Chapter 7

Time and Pose Cues for Motion Emphasis

In this chapter we extend the gamut of motion emphasis cues in the Video Paintbox to include animation timing effects; specifically, cartoon "anticipation" effects and motion exaggeration. Our process begins by automatically recovering articulated "dolls" of subjects moving in the plane. By fitting our previously tracked (Chapter 6) features to this articulated structure we are able to describe subject pose, at any instant, as a point in a high dimensional pose space. We then perform local and global distortions to this pose space to create novel animation timing effects in the video. We demonstrate that these timing cues may be combined with the visual cues of Chapter 6 in a single framework.

7.1 Introduction

Although animators may emphasise the motion of a subject using an array of purely visual tools (for example making marks, or performing deformations), they may also choose to manipulate the timing of an action to stylise an animation — we term this class of effects "time and pose" motion cues. In this Chapter we describe the subsystem for inserting these cues in to the video sequence, and explain how this work may be integrated with the motion cues of Chapter 6 to complete the motion emphasis framework within the Video Paintbox.

A common "time and pose" cue used by animators is "anticipation", which is used to indicate the onset of motion. Typically, an object exhibiting anticipation is observed to recoil briefly, as if energy is somehow being stored in preparation for a release into motion; the resulting effect is sometimes referred to as "snap" by animators (Figure 7-5). According to animator Richard Williams [169] (pp. 274–275) there are three rules

governing anticipation:

1. The anticipation is always in the opposite direction to where the main action is going to go.

2. Any action is strengthened by being preceded by its opposite.

3. Usually the anticipation is slower than the action itself.

A further motion emphasis technique is "exaggeration". Just as caricaturists often draw faces by exaggerating their characteristics [98], in a similar vein animators often exaggerate the individual characteristics of a subject's movement. Such manipulations of motion clearly demand large temporal windows for trajectory analysis. In the case of anticipation, at any given instant the system must be aware of future motion that may subsequently take place. Likewise the identification of patterns and characteristics within a subject's movement over time are a prerequisite to exaggerating those motions within the video sequence. A per frame, sequential approach to video driven AR therefore precludes the insertion of time and pose cues into the resulting animation.

The framework we propose operates by varying the pose of the subject over time, manipulating the features tracked by the Computer Vision component of Section 6.3 prior to their rendering by the Computer Graphics component of Section 6.4. Our initial experiments attempted to create time and pose cues by directly manipulating the tracked LCATs of features; this did not synthesise motion cues of satisfactory aesthetic quality for anything but the simplest of motions (for example a translating ball). We instead opted to recover the articulated structure of tracked subjects, to create a higher level description of subject motion — the subjects' pose. Manipulation of this pose over time allows us to augment the existing gamut of motion cues in the Video Paintbox with convincing temporal effects.

We begin by automatically recovering an articulated "doll" from the set of features tracked in Chapter 6. Once the "doll" structure has been recovered, the tracked polygons for each frame are fitted to the articulated structure, creating a series of "pose vectors". These pose vectors are points in a high dimensional "pose space", each of which specify the configuration of the articulated subject at a particular instant in time. As the articulated subject moves over time, the vectors can be thought of as tracing a smooth trajectory in this high dimensional pose space. By performing judicious transformations on the spatiotemporal trajectories within this space we may affect the pose of the subject over time, and so introduce "time and pose" cues into the video sequence.

Our desire to process post-production video of general content (for example a person, or a metronome) from a single view-point has restricted the range of literature we have

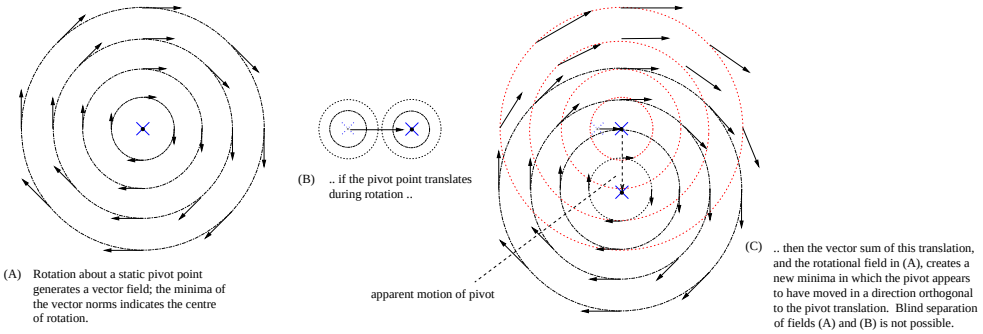

Figure 7-1 Illustrating the difficulty of recovering pivot point location from rotation, in the case of a moving pivot. Under instantaneous motion, the combination of rotation and pivot shift causes an apparent translation of pivot location orthogonal to the direction in which the pivot have moved (see Appendix A.6 for a proof).

been able to draw upon for the recovery of articulated structure and pose. Most passive motion capture systems assume provision of an *a priori* articulated model [17, 76], are tailored to the specific problem of tracking humans [10, 89], or require multiple camera viewpoints [11, 76, 88]. We have developed a technique for recovering articulated structure and pose that does not require such constraints, but does make a number of alternative assumptions which are more compatible with our application. First, we assume that the subject exhibits a rigid, hierarchical articulated structure. Second, we assume that the subject's motion is planar within the camera compensated sequence in which features are tracked (Section 6.3.1). Lastly, we assume that pivot points in the structure are static relative to their attached features. This avoids the problem of localising moving pivot points solely from polygonal data; this problem is intractable in principle (see Figure 7-1 and Appendix A.6).

7.2 Recovery of Articulated Pose in the plane

Our process begins by recovering the articulated structure of a subject moving in the plane; this is achieved by analysing the motion of the feature polygons tracked by the Computer Vision component in Chapter 6. The basic technique is to estimate a pivot point between each pair of tracked features, and then to assess the quality of these pivot points to determine which feature pairs are truly articulated in the physical world. These physically articulated feature pairs, and their associated pivot points, form a hierarchical representation of the rigid articulated subject being tracked. At each time instant we fit this articulated structure to the tracked feature polygons, creating a set of joint configurations represented numerically by a "pose vector". This pose representation is manipulated to introduce cartoon-like motion cues (Section 7.4).

7.2.1 Four Algorithms for Recovering inter-feature pivot points

We now describe four different algorithms for determining the pivot point between two, potentially articulated, features "A" and "B". Each feature is considered be static within its own reference frame. We write $\underline{\underline{F}}_{A(t)}$ and $\underline{\underline{F}}_{B(t)}$ as the affine transformations from each of these reference frames to world coordinates, respectively. In the first frame ($t = 1$) the reference frames are coincident with the world basis, and these transforms are the identity.

We have to assume a static pivot point in our work; by this we mean the pivot of A about B will be static relative to both frames $\underline{\underline{F}}_{A(t)}$ and $\underline{\underline{F}}_{B(t)}$ — in world coordinates the pivot can, of course, move freely.

In our descriptions of each algorithm we consider the motion of A about B, within $\underline{\underline{F}}_{B(t)}$; i.e. we assume feature B's motion has been "subtracted" from feature A's. We denote the vertices of feature A as $\underline{\underline{A}}_t$ (a series of column vectors representing the homogeneous world coordinates of each vertex at time t). Likewise we write $\underline{\underline{B}}_t$ for feature B. In homogeneous form, we may express the motion of feature A in the frame of B as $\underline{\underline{A}}'_t$, where:

$$\begin{aligned} \underline{\underline{A}}'_t &= \underline{\underline{F}}^{-1}_{B(t)} \underline{\underline{A}}_t \\ &= (\underline{\underline{B}}_1 (\underline{\underline{B}}_t)^+) \underline{\underline{A}}_t \end{aligned} \tag{7.1}$$

where superscript + specifies the Moore-Penrose "pseudo inverse" [123]. We use the notation $\underline{\underline{A}}'_t$ throughout our description of the four algorithms.

7.2.2 Closed Form Eigen-solutions

Algorithm 1: Decomposition of affine transform via SVD

Our first algorithm accepts two samples of the feature polygon at different time instants $[t, t + \Delta]$ (where Δ is a user defined temporal interval), and outputs a least squares algebraic approximation to the pivot point location. This is achieved by decomposing the compound affine transform for rotation about an arbitrary point in 2D. The motion of A over time interval $[t, t + \Delta]$ may be described as $\underline{\underline{A}}'_{t+\Delta} = \underline{\underline{M}}\,\underline{\underline{A}}'_t$, where $\underline{\underline{M}}$ is an affine transform defined as follows:

$$\underline{\underline{M}} = \begin{bmatrix} \underline{\underline{R}}_{(\theta)} & -\underline{\underline{R}}_{(\theta)} \underline{p} + \underline{p} \\ 0 & 1 \end{bmatrix} \tag{7.2}$$

Where $\underline{p}$ denotes the pivot point, $\underline{\underline{R}}_{(\theta)}$ the 2D rotation matrix and θ the degree of

rotation. Introducing the notation $\underline{m} = (m_1, m_2)^T$:

$$\underline{m} = -\underline{\underline{R}}_{(\theta)}\underline{p} + \underline{p} \tag{7.3}$$

$$\underline{\underline{M}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & m_1 \\ \sin(\theta) & \cos(\theta) & m_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

Each of the unknowns in $\underline{\underline{M}}$ is recovered by forming a homogeneous linear system, using all n vertices of the feature A at time t ($[x_{1..n}, y_{1..n}]^T$) and at time $t+\Delta$ ($[x'_{1..n}, y'_{1..n}]^T$). The system is solved using SVD:

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 & -x'_1 \\ y_1 & x_1 & 0 & 1 & -y'_1 \\ & ... & & ... & \\ x_n & -y_n & 1 & 0 & -x'_n \\ y_n & x_n & 0 & 1 & -y'_n \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ m_1 \\ m_2 \\ 1 \end{bmatrix} = 0 \tag{7.5}$$

Rearrangement of $\underline{m}$ yields an expression for $\underline{p}$:

$$\underline{m} = (\underline{\underline{I}} - \underline{\underline{R}})\underline{p} \tag{7.6}$$

$$\underline{p} = (\underline{\underline{I}} - \underline{\underline{R}})^{-1}\underline{m} \tag{7.7}$$

In cases of small rotary motion the value for $\underline{p}$ becomes increasingly unreliable — very small errors in the measurement of tracked feature vertices have a very large influence on the estimated pivot location. If we write out the components of $\underline{p} = (p_x, p_y)^T$ then this behaviour is explained by the large denominators that result as θ tends to zero:

$$p_x = (m_2 \sin\theta - m_1(1 - \cos\theta))/2(\cos\theta - 1) \tag{7.8}$$

$$p_y = (m_1 - p_x(1 - \cos\theta))/\sin\theta \tag{7.9}$$

With no rotational component at all, $\underline{p}$ is undefined.

Extension to Algorithm 1: Estimation over multiple frames

With only two frames (t and $t+\Delta$) to estimate a pivot point, we can do no better than this single estimate. However, since we assume the pivot to be static in $\underline{\underline{F}}_{B(.)}$, we can perform several such estimates over the video sequence and compute an average for $\underline{p}$ over all time. We have observed that large scale rotations produce a better estimate for $\underline{p}$ than small scale rotations. We therefore take a weighted mean of the estimates

for $\underline{p}$:

$$\frac{1}{N}\sum_{t=1}^{N}\omega(\theta(t))\underline{p}(t;\Delta) \tag{7.10}$$

Where N is the number of frame estimates, and $\omega(.)$ is a confidence weight for the pivot estimate between time t and $t+\Delta$. We model this as a smoothly varying distribution over θ — zero in the case of no rotation (or equivalently, a full rotation of $\theta = 2\pi$), and unity under maximum rotational shift ($\theta = \pi$).

$$\omega(\theta) = |\sin(2\theta)| \tag{7.11}$$

θ is obtained from the solution to equation 7.5 using arc-tangents.

Algorithm 2: Minimum of inter-frame motion field

Our second algorithm accepts two samples of the feature polygon at differing time instants $[t, t+\Delta]$ (where Δ is a user defined temporal interval). We obtain an affine transformation $\underline{\underline{M}}$ which maps polygon vertices at the former time instant to the latter:

$$\underline{\underline{M}} = \underline{\underline{A'}}_{t+\Delta}(\underline{\underline{A'}}_{t})^{+} \tag{7.12}$$

Now consider three non co-linear vertices of $\underline{\underline{A'}}_t$ at locations $\underline{\underline{X}} = [\underline{x}_1 \;\; \underline{x}_2 \;\; \underline{x}_3]$ transformed to locations $\underline{\underline{Y}} = \left[\underline{y}_1 \;\; \underline{y}_2 \;\; \underline{y}_3\right]$:

$$\underline{\underline{MX}} = \underline{\underline{Y}} \tag{7.13}$$

We define a static point $\underline{p}$ with reference to the feature, in barycentric coordinates as $\underline{\alpha}$:

$$\underline{p} \;\; = \;\; \underline{\underline{X}\alpha} \tag{7.14}$$

Applying transformation $\underline{\underline{M}}$ to the point $\underline{p}$:

$$\underline{\underline{Mp}} \;\; = \;\; \underline{\underline{MX}\alpha} \tag{7.15}$$

$$\underline{\underline{Mp}} \;\; = \;\; \underline{\underline{Y}\alpha} \tag{7.16}$$

we observe that $\underline{\alpha}$ remains constant relative to the feature reference frame, which has changed. The distance $\underline{d}(\underline{p})$ which point $\underline{p}$ moves, relative to reference frame $\underline{\underline{F}}_{B(.)}$ is:

$$\underline{d}(\underline{p}) = |\alpha_1(\underline{x}_1 - \underline{y}_1) + \alpha_2(\underline{x}_2 - \underline{y}_2) + \alpha_3(\underline{x}_3 - \underline{y}_3)| \tag{7.17}$$

Consider a field $d(.) \in \Re$ defined over all points. In cases of pure rotation, the minimum should be zero valued and coincident with the location of the static pivot i.e. $\underline{\underline{M}}\underline{p} = \underline{p}$:

$$\underline{\underline{M}}\underline{p} - \underline{p} = \underline{0} \quad (7.18)$$

$$\underline{\underline{M}}\underline{\underline{X}}\underline{\alpha} - \underline{\underline{X}}\underline{\alpha} = \underline{0} \quad (7.19)$$

$$(\underline{\underline{M}} - \underline{\underline{I}})\underline{\underline{X}}\underline{\alpha} = \underline{0} \quad (7.20)$$

Introducing the notation $\underline{\underline{V}} = (\underline{\underline{M}} - \underline{\underline{I}})\underline{\underline{X}}$, we can solve the following homogeneous linear system to locate the pivot point $\underline{\alpha}$ in barycentric coordinates:

$$\underline{\underline{V}}\underline{\alpha} = 0 \quad (7.21)$$

As with Algorithm 1, in noisy conditions angular velocity governs the accuracy of the estimate. In the case of no rotation, for example pure scale or translation, there will not be a single minimum. We therefore extend this algorithm to operate over multiple frames as with Algorithm 1; averaging pivot estimates over each frame and weighting these estimates according to the amount of rotary motion detected. Since a quantitative estimate for the amount of rotary motion is unavailable using this algorithm, we obtain an estimate of θ (equation 7.11) for this weighting using the algebraic technique of Algorithm 1.

7.2.3 Evidence Gathering, Geometric Solutions

Algorithm 3: Circle fitting method

Our third algorithm accepts three samples of the feature polygon at different time instants $[t, t + \frac{\Delta}{2}, t + \Delta]$ (where Δ is a user defined temporal interval), to produce an estimate of pivot point location.

If the motion of vertices $\underline{\underline{A}}'_t$ in frame $\underline{\underline{F}}_{B(t)}$ is approximately described by rotation about a fixed pivot $\underline{p}$, we may consider the trajectory of single vertex $\underline{a}'_t$ of that feature to follow a circular path. We sample the position of $\underline{a}'_t$ at three distinct time intervals, and fit a circle to interpolate those points; described by both a centre $\underline{p} = (i, j)^T$ and a radius r. The circle fitting method first computes the two chords of the circle, $\underline{H}_1 = \underline{a}'_{t+\Delta} - \underline{a}'_{t+\frac{\Delta}{2}}$ and $\underline{H}_2 = \underline{a}'_{t+\frac{\Delta}{2}} - \underline{a}'_t$. We then intersect the perpendicular bisectors of $\underline{H}_1$ and $\underline{H}_2$, to obtain $\underline{p}$. Radius r is obtained as the L_2 norm of the vector from $\underline{p}$ to either of the midpoints on $\underline{H}_1$ or $\underline{H}_2$. The centre $\underline{p}$ of the fitted circle is the pivot point. This process is repeated for each vertex $\underline{a}'_t$ in $\underline{\underline{A}}'_t$ and an average pivot point computed over all vertices.

As with the previous algorithms, this method extends to take into account multiple temporal samples to compute an improved estimate for the pivot location. However we found that averaging estimates for $\underline{p}$ over time did not give reliable results since outliers, which occur frequently due to noise, greatly skew the average. These outliers often exhibit very different radii from inlier estimates. Rather than throw away this useful indicator, we use the information to help combine estimates for $\underline{p}$ using a Hough-like accumulator approach.

By fitting multiple circles over different time intervals we accumulate "votes" for circle parameters. We define a 3D accumulator space, and for every vote a 3D Gaussian of constant standard deviation and mean is added to the space, centred at $[\underline{p}^T \quad r]^T$. An associated "confidence" weight $\omega(\theta)$ is assigned to each of these votes using an identical scheme to algorithms 1 and 2 (see equation 7.11). In this algorithm, we obtain θ from the inner product of the normalised perpendicular bisectors. The values of each vote's distribution, in accumulator space, are weighted by $\omega(\theta)$. Similar circle parameterisations accumulate votes to form heavily weighted distributions in local regions of space. Outliers are separated from this distribution by their poor estimates for r. After several temporal samples, the maximum vote in 3D space is deemed to represent the best fit circle parameters. The centre of this circle corresponds to the pivot point estimate. The use of Gaussians, rather than points, to cast votes in the accumulator space enables rapid vote accumulation without demanding a large number of temporal intervals to be sampled.

Algorithm 4: Linear accumulation method

Our fourth algorithm is an adaptation of algorithm 3, again requiring three temporal samples of the feature polygon (at time instants $[t, t + \frac{\Delta}{2}, t + \Delta]$). We reconstruct the two chords of the circle as before, and compute the two perpendicular bisectors. However we do not intersect the two bisectors to obtain a single pivot point centre and radius as with algorithm 3. Rather, we densely sample the distribution of points which lie both within the bounds of the video frame, and upon the infinitely long lines congruent with the two perpendicular bisectors. Each of these points is regarded as a potential pivot point, and cast into a 2D accumulator array. As with algorithm 3, the votes are weighted according to the magnitude of rotary motion using the functional of equation 7.11. Over multiple temporal samples, votes accumulate in this space creating maxima around the best estimates for pivot point location.

7.2.4 Summary of Algorithms

We have described four algorithms for pivot point recovery: the former two driven by the closed form solution of eigen-problems, and the latter two driven by iterative,

Hough-like evidence gathering procedures. All algorithms can be applied to estimate motion between two instants t and $t + \Delta$ (i.e. over a single interval). Under zero noise (i.e. synthetic) conditions all algorithms recover the pivot point location exactly. However the presence of such noise typically renders single interval estimates unusable, and as we will show, performance of each of the four algorithms degrades differently as noise increases.

To improve the accuracy of the estimate under noise, we have described how multiple temporal intervals may be examined simultaneously using each of the four algorithms. In the case of the eigen-problem solutions, we have described a method of combination for multiple frame pairs using a weighted average (where greater credence is attributed to pivot measurements resulting from larger rotary motion). This works well for algorithms 1 and 2, however the approach is impractical for algorithms 3 and 4. In the case of these evidence gathering approaches, the accumulator space is instead populated with multiple "votes" to create an improved estimate over multiple temporal intervals.

We now present the results of comparing the performance of each algorithm, using both real and synthetic data.

7.2.5 Comparison of Pivot Recovery Algorithms

For the purposes of algorithm comparison and evaluation we constructed a test rig (shown in Figure 7-12), and used the Computer Vision component of Chapter 6 to track the planar motion of the several coloured rectangles pivoting upon one another. This assembly was mounted upon a sliding base, which allowed the rig to translate. We filmed 800 frames of the rig being manipulated in a random fashion by a human operator; these form the *CONTRAPTION* sequence which we used as the sample of "real" data when evaluating the four algorithms. The physical pivot points in the sequence were also manually located via a point-and-click operation to provide a ground truth for *CONTRAPTION*, which we used to compare the performance of the algorithms.

Behavioural Predictions

All the algorithms described use samples of either two or three frames in the sequence (spanning a temporal interval $[t, t+\Delta]$), over which a single estimate of the pivot point location is derived. Recall that in cases of slight rotary motion, the estimated pivot is likely to be in error since very small movements of the polygons (perhaps, due to noise) will cause large shifts in the predicated pivot positions. By contrast, we predict large rotations should cause the estimated pivot location to be much more robust to noise. Therefore, our first prediction is that the estimation of the static pivot should improve in accuracy when we use larger temporal intervals (Δ), which are more likely

to span movements with large rotary components. Second, when the estimates from multiple temporal intervals are combined together, we predict the system will perform with greater accuracy. Third, we predict that increasing levels noise in the positioning of the tracked polygons will cause the estimated pivot location exhibit increasing levels of error.

Our three test variables are therefore temporal interval size (Δ), the number of temporal intervals to be combined in producing the estimate, and the level of measurement noise in the positions of polygon vertices.

Variation of temporal interval size (Δ)

We first applied each of the algorithms to a synthetic data set, in which a polygon from the *STAIRS* sequence was subjected to rotation within 70° over 100 frames. All four algorithms recovered the pivot point exactly when examining single temporal intervals (subject to some very small numerical error), regardless of the temporal interval size (Δ) used. Figure 7-3 gives a representative result.

We then assessed performance on real data, by applying each algorithm to features corresponding to the white and pink slabs within the *CONTRAPTION* sequence. We tested only a single temporal interval within this sequence, and examined the effect of varying the temporal interval size (Δ). For all algorithms, as we increased Δ, the estimated location of the pivot point tended toward the manually specified ground truth. Figure 7-4 (right) gives a representative plot showing algorithm two's performance on this sequence. An optimal value for Δ is video dependent, since such a choice is determined by the nature of motion within the temporal window spanned by $[t, t + \Delta]$. However the trend suggests that higher values of Δ produce improved results, and our experimentation on other slab combinations in the *CONTRAPTION* sequence led to similar conclusions. Our empirical evaluation suggests that a temporal interval size of 25 frames is suitable for accurate pivot point recovery on this sequence. Keeping the interval constant at 25, we again applied each of the algorithms to the *CONTRAPTION* sequence. The resulting error (Euclidean distance between the estimated and ground truth pivot locations) is shown in Figure 7-4, left. Algorithm 2 exhibited superior performance (i.e. minimum Euclidean distance between estimated and ground truth pivot points) relative to the other algorithms.

Combining multiple temporal intervals

The scattering of individual pivot estimates for real data clearly demonstrates the need for combining estimates from multiple temporal intervals (Figure 7-4, bottom). Individual estimations form an approximately Gaussian distribution of pivot points, the

mean of which closely approximates the ground truth location for the physical pivot. As the number of temporal intervals used to create a pivot estimate increases, a smaller error (Euclidean distance between the estimated and ground truth pivot points) was observed for all four algorithms. However the improvement was most notable for the evidence gathering algorithms (3 and 4). Algorithms 1 and 2 required fewer temporal intervals than algorithms 3 and 4 to produce reliable mean estimates of the pivot location. A likely explanation is that the vote accumulation approaches adopted by the latter two algorithms require a larger number of temporal samples to be recorded in the accumulator space before maxima begin to emerge above the level of noise.

Impact of tracker noise on performance

Although algorithm 2 exhibited superior accuracy for the *CONTRAPTION* real video sequence, this is by no means conclusive since different video sequences may exhibit differing degrees of noise (so affecting the accuracy of pivot point recovery). We therefore created a synthetic simulation of two articulated features — the positions of polygon vertices in this simulation were subjected to zero centred, additive Gaussian noise $G(0, \sigma)$. The performance of each algorithm was compared under increasing standard deviations σ of this additive noise. The temporal interval size (Δ) was held constant at 25 frames (1 second) during these experiments, and the number of temporal intervals tested was held constant at 74. Thus pivot point estimation was based on 100 frames (4 seconds) of data.

Figure 7-2 (left) shows how increasing noise causes an increase in error (measured as the Euclidean distance between measured and ground truth pivot locations) for all four algorithms. By inspection we make the qualitative observation that algorithm two exhibits superior robustness to such noise. We can quantify this robustness using Student's t-test (Figure 7-2, right). Student's t-test [152] is a standard method used to determine, to a specified level of significance, whether two distributions (in our case the ground truth and estimated pivot distributions) are significantly different. The process involves deriving a scalar "t-value" from the means and variances of both distributions. This "t-value" is compared with a corresponding value from a statistical table representing a particular confidence level. If the measured "t-value" is greater than the tabulated value, then the distributions are different.

In our experiments we increased the level of noise (σ) until we were "significantly sure" (95% certain) that the distributions differed. We entered the t-table at this level of certainty, and obtained a threshold t-value of 2.02. Thus the value of σ that caused the measured t-value to rise above 2.02 (marked as a black line in Figure 7-2), corresponds to the level of noise at which the algorithm became unreliable. The results confirm our

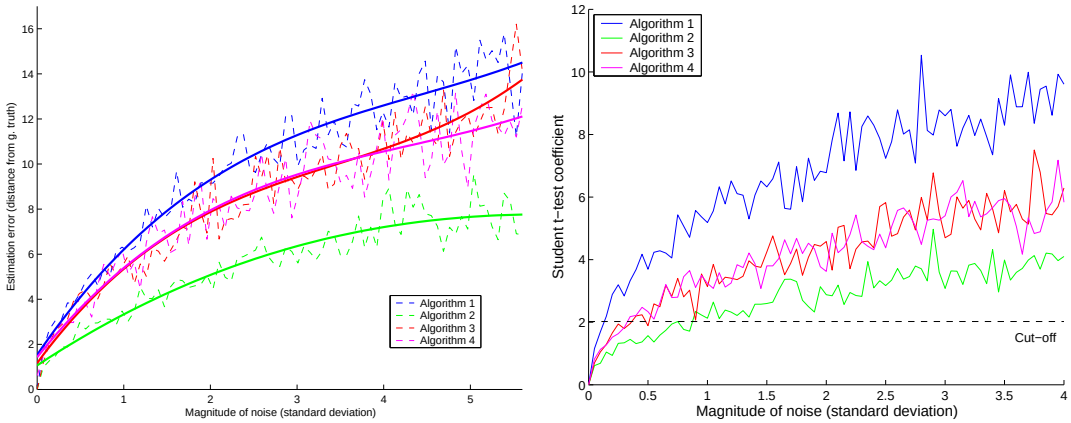

Figure 7-2 Comparing the performance of the four pivot point location algorithms using simulated data. The polygon vertices were subjected to zero mean Gaussian noise of increasing standard deviation, resulting in increasing levels of estimation error when recovering the pivot point — error measured as the Euclidean distance between estimated and ground truth pivot (left). The trend is plotted as a solid line, measurements as a dotted line. The test ran over one hundred simulated frames, sampling frame pairs at time intervals of 25 frames (1 second). Student's t-test (right) was employed to decide the level of noise at which the distribution of estimated pivots differed from the ground truth significantly (i.e. with 95% confidence). This threshold value (derived from statistical tables) is plotted in black. Both graphs show algorithm 2 to out-perform the others, the latter graph demonstrates tolerance of up to one standard deviation of noise.

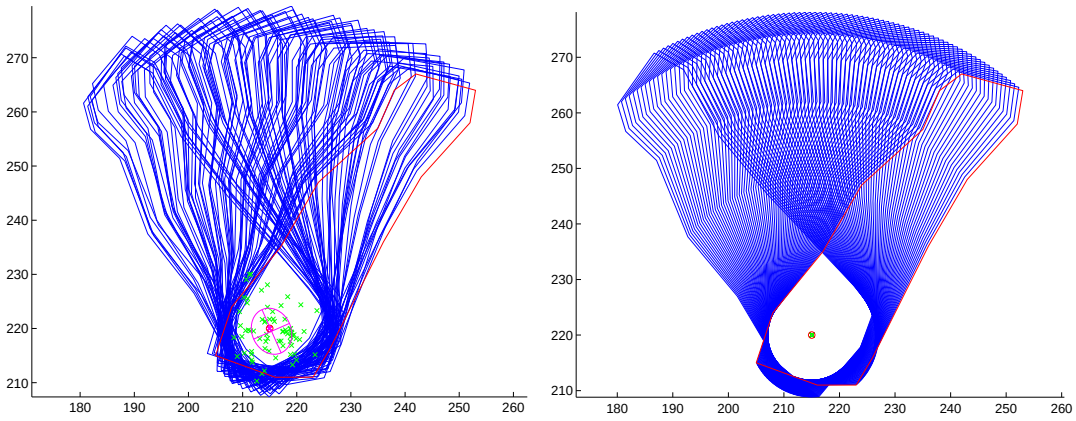

Figure 7-3 Examples of pivot point recovery using algorithm two, forming part of the synthetic test case used to generate Figure 7-2. A limb (feature E, original position in red — see Figure 6-2) from the *STAIRS* sequence was subjected to synthetic rotation within 70° for 4 seconds (100 frames). Left: polygon vertices perturbed by two standard deviations of zero centred Gaussian noise, produces a small cluster of potential pivots the mean of which closely corresponds to the ground truth (two standard deviations plotted in magenta). Right: in the zero noise case the pivot is recovered exactly — this is true for all four algorithms.

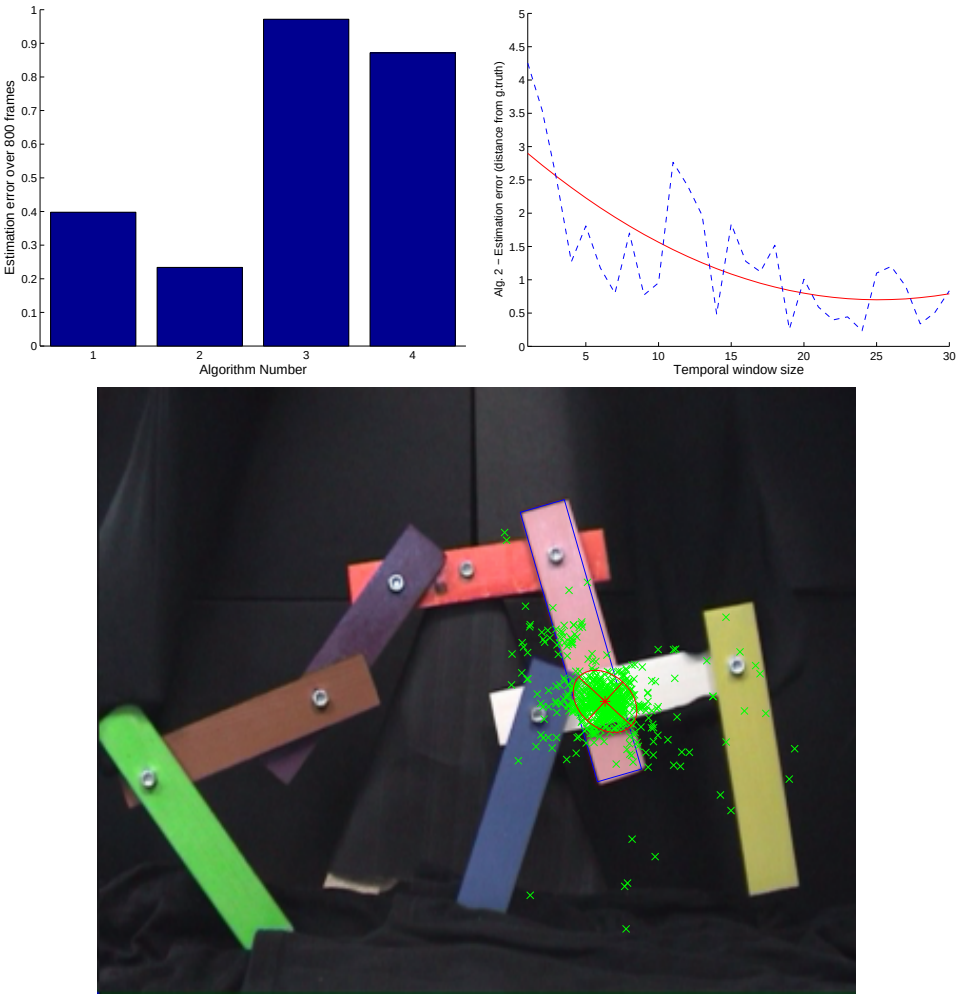

Figure 7-4 Applying the algorithms to 800 frames of the *CONTRAPTION* sequence, specifically to determine the pivot point between the pink and white slabs. Bottom: The estimated pivot points due to algorithm 2, between multiple frame pairs sampled at an interval of $\Delta = 25$; the error distribution of estimated pivots about the ground truth approximates a zero mean Gaussian. The estimated (weighted) mean pivot location is very close to the ground truth (red star). Top left: per algorithm comparison of errors in pivot position after 800 frames. Top right: Showing that pivot estimation error declines as the sampling distance between frames in the frame pair increases (algorithm 2).

earlier observations; algorithm two exhibits superior robustness to noise — differing from the ground truth significantly at around $\sigma = 1$. The values for σ at which the other algorithms became unreliable were, at best, a third of this value.

7.3 Recovering Hierarchical Articulated Structure and Pose

Algorithm 2 appears to estimate pivot points with superior accuracy and robustness than the other proposed algorithms. We therefore applied this algorithm to recover one pivot point for each ordered pair of tracked features in the video (denoting a sin-

gle ordered pair as $\{A, B\}$). Observe that due to noise in the estimation process, the computed pivot of A about B, and B about A, may not be identical; thus the pivots resulting from pairing $\{A, B\} \neq \{B, A\}$. Computing a pivot for each ordered pair of features requires $2C_2^n$ ordered pairings for n tracked features; this is not a large number for typical sequences (for example, 132 for the *STAIRS* sequence), and this computational load can be further reduced by only testing feature pairs whose polygons intersect for a significant proportion of their lifetime.

Many of the computed pivots do not correspond to physically existing articulations between feature pairs, and should be discarded. We decide which to discard by evaluating a "quality" metric for each estimated pivot, and discarding pivots that score poorly. This filtering process leaves us only with true pivot points, and so with a final articulated structure for the tracked subject. We can may then fit the original tracked polygons to the derived articulated structure, so generating a pose vector for each frame.

Quality Metric

If two features A and B are articulated, then in cases where we can observe those features with no measurement noise, the following properties hold:

1. The world coordinate position of the pivot between features A and B should, at all times, lie within the area of intersection of both features A and B.

2. The motion of feature A relative to feature B should be well described by a rotation of A in the reference frames of B, about a static pivot.

3. The motion of feature B relative to feature A should, similarly, be well described by a rotation of B in the reference frame of A, about a static pivot.

4. The world coordinate position of the pivot point of A on B and the pivot point of B on A should at all times be coincident.

Our assumption is that if these properties are violated, then features are not articulated. Of course estimation errors may cause violation of these properties, and so they are better viewed as heuristics — compliance with these heuristics indicates a "good quality" estimated pivot point, and so a likely articulation between A and B. We construct the quality function for a potentially articulated feature pair $\{A, B\}$ as follows.

For a well estimated pivot point $\underline{p}$ in $\underline{\underline{F}}_{B(t)}$, the motion of $\underline{\underline{A}}'_t$ over time should be well described by a rotation $\underline{\underline{R}}_{\theta(t)}$ about $\underline{p}$; thus we desire a small residual r_1 averaged over

all n frames:

$$r_1 = \frac{1}{n}\sum_{t=2}^{n}\left|\underline{\underline{A}}'_{t-1} - \underline{\underline{R}}_{\theta(t)}(\underline{\underline{A}}'_t - \underline{p}\underline{1}^T) + \underline{p}\underline{1}^T\right| \tag{7.22}$$

This r_1 forms the first term in the quality function.

Now consider that the world coordinates of the pivot of A with respect to B at time t (which we write as $\underline{p}_{A(t)}$) and the pivot of B with respect to A (which we write as $\underline{p}_{B(t)}$) should be coincident under zero noise conditions. Thus we desire a small residual r_2 averaged over all n frames:

$$r_2 = \frac{1}{n}\sum_{t=1}^{n}\left|\underline{p}_{A(t)} - \underline{p}_{B(t)}\right| \tag{7.23}$$

The position of the pivot $\underline{p}_t$ between features A and B at time t is computed as $\underline{p}_t = \frac{1}{2}(\underline{p}_{A(t)} + \underline{p}_{B(t)})$.

Finally, we add a penalty term for the distance that the pivot lies outside the area of intersection of A and B; specifically the Euclidean distance between $\underline{p}_t$ and the closest pixel within the intersection area. This distance is averaged over time to obtain penalty term Φ. We write the complete quality function $Q[A, B]$ as a sum of weighted terms:

$$Q[A, B] = \exp(-k(r_1 + r_2 + \Phi)) \tag{7.24}$$

where k controls the decay of a pivot point's viability as this sum of error terms increases, we find a value of around $k = 0.1$ suitable for our source data.

Note that our system does not yet cater for cases where features are rigidly joined, with little or no rotary motion present. Currently such features are classified as non-articulated, and we assume for the time being that such groups of features have been segmented as a single rigid component.

Recovering Pose Vectors

The algorithms described thus far are capable of recovering general articulated motion of features in the plane. For ease of representation and later manipulation, we assumed that such features form a hierarchical structure. Although this is a specialisation of the system, most common subject matter, for example people, are admitted under this model. It is a simple matter to detect the presence of cycles in the recovered articulated structure, and currently we return an error in such situations, specifying that time and pose cues can not be applied to that particular object.

Like most hierarchical pose representations, we specify the position of a node, say an arm, in relation to a component higher in the hierarchy, which in turn may be specified in relation to a component higher still in the hierarchy, and so on recursively to the root node. This can produce positional inaccuracies in the leaf nodes due to a build up of multiplicative errors as we descend the hierarchy. We therefore carefully choose a feature to serve as a root node, such that this choice minimises the longest hop count from the root node to a leaf node. In the case of the *STAIRS* sequence, the choice of root feature is the torso. In the *CONTRAPTION* sequence, the root is the red slab.

We wish to construct a numerical representation of the subject's pose at each time instant t, by searching for a best fit of the recovered articulated structure to the set of tracked features. Recall that each feature was tracked independently in Section 6.3.2 — the resulting tracked polygons each represent the "best" estimate obtainable for a feature's position in the raw footage, given that no global model or constraints were available at the time. By combining the raw feature position estimates with the recovered articulated model, we not only produce a numerical "pose vector" for each time instant, but also impose physical constraints to further refine the accuracy of tracked polygons. We form an initial estimate for this "pose vector", then search for an "optimal" pose local to this using a Nelder-Mead search [114]. For our purposes, the optimal pose is the configuration which minimises the Euclidean distance between the tracked feature polygon vertices, and the vertices generated by re-synthesising the feature polygon positions from the putative "optimal" pose.

The structure and initial estimate of the pose vector is formed as follows. The first four elements of the vector are a representation of the four LCAT parameters which transform the root feature from its initial position (in the first frame), to its current position (in frame t). This is extracted directly from the feature tracker. The translational component of the LCAT is converted from Cartesian to polar form, thus the first four elements of the pose vector are $[\phi,\ r,\ \theta,\ s]$; where ϕ is the direction of translation, r is the translation magnitude, θ is the orientation of the root object and s is a uniform scaling.

$$\underline{V}(t) = \left[\begin{array}{ccccccccc} \phi & r & \theta & s & \theta_1 & \theta_2 & ... & \theta_n \end{array} \right]^T \tag{7.25}$$

Features in the structure are visited via a pre-order traversal of the hierarchy. Each subsequent entry in the pose vector specifies the angle at which the newly visited feature is orientated, relative to its parent feature (i.e. about the parent-child pivot point). This angle is relative to the child's position in the first frame; at time $t = 1$ all such angles will be zero — the angle in the pose vector encodes the change in orientation between frame 1 and frame t. In our initial estimate, this angle is extracted from the

LCAT between the child's position relative to the parent in frame 1 and frame t. In equation 7.25 each of these angles is denoted θ_i where $i = [1, n]$, n being the number of features in the hierarchy, and i being an index into the ordering in which the feature hierarchy is traversed.

7.4 Temporal Re-sampling

The process of Section 7.2 results in a series of vectors $\underline{V}(t)$ for each frame t. These vectors form individual points in a high dimensional space, representing the subject's pose at a given instant. The trajectory of these points encode the pose of the subject moving over time. Manipulating this pose space gives rise to novel time and pose cues which may be used to emphasise motion in our system. We may choose to manipulate only a local region of one dimension of this space; affecting the position of one joint over a small temporal window. We term these "local" pose transformations, and show in the next subsection that cartoon "anticipation" effects can be created by this class of transformation. We may also choose to scale or deform one or more dimension of this pose space globally, i.e. over all time. This has the effect of modifying the basis of the pose space. We refer to these as "global" pose transformations, and discuss these in subsection 7.4.2.

7.4.1 Temporally Local transformation (anticipation)

Anticipation is an animation technique applied to objects as they begin to move; the technique is to create a brief motion in the opposite direction, which serves to emphasise the subsequent large scale movement of an object (Figure 7-5). The anticipation cue communicates to the audience what is *about to* happen. Anticipation acts upon a subject locally — only within a temporal window surrounding the beginning of the movement to be emphasised, and only upon the feature performing that movement.

We have implemented anticipation as a 1D signal filtering process. Each individual, time varying component of the pose vector $\underline{V}(.)$ (for example, the angle a metronome beater makes with its base) is fed through an "anticipation filter", which outputs an "anticipated" version of that pose signal (Figure 7-5). The filter also accepts six user parameters which control the behaviour of the anticipation motion cue. The filtering process operates in two stages. First, the 1D signal is scanned to identify the temporal windows over which anticipation should be applied. Second, the effect is applied to each of these windows independently.

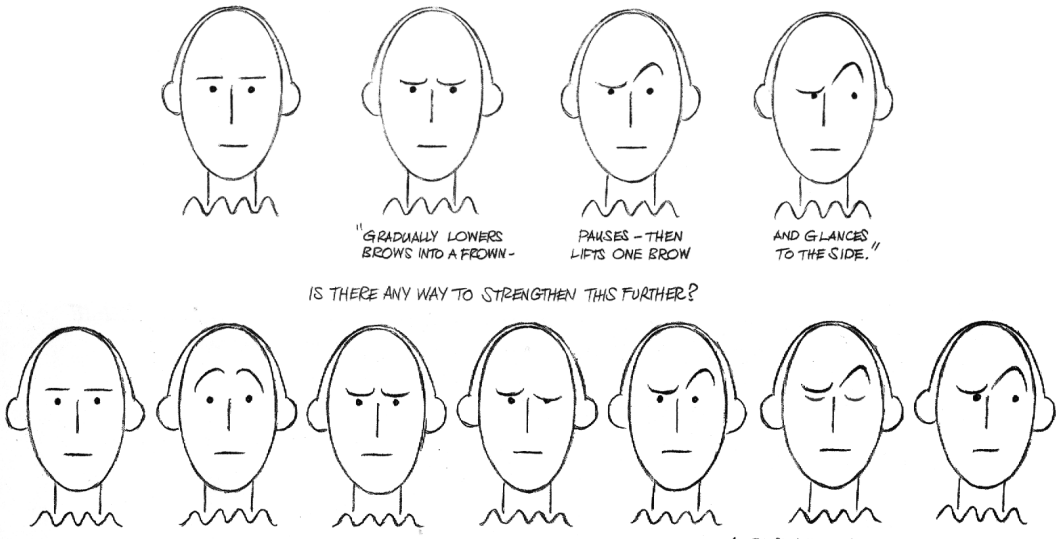

Figure 7-5 Illustrating how animators can apply anticipation to emphasise motion, in this case a Disney-style facial animation (reproduced from [169]).

Identifying Temporal Windows for Anticipation

Given a 1D input signal, the filter first identifies temporal windows for application of anticipation. These are characterised by the presence of high acceleration magnitudes (above a certain threshold), which exist for a significant number of consecutive frames (a further threshold) in the signal. These two thresholds form part of the set of user parameters that control the effect. This process allows us to identify a set of temporal windows corresponding to large motion changes, which an animator would typically emphasise using anticipation. A high acceleration magnitude may or may not generate a change of direction in the signal and, after numerous conversations with animators [130], we have determined that the manifestation of the anticipation cue differs slightly between these two cases:

Case 1. First, consider the case where acceleration causes a change of direction in the 1D signal; for example, a pendulum at the turning point of its swing. Regardless of the acceleration magnitude of the pendulum beater (which may rise, remain constant, or even fall during such a transition), the anticipation effect is localised to the instant at which the beater changes direction i.e. the turning point of the signal; the *minimum of the magnitude of the first derivative with respect to time*. In the case of the *METRONOME* sequence (Figure 7-15), a brief swinging motion would be made, say to the left, just prior to the recoil of the metronome beater to the right. The object then gradually "catches up" with the spatiotemporal position of the original, un-anticipated object at a later instant.

Case 2. Now consider the second case where acceleration does not cause change of direction in the 1D signal; for example, a projectile already in motion, which acquires

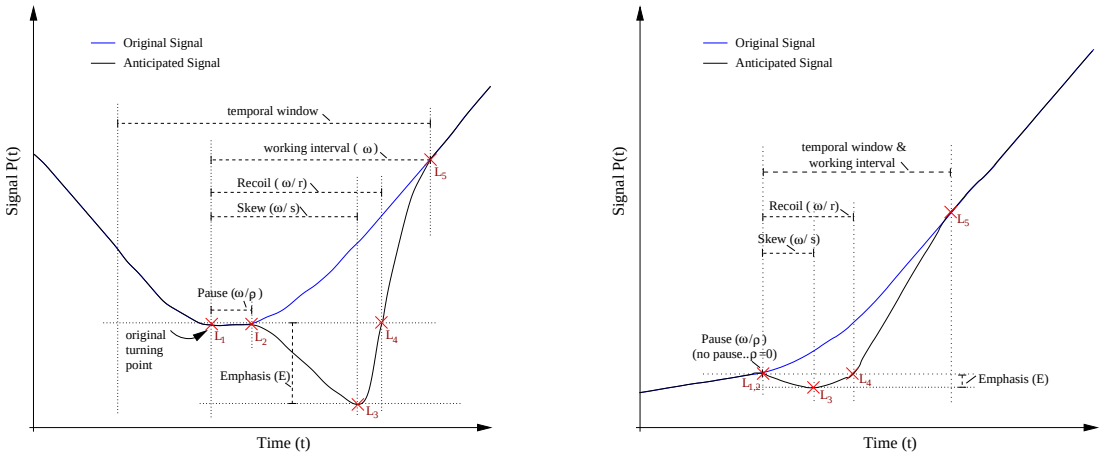

Figure 7-6 Schematic examples of the anticipation filter under case one (signal direction of motion changes) and case two (signal direction of motion unaffected). Case two has been illustrated with pause parameter $\rho = 0$. Section 7.4.1 contains an explanation of the user parameters ρ, s, r, and ϵ which influence the behaviour of the effect.

a sudden burst (or decrease) in thrust, i.e. a change in acceleration magnitude. The anticipation effect is manifested as a short lag just prior to this sudden acceleration change; i.e. at the *maximum in the magnitude of the third derivative with respect to time*. As with case 2, the projectile swiftly accelerates after anticipation to catch up with the spatiotemporal position of the original, unaffected projectile. Interestingly a projectile moving from rest is equally well modelled by either the first or second case, since the locations of zero speed (minimum first derivative) and maximum acceleration change (maximum third derivative) are coincident.

Synthesising Anticipation within a Temporal Window

Each temporal window identified for application of the anticipation cue is processed independently, and we now consider manipulation of one such a window. The first task of the "anticipation filter" is to scan the pose signal to determine whether a change of direction occurs within the duration of the temporal window. This test determines which criterion from the respective case (1 or 2) is used to determine the instant at which anticipated motion should be "inserted" into the sequence; we denote this time instant by τ. We define a temporal "working interval" as the time window within which we pose is varied from the original signal, in order to introduce anticipation. This working interval extends from time τ to the end of the temporal window, which we write as $\tau + \omega$. In all cases the direction of the anticipatory motion will be in opposition to the direction in which acceleration acts. We refer the reader to Figure 7-6 to assist in the explanation of the subsequent signal manipulation.

We create the anticipation effect by modifying the 1D pose signal to follow a new curve, interpolating five landmark points in space $[t, P(t)] \in \Re^2$, where $P(t)$ indicates the value of the pose signal at time t. Aside from the two parameters used to control activation of the effect, there are four user parameters ρ, s, r, and ϵ (where $\rho \leq s \leq r$). These influence the location of the five landmark points $[\underline{L}_{1\ldots5}]$, which in turn influences the behaviour of the anticipation. We now explain the positioning of each of the five landmarks and the effect the user parameters have on this process. Throughout, we use notation $p(t)$ to indicate the original (unanticipated) pose signal at time t, and $p'(t)$ to denote the new, anticipated signal.

$\underline{L}_1$. The first landmark marks the point at which the original and anticipated pose signals become dissimilar, and so $\underline{L}_1 = (\tau, p(\tau))^T$. Recall τ is the determined by the algorithm of either case 1 or 2, as described in the previous subsection.

$\underline{L}_2$. At the instant τ, a short pause may be introduced which "freezes" the pose. The duration of this pause is a fraction of the "working interval" — specifically ω/ρ frames, where ρ is a user parameter. The second landmark directly follows this pause, and so $\underline{L}_2 = (\tau + \omega/\rho, p(\tau))^T$.

$\underline{L}_3$. Following the pause, the pose is sharply adjusted in the direction opposite to acceleration, to "anticipate" the impending motion. The magnitude (E), and so the emphasis of, this anticipatory action is proportional to the magnitude of acceleration: $E = \epsilon|\ddot{p(\tau)}|$. Here ϵ is a user parameter (a constant of proportionality) which influences the magnitude of the effect. A further user parameter, s, specifies the instant at which the anticipation is "released" to allow the movement to spring back in its original direction. We term s the "skew" parameter, since can be used to skew the timing of the anticipation to produce a long draw back and quick release, or a sharp draw back and slow release. Referring back to Williams' guidelines for anticipation, one would typically desire the former effect ($s > 0.5$), however our framework allows the animator to explore alternatives. The third landmark is thus located at the release point of this anticipated signal, and so $\underline{L}_3 = (s, E)^T$.

$\underline{L}_4$. The rate at which the feature springs back to "catch up" with the unanticipated motion is governed by the gradient between the third and fifth landmarks. This can be controlled by forcing the curve through a fourth landmark $\underline{L}_4 = (\tau + \omega/r, p(\tau))^T$.

$\underline{L}_5$. Finally the point at which the anticipated and original pose signals coincide is specified by the final landmark, $\underline{L}_5 = (\tau + \omega, p(\tau + \omega))^T$.

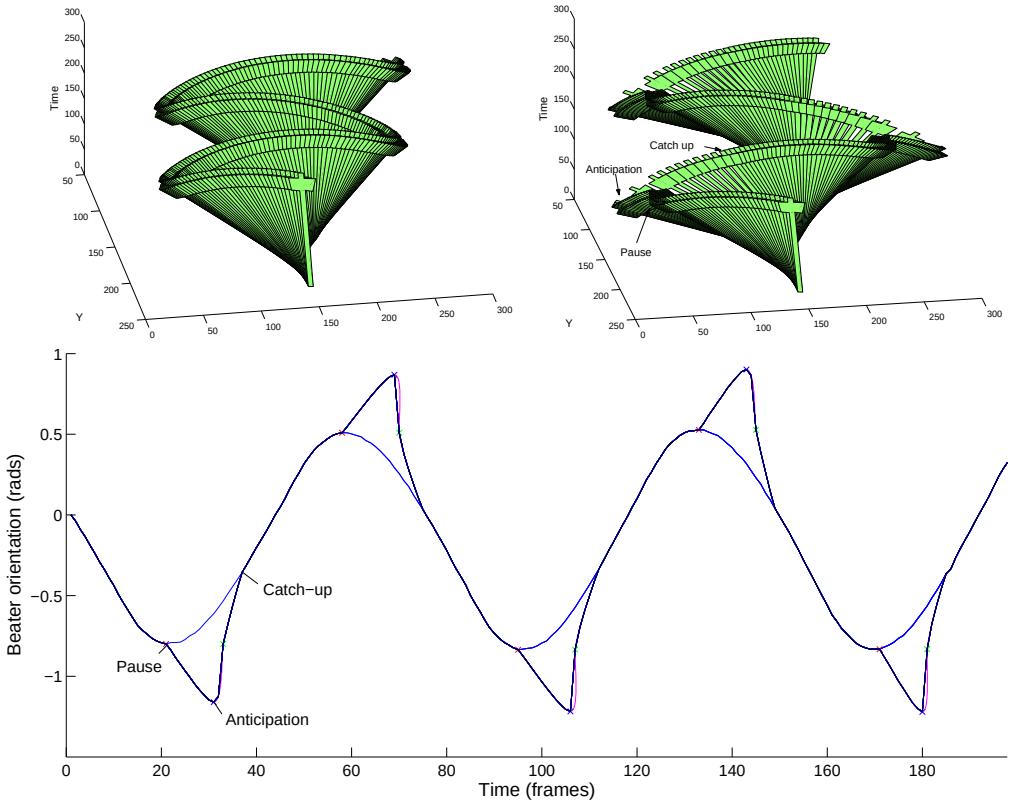

Figure 7-7 Top: Time lapse representation of the beater in the *METRONOME* sequence, before (left) and after (right) application of the anticipation filter to the pose vectors. Bottom: Visualisation of the 5^{th} element of the *METRONOME* pose vector (encoding the angle between metronome beater and body), before (blue) and after (black) passing through the anticipation filter with $\rho = 0$, $\epsilon = 150$, $s = 0.8$, $r = 0.9$. Green and red vectors indicate the gradients at $\underline{L}_4$ and $\underline{L}_5$ used to interpolate those two control points.

The anticipated signal $p'(t)$ at any time t within the temporal window is created by interpolating these five landmarks, the following manner. In the first stage of the anticipation, landmarks $\underline{L}_1$ and $\underline{L}_2$ are linearly interpolated to create a simple pause in the signal. The pause component of the anticipation is created using the parametric line $\underline{\pi}_1(c)$ where $c = [0, 1]$:

$$\underline{\pi}_1(c) = \underline{L}_1 + c(\underline{L}_2 - \underline{L}_1) \tag{7.26}$$

The second stage of the anticipation is the movement in the opposite direction to the impending motion. We interpolate landmarks $\underline{L}_2$, $\underline{L}_3$ and $\underline{L}_4$ using a cubic Catmull-Rom spline [51], creating a smooth transition between the pause stage and the anticipatory

action. Using notation $\underline{\pi_2}(c)$ we have:

$$\underline{\pi_2}(c) = \begin{bmatrix} \underline{L}_2 & \underline{L}_2 & \underline{L}_3 & \underline{L}_4 \end{bmatrix} \begin{bmatrix} -0.5 & 1 & -0.5 & 0 \\ 1.5 & -2.5 & 0 & 1 \\ -1.5 & 2 & 0.5 & 0 \\ 1 & -0.5 & 0 & 0 \end{bmatrix} \begin{bmatrix} c^3 \\ c^2 \\ c \\ 1 \end{bmatrix} \tag{7.27}$$

where the 4×4 matrix term is the Catmull-Rom cubic blending matrix. Finally, we interpolate between landmarks $\underline{L}_4$ and $\underline{L}_5$ using a cubic Hermite spline [51]. This family of splines ensures C_1 continuity at both landmarks, so blending the anticipated signal smoothly with the original signal. The Hermite curve requires specification of position and velocity at both landmarks. Again, using c as a dummy parameter, this segment of the anticipation is described by $\underline{\pi_3}(c)$:

$$\underline{\pi_3}(c) = \begin{bmatrix} \underline{L}_4 & \underline{L}_5 & \dot{\underline{L}}_4 & \dot{\underline{L}}_5 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c^3 \\ c^2 \\ c \\ 1 \end{bmatrix} \tag{7.28}$$

where the 4×4 matrix term is the Hermite cubic blending matrix. We can obtain the velocity at $\underline{L}_5$ using a simple finite difference approach on the original, discrete signal at $p(\tau + \omega)$. The velocity at $\underline{L}_4$ is obtained from the partial derivative of equation 7.26 with respect to c.

Figure 7-6 gives two schematic examples of signals undergoing anticipation in cases 1 and 2. Figure 7-7 (bottom) shows the original and anticipated signals used for rendering the *METRONOME* sequence, a time lapse representation of which is given in Figure 7-7 (top). The reader is referred to Appendix C for this and other rendered video clips.

7.4.2 Temporally Global transformation (motion exaggeration)

A portrait caricaturist will emphasise unusual or characteristic features of a subject's face. Likewise, cartoonists will emphasise unusual motion characteristics, for example a limp, exhibited by their subjects [98]. Such characteristics may be considered to be outliers in the cartoonist's mental model, for example, of people; the more an individual's characteristics diverge from the "norm", the more those characteristics tend to be emphasised.

Although our system is not equipped with a model of the population, we can learn the pattern of simple repetitive motions made by a tracked subject, and exaggerate variations in this pattern. Periodic motion, such as human gait, causes pose vectors to trace

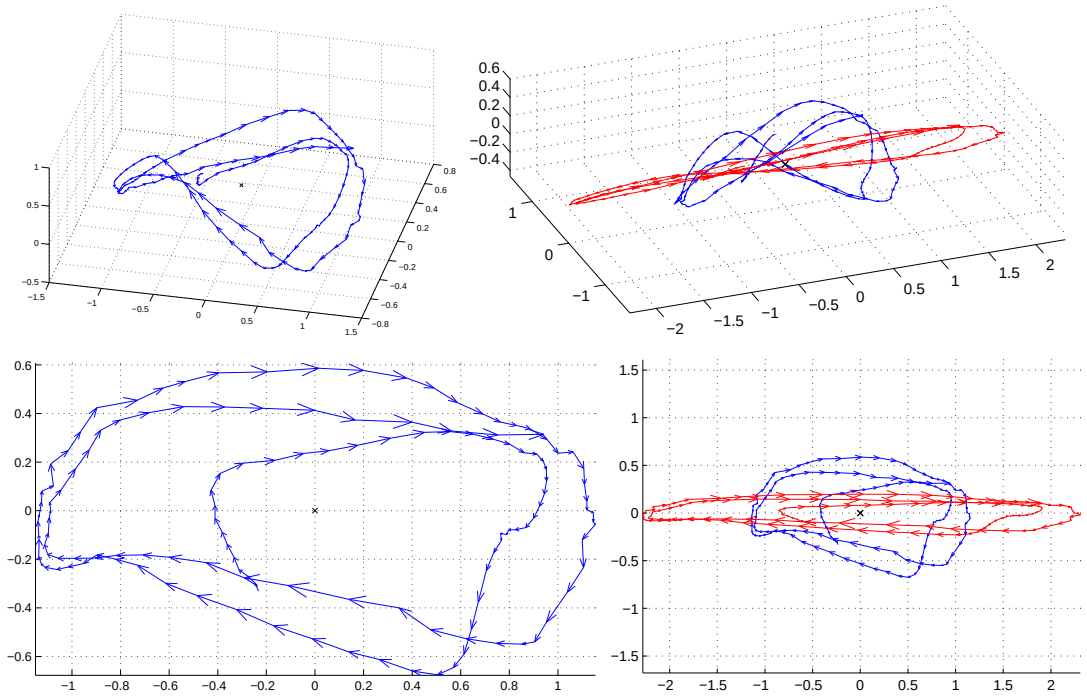

Figure 7-8 Visualisation of the pose space for *STAIRS* before (blue) and after (red) motion exaggeration with $\mathcal{F} = 2$. Graphs produced by projecting the high dimensional space down into first 3 (above) and 2 (below) principal axes, centring the mean upon the origin. Arrowheads indicate the direction of time.

a cyclic trajectory within a subspace of the pose space (consisting of all dimensions minus the first two, which represent translation of the root feature). If several cycles of motion exist within the video sequence (for example, *STAIRS*), then it is possible to reliably compute a mean point in the pose space. Performing a global scaling transformation on the space, with centre of projection at the mean, serves as a basis for exaggerating variations in the motion over time.

This simple approach not only emphasises variations, for example in gait, but also any noise introduced by the tracking process. In our motion cartooning application we desire exaggeration of only the important, principal, motions leaving noise components unchanged. Our strategy is to perform a principal component analysis (PCA) of the pose samples taken over duration of the video sequence, to isolate the sub-space within which the motion principally varies. We produce an eigenmodel of all pose samples $\underline{V}(t)$ over time (yielding a mean pose $\underline{V}_\mu$, a collection of column eigenvectors $\underline{\underline{U}}$ and a diagonal matrix of eigenvalues $\underline{\underline{\Lambda}}$). By scaling the basis specified in each eigenvector by a multiplicative factor $\mathcal{F}$ of its corresponding eigenvalue, we produce a novel set of pose vectors $\underline{V'}(t)$ in which motion appears to be exaggerated.

$$\underline{V'}(t) = \underline{\underline{U}}^{+} \mathcal{F} \underline{\underline{\Lambda U}} (\underline{V}(t) - \underline{V}_\mu) + \underline{V}_\mu \tag{7.29}$$

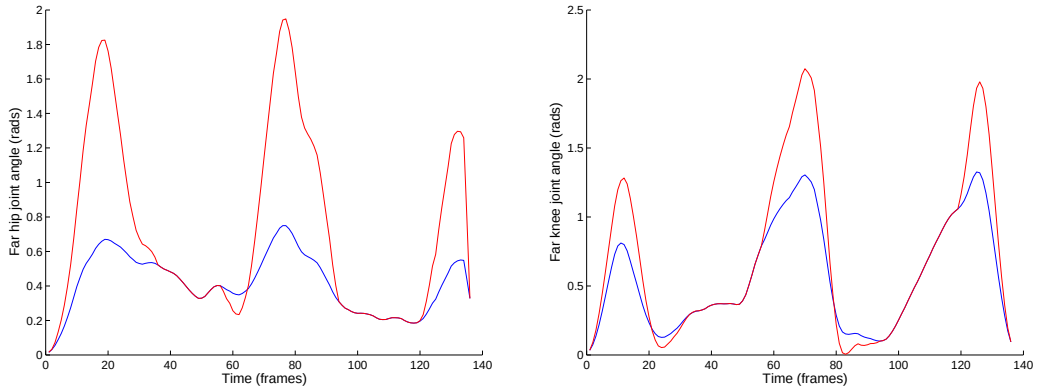

Figure 7-9 Visualisation of signal within two dimensions of the pose space for *STAIRS*, corresponding to orientation of the far hip joint (left) and of the far knee joint (right). Observe the periods of invariance between the original signal (blue) and exaggerated signal (red), generated by the constraints imposed by the animator.

Figure 7-15, sequence A superimposes the modified positions of feature polygons when $\mathcal{F} = 2$ over the original video.

Introducing Physical Constraints

While sequence A in Figure 7-15 demonstrates exaggerated motion, the manipulation of the pose does not yet take into account physical constraints upon the articulated structure. For example, the feet appear to glide over the steps, and do not appear to interact with the ground surface in a naturally plausible manner. Constraints to counter this behaviour must be interactively specified by the animator; we allow users to select temporal windows over which certain features (for example the feet), are to remain coincident with the original sequence. Although a complete solution to the problem of motion exaggeration under constraints lies within the realms of inverse kinematics, we have found that our combination of localised input from the animator with the global transformation of the motion exaggeration, to produce aesthetically acceptable output.

To simplify matters we state that if, at a given instant, a feature is marked to be held in its original position, then all features between that feature and the root feature must also to match their original position. Thus we can derive a series of temporal windows for each feature during which their motion should not be emphasised.

We encode these temporal windows in a matrix of weights, which we will write as $\underline{\underline{\omega}}$. This matrix is formed as a concatenation of column vectors $\underline{\omega}(t)$, each corresponding to the pose vector $\underline{V}(t)$ at time $t \in [1, n]$.

$$\underline{\underline{\omega}} = \left[\begin{array}{cccc} \underline{\omega}(1) & \underline{\omega}(2) & \ldots & \underline{\omega}(n) \end{array} \right] \tag{7.30}$$

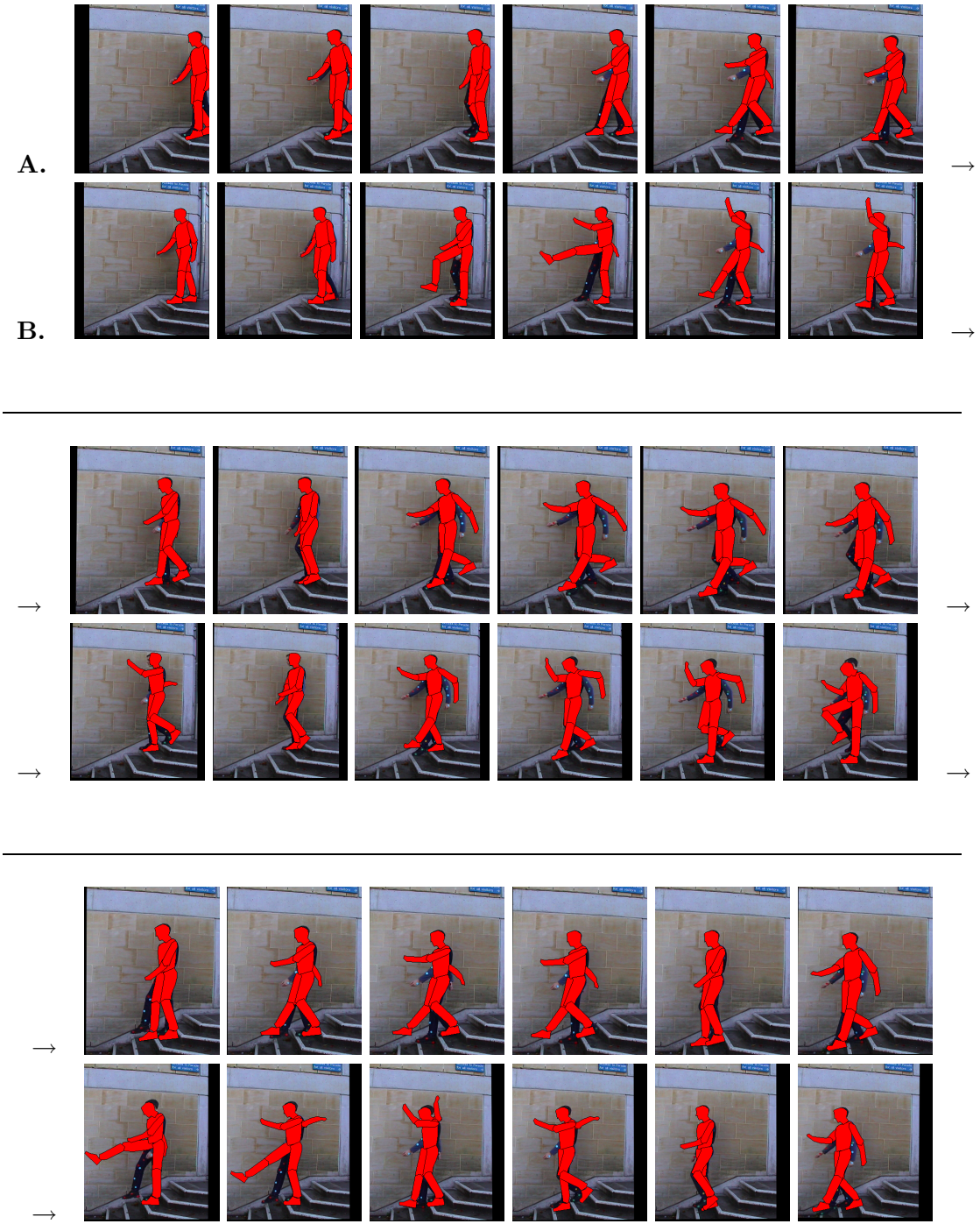

Figure 7-10 Stills from the source *STAIRS* sequence, with the positions of the exaggerated feature polygons overlaid. Motion has been exaggerated using physical constraints in sequence B, and without constraints in sequence A. Observe the improved placement of the feet in B. Approximately one still per five frames (see animations `videos/stairs_exaggerate` and `videos/stairs_exaggerate_polys`).

Figure 7-11 Four stills from the `videos/stairs_exaggerate` animation in Appendix C. We have introduced motion exaggeration into the *STAIRS* sequence, and applied moustache and pith helmet (using the rotoscoping features of the Video Paintbox, Section 8.4.3) to produce an animation reminiscent of Monty Python's Flying Circus [BBC 1969–74].

If, at particular instant t, we desire the i$^{\text{th}}$ component of the pose to follow its original rather than exaggerated trajectory, we set the respective matrix element $\omega_{t,i}$ to be zero. Consecutive zero valued elements within a particular row of $\underline{\underline{\omega}}$ therefore correspond to temporal windows during which the motion of a particular pose component should not to be exaggerated. We iterate through each row of $\underline{\underline{\omega}}$ in turn. Elements in the row are assigned values in proportion to their distance from the nearest zero valued element on that row. These values are normalised to rise to unity halfway between two given temporal windows. Each matrix element $\omega_{t,i}$ now contains a scalar weight, representing the degree of exaggeration to apply to each pose component i at each time t. To produce our exaggerated, yet physically constrained pose $\underline{V}''(t)$ at time t we linearly interpolate between the original pose ($\underline{V}$(t)) and the unconstrained, exaggerated pose ($\underline{V}$'(t), see equation 7.29) using:

$$\underline{V}''(t) = \underline{V}(t) + \underline{\omega}(t)(\underline{V}'(t) - \underline{V}(t)) \tag{7.31}$$

Figure 7-10 (sequence B) gives an example of the resulting animated sequence using the *STAIRS* data. As with sequence A, feature polygons have been superimposed on the original video to compare the two sequences. The animator has interactively specified various temporal windows within which the feet should coincide with the original data (whilst they are on the ground), and also specified that all features should start and end coinciding with the original data.

7.5 Video Re-synthesis

Once the pose vectors have been manipulated to introduce motion cues, it remains to re-synthesise the video sequence using the novel pose data. This involves painting each feature polygon with a suitable texture sampled from the video (and then composit-

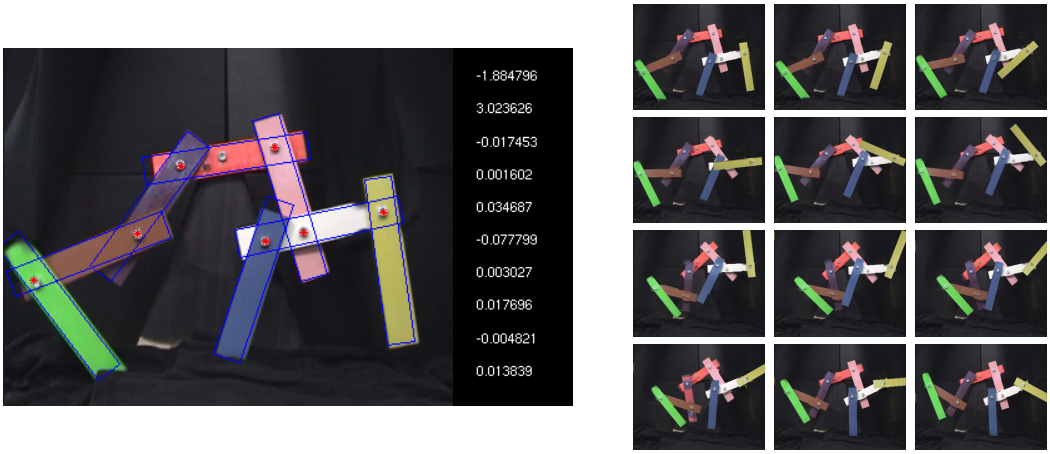

Figure 7-12 Pose recovery and manipulation. Left: A frame from the *CONTRAPTION* sequence, with pivot points and articulated structure recovered automatically. The recovered pose vector is shown inset. Right: Overwriting the last dimension (angle between white and yellow slabs) of the first 50 frames with values from 0 to 2π in equal increments.

ing the polygons in the correct depth order, in accordance with the depth information recovered in Section 6.3.3). This re-texturing may introduce difficulties, since the modified pose may expose regions of features that were occluded in the source video.

Our solution to sampling texture is similar to that used in the occlusion buffer of Section 6.4.3. We sample as much unoccluded texture as possible at a time instant t, and then fill in remaining "holes" in the texture by sampling neighbouring frames in alternation, i.e. $t-1$, $t+1$, $t-2$, $t+2$, and so on. This gives a reasonable reconstruction of the feature texture local to time t, so allowing for some variation in the reconstructed texture due to temporally local illumination changes in the video. If any holes remain in the texture they are filled using the mean RGB colour of the recovered texture.

All that remains is to decide the instant t from which to begin sampling texture. In the occlusion buffer of Section 6.4.3, the feature to be re-textured invariably occupied the same spatiotemporal position as in the original video. Thus if we wished to re-texture an object in frame i, we began sampling texture at time $t = i$. In the case of time and pose cues, this is not necessarily true; the spatiotemporal position of a feature may differ considerably from its position in the original video, due to our pose manipulations. We therefore begin sampling texture from an instant t in the original video where the feature is in approximately the same position as the feature in the manipulated pose, to take into account local lighting changes. Furthermore, there may be many such instants in the original video and we should sample from the instant closest in time to the current frame being rendered; this permits local lighting to vary over time.

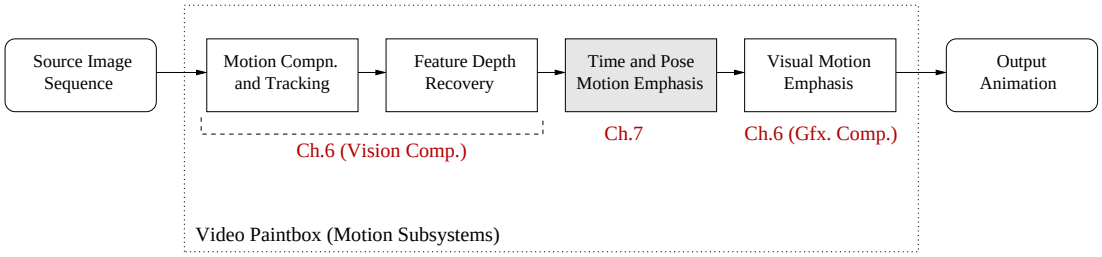

Figure 7-13 Schematic illustrating the flow of control and data in the Video Paintbox subsystems dealing with motion emphasis.

Fortunately t is straightforward to determine, since we have access to pose vectors which describe the feature's position in both the original and motion emphasised sequences. Suppose we wish to render a particular frame i. We first determine the feature's position at time i in the emphasised pose. We then examine each vector j of the original pose computing a spatial distance measure d_j — the mean Euclidean distance between the vertices of the original and emphasised feature at time j. We choose the start instant t as:

$$t = \text{argmin}_t(\alpha d_t + \beta|t - i|) \tag{7.32}$$

Choice of α and β depend on the rate of lighting variation in the video; in rapidly varying lighting conditions β should greatly outweigh α. For our data we have used $\alpha = 1$, $\beta = 0.1$. Figure 7-12 demonstrates the results of re-texturing the *CONTRAPTION* sequence following pose manipulation. We have simply overwritten the last dimension (angle between white and yellow slabs) of the first 50 frames with values from 0 to 2π in equal increments; this causes the slab to appear to spin about its pivot, whilst its motion remains coherent with the remainder of the contraption. Occlusions between the yellow and white slabs are correctly handled.

7.6 Integrating Time and Pose Cues within the Video Paintbox

Time and pose motion cues are applied directly to the output of the Computer Vision (Chapter 6), which tracks objects within a camera motion compensated version of the source video sequence. The result is a modified version of the video and associated tracked features, which are then passed to the Computer Graphics component of Chapter 6. Visual augmentation and deformation cues are then inserted into the animation — so completing the motion emphasis framework of the Video Paintbox. The benefit of this arrangement is that visual motion cues, such as object deformations, are seen to react to the changes in pose caused by effects such as anticipation. Figure 7-15 gives such an example where the *METRONOME* sequence has been subjected to the

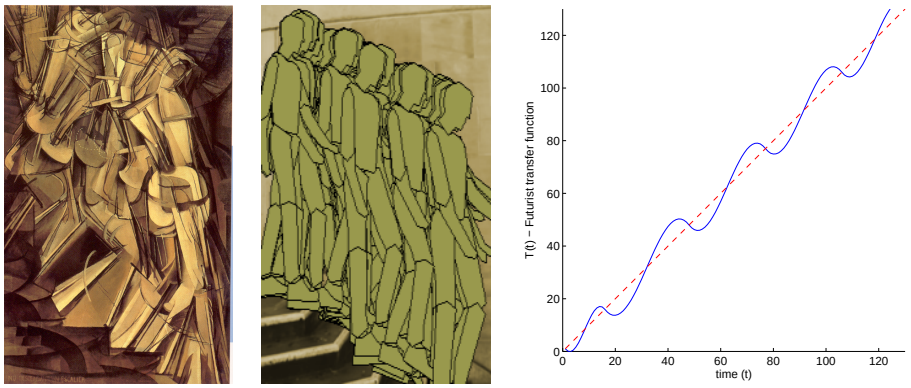

Figure 7-14 Toward Futurist-like rendering via irregular temporal sampling of the Video Paintbox output. Left: Duchamp's "Nude Descending a Staircase II" [1912], and samples of corresponding source photography by Muybridge [1872]. Middle: Our approximation to Futurist art using a temporal sampling strategy modelled upon Duchamp's work. Right: The transfer function $\mathcal{T}(.)$ which created our artwork in blue, with the standard (identity) transfer function in red.

anticipation process, after which a non-linear (velocity based) deformation has been used to emphasise drag.

7.7 Varying the Temporal Sampling Rate

Our explanation so far has assumed that the animator desires only to render pose vectors at uniform temporal intervals, and at precisely the same frame rate as the source video. However there is no technical reason why the animator should be restricted to this sampling strategy. As a final step we introduce a transfer function $\mathcal{T}(t)$ which accepts a time instant (i.e. a frame) to be rendered (t), and outputs the time instant of the animation to synthesise via our previously described framework. Standard rendering would simply require $\mathcal{T}(t) = t$. However, interesting effects can be created by choosing an alternative $\mathcal{T}(.)$.

The Futurist art movement created static artworks, many of which depict motion through the composition of time lapse images sampled at irregular intervals [81]. One classic example of such a work is Marcel Duchamp's "Nude Descending a Staircase II" [1912], which took its inspiration from the photographic work of Edwaerd Muybridge [1872] (see Figure 7-14, left). As an investigation to conclude our temporal motion emphasis work, we considered whether it would be possible to combine our tracked features (for example, arms and legs), with our temporal analysis to produce artwork similar in spirit to Duchamp's. To do so required the compositing of frames into a single image (a minor modification) and a more general functional $\mathcal{T}(.)$.

We observe that, rather than painting regular temporal instants, Duchamp painted

salient key-frames of the subject descending the staircase (Figure 7-14, left). These glimpses of motion are enough to suggest the full course of the movement, much as a few salient lines are enough to suggest form in a well drawn piece of artwork. We say the instants chosen by Duchamp are "temporally salient".

With this observation in mind, consider the smooth sections of Chapter 6 which form streak-line motion cues. The temporal regions around the start and end points of these streak-lines have high temporal salience relative to the remainder of the cue. A few points around each end of the streak-line are often sufficient for one to correctly predict the smooth trajectory of the object. Temporal salience is high at the streak-line origin, but decays as the streak-line progresses — falling to a minimum halfway along the trajectory. Temporal salience then increases again as we approach the terminus of the streak-line.

We have devised a non-linear transfer function $\mathcal{T}$, which varies sampling rate according to the magnitude of temporal salience at the current instant (i.e. proportional to the minimum time difference between the start and end of a streak-line). A plot of the resulting functional $\mathcal{T}(.)$ for the *STAIRS* sequence is shown in Figure 7-14, right. By compositing frames to create a single still image (we have chosen to also assign painting order in this composition to be proportional temporal salience), we obtain a result such as that of Figure 7-14, middle. Although the aesthetics of this output do leave something to be desired (static AR techniques could be applied to improve the final rendering), the composition of the artwork is, ostensibly, of a similar genre to that of Duchamp. We suggest that the notion of temporal salience warrants further investigation, and that the ability to define a user functional $\mathcal{T}(.)$ as a final stage in the motion emphasis pipeline serves as a simple, but interesting, means to access novel temporal rendering effects.

7.8 Summary and Discussion

We have described a subsystem within the Video Paintbox for automatically introducing "time and pose" cues into a video driven AR animation. This class of cue manipulates the timing of the animation, and includes traditional animation effects such as motion anticipation and exaggeration (motion cartooning). The "time and pose" subsystem integrates well with the visual motion emphasis work of the previous Chapter, and serves to broaden the gamut of motion cues available through our Video Paintbox.

We produce our time and pose cues by manipulating the positions of tracked features

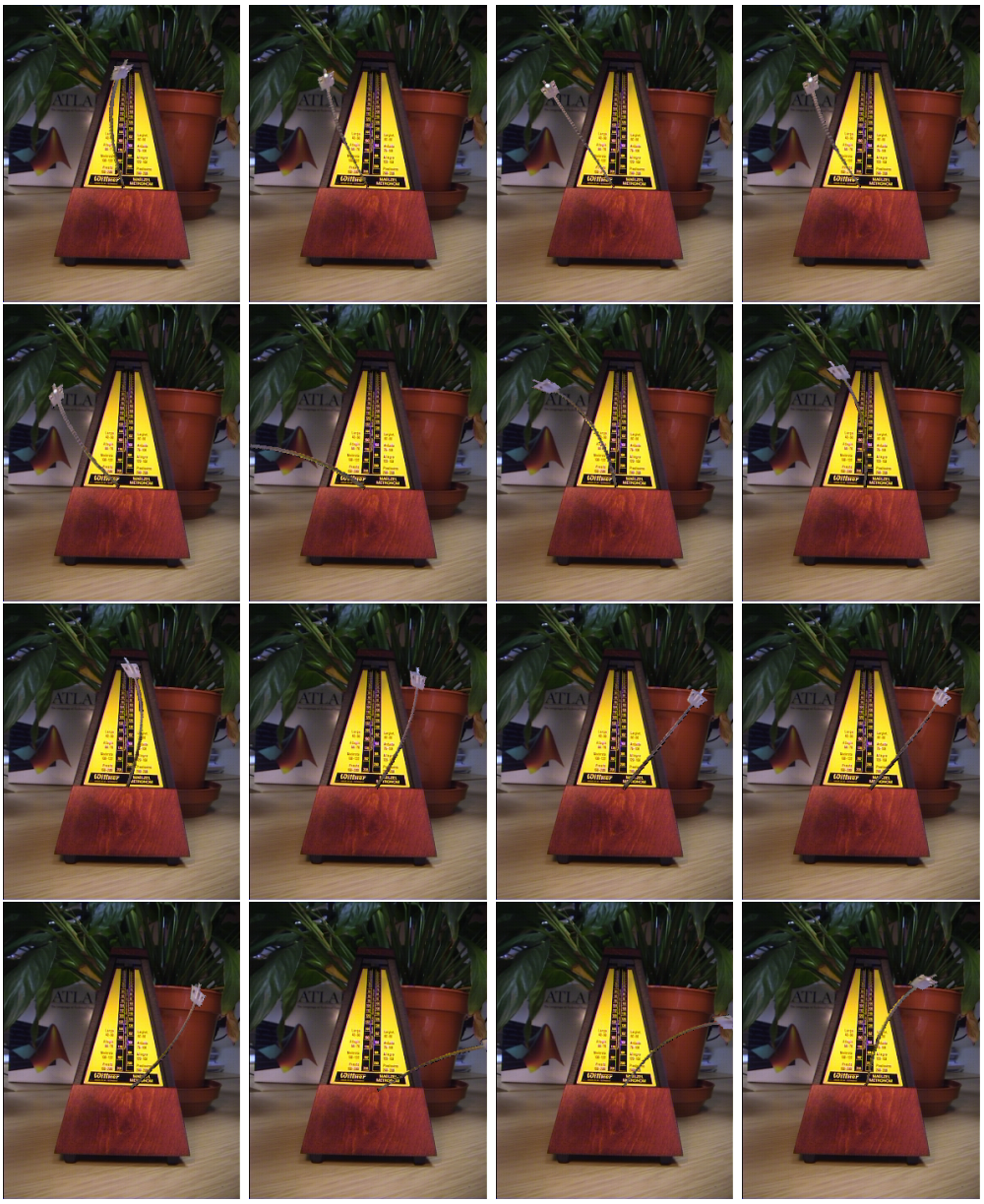

Figure 7-15 Stills taken from a section of the rendered *METRONOME* sequence, exhibiting the anticipation cue combined with a deformation motion cue emphasising drag (described in Section 6.4.2). Approximately one still per five frames. Observe that the visual deformation cues enable us to characterise movement in the scene, using only these still frames (see `videos/metro_warp_anticipate` and `videos/metro_anticipate` for animations).

over time. Recall that our initial experiments manipulated features independently by varying their LCAT transforms. The disappointing results which emerged motivated us to manipulate features using a hierarchical articulated structure, resulting in aesthetically superior animations. It is likely that the conceptually high level model of the articulated structure created more believable movement, because it more closely matches our mental model of the manner in which objects move — a subject's motion

is constrained by its inter-connecting joints, rather than allowing free motion of each component. If we refer back to the hand-drawn example of anticipation in Figure 7-5 it is clear that features of the face, for example eyebrows, are not anticipated using functions of their rotation, translation, etc. but according to a mental model of how facial parts move. This is again an example of how "time and pose" cues require a high level underlying model, in this case a facial muscle model rather than a rigid hierarchical structure. Future work might allow substitution of the current hierarchical articulated model for other models, so improving the generality of the system. We have also shown that the very nature of "time and pose" motion cues demands large temporal windows for analysis of the video sequence. Both the use of high level spatial models, and large temporal windows for motion analysis, are pre-requisites to synthesising time and pose cues.

There are a number of ways in which we might improve the work in this Chapter. We might seek to relax the assumptions on the nature of the motion, perhaps extending the system to emphasise non-planar motion. Alternatively we might revisit the problem of localising moving pivot points by allowing the animator to introduce a model of pivot motion. It may also be possible to improve accuracy of the pivot recovery algorithm (and so of subsequent pose recovery) using a Kalman filter to take advantage of the Gaussian distribution of error in pivot estimates, and so refine the pivot estimate over time.

A selection of source and rendered video clips have been included in Appendix C.