Dinosaur Input Device *

Brian Knep
Industrial Light and Magic
San Rafael, CA

Craig Hayes
Tippett Studios
Berkeley, CA

Rick Sayre
Pixar
Richmond, CA

Tom Williams
Industrial Light and Magic
San Rafael, CA

ABSTRACT
We present a system for animating an articulate figure using a physical skeleton, or *armature*, connected to a workstation. The skeleton is covered with sensors that monitor the orientations of the joints and send this information to the computer via custom-built hardware. The system is precise, fast, compact, and easy to use. It lets traditional stop-motion animators produce animation on a computer without requiring them to learn complex software. The working environment is very similar to the traditional environment but without the nuisances of lights, a camera, and delicate foam-latex skin. The resulting animation lacks the artifacts of stop-motion animation, the pops and jerkiness, and yet retains the intentional subtleties and hard stops that computer animation often lacks.

KEYWORDS: Entertainment applications; Motion capture; Animation

INTRODUCTION
Motivated by the large amount of high-quality computer graphics animation called for by the film *Jurassic Park* [8] and by the desire to use the talent of experienced traditional animators, we built a system that allows them to animate computer graphics characters easily. We wanted the animators to be able to use their developed skills without first climbing the learning curves of computers and computer graphics, and in particular, without learning to use a complex interface such as that of a large commercial animation package. The result is a *stop-motion-capture device* called the Dinosaur Input Device, or DID. The DID is a highly intuitive three-dimensional (3D) input device, and many of its features are applicable to ordinary motion capture and to 3D input generally.

To accomplish our goal, we use a tool familiar to stop motion: the *armature*, an articulate skeleton often made of aluminum and steel rods, hinges, and swivels. In traditional stop-motion, the armature is posed by the animator and photographed onto a single frame of film. The armature is then moved incrementally, and photographed again onto the next frame of film. After many poses have been photographed, the resulting film will appear to show a moving armature. Instead of using film, our system captures the poses of an armature and uses them to position and animate a computer model.

To produce high-quality animation, the system must be very precise. One of the advantages of stop-motion animation over computer animation is the subtle movements and hard stops that are often smoothed away by computer interpolation. An imprecise or error-prone system would destroy these subtleties. The system must also be fast so the animators can immediately see their results composited over a background picture. This allows them to animate to the background, resulting in better integration of the live and computer-generated footage. Finally, the system cannot impair the movement of the armature or hinder the animators.

The most commonly used motion-capture devices are image based, such as SuperFluo's ELITE Motion Analyzer [3] and Motion Analysis's ExpertVision [7]. They rely on two or more high-quality CCD cameras, several reflective markers, and some specialized hardware. The markers are placed on the objects or actors and the hardware uses the images produced by the cameras to find the 3D position of each marker. The cameras and other hardware make these systems expensive. They are also imprecise—the calculated position varies over the surface of the marker—and slow—once the positions have been found, they need to be tracked and correlated with the computer joints. If the actors' motion causes markers to cross, the systems often get confused and the user must intervene to identify markers. They also require a large, unobstructed space to set up the cameras. These systems are useful for capturing the real-time gross motion of an object or actor, but lose the high-detailed motion and introduce too much noise into the data.

There are also systems such as the Polhemus Fastrak and the Ascension Bird that use electromagnetic transmitters and receivers to determine the location and orientation of an object. The object is connected to the computer via a cable tether. These systems work in real time, but they are noisy and imprecise, especially around metal objects like the armature.

Then there are systems that use potentiometers to record joint rotations. Examples are the Dextrous Handmaster [9], the Compact Master Manipulator [4], and the body suit used in the feature film *Toys* [6] [10]. These systems are fast and cheap, but the potentiometers are noisy and imprecise.

The above approach is the closest to what we need, and we overcome its shortcomings by using optical encoders rather than potentiometers. In the Armature section, below, we

*This device can of course be used to animate non-dinosaurs, but the name has stuck.

describe these encoders in detail and compare them against potentiometers.

The system we built, the DID, allows an animator, either a traditional stop-motion artist or a computer-graphics artist, to create computer animation by posing a physical armature. Digital sensors and specialized hardware monitor the global position and orientation of the armature and the orientations of the armature's joints and make them available to a computer. The computer uses these orientations to pose and display a 3D computer model composited over a background frame. Once the animator is satisfied with the pose, the computer records it and the animator moves on to the next pose. The recorded animation can be treated like any other computer animation, such as those produced on the SoftImage, Alias, Wavefront, and TDI modeling and animation systems. The animation curves can be edited, and the animation can be rendered with a high-quality renderer using motion blur. These are things that cannot be done with traditional stop-motion animation.

As an input device, the DID is very intuitive: To put the model in a particular pose, users pull and push physical joints; to look at a different view, users walk around the model or move their heads. This is much simpler than a hierarchy of nodes with rotate, translate, and scale values at each node, accessed by menu choices, sliders, and mouse movements. The DID is also rock solid: the computer model doesn't move unless the armature does.

The DID is composed of three parts: the physical armature, the controller, and the computer software. The next three sections describe each of these parts in detail.

ARMATURE

The armature is similar to those used in traditional stop-motion animation but with important differences: It is bare, with no foam-latex skin or adornments; its joints are monitored by digital sensors; it is larger than the traditional armature; and it uses a different set of joints than the traditional armature (see Color Plate 1).

No Skin

Armatures are usually covered in painted foam-latex skin and adorned with hair, nails, and teeth. The DID armature is bare: Animators manipulate the joints directly. The advantages are that animators don't have to worry about damaging delicate details, they can find a joint easily if it needs to be tightened or loosened (necessary for certain movements), they can make tiny micro-movements that otherwise wouldn't be possible, and they can see the range of movement left in a joint before it reaches its mechanical bounds.

Sensors

Each joint in the armature is monitored by a set of sensors. When a joint moves, each of the corresponding sensors sends out a signal describing the motion. Sensors also detect movements of the rig on which the armature is mounted. This rig controls the position and orientation of the entire armature.

For sensors we use optical encoders. They are encased in plastic boxes measuring about three-quarters of an inch on each side with a cylindrical shaft projecting from one side and wires projecting from the opposite side. When the shaft turns, the amount of rotation is digitally encoded and sent out along the wires.

The raw encoders can detect rotations as small as one-third of one degree. Using reduction gearing we can attach an encoder to a joint so that a single revolution of the joint results in many revolutions of the encoder shaft. This lets the encoders detect joint rotations much smaller than one-third of one degree. The gears, however, introduce an upper limit on the precision due to *mechanical backlash*, the amount a gear can move before neighboring gears move. We have found that with our armatures one-third of one degree is precise enough for high-quality animation.

Two types of optical encoders are common: absolute and relative. An absolute encoder has a code for its entire resolution etched onto an internal disk. As the disk turns, a detector reads this code and gives an absolute and instantaneous readout of the encoder's current rotation. Relative encoders have an internal disk with simple alternating stripes (255 stripes in our encoders). Two out-of-phase detectors generate equal-period pulse trains when the shaft is rotated; counting pulses tells how far the disk has turned, while the relative phase of the two signals tells the direction. Absolute encoders are by necessity larger than relative encoders of equivalent resolution, and considerably more expensive. We therefore use relative encoders. Unfortunately, this means we have to maintain the absolute rotations of the joints elsewhere (see the Controller section below).

We use optical encoders rather than potentiometers and an analog-to-digital converter for several reasons. First, potentiometers are noisy even if perfectly shielded from electromagnetic interference. Optical encoders are much less susceptible to interference because they produce signals in the digital domain. This makes expensive, bulky, shielded cables unnecessary. Second, potentiometers are highly nonlinear, making accurate and consistent measurement of small incremental movements impossible. Optical encoders are very linear (on the order of fractions of the encoder sensitivity). Third, potentiometers produce a noisy spike when turned more than 360 degrees, making reduction gearing and multi-revolution motions impractical. Optical encoders do not have this limit. Finally, potentiometers are not perfectly *repeatable*—they might give different values at the same physical rotation. Optical encoders are perfectly repeatable.

Size

In order to fit the encoders on the armature without hindering its range of motion, we had to enlarge the underlying skeleton. Our armatures are about three feet from the head to the tip of the tail, which is about 30% larger than they would be in a traditional stop-motion environment. We were concerned that the larger size would impair the animators, but once they got used to it they found that the larger armature was in fact easier to manipulate.

Joints

Each encoder can monitor only a single axis of rotation, so each joint is either a hinge, a swivel, or a universal joint that we developed that allows three encoders to monitor all possible rotations around a central point. Unfortunately, the

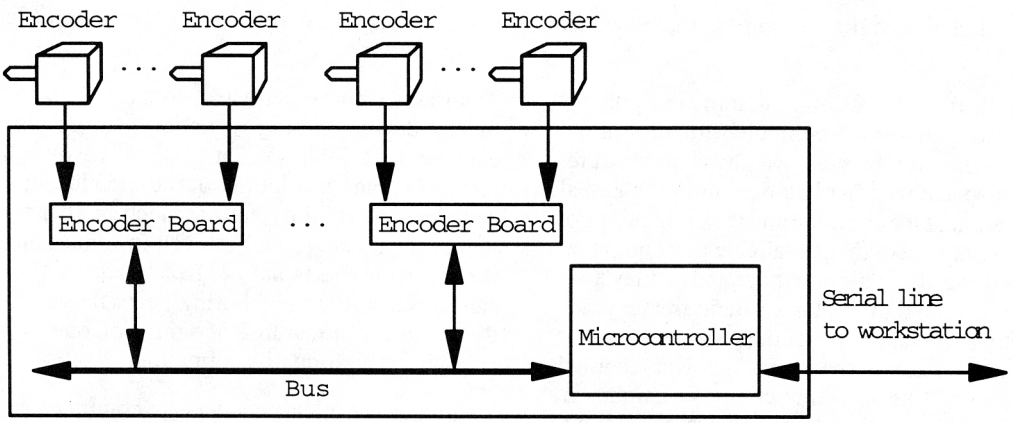

Figure 1: A simplified schematic of the controller.

universal joints are large and can only be used in uncluttered areas, like the neck and ankles. There are no ball joints in the armature.

CONTROLLER

Because we use relative encoders, we need to maintain the absolute rotations somewhere. We built specialized hardware to maintain these values and feed them to a host computer. We call this hardware the *controller*.

The controller is highly modular. An embedded microcontroller communicates with the workstation over a high-speed serial link with a bus designed to hold various interface cards (see Figure 1). We built interface cards that work with the optical encoders, but the controller can handle other cards that support different types of sensors. For example, an analog-input card could be built for strain gauges for measuring very small movements, such as those of armature toes. These interface cards plug into the bus, making it easy to configure a system with support for the desired number and type of sensors. The controller manages the cards, and allows the host to read the monitored armature joints and reset or preload their values for calibration.

Each interface card is capable of supporting sixteen encoders. There is one 24 bit counter for each encoder. By observing the leading and trailing edges of both encoder pulses, the hardware can increase the available resolution by a factor of four. These extra bits of precision are not guaranteed to be strictly linear, but they are guaranteed to be strictly monotonic and repeatable, and the non-linearity is consistent between defined counts. This therefore lets us track 16,384 revolutions of the encoder shaft (2^{24} values / 1024 pulses-per-revolution).

In order to calibrate the counters, we first pose the armature in a *zero-position* with the spine and legs straight and perpendicular to each other. Mechanical stops help us set this pose. A metal cube, for example, helps ensure a right angle between the legs and the spine. Although this is a completely unnatural pose, it is much easier to form and verify than a neutral, or relaxed pose. We then reset the counters in the controller to zero.

SOFTWARE

The software reads the encoder values from the controller and uses them to pose and display a 3D model of the object being animated. By compositing the model over a background picture, and by using a simulated camera whose motion has been matched to the camera that filmed the background pictures, the animator can see the pose in context as it will be seen on film. In other words, the system is *WYSIWYG*—what you see is what you get.

We can also display 3D geometry representing objects in the scene, such as tables and lamps. Animators use these to gauge the spatial relations between the animated object and the other objects. We can make sure, for example, that a foot is solidly on top of a table. To aide this, the camera can be snapped to the x, y, and z axes and moved around the scene using standard camera controls.

We can display either the computer model or a simple ball-and-stick model depicting the joints and rods of the armature. For prototyping speed, information about the armature is stored in a file that is interpreted at run time. This file describes the armature hierarchy, the names of the joints, the axes around which they rotate, and the ratio of joint revolutions to encoder revolutions. An example of a leg as described in this file is in Figure 2.

The whole process—reading the rotations from the controller, matching the computer model to the armature, and displaying the computer model composited over a background plate—takes one second on a Silicon Graphics Indigo R4000 and thus gives the animator quick feedback. Once he or she is satisfied with the pose, the software records it and moves to the next frame—moving the camera and bringing in a new background plate if necessary. The software can also generate a quick flipbook-type rendering of all the recorded animation for preview.

Matching

Due to software, animation and physical constraints, the skeletons of the physical models and the computer models often do not match. Software and animation constraints force us to build our computer models with joints in specific locations while physical constraints do not allow us to match

```
node {
  name rightKneeBend     # joint name
  axis   1 0 0           # joint axis
  sensor 5               # sensor index
  scale  25              # gear ratio
  node {
    name   rightCalve    # rod name
    length 4             # rod length
    node {
      name rightAnkleSwivel
      axis 0 0 1
      sensor 6
      scale -10
      node {
        name rightAnkleBend
        axis   1 0 0
        sensor 7
        scale  25
      }
    }
  }
}
```

Figure 2: An excerpt from an armature description file.

these joint locations in the armature. Each joint in the computer model is a universal joint: it can rotate about the x, y, and z axes. As explained in the Armature section, not all of the joints on the armature are universal, and so we cannot make a one-to-one match between them and the joints on the computer model—where the computer model has one joint the armature might have two joints, one for x rotation and one for y rotation, separated by a short distance of perhaps one inch. The models have different numbers of joints often in different locations and bending in different directions. Posing the computer model based on the physical joint data is therefore a non-trivial task.

This problem is over-constrained. The poses of the two different skeletons cannot be matched exactly—the best we can do is get a good pose based on a chosen metric. Our concern is that key joints, or *anchors*, match up as closely as possible; the rest of the joints need to approximate the pose but need not follow it exactly. The anchors are the leg joints, the hips, the head, and the tip of the tail (see Figure 3)

To match the models, we first group the joints into chains with the anchors as the endpoints of the chains. In our example there's a chain between the head and the hips and between the hips and the tip of the tail. There are also chains between the leg anchors, but since they have no intermediate joints, these chains are simply straight lines.

Each chain of joints in the physical model represents a curve in 3-space. Our goal is to match this curve with the corresponding chain of joints in the computer model. This is analogous to fitting a spline to a curve [5] [2]. In our case, however, the distances between the control points of the spline (the joints of the chain) must remain constant.

The method we use to match the chains is simple. We translate the anchor at one end of the computer chain to the position of the corresponding anchor of the physical chain. We then rotate the translated anchor until the second joint in the chain touches the physical chain. We then rotate the second joint until the third joint touches the physical chain. We continue until we reach the end of the computer chain. This process is illustrated in figure 4.

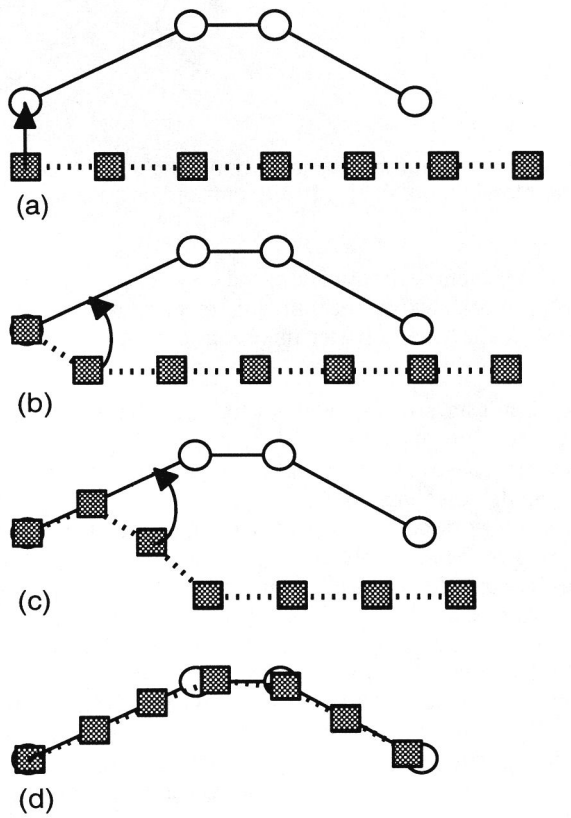

Figure 4: Matching a computer chain (grey squares) to a physical chain (white circles).

This method is easy for users to understand and has very predictable behavior. We get a perfect match for one of the anchors and a good fit for the chain, but because we push the error out toward the far anchors, these anchors may not match well. In practice, our chains are similar enough that this error is not a problem. We also don't put the armature into very extreme poses that could make it difficult to find a good match. We start at the hips and match toward the head (pushing the error toward the head), then match each leg (no error since the legs match one-to-one), and finally match the tail (pushing the error toward the tip of the tail). Since all the joints in the legs are anchors and match up one-to-one, we skip the curve-fitting stage in the legs.

For prototyping speed, the spline-matching information is stored in a file that is interpreted at run time. This file describes which joints are anchors, and which chains to match.

RESULTS

The DID is precise enough for high-quality animation, fast enough for interactive feedback, compact enough not to hin-

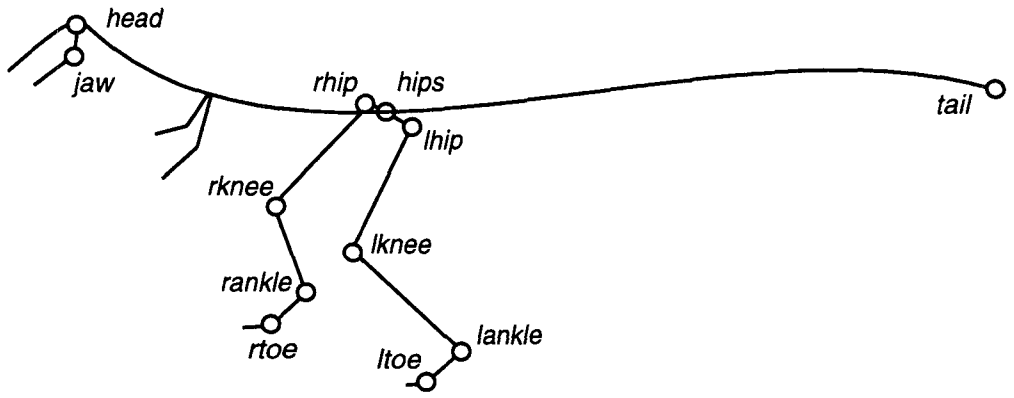

Figure 3: The *anchor* joints. When matching the computer model to the armature, we try to match these joints exactly. The remainder of the joints are matched using a simple curve-fitting technique.

der the movement of the armature, and easy to use. Animators with stop-motion experience are able to start animating immediately, and usually prefer this setup over the traditional setup. Here are some advantages they note:

- They don't have to worry about lights, cameras, or other stage impediments.
- They don't have to worry about hiding the support rods that hold up the armature.
- They are working with a naked armature, and thus can manipulate the joints directly.
- They can edit the resulting animation.
- They can render the resulting animation using computer graphics techniques for more natural motion-blur, textures, and integration with other elements in the scene.
- They can generate a quick, flipbook-type rendering of the animation composited over a background sequence, so they can see how the motion looks in the shot right away, instead of having to wait for film to return from a lab.

Jurassic Park

The DID was used with much success on *Jurassic Park* [8]. Of the 52 shots with computer animation, 15 were animated with the DID. Some of these shots had two creatures, making a total of 20 DID-animated creatures.

Two sequences were composed mostly of DID shots: the main-road sequence, where a tyrannosaur breaks out of her paddy and attacks the park tourists and destroys their jeeps, and the kitchen sequence, where two velociraptors hunt the two children in a large kitchen. To animate these shots we built four functional systems: two for the tyrannosaur and two for the velociraptors. The largest has 74 sensors, each with four wires—two for control, one for power, and one for ground—making a total of 296 wires. In comparison, the body suit used to create the war-room effects in the feature film *Toys* [6] uses only 24 sensors, all potentiometers, each with two wires [10]. The Dextrous Handmaster [9] and the Compact Master Manipulator [4] also both use a small number of potentiometers.

As the animators became more familiar with the DID, they began experimenting with *keyframing*—posing only every fifth or tenth frame and letting the computer interpolate between the poses. They found the resulting motion too smooth, however, so they used keyframing only to experiment with different movements before animating the final shot, which they did frame by frame.

For more on the human-interest side, see the *Cinefex* article on *Jurassic Park* [1].

CONCLUSIONS

The DID is easy to use because the physical device corresponds directly to the computer model it is controlling. Movements of the armature correspond to the same movements in the computer model, and the armature gives tactile feedback and spatial cues that match the computer environment. This is an example of a physical device that is easier to control than a virtual one.

The DID is distinct from a puppeteering device because of the direct correspondence it has with the computer model. The ideas we present, however, could be used to build devices that don't correspond directly to the models they control. For example, we could build a generic bipedal armature and use it to animate several bipedal computer models, each with different proportions. The matching algorithm would have to map the movements of the generic armature onto the computer models, perhaps by scaling or warping the movements. Although the correspondence wouldn't be as direct as the DID, the user would still enjoy the advantages of manipulating a physical 3D input device.

ACKNOWLEDGMENTS

We would like to thank Phil Tippett, Bart Trickel, Tom St. Amand, Stuart Ziff, Adam Valdez, and Randal M. Dutra from Tippett Studios; Dennis Muren, Janet Healy, and the software staff and technical directors at Industrial Light and Magic.

REFERENCES

1. Jody Duncan. The beauty in the beasts. *Cinefex*, 55:42, August 1993.

2. Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*, chapter 23, pages 293–300. Academic Press, 1988.

3. Giancarlo Ferrigno and Antonio Pedotti. ELITE: A digital dedicated hardware system for movement analysis via real-time TV signal processing. *IEEE Transactions on Biomedical Enginerring*, 32:943, November 1985.

4. Hiroo Iwata. Aritificial reality with force-feedback: Development of desktop virtual space with compact master manipulator. In *Proceedings of the ACM SIGGRAPH, Computer Graphics*, volume 24(4), page 165, August 1990.

5. Peter Lancaster and Kestutis Salkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, 1986.

6. Barry Levinson. *Toys*. 20th Century Fox/Baltimore Pictures, November 1992. Motion picture.

7. MotionAnalysis Corporation, Santa Rosa, CA. *System Specifications*, 1993.

8. Steven Spielberg. *Jurassic Park*. Universal Studios, July 1993. Motion picture.

9. Dave Sturman. *Whole-hand input*. PhD thesis, Media Arts and Sciences, Massachusetts Institute of Technology, 1992.

10. Mark Cotta Vaz. Toy wars. *Cinefex*, 54:54, May 1993.