Computational Design of Walking Automata

Gaurav Bharaj[1]* Stelian Coros[2] Bernhard Thomaszewski[3] James Tompkin[1] Bernd Bickel[4] Hanspeter Pfister[1]
[1]Harvard SEAS [2]Carnegie Mellon University [3]Disney Research Zürich [4]IST Austria

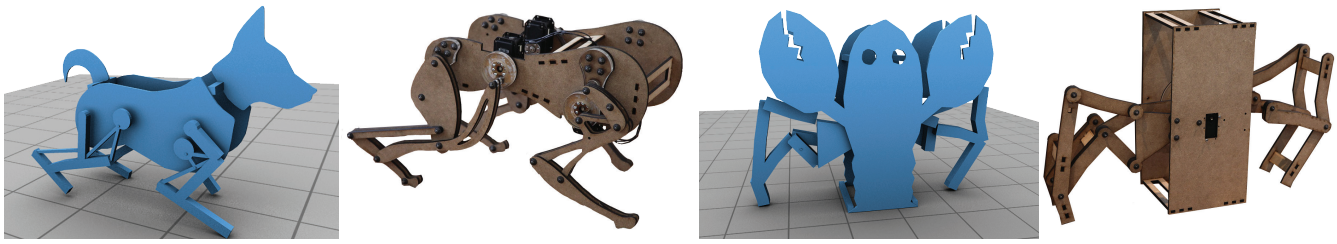


Figure 1: *Two walking automata designed with our system and the corresponding fabricated prototypes.*

Abstract

Creating mechanical automata that can walk in stable and pleasing manners is a challenging task that requires both skill and expertise. We propose to use computational design to offset the technical difficulties of this process. A simple drag-and-drop interface allows casual users to create personalized walking toys from a library of pre-defined template mechanisms. Provided with this input, our method leverages physical simulation and evolutionary optimization to refine the mechanical designs such that the resulting toys are able to walk. The optimization process is guided by an intuitive set of objectives that measure the quality of the walking motions. We demonstrate our approach on a set of simulated mechanical toys with different numbers of legs and various distinct gaits. Two fabricated prototypes showcase the feasibility of our designs.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics And Realism—Animation

Keywords: Mechanical characters, animation, fabrication

1 Introduction

Stories of walking automata date back to ancient Greece, with the Renaissance reviving the tradition and providing prominent historical examples: da Vinci's mechanical Lion, from the 15^{th} century, delighted audiences upon its rebuilding in 2009. Although toy stores abound with mass-produced walking automata, their design remains challenging. It is arguably for this reason that many commercial walking automata are based on a small number of template mechanisms such as the famous Klann or Jansen linkages. But, even in this restricted setting, the design problem is still far from obvious: in addition to satisfying kinematic requirements on *individual* mechanisms (i.e., producing desired end-effector motion), the designer must carefully select parameters for *all* mechanisms, including timing and inertia, to obtain stable walking motion. This task is challenging for experts, and so is often beyond the capabilities of average users.

*Email: bharaj@g.harvard.edu

Modern rapid and additive manufacturing techniques expand the scope of mechanical design and construction. Motivated by this tremendous progress, the graphics community has started to embrace the challenge of translating digital characters into physical artifacts [Zhu et al. 2012; Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014]. These methods can create virtual characters with a broad range of complex motions, such as an in-air mimic of a walking motion, but their fabricated physical counterparts have yet to demonstrate an ability to actually walk stably. This is because neither the mechanics nor the geometry of stable locomotion are considered during the design. Considering these constraints would allow users to explore the space of feasible linkages for stable walking, and so to more easily design walking automata.

Capitalizing on this work, our method allows users to intuitively create unique automata designs that walk stably once fabricated. First, we learn a space of linkage configurations which are likely to lead to stable walking. Then, the user designs the automata by placing 2–4 linkage templates onto a body at arbitrary positions. From this initialization, we optimize the overall mechanical structures of the automata, allowing them to automatically discover how to walk with the same intuitive set of objective functions. This approach integrates physics-based simulation of mechanical assemblies with an evolutionary optimization algorithm that is able to explore the complex design space of these structures. We demonstrate our approach with simulated designs, and we validate our simulations by fabricating two very different walking designs for a dog and a crab.

2 Related Work

Character Animation One major inspiration is Sims' [1994] pioneering work on virtual creatures that discover how to locomote. Sims uses genetic algorithms — a class of evolutionary optimization methods inspired by natural selection — to discover the structure and morphology of virtual creatures, and to discover time-varying actuation forces that lead to crawling or jumping motions. This general approach has been successfully adapted to a large range of additional animation domains, such as realistic control of swimming [Tan et al. 2011], muscle-driven biped simulation [Geijtenbeek et al. 2013], gait discovery for quadrupeds [Lee et al. 2013], or learning bicycle stunts [Tan et al. 2014].

Translating virtual walk simulations into the real world is non-trivial. For humans and animals, hundreds of muscles have to act in unison via a central nervous system for stable and efficient gaits. In a robot, the orchestration of actuators as muscles requires many sensors and a complex controller. In this context, Sims-like virtual characters are difficult to fabricate as, even if one could find physical actuators and

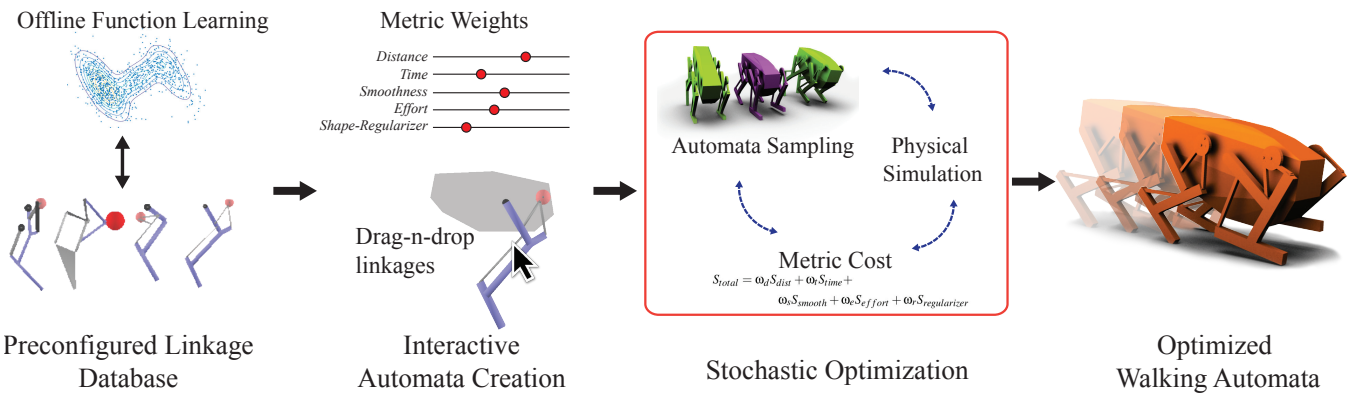


Figure 2: *Overview:* Left: *We learn valid linkage configuration functions from a database of prefigured linkages.* Mid-left: *The user designs an automata with a drag-and-drop interface, and specifies a metric by changing weights.* Mid-right: *Stochastic optimization using the valid linkage functions and physical simulation allows the automata to walk.* Right: *The user analyzes the final walk optimized by our system.*

joints for all virtual motors, the resulting cost would exceed what is acceptable for most applications, and especially for the simple automata that we consider: for toy and educational use, they have one motor per limb, no sensors, and no high-level controller; yet, they can walk successfully once fabricated.

Walking Motions Control in Animation and Robotics A variety of advanced control methods have been proposed for antropomorphic physically-simulated humans [Geijtenbeek et al. 2013; Lee et al. 2010] and animals [Wampler and Popović 2009; Coros et al. 2011]. Such methods have been applied to sophisticated legged robots to generate controllers [Gehring et al. 2013], or to increase the agility of locomotion controllers [Gehring et al. 2014]. However, complex control strategies require complicated mechanics, sensors, and actuators, and the StarlETH robot is well beyond the complexity and cost of our target of automata as toys. Our designs are significantly simpler in nature, yet are still able to perform walking motions. As such, our work is much closer to recent work in computational design than to the general field of robotics.

Computational Design and Fabrication This field reduces the difficulty of design and manufacturing problems by creating tools which forego or reduce the need for expert domain knowledge. For instance, recent works present custom-shaped objects that can fly [Umetani et al. 2014], stand on their own [Prévost et al. 2013], or spin stably [Bacher et al. 2014]. Some methods aim to bring virtual characters to the real world, and it is now possible to create 3D printable representations of virtual characters with joints [Bächer et al. 2012; Calì et al. 2012], to design mechanical toys capable of interesting (non-walking) motions [Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014], or to manufacture physical characters using elastic materials such that their deformation under the influence of external forces can be controlled.

Coros et al. [2013] note that even if the motion of a mechanical character at first glance resembles walking, this does not mean that the character would actually walk if fabricated. In initial experiments, we were not able to create any automata in this way that were capable of walking stably, unless we used a large number of legs (i.e., hexapod). This highlights the need for automated methods. To our knowledge, our work is first to investigate the challenge of designing automata that can walk stably.

3 Overview

Each of our automatons has a body with 2–4 linkages attached. Each linkage belongs to a set of linkage classes, with each parameterizable by pin joints and timings (Fig. 3). The 12-16 kinematic parameters $\mathbf{p}_k$, encode the location of each pin joint, and so define the mechanical configuration of the automata. Each linkage is assumed to be driven by a single servo motor with speed control, so the four timing parameters $\mathbf{p}_t$ per timing mechanism are used to control the relative phase between different linkages by directly specifying the phase profile functions of the virtual actuators. These linkages can be standard mechanisms, such as Klann or Jansen linkages, or designed automatically [Coros et al. 2013; Thomaszewski et al. 2014]. Our initial database of pre-configured linkages was created using the method of Coros et al. [2013]. We show two initial linkage configurations in supplemental appendix B.

Initially, we provide a simple drag-and-drop design tool to the user, who chooses and places linkage classes at arbitrary positions on the body (Fig. 2a, and supplemental video). Following this, we employ a stochastic genetic algorithm (CMA) and a walking objective function (§4) to optimize the linkage instance parameters with respect to the body in a rigid-body physical simulation (Supplemental appendix A). The simulation measures the quality of walking motions as the parameters of the automata — the mechanical linkage structures — change. It also previews the physical prototype output.

Not all linkage parameterizations are valid configurations which will move or lead to smooth motion, and so to speed up this optimization, we precompute within linkage space a subset of good linkage configurations. This data-driven approach is a reparameterization equivalent to learning and navigating a manifold on the original high-dimensional space (§5).

Finally, we fabricate the optimal automata. The fabrication process is manual, with all kinematic parameters and motor controls output directly by the optimization process. Physical prototypes are manufactured using a laser cutter and plywood, with metal bolts for joints, and servo motors for drive (§6).

4 Quantifying Walking

Starting from the input configuration created by the user with our simple interface, which is unlikely to walk, we must automatically optimize the design to walk. Walking is complex, requiring muscles (or motors and sensors) to act in unison for stable and efficient gaits. We assume arbitrary kinematic configurations, formed from high-dimensional parameterizations. Rather than attempting the very difficult task of explicitly applying locomotion control knowledge from robotics, instead we describe an intuitive set of sub-objectives, which allows for stable, efficient, and sometimes interesting gaits to

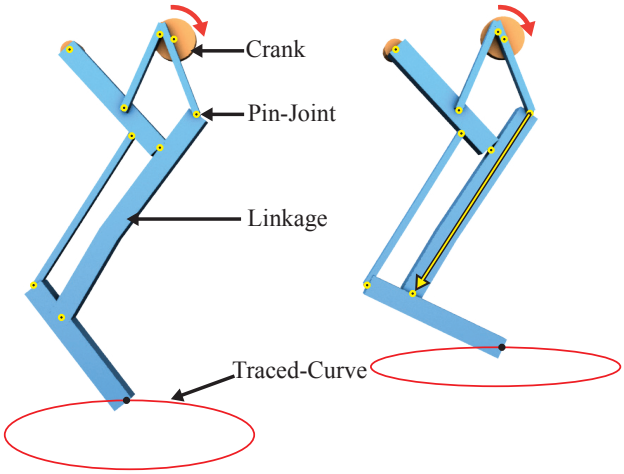


Figure 3: *The locations of the pin joints (yellow circles), which define the kinematics of the mechanical toys, are parameters optimized by our method. Changing the location of a pin joint (yellow arrow) leads to a change in the structure of the mechanical leg, as well as to a change in the motion of its end effector (red curves).*

arise. We define the walking objective as the weighted sum:

$$S_{total} = \omega_d S_{dist} + \omega_t S_{upright} + \omega_s S_{smooth} + \omega_e S_{effort} + \omega_r S_{regularizer} \quad (1)$$

During optimization, for each explored set of parameters, i.e., each automaton configuration, we compute the mass and moment of inertia of each rigid body part, and then physically simulate the automaton. The simulation proceeds until a failure mode is encountered, or until a fixed amount of simulation time $T = 30s$ has elapsed. Simulation position and orientation states and derivatives, $\mathbf{s} = (\mathbf{q}, \dot{\mathbf{q}})$, are recorded for each timestep t. The recorded simulation states are used to evaluate each individual sub-objective (below). Some parameter settings will result in infeasible mechanisms, so we monitor constraint values and terminate the simulation early if they exceed a threshold value.

Distance: We wish the mechanism to walk *forwards*, not on the spot, and so we measure the vector $\mathbf{d}$ between the center of mass at the beginning and at the end of the simulation, and define S_{dist} simply as $-\mathbf{v}^T\mathbf{d}$, where $\mathbf{v}$ is a unit vector that points along the desired walking direction. The negative sign is used to promote walking longer distances, since we minimize the total score S_{total}.

Upright: In early experiments, the automaton would often perform a somersault, as this quickly increases the distance traveled. This is not successful walking, nor is it an incremental solution towards better walking. Worse, we found that the optimization could not recover from such local minima. To avoid these solutions altogether, we add a term which encourages the automaton to remain upright for as long as possible. We define $S_{upright} = (T - t)^2$, where t is the time when the simulation is stopped. The simulation is stopped early (i.e., $t < T$) if parts of the automaton other than the feet or bottom of the body touch the ground.

Smoothness: We penalize the acceleration of the center of mass at every timestep. The acceleration $\mathbf{a}$ is estimated using finite differences on the generalized velocity vectors $\dot{\mathbf{q}}$ at consecutive time steps. The penalty term is defined as $S_{smooth} = \mathbf{a}^t\mathbf{a}$. Increasing the weight of this term leads to more conservative motions which we posit increases the likelihood of successful walking once fabricated.

Effort: While infeasible mechanisms (i.e., singular configurations) are pruned early on, this does not mean that all mechanisms are equally desirable. For instance, small moment arms require larger motor torques and result in large internal forces acting on the mechanical structures, increasing the likelihood of mechanical failure. Thomaszewski et al. [2014] analyzed the singular values of the constraint Jacobian $\partial\mathbf{C}/\partial\mathbf{q}$ (see Supplemental appendix A) as a continuous measure of the distance away from singular configurations. Since we physically simulate, we have direct access to the magnitudes of the forces needed to satisfy the joint and motor constraints (i.e., λ in Supplemental appendix A, eq.1).

Therefore, we define $S_{effort} = \lambda^T\lambda$ and add it to S_{total} to minimize the net internal forces acting throughout the mechanisms.

Regularizer: Since the input to our system consists of a user-created automaton, we would like to change the design as little as possible. Therefore, we introduce a simple regularizer for the kinematic parameters $\mathbf{p}_k$: $S_{regularizer} = (\mathbf{p}_k - \mathbf{p}_k^0)^T(\mathbf{p}_k - \mathbf{p}_k^0)$, where $\mathbf{p}_k^0$ represents the initial parameter values.

5 Optimization

The optimization problem introduced in the previous section is high-dimensional, non-linear, and non-smooth due to the unilateral and intermittent nature of the contact forces. Consequently, gradient-based methods are ill-suited, and we resort to a derivative-free stochastic evolutionary optimization technique based on the Covariance Matrix Adaptation algorithm [Hansen 2006]. Some works attempt to improve CMA by adding derivative-based local constraint satisfiers [Wampler and Popović 2009]; however, for our problem this would make little improvement as the spaces are not smooth.

CMA generates parameter samples according to an internal Gaussian distribution. After evaluating the objective value for each sampled point, the Gaussian distribution is updated, and the process repeats until convergence. However, as noted by Coros et. al. [2013], the parameter spaces of mechanical linkages are highly non-linear and random sampling, even around valid configurations, can quickly lead to degenerate mechanism that grind to a halt in mid-step or cannot move at all. Our experiments confirmed these observations, indicating that only ≈30–40% of the randomly generated samples correspond to valid configurations. To not waste computation time, it would be desirable to rule out such degenerate configurations even before running any simulations. One solution might be to first check all linkages of the automaton in isolation and reject samples for which at least one degenerate linkage was detected. However, while seemingly simple and efficient, this strategy would not allow CMA to adapt its internal distribution and ultimately prevent the algorithm from exploring more relevant regions of the parameter space (see also Xu et al. [2012]).

To alleviate this problem, we note that linkages can be analyzed for validity before optimization to learn valid configurations for walking. For example, an errant pin joint configuration which locks linkages is unlikely to lead to a successful walk. We would like to construct a reparameterization of the space of linkages such that (almost) all samples generated by CMA correspond to valid linkage configurations. Therefore, we compartmentalize the valid linkage kinematic parameter problem, and attempt to learn a function of valid kinematics parameters from which to draw more useful samples during CMA. To this end, we turn to Gaussian Mixture Models.

5.1 Linkage Configuration Function Learning

Learning Parametric Function We observe that there are many pockets of space in which valid samples exist. As such, we use a multivariate Gaussian mixture model (GMM) [Bishop et al. 2006] to create a parametric function over valid spaces:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{2}$$

Here, $0 \leq \pi_m \leq 1$ and $\sum_{m=1}^{M} \pi_m = 1$ are the mixing coefficients, $\boldsymbol{\mu}_m$ is an n-dimensional mean vector, and $\boldsymbol{\Sigma}_m$ is an $n \times n$ covariance matrix of the m^{th} multivariate Gaussian in the mixture.

We want to model the space of valid configurations with multiple Gaussians in a mixture. Given a number of mixtures (which we discuss discovering later on), we introduce a random *latent* variable C which assigns a linkage configuration to a Gaussian in the mixture. This is indicated by an m-dimensional binary variable where only one element is ever one, with all others zero (1-of-M representation). C is modeled as a distribution $p(\mathsf{C}) = \prod_{m=1}^{M} \pi_m^{\mathsf{c}_m}$ s.t. the marginal distribution over C is given by the mixing coefficients, $p(\mathsf{c}_m = 1) = \pi_m$. Random variable K is a continuous and observed, and is given by a multivariate GMM, $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, and models the kinematic parameters of the linkage configuration.

Data Preprocessing For a linkage, each pin joint location, i.e., the kinematic parameter $\mathbf{p}_k$ between two bars, can vary over the lengths of the connecting bars. We vary these points of location within $-\frac{\mathbf{l}}{4} \leq \mathbf{p}_k \leq \frac{\mathbf{l}}{4}$, where $\mathbf{l}$ is a vector of lengths of the corresponding bars in the linkage. Using these box limits, we draw samples $\mathcal{D}$ from the space of possible linkage configurations using Latin-Hypercube Sampling [McKay et al. 1979]. This sampling strategy uniformly subdivides the sample space and ensures that a sample is drawn from each division, thus sampling the whole space. Then, for each sample, we simulate the motion of the corresponding linkage by minimizing the constraint energy (see supplemental Appendix A) for pin-joints and motor-constraints. We sum the energies for each step of a full motion cycle and label a sample as valid, if the summed energy is below a given threshold value (we use 10^{-2}).

Parameter Learning Given sampled data $\mathcal{D}$, we would like to calculate Gaussian parameters $\Theta_m = \{\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}$ and $\boldsymbol{\pi}_m$ which best model it: we wish to maximize the likelihood of $\mathcal{D}$ given $\Theta_m, \boldsymbol{\pi}_m$. Mathematically (in log space) this is defined as:

$$\sum_{i=1}^{N} ln\left(\sum_{m=1}^{M} \boldsymbol{\pi}_m \mathcal{N}(d_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \tag{3}$$

Initially, the dimensionality of the 1-of-M representation of C is unknown, so we use Expectation Maximization [Bishop et al. 2006] to find the maximum likelihood estimation of the random variable parameters.

Number of Mixtures & Over-Fitting The number of components in the mixture plays an important role in determining the effectiveness of the learned model. Depending on the dimensionality of the problem, using too few mixture components can lead to inaccurate model-parameters, while too many mixture components can lead to over-fitting. Hence, we use the Bayesian Information Criteria [Schwarz et al. 1978] to determine the number of categories, i.e., the dimensionality of C, and thereby the number of Gaussians in the mixture. Similar to [Chaudhuri et al. 2011], we penalize Equation 3 by $\frac{1}{2} M log(|\mathcal{D}|)$, the BIC criteria, by sequentially varying the number of mixtures, and use the one that maximized the cost. For a given linkage, we found that $6 - 12$ mixtures gave us good results. For example for the linkage configuration given in Figure 4, we found that six components maximized the EM energy in Equation 3.

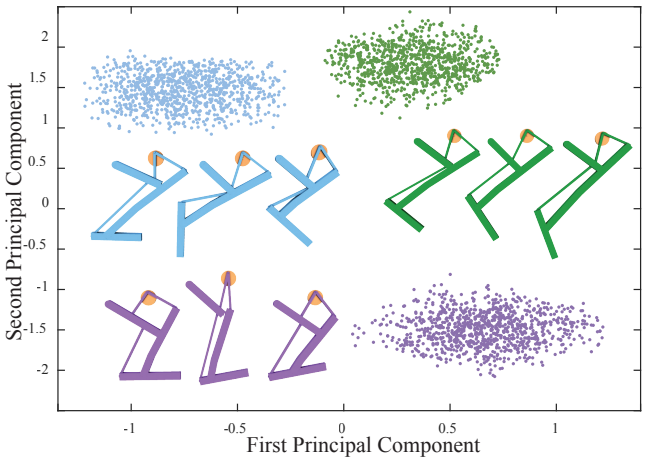


Figure 4: *We show a sampling of the learned linkage space, with examples from three different Gaussians in the mixture (color coded). As this is a high-dimensional space, we perform PCA on the learned space and show only the first two principal components.*

Another method that we employ to avoid over-fitting the data is Leave-p-out cross validation [Hastie et al. 2009], we perform this over multiple ps. This validation indicates how well the test samples fit the GMM model's probability-density function. It is noted that full covariance matrices $\boldsymbol{\Sigma}_m$ can lead to data over-fitting, we experimented with *only diagonal* covariance matrices, when number of Gaussians was twice as large as the number of mixtures in the GMM. However, for our problem using $6 - 12$ mixtures with full covariance matrices gave us the better results while still satisfying the above mentioned over-fitting checks. Finally, for a given linkage, we found that using $|\mathcal{D}| = 25,000$ samples sufficed.

Sampling by Inference We have learned a GMM over the space of kinematic parameters for a linkage, and now need a structured way to sample from the GMM for use in CMA. One advantage of this probabilistic approach is that the learned random variable parameters encode the joint probability distribution $P(\mathsf{X})$, where $\mathsf{X} = \{\mathsf{C}, \mathsf{K}\}$ and factorizes as $P(\mathsf{X}) = P(\mathsf{K}|\mathsf{C})P(\mathsf{C})$. Hence, this can be used to answer queries such as:

$$p(\mathbf{x} \mid \mathsf{c}_m = 1) = \mathcal{N}_m(\mathbf{x} \mid \Theta_m) \tag{4}$$

More intuitively, this means that it is highly likely that a given linkage parameter $\mathbf{x}$ belongs to the valid space and is modeled by the m^{th} Gaussian (a conditional distribution). Using similar inference queries for all valid m mixtures, we precompute M multivariate distributions modeled by the GMM. This gives us the desired linkage-configuration parametric function (equation 2). Next, in an automaton, the kinematic parameters of a linkage p_k are replaced by the following vector:

$$\mathbf{p}_k^* = [m, \mathbf{x}] \tag{5}$$

Here, m gives the mixture index and $\mathbf{x}$ (a bijective mapping to $\mathbf{p}_k$) are the multivariate Gaussian function parameters. When the m^{th} index is selected, we set $\mathsf{c}_m = 1$, which selects the corresponding precomputed Gaussian (Eq. 4). Since all the covariance matrices are symmetric positive definite, first we perform eigenvalue decomposition and extract the eigenvectors ($\mathcal{E}^m$) and eigenvalues ($\mathcal{A}^m$) of

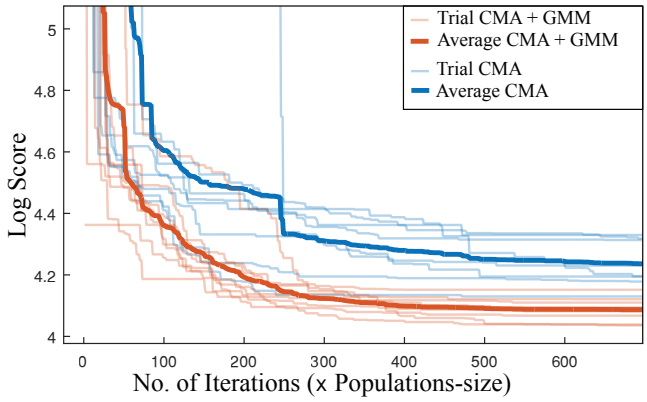


Figure 5: *Trial runs with CMA vs. CMA-GMM.*

$\mathbf{\Sigma}_m$. Then second, a sample is draw as:

$$\mathbf{p}_k = \boldsymbol{\mu}_m + (\mathbf{x}\sqrt{\mathcal{A}^m})\mathcal{E}^m \tag{6}$$

Since the mixture weights π_m give the probability that a sample from the valid configuration space belongs to Gaussian, it makes sense that Gaussians with higher probability are sampled more. Hence, the mixture-index parameters m are weighted by π_m and mapped. Figure 4 shows samples with random $\mathbf{x}$ inputs, drawn from three such Gaussians in the mixture.

Optimization Summary We precompute a database of linkage configurations, over which we learn an M-dimensional GMM. For each initial user design, we run CMA, but now in each iteration, CMA's internal multivariate Gaussian is sampled giving $\mathbf{p}_k^*$. Then, the linkage parameters $\mathbf{p}_k$ are discovered using $\mathbf{p}_k^*$ and Equation 6. This process is transparent to any other workings of CMA, so is a drop-in replacement. With this procedure, the walking optimization can quickly move between different Gaussians and can explore valid configuration spaces quickly and effectively. This strategy of optimization can also be interpreted as a discrete (mixture-component selection)-to-continuous (Gaussian) sampling.

6 Experiments

We validate our CMA-GMM improvements over many trials, and design five walking automata, for two of which we also create physical prototypes. As suggested by Auger and Hansen [2012], we generate $4 + \lfloor 3 \times log(N) \rfloor$ samples for every CMA iteration, where N is the number of parameters that are optimized in parallel. Statistics for all examples are listed in Table 1.

CMA vs. CMA-GMM To validate the proposed alternative data-driven sampling strategy for our linkages, we ran eight trails for each optimization strategy for the *Dog* automaton. Figure 5 shows a comparison of the convergence rates for CMA and CMA with GMM linkage sampling. We seen that, on average, CMA-GMM converges both faster and produces solutions with better scores—a trend that was confirmed in all examples that we considered.

Dog Our first example is a quadrupedal automata with dog-like appearance. Its four legs consist of 6 bars each, with the two front and hind legs having the same mechanical structure. While each leg can have a different phase, we enforce a symmetry relation between left and right legs to reduce the complexity of the design. Accordingly, the set of parameters exposed to the optimization consist of

Robot Name	# Components	Timing Mechanisms	# Params	CMA-GMM Iterations	Time (hrs)
Dog	34	4	20 (Purple)	450	3
			40 (Blue)	500	4
Grandpa Bot	20	2	13 (Purple)	350	2
			16 (Blue)	350	2
Lobster	18	2	17 (Purple)	450	2
			31 (Blue)	450	2.5
			31 (Green)	600	3
Gorilla	38	3	12	300	2
Giraffe	38	4	36	550	4.5

Table 1: *Statistics for all examples.*

14 structural and 4 velocity profile parameters for the two front and hind legs, as well as 4 phase offset, making for a total of 40 parameters. The initial design created by the user, and the starting configuration for the optimization, can be seen in Fig. 6a, left. While this starting configuration falls over immediately at the beginning of the simulation, our optimization automatically discovers a gait with a lowered center of mass (Fig. 6a, right), which increases stability and thus leads to a successful walking motion.

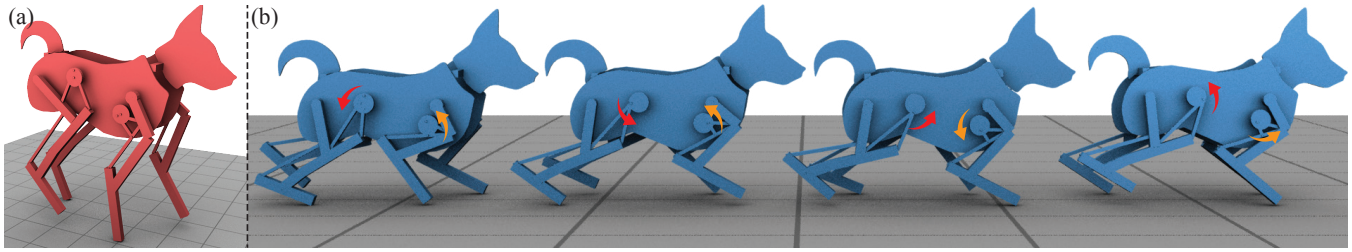
It is worth noting that the optimization of the structural parameters is essential for obtaining a successful gait: as can be seen in the accompanying video, optimizing only for the timing parameters, i.e., the 4 phase offsets and the 8 velocity profile parameters, leads to a very inefficient gait with the dog essentially moving in place.

Lobster Our second example, a lobster automata, consists of two legs with identical structure, each comprising 6 bars and 10 parameters. Again, the initial design, shown in Fig. 6b (*a*), is not able to walk. Optimizing for the full set of 20 structural and 10 timing parameters leads to a walking motion, but the gait is very dynamic with the center of mass shifting back and forth (see accompanying video). While this is not a problem in simulation, the various imprecisions introduced through simplifying assumptions and manufacturing make such dynamic motions more susceptible to instabilities. To obtain a less dynamic gait, we increased the weights of the smoothness and effort objective and introduced the height of the body as an additional parameter for the optimization. As a result, the optimization discovered a new, less dynamic gait that uses the body as a third leg.

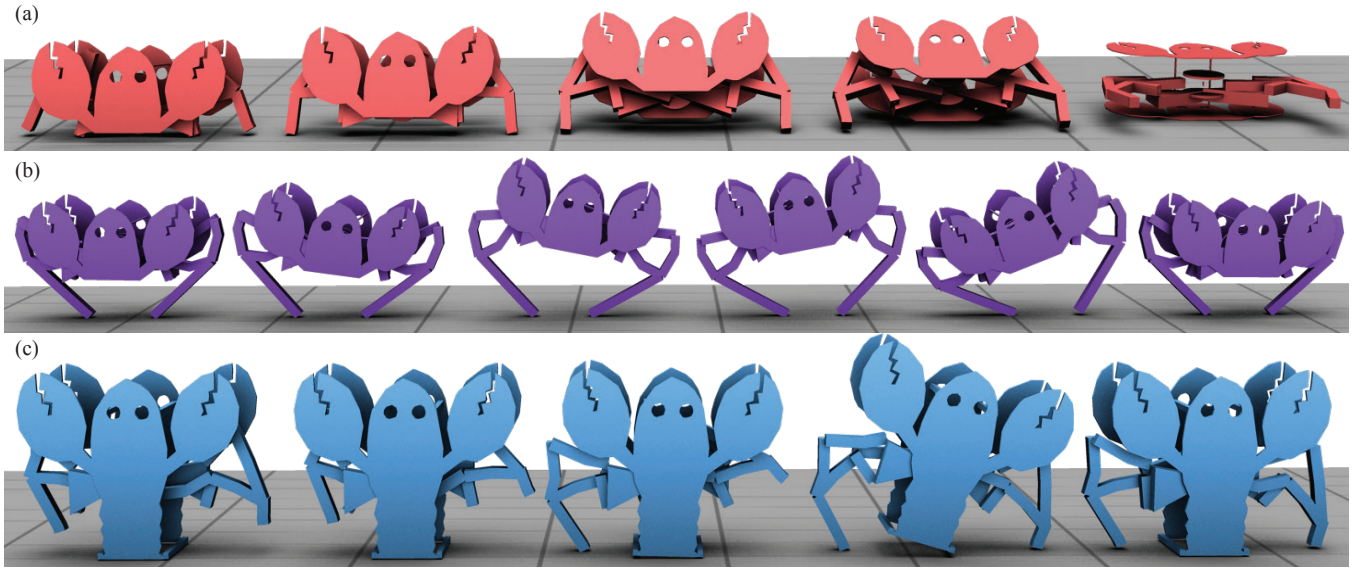
As an additional observation, we found that, when enforcing a symmetry relation between the two legs, the resulting motion was inefficient, i.e., led to a slow propulsion of the center of mass. By allowing for an asymmetric structure, we invite configurations in which the hind leg pushes while the front leg pulls, leading to a more efficient gait.

Grandpa-Bot, Gorilla, and Giraffe We designed three additional automata in simulation to further explore the behavior of our method. Our *Grandpa-Bot* is based on an example used by Coros et al. [2013]. Although conceived with the intent to walk, having no notion of the physics or geometry of locomotion, their kinematic design tool did not produce a walking motion for this character. While our initial design (Fig. 7a, top) was also not able to walk, the optimization discovered a parameter set that leads to a succesful and aesthetically pleasing gait (see Fig. 7a, bottom, and video).

The fourth example is our *Gorilla*-inspired autoamata (Fig 7b), we explore a three-legged design for which our optimization discovers a gait that alternates between swing-through and support for the outer legs and the middle leg, respectively.



(a) *Dog Model: (a) initial configuration that is unable to walk; (b) optimized automata*



(b) *Crab Model: a) the initial configuration is unable to walk; b) a dynamic walking motion is found if only the leg parameters are optimized; c) allowing the body to also change its shape, an alternate locomotion mode is discovered.*

Figure 6: *The two automata for which we fabricate results.*

Finally, we show our *Giraffe*-inspired automata. We explore how the optimization adapts to asymmetric design with a shifted center of mass. As can be seen in the video, with a long neck, the initial design for the automata falls over and hits the head (Fig. 7c, top). Our system is able to adapt the front linkages and make the automata walk with a smooth symmetric motion while balancing the heavy neck (Fig. 7c, bottom).

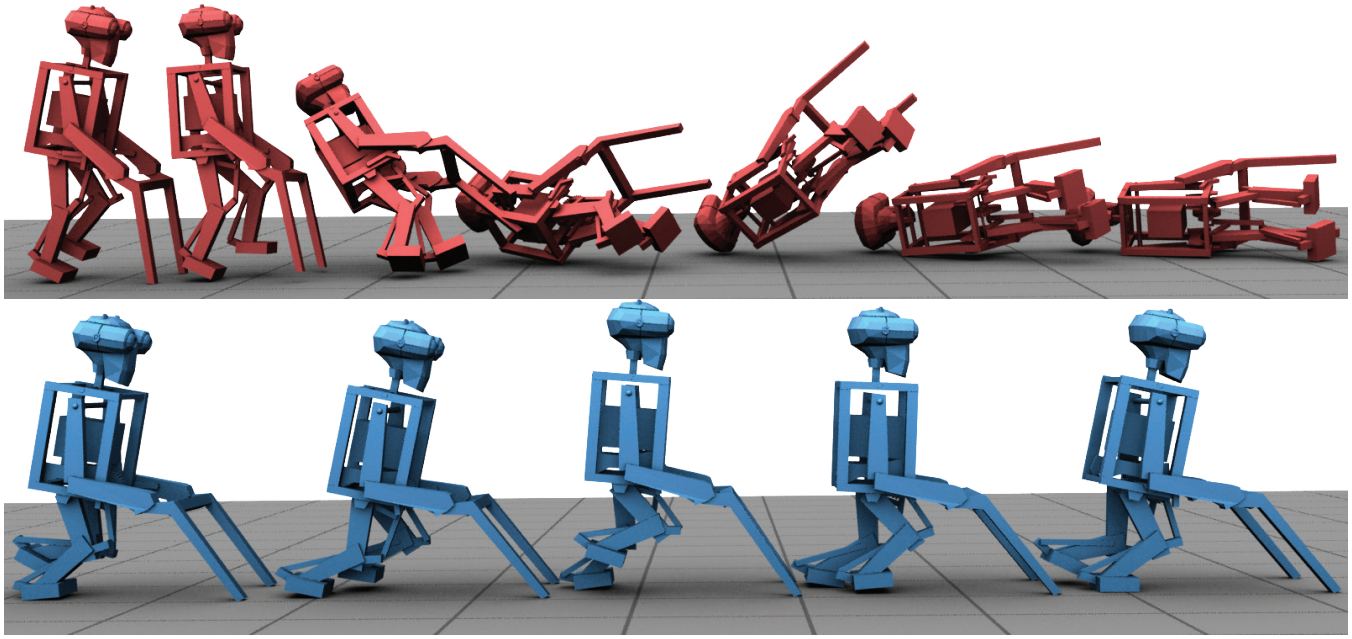
Physical Prototypes To validate the feasibility of our designs, we created physical prototypes for the *Lobster* and the *Dog* character, representative photographs of which are shown in Fig. 1. All body parts were laser-cut from a plywood material with 5mm thickness. We used metal bolts and quicklock rings for the pin joints as well as Dynamixel MX-64 servo motors and a Robotis CM-700 controller board for actuation. The resulting walking motion for these two prototypes can be observed in the accompanying video. Although our simulation—and especially the treatment of frictional contacts—is inevitably approximate, we observed agreement between the walking motion of the prototypes and their simulated counterparts. Finally, it should be noted that we show simulation results for geometrically more elaborate models to better portray the artistic intent of the characters. However, simplified geometries that are consistent with the fabricated prototypes were eventually used for optimization. While 3D printing the robots is technically possible (ignoring the motors), pin joints may break due to internal strains. Laser-cutting linkages and using metal pins proved to be sufficiently robust.

Robustness To investigate the robustness and convergence properties of our method, we reran the optimization for the dog character with three slightly perturbed initial guesses (by 5% in relative magnitude). As can be seen in the accompanying video, each run converges to a different solution, although each one results in a successful walking motion. We conjecture that this behavior is owed to the complexity of the problem and the randomized nature of our evolutionary optimization scheme.

7 Limitations And Future Work

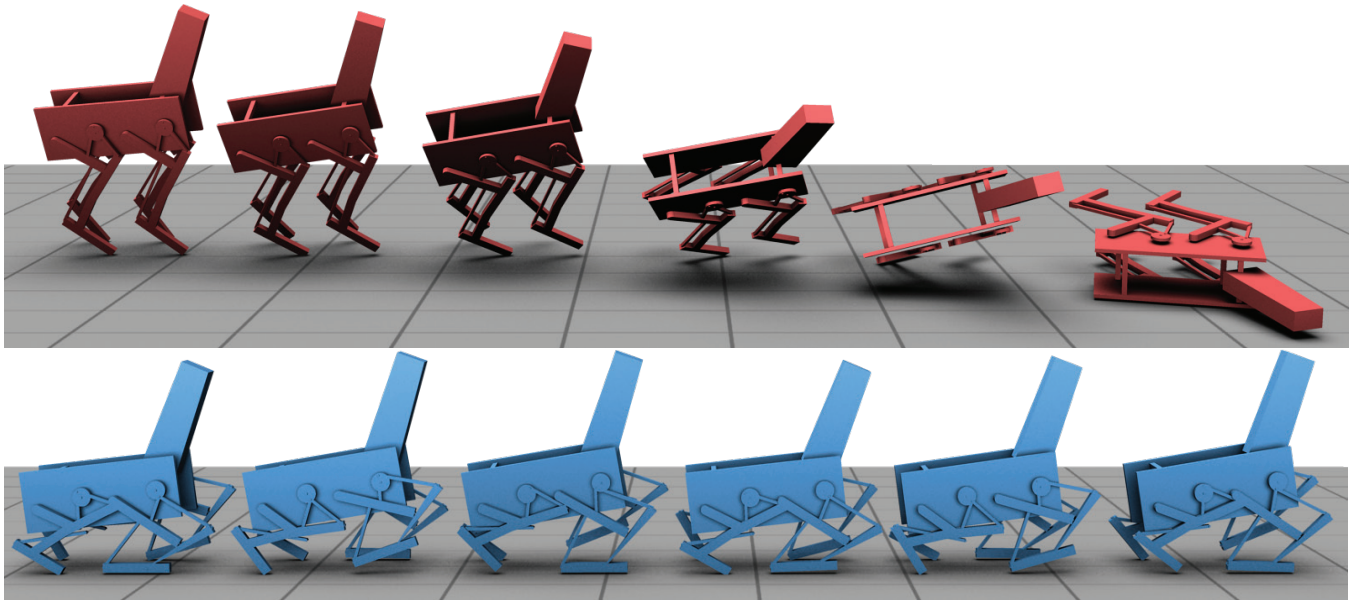
Our work succesfully demonstrates that computational design can create walking automata from initial non-walking designs. A physics-based simulation framework is used in conjunction with an intuitive set of objectives to measure the quality of the walking motions as the mechanical structure of the automata is automatically adapted. The framework takes advantage of learned valid linkage spaces to explore the parameters more efficiently. However, there are limitations to our approach and these are only the first steps.

Due to the non-linear nature of the problem, we adopted a stochastic sampling strategy to search for a global minima. However, experiments suggest that the optimization landscape exhibits a multitude of local minima. Therefore, we would like to experiment with continuation methods that incrementally increase the difficulty of the problem to increase the chance of finding global optima. One way to do this, for instance, is to employ external hand-of-god forces that initially support the weight of the mechanism but are progressively



(a) *Grandpa-Bot: A human-like automata that exploits a walker for additional support.*

(b) *Gorilla: A three-legged design with each leg alternating between support and swing-through phase.*



(c) *Giraffe: Heavy neck makes the optimization adapt the automata-linkages accordingly.*

Figure 7: *Various automata produced by our system. In red are inital configurations, while blue are optimized walking automata.*

made weaker until they eventually vanish altogether. This strategy is akin to a child learning to ride a bicycle, and could allow the optimization process to more thoroughly explore the parameter space, as fewer trials would lead to failures.

The complexity of the optimization leads to longer convergence rates, and so the user cannot directly interact with the system during the gait design and optimization process. Hence, we would also like to investigate methods that allow users to control the gaits at a finer level of granularity. This could be investigated with additional style objective terms, or by pre-computing a library of walking templates to use as a starting point for the optimization.

To decrease the discrepancy between simulated and fabricated motion, we could accurately estimate the force output of the motors, as well as the static and dynamic friction coefficients of the fabricated material and floor — currently these are generic. To improve motion quality in general, we could add on-board sensors and additional actuators to monitor and control the gait cycle dynamically as the robot moves across different surfaces.

Finally, our simulation has no knowledge of intra- or inter-linkage collisions, and so when fabricating the robots these conflicts are resolved manually by stacking linkage bars along the crank rotation axis. For example, the legs of our fabricated lobster are broader than those in the simulation. For novice users, these collision conflicts must be automatically resolved, and is an area of future work.

Acknowledgments

A warm thank you to Mélina Skouras and Rocco Ghielmini for fabricating the robots, without which this paper would not be possible.

References

AUGER, A., AND HANSEN, N. 2012. Tutorial CMA-ES: Evolution strategies and covariance matrix adaptation. In *Proceedings of ACM GECCO*, 827–848.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4.

BACHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. 33*, 4.

BISHOP, C. M., ET AL. 2006. *Pattern Recognition and Machine Learning*, vol. 4. Springer.

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 Trans. Graph. 31*, 6, 130:1–130:8.

CEYLAN, D., LI, W., MITRA, N. J., AGRAWALA, M., AND PAULY, M. 2013. Designing and fabricating mechanical automata from mocap sequences. In *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. Graph. 30*, 4, 35:1–35:10.

COROS, S., KARPATHY, A., JONES, B., REVERET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, 59:1–59:12.

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

GEHRING, C., COROS, S., HUTTER, M., BLOESCH, M., HOEPFLINGER, M., AND SIEGWART, R. 2013. Control of dynamic gaits for a quadrupedal robot. *IEEE ICRA*.

GEHRING, C., COROS, S., HUTTER, M., BLOESCH, M., HOEPFLINGER, M., AND SIEGWART, R. 2014. Towards automatic discovery of agile gaits for quadrupedal robots. *IEEE ICRA*.

GEIJTENBEEK, T., VAN DE PANNE, M., AND VAN DER STAPPEN, A. F. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Trans. Graph. 32*, 6.

HANSEN, N. 2006. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 75–102.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., HASTIE, T., FRIEDMAN, J., AND TIBSHIRANI, R. 2009. *The elements of statistical learning*, vol. 2. Springer.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Trans. Graph. 29*, 4, 129:1–129:8.

LEE, S., YOSINSKI, J., GLETTE, K., LIPSON, H., AND CLUNE, J. 2013. *Evolving gaits for physical robots with the hyperneat generative encoding: The benefits of simulation*. Springer.

MCKAY, M. D., BECKMAN, R. J., AND CONOVER, W. J. 1979. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2, 239–245.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4, 81:1–81:10.

SCHWARZ, G., ET AL. 1978. Estimating the dimension of a model. *The annals of statistics 6*, 2, 461–464.

SIMS, K. 1994. Evolving virtual creatures. In *ACM SIGGRAPH*, 15–22.

TAN, J., GU, Y., TURK, G., AND LIU, C. K. 2011. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, ACM, SIGGRAPH '11, 58:1–58:12.

TAN, J., GU, Y., LIU, C. K., AND TURK, G. 2014. Learning bicycle stunts. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, ACM.

THOMASZEWSKI, B., COROS, S., GAUGE, D., MEGARO, V., GRINSPUN, E., AND GROSS, M. 2014. Computational design of linkage-based characters. *ACM Trans. Graph. 33*, 4, 64:1–64:9.

UMETANI, N., KOYAMA, Y., SCHDMIT, R., AND IGARASHI, T. 2014. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph. (Proc. SIGGRAPH) 34*, 4.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, 60:1–60:8.

XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Trans. Graph. 31*, 4, 57:1–57:10.

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