Rig Animation with a Tangible and Modular Input Device

Oliver Glauser[1] Wan-Chun Ma[1] Daniele Panozzo[1,2] Alec Jacobson[3] Otmar Hilliges[1] Olga Sorkine-Hornung[1]
[1]ETH Zurich [2]New York University [3]Columbia University

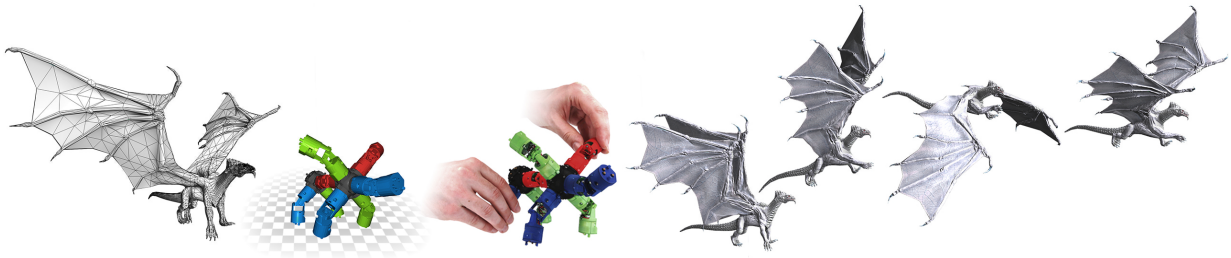


Figure 1: *Left to right: Taking a rigged 3D character with many degrees of freedom as input, we propose a method to automatically compute assembly instructions for a modular tangible controller, consisting only of a small set of joints. A novel hardware joint parametrization provides a user-experience akin to inverse kinematics. After assembly the device is bound to the rig and enables animators to traverse a large space of poses via fluid manipulations. Here we control 110 bones in the dragon character with only 8 physical joints and 2 splitters. Detailed pose nuances are preserved by a real time pose interpolation strategy.*

Abstract

We propose a novel approach to digital character animation, combining the benefits of tangible input devices and sophisticated rig animation algorithms. A symbiotic software and hardware approach facilitates the animation process for novice and expert users alike. We overcome limitations inherent to all previous tangible devices by allowing users to directly control complex rigs using only a small set (5-10) of physical controls. This avoids oversimplification of the pose space and excessively bulky device configurations. Our algorithm derives a small device configuration from complex character rigs, often containing hundreds of degrees of freedom, and a set of sparse sample poses. Importantly, only the most influential degrees of freedom are controlled directly, yet detailed motion is preserved based on a pose interpolation technique. We designed a modular collection of joints and splitters, which can be assembled to represent a wide variety of skeletons. Each joint piece combines a universal joint and two twisting elements, allowing to accurately sense its configuration. The mechanical design provides a smooth inverse kinematics-like user experience and is not prone to gimbal locking. We integrate our method with the professional 3D software Autodesk Maya® and discuss a variety of results created with characters available online. Comparative user experiments show significant improvements over the closest state-of-the-art in terms of accuracy and time in a keyframe posing task.

Keywords: tangible input, skeletal deformation, animation system

SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925909

1 Introduction

At the heart of interactive character animation lies a contradiction. Animators must draw from a large space of poses to breath life and depth into the character, but animators also wish to do so with a very small number of control parameters. Standard rigging tools offer sophisticated ways to span a large space of shape deformations, but their parameters are endless, and indirectly mapping them to the keyboard and mouse makes existing interfaces cumbersome and difficult to learn. Recent tangible input devices promise direct and natural manipulation, but at the cost of either grossly simplifying the pose space or of accepting complex and bulky physical setups. In contrast, we present a novel software/hardware approach co-designed to help animators traverse a large space of poses via *fluid* manipulation of a tangible controller.

Specifically, we contribute: (*a*) a novel hardware design, overcoming major limitations in prior work via a novel physical angle parametrization; (*b*) an algorithm to compute a device configuration and instructions to assemble it, using only a small set of modules; and (*c*) a method to bridge the disparity between the input device's *few* degrees of freedom to the character's *many* control parameters. Our contribution allows users to move beyond static keyframing towards fluid animation of a variety of complex characters with arbitrary topologies.

The approach is general and can directly control professional-grade rigs without relying on a specific shape deformation algorithm. It only relies on a rig with a skeletal structure and a set of sample poses as input. From this input rig, the most important control parameters are extracted. Using a given hardware kit, the algorithm then computes a device configuration and a mapping of physical rotations onto the extracted rig control parameters. The algorithm is guided by an objective functional that measures reachability in a set of sparse sample poses. Manipulating the physical proxy induces a new pose in the rig. To preserve pose details, the rest of the rig controls are synthesized from the sample poses via pose space interpolation in real-time (see Fig. 1).

We demonstrate that complex characters, often containing hundreds of bones, can be controlled with a compact tangible device consisting of much fewer pieces. Furthermore, our method is integrated directly into Autodesk's Maya® 3D animation software, emphasiz-

ing its practical applicability. We illustrate the method's utility by downloading and animating a variety of rigs without further modification. Results from a user study compare favorably to the hardware design of [Jacobson et al. 2014b] both in terms of accuracy and posing time, providing an average speed-up of $2\times$. Finally, we qualitatively show that our method enables more fluid control, of more complex characters, with less complex physical devices.

2 Related work

Real-time character articulation is a core subject of computer graphics, receiving much attention in the literature. We refer to a recent survey of skinning techniques [Jacobson et al. 2014a], and turn our attention to methods of reducing deformation control, and specifically to those assuming a skeletal rig. We discuss comparisons to existing joint hardware designs in Section 3.1.

Inverse kinematics. The well-studied problem of reducing the control of a complicated *rig* to sparse user input is typically approached by trying to distill high-level controls from a given set of low-level controls. For example, traditional inverse kinematics (IK) abstracts an arbitrarily complicated kinematic tree of rotational or translational joints into just the position of leaf nodes in the tree: end-effectors in robotics, or more saliently to character animation, the head, hands, and feet.

Classic IK ignores the effect of the kinematic tree's implied deformation of the character's surface geometry. So-called MeshIK approaches remedy this by replacing the usual "joint work" with a geometric energy capturing the quality or physical plausibility of the deforming surface geometry [Sumner et al. 2005]. MeshIK (and all mesh-based deformation methods [Botsch and Sorkine 2008]) reduce control from the individual mesh vertex level to the placement of a few user-specified handles. Under the framework of linear blend skinning, MeshIK can achieve real-time performance [Jacobson et al. 2012]. Given sparse user constraints, skeletal skinning transformations are optimized according to their impact on the shape's surface.

Although the user effort is reduced by the above techniques, so is the space of possible deformations: linear blend skinning cannot feasibly represent high-frequency effects such as muscle bulging.

Pose and rig space. Pose space interpolation [Buck et al. 2000; Lewis et al. 2000] has been successfully applied to character animation. These example based techniques leverage both the desired shapes, sculpted by trained modeling artists, and the abstract *pose space* which is formed by the rig controls. In particular, deformations may be interpolated and extrapolated according to associated lower-dimensional abstract poses. We take advantage of such pose space interpolation techniques, specifically [Buck et al. 2000], to animate the rig with the reduced degrees of freedom from our input device, since it is impossible to directly control a complex rig with hundreds of bones through our modular input device.

Driving an arbitrary character rig has also been considered in the context of physical simulation. For example, a physical simulation's deformation of a character's geometry can be projected onto a generic rig [Hahn et al. 2012]. Rather than mapping sparse user controls, this maps a potentially very dense physical simulation to a dense rig. Though distinct in motivation, our rig reduction bares similarity to [Hahn et al. 2012], but we require only rig evaluation without resorting to finite differencing.

Input devices. Despite their ubiquity, 2D mouse and discrete keyboard interfaces struggle to control multi-modal 3D entities such as rotations or translations [Jacob et al. 1994]. Previous works have overcome this via sketching [Öztireli et al. 2013; Hahn et al. 2015; Guay et al. 2013; Guay et al. 2015], but manipulation is still indirect through a projection onto 2D device coordinates. Following [Ishii and Ullmer 1997], recent physical input devices demonstrate technologies for direct manipulation. The software of [Jacobson et al. 2014b] does not consider gross mismatches between the virtual character's skeleton and their modular input device. Crucially, it assumes a simple skeleton and a matching input device that has as many physical joints as virtual bones in the skeleton. In reality, rigs are very complicated, causing physical configurations to grow too bulky quickly. Furthermore, determining an appropriate configuration of components is far from trivial. The "dinosaur input device" of [Knep et al. 1995] also had to address the problem of having too many rig parameters for a given input device. Their solution was to map a short sequence of one-dimensional measurements from the input device (say, along a T-rex's tail) to a chain of rotations along a virtual bone chain. In contrast, our proposed mapping is widely general. Our design integrates rotational degrees of freedom (DoFs) per joint but still allows to separate translation and rotation, which is preferable and in accordance with prior findings ([Zhai and Milgram 1998; Masliah and Milgram 2000]).

The recent flexible bending input devices of [Chien et al. 2015] and [Nakagaki et al. 2015] consist of a chain of single DoF elements. They do not directly offer the degrees of freedom needed for character animation, since they do not support axial rotational DoFs and lack the modularity required to control a large variety of characters.

Other physical input systems exist, such as computer vision systems for 6-DoF tracking [Held et al. 2012; Shiratori et al. 2013] and specialized dolls for humans and hands [Esposito et al. 1995; Feng et al. 2008; Yoshizaki et al. 2011; Celsys, Inc. 2013; Achibet et al. 2015]. See [Jacobson et al. 2014b] for a more thorough history of tangible animation input devices.

Motion capture and retargeting. The human body itself becomes an input device via performance capture. The mapping of an actor's performance to a character can be non-trivial in the presence of proportional disparities (a short actor controlling a giant [Gleicher 1998]) or gross differences in the source and target (a human actor controlling a sheep [Rhodin et al. 2014; Fender et al. 2015]). However, the input device is fixed and known: hands are always connected to arms connected to torsos, etc.

The more general problem of animation retargeting considers mapping the deformation of one object onto a similar object [Sumner and Popović 2004; Baran et al. 2009]. Most relevant to our work is the recent effort to map the deformation of a simplified skeleton (e.g. from a control optimization) to a character controlled by a complex rig [Holden et al. 2015]. Similarly, Jin et al. [2015] consider mapping skeleton deformations to a sub-skeleton or vice-versa. In contrast, our work optimizes a quality metric for an input device to rig mapping as means to *propose* a particular input device configuration. That is, the very set of control parameters is determined by our optimization.

Skeleton simplification. To propose a modular device configuration, we rely on the ability to reduce a combinatorial tree of rig deformers to a simple geometric skeleton closely matching a constellation of device components. Pure skeleton simplification has been considered previously in the context of level-of-detail animations [Ahn and Wohn 2004; Savoye and Meyer 2008] and curved skeleton extraction [Au et al. 2008; Tagliasacchi et al. 2012]. These methods either assume a skeleton or rely on analysis of the shape's surface geometry. We consider rigs composed of arbitrary deformers, some of which have no associated geometric "bone". We avoid relying too heavily on specific character surface geometry, as it could change during animation prototyping or may simply be unavailable. Instead, in the spirit of pose space deformation [Lewis

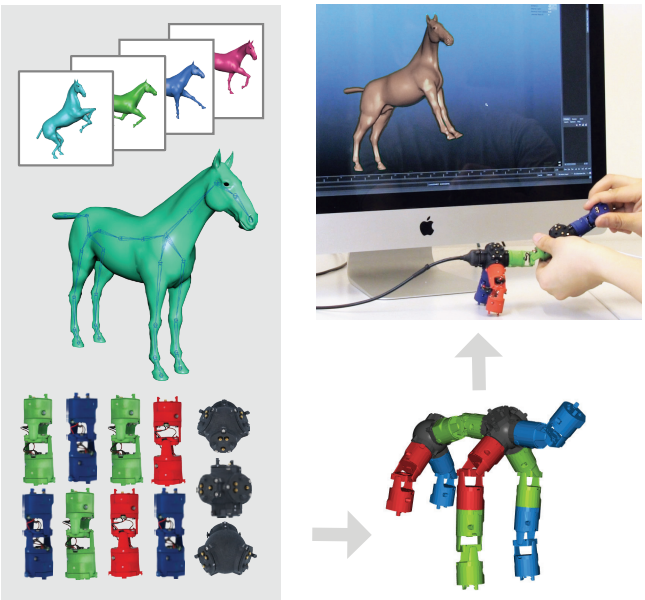


Figure 2: *Illustration of our pipeline from input character to fluid tangible animation. The horse has 29 bones, controlled by 8 joints.*

et al. 2000], our optimization assumes a small set of sample *rig* poses.

Example-based methods are a popular means to solving a problem that is in some sense the inverse of ours: Given a full animation of a character's geometry, determine a skeletal rig [Schaefer and Yuksel 2007; Le and Deng 2014]. Huang [2015] recently demonstrated automatic generation of plausible example poses. Our method would immediately benefit from such example poses.

3 Method

Our approach provides a natural interface for posing and animating complex 3D characters using only a small set of physical controls, allowing novice users and experts to create animation sequences. The proposed solution consists of hardware and software contributions, designed in unison and complementing each other.

To gain an intuition, consider the following example: the user provides a rigged 3D character with a sparse set of sample poses (readily available online). Furthermore, the user indicates the kit (number of joints and splitters) to use. We then analyze the rig and the poses, identifying the DoFs with the most influence on pose reachability, weighted by the amount of controlled surface. Optimizing for direct control of these most important nodes, and using only the available parts, a device configuration and assembly instructions are computed. We solve a challenging, discrete assignment problem: finding a configuration of modular hardware pieces that maximizes coverage of a large pose space. In addition to the physical setup we also compute a mapping between sensed rotations and the rig's parameters, which ultimately control the character's pose.

After assembly the physical device is bound to the virtual rig and user inputs are mapped onto the rig. Manipulating the device induces a similar deformation onto the 3D character. In most cases the physical configuration has significantly fewer DoFs than the rig (see Fig. 2). To maintain expressiveness and add details back into the resulting motion, we use a pose space interpolation scheme to synthesize detailed pose nuances (cf. Fig. 1).

3.1 Hardware

Our design, inspired by [Jacobson et al. 2014b], follows a modular approach, decomposing the control structure into joints that measure 3D rotations, and splitters, which allow for branching. This design enables dynamic rearrangement of the parts into arbitrary topologies. However, one of the main limitations of [Jacobson et al. 2014b] is due to the mechanical joint design. Relying on twist-bend-twist joints (Fig. 3, left) the user has to decompose rotations into Euler angles, making the device prone to gimbal lock: moving the endpoint of a single joint between two coordinates on a unit sphere typically cannot be done by tracing the shortest path (i.e., the geodesic) but rather requires a sequence of individual rotations, with the endpoint zig-zagging over the surface (see Fig. 4).

Mechanical design. We propose a new joint design to overcome this limitation and provide a much improved user-experience by allowing for smooth tracing of geodesics – both for individual joints and chains of joints. The design is based on the Universal (or Cardan) joint augmented with two infinite twisting rings. The joint itself consists of a pair of hinges, oriented orthogonally to each other and connected via a ring shaped cross-shaft (Fig. 3, right). This design enables the joint to bend in any direction and maintains accuracy with an angular error below 0.5°. Fig. 4 illustrates the effect of our design when tracing geodesics on a sphere. Our design allows for much smoother and faster trajectories and a shorter path compared to [Jacobson et al. 2014b]. To enable rotation about the shaft itself, we incorporate twisting rings at either end of the joint. While a single twist would suffice, we experimentally found that two make user interaction smoother.

The joints in [Jacobson et al. 2014b] also featured twist rings but were limited in their rotation range due to the need for wired connections. We overcome this limitation using a pair of slip rings (inset), conducting power and transmitting electrical signals while allowing for infinite rotations. We use the same splitter geometries as [Jacobson et al. 2014b].

Sensing and communication. To recover the joint's rotations we utilize magnets and Hall effect sensors. The two bending angles are measured by embedding a passive axial magnet into the cross-shaft. This generates a local magnetic field and its orientation is sensed by a 2D Hall sensor, located at the base of the outgoing shaft. Analogously, we sense twist by leveraging diametrical magnets and 1D Hall sensors (see Fig. 5).

Each joint contains three sensors and two micro-processors, which reconstruct angles from sensor readings, serialize them and transmit downstream to the host computer. While the slip rings have desirable mechanical properties, their electrical connectivity is less reliable than wire-based solutions. To deal with connection dropouts

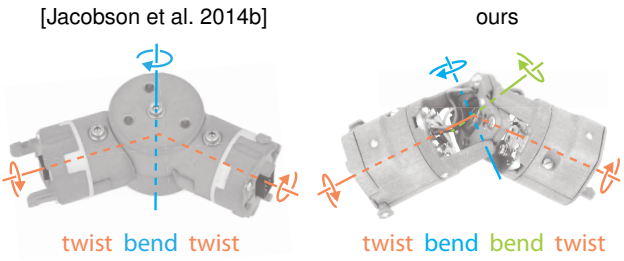


Figure 3: *Comparison of joint designs. Left: twist-bend-twist design of [Jacobson et al. 2014b]. Right: our proposed joint design.*

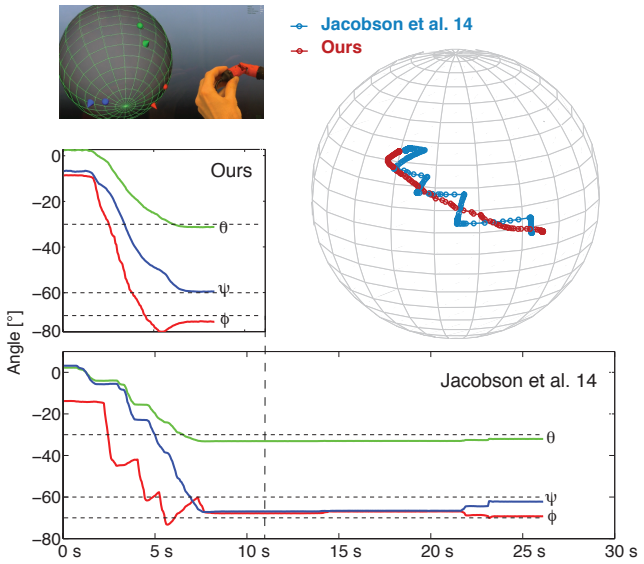


Figure 4: *Our physical angle parametrization allows for direct tracing of geodesics. The joints of [Jacobson et al. 2014b] force angular decomposition of rotations, resulting in a zig-zag pattern. Ours is also significantly faster in reaching the target position.*

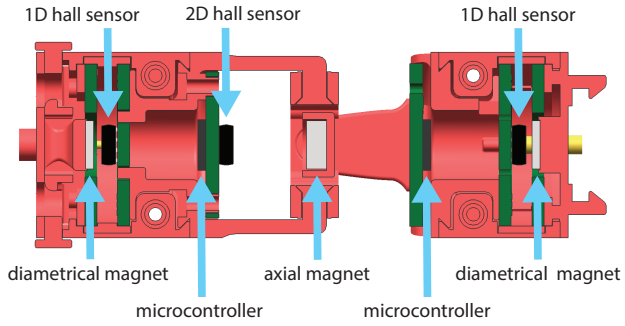


Figure 5: *Joint cross-section showing passive diametrical and axial magnets, Hall sensors and micro controllers.*

we designed a fault tolerant protocol, robust to random disconnects and missing packages, allowing us to reconstruct the topology of the device consistently. For details we refer to Appendix A.

3.2 Rig retargeting

Tangible input devices (e.g., [Esposito et al. 1995; Knep et al. 1995; Yoshizaki et al. 2011; Celsys, Inc. 2013; Jacobson et al. 2014b]) promise natural manipulation of 3D characters. However, because they map degrees of freedom directly from device to virtual character, these are limited to controlling simple skeletons. In reality animation rigs are very complicated and have tens to hundreds of degrees of freedom (see Fig. 9), making direct control via tangible input devices impractical. We propose a set of algorithms to overcome this inherent limitation.

In professional rigs, a small number of degrees of freedom control large surface areas and hence predominantly define the character's pose. The rest of the rig then adds more subtle details (see Fig. 7). This observation is crucial to our approach. We propose to control these most important nodes directly using our hardware, while driving the remaining DoFs implicitly. To accomplish this we have developed methods to compute an optimized hardware configuration, a mapping that induces realistic poses of the character, and an interpolation scheme to add lost expressiveness at animation time.

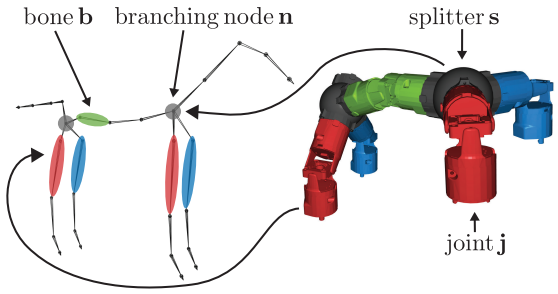


Figure 6: *Our algorithm converts a rig of bones connecting nodes (left) to a device consisting of joints and splitters (right).*

At the heart of our algorithm is an objective functional that measures how well a given input character rig $\mathcal{C}$, with a set of sparse sample poses $\mathbf{P}^{\mathcal{C}}$, can be approximated by a device configuration $\mathcal{D}$. We define the pose error as the L^2 distance between the position of rig nodes, resulting in the following optimization problem:

$$\underset{\mathcal{D},\mathbf{P}^{\mathcal{D}}}{\arg\min} \sum_i \mathbf{W} \left(F(\mathcal{C}, \mathcal{M}(\mathbf{p}_i^{\mathcal{D}})) - F(\mathcal{C}, \mathbf{p}_i^{\mathcal{C}}) \right)^2, \qquad (1)$$

where $F(\mathcal{C}, \mathbf{p})$ is the forward kinematic function that returns the position of the nodes of $\mathcal{C}$ in the pose $\mathbf{p}$ and $\mathbf{W}$ is a diagonal matrix weighting each node. A node's weight is proportional to the surface area which it directly controls. Each node also moves its connected subtree: by subtracting the area of each subtree, we ensure that each part of the surface is considered only once. Minimizing the energy in Eq. (1) computes $\mathcal{D}$, which is an injective assignment of hardware joints $\mathbf{j}$ to bones $\mathbf{b}$, and splitters $\mathbf{s}$ to branching nodes $\mathbf{n}$ of the rig (see Fig. 6). This also induces a retargeting function $\mathcal{M}$ which converts the device poses $\mathbf{p}^{\mathcal{D}}$ to rig poses $\mathbf{p}^{\mathcal{C}}$ by relating local joint rotations directly to bones. Generally speaking, there are more bones than joints and all bones without joint assignment move rigidly following the forward kinematic chain during fitting. Later these will be controlled by the pose interpolation scheme, discussed below. The device configuration $\mathcal{D}$ uniquely determines how to assemble the input device, and we visualize it as a set of assembly instructions, presented to the user as a 3D rendering (Fig. 6, right).

Evaluating Eq. (1) is fast since it only compares the positions of the rig nodes. However, finding the assignment $\mathcal{D}$ that spans the largest pose space is very challenging since it contains a discrete element: deciding which pieces of hardware to use and how to connect them.

To make the assignment problem computationally tractable, we propose an iterative algorithm with two alternating phases: First, we assign all available joints $\mathbf{j}$, ignoring all branching nodes in the rig. Second, we assign splitters $\mathbf{s}$ to branching nodes $\mathbf{n}$. Since the last step may remove already assigned joints, we repeat this procedure until there are no more joints or splitters left.

Fig. 7 illustrates the effect and importance of our algorithm. We plot pose error (measured in Eq. (1)) as function of the hardware kit size. Initially the error is reduced by 90% once reaching 5 joints and by 98% with 10 joints. However, the remaining 17 DoFs only contribute marginally to the reachability of poses. Hence, a full device would be unnecessarily bulky and require a large set of parts.

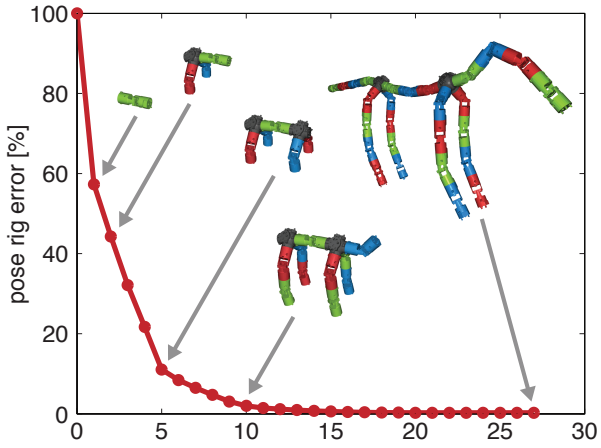


Figure 7: *Influence of joint number in the kit on pose accuracy.*

Joint assignment. The first phase of the algorithm assigns all available joints in the kit, maximizing $\mathcal{D}$'s span of the pose space. This assignment is computed in a greedy way: we try to assign a joint $\mathbf{j}$ to each unassigned bone $\mathbf{b}$ and for each assignment we minimize Eq. (1), which reduces to a smooth nonlinear optimization (details in Appendix B). We then pick the assignment with the lowest energy and we add it to $\mathcal{D}$. This procedure is repeated until all joints are assigned. The resulting sparse assignment $\mathcal{D}$ inherits the connectivity of the original rig $\mathcal{C}$: intuitively, the algorithm computes a sparse assignment, and then collapses all unassigned edges.

Splitters assignment. The second phase of the algorithm assigns splitters to branching nodes. The branching of the input rig can be arbitrary both in terms of valence and geometry. This makes a straightforward application of the above assignment strategy infeasible, since we only want to use a limited number of splitters with fixed valences and geometries.

Instead of greedily assigning splitters, we propose a global strategy based on integer linear programming (ILP) to optimally assign available splitters to branching nodes. The program is augmented with linear constraints that encode hierarchical dependencies and physical feasibility. Intuitively, we seek to globally minimize the mismatch in valence and geometry of branching nodes and splitters, weighted by the impact on the total pose error of the downstream assignment. For example, sometimes it may be acceptable to prune an entire subtree of the rig if its influence on the final poses is low. More formally we write:

$$\arg\max_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x}, \tag{2}$$

where $\mathbf{c}$ is the gain of assigning a specific splitter $\mathbf{s}$ to a specific branching node $\mathbf{n}$. The corresponding gain is defined as the reduction in energy (Eq. (1)) when replacing all fully rigid (not assigned with splitter $\mathbf{s}$) outgoing branches of $\mathbf{n}$ with non-rigid bones up to the next branching node. And $\mathbf{x}$ is the vector of binary variables that encode the assignment of splitters to branching nodes. To compute $\mathbf{c}$ and $\mathbf{x}$ we perform the following procedure.

First, we enumerate all possible combinations of branching nodes $\mathbf{n}$ in the rig and all splitters $\mathbf{s}$. Branching nodes $\mathbf{n}$ have one inlet $\mathbf{g}^n$ and 2 or more outlets $\mathbf{g}_i^{\mathbf{n}}$. Each of the outlets has an orientation, represented as rotation from $\mathbf{g}^{\mathbf{n}}$ to $\mathbf{g}_i^{\mathbf{n}}$. The splitter also has one inlet $\mathbf{g}^{\mathbf{s}}$ and outlets $\mathbf{g}_j^{\mathbf{s}}$, where generally the valence and orientations differ with respect to $\mathbf{n}$. See Fig. 8 for an illustration.

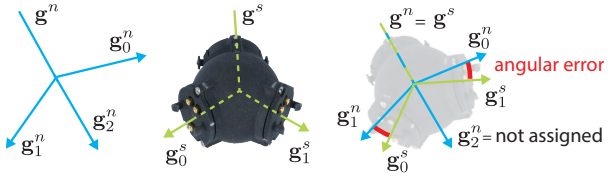


Figure 8: *Branching node $\mathbf{n}_0$ with outlets $\mathbf{g}_0^n$, $\mathbf{g}_1^n$ and $\mathbf{g}_2^n$ (left). Candidate splitter $\mathbf{s}_0$ with outlets $\mathbf{g}_0^s$ and $\mathbf{g}_1^s$ (middle). Assignment $\mathbf{x}_0$, aligning the pair $\mathbf{n}_0$ and $\mathbf{s}_0$ (right).*

Second, for each $(\mathbf{n}, \mathbf{s})$-pair (as in Fig. 8, left and middle) we map the inlet and enumerate all permutations of outlet assignments. For each such mapping we find the best possible orientation for the splitter $\mathbf{s}$ via [Kabsch 1976] The resulting assignment is then used to compute the gain via Eq. (1), comparing to a fully rigid configuration. Note that we set the remaining rotational offset as rest-pose rotation on the outgoing physical joint. Furthermore, we filter out assignments where matched gates have a remaining angular error exceeding 60° (Fig. 8, right).

Physical validity. We augment the ILP in Eq. (2) with a set of linear equality constraints, ensuring that for each splitter $\mathbf{s}_j$ and assignment vector $\mathbf{x}$ only one entry is set to 1. Let $\mathcal{A}(\mathbf{s}_j)$ be the set of assignments which contain splitter $\mathbf{s}_j$, then:

$$\sum_{i \in \mathcal{A}(\mathbf{s}_j)} \mathbf{x}_i \leq 1, \ \ \forall j, \tag{3}$$

enforces that no splitter is assigned more than once. Analogously, let $\mathcal{A}(\mathbf{n}_j)$ be the set of assignments which contain branching node $\mathbf{n}_j$, then:

$$\sum_{i \in \mathcal{A}(\mathbf{n}_j)} \mathbf{x}_i \leq 1, \ \ \forall j, \tag{4}$$

precludes double assignments to nodes.

As previously noted, often it is the case that the valence of $\mathbf{n}$ and $\mathbf{s}$ differs, resulting in configurations of $\mathcal{D}$ where entire subtrees of the rig $\mathcal{C}$ are not present. The following set of hierarchical constraints ensures that no splitters are wasted on assignments to such "virtual" branches and that they do not contribute to the gain:

$$\sum_{i \in \mathcal{A}(\mathbf{n}_c)} \mathbf{x}_i \leq \sum_{k \in \mathcal{A}(\mathbf{n}_p, m)} \mathbf{x}_k, \tag{5}$$

for all nodes $\mathbf{n}_p$,$\mathbf{n}_c$ s.t. $\mathbf{n}_p$ is the parent of $\mathbf{n}_c$ and $\mathbf{n}_c$ is connected to the chain of the m-th outlet of $\mathbf{n}_p$. The notation $\mathcal{A}(\mathbf{n}_p, m)$ denotes the set of assignments that contains $\mathbf{n}_p$ and have its m-th outlet used, i.e., the branch attached to the m-th outlet is not "virtual".

Implementation details. The rig reduction and hardware interface are implemented in C++ using Eigen, and we use Gurobi [Gurobi Optimization, Inc. 2015] to solve the ILP program Eq. (2) with constraints (3,4,5). The solver may return several solutions, for example having multiple identical splitter outlet assignments for one node. We favor the solution with the lowest average rotational error of the splitters, meaning that the geometry of the solution better approximates the rig geometry.

Typically hardware kits only contain few splitters. Hence it is likely that there are remaining, unassigned branchings. Obviously, joints can be seen as splitters with a single outlet, allowing us to cover at least one branch for such cases. This is realized by assigning "phantom" splitters of valence 1 to the kit while solving the ILP. These are simply removed before generating the assembly instructions, leaving only the attached joint (cf. Fig. 9, bottom row).

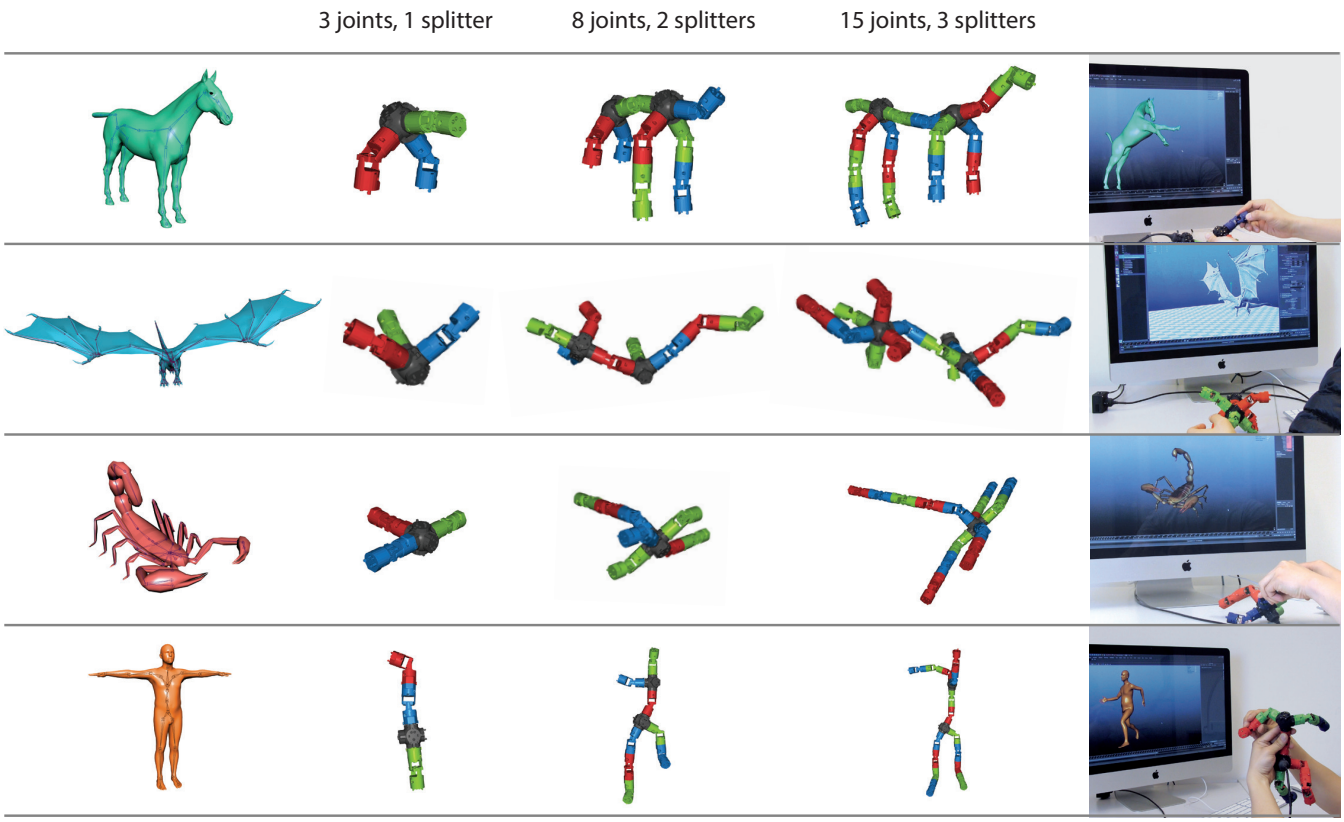


Figure 9: *Depending on the available kit, device build instruction plans with different complexity are generated by our algorithm. Note that the models have much higher degrees of freedom than the generated control structures. The inputs were (nr. bones/nr. sample poses): Horse: (29/25 galloping, going up) – Dragon: (110/12 flying, some walking); Scorpion (62/20 walking, attacking); Dancer (22/6). Note that the device for the Dancer is asymmetric due to the asymmetry in the input poses: the left arm of the character moves almost rigidly with the torso and it is thus not necessary to have any joint controlling the left arm.*

Note that having multiple copies of each splitter in a kit is common. For efficiency, we compute the candidate assignments only once for each kind of splitter. Constraints Eq. (3) are also grouped to act on classes of splitters instead of individual $\mathbf{s}$.

Realtime pose interpolation. The algorithm discussed so far produces a device $\mathcal{D}$ controlling a subset of the DoFs of the rig, leaving the remaining DoFs unchanged. To synthesize those missing rig control values which can add detailed pose nuances back into the final animation, we employ [Buck et al. 2000] for pose-space interpolation. Principal component analysis on a set of sample poses is used to choose the largest principal vectors of the dominant dimensions. These map the original high dimensional poses, defined as stacked quaternions representing rotations of the active rig controls, into a lower dimensional space. Delaunay triangulation is used to create a piecewise linear manifold for interpolation.

Given a new pose, we first project it onto this space, locate the projected point in the triangulation, then find the barycentric coordinates inside the simplex that contains it. The coordinates then act as blending weights. We then use Slerp to interpolate the quaternions and feed them to the previously rigid joints. The technique is efficient enough for realtime computation of the barycentric blending weights and generates continous interpolation results. More samples in the pose interpolation will generally increase the quality and there is no upper limit on input poses. Since our experiments with higher-dimensional pose spaces did not lead to improved results likely due to the "curse of dimensionality", a 2D pose space is used (as proposed in [Buck et al. 2000]).

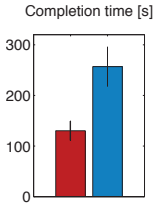
4 Evaluation

We briefly report on a series of experiments that we conducted to evaluate our algorithms for the computation of the physical device configurations and to compare our proposed hardware design to the closest state-of-the-art [Jacobson et al. 2014b].

For our experiments we fabricate a hardware kit containing 18 joints and 7 splitters. We implemented a Python program for Autodesk's Maya® [Maya 2014], driving characters which we acquired online.

Rig reduction and device configuration. Fig. 9 qualitatively demonstrates the results from our method for different 3D characters ranging in complexity from 22 bones (*Dancer*) to 110 bones (*Dragon*), using a small number of sample poses as input (*Dancer*: 6; *Horse*: 25). Optimized device configurations are computed based on three hardware kits of increasing size. Our method produces intuitive and sensible configurations, placing splitters and joints such that they articulate the most important features of the character. For each of the examples the amount of direct control a user has increases with number of joints. However, there clearly is a trade-off between directness of the mapping and physical complexity. Fig. 7, plotting pose error as function of joints, illustrates

this further: initially additional joints drastically decrease the error but the curve levels off, and beyond 10 joints gains are marginal. Experimentally we found that most characters can be controlled well with 5-10 joints. This supports our initial goal of combining the benefits of tangible control and rig animation algorithms. Please note that the algorithm can produce asymmetric configurations even for symmetric characters, e.g., for the *Dancer*. This can be due to asymmetric input poses (rightmost column), or due to the limited number of joints in the kit (leftmost column), which forces the algorithm to introduce a "splitter" replacement (in red). This enables the articulation of the character's arm despite the lack of splitters. In such cases pose interpolation controls the un-mapped limbs causing whole character to move smoothly, albeit at the cost of direct control (cf. Sec. 5).



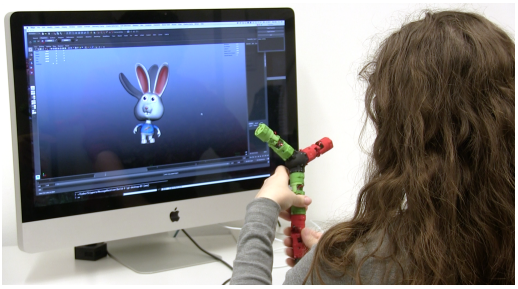
Hardware comparison. Intertwined with our algorithmic contributions is the hardware design, which we discussed in Sec. 3.1. To isolate the effect of the new joint design we repeat the posing experiment reported in [Jacobson et al. 2014b]. We concentrate on directly comparing *their* device with *ours*, assuming that results will carry over to the established baseline (a mouse-based Maya-like UI). We faithfully recreated the reported experimental conditions with 10 participants, recruited from our university (2 female, 8 male; ages ranging from 25 to 35). Device presentation was counterbalanced by half of the participants starting with *our* device, half with *theirs*. Per posing experiment, participants were asked to replicate poses shown on-screen as close as possible and stop once they reached a satsifactory accuracy. The results indicate that the new hardware considerably outperforms the existing design by [Jacobson et al. 2014b]. *Ours* requires significantly less time (see inset; *ours* mean = 130.24 s, standard deviation = 19.07; *theirs* mean = 256.94 s, SD = 38.62) and achieves higher relative accuracy, represented by the remaining pose error in percent of the original pose error at the beginning of each experiment (*ours*: mean = 21.66% SD = 3.62; *theirs* mean = 36.23% SD = 5.48). These differences also translate into the more lenient metric of "work" reported in [Jacobson et al. 2014b], essentially the area under the plot lines in Fig. 10 (*ours*: mean = 89.65% SD = 18.8; *theirs* mean = 474.1% SD = 123.7). A Student's t-test reveals that all differences are statistically significant (all p-values ≤ 0.05).

Please note that the speed-up factor of 2 (130.24 s to 256.94 s) reported here is a very conservative estimate. It compares the time elapsed until both devices reach *their own minimum pose error*. In contrast, when measuring the more meaningful average ratio between the time elapsed until the more accurate device reaches the *minimum pose error of the less accurate* device, *ours* achieves a speed-up of 3× (see also Fig. 10).

This difference was also clearly noticeable by observing the subjects during the study: with our hardware the posing can be done rapidly and with minimal experimentation, while many participants needed a long time to understand how to reach the target pose with the old design, due to the requirement to decompose rotation. Fig. 4 illustrates this further showing the effect of directly tracing geodesics with our design versus sequential manipulation of individual Euler angles in the old device.

5 Additional results

To assess the resulting animation qualitatively, we report on a number of examples produced with our approach, consisting of hardware and algorithms that compute the device configuration and pose interpolation at runtime. Please also see the accompanying video.



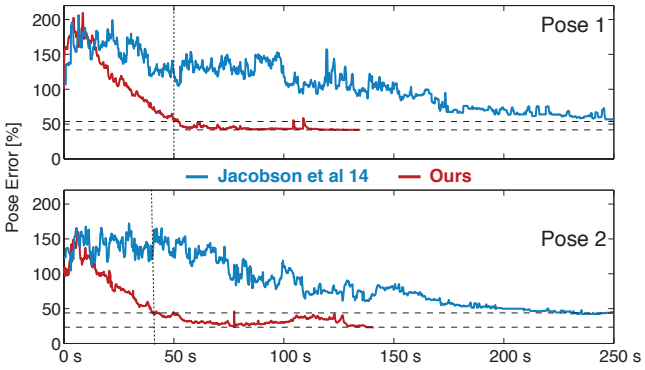


Figure 10: *Top: Experimental condition with on-screen stimuli. Bottom: Two of the poses in the user study, averaged across all participants, show* ours *(red) and (blue) decreasing pose distance from 100% to minimal values at completion (dashed lines).* Ours *is significantly faster and achieves better accuracy.*

Interactive rig control. Fig. 11 shows a sequence of a user driving the *Human* (24 bones) with 7 joints and 2 splitters. The joints are bound to the bones in the torso and the limbs. Since we only control 7 bones directly, the remaining DoFs are controlled via pose interpolation, which reintroduces pose nuances.

Physical inverse kinematics. An important design goal was to create a user experience that resembles inverse kinematics, allowing the user to work directly with positional constraints, rather than having to deal with individual rotations. In Fig. 12, a user positions only the end-effectors of the chain of joints, and the remaining con-

Figure 11: *Interactive animation of the Human character.*

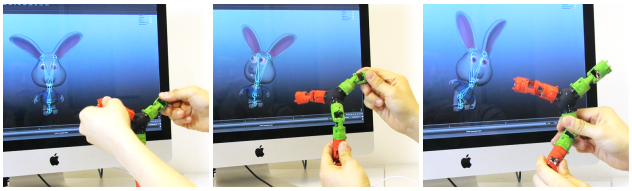


Figure 12: *Physical inverse kinematics. The user positions end-effectors with the device, and the 3D character smoothly follows.*

figuration follows smoothly, resulting in a fluid animation of the on-screen character.

Pose interpolation. The impact of the pose interpolation scheme is demonstrated in Fig. 13. Here we drive the *Scorpion* (64 bones) without (top) and with (bottom) pose interpolation. In both cases we use an identical device that only controls 6 of the bones directly. The approach is particularly useful for such complex characters, which are challenging to pose, especially for novice users. In the extreme, the technique can be used to drive animation sequences with very few pieces, an example is shown in Fig. 14.

Keyframing. Prior tangible devices heavily focus on keyframe posing. This is also possible with our approach. Fig. 16 shows a selection of frames from an animated sequence of the Dragon rig (110 bones), driven by only 8 joints and 2 splitters. The posing of this 30 seconds long animation consisting of 32 key frames took an inexperienced user 35 minutes, of which 15 were spent on non-posing tasks, such as global transformations and camera positioning.

Abstract parameter control. Our device is also useful for controlling non-rotational rig parameters. In Fig. 15, we control a blendshape rig with four expressions. We manually map the angles of a single joint onto the four expressions to create a simple facial animation.

Figure 13: *Effect of pose interpolation. Top: without interpolation. Bottom: with interpolation. Note the additional pose details in the tail and the legs.*

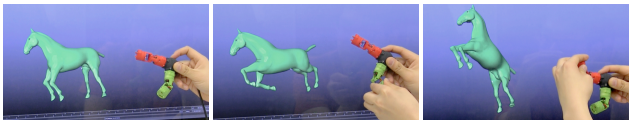


Figure 14: *An extreme case for pose interpolation. We control the Horse rig with only two joints. The red joint controls the torso and the green joint controls one of the rear legs directly; the rest of the pose is synthesized by pose interpolation.*

Figure 15: *Controlling a face blendshape rig with a single joint. We manually map the angles of the joint such that they semantically match the facial expressions (i.e., a $\vee$ shape for smiling, a $\wedge$ for frowning, and twist opens the mouth).*

6 Discussion and conclusion

We proposed a symbiotic pair of novel software and hardware to help animators traverse a large space of poses via fluid tangible manipulation. The key insight is to drive only the most important degrees of freedom of a rig directly, limiting physical device size and simultaneous manipulations. Furthermore, we have demonstrated that our hardware provides a much improved user experience, resembling inverse kinematics. Empirically we have shown that this yields speeds-up of a factor of 3 compared to the most related work in terms of character posing time.

Together, our software and hardware are an important step beyond static keyframing and into the territory of motion sculpting. We believe our approach already provides a valuable tool chain to quickly sketch animations using a variety of professional rigs. However, it is not without limitations and opportunities for future work.

Currently, we employ a standard pose interpolation scheme. While this already produces good results, it is sometimes an issue that desired poses are not represented in the example set, or that the pose set is biased. For example, our Horse character comes with an extended set of galloping poses, which causes the horse's legs to curl even for non-running poses. Exploring more sophisticated interpolation schemes, prioritizing user-specified DoFs over those from example poses, would be interesting future work.

In terms of hardware, avoiding gimbal lock is a large leap in usability, but further improvements could be achieved. Especially with complex device configurations, it would be desirable to be able to control joint friction, perhaps dynamically. However, this is a non-trivial design challenge both in terms of mechanics and potential negative impact on the desired IK properties. Similarly, designing additional types of joints, such as prismatic or telescopic joints, would allow us to support rigs with translational degrees of freedom (such as cartoon characters) and is a challenging but interesting area for future work.

Our device only senses local rotations, while the virtual character's global position and orientation have to be controlled externally. Augmenting the device with an additional sensing mechanism, perhaps vision-based, or leveraging inertial sensors, would further decrease the barriers to fully fluid character animations.

We have concentrated exclusively on character animation. However, we believe that the wide range of motion, modular nature and high accuracy provided by our hardware make it an ideal candidate to be explored in other application domains, such as general purpose input devices (e.g., game-pads, joysticks) or as controllers for virtual or augmented reality applications. To allow for easy reproducibility and foster future work, we will release the open-hardware specification for the device modules and the implementations of the retargeting algorithm and the Maya interface[1].

Acknowledgments

We are grateful to Cédric Pradalier and Evgeni Sorkine for invaluable discussions and engineering support, to Sebastian Schoellhammer for his assistance on 3D modeling and rigging in Maya, to Olga Diamanti for composing the accompanying video, to Cécile Edwards-Rietmann for narrating it and to Jeannine Wymann for her help in assembling the prototypes. We also thank our user study

[1] http://oliglauser.github.io/atamid/

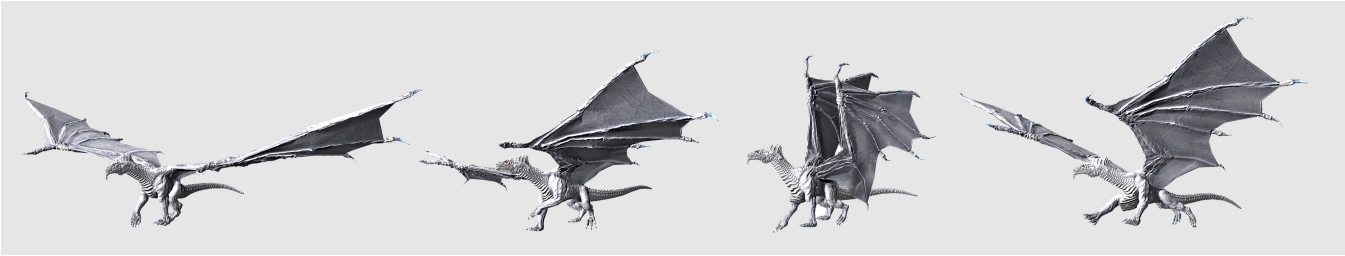


Figure 16: *We show some of the keyframes generated by an inexperienced user from a key framing session. This Dragon rig contains 110 bones, and a 30 seconds long animation can be generated in 35 minutes, of which 15 were spent on non-posing tasks.*

participants. This work was supported in part by the SNF grant 200021_162958 and the ERC grant iModel (StG-2012-306877). Alec Jacobson is funded in part by NSF grants IIS-14-09286 and IIS-17257.

References

ACHIBET, M., CASIEZ, G., LÉCUYER, A., AND MARCHAL, M. 2015. Thing: Introducing a tablet-based interaction technique for controlling 3D hand models. In *Proc. CHI*.

AHN, J., AND WOHN, K. 2004. Motion level-of-detail: A simplification method on crowd scene. In *Proc. CASA*.

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3.

BARAN, I., VLASIC, D., GRINSPUN, E., AND POPOVIĆ, J. 2009. Semantic deformation transfer. *ACM Trans. Graph. 28*, 3.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 213–230.

BUCK, I., FINKELSTEIN, A., JACOBS, C., KLEIN, A., SALESIN, D. H., SEIMS, J., SZELISKI, R., AND TOYAMA, K. 2000. Performance-driven hand-drawn animation. In *Proc. NPAR*.

CELSYS, INC., 2013. QUMARION. http://www.clip-studio.com.

CHIEN, C.-Y., LIANG, R.-H., LIN, L.-F., CHAN, L., AND CHEN, B.-Y. 2015. Flexibend: Enabling interactivity of multi-part, deformable fabrications using single shape-sensing strip. In *Proc. UIST*.

ESPOSITO, C., PALEY, W. B., AND ONG, J. 1995. Of mice and monkeys: a specialized input device for virtual body animation. In *Proc. I3D*.

FENDER, A., MÜLLER, J., AND LINDLBAUER, D. 2015. Creature teacher: A performance-based animation system for creating cyclic movements. In *Proc. SUI*.

FENG, T.-C., GUNAWARDANE, P., DAVIS, J., AND JIANG, B. 2008. Motion capture data retrieval using an artist's doll. In *Proc. ICPR*, 1–4.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proc. SIGGRAPH*.

GUAY, M., CANI, M.-P., AND RONFARD, R. 2013. The line of action: An intuitive interface for expressive character posing. *ACM Trans. Graph. 32*, 6 (Nov.), 205:1–205:8.

GUAY, M., RONFARD, R., GLEICHER, M., AND CANI, M.-P. 2015. Space-time sketching of character animation. *ACM Trans. Graph. 34*, 4 (July), 118:1–118:10.

GUROBI OPTIMIZATION, INC., 2015. Gurobi optimizer reference manual. http://www.gurobi.com.

HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. 2012. Rig-space physics. *ACM Trans. Graph. 31*, 4.

HAHN, F., MUTZEL, F., COROS, S., THOMASZEWSKI, B., NITTI, M., GROSS, M., AND SUMNER, R. W. 2015. Sketch abstractions for character posing. In *Proc. SCA*.

HELD, R., GUPTA, A., CURLESS, B., AND AGRAWALA, M. 2012. 3D puppetry: A kinect-based interface for 3D animation. In *Proc. UIST*, 423–434.

HOLDEN, D., SAITO, J., AND KOMURA, T. 2015. Learning an inverse rig mapping for character animation. In *Proc. SCA*.

HUANG, W., 2015. Deformation learning solver. https://github.com/WebberHuang/DeformationLearningSolver.

ISHII, H., AND ULLMER, B. 1997. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proc. CHI*.

JACOB, R. J. K., SIBERT, L. E., MCFARLANE, D. C., AND MULLEN, JR., M. P. 1994. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact. 1*, 1 (Mar.), 3–26.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4.

JACOBSON, A., DENG, Z., KAVAN, L., AND LEWIS, J. 2014. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*.

JACOBSON, A., PANOZZO, D., GLAUSER, O., PRADALIER, C., HILLIGES, O., AND SORKINE-HORNUNG, O. 2014. Tangible and modular input device for character articulation. *ACM Trans. Graph. 33*, 4.

JIN, M., GOPSTEIN, D., GINGOLD, Y., AND NEALEN, A. 2015. AniMesh: Interleaved animation, modeling, and editing. *ACM Trans. Graph. 34*, 6.

KABSCH, W. 1976. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32*, 5, 922–923.

KNEP, B., HAYES, C., SAYRE, R., AND WILLIAMS, T. 1995. Dinosaur input device. In *Proc. CHI*, 304–309.

LE, B. H., AND DENG, Z. 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph. 33*, 4, 84.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH*.

MASLIAH, M. R., AND MILGRAM, P. 2000. Measuring the allocation of control in a 6 degree-of-freedom docking experiment. In *Proc. CHI*.

MAYA, 2014. Autodesk, `http://www.autodesk.com/maya`.

NAKAGAKI, K., FOLLMER, S., AND ISHII, H. 2015. Lineform: Actuated curve interfaces for display, interaction, and constraint. In *Proc. UIST*.

ÖZTIRELI, A. C., BARAN, I., POPA, T., DALSTEIN, B., SUMNER, R. W., AND GROSS, M. 2013. Differential blending for expressive sketch-based posing. In *Proc. SCA*.

RHODIN, H., TOMPKIN, J., KIM, K. I., KIRAN, V., SEIDEL, H.-P., AND THEOBALT, C. 2014. Interactive motion mapping for real-time character control. *Comput. Graph. Forum 33*, 2.

SAVOYE, Y., AND MEYER, A. 2008. Multi-layer level of detail for character animation. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)*.

SCHAEFER, S., AND YUKSEL, C. 2007. Example-based skeleton extraction. In *Proc. SGP*.

SHIRATORI, T., MAHLER, M., TREZEVANT, W., AND HODGINS, J. 2013. Expressing animated performances through puppeteering. In *Proc. 3DUI*.

SUMNER, R., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3, 399–405.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3, 488–495.

TAGLIASACCHI, A., ALHASHIM, I., OLSON, M., AND ZHANG, H. 2012. Mean curvature skeletons. *Comput. Graph. Forum 31*, 5.

YOSHIZAKI, W., SUGIURA, Y., CHIOU, A. C., HASHIMOTO, S., INAMI, M., IGARASHI, T., AKAZAWA, Y., KAWACHI, K., KAGAMI, S., AND MOCHIMARU, M. 2011. An actuated physical puppet as an input device for controlling a digital manikin. In *Proc. CHI*, 637–646.

ZHAI, S., AND MILGRAM, P. 1998. Quantifying coordination in multiple dof movement and its application to evaluating 6 DOF input devices. In *Proc. CHI*, 320–327.

A Fault-resistant distributed protocol

The presence of slip-rings makes the electric connections between the pieces unreliable: when many pieces are moved at the same time, it is common for pieces to temporarily lose power or get disturbed on the data channel that corrupts some packages. We evaluated the protocol proposed by [Jacobson et al. 2014b], but discovered that it is not applicable to our scenario, since it requires the global bus to be stable for long periods to do a synchronized read on all sensors. We propose a fault-resistant protocol that can gracefully recover from random disconnects or missing packages, while allowing to easily reconstruct the topology of the device.

We assign a global identifier to all components (the hash of the timestamp of the programming time) and we equip each component with two messaging queues, one for incoming and one for outgoing messages. Each component acts as a repeater for messages that come from pieces below it, sending them to its parent. All messages contain the ID of the creator of the message, the ID of the parent, the message itself and a consistency hash to detect communication errors. The ID of the parent is initialized as empty by the message creator; each time a message is received with an empty parent ID, the receiving component fills the field with its own ID. The message distribution is done using a polling mechanism: every 5 ms, each component probes its children for messages, and queues messages, which are in turn passed to the parent when queried.

Our protocol supports three types of messages, generated by the pieces at different frequencies: data messages (50 Hz), which contain angle readings, type messages (2 Hz), which contain the color and type of the component, and child messages (2 Hz), which describe the rotation associated with each slot of a splitter. This protocol robust thanks to its distributed nature and built-in error checking: the topology can be reconstructed by storing the parent-child relations in a hash table, and disconnected pieces can be detected by keeping a timestamp for every component. If no messages are received from a component for more than 2 s, then the subtree starting at that component has been detached from the device.

B Evaluating Pose Reachability

Given a candidate assignment $\mathcal{D}$, we want to find the device poses $\mathbf{P}^{\mathcal{D}}$ which minimize the energy in Eq. (1). Each pose $\mathbf{p}_i^{\mathcal{D}}$ of the device should be optimized to lead to the corresponding sample pose $\mathbf{p}_i^{\mathcal{C}}$, and it is uniquely defined by a set of rotations $R_{ij}^{\mathcal{D}}$, each associated with a joint j. The assignment $\mathcal{D}$ associates the rotations $R_{ij}^{\mathcal{D}}$ to rotations in the rig; as discussed in Sec. 3.2, all the unassigned rig nodes inherit the rotation of their parent.

The overall pose error E is defined as the sum of the individual pose errors E_i (Eq. (1)). Since we have a fixed candidate assignment $\mathcal{D}$, this error can be optimized separately for each pose:

$$E = \arg\min_{\mathbf{P}^{\mathcal{D}}} \sum_i E_i(\mathbf{p}_i^{\mathcal{D}}) = \sum_i \arg\min_{\mathbf{p}_i^{\mathcal{D}}} E_i(\mathbf{p}_i^{\mathcal{D}}). \quad (6)$$

For the sake of clarity, from now on we omit the index i, since all the equations are referred to a single pose (except w_b, which are constants shared by all poses):

$$E_i = \arg\min_{R^{\mathcal{D}}} \sum_b^m w_b |\mathbf{n}_b - \tilde{\mathbf{n}}_b(R^{\mathcal{D}})|_2^2. \quad (7)$$

The weights w_b account for the difference in surface area (see Sec. 3.2). Eq. (7) highlights the variables we wish to optimize for, that is, the rotations of the device that induce a deformation of the rig with nodes $\tilde{\mathbf{n}}_b$ that better approximate the position of the sample nodes $\mathbf{n}_b$. We denote by $\tilde{\mathbf{n}} = (\tilde{\mathbf{n}}_0, \tilde{\mathbf{n}}_1, ..., \tilde{\mathbf{n}}_m)$ the induced node positions through the device $\mathcal{D}$ in the sample pose; b is a stack of the rest pose bones of rig $\mathcal{C}$ and H a $3m \times 3m$ matrix encoding the rig hierarchy, and

$$\tilde{R}^{\mathcal{C}} = M_{\mathcal{D}} R^{\mathcal{D}}, \quad (8)$$

where $M_{\mathcal{D}}$ is a $3m \times 3l$ matrix representing the map from the pose of the device to the set of bone rotations $\tilde{R}^{\mathcal{C}}$ ($3m \times 3$ matrix) in rig pose $\tilde{\mathbf{p}}^{\mathcal{C}}$, $R^{\mathcal{D}}$ is a $3l \times 3$ matrix vertically stacking joint rotation matrices $R_j^{\mathcal{D}}$, and l is the number of joints in device $\mathcal{D}$. The value of $\tilde{\mathbf{n}}$ relates to $R^{\mathcal{D}}$ in the following way:

$$\tilde{\mathbf{n}} = H \operatorname{diag}(\tilde{R}^{\mathcal{C}}) b = H \operatorname{diag}(M_{\mathcal{D}} R^{\mathcal{D}}) b, \quad (9)$$

where $\mathrm{diag}(\tilde{R}^{\mathcal{C}})$ is a $3m \times 3m$ matrix with the vertically stacked rotation matrices in $\tilde{R}^{\mathcal{C}}$ on its diagonal.

Note that the pose error is a sum of squares (Eq. (7)), which we optimize using the Gauss-Newton minimization method. For the optimization update steps, we use a differential representation, where each rotation associated with a joint j is encoded as a vector $\omega_j = (\omega_{j,1}, \omega_{j,2}, \omega_{j,3})$:

$$R_j^{\mathcal{D}^{k+1}} = R_j^{\mathcal{D}^k} e^{J(\omega_j)}, \tag{10}$$

where $R_j^{\mathcal{D}^k}$ is a fixed rotation computed at the previous step, and $e^{J(\omega_j)}$ is the change computed in the current iteration. The optimization variables are the entries of $J(\omega_j)$, which is defined as:

$$J(\omega_j) = \begin{bmatrix} 0 & -\omega_{j,3} & \omega_{j,2} \\ \omega_{j,3} & 0 & -\omega_{j,1} \\ -\omega_{j,2} & \omega_{j,1} & 0 \end{bmatrix}. \tag{11}$$

This exponential map parametrization of the rotation space elegantly and efficiently ensures, without additional constraints, that $R_j^{\mathcal{D}^{k+1}}$ is a rotation. We stack all the variables in a vector $\omega = (\omega_1, \omega_2, ..., \omega_l)$. In every iteration, we set ω to:

$$\omega = -(D^T D)^{-1} D\mathbf{r} \tag{12}$$

where $\mathbf{r}$ is a $3m$-vector

$$\mathbf{r} = \mathbf{W}(\mathbf{n} - \tilde{\mathbf{n}}(R^{\mathcal{D}^k})) \tag{13}$$

and D is a $3m \times 3l$ matrix whose element at row a and column c is:

$$D_{ac} = \frac{\partial \mathbf{r}_a}{\partial \omega_c}. \tag{14}$$

The optimization is initialized with the rotations of the sample pose. Note that this initial guess does not reproduce the sample pose, since not all bones have a joint assigned in $\mathcal{D}$. We stop the iterations when:

$$|E_i^{k+1} - E_i^k| \leq |E_i^{k+1}| \cdot 10^{-6} + 10^{-6}. \tag{15}$$

C User study details

We ran an experiment to compare *our* design to [Jacobson et al. 2014b], which in turn already establishes a baseline of equivalence to mouse and keyboard. We asked 10 users (2 female, 8 male) to participate in our experiment (cf. Sec. 4). Most of the users reported some prior experience with 3D tools and even with Maya, but none were professional animators. The device presentation was balanced, and exactly half of the participants started with ours device. We instructed the participants to adjust an on-screen character's pose according to a semi-transparent target pose. The participants self-reported as soon as they considered their posing to be "good enough". The procedure consisted of one practice block to familiarize oneself with the device, followed by three blocks of timed posing. After a relaxation period, we repeated the procedure with the second device. Finally, we conducted exit interviews with each participant.

Experiment metrics. Based on the three timed blocks, we computed three metrics. First, *task completion time* is the elapsed time until the minimal pose error is reached in each posing task. Second, we recorded the *minimal pose error* [%] reached per pose. Finally, we computed the *work* metric, as proposed in [Jacobson et al. 2014b]. This is the integral of the pose error [%] over time (cf. Fig. 10). In all three metrics, pose error percentages are always in relation to the initial pose error.

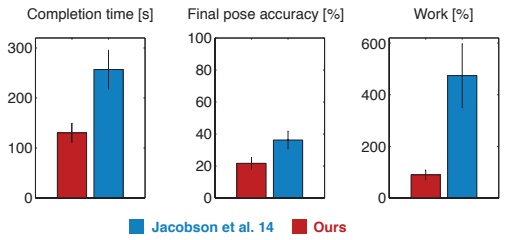


Figure 17: *Additional experimental results: Comparing our design (red) to the design of [Jacobson et al. 2014b] (blue). Our device performs better in every of the three metrics (time, accuracy, work).*

Additional results. For an in-depth discussion of results please refer to Sec. 4. Fig. 17 shows that our device outperforms [Jacobson et al. 2014b] in all of the three metrics.

In addition to the above metrics, we also computed absolute angular errors and compared these across device designs. Note that the presented numbers should not be confused with the accuracy of our device (0.5 degrees).

The absolute average angular error per joint, as reached by our *best* participant, was 5.06 degrees (elapsed time: 216.3 s). In comparison, the same user achieved an average error of 14.51 degrees (elapsed time: 245.5 s) with the old design.

The minimal absolute average angular error a user reached was 6.32 degrees with *both* devices (rounded to 2 digits). However, the user reached this minimal error 41.7 s faster with our device (126.6 s) than with [Jacobson et al. 2014b] (168.3 s).

The final average angular error (across all participants) for our device was 11.07 degrees (standard deviation: 4.51 degrees) versus 19.68 degrees (SD: 10.40 degrees) for [Jacobson et al. 2014b].