Designing and Fabricating Mechanical Automata from Mocap Sequences

Duygu Ceylan[1] Wilmot Li[2] Niloy J. Mitra[3] Maneesh Agrawala[4] Mark Pauly[1]
[1]EPFL [2]Adobe Research [3]University College London [4]University of California Berkeley

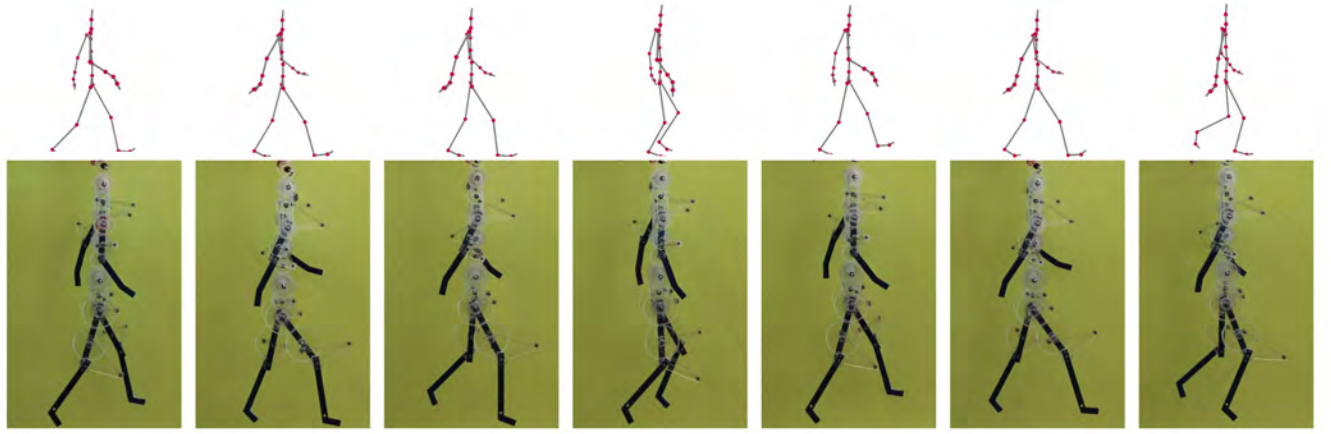


Figure 1: *We present a method for generating the design of a mechanical automaton (bottom row) that approximates an input motion sequence (top row). Our algorithm automatically determines the configuration, dimensions, and layout of all mechanical components.*

Abstract

Mechanical figures that mimic human motions continue to entertain us and capture our imagination. Creating such automata requires expertise in motion planning, knowledge of mechanism design, and familiarity with fabrication constraints. Thus, automaton design remains restricted to only a handful of experts. We propose an automatic algorithm that takes a motion sequence of a humanoid character and generates the design for a mechanical figure that approximates the input motion when driven with a single input crank. Our approach has two stages. The *motion approximation* stage computes a motion that approximates the input sequence as closely as possible while remaining compatible with the geometric and motion constraints of the mechanical parts in our design. Then, in the *layout* stage, we solve for the sizing parameters and spatial layout of all the elements, while respecting all fabrication and assembly constraints. We apply our algorithm on a range of input motions taken from motion capture databases. We also fabricate two of our designs to demonstrate the viability of our approach.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: mechanical assembly, humanoid motion, computational design, fabrication constraints, automaton

Links: DL PDF WEB VIDEO DATA CODE

1 Introduction

Mechanical automata are machines that use a combination of interconnected mechanical parts such as cranks, gears, and pulleys to convert a driving force into a specific target motion. Since antiquity, mechanists have created a wide variety of such machines for many different purposes (e.g., clocks, music boxes, fountains). Automata designed to look like human figures performing every-day actions like walking, waving, etc. are especially popular. Famous historical examples include Leonardo Da Vinci's life-sized armor-clad "robot" from 1495 that could sit, stand and move its arms, as well as the"Draughstman-Writer" of Henri Maillardet from the late 18th century, which was the primary inspiration for the automaton in Brian Selznick's book, *The Invention of Hugo Cabret.* Today, wind-up toys in the form of characters are the most common types of mechanical figures. Our longstanding fascination with humanoid automata likely stems from their ability to produce surprisingly complex, lifelike motions from simple input forces through purely mechanical means (see Figure 2).

Despite their widespread appeal, the design of automata is currently restricted to a very small group of experts. Consider the challenges involved in creating a mechanical figure that performs a specific target motion. First, a designer must choose a set of mechanical parts and decide how to connect the parts together to approximate the

Figure 2: *Examples of mechanical automata: Da Vinci's mechanical knight; Maillardet's writer; an aviation chronograph.*

prescribed motion. Creating this initial design typically requires extensive knowledge of different part types and how part interactions transfer movement through the assembly. Next, the designer must determine the appropriate part parameters (e.g., gear radius, crank length) to best match the target motion. This step may require simplifying the target motion so that it falls within the range of achievable motions for the set of parts. The final step is to arrange the parts spatially to create a valid physical realization of the automaton. Few people have the necessary skills and expertise to perform all of these design tasks.

In this work, we present an automated approach for generating mechanical figures that approximate a given target motion. The input is an animation sequence for an articulated 3D humanoid figure (e.g., from a motion-capture database) that specifies the target motion. Since automata typically perform repetitive actions, we assume that the input motions are roughly periodic (e.g., a walk cycle, an arm waving back and forth). We also assume that the motion of individual limbs are nearly planar. While these restrictions limit the space of motions that our approach can handle, we show that there are many human movements that meet these assumptions. For example, in most walking motions, the arms and legs move primarily in planes that are parallel to the saggital plane of the body. Given such a target motion, our system designs a mechanical automaton in which the moving limbs are treated as a set of rigid links connected by revolute joints and driven by gears, pulleys, and four-bar linkages (see Figures 1 and 6).

Our method has two main stages. First, in the *motion approximation* stage, we convert the input sequence into a mechanically realizable motion based on the constraints imposed by the types of joints (revolute) and parts (gears, pulleys and four-bar linkages) in our designs. More specifically, we convert the motion of each limb in the input figure to a set of oscillating bone rotations that can be reproduced by a kinematic chain of rigid links in our automaton design. Then, in the *layout* stage, we solve for the parameters (e.g., tooth counts, link lengths) and spatial layout of the mechanical parts to produce the desired motion of each link. We formulate a constrained optimization that tries to match both the target motion and the bone proportions of the input figure, while satisfying various physical and fabrication constraints.

Our system generates designs with the following characteristics:

Simple components. Our designs use only a few different types of components (rigid links, spur gears, bevel gears, pulleys, and four-bar linkages, see Figure 6) that are easy to fabricate or obtain. The use of four-bar linkages allows us to generate oscillating motions with continuous acceleration rather than abrupt changes in direction, which can put significant stress on the components. In our physical examples, we fabricate the links, spur gears and four-bar linkages with a laser cutter and obtain pre-made bevel gears and pulley components from a craft store.

Single input crank. Our mechanical figures are designed to be driven with a single input crank that rotates at a constant angular speed. The automaton converts the crank rotation to oscillatory movements that are propagated to the appropriate links in the figure. This propagation can generate oscillations with different phases and frequencies for different body parts, which greatly expands the expressive range of our designs. For example, in running motions, the oscillations of the upper and lower arms are typically out-of-phase.

Freestanding. Our system generates freestanding automata where the mechanisms that produce motion are attached directly to the links of the figure. In contrast, tethered automata often place mechanisms in a box below the figure and create motion by physically connecting moving parts in the box to the limbs above (e.g., [Zhu et al. 2012]). Freestanding designs enable a wider range of motion because limbs are not connected to an external structure (e.g., arms tethered from below cannot cross each other in arbitrary ways because their connectors may collide).

We have evaluated our framework using a variety of motion capture sequences from the CMU motion database [Carnegie Mellon University 2003]. We generated physically-realizable mechanical designs for several different motions and validated each of them in a kinematic simulator. In addition, we physically manufactured, assembled, and tested two of our designs (see Figure 1 and 7 and the supplementary video).

2 Related Work

Mechanism design. Mechanism design is traditionally divided into two phases. Creating a *conceptual design* involves identifying the types of mechanical parts that are capable of realizing desired motions. Previous work attempts to automate this task by decomposing specific functional goals into smaller subgoals and matching them to a database of common building block mechanisms ([Chiou and Sridhar 1999; Roy et al. 2001; Han and Lee 2006] and references therein). Since our work focuses on one specific type of motion conversion (i.e., converting the rotation from an input crank to rotations of bones), we do not require an explicit conceptual design phase. As mentioned earlier, our designs use a small set of predefined parts, which simplifies the fabrication process.

The second phase of mechanism design is *dimensional synthesis*, which determines the configuration parameters and spatial layout of the identified parts. A common approach is to identify the relevant spatial constraints between parts and then satisfy these constraints to compute part positions and orientations [Anantha et al. 1996; Kim et al. 2000]. Recent methods encode the mating and alignment relations between different parts of an assembly using a graph of geometric constraints [Peng et al. 2006; Haller et al. 2009]. We adopt a similar constraint-based approach to synthesize our mechanical designs. However, the complex, articulated human motions that we aim to reproduce impose a unique set of constraints on both the configuration parameters and spatial layout of our gears. In particular, our dimensional synthesis algorithm must generate a system of gears, pulleys and four-bar linkages that produce hierarchical bone rotations that match a given target motion.

In recent work, Zhu et al. [2012] present a system for producing mechanical assemblies that approximate user-specified input motions. Two key differences distinguish their problem setting from ours. First, they produce tethered assemblies where characters or objects (represented as rigid feature components) are driven from below by mechanisms hidden inside a box. We generate freestanding articulated automata, which imposes very different requirements on our dimensional synthesis algorithm. Furthermore, Zhu et al. focus on target motions that are different from ours. Their tethered designs drive each rigid feature component with one or two mechanical parts, which allows the component to perform a simple rotation, translation or certain combinations of two rotations/translations along orthogonal planes or axes. In contrast, our mechanisms reproduce articulated human motions with moving kinematic chains of rigid links, each of which can undergo complex motions based on the combined motions of its ancestor links in the skeleton.

In concurrent work, Coros et al. [2013] present an interactive system for designing animated mechanical characters. Their method supports complex target motions that are specified by sketching planar motion curves. For each kinematic chain of the given articulated character, a single actuation point is selected, and the motion is propagated to the rest of the chain by rigid connections provided as part of the input. In contrast, we present a method to realize the motion of each bone in a kinematic chain as oscillations with possibly

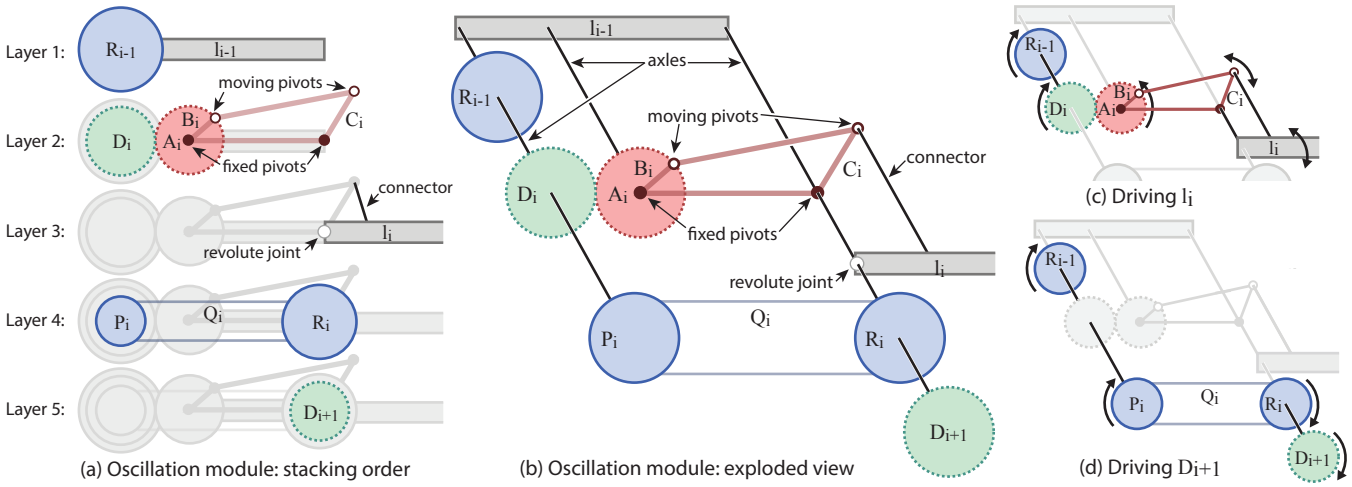


Figure 3: *Oscillation module. In our design, we drive the rigid links of the automaton with oscillation modules that consist of gears, pulleys, and four-bar linkages (a–b). These components are stacked onto axles that are connected to each link. We show the stacking order of the parts (a) starting from the bottom layer (Layer 1), as well as an exploded view (b) of the module that shows the axles and rigid connectors. The module converts an input rotation to an oscillating motion (b) and also propagates a uni-directional rotation to the next module (c).*

different phases and frequencies. Our approach is fully automatic consisting of a motion approximation stage to convert the input sequence into a mechanically realizable motion and a layout stage to optimize for the placement and parameters of the elements of the designed automaton.

Shape analysis. Researchers have investigated how to retarget input poses/motions using articulated motions by determining joint locations and types. For example, Xu et al. [2009] analyze input models based on local slippage analysis to assign appropriate joint types and support joint aware deformations. In the context of mapping human motion to humanoid robots, Pollard et al. [2002; 2003] impose joint angle and velocity limits to a captured human motion to match the less flexible skeleton of a robot. Mitra et al. [2010] analyze static 3D meshes of mechanical assemblies to understand their part interactions and resulting inter-part motion possibilities. Recently, Bacher et al. [2012] augment skinned meshes with 3D printable joints that do not require any assembly. Users can then pose these models into different configurations. In contrast, we start from an articulated 3D figure and focus on automatically generating a mechanical automaton that recreates an input motion sequence when driven by a single input crank.

Fabricatable models. The advent of accessible 3D fabrication techniques has led to recent work on various fabrication-aware modeling techniques, including optimizing geometry towards creating functionally valid furniture [Umetani et al. 2012]; improving stability and robustness of printed models with thin structures [Stava et al. 2012]; creating complex assemblies of planar intersecting pieces [Schwartzburg and Pauly 2013]; creating assembly-free printable articulated models [Calì et al. 2012]; and decomposing large models into smaller parts that can be printed and assembled [Luo et al. 2012]. These methods generate static models, while our goal is to create moving mechanical figures that recreate a target motion.

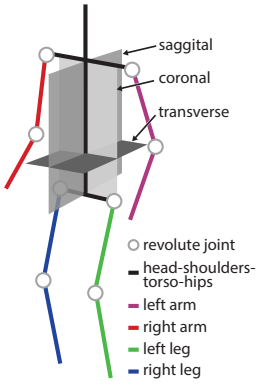
3 Generating mechanical designs

The input to our system is an animation sequence of an articulated 3D human figure. Since automata typically perform repetitive actions, we assume the input is a periodic motion where parts of the figure oscillate. From this animation, our system automatically produces a humanoid automaton design that uses a combination of rigid links, pulleys, gears and four-bar linkages to approximate the input motion. The first stage in our method is *motion approximation*, where we convert the input animation to a simpler motion that can be realized mechanically. In particular, we decompose the motion of the entire figure into the motions of kinematic chains (arms and legs) that can be approximated by planar oscillations of individual bones (see Section 3.2). Then, in the *layout* stage, we generate a physically valid layout of the mechanical components that realizes the simplified target motion and respects the proportions of the bones in the input figure. We first solve for the layout of each individual kinematic chain (Section 3.3). Then we generate additional gears and support structures to compose all the chains into a unified design that is driven by a single input motor (Section 3.4).

3.1 Mechanical automaton design

Before describing the details of our motion approximation and layout optimization, we first introduce the basic mechanical design that we use to generate a moving automaton.

Our design represents the bones of the input figure with rigid links. Since many human motions are characterized primarily by the movement of the limbs, we connect the arm and leg links with revolute joints to create points of articulation at the shoulders, elbows, hips and knees of the automaton. We orient these joints such that the motion of all the links within a limb lies in a single plane that is parallel to the saggital, coronal or transverse plane of the figure (see the inset figure).



While constraining limb motions to these orthogonal planes limits the range of movements that our mechanisms can achieve, this design makes it possible to transfer the rotation of a single input motor to the motion plane of each limb using standard bevel gears, which convert rotation across orthogonal planes. Furthermore, as our results show, there are many human movements in which the motion of the limbs is mostly parallel to one of these planes (e.g.,

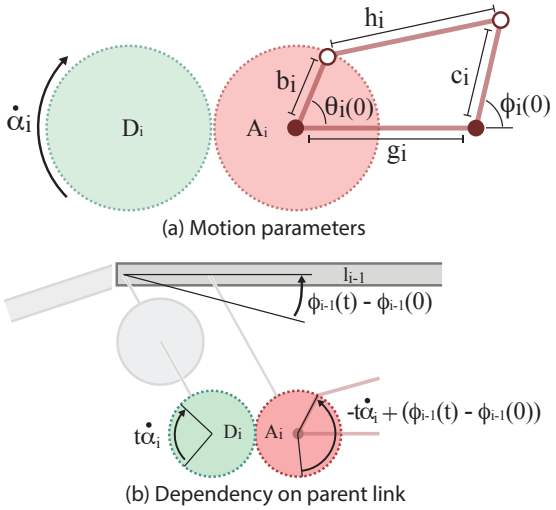


Figure 4: *Motion parameters. The output oscillation of a module is defined by a set of motion parameters (a). When we connect modules into a chain, the motion of one module depends on the motion of its parent link (b).*

walking, jumping jacks). Since we focus primarily on the motion of the limbs, we combine the head, shoulder, torso and hips to form a single rigid component.

To make the links move, we introduce an *oscillation module* that uses gears, a pulley, and a four-bar linkage to convert unidirectional rotation into an oscillating motion (see Figure 3a–b and Figure 6). The input to a module M_i is gear D_i. Applying a unidirectional rotation to D_i causes A_i to rotate, which drives the input crank B_i of a four-bar linkage. The linkage converts this input rotation into an oscillating rotation of its output crank C_i. The axles of D_i, A_i and the fixed pivot of C_i are all attached to link l_{i-1} while the moving pivot of C_i is connected rigidly to link l_i. Thus, the oscillation of C_i causes l_i to oscillate as well (Figure 3c). Rotating D_i also drives pulley wheel P_i, which is attached rigidly to the same axle as D_i. The pulley belt Q_i transfers the rotation of P_i to pulley wheel R_i. To prevent slipping between P_i, Q_i and R_i, we use pulley wheels and belts that have interlocking teeth. Connecting R_i rigidly to the same axle as the input gear D_{i+1} of the next module M_{i+1} propagates the motion to the next link in the chain (see Figure 3d). Thus, we can move an entire limb of the automaton by attaching a chain of oscillation modules to the rigid links and driving the root module M_1 (which must be attached to a stationary link of the figure) with a uni-directional rotation. Note that the module that drives the last link in the chain does not have pulley parts because no further rotation propagation is required. To simplify the motion and layout parameters described below, we restrict D_i and A_i to be the same size.

Motion parameters

There are several parameters that define the mechanical motion of a chain of oscillation modules. The motion of a single module is defined by the rotation speed $\dot{\alpha}_i$ of the input gear D_i, the bar lengths b_i, c_i, g_i in the four-bar linkage, and the initial angles $\theta_i(0)$ and $\phi_i(0)$ of the input and output cranks in the linkage (see Figure 4a). Note that the length h_i of the bar connecting the ends of the two cranks B_i and C_i is fully determined by the other linkage parameters. Following [McCarthy 2000], we can express the angle $\phi_i(t)$ of the output crank C_i at time t as:

$$\phi_i(t) = \arctan\left(\frac{S_i(t)}{T_i(t)}\right) + \arccos\left(\frac{U_i(t) - V_i(t)}{W_i(t)}\right) \quad (1)$$

$$S_i(t) = 2b_i c_i \cos(\theta_i(t)) - 2g_i c_i$$
$$T_i(t) = 2b_i c_i \sin(\theta_i(t))$$
$$U_i(t) = 2b_i g_i(\cos(\theta_i(0)) - \cos(\theta_i(t)))$$
$$V_i(t) = 2c_i(g_i \cos(\phi_i(0)) - b_i \cos(\theta_i(0) - \phi_i(0)))$$
$$W_i(t) = 2c_i\sqrt{g_i^2 + b_i^2 - 2b_i g_i \cos(\theta_i(0))},$$

where $\theta_i(t)$ denotes the rotation of the input crank B_i at time t. If D_i is rotated by an angular speed of $\dot{\alpha}_i$, both A_i and B_i will have an angular speed of $-\dot{\alpha}_i$. Thus, we can conclude that $\theta_i(t) = \theta_i(0) - t\dot{\alpha}_i$.

When we connect a chain of oscillation modules together, we must account for the fact that the input crank rotation $\theta_i(t)$ for each module depends on the motion of the parent link l_{i-1}. Since the axle of A_i is connected to l_{i-1}, A_i rolls around D_i as l_{i-1} rotates, which affects the total rotation of A_i (see Figure 4b). Specifically, if we apply the rotation speed $\dot{\alpha}_i$ to D_i, then the angular speed of A_i is a combination of this input rotation and the rotation of l_{i-1}. As a result, the definition for $\theta_i(t)$ in a chain of modules becomes

$$\theta_i(t) = \theta_i(0) - t\dot{\alpha}_i + (\phi_{i-1}(t) - \phi_{i-1}(0)), \quad (2)$$

where $(\phi_{i-1}(t) - \phi_{i-1}(0))$ denotes the total rotation of C_{i-1} (and thus the attached link l_{i-1}) with respect to its initial angle at time t. Since the root module M_1 that drives the first link l_1 in the chain is attached to the main support structure of the automaton rather than a moving link, we define $\phi_0(t)$ to be 0 for all t.

Our design imposes a few contraints on the motion parameter values. To ensure that the input crank B_i of the linkage can rotate fully when driven by D_i, we constrain the relative lengths of the linkage bars as follows [McCarthy 2000]:

$$\begin{aligned} g_i + b_i - h_i - c_i &< 0 \\ (h_i - c_i)^2 - (g_i - b_i)^2 &< 0 \end{aligned}$$

Further, since the argument of the arccos function in Equation 1 must be in the range -1 to 1, we enforce the following constraint:

$$2g_i(b_i \cos(\theta_i(0)) - c_i \cos(\phi_i(0))) + 2b_i c_i \cos(\theta_i(0) - \phi_i(0)) - b_i^2 - c_i^2 - g_i^2 \leq 0$$

The equations above define the space of all possible motions for a chain of rigid links using our design. Section 3.2 describes how we use these relationships to compute motion parameters that best approximate an input animation sequence.

Layout parameters

While the motion parameters partially specify the configuration of the oscillation module components, we must specify several additional parameters to fully define the layout of the module. The parameters b_i, c_i, g_i determine the lengths of the linkage bars, but since scaling the entire linkage uniformly does not change its motion, we treat the overall scale ω_i as a free layout parameter. The other layout parameters include the radii of the gears and pulley wheels $(r_{D_i}, r_{A_i}, r_{P_i}, r_{R_i})$ as well as the pulley belt length s_{Q_i} and link length s_{l_i}. As mentioned earlier, we restrict D_i and A_i to be the same size, so we set $r_{A_i} = r_{D_i}$ and eliminate r_{A_i} as a free parameter. Given a fixed tooth size, the tooth count of the gears, pulley wheels and pulley belt determines the radius/length of those components. Thus, our layout optimization expresses $r_{D_i}, r_{P_i}, r_{R_i}, s_{Q_i}$ in terms of integer tooth counts $k_{D_i}, k_{P_i}, k_{R_i}, k_{Q_i}$.

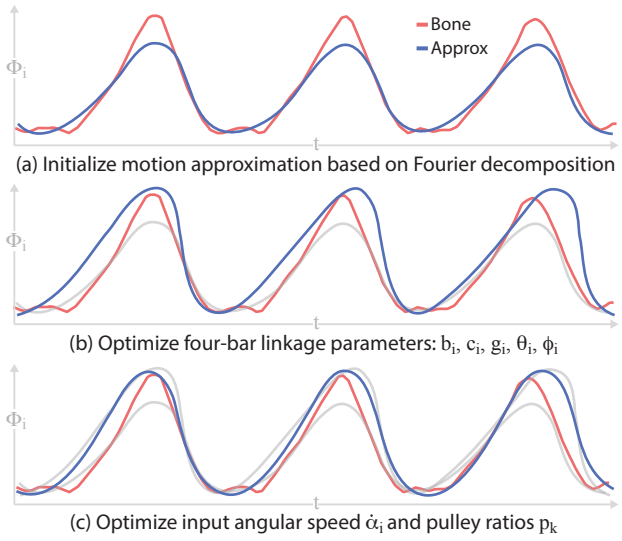


Figure 5: *Motion approximation graphs.*

Not all assignments of layout parameters result in physically valid layouts. There are three relevant constraints:

1. To prevent A_i from colliding with the axle of the next link l_{i+1}, we must ensure that $r_{A_i} < \omega_i g_i$.

2. Similarly, to avoid collisions between the pulley wheel R_i of the module M_i and the axle of the gear A_{i+1} of the next module M_{i+1}, the relation $r_{R_i} < 2r_{A_{i+1}}$ must hold.

3. The length of each link must match the size of the components placed on the link, so that $s_{l_i} = 2r_{A_{i+1}} + \omega_{i+1}g_{i+1}$. The link length also influences the pulley belt length s_{Q_i}. However, since belts with a small amount of slack do not prevent the pulley from functioning, we represent this relationship as an energy term in our layout optimization rather than a hard constraint (see Section 3.3).

There are also a few constraints that reflect limitations of our fabrication process.

1. We manufacture gears D_i and A_i with a laser cutter, which limits the minimum and maximum sizes (i.e., tooth counts) of these components. In our designs, we constrain the tooth count to be between 14 and 80.

2. Similarly, we constrain the lengths of the linkage bars based on the minimum bar length ($8mm$) that we can cut.

3. Finally, since it is difficult to create robust pulley wheels and belts with a laser cutter, we use stock pre-made pulley parts that are only available in a discrete set of wheel and belt sizes. To encode this constraint in our layout parameterization, we introduce a set of binary indicator variables $\{u_1, u_2, ..., u_N\}$ to select between the N available part sizes $\{v_1, v_2, ..., v_N\}$, and we represent the size of each pulley part as $\sum_{i=1}^{N} u_i v_i$ with the constraint $\sum_{i=1}^{N} u_i = 1$.

This set of layout parameters and constraints fully specifies the physical layout of a limb mechanism in our design. Section 3.3 describes how we solve for a layout that matches a set of target motion parameters.

3.2 Motion approximation

Given a mocap sequence as input we simplify the animation by projecting the motion of the bones within each limb onto a plane. We then run an optimization to find the motion parameters that yield the best approximation of the planar motion that can be generated with a chain of oscillation modules.

Planar approximation

For each kinematic chain, we start by tracing the path $X_i = \{x_i(1), x_i(2), ..., x_i(n)\}$ of the endpoint of the ith bone at each frame of the input animation sequence with respect to its parent joint where n is the total number of frames. We then compute the least-squares plane of the traced path and project each point onto this plane to obtain a planar path $Y_i = \{y_i(1), y_i(2), ..., y_i(n)\}$. Finally, for every frame j, we convert the motion of each bone to a rotation $\Phi_i(j)$ around the normal of the fitted plane as

$$\cos(\Phi_i(j)) = \cos(\Phi_i(j-1)) + \frac{y_i(j)}{\|y_i(j)\|} \cdot \frac{y_i(j-1)}{\|y_i(j-1)\|},$$

where $\cos(\Phi_i(1))$ denotes the initial orientation of the bone with respect to its parent and $\cos(\Phi_i(0)) = 0$. Converting the rotation of a single input crank to motion in different planes requires the use of bevel gears. Since standard bevel gears only allow changing the rotation axis across orthogonal planes, we snap the plane of motion for each limb to the saggital, coronal or transverse plane of the figure.

Motion parameter optimization

Given the planar bone rotation angles $\Phi_i(j)$ for each chain, our goal is to compute motion parameters that approximate the mocap bone motion with the corresponding rigid links in the automaton. Since each link l_i is driven by the output crank C_i, we aim to produce a mechanical motion where changes in the crank angle $\phi_i(t)$ match changes in the bone angles $\Phi_i(j)$. Thus we define the following energy term for each chain:

$$E_{\Phi_i} = \sum_i \sum_j \left(\sin \frac{\Delta_{\Phi_i}(j) - \Delta_{\phi_i}(j)}{2} \right)^2 \qquad (3)$$

where i indexes the bones, j indexes the frames of the animation sequence, $\Delta_{\Phi_i}(j) = \Phi_i(j+1) - \Phi_i(j)$ denotes the change in the bone angle between frames j and $j+1$ and $\Delta_{\phi_i}(j) = \phi_i(j+1) - \phi_i(j)$ denotes the corresponding change in angle of the output crank C_i.

As explained in Section 3.1, $\phi_i(t)$ depends on the four-bar linkage parameters $b_i, c_i, g_i, \theta_i(0), \phi_i(0)$, and the input angular speed of each module $\dot{\alpha_i}$. We minimize E_{Φ_i} by optimizing these parameters. Instead of directly optimizing for $\dot{\alpha_i}$, we represent $\dot{\alpha_i}$ in terms of the pulley ratios for all the preceding modules in the chain. In particular, for all but the root module M_1, the input angular speed of M_i is defined as $\dot{\alpha_i} = \prod_{k=0}^{i-1} p_k$ where $p_k = k_{P_k}/k_{R_k}$ is the pulley ratio of module M_k. Based on this relationship, we optimize for the linkage parameters, the angular speed $\dot{\alpha_1}$ of the root module, and the pulley ratios p_i of each remaining module in the chain.

We apply a two-step optimization procedure to minimize E_{Φ_i}. In the first step, given $\dot{\alpha_1}$ and p_i, we solve for the linkage parameters $b_i, c_i, g_i, \theta_i(0), \phi_i(0)$ of each module. In the second step, we use the computed linkage parameters to update $\dot{\alpha_1}$ and p_i across all the bones in the chain. In both of these steps, we minimize the non-linear error function given in Equation 3 with respect to the

constraints involving the linkage parameters described in the previous section. We use the *SQP* method implemented in the *MATLAB Optimization Toolbox* to solve these optimization problems, which typically converges in 5-10 iterations.

The non-linearity of the objective function defined in Equation 3 mandates a good initialization of the optimization variables to obtain a valid minimum. Since we are considering periodic motions, we employ a frequency-space analysis for this purpose. More precisely, we initialize the motion parameters using a Fourier decomposition on the rotation angles of each bone in the chain. The motion of each bone is then approximated using the Fourier component of highest magnitude as

$$\Phi(j) = \mu * \cos(2\pi f(j/n) + \rho), \tag{4}$$

where μ represents the magnitude, f represents the frequency, and ρ represents the phase of the Fourier component. This cosine function represents the change in rotation of the output crank of the four-bar linkage as the input crank completes one full rotation cycle in n/f frames. Assuming a fixed initial orientation $\theta_i(0)$ of the input of the linkage, each sampled value of j corresponds to a desired pair of input and output crank rotation angles $(\theta_i(0)+2\pi j(f/n), \Phi(j))$. We sample 3 such angle pairs which uniquely specifies a set of linkage bar lengths that interpolate the sample points [Freudenstein 2010]. We initialize $\dot{\alpha_1}$ as $2\pi(f_1/n)$ where f_1 is the frequency of the Fourier component approximating the rotation of the first bone in the chain. We assume the initial values for the pulley size ratios are $p_i = (2\pi(f_i/n))/(2\pi(f_{i-1}/n))$.

To prevent having too short or too long bars upon scaling of the four-bar linkage, we add an additional energy term to penalize large ratios between the bar lengths. Specifically, we define the following relations between the bar lengths:

$$\kappa_i = \frac{b_i g_i \cos(\theta_i(0)) - g_i c_i \cos(\phi_i(0)) + b_i c_i \cos(\theta_i(0) - \phi_i(0))}{(b_i c_i)}$$
$$\lambda_i = g_i / b_i$$
$$\mu_i = g_i / c_i$$

and add the energy term below to the first step of the optimization:

$$E^i_{bar} = \left(\kappa_i - \frac{1}{\kappa_i}\right)^2 + (\lambda_i - 1)^2 + \left(\frac{1}{\lambda_i} - 1\right)^2 + (\mu_i - 1)^2 + \left(\frac{1}{\mu_i} - 1\right)^2$$

Finally, in order to reduce drift in the motion of the links over time, we introduce a periodicity constraint in the second step of the optimization. Specifically, assuming the provided motion sequence contains one cycle of the desired motion, the period of the overall motion is n frames. Each oscillation module should return back to its initial state at the end of the motion cycle. Thus, the periodicity of each bone in the chain is an integer divisor of n, and the input crank of the corresponding four bar linkage should be rotated by an integer multiple of 2π, $2\pi m_i$ during the given motion cycle. In addition to the propagated input rotation parameters, we also solve for these discrete multipliers m_i. However, since both E_Φ and the constraints defined in Section 3.1 are non-linear, we treat m_i as a continuous variable and adopt a branch and bound strategy to round them to integer values. Specifically, starting at the root module, at each iteration we round m_i to $\lfloor m_i \rfloor$ and $\lceil m_i \rceil$ and choose the value resulting in smaller optimization error. We observed that duplicating the input motion several times (10 in our experiments) and running the motion approximation to this longer cycle improves the approximation results while still satisfying the periodicity constraints.

3.3 Layout: kinematic chains

Given a simplified motion, the next step is to compute layout parameters for the oscillation modules of each kinematic chain that match both the target motion parameters and relative bone lengths of the input figure as closely as possible. As described in Section 3.1, the layout is defined by continuous parameters (ω_i, s_{l_i}), discrete parameters $(k_{D_i}, k_{P_i}, k_{R_i}, k_{Q_i})$, and the binary indicator variables for the pulley part sizes. Thus, we use mixed-integer programming to search over the space of possible solutions.

Energy function

Our energy function encodes several desired properties for the layout of each kinematic chain.

Propagated rotation. At the end of the motion parameter optimization step, we have computed a desired angular speed $\dot{\alpha_1}$ driving the root module M_1 of each chain and the ratios of the pulley wheel radii, p_i, that specify how the input rotation should be propagated to the remaining modules in the chain. Several layout parameters determine the actual values of $\dot{\alpha_1}$ and p_i in the final automaton. In our designs, the input crank of the automaton drives an input gear whose rotation is propagated to the driving gear D_1 of each root module M_1. Thus, the angular speed driving M_1 can be expressed as $\dot{\alpha_1} = \alpha_I k_I / k_{D_1}$ where α_I is the constant angular speed of the automaton input crank, k_I is the tooth count of the input gear, and k_{D_1} is the tooth count of D_1. The pulley ratios p_i can simply be expressed as k_{P_i}/k_{R_i} (i.e., the ratios of the corresponding pulley wheel tooth counts).

Based on these expressions, we add the following energy term to penalize deviations from the desired angular speed and pulley ratios:

$$E_{rot_i} = \begin{cases} (\dot{\alpha_I} k_I / k_{D_1} - \dot{\alpha_1})^2 & \text{if } i = 1 \\ (k_{P_i}/k_{R_i} - p_i)^2 & \text{if } i > 1 \end{cases} \tag{5}$$

Pulley belt length. In our oscillation module design, the wheels of each pulley, P_i and R_i, are attached to the axles at either end of link l_{i-1} (see Figure 3b). Thus, we can express the tooth count (i.e., length) of the pulley belt k_{Q_i} in terms of the tooth counts of the pulley wheels, k_{P_i} and k_{R_i}, and the length $s_{l_{i-1}}$ of link l_{i-1}:

$$k_{Q_i} z_p = z_p(k_{P_i} + k_{R_i})/2 + 2 s_{l_{i-1}}, \tag{6}$$

where z_p is a constant defining the distance between two consecutive teeth of the pulley wheels or belt.

The pulley wheels P_1 and R_1 that drive the second link l_2 of each chain are part of the root module M_1. Since M_1 is attached to the main support structure of the automaton rather than a link in the chain, the tooth count k_{Q_1} of the pulley belt Q_1 is not constrained by the length of a link. Instead, k_{Q_1} depends on the tooth counts of the pulley wheels, P_1 and Q_1, the input gear D_1, and the distance $\omega_1 g_1$ between the fixed pivots of B_1 and C_1:

$$k_{Q_1} z_p = z_p(k_{P_1} + k_{R_1})/2 + 2(z_p k_{D_1} + \omega_1 g_1) \tag{7}$$

In practice, a small amount of slack in the pulley belt does not prevent the mechanism from functioning properly. Therefore, instead of treating the above equalities as hard constraints, we define an energy term to penalize the difference between the belt length k_{Q_i} chosen from the discrete set of available options and the ideal belt length defined by the layout of the mechanism:

$$E_{pul_i} = \begin{cases} (k_{Q_i} z_p - z_p \frac{k_{P_i} + k_{R_i}}{2} + 2(m_p k_{D_{i-1}} + \omega_{i-1} g_{i-1}) - \epsilon)^2 & \text{if } i = 1 \\ (k_{Q_i} z_p - z_p(k_{P_i} + k_{R_i})/2 + 2 s_{l_{i-2}} - \epsilon)^2 & \text{if } i > 1 \end{cases}$$

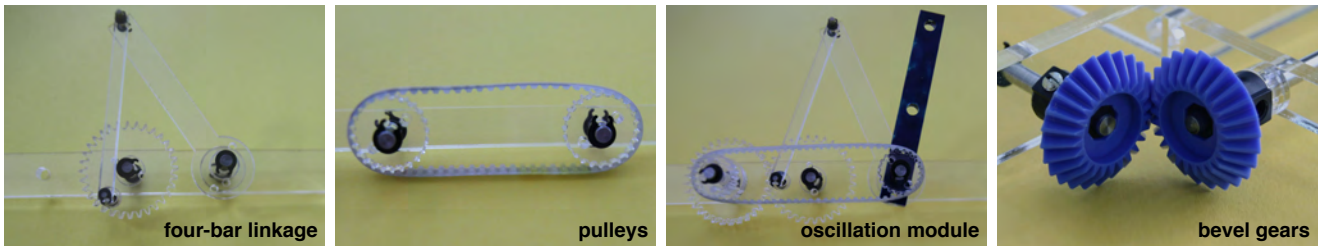


Figure 6: *The basic components of our automata. From left to right: Four-bar linkages transfer a rotational motion into an oscillating one. Belts and pulleys are used to propagate the input rotation along the kinematic chain. An oscillation module joins these components to represent one bone of the figure. Bevel gears change the orientation of the input rotation. Our optimization solves for the configuration, dimensions, and placement of all these components to best approximate a given input motion sequence.*

where we set $\epsilon = 1$ to allow for the slack. Note that the pulley part sizes referenced in this formulation are represented by the corresponding binary indicator variables as described in Section 3.1.

Link lengths. Each link in the chain should be long enough to accommodate the parts attached to itself. However, in order to preserve the shape of the input figure, the length s_{l_i} of each link should also match as closely as possible the corresponding relative bone length s_{b_i}. The dimensions of the bones are defined up to a scale using a free scale variable ξ. Thus, the desired length of each rigid link is $s_{l_i} = \xi s_{b_i}$. To penalize the deviations from the original bone lengths, we add the following term to our energy function for each link:

$$E_{s_{l_i}} = (s_{l_i} - \xi s_{b_i})^2$$

We further desire symmetric bone pairs (e.g., left and right upper leg bones) to have similar lengths. For each such symmetric bone pair, we penalize the differences between the corresponding link lengths as,

$$E_{s_{ij}} = (s_{l_i} - s_{l_j})^2.$$

Combined energy function. We combine all the energy terms for each link l_i in each serial chain c_j to obtain the final energy function:

$$E = \sum_{c_j} \sum_{l_i \in c_j} w_{rot_i} E_{rot_i} + w_{pul_i} E_{pul_i} + w_{s_{l_i}} E_{s_{l_i}} + \sum_{i,j \in S} w_{s_{ij}} E_{s_{ij}},$$

where S denotes symmetric bone tuples and w_{rot_i}, w_{pul_i}, $w_{s_{l_i}}$, and $w_{s_{ij}}$ are the weights for each energy term. In our examples, we set w_{rot_i} and w_{pul_i} to 5 and set $w_{s_{l_i}}$ and $w_{s_{ij}}$ to 1.

Mixed-integer progamming

Given the energy function E and all the linear hard constraints described above, we formulate the optimization as a mixed integer programming problem and solve it using a branch-and-bound technique. More specifically, we use the constrained mixed-integer solver CoMISo [Bommes et al. 2012] that provides a wrapper to the Gurobi solver [Gurobi Optimization 2012]. During the optimization process, the solver treats the discrete unknowns as continuous variables and generates a solution where these variables have fractional values. The solver then constructs a search tree where each branch represents the result of constraining a discrete variable to either the floor or ceiling of its fractional value from the continuous solution. In our experiments, this optimization process takes around 100 seconds to find a feasible solution for a problem with 360 binary, 10 integer, and 10 continuous variables. The result of the optimization is a set of layout parameters that fully define the size of the mechanical parts in each oscillation module.

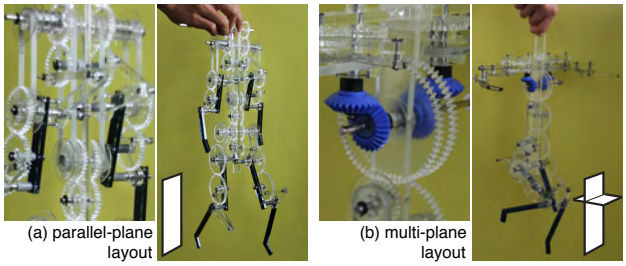


Figure 8: *The input crank rotation is propagated to the root module of each chain using pulleys. For multi-plane layouts, we also use bevel gears (right).*

3.4 Layout: unified design

The final step in our pipeline composes all the kinematic chains into a unified design that can be driven by a single input crank. This process involves choosing the orientation and position of the crank and then generating mechanisms that propagate the crank rotation to the root module of each moving chain in the automaton. In our designs, we attach the input crank to the torso link of the figure and use pulleys and bevel gears (if necessary) to propagate the rotation. We arrange these pulleys and bevel gears on the stationary torso, head and (in some cases) shoulder links based on the orientations of the motion planes for the moving limbs.

Parallel-plane layout: If the motion planes of the moving limbs are all parallel to either the saggital or coronal planes of the figure, we orient the input crank, torso and head links parallel to the motion plane (see Figure 8(a)). Recall from Section 3.1 that the root module of each chain must be placed on one of the stationary links. We position the root modules of the arms on the head link, and we put the root modules of the legs on the torso link. Finally, we use pulleys to connect the input crank to the driving gear of each root module. If the limbs move parallel to the saggital plane, we position the pulleys on the left and right sides of the torso/head links, and for coronal plane motion, we put the pulleys on the front and back sides of the torso/head links.

Multi-plane layout: If the motion planes of the moving limbs are all parallel to the transverse plane, or if the motion planes are not all parallel to each other, we incorporate bevel gears into our unified design (see Figure 8(b)). If all of the non-tranverse motion planes are parallel to the coronal plane, then we orient the input crank, torso and head links parallel to the coronal plane as well. Otherwise, we orient these components parallel to the saggital plane. We orient the root module of each chain so that its rotation plane is parallel to the rotation plane of the input crank. As in case 1, we place the root modules for the arms and legs on the head and torso links,

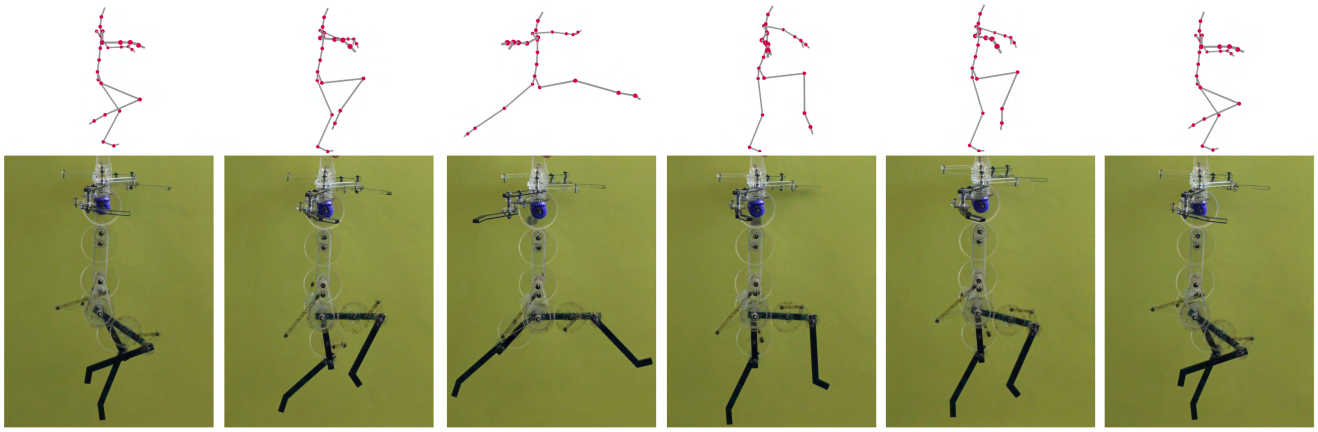


Figure 7: *Physical prototype of dancing motion sequence. (Please refer to the supplementary video.)*

respectively. For chains whose motion plane is orthogonal to the rotation plane of the input crank, we use bevel gears to convert the rotation of the root module output crank C_1 and pulley wheel R_1 to the appropriate motion plane.

Special case: Where the motion plane of an arm is parallel to the coronal or transverse plane (as in the dancing example in Figure 9) we use the shoulder link as an additional support structure to accommodate the root module of the arm. We orient the shoulder to be parallel to the motion plane of the arm, and if the input crank rotates in a different plane, we add a bevel gear configuration to convert the input crank rotation to the appropriate motion plane.

4 Evaluation

Data sets. In order to evaluate how well our planar motion simplification algorithm approximates various human motions, we analyzed a large number of mocap sequences from existing databases.

We first trace the path $X_i = \{x_i(1), x_i(2), ..., x_i(n)\}$ of the endpoint of the ith bone of a kinematic chain c_j at each frame of the input motion. We compute the common least squares plane $P_j(\mathbf{n}, d)$ of all such paths for c_j. We then measure the fitting error as the deviation from a planar motion over n frames as $E_j := \sum_{b_i \in c_j} \sum_n (\mathbf{n} \cdot x_i(n) + d)$. In order to account for differences in the lengths of the paths traced by each bone, we normalize the error by the total length of the paths, measured as $m_j := \sum_{b_i \in c_j} \sum_n \|x_i(n+1) - x_i(n)\|$. The final planar fitting score of the motion is then computed over all the chains as $M_p := \min_j m_j/(E_j + \epsilon)$ with $\epsilon = 0.01$ used as a regularizer. Essentially, M_p measures how well a non-trivial motion can be approximated using a set of planes, one for each chain.

We tested our algorithm on a total of 80 mocap sequences from the CMU motion database. We found the following motion types (about 40% of the database) to be well suited for our algorithm: walking, running, jogging, exercising, waving, drinking as they consistently get high scores ($M_p > 10$). We call them good motion types. On the other hand, sequences like swimming, swordplay, fishing ($M_p < 2$) contain large out-of-plane motions, and hence our algorithm cannot replicate them faithfully. In Figure 9, we present representative realizations for some of the good motion types (see also the accompanying video).

Simulation. Our system produces a mechanical automaton design for a given input motion. We simulate the mechanism using forward kinematics to qualitatively validate how faithfully the sequences recreate the input motion. Specifically, we build a mechanism graph $G = (V, E)$, where each node $v_i \in V$ represents a part (e.g., a gear or a pulley) and each edge $e_{ij} \in E$ connecting the vertices v_i and v_j represents the relation between the corresponding parts. In our setup, the typical relations include the coaxial or parallel-axis properties of the gears, and the motion chains comprising of pulleys and four-bar linkages. At every frame of the motion sequence, we assume that the input crank rotates clockwise with a constant speed. We propagate this rotation along the edges of the mechanism graph to compute the position and orientation of each component in the mechanism. Note that by construction, the graphs are free of any loops.

Fabrication. We fabricated automata for two of the designed sequences, one for each of the major types of layout configurations: the *walking* sequence (see Figure 1), which does not require any bevel gears; and the *dancing* sequence (see Figure 7), which involves motion in two different planes. We used off-the-shelf pieces for the bevel gears, pulleys, and pins. The linkages and gears were laser cut based on the dimensions as prescribed by our optimization. The most time consuming part was assembling the pieces, which was done manually. However, given that our design consists of similar modular components (i.e., the oscillation modules) the assembly process was relatively straightforward (albeit somewhat tedious). Finally, we used a single speed motor to drive the automata. Figure 1 and 7 and the supplementary video show our fabricated automata in motion. Note how subsequent joints oscillate at different phases and with different periods.

Results. Figure 9 shows the mechanical automata generated by our system for several motion sequences. For each motion type, several snapshots of the original animation and the corresponding simulation frames are provided. Our system generates automata consisting of a chain of oscillation modules driven by a single actuator. While the use of standard planar mechanical components simplifies the fabrication and assembly processes, there are aspects of certain motions that our method fails at reproducing accurately (see the accompanying video).

In all of our example automata the torso of the figure has been used as the main support structure. In the *dancing* example, the shoulders have been used as an additional support structure since the arms move parallel to the transverse plane of the figure. As described in Section 3.4, to propagate the rotation of the driver gear to each kinematic chain, pulleys are used to fill up the space between the

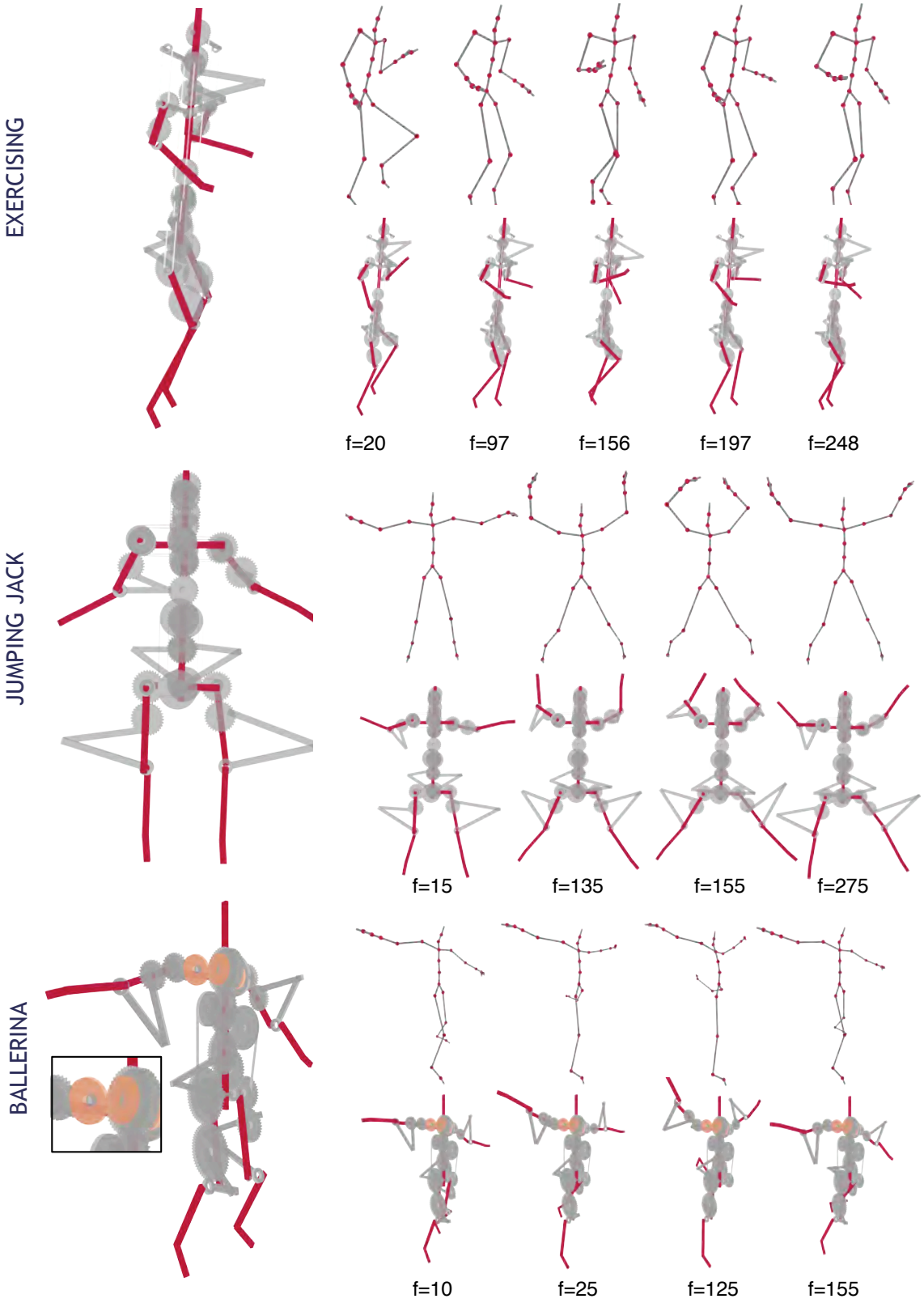


Figure 9: *Example automata generated by our system. For each example, snapshots of the original motion sequence in different time frames (denoted by the numbers) and the corresponding simulation result of the automata are given. The bevel gear configuration used in the ballerina sequence is shown in orange.*

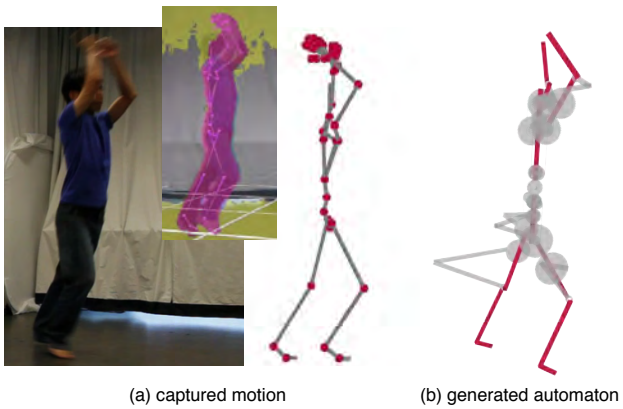


Figure 10: *Commercial motion sensing input devices such as the Microsoft Kinect enable direct capturing of input motions to our system. A snapshot of the captured motion sequence (a) and the generated automaton (b) is shown.*

driver gear and the root of the chain along the support structure.

While bevel gears enable mechanisms with arbitrary orientation of motion planes for different kinematic chains, such gears can be difficult to manufacture depending on the fabrication method. For example, it is not possible to create bevel gears with a standard laser cutter. In motion sequences where different kinematic chains move in nearly-parallel planes, the use of bevel gears can be avoided by mapping the input sequences to motions restricted to parallel planes. In the *walking, exercising, and jumping jack* examples, we show that plausible motion approximations can be generated despite this restriction. However, in the *dancing* automata, the use of bevel gears is necessary since the arms move in an orthogonal plane to the legs.

In our examples, we have primarily used motions selected from the existing motion databases as input. However, recent advances in commercial motion sensing input devices such as the Microsoft Kinect enable easy tracking of the human skeleton and thus provide an interface to directly capture input motions. We demonstrate the flexibility of our input requirements with the *kendo* sequence (see Figure 10 and the supplementary video). To produce this result, we used the Kinect to record a performance, extracted the human skeleton with the commercial tool *ipi Soft* [2013], and then used the skeleton motion as input to our system.

User control. Although our system has a fully automatic mode, the user can also control the final design in various ways.

(i) Depending on the target fabrication method, the user can provide additional constraints. Based on the specification of our laser cutter, we used minimum and maximum gear tooth counts of 14 and 80 (given a 1mm tooth size). These constraints have a direct effect on the realization of the desired motion, specifically for links that require a large change in input angular speed.

(ii) Our system restricts the motion planes of the limbs to be parallel to the saggital, coronal or transverse plane of the figure. The user can also manually specify the motion plane orientations. We use the automatically generated motion planes in our results.

(iii) Finally, the user can set a threshold for the minimum angular range for the motion of the bones (0.35 radians in our results). We set bones oscillating with an angular range less than this threshold to be fixed, thus reducing the complexity of the generated mechanisms.

Limitations. Our algorithm has several limitations:

(i) Since our automata are based on predominantly planar linkages, our initial planar projection step for each kinematic chain can significantly distort certain types of mocap sequences (e.g., swimming). Further, in order to simplify the fabrication/assembly processes, we generate automata driven by a single actuator that use standard planar mechanical components and standard bevel gears. These decisions impose additional constraints in the motion approximation step of our method. Better approximations can possibly be obtained by adding other types of mechanical elements (e.g. non-circular pulleys) at the cost of increased complexity of the optimization and the resulting automata.

(ii) We assume periodic motions as input to allow driving the automaton indefinitely with a constant-speed input rotation. While we did not observe noticeable drift for extended operation of our assemblies, machining imprecisions or mechanical wear could potentially cause temporal deviations that over time reduce the accuracy of the motion approximation. At a certain point, the automaton might have to be partially disassembled to re-align the parts to the original configuration.

5 Conclusion

In this paper, we have presented an algorithm to automatically realize a mechanical automaton starting from an input mocap sequence. Our work includes three main contributions. We introduce an oscillation module design that can generate kinematic chain motions with links that oscillate at different phases and frequencies. We also describe a motion approximation algorithm that converts an input sequence to a mechanically realizable motion. Finally, we present an automated method for determining the parameters and spatial layout of mechanical parts that takes into account physical and fabrication constraints. Our results demonstrate that our proposed approach can generate mechanical automata that reproduce non-trivial human motions with multiple moving limbs.

We see several interesting avenues for future work:

Perception of human motion: In the motion approximation part of our method, we use an objective function based on the rotation angles of each joint. Our intuition is that the changes in joint angles have a significant effect on how we perceive a motion since they are visually well exposed in the static body posture. Developing better perceptual metrics for comparing human motions is an interesting avenue for future work.

Joint optimization of motion and layout parameters: Our current method optimizes for the motion parameters of the mechanism first, and then solves for the layout of the parts. It is possible that a joint (or perhaps iterative) optimization of these parameters could produce better motion approximations in the final automaton designs.

Self-standing automaton: It would be interesting to consider the design of self-standing mechanical figures that remain balanced while they move. Generating such automata would require a method that accounts for both the stability of the design as well as the target motion. One challenge here is that the placement of mechanical components (e.g., gears, linkages) affects the weight distribution and thus the stability of the automaton.

Integrated designs: It might be possible to fabricate automata with embedded movable parts that do not require assembly. If the goal is to place all of the mechanisms within the volume of the figure, this would likely require very compact arrangements of mechanical parts.

Extension to other mechanism types: The basic building block of

our design is a planar four-bar mechanism, which is a very common type of linkage. Our motion approximation and parameter optimization algorithms can be useful in designing a variety of linkage-based mechanisms (e.g., windshield wipers, vehicle suspensions, moving parts in industrial machinery, path tracing mechanisms, etc.). Furthermore, by specifying different constraints between the lengths of the linkage bars, our method can be adapted to handle other types of linkages, such as slider-crank and folding linkages.

Acknowledgements

We thank the anonymous reviewers and Nobuyuki Umetani for their valuable comments. We thank Minh Dang for performing the *kendo* sequence and Zohreh Sasanian for helping with the assembly of the physical prototypes. We are grateful to the ENAC Output Center at EPFL for providing us with the laser cutting facilities and Tina J. Smith for the video voice over. This research has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 257453, ERC Starting Grant COSYM, a Marie Curie CIG, an Adobe Research grant, and a UCL impact award.

References

ANANTHA, R., KRAMER, G. A., AND CRAWFORD, R. H. 1996. Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design 28*, 9, 707 – 722.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *Proc. ACM SIGGRAPH 31*, 4, 47:1–47:9.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2012. Practical mixed-integer optimization for geometry processing. In *Intl. Conf. of Curves and Surfaces*, 193–206.

CALÌ, J., CALIAN, D. A., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 2012. 3d-printing of non-assembly, articulated models. *ACM TOG (SIGGRAPH Asia) 31*, 6, 130:1–130:8.

CARNEGIE MELLON UNIVERSITY, 2003. Cmu graphics lab motion capture database.

CHIOU, S.-J., AND SRIDHAR, K. 1999. Automated conceptual design of mechanisms. *Mechanism and Machine Theory 34*, 3, 467 – 495.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. *Proc. ACM SIGGRAPH 32*, 4 (July), 83:1–83:12.

FREUDENSTEIN, F. 2010. Approximate synthesis of four-bar linkages. *Resonance 15*, 8, 740–767.

GUROBI OPTIMIZATION, I., 2012. Gurobi optimizer reference manual.

HALLER, K., JOHN, A. L.-S., SITHARAM, M., STREINU, I., AND WHITE, N. 2009. Body-and-cad geometric constraint systems. In *Proc. Symp. on Applied Computing*, ACM, 1127–1131.

HAN, Y.-H., AND LEE, K. 2006. A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms. *Computers in Industry 57*, 4, 305 – 318.

IPI SOFT LLC., 2013. ipi soft - markerless mocap.

KIM, J., KIM, K., CHOI, K., AND LEE, J. 2000. Solving 3d geometric constraints for assembly modelling. *The International Journal of Advanced Manufacturing Technology 16*, 843–849.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: partitioning models into 3d-printable parts. *ACM TOG (SIGGRAPH Asia) 31*, 6, 129:1–129:9.

MCCARTHY, J. 2000. *Geometric Design of Linkages*. Interdisciplinary Applied Mathematics Series. Springer Verlag.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2010. Illustrating how mechanical assemblies work. *Proc. ACM SIGGRAPH 29*, 3.

PENG, X., LEE, K., AND CHEN, L. 2006. A geometric constraint solver for 3-d assembly modeling. *The International Journal of Advanced Manufacturing Technology 28*, 561–570.

POLLARD, N., HODGINS, J. K., RILEY, M., AND ATKESON, C. 2002. Adapting human motion for the control of a humanoid robot. In *IEEE ICRA*.

ROY, U., PRAMANIK, N., SUDARSAN, R., SRIRAM, R., AND LYONS, K. 2001. Function-to-form mapping: model, representation and applications in design synthesis. *Computer-Aided Design 33*, 10, 699 – 719.

SAFONOVA, A., POLLARD, N., AND HODGINS, J. K. 2003. Optimizing human motion for the control of a humanoid robot. In *Intl. Symp. on Adaptive Motion of Animals and Machines*.

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware Design with Intersecting Planar Pieces. *Computer Graphics Forum 32*, 2, 317–326.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: improving structural strength of 3d printable objects. *Proc. ACM SIGGRAPH 31*, 4, 48:1–48:11.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *Proc. ACM SIGGRAPH 31*, 4, 86:1–86:11.

XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. 2009. Joint-aware manipulation of deformable models. *ACM TOG (SIGGRAPH Asia) 28*, 3, 35:1–35:9.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. *ACM TOG (SIGGRAPH Asia) 31*, 6, 127:1–127:10.