Computational Design of Wind-up Toys

PENG SONG, University of Science and Technology of China
XIAOFEI WANG, University of Science and Technology of China
XIAO TANG, University of Science and Technology of China
CHI-WING FU, The Chinese University of Hong Kong
HONGFEI XU, University of Science and Technology of China
LIGANG LIU*, University of Science and Technology of China
NILOY J. MITRA, University College London

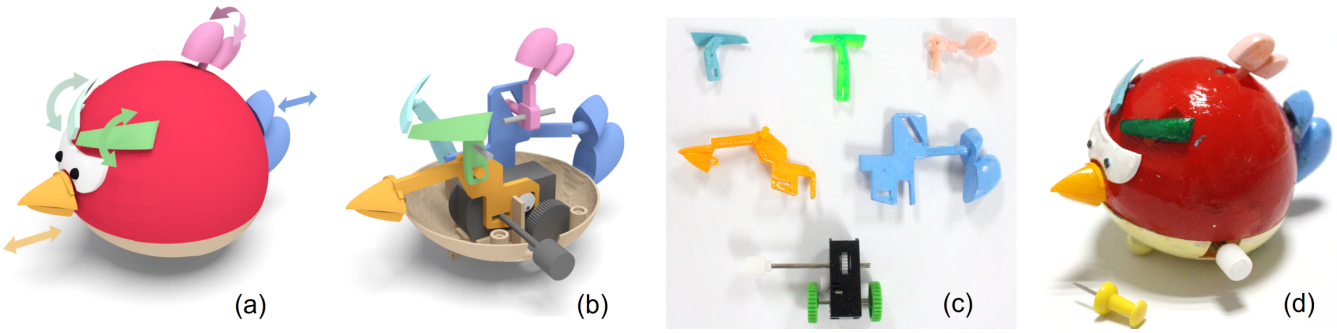


Fig. 1. (a) Input model with user-provided parts geometry and motion; (b) wind-up mechanism constructed by our method; (c) 3D-printed parts (top two rows) and spring motor (bottom); and (d) assembled toy (9.2×6.7×6.4cm^3; and 16.1g shell + 7.9g motor + 7.3g mechanism).

Wind-up toys are mechanical assemblies that perform intriguing motions driven by a simple spring motor. Due to the limited motor force and small body size, wind-up toys often employ higher pair joints of less frictional contacts and connector parts of nontrivial shapes to transfer motions. These unique characteristics make them hard to design and fabricate as compared to other automata. This paper presents a computational system to aid the design of wind-up toys, focusing on constructing a compact internal wind-up mechanism to realize user-requested part motions. Our key contributions include an analytical modeling of a wide variety of elemental mechanisms found in common wind-up toys, including their geometry and kinematics, conceptual design of wind-up mechanisms by computing motion transfer trees to realize the requested part motions, automatic construction of wind-up mechanisms by connecting multiple elemental mechanisms, and an optimization on the part and joint geometry with an objective of compacting the mechanism, reducing its weight, and avoiding collision. We use our system to design wind-up toys of various forms, fabricate a number of them using 3D printing, and show the functionality of various results.

CCS Concepts: • **Computing methodologies** → *Shape modeling*; • **Applied computing** → *Computer-aided manufacturing*;

Additional Key Words and Phrases: toy modeling, mechanical assembly, wind-up mechanism, kinematic pair, computational design, 3D printing

ACM Reference Format:
Peng Song, Xiaofei Wang, Xiao Tang, Chi-Wing Fu, Hongfei Xu, Ligang Liu, and Niloy J. Mitra. 2017. Computational Design of Wind-up Toys. *ACM Trans. Graph.* 36, 6, Article 238 (November 2017), 13 pages.
https://doi.org/10.1145/3130800.3130808

*Corresponding author: lgliu@ustc.edu.cn (Ligang Liu)

0730-0301/2017/11-ART238 $15.00
https://doi.org/10.1145/3130800.3130808

1 INTRODUCTION

Wind-up toys have a long history dating back to an early design of a wind-up lion by Leonardo da Vinci for welcoming Louis XII. Today they come as lightweight toys with compact internal mechanical assemblies that are powered by clockwork motors attached to a spring key. Once such a key is rotated to tighten the spring and released, the stored potential energy drives the toy's internal mechanical parts, which in turn drive the end-effector parts of the toy to perform intriguing motion(s). Examples include moving along a designated path, swinging body parts, or combinations of multiple motions over time and space. Fig. 2(a) shows a typical wind-up toy that moves along a wobbling path while swinging its tail.

Compared to mechanical assemblies in previous research [Bächer et al. 2015; Ceylan et al. 2013; Coros et al. 2013; Thomaszewski et al. 2014; Zhu et al. 2012], wind-up toys have several distinctive characteristics. First, they are driven by a simple spring motor that can only give out limited energy and torque. Second, their parts usually perform simple periodic motions that are only roughly specified. Third, they often have a small lightweight body, particularly for those that perform locomotion. Lastly, they are usually presented as

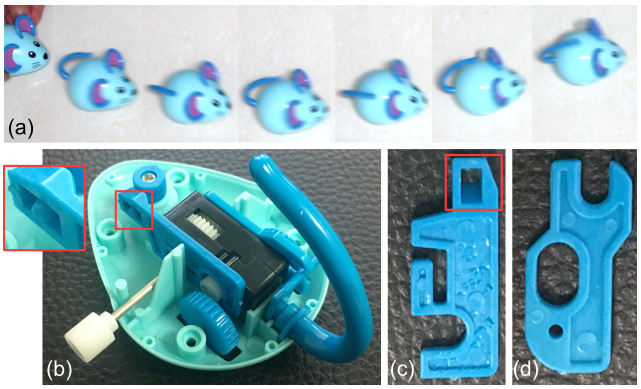


Fig. 2. An example wind-up toy (see its live motion in supp. video). (a) It moves along a curved path while swinging its tail; (b) the internal wind-up mechanism and motor; and (c&d) nontrivial parts for motion transfer, e.g., from a continuous rotation (spring motor) to a periodic swing (tail).

cartoon shapes with the wind-up mechanism and motor enclosed inside. Considering these characteristics, wind-up toys call for compact internal mechanisms that incur less frictional force, reduce avoidable energy loss, and offer richer motion variants.

In mechanics, a kinematic pair is a joint between two rigid components that keep them in contact under *relative* motion. In general, there are two families of kinematic pairs based on the type of contacts [Ambekar 2007]: (i) *lower pairs* (e.g., linkages, pulley and belt) with surface contacts, and (ii) *higher pairs* (e.g., gears, cam and follower) with point, line, or curve contacts; see the red boxes in Fig. 2(b & c) for an example. Compared with lower pair joints, higher pair joints have larger contact stress due to smaller contact areas, but enjoy smaller friction due to low relative sliding between the contact surfaces and larger diversity of motions that the associated parts can perform. However, motions performed with the higher pair joints are usually not as precise as those with the lower pair joints [Ambekar 2007]. Considering these features, wind-up toys heavily employ higher pair joints to transfer motion.

In this paper, we present *an interactive system to aid the design and fabrication of wind-up toys*, motivated by the use of 3D printing for making personalized wind-up toy designs. Particularly, we focus on automated construction of internal wind-up mechanisms to realize user-requested part motions. This problem involves several unique challenges that were unexplored in previous works. First, unlike the hand-operated cranks and electric motors employed in previous works, common spring motors in wind-up toys provide only small torque. To drive the various toy parts, such torque has to overcome both friction (contacts) and weight (parts). Second, wind-up toys employ a rich variety of higher pair joints for transferring different types of motions; many of them have not been explored. Lastly, connector parts inside wind-up toys have nontrivial shapes (see Fig. 2(c & d)); they are specially designed for transferring motions while being able to fit inside the toy with minimized weight.

To meet these challenges, our key idea is to construct and optimize a compact and lightweight wind-up mechanism to progressively transfer motion from the internal spring motor to toy parts through higher pair joints. Our work has the following contributions:

- First, by exploring common wind-up toys, we constructed a table of elemental mechanisms that involve higher pair joints, and categorized a wide range of fundamental motion transfer patterns. Then, for each elemental mechanism, we modeled and parameterized the geometry of the associated joints, parts, and support structures, and formulated analytical equations to compute the kinematics of parts from the associated geometric parameters.
- Next, to produce conceptual designs, we developed computational methods to automatically enumerate all possible valid mechanism designs composed from the elemental mechanisms, and to select candidate designs with fewer mechanical parts and shorter kinematic chains; this helps reduce the mechanism's complexity and weight, as well as the friction involved in the mechanism.
- Third, we turned each conceptual candidate design into a working mechanism with full geometry by connecting the involved elemental mechanisms from the driving source (usually composed of a round cam) to the toy's end-effector parts recursively.
- Lastly, we devised an optimization model to maximize the similarity between the user-requested part motions and resulting toy motions, to compact the wind-up mechanism, and to minimize its weight, using a coarse-to-fine optimization model. Such optimization also aims to keep the mechanism inside the toy's body without self-collision or collision with the spring motor and body shell, for both static and dynamic conditions.

We implemented a prototype system for our method, and validated it by designing and fabricating wind-up toys of various forms using 3D printing, and then showing the functionality of these results; see Fig. 1 for one of them. Moreover, we recruited several users (without background knowledge in mechanical designs) to use our system. Results show that all of them can make use of our system to design their own wind-up toys.

2 RELATED WORK

Wind-up mechanism, also known as clockwork mechanism [Trotoys 2016; Woodford 2016], generally employs a winding key for one to deposit energy in an internal metal spring. Once released, the spring triggers the attached shaft to rotate and drive the motions of the toy parts through the various mechanisms inside the toy. Although hugely popular as lightweight small toys, to the best of our knowledge, this is the first work that presents a computational system on constructing mechanisms in wind-up toys.

Mechanism modeling assumes that the mechanism already exists and considers the geometry and motion of the participating mechanical parts. For example, Mitra et al. [2010] visualizes the motion of a given mechanism by inferring the motions of individual parts and their interactions based on the geometry of the parts, while others focus on modeling 2D mechanisms for 3D fabrication [Hergel and Lefebvre 2015], reconstructing existing mechanisms from multiple images [Xu et al. 2016], or creating joints in mechanical objects [Ureta et al. 2016].

Mechanism design involves a conceptual design stage that describes the input motion and target output motion, identifies the type of mechanical parts, and extracts their associated parameters for realizing the user-specified tasks. Rather than requiring expert designers to manually design such mechanisms, researchers have

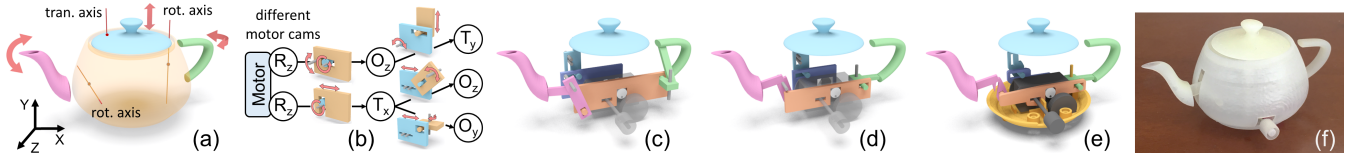


Fig. 3. Overview of our system. (a) Segmented Teapot model (i.e., body, handle, spout, and lid) with user-provided motion on each end-effector part (motion arrows and axes colored in red); (b) our system first constructs motion transfer trees to guide the selection of elemental mechanisms; (c) then, it orientates and connects elemental mechanisms to form an initial wind-up mechanism, (d) optimizes the joints and parts in each elemental mechanism with a kinematic analysis, subject to fabrication and collision constraints, and (e) post-processes the toy to refine parts boundary shape, and create the body shell and support structures for fabrication. (f) The 3D-printed wind-up toy result.

attempted to automate the process by identifying a finite set of basic mechanisms as building blocks for constructing abstract representations of complex mechanisms. For example, Chiou and Sridhar [1999] developed a matrix representation scheme to automate the conceptual design process by decomposing a given task into simpler sub-tasks and matching each sub-task with predefined kinematic building blocks that transform basic motions. Han et al. [2006] abstracted the underlying design concepts in existing mechanisms, and developed a case-based framework to reuse prior design concepts. Once a conceptual design is available, the next stage is to find the associated model parameters and spatial layout of the mechanical parts. This is generally achieved by identifying the geometric constraints among the parts (e.g., mate and align) and computing the parts parameters and layout accordingly [Li et al. 2002; Peng et al. 2006]. In this work, we also design with modular mechanisms, but we focus on compact wind-up mechanisms with higher pair joints. In short, we enumerated a variety of elemental mechanisms for wind-up toys, classified them by their motion transfer patterns, modeled their kinematics and connections, and devised methods to parameterize and optimize their joints and layouts for producing working wind-up mechanisms. This is a complete system that has not been developed in previous works.

Fabricating mechanism designs has been heavily investigated recently in computer graphics literature to facilitate design of personalized mechanical assemblies. Zhu et al. [2012] created animated toys by computing mechanical parts that perform linear and circular motions driven by cam and crank-sliders that are arranged in a box underneath the toy display. Coros et al. [2013] used parameterized mechanisms composed of gears and linkages to drive more complex motions of a mechanical character, where the end effectors can display 2D motion along user-sketched curves. Ceylan et al. [2013] arranged specialized mechanical oscillators composed of gears and linkages at joints of a humanoid character, and connected them through belts and pulleys to drive the character's limbs based on an input mocap sequence. More recently, Thomaszewski et al. [2014] and Bächer et al. [2015] presented interactive methods for users to edit planar linkage-based systems, while safeguarding the edits against various degeneracies, while Bharaj et al. [2015] and Megaro et al. [2015] presented computational methods for designing legged robots and characters that can walk on the ground driven by off-the-shelf servo motors installed at the limbs.

Similar to [Zhu et al. 2012] and [Coros et al. 2013], we also construct mechanisms by connecting a finite set of parameterized basic mechanisms as building blocks. However, this work differs from them in terms of the problem formulation, design requirements, and building blocks. First, we focus on designing wind-up toys, where the spring motor can only deliver limited force, and the mechanism has to be kept inside a small toy body. Second, our problem requires the mechanism to involve short kinematic chains with fewer parts, and be sufficiently compact. Prior works do not have these requirements. Lastly, prior works reuse common building blocks such as gears and linkages, while this work enumerates eleven elemental mechanisms as building blocks, mostly with higher pair joints.

Assembly sequences focus on computational methods to support non-expert users to design and fabricate assemblies [Schulz et al. 2014], for examples, assembly-free articulated models [Bächer et al. 2012; Calì et al. 2012], works-like prototypes [Koo et al. 2014], objects and furniture design [Li et al. 2015; Zhou et al. 2014], interlocking puzzles [Song et al. 2012], twisty puzzles [Sun and Zheng 2015], and Iris folding [Igarashi et al. 2016]. Our work also indirectly contains an assembly component, where we mainly consider the layout and geometry of parts in a small body subject to kinematics and weight, rather than the assembly ordering process.

3 OVERVIEW

Our system takes the following inputs: (i) a 3D mesh model that has been pre-segmented into logical parts (see Fig. 3(a)); these segmented parts are slightly moved away from the toy body to help avoid collisions with the toy body when the parts perform motion; (ii) user-prescribed motion (an axis and a range) for each segmented toy part (shown with red axes and arrows in Fig. 3(a)); see Table 1 for the part motions supported by our system; and (iii) the pose of the motor; our system puts it near the bottom of the 3D model as its initial pose, but users may reposition it on demand.

Before presenting our system, we list the high-level requirements identified by studying several existing wind-up toys in the market:

(i) They should have compact and lightweight wind-up mechanisms that can be driven by small spring motors.

Table 1. Part motions supported by our system.

Part motion	Symbol	Axis	Trajectory	Example action
rotation (**R**)		rotation axis	circular (2D)	cam rotation; wheeling
oscillating (**O**)		rotation axis	arc (2D)	swing head, tail; wave wing, arm
translation (**T**)		translation direction	line (2D)	push and pull a lid; move beak front and back
oscillating & translation (**OT**)		rotation axis	closed curve (2D)	a walking step (for leg)

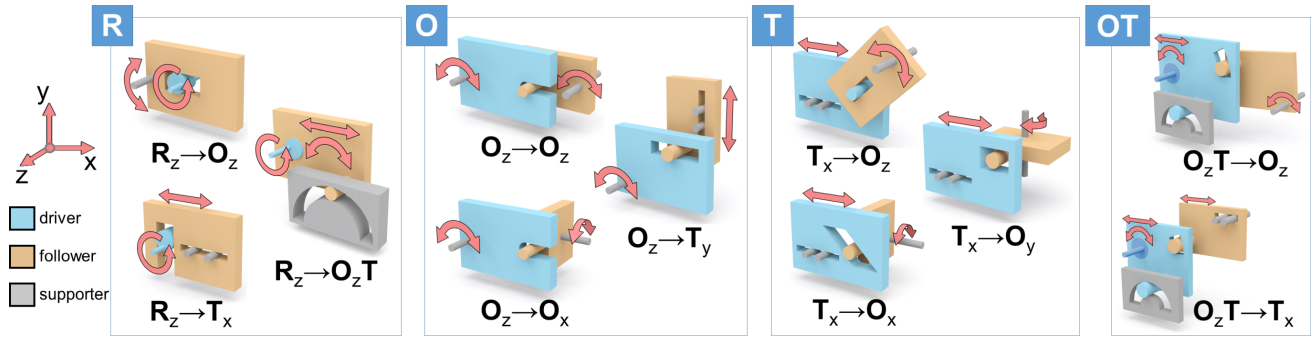


Fig. 4. Elemental mechanisms we modeled in our system. Each mechanism includes a driver (blue), a follower (brown), and one or more fixed supporters (grey), for transferring motion from the driver to the follower; see the equation below each mechanism, and also the red arrows next to each driver and follower. We categorize the mechanisms into four groups (**R**, **O**, **T**, and **OT**) according to the motion of the driver.

(ii) They should use higher pair joints to shorten kinematic chains, compact the mechanism, and reduce avoidable energy loss.
(iii) The motion of external toy parts (end effectors) should roughly resemble the prescribed target motions.
(iv) All internal parts of the mechanism should be contained inside the toy body, and their motions should be collision-free.

To construct a wind-up mechanism from the given inputs, we progressively consider the followings in the mechanical design:

- *Abstract modeling.* We consider the *topology* of the mechanism. To this end, we build a table of elemental mechanisms to describe the connecting joints (mostly higher pair) and their kinematic properties, and construct *candidate conceptual designs* by selecting and connecting elemental mechanisms; see Fig. 3(b).
- *Geometry modeling.* We need to consider the geometry of the wind-up mechanism. To this end, from each candidate conceptual design, we initialize the geometry of mechanical parts and joints for the involved elemental mechanisms, and connect them to form an initial wind-up mechanism; see Fig. 3(c).
- *Optimization.* For further compacting the wind-up mechanism and maximizing the parts motion similarity, we optimize the geometry and layout of parts and joints over the parameter space of the associated elemental mechanisms; see Fig. 3(d).
- *Post-processing and fabrication.* Finally, we refine the boundary shape of each mechanical part to further compact the mechanism, construct a body shell, and divide it (if required), for accommodating the internal wind-up mechanism; see Fig. 3(e). Besides, we build support structures for holding the mechanical parts and motor inside the toy, 3D-print the parts and shell, and assemble them with the motor to create the final result; see Fig. 3(f).

4 MODEL ELEMENTAL MECHANISMS

By exploring the internal mechanisms in a large collection (around 40) of wind-up toys (see Fig. 5), we enumerate eleven elemental mechanisms (abbreviated as *eleMech*) that build up the wind-up mechanisms, each of which delivers a fundamental motion transfer task; see Fig. 4. Each eleMech has three kinds of parts: (i) a *driver*, which initializes an eleMech's motion, (ii) a *follower*, which reacts and moves accordingly, and (iii) *supporter(s)*, which are fixed structures that constrain the motion of driver and follower. As in Fig. 4 and throughout this paper, we consistently color them in blue, brown, and grey, respectively. Note that we categorize the eleven eleMechs into four groups: **R**, **O**, **T**, and **OT**, according to the motion type of the driver; see again Fig. 4. Moreover, a driver can directly be the motor cam (for group **R**), or it can be the follower of the parent eleMech, if we chain up two eleMechs together.

There are two kinds of joints between contacting parts in an eleMech: (i) *driver-follower joint* for motion transfer; and (ii) *support joint* between a supporter and its associated driver/follower. Below, we first present how we model an eleMech's geometry and kinematics, and then present how we connect two eleMechs.

EleMech geometry. We model the geometry of an eleMech using three sets of parameters (defined in the local space of the eleMech):

(i) *Parameters of joints* $\{J_{joint}\}$. Since there are nine unique types of joint geometry among the eleven eleMechs shown in Fig. 4, we define geometric parameters for each of them; see Fig. 6. Hence, given an eleMech, we enumerate the parameters of each of its joints as its first parameter set.
(ii) *Parameters for joints layout* $\{J_{layout}\}$. Besides $\{J_{joint}\}$, the positioning of joints over an eleMech's driver and follower also affects the kinematics. Such a layout is essentially 2D over the *major plane* of driver and follower, so we take the 2D orientation and position of each joint on driver's and follower's major plane as an eleMech's second parameter set.
(iii) *Parameters for fabrication* $\{J_{fab}\}$. The third parameter set includes the thickness of driver and follower, their boundary shape (initially modeled as a rectangular box with cut-away/protrusion for accommodating joints), and tolerance

Fig. 5. Some of the wind-up toys we purchased from the consumer market for exploring the internal wind-up mechanisms.

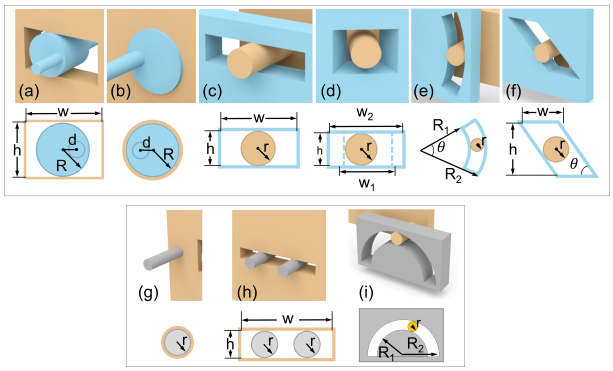


Fig. 6. Driver-follower joints (top) and support joints (bottom) in the eleMechs shown in Fig. 4. All are higher pair joints, except (b) and (g).

adapted onto the joints. Unlike the first two sets, these parameters affect mainly the fabrication but not the kinematics.

EleMech kinematics. The goal of modeling an eleMech's kinematics is to be able to compute the follower's pose from the driver's pose over time. We derive analytical equations to support the computation by taking $\{J_{joint}\}$ and $\{J_{layout}\}$ as inputs. Further, we derive inverse equations, again for each eleMech, to compute necessary joint parameters ($\{J_{joint}\}$) of its driver-follower joint given the motion range of the driver; see Fig. 7 for an example. Please see the supplementary material for the kinematic equations of each eleMech and the accompanying video for the simulated motions.

EleMech motion transfer. Each eleMech shown in Fig. 4 can transfer motion from one type (**R**, **O**, **T**, and **OT**) to another, possibly with a change in orientation. For example, eleMech $\mathbf{R_z}\rightarrow\mathbf{T_x}$ transfers a continuous rotation about the z axis to a periodic translation along the x axis; see the axes icon on the left side of Fig. 4. Obviously, by re-orientating this eleMech, we can obtain other motion transfer patterns such as $\mathbf{R_z}\rightarrow\mathbf{T_y}$ and $\mathbf{R_x}\rightarrow\mathbf{T_y}$. In fact, when we define the eleven eleMechs shown in Fig. 4, they are unique upon rotation, and we only consider axis-aligned motions for simplicity in early stages. Therefore, by re-orientating each eleMech in different ways, we can enumerate all possible axis-aligned motion transfer patterns supported by the eleven eleMechs; see Table 2. Note also that for now, we consider only axis-aligned orientation for motion transfer, non-axis-aligned motion will be considered later when we optimize a wind-up mechanism; see Section 6.

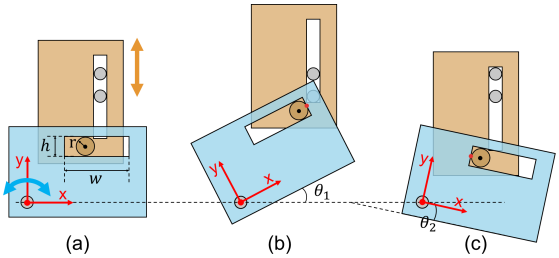


Fig. 7. Given the target motion range (θ_1 and θ_2 in (b & c)) of the driver, a predefined value for r in (a), and the joint layout $\{J_{layout}\}$, we can derive the joint geometry $\{J_{joint}\}$ of this driver-follower joint accordingly, i.e., h and w in (a). Please refer to the supplemental material for details.

Table 2. Motion transfer table. Rows denote driver motion and columns denote follower motion. Each ✓ in a cell denotes a motion transfer pattern supported by re-orientating an associated basic eleMech in Fig. 4 (black means no rotation, while red, green and blue mean a rotation about x-, y- and z-axis, respectively), and two ✓ in a cell means two successive rotations on the basic eleMech. Note that we only need to consider $\mathbf{R_z}$ but not $\mathbf{R_x}$ and $\mathbf{R_y}$ since only the motor produces a continuous rotation, and we fix the motor's rotational axis along z; likewise, we consider only $\mathbf{O_zT}$, since **OT** can only be produced by a continuous rotation, which must be $\mathbf{R_z}$.

Driv. \ Foll.	$\mathbf{R_z}$	$\mathbf{O_x}$	$\mathbf{O_y}$	$\mathbf{O_z}$	$\mathbf{T_x}$	$\mathbf{T_y}$	$\mathbf{T_z}$	$\mathbf{O_zT}$
$\mathbf{R_z}$				✓	✓	✓		✓
$\mathbf{O_x}$		✓	✓✓	✓		✓	✓✓	
$\mathbf{O_y}$		✓		✓✓	✓		✓✓	
$\mathbf{O_z}$		✓	✓	✓	✓	✓		
$\mathbf{T_x}$		✓	✓	✓				
$\mathbf{T_y}$		✓	✓	✓				
$\mathbf{T_z}$		✓	✓	✓				
$\mathbf{O_zT}$				✓	✓	✓		

EleMech connection. To link together two eleMechs, where one (parent) drives the other (child), the follower of the parent eleMech (denoted as F_{parent}) should merge with the driver of the child eleMech (denoted as D_{child}), so that the parent's driver can drive the merged part, which in turn drives the child's follower. To realize such a connection, the motion type of F_{parent} and D_{child} must be the same. If so, we can first re-orientate the child eleMech while fixing the parent in 3D, so that the major planes of D_{child} and F_{parent} align with each other. Next, we re-orientate and translate the child eleMech over the aligned major plane to further align D_{child}'s supporter(s) with F_{parent}'s supporter(s). As a result, we can merge D_{child} and F_{parent} into a single part, whose motion is constrained by common supporter(s); see Fig. 8 for three examples.

5 INITIALIZE WIND-UP MECHANISM

This section presents how we construct candidate conceptual designs that roughly match the user-prescribed motions at the end effectors (Section 5.1) and connect eleMechs into an initial wind-up

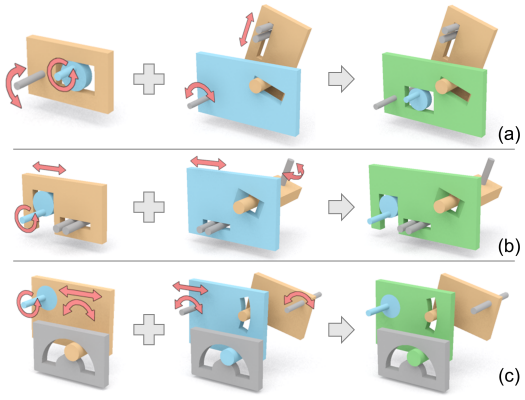
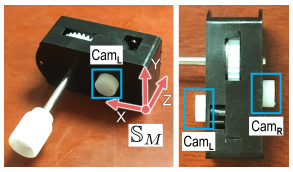


Fig. 8. Different cases of connecting two eleMechs (based on the type of supporters), we merge the follower of the parent (left) with the driver of the child (middle) into a single part in green (right).

mechanism for a candidate design (Section 5.2). Although we may produce non-axis-aligned motions by adjusting the joint orientation in an eleMech, we consider only axis-aligned motions in conceptual designs and wind-up mechanism initialization, since these two early stages mainly aim for a coarse initialization of the mechanism.

5.1 Generate Candidate Conceptual Designs

To represent a conceptual design, we use a *motion transfer tree* (see Fig. 3(b) for an example), where each node denotes a specific motion about a principle axis and each edge denotes an eleMech that transfers the motion from the associated parent node to the motion at the child node. For the wind-up motors we employed, there are two round cams, one on the left and one on the right (see inset figure), which serve as the primary drivers of the wind-up mechanism. Hence, we define a motion transfer tree, one for each cam, with a continuous rotation about z-axis ($\mathbf{R_z}$) as the motion at root nodes.



Given the prescribed end-effector motions and an initial pose of the spring motor in the object space of the toy (denoted as $\mathbb{S}_T$), we first transform each end-effector motion from $\mathbb{S}_T$ to the motor space (denoted as $\mathbb{S}_M$); see the inset figure above. If a transformed end-effector motion is not axis-aligned in $\mathbb{S}_M$, we find the closest principle axis direction in $\mathbb{S}_M$ as the axis of the end-effector motion when constructing a conceptual design. Hence, in the motion transfer tree, the internal and leaf nodes are always $\mathbf{O_x}$, $\mathbf{O_y}$, $\mathbf{O_z}$, $\mathbf{T_x}$, $\mathbf{T_y}$, $\mathbf{T_z}$, or $\mathbf{O_zT}$, where suffix $\{x, y, z\}$ indicates the associated principle axis in $\mathbb{S}_M$. Moreover, for each edge in the tree, the parent and child nodes should have a driver-follower relationship, following one of the motion transfer patterns in Table 2. Furthermore, to ensure short kinematic chains, we define $D_{\max}$ as the maximum tree depth; it is set to be 3 in all our experiments.

Generate conceptual designs. There are three major steps in generating candidate conceptual designs. In the first step, for each end effector, we exhaust all possible kinematic chains (with length $\leq D_{\max}$) that propagates $\mathbf{R_z}$ (motor) to the end effector's target axis-aligned motion in $\mathbb{S}_M$. To do so, we first check if "$\mathbf{R_z}\rightarrow$target motion" is already in Table 2; if so, we put it into a candidate set denoted as $\mathbf{C_i}$. Next, we treat Table 2 like a matrix, multiply it with itself, and look for candidate chains with length = 2 for "$\mathbf{R_z}\rightarrow\ldots\rightarrow$target motion"; see Fig. 9 for examples. We repeat the process until length $= D_{\max}$. Given $D_{\max} = 3$, the number of possible chains for each type of motion are 17 ($\mathbf{O_x}$), 14 ($\mathbf{O_y}$), 19 ($\mathbf{O_z}$), 10 ($\mathbf{T_x}$), 10 ($\mathbf{T_y}$), 6 ($\mathbf{T_z}$), are 1 ($\mathbf{O_zT}$), respectively. Please refer to the supplementary material for all these chains.

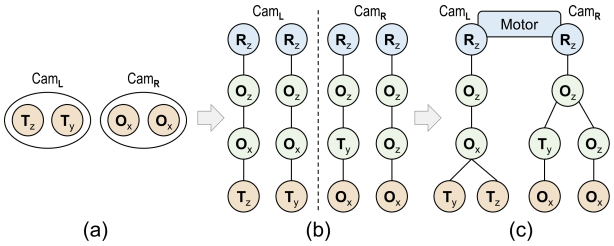


Fig. 10. A typical example: (a) end effectors assigned to left and right cams; (b) kinematic chains picked for each end effector; and (c) chains merged into a motion transfer tree for each cam.

Denoting E as the set of end effectors, the second step enumerates cases of partitioning E into two subsets, each to be driven by the left or right motor cam; see Fig. 10(a & b) for a simple example. For balanced mechanism layout, the partitioning should be more even, so we consider only the cases that the number of end effectors assigned to the two cams differs no more than two.

In the final step, we aim to pick one kinematic chain from each $\mathbf{C_i}$ (see Fig. 10(b)) and merge the picked chains into a low-complexity motion transfer tree for each cam with the following two strategies:

- First, given picked chains for different end effectors associated with the same cam, we can merge them into a tree at their k-th nodes, if all their nodes from the root to the k-th nodes are of the same motion type correspondingly. Hence, by iteratively trying different orders of merging the picked chains, we can form a tree with minimal edges for each cam; see Fig. 10(c).
- Denoting C_0 as the average number of chains among the $\mathbf{C_i}$'s and E_0 as the average number of end effectors assigned to a cam (roughly $|E|/2$), we will have $C_0^{E_0}$ different ways of picking chains from the $\mathbf{C_i}$'s for forming a motion transfer tree with minimal edges in a cam by using the first strategy. The second strategy is for selecting the best M left+right trees with the following criteria: (i) fewer eleMechs (say N) and (ii) shorter kinematic chains. Denoting L_i as the length of the picked chain from $\mathbf{C_i}$, we formulate the cost function as $N + \gamma\text{mean}(\{L_i\})$, where γ is a weight set as 2, and M set to be 10000. After we obtain the best M left+right trees for each case of partitioning E, we further use the cost function to compare candidate trees from different cases and output the best M left+right trees (among all cases) as the resulting candidate conceptual designs.

For the M conceptual designs, we explore them in the later stages of the pipeline in ascending order of their costs until we find a desirable wind-up mechanism. Hence, the number of actual conceptual designs that we need to explore is usually much lower than M; see Section 7 for the actual numbers involved in the experiments.

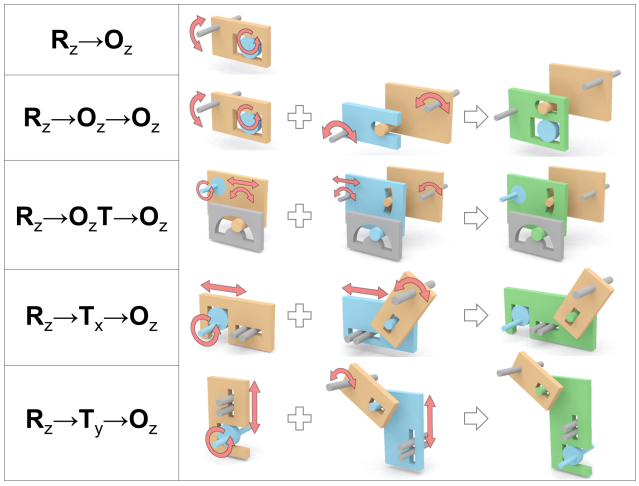


Fig. 9. All possible motion transfer chains with two or fewer eleMechs for achieving $\mathbf{R_z} \rightarrow \mathbf{O_z}$ (the axis-aligned target motion).

5.2 Initialize Mechanism Geometry

Next, we aim to initialize a wind-up mechanism for each candidate conceptual design, such that the mechanism is functioning but may not stay inside the toy body, and its end effectors may not follow the prescribed motions. Given a motion transfer tree associated with left or right cam, we first pick eleMechs associated with the tree edges, initialize them with default geometric parameters (due to space limits; please see supplemental material for details), and orientate them based on their motion patterns in Table 2. Then, we connect them from root until leaves in a breadth-first manner to form an initial geometry of the wind-up mechanism in $\mathbb{S}_M$.

Particularly, if a node has multiple child eleMechs, we have to merge the parent's follower with the drivers of all its children into a single merged part, so that the parent's follower can drive all the child eleMechs. In this way, the major plane of the parent's follower will contain the parent's driver-follower joint, its support joint(s), as well as the driver-follower joints of all its children; see Fig. 11(b-d). However, if the driver-follower joints of the children are too close to one another, the merged part will become structurally weak (see Fig. 11(b)), or even malfunction if the joints overlap. On the other hand, if the joints are too far apart (see Fig. 11(c)), the merged part will become excessively large. Hence, we compute the distance between joint pairs on the major plane of the merged part, and move the driver-follower joints away from any other if they are too close to any other joint; see Fig. 11(d). Clearly, repositioning a driver-follower joint on the major plane may lead to undesired motion or even faulty eleMechs; we leave it to the optimization stage for further adjusting the joint layout parameters $\{J_{layout}\}$ for achieving the target kinematics (see Section 6.2 for details).

Lastly, we employ the inverse equations described in Section 4 to deduce $\{J_{joint}\}$ of each eleMech from $\{J_{layout}\}$ given the driver's motion range, starting from the root. In addition, we fit a bounding box to enclose all the joints on the major plane of each merged part to initialize $\{J_{fab}\}$. However, it is important to note that these initializations are not final. Later in the optimization process, we will update $\{J_{layout}\}$ over eleMechs for meeting various criteria, including matching the prescribed end-effector motion. In addition, $\{J_{joint}\}$ and $\{J_{fab}\}$ will be recomputed accordingly.

6 OPTIMIZE WIND-UP MECHANISM

6.1 Formulate the Optimization

Configuration space. Our optimization model considers the following sets of independent variables, which together make up a configuration of the wind-up mechanism (denoted as g):

(i) joint layout parameters $\{J_{layout}\}$ of each eleMech; and
(ii) motor's pose (orientation and position) $\mathbf{P}_m$ in toy space $\mathbb{S}_T$.

We define the wind-up mechanism relative to the motor, so it transforms with the motor in $\mathbb{S}_T$ when we modify $\mathbf{P}_m$. Moreover, note that even though the initial wind-up mechanism only exhibits axis-aligned motions, the optimization, which modifies the joint orientation in $\{J_{layout}\}$ on a part's major plane, can adjust the relative orientation between connected eleMechs. As a result, we can allow end effectors to exhibit non-axis-aligned motions.

Cost function. We formulate the following cost function to characterize the compactness of a wind-up mechanism, to avoid collision

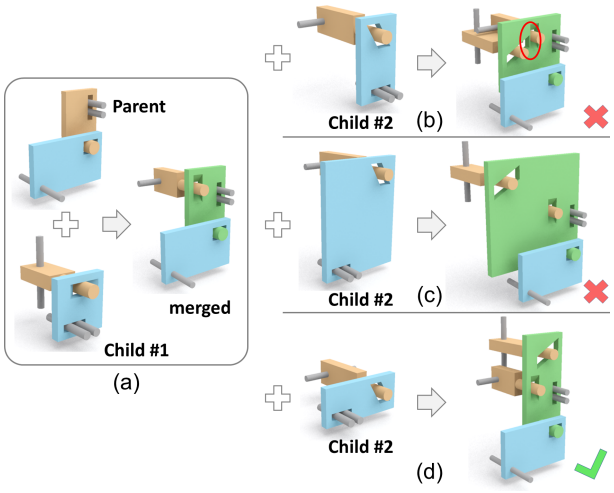


Fig. 11. Connect a parent eleMech to two child eleMechs. We first connect the parent eleMech with (a) child #1 eleMech, and then with (b-d) child #2 eleMech. Three $\{J_{layout}\}$ examples are shown: (b) the merged part is structurally weak (see the red circle), since the joints are too close to one another; (c) the merged part is large and heavy, since the joints are too far from one another; and (d) the merged part in this example is desired.

between mechanical parts and the body shell, and to make the end effectors follow the prescribed end-effector motions:

$$F(g) = \omega_1 F_1(g) + \omega_2 F_2(g) + \omega_3 F_3(g) + \omega_4 F_4(g) \qquad (1)$$

$$\begin{aligned}
\text{with} \quad F_1 &= \sum_{i=1}^{N} \mathrm{vol}(P_i)\,, \\
F_2 &= \max_{t\in T} \max_{i,j} \Omega_{\mathbf{S}}(J_{i,j})\,, \\
F_3 &= \sum_{e\in E} \left(\lambda |\cos^{-1}(\mathbf{n}(e)\cdot\bar{\mathbf{n}}(e))| + ||\mathbf{c}(e)-\bar{\mathbf{c}}(e)|| \right)\,, \\
\text{and} \quad F_4 &= \sum_{e\in E_{\mathrm{O}}} \lambda\left(|a(e)-\bar{a}(e)| + |b(e)-\bar{b}(e)| \right) \\
&\quad + \sum_{e\in E_{\mathrm{T}}} \left(|a(e)-\bar{a}(e)| + |b(e)-\bar{b}(e)| \right)\,,
\end{aligned}$$

where:

- F_1 is the approximate volume of all the mechanical parts, where P_i is the i-th part, typically the follower of the i-th eleMech (i=1..N); and $\mathrm{vol}(P_i)$ is the volume of the tightest bounding box that encloses the joints on P_i. Note that the number of internal mechanical (follower) parts equals the number of eleMechs.
- F_2 denotes the maximum signed distance among the signed distances ($\Omega_{\mathbf{S}}$) of all the joints from the toy's body shell $\mathbf{S}$ over time, where $J_{i,j}$ is the j-th joint on part P_i; and $\Omega_{\mathbf{S}}$ is the signed distance field, whose value is zero on $\mathbf{S}$, negative inside $\mathbf{S}$, and positive outside $\mathbf{S}$. Hence, to keep all the joints (and parts) inside the toy, all $\Omega_{\mathbf{S}}(J_{i,j})$ should be negative. Note that we have to simulate part motions in the mechanism over a full period with $|T|$=36 samples, in order to find all $J_{i,j}$ over time.
- F_3 measures the similarity between end-effector motion and prescribed motion for all end effectors (E). In the user input specification, for translation, $\bar{\mathbf{c}}$ is the starting point and $\bar{\mathbf{n}}$ is the translation vector, while for other motions ($\mathbf{R}$, $\mathbf{O}$ and $\mathbf{OT}$), $\bar{\mathbf{n}}$ is the rotational

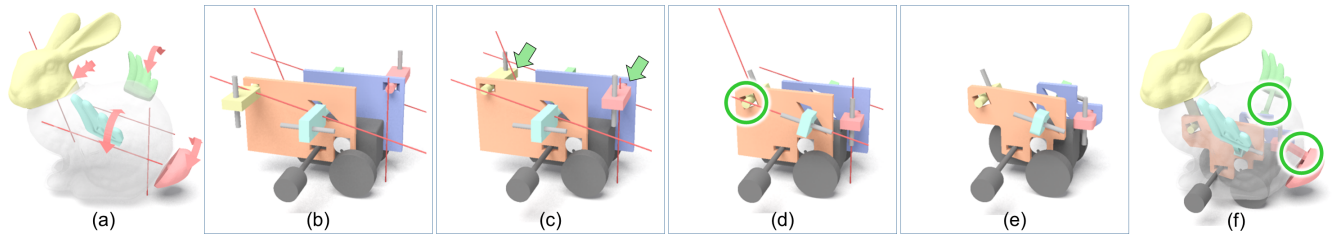


Fig. 12. (a) Inputs; (b) initial mechanism; (c) coarsely-optimized mechanism, where some parts relocate to different sides (see green arrows); (d) finely-optimized mechanism, where some joints (see the green circle) reorientate on the major plane to better match the prescribed motion; (e) boundary shapes of parts are refined; and (f) end-effector parts connect with associated external toy parts (see green circles).

axis and $\bar{\mathbf{c}}$ is the midpoint along the user-specified rotation axis line. On the other hand, $\mathbf{c}(e)$ is the motion center of end effector e and $\mathbf{n}(e)$ is its motion vector (translation vector or rotational axis) in the current wind-up mechanism. We measure angles obtained from $\cos^{-1}$ in radian, and use λ as a weight (set as 1.7 in all experiments) to combine angles and distances.

- F_4 measures the motion range similarity for end effectors ($E_{\mathbf{O}}$ and $E_{\mathbf{T}}$) that exhibit periodic motions, i.e., oscillating rotation (**O**) and translation (**T**), where $[a, b]$ is the motion range and $[\bar{a}, \bar{b}]$ is the target motion range. We express motion ranges using angles in radian for the case of **O** and using lengths for the case of **T**, so we again use λ to combine the two quantities.
- ω_1, ω_2, ω_3, and ω_4 are weights set as 0.13, 0.08, 3.0, and 0.5, respectively, in all our experiments.

The optimization goal is to search for a configuration (g) that minimizes F. Therefore, F_1 encourages a compact and lightweight wind-up mechanism, where the joints on each part are close to one another, thus leading to smaller mechanical parts that can be generated later in the post-processing stage. Then, F_2 helps to compact the whole mechanism and fit it inside the toy body without collision by encouraging the joints (and thus parts) to be far away from the shell. While both F_3 and F_4 aim to match the prescribed end-effector motions, F_3 focuses on the motion geometry and F_4 focuses on the motion range. Note also that we consider F_2 as a term in the minimization rather than as a constraint, since the joints in the initial mechanism may not fit inside the toy body, and formulating it a term also helps to compact the whole mechanism.

Constraints. We consider three types of constraints in the optimization to avoid collisions (i) between parts and motor over the motion simulation period (T), (ii) among different parts over T, and (iii) among different joints on the major plane of each part. In our implementation, we use oriented bounding boxes (OBB) to enclose each joint and each part (base geometry) for collision detection. Note that OBB is sufficient for detecting part collisions in our problem because we only need to care about the collision between parts that are not connected by joints. Parts connected with joints are ensured to be collision free (they only contact each other), since they form an eleMech with well-defined kinematics; see Section 4.

6.2 Solve the Optimization

Since we ignore the geometry of the body shell, external toy parts, and motor when initializing a wind-up mechanism, the initial mechanism may be too large to fit inside the toy body and it may collide with the motor when the wind-up toy is in motion. Moreover, the end-effector part (e.g., the follower of a leaf node) may be too far from its associated external toy part in $\mathbb{S}_T$ (see Fig. 12(b)), thus requiring a long physical connector to link them up for motion transfer. Hence, we first address these issues using a coarse optimization, and then further refine the mechanism with a fine optimization. Note that conventional gradient descent method is infeasible to solve this optimization problem, since we cannot derive an explicit analytical gradient of the cost function $F(g)$.

Positioning driver-follower joints. Before presenting the optimization methods, we first discuss the constraints in positioning driver-follower joints on major plane. Besides global space such as toy space $\mathbb{S}_T$ and motor space $\mathbb{S}_M$, we define local space $\mathbb{S}_P$ on each major plane in the wind-up mechanism. The x- and y-axis of $\mathbb{S}_P$ aligns with the major plane, and its origin locates at the supporter's center on the plane; see the axis labels (in red) in Figs. 7 and 13 for examples. Indeed, $\{J_{layout}\}$ is defined in this local space.

From the kinematics equations and virtual simulations we derived, we find that most eleMechs (see Fig. 4) allow us to position driver-follower joints freely in $\mathbb{S}_P$, as long as the joint does not collide with others. Of course, the follower's motion range may change, but it can always perform the same type of motion. However, for some eleMechs such as $\mathbf{O}_z \rightarrow \mathbf{O}_z$, moving its driver-follower joint may lead to a faulty eleMech; please refer to the supplementary material. In these cases, we precompute feasible parameter ranges for $\{J_{layout}\}$ to constrain the coarse and fine optimizations.

Coarse optimization. The goal of coarse optimization is to adjust the layout of parts in the mechanism by trying different ways of re-arranging each driver-follower joint on its major plane, for compacting the mechanism (F_1 and F_2 in the cost function; see Eq. (1)) and for making each end-effector part close to its target location next to the external toy part (F_3 in the cost function).

As a coarse optimization, it ignores the matching of motion range (F_4), and considers only a rough layout of joints represented by

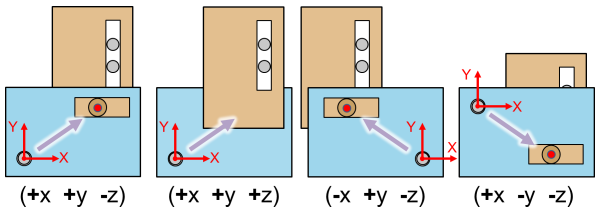


Fig. 13. Four of the eight choices in roughly positioning a driver-follower joint in the local space $\mathbb{S}_P$ of the driver's major plane.

Fig. 14. Some of the wind-up toys produced from our system; from left to right: Bunny, Horse, Mickey, Dragon, Tree, and House. From top to bottom: input models and prescribed toy part motions, the optimized wind-up mechanisms, and the associated motion transfer trees, where we color the nodes connected to the external toy parts in orange.

the signs of joint coordinates in the local space. In particular, we consider the sign of Z coordinates in $\mathbb{S}_P$, since repositioning joints along Z allows us to move the follower's major plane above or below the driver's major plane, so we can further compact the mechanism and better match the parts motion; see the left two cases in Fig. 13. Therefore, there are eight combinatorial choices for each joint, and 8^N choices altogether in the search space, since the number of driver-follower joints equals the number of eleMechs (N) in the wind-up mechanism. Note that coarse optimization is important in the sense that it allows topological modification, which is not considered in the fine optimization.

Such search space could still be too large for exploration, so we resort to a heuristic, motivated by the fact that in the initial mechanism, the major planes and part motions are all axis-aligned. Hence, we explore the signs of joints along each of the three dimensions in motor space ($\mathbb{S}_M$) one by one. First, for each joint, we identify the joint coordinate (X, Y, or Z) that aligns with the Z-axis of $\mathbb{S}_M$, so there are 2^N choices of signs. Since N is not that large ($N \leq 10$ in all our experiments), we simply exhaust all possible combinations of signs, initialize corresponding mechanisms, discard those that violate the optimization constraints, and rank the remaining cases based on the simplified cost function (without F_4). We then pick the best N_c configurations as candidates (N_c=10), repeat the same process for each of the other two dimensions in $\mathbb{S}_M$, and output the configuration with the minimal cost.

Fine optimization. Next, the fine optimization aims to further refine the output from coarse optimization for minimizing the whole cost function, particularly for better matching the prescribed end-effector motions and their ranges. There are $4N + 6$ variables, since each driver-follower joint has a 3D position and a 2D orientation on the major plane in $\mathbb{S}_P$, while the motor pose $\mathbf{P}_m$ includes a 3D position and a 3D orientation (in three euler angles).

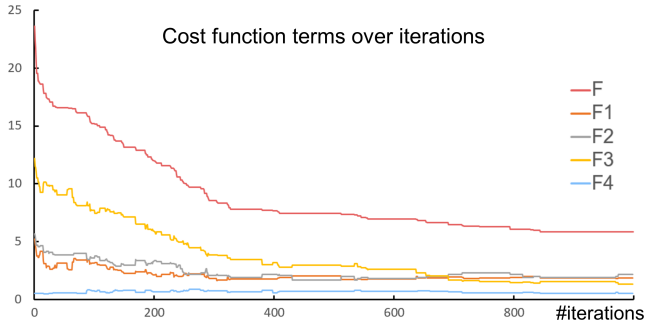


Fig. 15. Plot of cost function over iterations (see Eq. (1)) during the fine optimization process, for Bunny shown in Fig. 12(d).

We use simulated annealing to minimize the full cost function with two operators: (i) modify joint parameters; and (ii) modify motor pose. The annealing parameter is initialized to be 10, and decreases by 10% for every 50 iterations. After each move, we construct a mechanism from the new configuration, simulate its motion over a full simulation period (T), and evaluate it with the cost function to decide whether to accept the move or not. The optimization terminates when it cannot further reduce the cost, or has reached the maximum number of iterations (which is set as 1000); see Fig. 12(d) for the optimized result on Bunny and Fig. 15 for the corresponding plot of the cost function values in the optimization process.

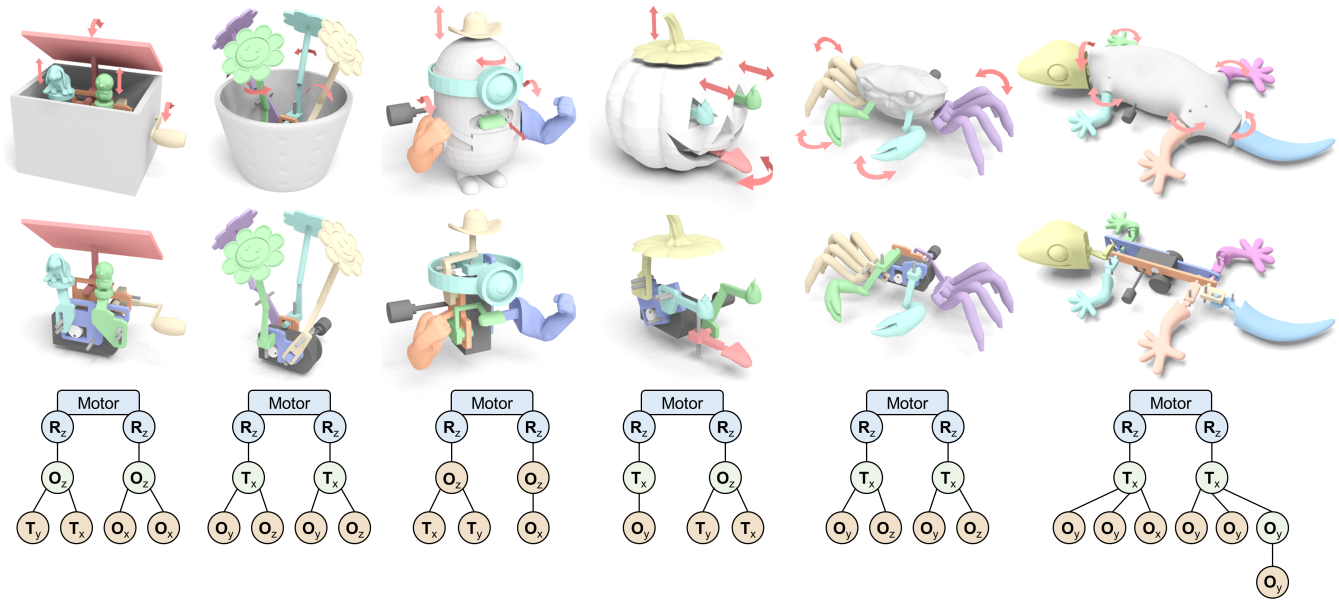


Fig. 16. Wind-up toys designed by the participants in the user evaluation and generated from our system accordingly; from left to right: Box, Flower Pot, Minion, Jack O'Lantern, Crab, and Gecko. From top to bottom: input models and user-prescribed toy part motions, the optimized wind-up mechanisms, and the associated motion transfer trees.

7 EXPERIMENTAL RESULTS

Mechanism finishing. After the optimization, we still have a few post-processing steps to complete the wind-up toy, among which the last three steps require certain amount of manual effort:

(i) First, our system refines the boundary of each part to compact the part and reduce its weight; see Fig. 17 for an example that illustrates the detail of the procedure. After that, we create tolerance around each joint on the part.
(ii) Second, our system connects each end effector to its associated external toy part using an elongated cuboid as the connector geometry; see the green circles in Fig. 12(f). However, using this simple strategy, the cuboid may collide with other mechanical parts. When a collision is detected by the system, the user has to manually modify the connector geometry to avoid the collision, e.g., using a U-shaped connector.
(iii) Third, the user manually divides the toy's body shell into two halves, and requests the system to create connectors between the supporters of each mechanical part and the shell. This is done similar to the way how end effectors are connected.

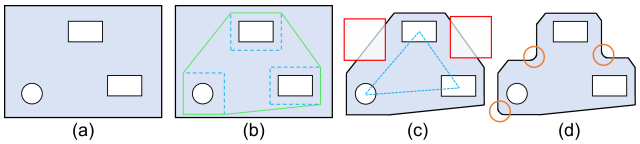


Fig. 17. Procedure to refine the boundary shape of mechanical parts: (a) given joints on major plane; (b) we compute a convex hull (in green) that encloses the joints with a small margin (in blue); (c) we then identify structural lines (in blue) in-between joints and crop the convex hull without cutting through the structural lines (in red); and (d) lastly, we round the corners of the shape (in orange).

(iv) Lastly, we 3D-print the refined mechanical parts and the body shell parts, and then manually paint the 3D-printed parts and assemble the parts with the motor into the target toy.

Results. We implemented our system in C++ and tested it on a desktop PC with a 3.4GHz CPU and 8GB memory. Fig. 14 showcases some of the results produced by our system. While most external toy parts connect to the leaf eleMechs in the motion transfer trees, our system also allows an external toy part to connect to intermediate nodes in the tree to help achieve even more compact mechanisms; see House in Fig. 14 for an example. Note that the phase of the end-effector parts can be controlled by adjusting the phase of the associated cam. Please watch the supp. video for the animations.

User evaluation. We recruited six participants (5 males and 1 female) aged 21 to 24. All of them had experience of playing with wind-up toys from the consumer market, and one of them has fabricated his own models using 3D printing. Before the evaluation started, we introduce to each participant the requirements on the input models and part motions supported by our system. Then, each of them searched for desired models by browsing photos in the Internet, and we downloaded/created the chosen 3D model for them. After that, we segmented each model into parts according to the participant's request, and the participant can then use our system to design a wind-up toy by specifying the end-effector motions.

Fig. 16 presents the six wind-up toys designed by the participants. The input models specified by them have a wide variety, including human-made object, cartoon character, fruit, and animal. After we presented the simulation of the designed toys to them, all of them felt that the mechanisms are impressive and the resulted toy motions are close to what they want, although two of them expected a larger motion range for the end effectors. Two of the participants also

Fig. 18. Four of the six wind-up toys (top) that we have fabricated and assembled from 3D-printed parts (bottom). From left to right: Bunny (13.0g shell + 7.8g motor + 9.0g mechanism), Horse (15.3g shell + 5.9g motor + 26.1g mechanism), Flower Pot (34.8g shell + 5.8g motor + 8.0g mechanism), and House (48.9g shell + 6.0g motor + 6.4g mechanism). See their statistics in Table 3.

pointed out that the parts motion can be more natural by better motion synchronization, e.g., the eyes of Jack O'Lantern, while three participants suggested that our system can further support more variety of toy locomotion, such as crawling for Crab.

Statistics. Table 3 summarizes the statistics of all the results presented in the paper; from top to bottom: Angry bird in Fig. 1, Teapot in Fig. 3, Bunny to House in Fig. 14, and then the six results designed by users in Fig. 16. The second to fifth columns from the left show basic information about the results. The number of toy part motions (or end effectors) in user inputs ranges from three to eight, while the wind-up toys produced by our system have five to ten eleMechs and three to six kinematic chains. The rightmost three columns in the table show some of the timing statistics. Overall,

Table 3. Statistics of our results. The columns from left to right refer to the name of the data sets, the number of end effector parts (motions), the number of chains in the motion transfer tree, maximum length among the chains, the number of eleMechs, time to generate the conceptual designs, and time to coarsely and finely optimize *a single* initialized mechanism.

	# Motion	# Chains	Max. Chain	# EleMechs	Concep. Designs (sec.)	Coarse Optim. (sec.)	Fine Optim. (sec.)
ANGRY BIRD	5	3	2	5	5.1	4.8	185.9
TEAPOT	3	3	2	5	0.1	4.3	155.1
BUNNY	4	4	2	6	1.5	8.6	154.1
HORSE	5	3	2	5	0.5	3.7	149.9
MICKEY	5	5	3	9	8.0	76.8	275.1
DRAGON	7	6	3	9	44.7	83.1	243.1
TREE	8	6	3	10	57.2	141.2	342.1
HOUSE	8	5	3	8	71.5	43.3	140.3
BOX	4	4	2	6	0.8	10.3	187.4
FLOWER POT	4	4	2	6	1.8	10.1	179.7
MINION	5	3	2	5	6.4	4.4	115.6
JACK O'LANTERN	3	3	2	5	0.1	3.9	140.3
CRAB	4	4	2	6	1.8	8.3	162.3
GECKO	6	6	3	9	23.9	80.2	283.7

the *average* time taken to generate a set of candidate conceptual designs, to coarsely optimize *a single* mechanism, and to finely optimize *a single* mechanism are ~16, ~35, and ~190 sec., respectively. For mechanism initialization, it can be done in the order of ten milliseconds, e.g., 40 millisec. for Teapot. This step has to be done very quickly, since every time the optimization process modifies a mechanism's configuration, it needs to initialize a new mechanism, simulate its kinematics, and check for collision, etc. Lastly, the total time taken to generate a desirable wind-up mechanism from the user inputs takes from 20 minutes to 6 hours for the results presented in Table 3. Overall, we found that the total time taken is roughly proportional to the number of explored conceptual designs (or the number of candidate mechanisms), which is 10 to 80 in all our experiments, depending on the complexity of the mechanism. Hence, the total time roughly equals to the number of explored conceptual designs times the time taken to coarsely and finely optimize one mechanism (see the rightmost two columns in Table 3). Note also that these timing numbers exclude the time taken to prepare and segment the input model, and the manual work involved in the mechanism finishing step during the post processing.

Fabrication. We fabricate six of our wind-up toy results using a high-resolution desktop SLA printer (printing volume: 130×130×180 mm^3, printing resolution: 0.1mm) with photosensitive resin material (density: 1.10-1.15 g/cm^3). It took around five hours to print all the parts of a toy, where most of the time was actually used to print the body shell parts. After 3D printing, we further paint the parts and shells with different colors. Figs. 1, 3, and 18 show the six wind-up toys that we have fabricated, from which we can see our constructed mechanisms (and their parts) look like those of wind-up toys in consumer market (Figs. 2 and 5). Please watch the supplementary video for the animations.

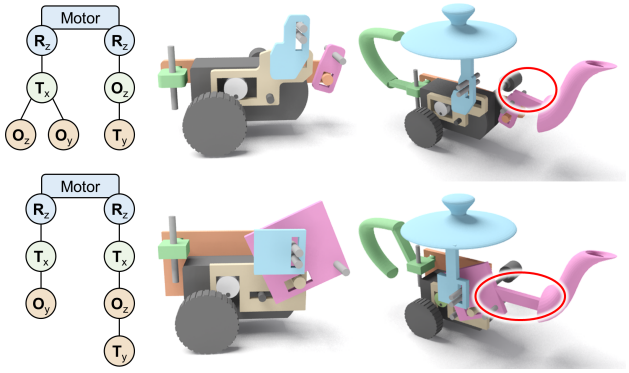


Fig. 19. Comparing mechanisms produced from our system (top) and from the baseline method (bottom). Although both results are functional, our optimized Teapot has shorter kinematic chains (top left), a more compact mechanism with smaller mechanical parts (top middle), and shorter connectors to join the end effectors with external parts (see the red circles).

Comparison with a baseline. We compare our method with a baseline method, which has the following two steps. First, the baseline method generates a conceptual design by randomly selecting kinematic chains from the enumerated candidates and merging them into a motion transfer tree; then, it initializes a wind-up mechanism simply by connecting the associated eleMechs following the transfer tree without further optimization. Such a result will likely have collision problems among the mechanical parts, body shell and motor. Hence, we repeat the above two steps and continue to generate more mechanisms until we find a valid one.

Fig. 19 shows the comparison result, from which we can see that our optimized Teapot has shorter kinematic chains and a more compact mechanism with smaller mechanical parts. We fabricate both results, and show that our optimized Teapot can perform motions for a longer duration (ours: 10 seconds vs baseline: 6 seconds) and move over a longer distance (see supplementary video), given roughly the same amount of winding on the same type of spring motor. This shows that our mechanism requires relatively smaller force and less energy to drive. Please note that although the parts have subtle geometry variations, the impact on final working time of the wind-up toys is significant.

8 CONCLUSION

We present computational methods to assist the design and fabrication of compact and lightweight wind-up toys from user-specified toy shape and motion. First, we identify eleven elemental mechanisms commonly found in wind-up toys, and model their geometry, kinematics, and connections with analytical equations. Second, we automatically construct conceptual designs represented as motion transfer trees and initialize mechanisms by connecting elemental mechanisms following the transfer trees. Finally, we iteratively optimize the geometry and layout of elemental mechanisms in a coarse-to-fine manner, so that the wind-up mechanism becomes more compact and its part motions resemble the user-specified motions. We have successfully designed mechanisms for a wide variety of wind-up toys, recruited six participants to try our system to design their own wind-up toys, and fabricated six of the results using 3D printing, demonstrating the functionality of the resulting toys.

Limitations and future work. First, we cannot guarantee a feasible mechanism for arbitrary user inputs, for example, when the input toy body is too small or too slim to accommodate the wind-up mechanism even after optimization. Second, we do not have an explicit force model that analyzes whether the torque/force of an existing wind-up motor is sufficient to drive our designed mechanism (with known fabrication material), and predicts frictional forces involved at the contacting joints. Third, our system only supports a single locomotion type (i.e., wheeling), and we plan to support more locomotion types such as walking by synchronizing motions of different end-effector parts (e.g., legs). Fourth, the motor cams employed in our mechanisms are round; we plan to optimize the cam profile, such that we can produce more interesting toy part motions. Fifth, our current conceptual design method is not scalable, say for a large number of end effector parts, due to the exponential complexity of the search space. Lastly, besides wind-up toys, we plan to explore the use of our methods on micro robots and mechanical toys that are driven with limited battery power.

ACKNOWLEDGMENTS

We thank the reviewers for the valuable comments, and Thingiverse for providing 3D models of Angry Bird, Horse, Dragon, Tree, Flower Pot, Minion, and Gecko. This work is supported in part by the National Natural Science Foundation of China (61403357, 61672482, 11626253), the One Hundred Talent Project of the Chinese Academy of Sciences, the Hong Kong RGC General Research Funding Project (14203416), the ERC Starting Grant SmartGeometry (StG-2013-335373), and gifts from Adobe.

REFERENCES

Ashok G. Ambekar. 2007. *Mechanism and Machine Theory.* Prentice-Hall of India Pvt.Ltd. 1004 pages.

Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. 2012. Fabricating Articulated Characters from Skinned Meshes. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012). Article 47.

Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing using Symbolic Kinematics. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article 99.

Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. 2015. Computational Design of Walking Automata. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 93–100.

Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 2012. 3D-Printing of Non-Assembly, Articulated Models. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article 130.

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). Article 186.

Shean-Juinn Chiou and Kota Sridhar. 1999. Automated Conceptual Design of Mechanisms. *Mechanism and Machine Theory* 34, 3 (1999), 467–495.

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. (SIGGRAPH)* 32, 4 (2013). Article 83.

Young-Hyun Han and Kunwoo Lee. 2006. A Case-based Framework for Reuse of Previous Design Concepts in Conceptual Synthesis of Mechanisms. *Computers in Industry* 57, 4 (2006), 305–318.

Jean Hergel and Sylvain Lefebvre. 2015. 3D Fabrication of 2D Mechanisms. *Computer Graphics Forum (Eurographics)* 34, 2 (2015), 229–238.

Yuki Igarashi, Takeo Igarashi, and Jun Mitani. 2016. Computational Design of Iris Folding Patterns. *Computational Visual Media* 2, 4 (2016), 321–327.

Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. 2014. Creating Works-Like Prototypes of Mechanical Objects. *ACM Trans. Graph. (SIGGRAPH Asia)* 33, 6 (2014). Article No. 217.

Honghua Li, Ruizhen Hu, Ibraheem Alhashim, and Hao Zhang. 2015. Foldabilizing Furniture. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article No. 90.

Yan-Tao Li, Shi-Min Hu, and Jia-Guang Sun. 2002. A Constructive Approach to Solving 3-D Geometric Constraint Systems using Dependence Analysis. *Computer-Aided Design* 34, 2 (2002), 97–108.

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-Printable Robotic Creatures. *ACM Trans. Graph. (SIGGRAPH Asia)* 34, 6 (2015). Article 216.

Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. 2010. Illustrating How Mechanical Assemblies Work. *ACM Trans. Graph. (SIGGRAPH)* 29, 4 (2010). Article 58.

Xiaobo Peng, Kunwoo Lee, and Liping Chen. 2006. A Geometric Constraint Solver for 3-D Assembly Modeling. *International Journal of Advanced Manufacturing Technology* 28, 5 (2006), 561–570.

Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and Fabrication by Example. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014). Article No. 62.

Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. 2012. Recursive Interlocking Puzzles. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article 128.

Timothy Sun and Changxi Zheng. 2015. Computational Design of Twisty Joints and Puzzles. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015). Article 101.

Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-Based Characters. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014). Article 64.

Trotoys. 2016. Wind Up Mechanics: How To Make a Wind-Up Toy Yourself. (2016). http://re.trotoys.com/article/wind-up-mechanics/.

Francisca Gil Ureta, Chelsea Tymms, and Denis Zorin. 2016. Interactive Modeling of Mechanical Objects. *Computer Graphics Forum (SGP)* 35, 5 (2016), 145–155.

Chris Woodford. 2016. Clockwork (windup) Mechanisms. (2016). http://www.explainthatstuff.com/how-clockwork-works.html.

Mingliang Xu, Mingyuan Li, Weiwei Xu, Zhigang Deng, Yin Yang, and Kun Zhou. 2016. Interactive Mechanism Modeling from Multi-view Images. *ACM Trans. Graph. (SIGGRAPH Asia)* 35, 6 (2016). To appear.

Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. 2014. Boxelization: Folding 3D Objects into Boxes. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014). Article 71.

Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-Guided Mechanical Toy Modeling. *ACM Trans. Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article 127.