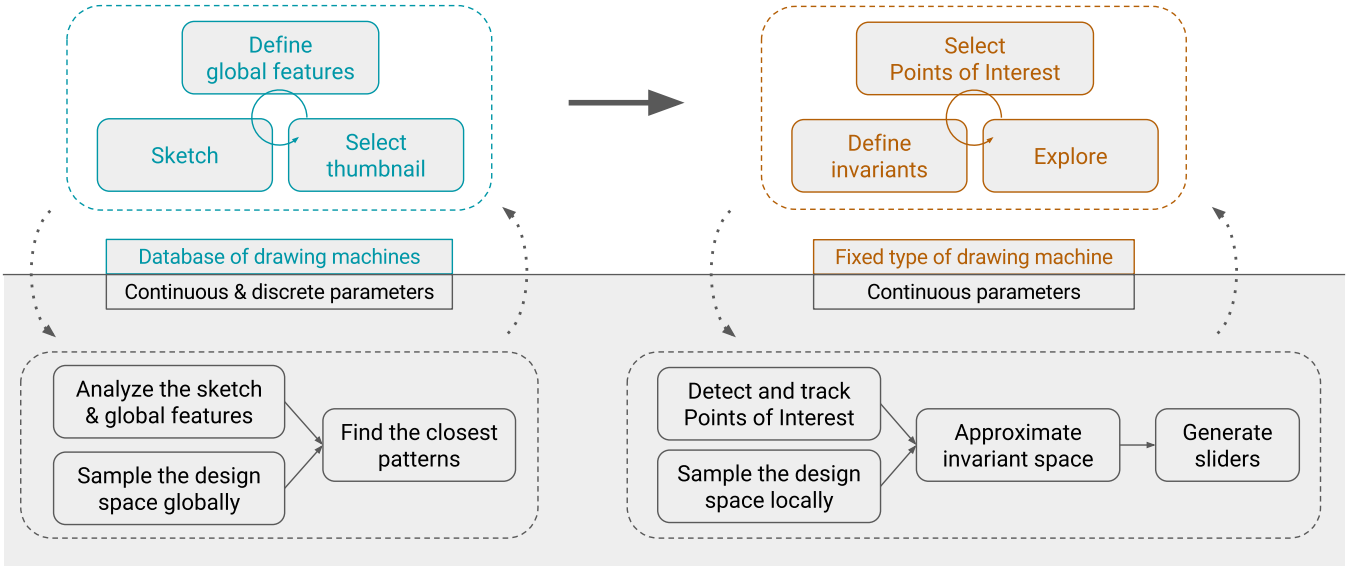

Figure 3: Our design workflow, separated into the actions performed by the user (top) and the system (bottom). Blue actions (left) correspond to the high-level sketching phase, while orange actions (right) belong to the constrained exploration phase. Dotted arrows represent transmission of information, and solid arrows represent sequences (straight) or iterations (circular).

4 PATTERN RETRIEVAL

The first step of the design workflow is a sketch-based exploration of the available patterns. Essentially we have an inverse problem: finding the parameter combination such that the simulated motion will give the feasible drawing closest to the user's sketch. Since the patterns produced by drawing machines are most often abstract, intricate, and generally tedious to sketch precisely, we only consider this step as a way for the user to define global characteristics of the drawing. By the properties of gearing, this notably amounts to assigning a value to the discrete parameters of the machine (please refer to the supplementary document for details).

The user can either input a hand-drawn sketch (Figure 1 (left)), or roughly sketch the desired curve directly in our system. In the latter case, she can use construction lines and pre-set the order of rotational symmetry, having her pen strokes automatically symmetrized in the other sectors (Figure 4 (top left)).

Our curve retrieval algorithm then runs through the database of mechanisms to find the closest matching patterns, as follows. Using a naive grid-based exploration of the feasible design space, each sample provides a drawing via a quick simulation, which is it turn normalized and compared against the input sketch. Our dissimilarity measure is based on the distance field of the normalized sketch (Figure 4 (top right)), which has the advantage of posing few requirements in terms of regularity (in particular, individual strokes can overlap and need not be connected). The previously defined construction lines are used to prune the search space, improving the efficiency of the query. The closest results are finally presented to the user (Figure 4 bottom).

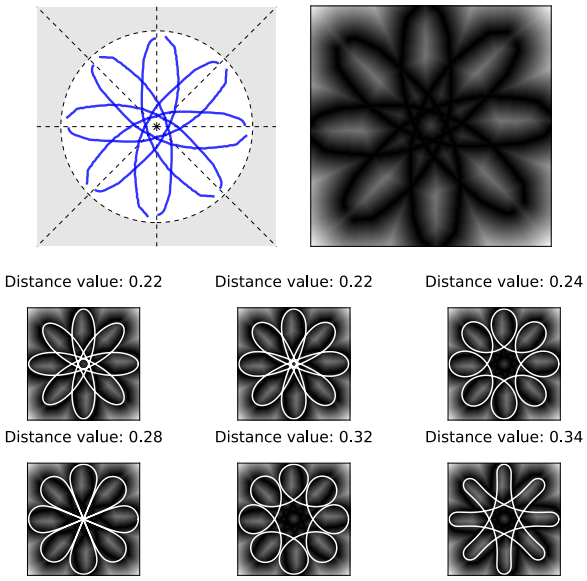

Figure 4: Starting from a user sketch (top-left), we compute a distance field (top-right), which is then used to score the different sampled patterns, pre-filtered with the clues given by the construction lines. The top 6 results are presented to the user – lower value indicates a better candidate.

5 CONSTRAINED EXPLORATION

Once the global features of the drawing and the discrete parameters for the selected drawing machine are fixed, the user can focus on fine tuning the continuous parameters. We note that an intuitive

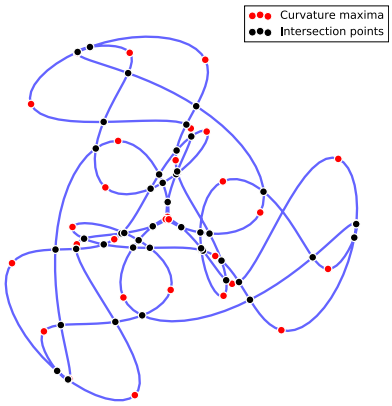

Figure 5: Example of Points of Interest in a drawing.

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).

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 Figure 5). 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 Figure 6).

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. Please 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 that allows the user to choose which feature should be frozen.

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 $\boldsymbol{\pi}_r^{\mathcal{D}}$ selected in $\mathcal{D}$, a naive criterion for such correspondence is to superimpose

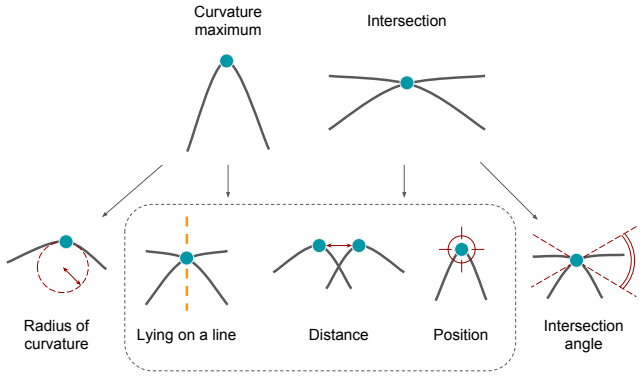

Figure 6: Types of Points of Interest (PoI) and associated invariants supported by our system.

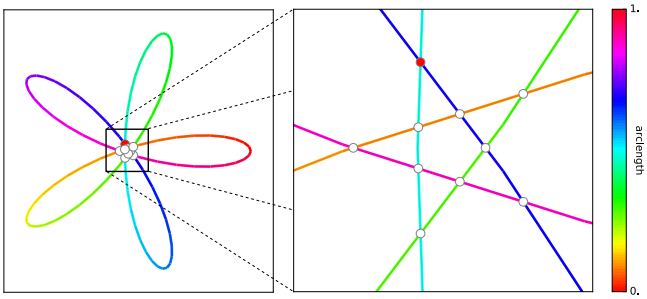

Figure 7: Using unambiguous features to track Point of Interests from one drawing to another. Although the highlighted red intersection point is close to the other intersection points, we can still uniquely identify it as the intersection of two arcs (purple and cyan) in the implicitly defined space of arc length parameters.

both drawings and take the closest PoI in $\mathcal{D}'$. However, in some configurations, 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 Figure 7):

$$\boldsymbol{\pi}_r^{\mathcal{D}'} \coloneqq \arg\min \left\| \Lambda(\boldsymbol{\pi}_r^{\mathcal{D}}) - \Lambda(\boldsymbol{\pi}_i^{\mathcal{D}'}) \right\|, \tag{1}$$

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 σ_{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 σ_{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 Figure 8 for an illustration).

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 a 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

$$d_i^F \left(F(\boldsymbol{\pi}_r^{\mathcal{D}_0}), F(\boldsymbol{\pi}_r^{\mathcal{D}_i}) \right) = 0 \tag{2}$$

where $\mathcal{D}_i$ is the drawing associated to a neighboring point $\boldsymbol{p}_i$, F is the feature of interest (which can be 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

$$S_i \coloneqq \exp(-d_i^F). \tag{3}$$

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.

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_{rel}^1 as the highest relative variance in the set, we keep all components whose proportion is superior to $\sigma_{var} = 0.1 v_{rel}^1$. 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

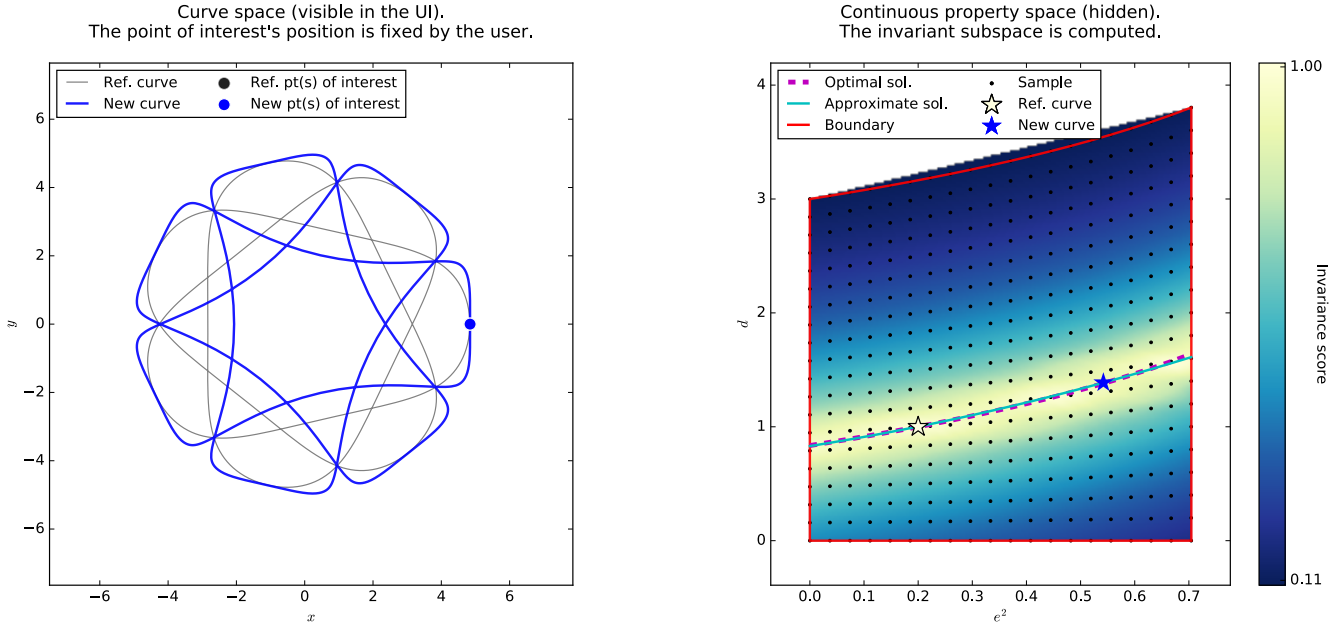

Figure 8: Illustrating the invariant space with 2 parameters. (Left) The user identifies a PoI directly on the curve and specifies its desired invariant. Our system then locally samples the parameter space (shown as dots on the right figure), evaluates an invariance score, and performs a regression on the sample values (shown as a polynomial fit here, but linear in the general case). Sweeping the regressed solution (indicated as the cyan curve) amounts to exploring desirable curves (e.g., blue curve pattern on the left). The magenta curve shows the analytical solution that could be obtained in this simple case.

position back onto the solution space, by finding the point closest to this position that maximizes the invariance score. Then, we recompute 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 Figure 9). 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 (Figure 9c), 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.

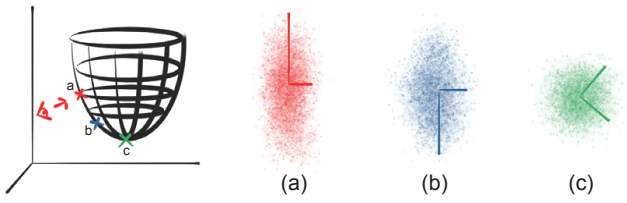

Figure 9: Ensuring temporal consistency. Since the Principal Components may flip during a slider move from (a) to (b), or rotate when the user explores a near position (c), axes belonging to the new linear approximation are flipped or rotated when necessary within the subspace of interest to remain as consistent as possible with the original principal directions.

6 RESULTS AND DISCUSSION

Our database of mechanisms contains four parametric models whose specifics are given in the supplementary document. Table 1 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 Figures 1, 8, 10 and 11. Constrained exploration results are provided in Figure 10 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 compare the intuitiveness of our system with regard to a forward simulator.

6.1 Constrained exploration results

We demonstrate examples of curve invariants for each row in Figure 10. 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.

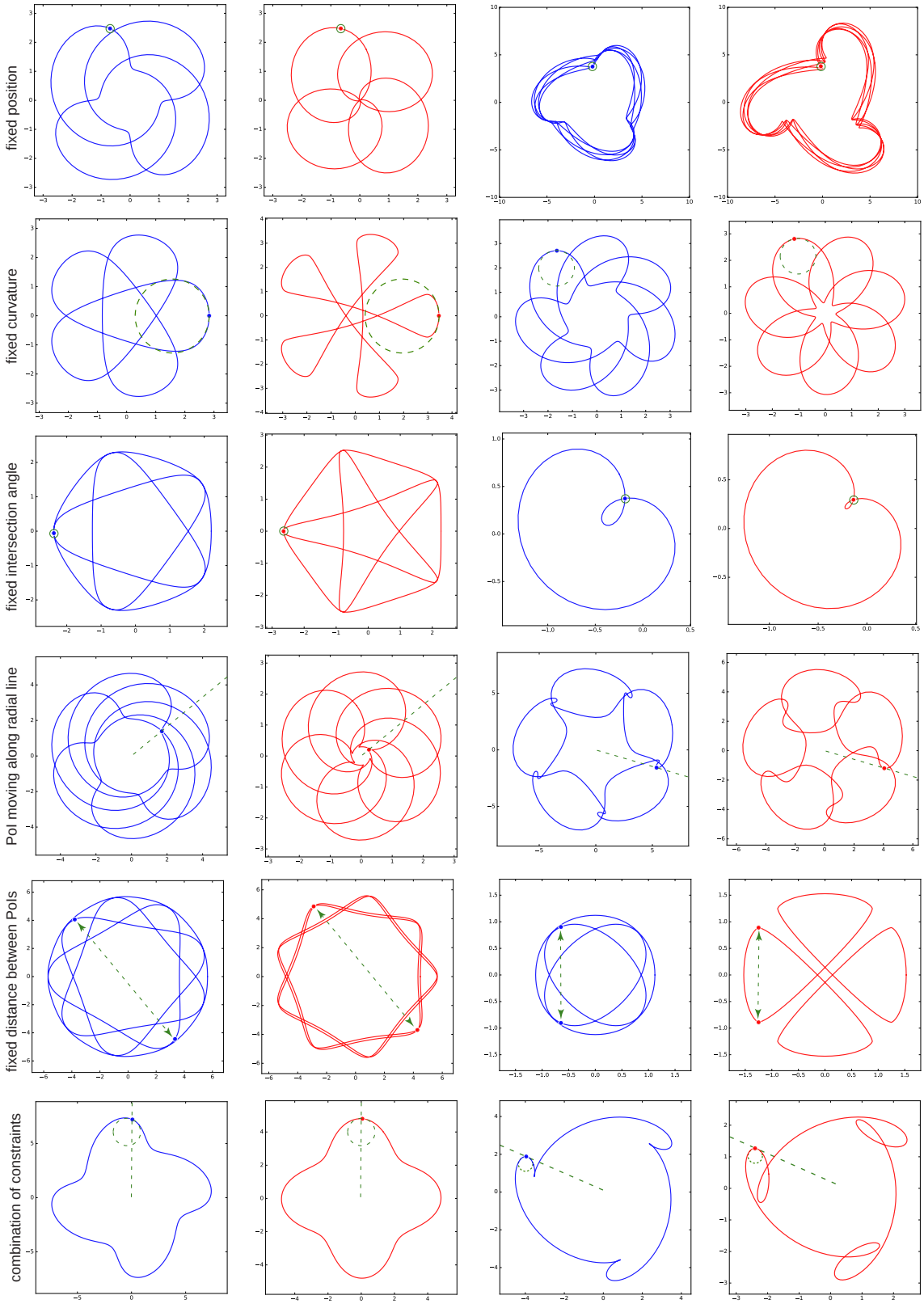

Figure 10: Examples of constrained variations obtained with our system (original curves in blue, user-selected curves after exploration in red). Each row represents a choice of visual constraint. Please refer to the text for discussion.

Table 1: Mechanisms implemented in our system.

Name	Nb. of 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 held 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 Figure 11):

- the elliptic Spirograph, an easily fabricable two-parts mechanism that allowed a quick validation of the first invariants;
- the Hoot-Nanny, which demonstrates our ability to manage mechanisms 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

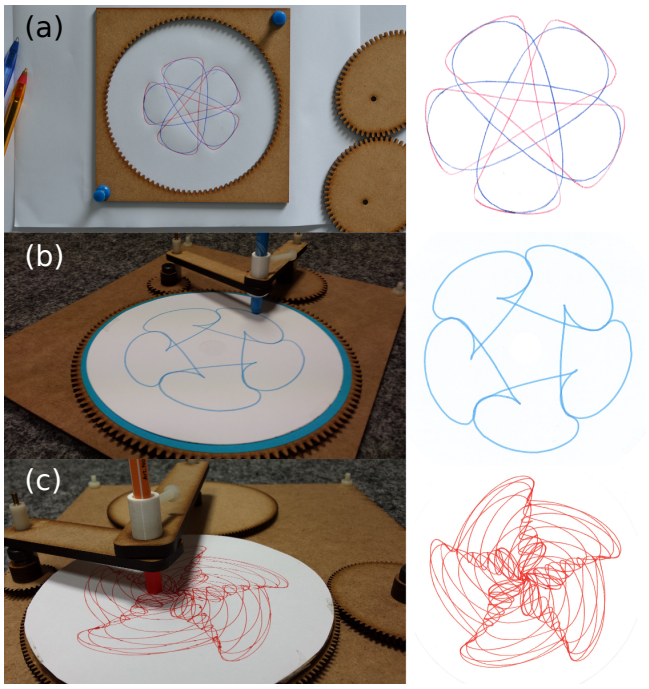

Figure 11: Examples of fabricated prototypes: (a) Elliptic Spirograph with two curves drawn using a different elliptic gear; (b) and (c) Instances of the Hoot-Nanny. Drawn patterns on the right.

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.

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), for both circular and elliptical gears. The latter, which is less common, was derived from a method by Bair [2002].

Pictures of some of the fabricated examples are given Figure 1 and 11. Demonstration of their usage is given in the main supplementary video.

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

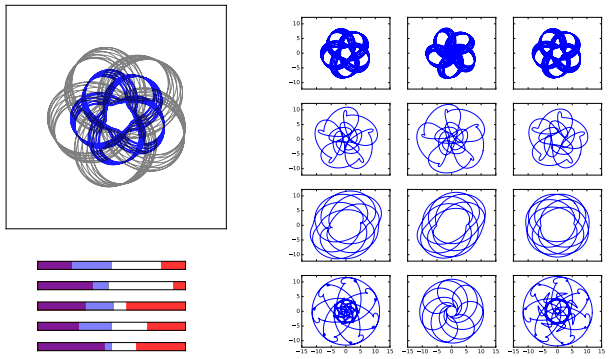

Figure 12: Left: interface for a subtask of the user study (target pattern in grey). Right: summary sheet presented to the user in order to rate the results (each column is respectively the target pattern, and results of subtask 1 and 2 in an arbitrary order).

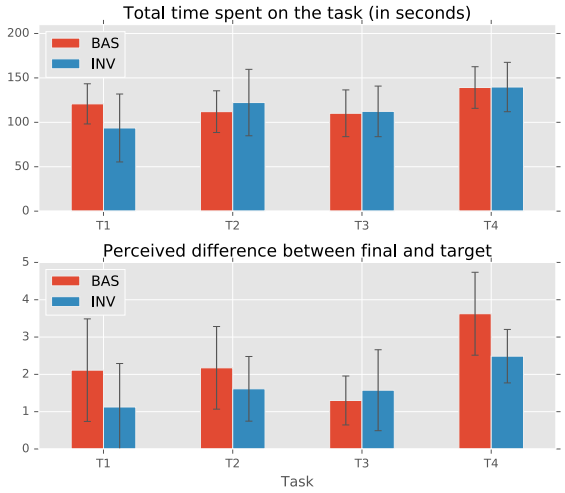

Figure 13: User study results (mean and standard deviation bars). "BAS" and "INV" respectively denote the base and invariant-space parameterizations.

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 Figure 12 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 Figure 12 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 Figure 13. With comparable times, candidates were in most cases able to reach a final result perceptually closer to the 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:

- While robust to some sketch defaults (disconnected strokes, noise), the curve metric used for pattern retrieval does not necessarily reflect perceived proximity between drawings and does not allow an efficient indexing of the search space. An improvement would be to train a feature-based curve metric with a perceptual study, as proposed by Coros et al. [2013], while trying to preserve the current versatility.
- 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. Building on the experience of previously fabricated drawing machines, further automation of the 3D model generation could be achieved.

The specific application domain presented in this paper, while interesting from an educational and artistic point of view, is arguably limited in terms of practical value. The core concepts of our method, however, are not bound to drawing machines. They 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 a measurable (and desirable) regularity. For instance, dynamic reparameterization could be used to interactively constrain the motion of mechanical characters [Coros et al. 2013] as well as more generic linkages [Bächer et al. 2015] to explore the different ways an end-effector can reach a specific point in space, such as a character kicking a ball or laying a kiss. Designing cyclic motions is also relevant in more 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.

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.

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

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

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

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

Hichem Barki, Lincong Fang, Dominique Michelucci, and Sebti Foufou. 2016. Reparameterization reduces irreducible geometric constraint systems. *CAD Computer Aided Design* 70 (2016), 182–192. https://doi.org/10.1016/j.cad.2015.07.011

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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