Combining Vision and Computer Graphics for Video Motion Capture

Jane Wilhelms and Allen Van Gelder
Computer Science Dept., University of California, Santa Cruz, CA 95064 *

Abstract

We describe innovative methods for extracting three-dimensional motion of humans and animals from unrestricted monocular video, using a combination of new and established computer vision and computer graphics techniques. We identity features using image processing and active contours. Active contours become anchored to model segments in specified image frames and automatically "pull" the segments to feature position in other frames. Adjustments are subject to joint limits and may use inverse kinematics. Occluded contour points are detected using object geometry and do not participate in feature tracking. Interactive adjustments are possible at any time in the process, allowing extraction of complicated movements regardless of background, camera movement, or feature clarity.

Keywords: Computer Graphics, Animation, Vision, Characters.

1 Introduction

Character animation remains one of the most difficult problems in computer graphics. In this term, we include both the realistic simulation of human and animal motion, and the possibly fantastic motion of imaginary creatures. Character animation has far-reaching practical applications in many fields. In the entertainment industry, fantastic virtual creatures provide a means of expression not available in the normal world, while those based on real humans and animals can partially or completely replace the real thing. Virtual models also have applications in ergonomics and design, and, if accurate, provide a quantitative encoding that is important for the scientific study of human and animal motion.

To be believable, a character must have many flexible joints, move with freedom and balance, and be consistent with the physical laws of the simulated world. Standard key-framing techniques are always laborious and rely on trained and artistic animators. Physical simulation offers realism, but control problems for complex creatures are horrendous. Therefore, capturing motion from real creatures is often the best way to achieve the desired effect.

Studio motion capture, using multiple high-speed cameras or sensors, can accurately encode many motions in real time. Our particular interests, however, involve freely moving animals and people in outdoor environments, such as children playing in the surf and horses galloping. Therefore, we have concentrated on creating methods for extracting motion from unrestricted, monocular (one camera) video. Video and film provide access to humans and animals now deceased or extinct. Video allows us to create virtual versions of our friends and pets inexpensively. Though the field of computer vision offers many automatic techniques for extracting information from images, extraction of three-dimensional articulated motion from complex unrestricted scenes remains a major challenge.

We have combined automatic methods from computer vision with goal-directed, parametric, and interactive techniques from computer graphics. Our parameterized hierarchical methods (referred to here as *supersegments*) allow motion to be controlled succinctly at different levels of complexity. Vertebrates have

* wilhelms,avg@cse.ucsc.edu

about eighty joints; if the number of joints is reduced so is the natural mobility of the body. Supersegment parameters condense the solution space to a tractable size. Goal-directed methods such as inverse kinematics automatically adjust groups of joints to a desired configuration.

In this paper, we describe a system for creating and animating virtual three-dimensional hierarchical models based on general monocular video. Many of our techniques are not new but have been combined in a new way to solve our particular problem. A significant new contribution is our method for automatically adjusting the virtual model to track the image sequence, taking into account joint limits. We concentrate on the horse as an example creature, with some results shown for a human. Our web site `www.cse.ucsc.edu/~wilhelms/fauna` provides more images and animations.

The paper is organized as follows. Section 2 summarizes previous work in related areas. Section 3 describes our basic model, aids for efficient manipulation, and the process of creating a generic version of a particular animal. Section 4 describes how we prepare the images and use active contours for feature extraction. Section 5 describes how we combine our model with active contours extracted from a video sequence to semi-automatically create three-dimensional creature motion. Section 6 describes our results and Section 7 mentions future work and draws conclusions.

2 Background

Our research combines techniques from computer vision and computer graphics. The computer vision methods most applicable to our work are model-based techniques for extracting articulated body motion from monocular image sequences. (The review of Aggarwal *et.al* is an excellent summary of applicable research [1].) In these approaches, a hierarchical geometric *model* is matched with the image *features* being tracked. The model may be a stick-figure or volumetric [7, 18]. It may be interactively created by the user [2], or generated automatically during extraction [13].

To prepare the images for motion extraction, low-level image processing techniques [12], such as grey-scale conversion, filtering, and edge detection are often used. We, like many of the cited papers, use some of these techniques.

Tracking is achieved by minimizing the error between the model and image features [8, 19]. Matching a 2D model with the image may be followed by a separate 3D reconstruction [16], or the two may be combined, with joint and image constraints treated together [10]. Results using optical flow and probabilistic methods are encouraging [6, 11, 5, 17].

We use *active contours*, or *snakes* [14, 4], as the means of relating the model to the image. Active contours are sets of connected control points which are associated with features (usually edges) in the underlying image. Active contours can both automatically track features in a sequence of images, and be easily manipulated by the user with geometric transformations. This makes them very appropriate for our combined vision-and-graphics approach.

3 The Model

In this section, we describe our three-dimensional hierarchical models, using the horse as an example. We describe how to create a base generic model whose parts and proportions are appropriate for one typical horse. First, we scanned diagrams showing the anatomy of a horse from several directions [9, 23]. We interactively adjusted an initial model to match these diagrams. This model is later altered to represent differently sized and proportioned horses.

3.1 3D Model Representation

Our horse model consists of an underlying hierarchy (tree structure) describing the parent-child relationships of body *segments*, the default initial geometric relationship of segments, and geometric representations of the anatomical components (bones, muscles, generalized tissue, and skin). This model has been described in previous papers [24, 20, 21]; we only present what is new or essential to following the research described here. Though the model has muscles, generalized tissue, and skin, only the geometric structure and bones were used in the motion capture process.

Our horse has 78 segments that are based on the skeletal system and 5 soft-tissue segments (eyes, ears, nose) (Table 3.1). The first lumbar segment is the root of the hierarchy.

Segments Groups	Number Segments
Skull	2
Cervical Vertebrae	7
Thoracic Vertebrae	18
Lumbar Vertebrae	6
Pelvis/Sacral	1
Coccygeal Vertebrae	18
Right Shoulder/Foreleg	7
Left Shoulder/Foreleg	7
Right Hindleg	6
Left Hindleg	6
Eyes,Ears,Nose	5
Total	83

The *default geometry* specifies the relationship between each segment and its parent in a default rest posture. Each segment has its own *local coordinate frame*, where the z-axis of the frame is the longitudinal axis of the segment. Each segment frame is located relative to its parent segment by a 3D translation vector and a 3D Euler rotation vector. The default posture for the horse is standing symmetrically. By altering the default geometry, the one horse model can be altered to a horse of different size and conformation. For animation and repositioning, a further *state transformation* (a rotation for interior joints, and a rotation and translation for the root segment) is applied to appropriate joints.

We can restrict the positions that the state geometry represents using joint limits on rotation. While simple range limits can be applied to any joint (i.e., maximum and minimum positions for the X, Y, and Z rotations), we have found that reach-cone joint limits are more appropriate for joints with more than one degree of freedom. Reach cone limits are defined by a spherical polygon on the workspace sphere representing positions that can be taken by any segment (Figure 1). Our reach cone limits have a minor effect on program speed, and are described in detail elsewhere [25].

Bones and generalized tissue (geometric filling material) are rigid and associated with a single segment, and can be designed as polygon meshes, ellipsoids, or cylinders. Muscles are associated with origin and insertion points on segments and bones, and adjust themselves as joints between origins and insertions changes. Skin is an elastic triangle mesh associated with underlying components. All are located parametrically either on a segment, or, for skin, relative to whatever component they are anchored (usually a muscle). This allows components to adjust automatically when the default geometry is changed [24].

Because it is unwieldy to manipulate a structure with 83 joints, we group linear chains of joints into *supersegments* [15]. The program automatically creates supersegments for each linear chain, from the model root segment outwards. Whenever a branch occurs (as when the legs branch from the pelvis), the current supersegment ends and new ones begin. The multi-segment default supersegments in our model are the lumbar vertebrae, thoracic vertebrae, cervical vertebrae, caudal vertebrae, head, and the four legs. The user decides whether to use supersegments, or treat the segments individually. Supersegments can be interactively broken into smaller supersegments.

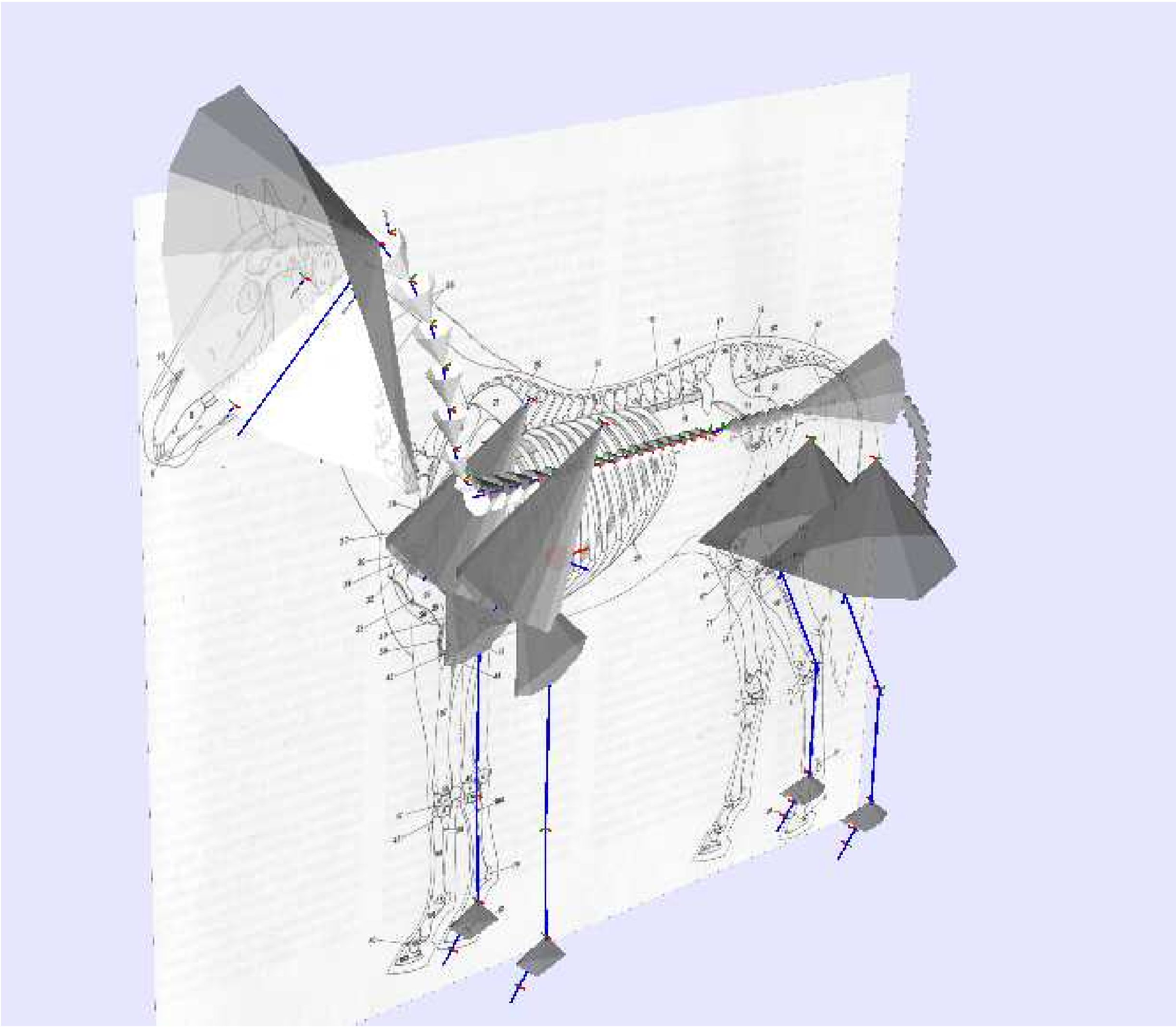

Figure 1: The reach cone joint limits for our generic horse model.

For example, the seven neck vertebrae can be manipulated coherently as a supersegment. A transformation applied to a supersegment is automatically pro-rated among its segments, either uniformly or according to a user-designed weighting. Supersegments define inverse kinematics chains; in the case of the neck, for example, such a chain reduces 21 degrees of freedom to three. However, supersegments are not limited to inverse kinematics; they are also used to apply distributed transformations such as proportionately scaling the legs or rotating the spine.

3.2 Matching Source Images

Using scanned-in images representing horse anatomy from several directions and with various anatomical parts [9] (images shown in figures are reproduced with permission of publisher), we interactively adjust our hierarchy geometry so that joints match the illustrated horse from side, front, back, and top (Figure 2). This gives us an appropriately proportioned initial horse geometry. Our geometric models of bones, muscles, and generalized tissue (most of which were initially from a human model) automatically adjust themselves to the new size of each body segment, but some interactive tweaking is necessary to match the diagrams (Figure 3).

The horse images were scanned in and textured-mapped to a movable rectangular image polygon, taking into account the estimated size of the horse so that its size in the virtual world was appropriate. The horse model is embedded in the plane of the image polygon as needed to match a particular image orientation. (By clearing the z-buffer after the image polygon is rendered, the horse is completely visible, although actually part is behind the image.) In order to maintain a consistent visual relationship between the model and the image, we translate and zoom the window using false perspective (including off-axis projection, `glFrustum()` in OpenGL). The focal length and other parameters of the perspective projection can be altered to mimic particular images.

To illustrate the relationship between the skeleton and the outside surface, and still be able to see the video images, we use the mouse to sketch in polyline outlines (*fauna snakes*) [3, 26] of the horse's surface and anchor the polyline vertices to relevant body segments. Anchoring is automatic, and can be restricted to a particular body segment or supersegment, or be unrestricted. This makes it possible to attach fauna snakes to an appropriate segment, even though it might not be the closest segment to the vertices (for example, the belly fauna snake vertices are attached to the vertebral segments, even though the legs are closer). While these fauna snakes are static, fauna snakes are also used for motion capture, and are described later (Section 5). Figure 3 shows the horse hierarchy and fauna snakes based on anatomical diagrams.

3.3 Adjusting to New Geometries

Horses come in very different shapes and sizes, from miniature horses as small as 20 inches (or 5 *hands*) at the withers, located between the back and neck, to large draft horses. The largest Percheron was 22 hands (7.33 feet). Typical riding horses are usually 15–17 hands. While the proportionate differences between breeds are not as extreme as for some species (such as dogs and cats), there are considerable differences between breeds and between horses of different ages.

Once we have established the generic horse geometry and anatomical components, we can adjust the geometry to different horses and the components scale proportionately. While the scaling is not always perfect, at worst, it provides a very good starting place for further adjustment. Figure 4 shows a year-old filly model adjusted to match video. Though not used in the motion capture described in this paper, our group has also developed automatic morphing algorithms for model adjustment (Figure 4, bottom) [21].

4 Feature Tracking

Once we have a three-dimensional model, we can attempt to animate it based upon video of an actual horse. The moving horse is an image *feature* that must be identified and extracted relative to the other pixels of the digitized image, and *tracked* from frame to frame. Images pixels belonging to a feature (the *foreground*)

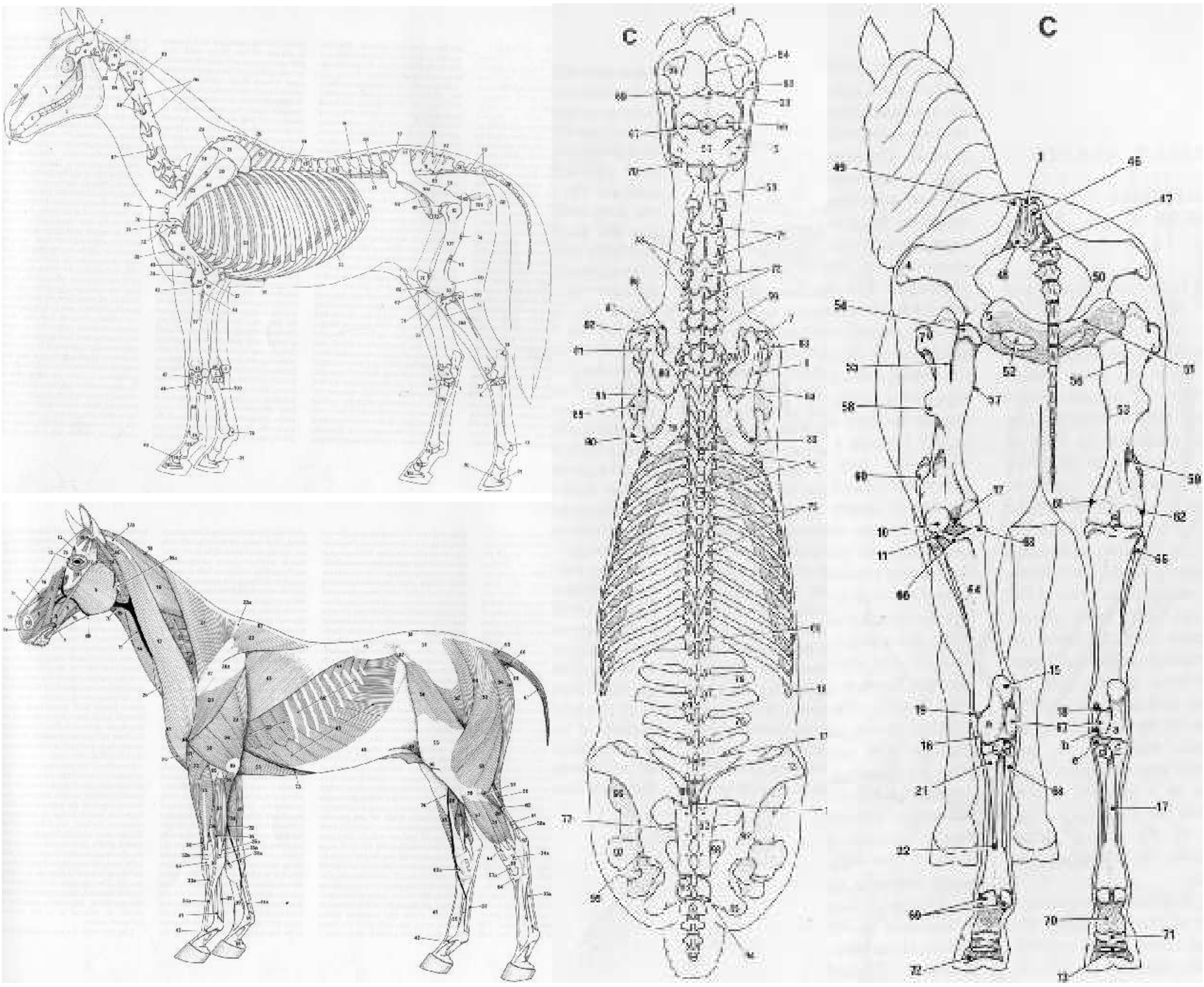

Figure 2: Examples of the horse diagrams used to create the generic horse[9] (reproduced with permission of publisher).

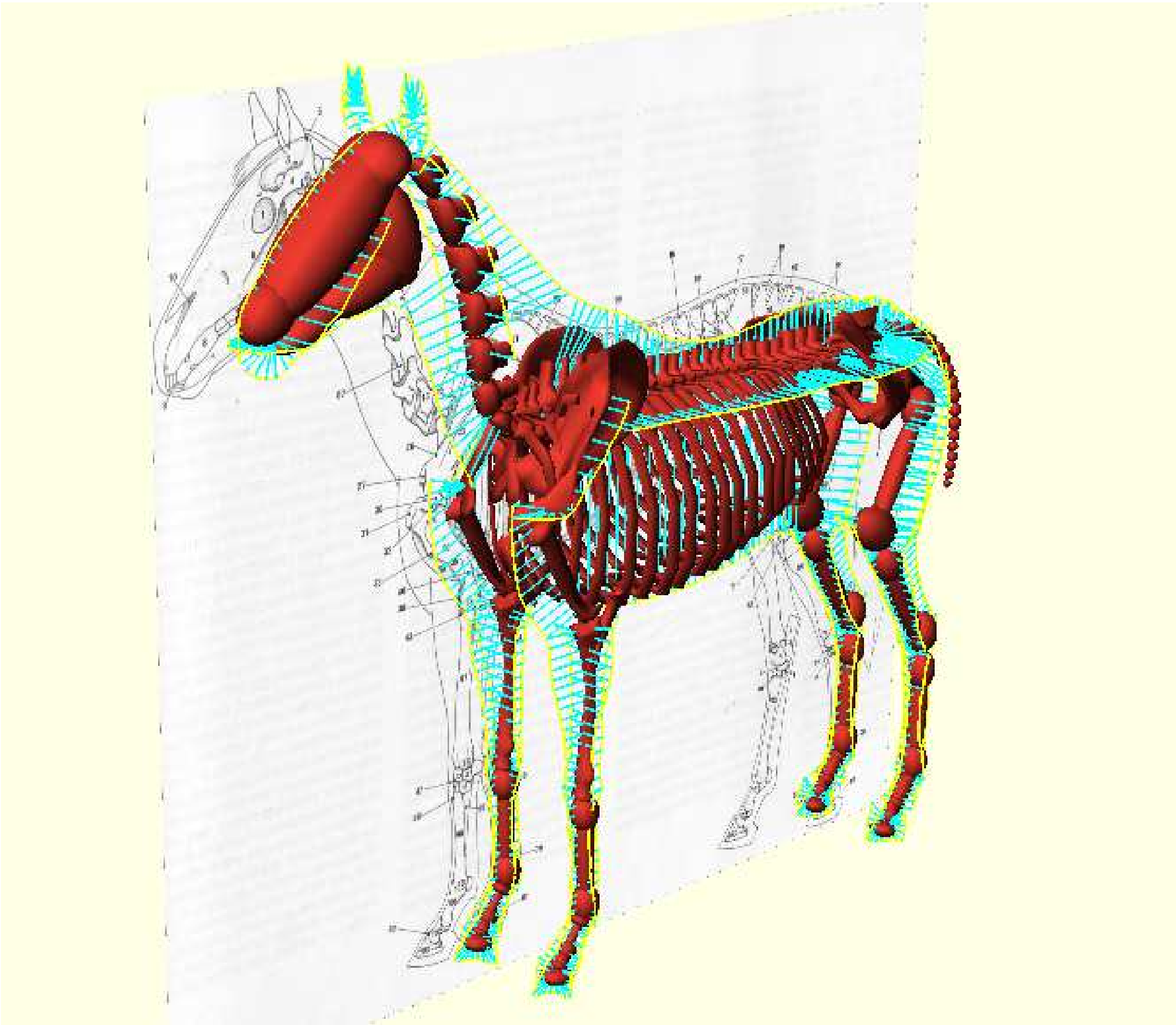

Figure 3: The generic horse created from the anatomical diagrams, showing the bones in red and the fauna snakes in yellow.

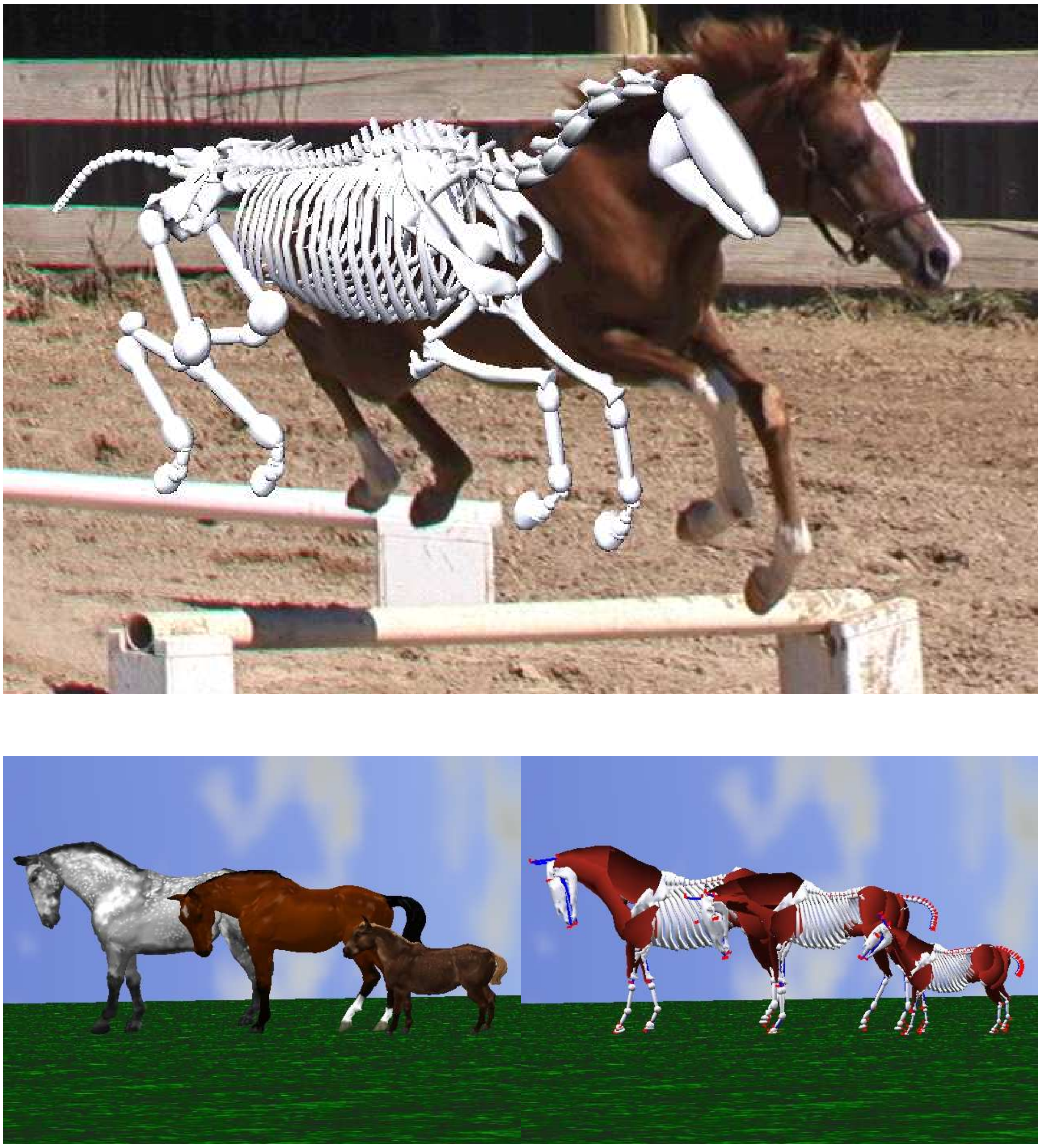

Figure 4: Horse models created by altering the generic model: top, a one-year-old Arabian filly (from video); bottom, models of a percheron, warmblood, and pony.

have identifiable aspects (texture, shape, position) that set them off from the image *background*, and these aspects remain similar and nearby in neighboring frames. Our strategy for identifying and tracking these two-dimensional features is explained in this section. In Section 5, we discuss the more difficult problem of deducing three-dimensional motion.

Because of the unrestricted nature of our video sequences, we depart from the concentration on fully automatic techniques which characterizes many computer vision approaches. We place no constraints on camera parameters, camera motion, feature motion, feature characteristics, lighting, or background. We accept the necessity of considerable user interaction to guide the feature extraction process. However, we provide the user with many techniques, interactive and automatic, to simplify the process.

For an image feature representation, we chose *active contours* [14]. Active contours (or *snakes*) are two-dimensional curves (in our case, polylines) that snap to and track image features. They are generally created by the user in an initial frame so that they are near to a recognizable image feature (an edge or a specific hue or texture). In later frames, the pixel neighborhood of the contours is examined and the contours moved to track features to which they were originally aligned.

We decided upon active contours because they so easily combine with user interaction. The user creates the contours by drawing them with the mouse around features considered important to the motion. During the tracking process, the user can intervene at any time and quickly adjust contours that are failing to track. Once satisfactory, the contour geometry is used in an inverse projection step to develop the three-dimensional articulated motion sequence (Section 5).

Our sequence of digitized images is taken from analog or digital video using video capture software and hardware common on personal computers. Our software reads video sequences in *avi* or *dv* format.

4.1 Low-Level Image Processing

Standard low-level image processing methods are useful for simplifying the image, removing extraneous detail and noise, and detecting edges. Image processing routines can be selected interactively, as the choice varies depending on the nature of the image. We experiment on a few images from the sequence to find a good combination, and then process the entire sequence. The most commonly used approaches are summarized below; a detailed account is given elsewhere [3].

4.1.1 Hue and Intensity Changes

Color images can be converted to greyscale first by replacing the RGB components of each pixel with the maximum of the original red, green, and blue components. If color is not a distinguishing feature of the character being tracked, this prepares the image for intensity-based edge detection. However, if the feature being tracked is of a distinct hue, it is better to retain the hue and use that in tracking.

Simple intensity changes can clarify the images. We can globally brighten or darken pixels, apply thresholds, or use a transfer function mapping present intensity values to new ones. Histogram equalization can be applied.

4.1.2 Smoothing

Smoothing filters remove noise and extraneous detail from an image. A simple noise reduction filter *convolves* an image with a *kernel* or *mask* such the value of each pixel is replaced by a weighted combination of its neighbors. Our three smoothing filters are *mean*, *median*, and *Gaussian*. The mean filter replaces a pixel with the average of itself and its eight immediate neighbors. The median replaces a pixel with the median value of these nine pixels . The Gaussian weights neighboring pixels according to a spatial Gaussian distribution. All filters can be applied multiple times as desired. We generally find the median filter most useful for our purposes, because it preserves sharp edges rather than blurring them.

4.1.3 Edge Detection

Our feature extraction largely depends on noticing *edges* in the image; i.e., regions where intensity or hue changes rapidly. A widely used and simple to implement edge detection algorithm was devised by Sobel [22]. The Sobel edge detector convolves each pixel of an image with a mask for horizontal differencing **H** and a mask for vertical differencing **V**.

$$\mathbf{H} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Each pixel is replaced with whichever convolution produces a larger absolute value. Pixel values are then normalized by the greatest pixel value, and scaled so that value is maximum brightness. Figure 5 illustrates this on the foal's head.

Figure 5: Left: the original image; middle: after change to greyscale; right: after Sobel edge detection.

4.2 Active Contours

We found active contours (or snakes) [14] desirable because they can drive the animation of the three-dimensional model. Our active contour is a sequence of line segments between control points (*polylines*) initially created interactively by the user in one frame, so that it is contiguous to a feature of interest, such as an edge.

When used for tracking, an active contour is reproduced in neighboring frames, where it self-adjusts or is manually adjusted to track the same feature. In the complicated sequences we use to capture animal motion, tracking requires heavy intervention by the user, but in simple sequences it is mostly automatic. There are generally several active contours in each frame, each associated with a particular feature of interest in the image, usually a part of the body of animal being tracked. Figure 6 shows the active contours used to extract the motion of the foal walking.

An active contour for one frame can be used to automatically generate new versions of itself in other frames in three ways: (1) by simply copying it from another frame; (2) by extrapolating the control points

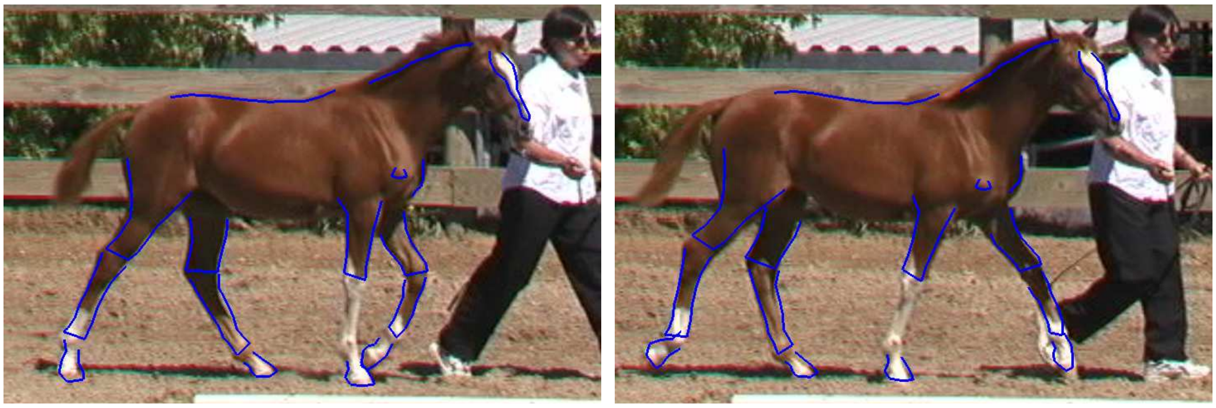

Figure 6: The active contours for the foal walking: at left, in the frame where they were created; at right, in a later frame.

from two previous frames, as if the velocity were constant; and (3) by interpolating between the positions of control points in specified frames previous or later than a given frame. Once added to new frame, several tracking heuristics are applied, as summarized in Section 4.3.

Active contours can have problems tracking when motion is fast (features are many pixels away in the next frame) and when features aren't consistent (features are occluded or fade away). Active contours can also be subject to "creeping" along image edges. Thus, the user can at any point interactively adjust either the whole contour (rotating and translating it as if it were rigid) or adjust individual points that sneak away from their features.

4.3 Feature Tracking with Active Contours

Given one or more initial positions where the contours are appropriately positioned relative to image features, the contours must then automatically adjust themselves to track these features. To do this, virtual *forces* are applied to the active contours. These forces may be categorized as *internal forces*, which represent a contour's interaction with itself; *external forces* which arise from the contour's interaction with the image; and *gravity forces*, which pull the contour in a particular direction. The heuristics used to determine virtual forces are under development and not very satisfactory at present, so the descriptions are brief, with additional details given elsewhere [3, 26].

Internal forces (length- and angle-preserving forces) help maintain the overall starting shape of the contour, while possibly allowing it to grow or shrink. The length-preserving force ($\mathbf{F}_{length}$) can either maintain the relative or absolute length of each active contour edge by applying an expansion or compression force. The angle-preserving force ($\mathbf{F}_{angle}$) maintains the original exterior angle θ between consecutive edges by applying a restoring torque.

Image forces are comprised of an *edge force* generated by the image intensity gradient in the areas covered by the contour and a *hue force*, generated by hue differences. Image forces cause the active contour to react to characteristics of the underlying image. The image *edge force* reacts to the image brightness; it is based on the gradient of a bilinear intensity function of the four image pixels surrounding an active contour point. The *hue force* reacts to image color in a similar fashion. The edge forces can be calculated either at the control points only, or at one-pixel intervals along the edge and then applied to the control points. Edge forces are most useful on images after computer vision edge detection is applied (Section 4.1.3).

Gravity forces can pull a contour edge along its perpendicular or toward a specified location. They are rarely used in tracking motion of animal limbs, but are sometimes useful for "shrink wrapping" a snake down to a closed contour in the image.

The relative weights of different forces can be controlled by the user. Forces are applied iteratively until the sum of the forces is very small or until a maximum number of iterations is reached.

5 Animation from Video

Finally, we use the two-dimensional active contours that represent feature motion to extract three-dimensional positions at joints in our model. This is an extremely under-constrained problem, particularly when motion is not parallel to the image plane. Again we rely on user input to make the capture feasible. The steps in the process are as follows:

1. As an preliminary step, the user must adjust the basic geometry of the generic model to fit the subject in the video. This is done interactively, using whichever frames best illustrate different parts of the body. The use of anatomical components and static fauna snakes described in Section 3 ease the process

2. The user then aligns the model with image features in one or more model key frames, where the relationship of model and image is clear. If more than one model key frame is used, interpolation can then be used between them to create an initial approximate animation of the model.

3. The active contours are automatically anchored to the model in *anchor frames*, at which time we rename them *fauna snakes*. The model must be aligned properly with the fauna snakes in these frames, which are generally also model key frames (Step 3).

4. The fauna snakes are used to "pull" the model into position in the remaining frames.

This is an iterative process, and proximal body parts are generally captured before distal ones. Filtering can be applied to the motion curves thus created, to reduce noise.

5.1 Anchoring Fauna Snakes to the Model

Individual fauna snakes are anchored to the model in *anchor frames* where the relationship between the model and the snake is clear. Snakes can be re-anchored at any other frame; this is useful where the motion capture is likely to be problematic, such as the beginning of an out-of-image-plane rotation. It is not necessary that all fauna snakes are anchored in the same frames.

Anchoring involves finding the nearest point on a segment longitudinal axis to fauna snake vertex. Distance is calculated between the two-dimensional projection of the longitudinal axis of a segment and the i-th snake vertex $\mathbf{P_i}$. The projection is to the image plane, which is the world-space XY plane. When the nearest point is found in two dimensions, each fauna snake vertex is given a Z-value equal to that of the near point on the segment axis. Each vertex is then converted to the local coordinate frame of the anchoring segment, establishing a relationship between the world space features represented by active contour points, and the three-dimensional model. The local representation of a fauna snake vertex $\mathbf{P_i}$ is the anchor $\mathbf{A_i}$.

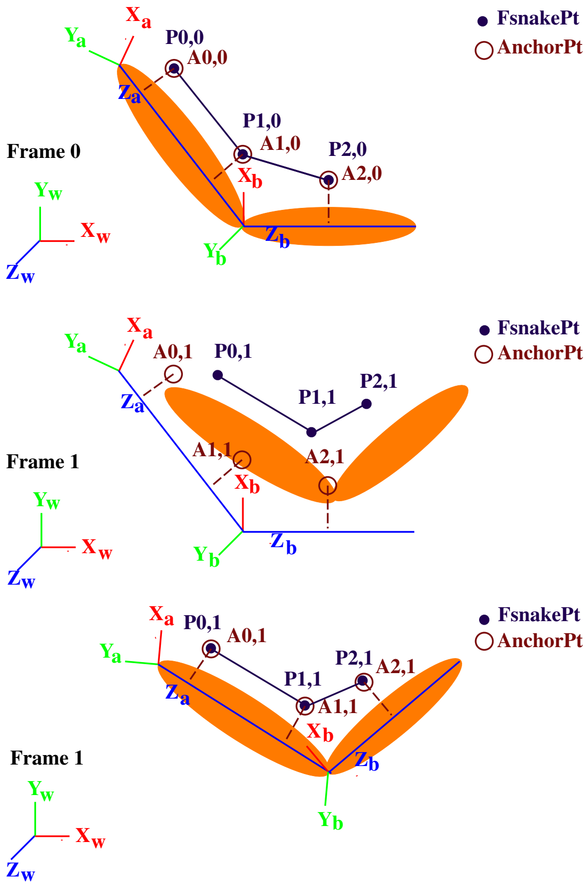

Figure 7: A two-segment model tracks a figure in an image sequence. In the upper image, the fauna snake is anchored to the two segments. In this frame the anchor vertices and fauna snake vertices project to the same position. In the middle image, a different frame is shown, where the image (in orange) and the tracking fauna snakes have moved, the model is not longer aligned with the image features. In the bottom image, a translation and rotation adjustment has occurred and the anchor vertices and fauna snake vertices and again in alignment.

The segments examined in anchoring any fauna snake can be all segments of the model, the segments of a supersegment (a chain of segments), or a particular segment. This is helpful where the desirable anchoring segments are not the nearest segments of the model, as in the case of the abdominal fauna snake. Snakes representing the lower edge of the abdomen should move with the appropriate vertebrae, though they may be closer to the longitudinal axes of the legs.

The world space fauna snake vertex $\mathbf{P_i}$ and its local segment representation $\mathbf{A_i}$ project to the same point in an *anchor frame*; they generally do not in other frames. The fauna snakes in other, non-anchor frames have moved to follow the image features (Section 4.3). The geometric relationship between the fauna snake anchor vertices and the fauna snake vertices on other frames is used to adjust the model to follow the moving features (see Figures 7 and 8).

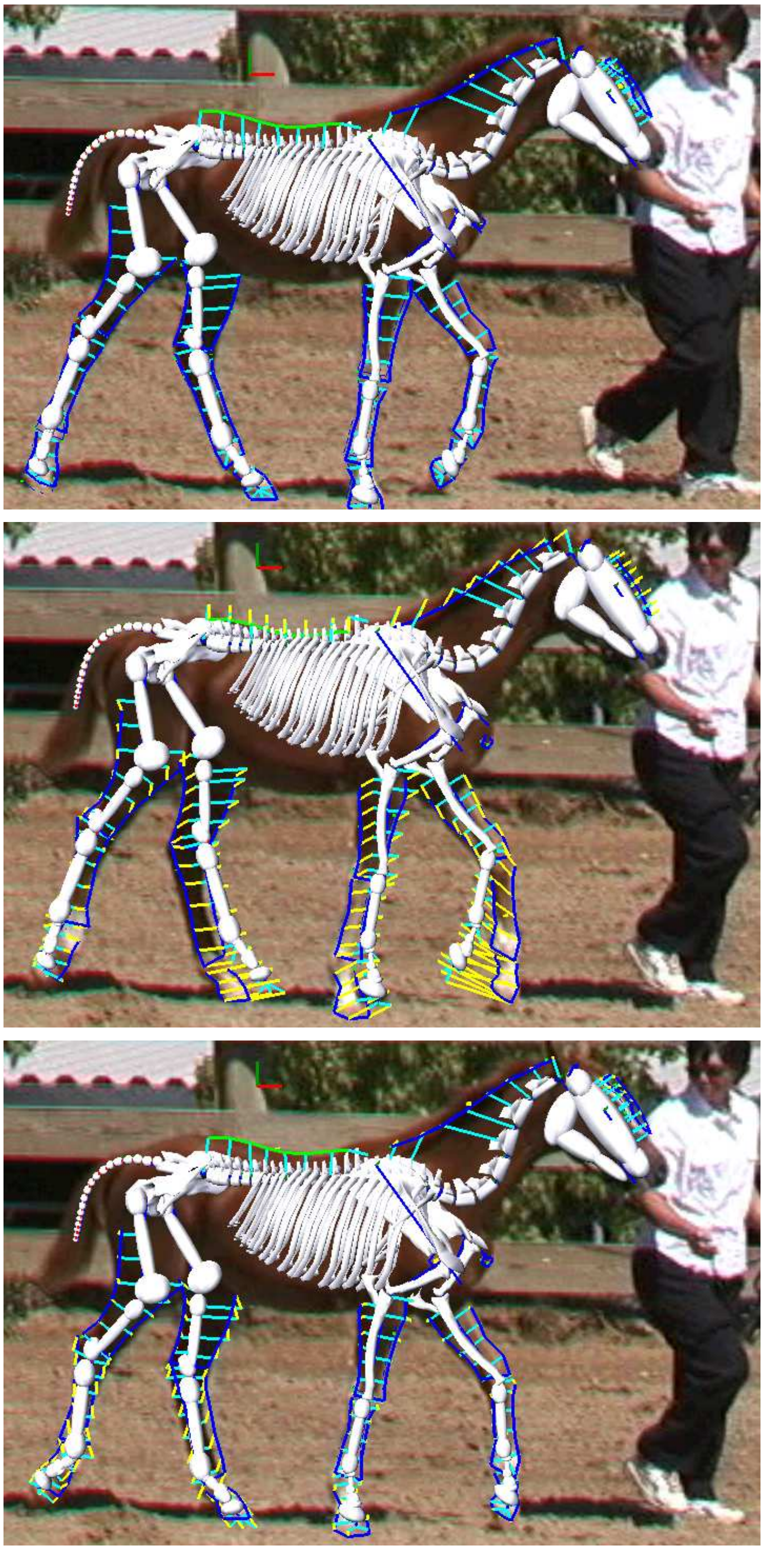

Figure 8: Adjusting the model to match the fauna snakes: top, the anchor frame; middle, the next frame before adjustment of the 3D model; bottom, the same frame after the 3D model has been "pulled" to the fauna snakes.

5.2 Automatically Repositioning the Model

Repositioning the model in non-anchor frames involves minimizing the distance between the world space fauna snake vertices and the world-space projection of their anchors on segments. The repositioning is "kinematic," although it is natural to speak of the adjustment as "pulling."

The user controls the type of adjustment that is used. Only the root segment of the model is translated, but any fauna snake can contribute to the translation. Any segment can be rotated to align with a fauna snake. If the motion is already parallel to the image plane (world space XY-plane), an accurate *image-plane* rotation in that plane can be found, but in general the real motion is also out of the image plane. For these rotations, rotation can be be directly out of the image plane, or may be an Euler rotation about the local segment X, Y, and Z axes. Local axes not involved in the adjustment retain the values calculated from the model key frames. This is useful for problematic rotations, such as those around the longitudinal (local Z) axis. Joint limits also restrict motion to reasonable values.

Any segment may be affected by: (1) only fauna snake vertices anchored to that segment; (2) by any fauna snake vertex anchored to the chain of segments (supersegment) to which the segment belongs; or (3) by any contour points distal to the segment. For example, if all the segments of the leg have their own fauna snakes, it would be best to use case 1. If only the hoof has a fauna snake, it would be best to use case 2 or 3.

Adjustments must be applied proximal to distal in the hierarchy. It is generally most useful to adjust the model iteratively, first finding a solution for the root segment, and then the spinal column, and then the appendages.

5.2.1 Translation

Fauna snakes involved in translation adjust the position of the model root segment parallel to the image plane. The translation is calculated using the difference between the projected positions in world space of each contour point $\mathbf{P_i^{pw}}$ and its anchor $\mathbf{A_i^{pw}}$. The total translation change of the root segment from n translate fsnake points in frame j is:

$$\mathbf{F_j} \quad = \quad \sum_{i=1}^{n} \frac{\mathbf{P_{i,j}^{pw}} - \mathbf{A_{i,j}^{pw}}}{n}$$

5.2.2 Image-Plane and Euler Rotations

For rotations, some set of n fauna snake vertices exert virtual torques on a segment (see Figure 9). For each fauna snake vertex affecting a given segment, the world-space vertex $\mathbf{P_i}$ and the local anchor vertex $\mathbf{A_i}$ (which is stored in the local frame of the segment to which it is anchored) are converted to local coordinate frame of the segment being adjusted. These are then projected to a plane perpendicular to the desired axis of rotation. The virtual torque is dependent on the angle θ between these two projected vectors, weighted by the relative squared distance of the projected anchor from the segment frame origin r_i. The actual angular change θ_s applied to segment s due to n fauna snake vertices is then:

$$\theta_s \quad = \quad \frac{\sum_{i=1}^{n} \theta_i r_i^2}{\sum_{i=1}^{n} r_i^2}$$

The desired axis of rotation would be the perpendicular to the image plane, for motion parallel to that plane. It may also be any of the local X, Y, Z-axes of the segment being adjusted.

5.2.3 Direct Out-of-Plane Rotation

If the above alignment rotation has been done parallel to the image plane, the segment is already aligned as well as it can be in that plane. However, if the actual motion has a component out of the plane, a further

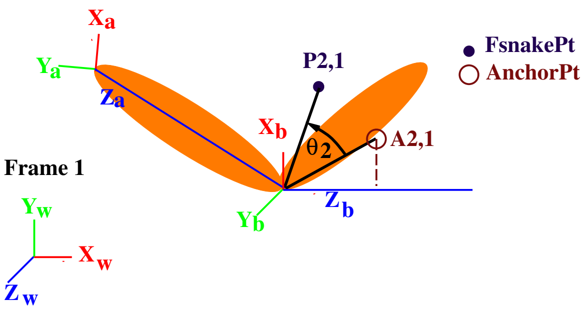

Figure 9: θ_2 is the image plane correction angle due to $\mathbf{P_{2,1}}$ and $\mathbf{A_{2,1}}$.

rotation outward, about an axis perpendicular to both the segment longitudinal axis and the image plane normal is appropriate. This rotation will not affect the projected direction of the longitudinal segment axis.

Calculations are done in world space, but applied in local segment space. To simplify the mathematics, we effectively translate the segment to the world origin and rotate around the world Z-axis to place the longitudinal segment axis in the world $Y = 0$ plane; call this rotation angle α. After computing and performing the out-of-plane rotation, we invert the rotation by α and the translation. In the new frame rotated by α, the out-of-plane rotation is simply a rotation around world Y by some angle ϕ and the error for each contour point is simply the difference in X-value between the point and its anchor. Let $\mathbf{P}_i^{\alpha w}$ and $\mathbf{A}_i^{\alpha w}$ be the positions of the contour point i and its anchor after rotation by α. Let $\mathbf{A}_i^{\alpha w}(\phi)$ be the anchor position after a further rotation by ϕ around the Y-axis. Subscripts of x, y, and z denote the X, Y, and Z components of vectors. Thus $A_{ix}^{\alpha w}(\phi) = A_{ix}^{\alpha w}\cos\phi + A_{iz}^{\alpha w}\sin\phi$.

To find the solution angle ϕ, we (approximately) minimize the squared error E, which is the sum over i of the squared distance in X between the i-th contour point and its anchor, in the rotated frame. We (approximately) find the ϕ for which $\partial E/\partial\phi = 0$.

$$
\begin{aligned}
E &= \sum_i \left(P_{ix}^{\alpha w} - A_{ix}^{\alpha w}(\phi)\right)^2 \\
\frac{\partial E}{\partial \phi} &= \sum_i 2\left(A_{ix}^{\alpha w}(\phi) - P_{ix}^{\alpha w}\right)\frac{\partial A_{ix}^{\alpha w}(\phi)}{\partial \phi}
\end{aligned}
$$

Dropping higher-order terms in $\tan\phi$, the solution is

$$
\tan\phi = \frac{\sum_i A_{iz}^{\alpha w}(P_{ix}^{\alpha w} - A_{ix}^{\alpha w})}{\sum_i \left(A_{ix}^{\alpha w}(P_{ix}^{\alpha w} - A_{ix}^{\alpha w}) + (A_{iz}^{\alpha w})^2\right)}
$$

5.3 Occlusion and Camera Motion

A virtual anatomy can be rendered with our model [24] and used to estimate when contour points become occluded. (So far, we have only tested this with ellipsoidal approximations to limbs.) Because contour anchor points have a world Z value determined by their anchor segment, they become hidden when another part of the anatomy is rendered in front. By giving the anchors a known unique color and checking the color at the projected anchor position in the frame buffer, visibility can be detected. When the anchor is invisible, forces are not calculated for the contour point, though it is still influenced by the neighboring points. Although contour points and their anchors do not project to the same position, they are generally close enough that this approach improves tracking.

Our rudimentary approach to removing camera motion is to treat the world coordinate system as an independent segment. Contour points are anchored to this segment in the normal manner and changes in world space are produced. As movement of the world, in which the image plane and the human model exist, is the inverse of the movement of the camera, this motion can be tracked and removed from the human model. In terms of a motion library, other methods of separating the human motion from the world can be applied, such as calculating the position of a known stationary body part or calculating the distance traveled in a stride.

6 Results

We have experimented with a variety of video sequences, both humans and animals Further results, including animations, can be seen on our web site: `www.cse.ucsc.edu/~wilhelms/fauna`.

The horse examples shown here are of a year-old Arabian foal. We first adjusted the generic horse geometry for this breed and age, using captured video of the foal walking. As we had video both from the side and front, we could easily adjust the geometry in three-dimensions.

We captured the foal walking parallel to the image plane largely using fauna snakes, with some user adjustment. Such motion is relatively easy to capture, the motion being largely two-dimensional. Figures 6 and 8 show frames from one walk cycle.

We then tried a more interesting sequence of the foal running free and jumping an obstacle. This sequence presented problems for active contour tracking, though they were easily adjusted interactively. In this case, the horse motion is at an angle to the image plane, although it stays relatively planar. We made three model key frames to gain the general motion and then used fauna snakes to align segments. For this motion, considerable user interaction was needed. Figure 10 shows the results with our yearling model. We did not create a yearling skin; however, the bottom two images show our warmblood model assuming the positions of the foal.

Finally we captured the motion of a child playing in the surf from a very difficult sequence where the boy spins entirely around and moves behind an adult. The results and the fauna snakes involved are shown in Figure 11. In this case, five fauna snakes were used.

7 Discussion and Conclusions

We have found that automatic tracking with active contours is most successful for moderately slow motions and where the background is relatively simple. Here we challenged the method with complex, fast, non-planar motion against non-trivial backgrounds. An improved active contour algorithm, or an alternate feature tracking approach would be helpful. Segmenting and tracking moving objects is an active area in the computer vision community; we hope to avail ourselves of some of their newer methods. However, for our scenes, we found interactively adjusting the active contours easy and fast.

We similarly found that user supervision was necessary to extract reasonable three-dimensional motion from complicated motion out of the image plane. However, we were able to extract motion that gives the timing and "feel" of a motion, and is reasonably close geometrically.

The motion extracted needs to be further processed to remove camera motion, and refined for constraints such as feet placement on the ground.

Acknowledgements

This research was supported by NSF Grants CCR-9972464 and CDA-9724237. We wish to thank Leon Atkinson-Derman, Luo Hong, Maryann Simmons, and Mark Slater for their help with this project.

References

[1] J. K. Aggarwal, Q. Cai, W. Liao, and B Sabata. Nonrigid motion analysis: Articulated and elastic motion. *Computer Vision and Image Understanding*, 70(2):142–156, 1998.

[2] K. Akita. Image sequence analysis of real world human motion. *Pattern Recognition*, 17(1):73–83, 1984.

[3] Leon Atkinson-Derman. Tracking on the wild side – using active contours to track fauna in noisy image sequences. Master's thesis, University of California, Santa Cruz, Santa Cruz, CA 95064, June 2000.

[4] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, 1998.

[5] C. Bregler and J. Malik. Estimating and tracking kinematic chains. In *Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998.

[6] Tat-Jen Cham and James M. Rehg. A multiple hypothesis approach to figure tracking. In *Proc. Computer Vision and Pattern Recognition*, pages 239–245, Ft. Collins, CO, June 1999.

[7] Z. Chen and H. J. Lee. Knowledge-guided visual perception of 3D human gait from single image sequence. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):336–342, 1992.

[8] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3D. In *Proc. of the IEEE Fifth International Conference on Computer Vision*, pages 764–770, Cambridge, Mass., 1995.

[9] Peter C. Goody. *Horse Anatomy: A Pictorial Approach to Equine Structure*. J. A. Allen, London, 1983.

[10] Y. Hel-Or and M. Werman. Constraint fusion for recognition and localization of articulated and constrained objects. *International Journal of Computer Vision*, pages 5–28, July 1996.

[11] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int'l Journal of Computer Vision*, 29(1):5–28, 1998.

[12] Anil Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall International Editions, 1989.

[13] I. A. Kakadiaris, D. Metaxas, and R. Bajcsy. 3D human body model acquisition from multiple views. In *Proc. of the IEEE Workshop on Non-Rigid and Articulated Objects*, pages 618–623, Boston, MA, June 20–23 1995.

[14] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[15] Jeff Lapierre and Jane Wilhelms. Matching anatomy to model for articulated body animation. In *Proceedings of 1999 IASTED Computer Graphics and Imaging Conference*, Palm Springs, Ca., 1999.

[16] Daniel D. Morris and James M. Rehg. Singularity analysis for articulated object tracking. In *Proc. Computer Vision and Pattern Recognition*, pages 289–296, Santa Barbara, CA, June 23–25 1998.

[17] Alex Pentland and Bradley Horowitz. Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7), 1991.

[18] F. J. Perales and J. Torres. A system for human motion matching between synthetic and real images based on a biomechanic graphical model. *Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 83–88, 1994.

[19] J. M. Rehg and T. Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. *Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 16–22, 1994.

[20] Philip J. Schneider and Jane Wilhelms. Hybrid anatomically based modeling of animals. In *Computer Animation '98*, pages 161–169, June 1998.

[21] Maryann Simmons, Jane Wilhelms, and Allen Van Gelder. Model-based reconstruction for creature animation. *ACM Symposium on Computer Animation)*, pages 139–146, July 2002.

[22] I. Sobel. An isotropic 3x3 image gradient operator. *Machine Vision for Three-Dimensional Scenes*, pages 376–379, 1990.

[23] George Stubbs. *The Anatomy of the Horse.* Dover Publications, Inc., New York, 1976.

[24] Jane Wilhelms and Allen Van Gelder. Anatomically based modeling. In *Computer Graphics (ACM SIGGRAPH Proceedings)*, Aug. 1997.

[25] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *Journal of Graphics Tools*, 6(2):27–41, 2001. Available at `ftp://ftp.cse.ucsc.edu/pub/avg/jtl.pdf.`

[26] Jane Wilhelms, Allen Van Gelder, Leon Atkinson-Derman, and Hong Luo. Human motion from active contours. In *IEEE Workshop on Human Motion*, Austin, TX, December 2000.

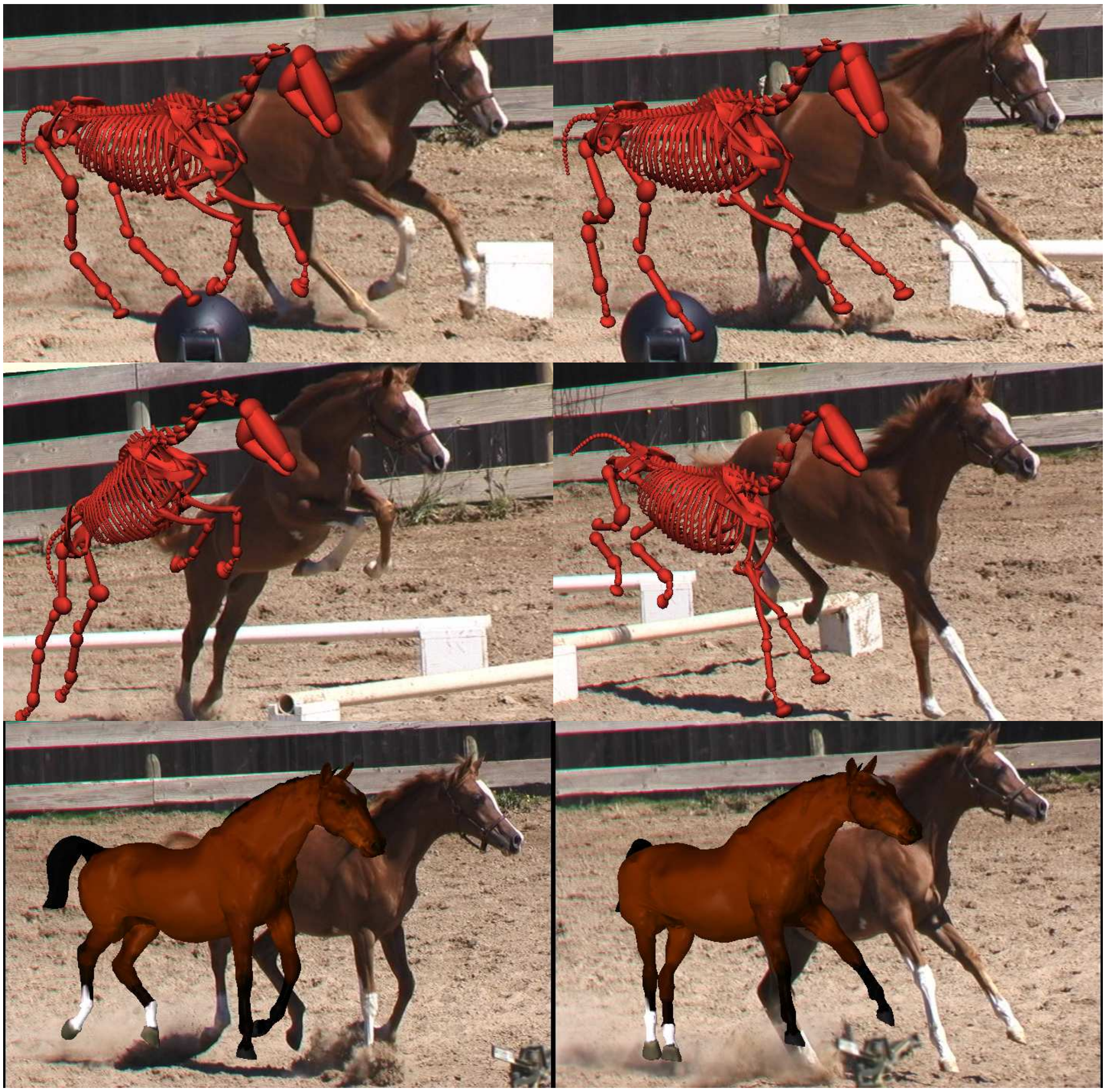

Figure 10: Foal jumping an obstacle, shown at angle angle to the image plane.

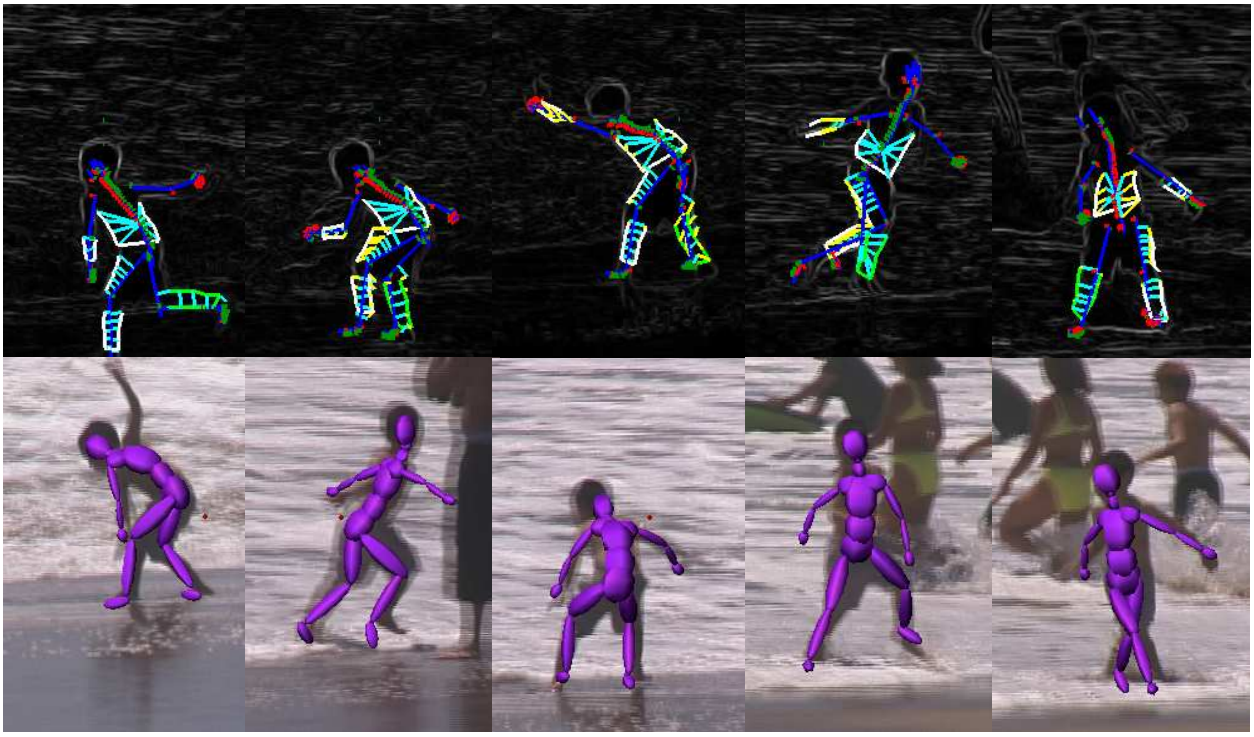

Figure 11: Boy in the surf: Top shows fsnakes, bottom ellipsoidal model. (Right arm not tracked.)