Video Phase-Locked Loops in Gait Recognition

Jeffrey E. Boyd
Dept. of Computer Science
University of Calgary
Calgary AB T2N 1N4
boyd@cpsc.ucalgary.ca

Abstract

In the perception of gaits, the relative timing of the individual motions in a gait is critical. When events occur periodically, as they do in a gait, then relative timing is equivalent to phase. Therefore, to perceive gaits requires frequency entrainment (finding the fundamental frequency of the gait) and phase locking (finding the phase relationships among the components of the gait). In this paper, we present an innovation in motion recognition we call the video phase-locked loop (PLL). While the innovation is simple, the resulting system is uniquely well suited to recognizing gaits because it performs both frequency entrainment and phase locking. A video PLL produces a pattern of phasors (rotating phase vectors) over a region in an image that encapsulates the relative timing of motion in the images. We show how the Procrustes method of statistical shape analysis can be used to compute means over a set of phasor patterns and distances between pairs of patterns. They combine to form the basis of a gait recognition system: the means provide exemplars, while the distance measure allows matching of patterns to exemplars. We provide a set of examples that demonstrates the operation of video PLLs and the application of Procrustes shape analysis to the resulting phasor configurations.

1. Introduction

In the perception of gaits, timing is everything; specifically, the relative timing of the individual motions in a gait, and when events occur periodically, as they do in a gait, then relative timing is equivalent to phase. For example, as a person walks, their left and right legs swing forward *alternately*, i.e., the legs swing in opposite phases. In contrast, consider a gait in which the legs swing *together*, i.e., in phase. The gait would appear to be a bizarre hopping motion. Clearly, the phases of component motions in a gait are essential to its perception.

The importance of phase in gaits appears in the medical, psychology and computer vision literature. Murray et al. [15] show a detailed analysis of the gaits for several *normal* humans. The analysis focuses on the timing of joint and limb movements throughout the cycle of the gait. Of particular interest are the measurements of *forward displacement* which show the sinusoidal variations in a subjects forward progress as a function of position in the gait cycle. In subsequent work, Murray [14] shows that the these sinusoidal variations change in phase for subjects that have abnormal gaits due to some physical affliction. The changes in the gaits are manifest in the timing patterns. Bertenthal and Pinto [3] identify *frequency entrainment* and *phase locking* as important properties in the perception of gaits. Frequency entrainment refers to the requirement that the various components of the gait must share a common frequency, while phase locking is the requirement that the phase relationships among the components of the gait remain approximately constant. The lock varies for different types of locomotion such as walking versus running.

The importance of phase has also been shown in systems that synthesize gaits. For example, Laszlo et al. [9] describe a limit-cycle control system for animation of walking figures. Varying phases in the limit cycles alters the type of synthesized gait. It also alters the perceived gait. Unuma et al. [17] describe a Fourier-based system for gait synthesis. Their system derives Fourier coefficients, consisting of magnitude and phase, for the periodic joint angles in a gait. One set of coefficients represents a *normal* gait while another set represents a variation such as a *tired* gait. By interpolating or extrapolating both the magnitude and phase coefficients they are able to blend subjectively different gaits and realize a continuum of gaits. The phase component of the Fourier series plays a significant role in the synthesis of the varying gaits. In related work, Amaya et al. [1] synthesize motions that convey emotion by rescaling the timing and amplitude of a set of non-periodic motions based on measurements of normal and expressive examples.

Other than the *forward displacement* measured by Murray [14], the analyses of gait cited above are based on a kinematic model. They refer to the motion of individual joints or limbs. Any perception of the gait must then reconcile the many measurements for the body parts. In contrast, Cutler and Davis [7] describe a method to detect periodic motions without a kinematic model. Little and Boyd [10] also use a *shape-of-motion* model-free approach that combines motion over an entire moving figure into a set of scalars that describes the shape of the region moving in an image. Phase differences in the variation of the scalars are able to distinguish among gaits of different human subjects. They suggest that the scalars are able to recognize gaits because they represent a spatial average of the phase information at individual points in the moving figure [5]. Boyd and Little [6] also showed that, at least with synthetic gaits, the phase-based shape-of-motion is sensitive, albeit weakly, to qualitatively different gaits.

Polana and Nelson [16] recognize periodic activities by finding the intensity of periodic variations in different image regions. Liu and Picard [11] also find periodic motion in gait sequences, relying on on the magnitude information in a Fourier spectrum. These methods do not exploit the phase and timing information that is essential for gait perception.

What we seek then, is a system that perceives gaits. We know that it must be sensitive to the phases of the component motions. We add to this a requirement that the system be model-free. This is based on a conviction that there is ample information in motion sequences for gait perception without the need to fit a kinematic model.

To that end, we introduce the innovation of combining gait perception with the well-established technology of the phase-locked loop (PLL) in what we call a video PLL. PLLs provide a mechanism to perform two critical tasks (as identified by Bertanthal and Pinto [3]) in gait perception: frequency entrainment and phase locking. A PLL can lock on oscillations in pixels that arise because of oscillatory motion. In doing so, the PLL matches its internal oscillator to the oscillations in a pixel, thus performing frequency entrainment. When a PLL is applied to each pixel in a motion image, the PLL oscillators can indicate the relative phases of the oscillations throughout the image, thus performing phase locking. The abundance of data extracted by the video PLL makes gait recognition possible, as we demonstrate.

A video PLL can detect oscillatory motion in images by treating the pixels independently. Thus, video PLLs are distinct from optical flow, and the two should not be confused. Video PLLs are a tool uniquely well-suited to dealing with oscillations, such as those in a gait.

2. Video Phase-Locked Loops

2.1. Phase-Locked Loops

Figure 1(a) shows a block diagram for a basic PLL. PLLs are found primarily in communications systems where they are used to lock on carrier signals. Its basic components are a phase detector (PD), a low-pass (LP) loop filter, and a voltage-controlled oscillator (VCO). The literature on PLLs is extensive. Best [4] provides an excellent introduction to PLLs.

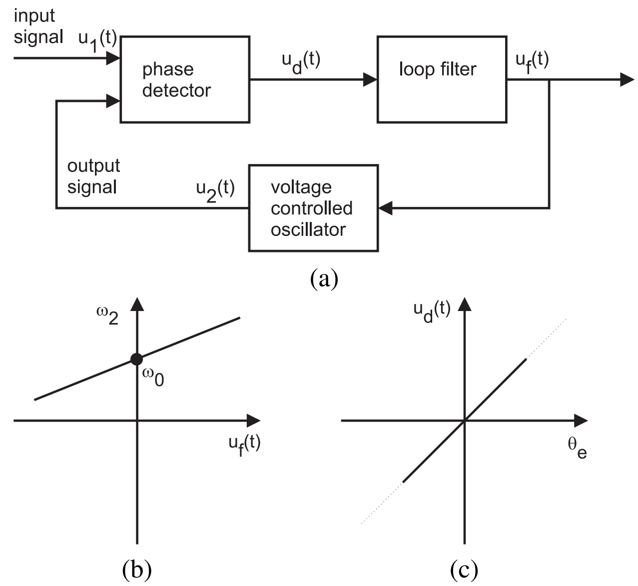


Figure 1. The basic phase-locked loop: (a) a block diagram of the loop, (b) the transfer function of the voltage-controlled oscillator, and (c) the transfer function of the phase detector. The loop filter is usually a low-pass filter.

To understand a PLL, suppose we initially know nothing about the frequency of the VCO output. The phase detector produces a *phase difference* signal, $u_d(t)$, that is proportional to the difference between the phases of $u_1(t)$ and $u_2(t)$. If u_1 and u_2 oscillate at different frequencies, u_d will increase or decrease at a constant rate. The loop filter removes high-frequency components of u_d to obtain the filtered phase difference signal, $u_f(t)$. u_f feeds back to the VCO, changing the frequency of VCO oscillations until the frequencies of u_1 and u_2 are equal. At this point the phase difference will remain constant, and the loop reaches a steady state and is said to be locked. The PLL has two outputs: the VCO output u_2, and the LP loop filter output, u_f. Figure 1(b) shows the linear relationship between the VCO control signal, u_f, and the frequency it oscillates at. K_0 is the slope of the line, while the intercept, $\omega_0 = 2\pi f_0$, is the center frequency of the oscillator. Figure 1(c) shows the linear relationship between the phase-detector output, u_d, and the phase difference between u_1 and u_2, θ_e. The slope is K_d and the intercept is zero. Best gives a set of PLL design equations summarized in Table 1.

There is a variety of ways to implement PLLs. For the

Parameter	Description	Equation
ω_0	oscillator center frequency	
τ_1,τ_2	loop filter time constants	
ω_n	natural frequency of PLL	$\omega_n = \sqrt{\frac{K_0 K_d}{\tau_1+\tau_2}}$
ζ	damping factor	$\zeta = \frac{\omega_n}{2}\sqrt{\tau_2 + \frac{1}{K_0 K_d}}$
$\Delta\omega_H$	hold range	$\Delta\omega_H = K_0 K_d$
$\Delta\omega_L$	lock range	$\Delta\omega_L \approx 2\zeta\omega_n$
T_L	lock-in time	$T_L \approx \frac{2\pi}{\omega_n}$

Table 1. Design equations for PLLs [4], based on linear PLLs, with passive lag loop filters.

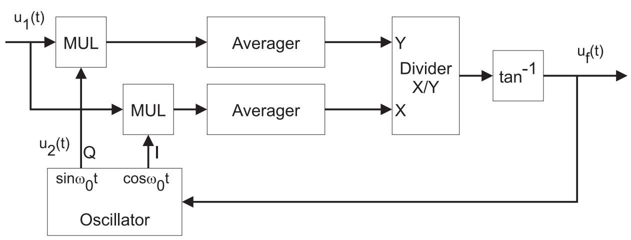


Figure 2. Phase-locked loop using a digital-averaging phase detector.

video PLLs described here we have opted for a software implementation using quantized signals sampled at discrete uniform time intervals, referred to as an all-software PLL. This allows us to use a digital-averaging PD, Figure 2. The oscillator produces both an in-phase and a quadrature signal (labeled I and Q respectively), each of which is multiplied with the input signal. The loop filter is brought inside the phase detector and averages or smoothes the in-phase and quadrature products. A division and inverse tangent operation give the phase angle between u_1 and u_2 as required. The digital-averaging PD is not sensitive to the magnitude of u_1 (as is the case with other PDs) giving $K_d = 1$ always. The in-phase and quadrature products can also be used to derive the magnitude of u_1 by computing $\sqrt{X^2 + Y^2}$.

2.2. Video Phase-Locked Loops

A video PLL consists of a set of independent PLLs, each operating at a single pixel site. The intensity value of a pixel taken over time gives a temporal sequence. Typically pixel intensities are non-negative so that when observing an oscillating motion the temporal sequence consists of a constant background value with a sinusoid (or perhaps several sinusoids) superimposed on top. Since the PLL locks on sinusoids only, the video PLL uses a temporal band-pass filter to pre-processes the signal to remove the background and pass only the sinusoid.

At first it might appear the independent operation of the PLLs would be troublesome. It turns out that this is not the case, at least for gait analysis, for several reasons. The band-pass filter allows sinusoids only in a given frequency band to pass through to the PLL. Also, the PLL can lock only within a given frequency band (the PLL and band-pass filter must have matching frequency bands for this reason). This combination restricts the range of frequencies that the PLL can lock. Given that a gait has only a fundamental frequency and harmonics (Boyd and Little [6]) the individual PLLs are likely to all lock to the same frequency, since the band-pass filter and PLL locking range can only lock onto a single frequency component of the gait. Still it may be possible that other frequency components show up in the background of a gait. The Procrustes shape analysis discussed in Section 4 sees these stray frequencies as noise that obscures a signal. The signal is recovered by averaging over time to remove the noise. Thus, there is little need to couple the individual PLLs.

This small innovation has a significant impact on machine perception of gaits and other oscillatory motion. As we will show in the remainder of this paper, video PLLs can perform frequency entrainment and phase locking quickly. Furthermore, they extract sufficient information to enable recognition of gaits and other oscillatory motions.

2.3. Signals in the Phase-Locked Loop

Within a video PLL there is an abundance of signals useful in the perception of oscillating motion. The output of the loop, u_f is a phase error, but also gives an instantaneous estimate of the frequency of oscillation in a pixel. The relationship between u_f and the frequency is given by the VCO transfer function, Figure 1(b). As stated previously, for a gait, all pixels for the walker will lock to the same frequency. This is how a video PLL does frequency entrainment.

With all oscillators all locked on the same frequency, the only difference between oscillators is the phase of their oscillation. The oscillator produces its output by computing the phase of the oscillation at each time interval, and from there computing the in-phase and quadrature components of u_2. The relative phases of the oscillators throughout the image give the relative phases of the intensity oscillations themselves. This performs the task of phase locking. Thus, the video PLL gives a picture of the relative phases of oscillations, at an entrained frequency, throughout the image.

The use of a digital-averaging PD has the added benefit of not only computing a phase difference between u_1 and

u_2, but it can also compute the magnitude of the u_1. This is useful in giving an indication of the amplitude, or strength, of the locked signal. By combining the magnitude with the phase, we get a phasor (phase vector) at each pixel site. The combination of phasors over several pixels forms a pattern that rotates at the locked frequency. We can use the pattern to classify and recognize oscillatory motion.

3. Examples

3.1. Sample Video PLL Design

Here we present a complete design for a video PLL based on a 30 *fps* sample rate, to lock on a step rate (foot fall to foot fall) of approximately 1.5 *steps/s*. We normalize all aspects of the design to the sample rate so the target frequency is $1.5/30 = 0.05$. Using the bilinear transform method described by Antoniou [2] we design a infinite-impulse-response (IIR) Butterworth band-pass filter with second order roll-off. The filter has cut-off frequencies at 0.0375 and 0.075 giving the transfer function:

$$H_{bp}(z) = \frac{0.011 - 0.024z^{-2} + 0.012z^{-4}}{1 - 3.47z^{-1} + 4.69z^{-2} - 2.93z^{-3} + 0.72z^{-4}}$$

The center-frequency of the PLL oscillator is set to the center of the pass band for $H_{bp}(z)$, $f_0 = 0.05625$. As stated previously, with the digital-averaging PD, the PD gain, K_d, is unity. The remainder of the PLL parameters, τ_1, τ_2, and K_0, we derive by trial and error using the design equations in Table 1. The transfer function of the passive-lag low-pass filter, in the Laplace transform domain, is given by:

$$H_{lp}(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)}$$

We find it more intuitive to work the two break frequencies of the filter, $f_1 = 1/(\tau_1 + \tau_2)$ and $f_2 = 1/\tau_2$. A narrower passband results in less ripple (caused by the multiplication step in the PD) passed through the filter, but increases the locking time of the PLL. We settled on $f_1 = 0.02$ and $f_2 = 0.4$. The frequencies are converted to the continuous frequency domain using a bilinear transform, τ_1 and τ_2 computed, and the filter transfer function transformed back into the discrete frequency domain [2]. The resulting filter is given by:

$$H_{lp}(z) = \frac{1.325 + 0.675z^{-1}}{16.89 - 14.89z^{-1}}$$

The gain of the oscillator is $K_0 = 0.2$. This parameter selection results in the following characteristics of the PLL.

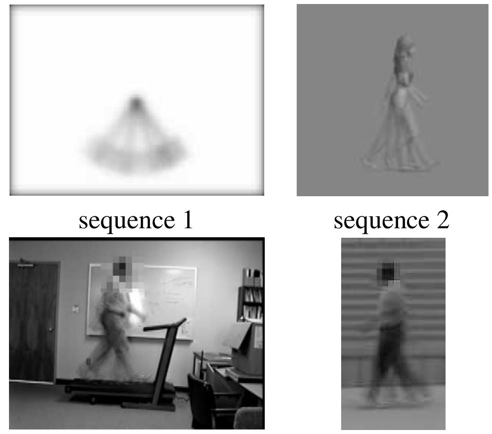


Figure 3. Sample sequences: (1) synthetic pendulum, (2) a synthetic gait generated by Poser [13], (3) a treadmill gait, and (4) a translating gait.

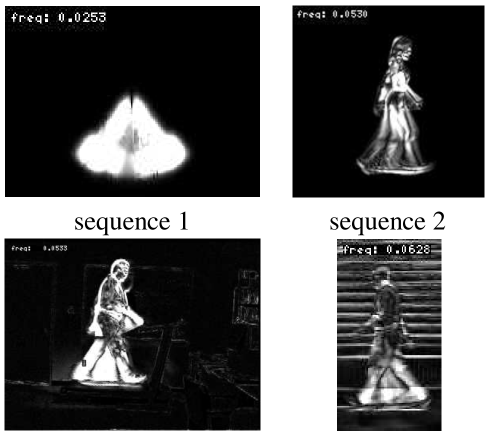
oscillator center frequency	$\omega_0 = 0.353$, $f_0 = 0.05625$
filter time constants	$\tau_1 = 7.785$, $\tau_2 = 0.162$
natural frequency	$\omega_n = 0.225$, $f_n = 0.0358$
damping factor	$\zeta = 0.41$
hold range	$\Delta\omega_H = 0.2$, $\Delta f_H = 0.0318$ $0.024 \leq f \leq 0.088$
lock range	$\Delta\omega_L = 0.13$, $\Delta f_L = 0.0207$ $0.036 \leq f \leq 0.077$
lock-in time	$T_L = 39.6 frames$

With this design, substantial ripple passes through to the output, u_f, but the PLL locks quickly, $T_L \approx 40 frames$. This does not affect the oscillator, but instantaneous frequency estimates exhibit the ripple. Additional filtering, outside the feedback loop, can be used to smooth instantaneous frequency readings.

3.2. Sample Sequences

Figure 3 shows, as superimposed frames, four image sequences containing oscillatory motion. The sequences show a synthetic pendulum, a synthetic gait create using Poser [13], a person walking on a treadmill, and a person walking across the field of view of a camera. In the first three sequences, the motion is at a fixed position in the image, while in the last the subject moves across the image. For the last case, we manually track the subject and extract a set of images for which subject's position is fixed.

Each of the sequences in Figure 3 is processed by a video PLL and the results are in Figure 4. The synthetic pendulum uses a video PLL with a center frequency $f_0 = 0.02375$,



sequence 1 sequence 2

sequence 3 sequence 4

Figure 4. Video PLL magnitude signals for sample sequences in Figure 3.

while the others use the video PLL design shown in the previous section. There are many signals within the PLL that we could display. We have chosen to show the magnitude signal, the strength of the locked oscillation; oscillations are bright, no oscillations are dark. There is nothing in these images to show that all bright areas have locked to the same frequency. However, the mean frequency locked over the image, weighted by the magnitude is shown in the image. These numbers correspond closely with expected values. For the gait examples, the video PLL is locking on the step rate.

4. Gait Recognition

There is ample information derived by a video PLL to recognize various oscillatory motions. Although the phases in a gait must be locked, the actual phase angles are constantly changing through time. We require a method that allows us to compare a sets of phasors allowing for their rotation. To that end, we describe Procrustes shape analysis, and then show its application to recognition of phasor patterns.

4.1. Procrustes Shape Analysis

Procrustes shape analysis [8] is a method in directional statistics [12]. It provides a method to find mean shapes, i.e., patterns of points in two-dimensions, and a measure of the similarity of two shapes. The following is a summary.

We describe a shape in two-dimensions using a vector of k complex numbers, $\mathbf{z} = [z_1, z_2, \ldots, z_k]^T$, called a configuration. Two shapes, $\mathbf{z}_1$ and $\mathbf{z}_2$, are the same shape if by a combination of translation, scaling, and rotation, their configurations are equal, i.e.,

$$\begin{aligned} \mathbf{z}_1 &= \alpha \mathbf{1}_k + \beta \mathbf{z}_2, \;\; \alpha, \beta \in \mathcal{C} \\ \beta &= |\beta| e^{i\angle\beta}, \end{aligned}$$

where $i = \sqrt{-1}$. That is, $\alpha \mathbf{1}_k$ translates $\mathbf{z}_2$, and $|\beta|$ and $\angle\beta$ scale and rotate $\mathbf{z}_2$. It is convenient to center shapes by defining the centered configuration $\mathbf{u} = [u_1, u_2, \ldots, u_k]^T$, $u_i = z_i - \bar{z}$, and $\bar{z} = \sum_{i=1}^{k} z_i / k$. From here we can define the *full Procrustes distance* between two configurations, $d_F(\mathbf{u}_1, \mathbf{u}_2)$ as:

$$d_F(\mathbf{u}_1, \mathbf{u}_2) = 1 - \frac{|\mathbf{u}_1^* \mathbf{u}_2|^2}{\|\mathbf{u}_1\|^2 \|\mathbf{u}_2\|^2} \tag{1}$$

which minimizes

$$\left\| \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} - \alpha \mathbf{1}_k - \beta \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \right\|^2 . \tag{2}$$

$\mathbf{u}^*$ denotes the transpose of the complex conjugate of $\mathbf{u}$. Note that $0 \leq d_F \leq 1$. The Procrustes distance allows us to compare two shapes independent of position, scale, and rotation. We can find the mean of a set of n shapes by finding the μ that minimizes the objective function

$$\min_{\alpha_j, \beta_j} = \sum_{j=1}^{n} \|\mu - \alpha_j \mathbf{1}_k - \beta_j \mathbf{u}_j\|^2 . \tag{3}$$

To find μ we compute the matrix

$$\mathbf{S}_u = \sum_{j=1}^{n} (\mathbf{u}_j \mathbf{u}_j^*)/(\mathbf{u}_j^* \mathbf{u}_j). \tag{4}$$

The *Procrustes mean shape*, $\hat{\mu}$, is the dominant eigenvector of $\mathbf{S}_u$, i.e., the eigenvector that corresponds to the greatest eigenvalue of $\mathbf{S}_u$.

4.2. Shape Analysis for Phasor Patterns

Although Procrustes shape analysis is intended for treating two-dimensional shapes, it is easily adapted to handling spatial patterns of rotating phasors. For any single frame, a video PLL produces a phasor that expresses a magnitude and angle, which equivalently can be expressed as a complex number. Thus we can treat the phasor pattern in the pixels as a shape configuration vector $\mathbf{z}$. For an image sequence with oscillatory motion, the configuration rotates.

Procrustes shape analysis says that shapes are invariant through translation, scaling, and rotation. It handles translational invariance by using the centered configuration $\mathbf{u}$ to move all shapes to a common center. When using phasors the issue of translation becomes irrelevant. All phasors rotate about the origin, $0 + 0i$, at the entrained frequency. The configurations, $\mathbf{z}$, are already centered, so $\mathbf{z} = \mathbf{u}$.

The following steps indicate how to find the Procrustes mean shape for a motion sequence of n frames.

1. Select a set of k pixel sites from the images. The set is the same for every image.
2. For each frame in the sequence, form a phasor from the magnitude and phase of the locked signal at the selected pixel sites. Place the complex phasors into a $k \times 1$ vector, $\mathbf{z}_j$, $j = 1, 2, \ldots, n$, a phasor configuration.
3. Set the centered configurations $\mathbf{u}_j = \mathbf{z}_j$, $j = 1, 2, \ldots, n$.
4. Compute the matrix $\mathbf{S}_u$ using Equation 4.
5. Compute the eigenvectors and eigenvalues of $\mathbf{S}_u$.
6. Set the Procrustes mean shape, $\hat{\mu}$, to the eigenvector that corresponds to the maximum eigenvalue.

We can then use the Procrustes mean shape to compare two motion sequences as follows.

1. Compute the Procrustes mean shape for the phasor patterns in each sequence.
2. Find the full Procrustes distance between the two mean shapes using Equation 1.

Note that this procedure implies some sort of registration between sequences to ensure correct correspondence of the selected pixel sites.

Suppose that in the set of k pixels some pixels are not oscillating at the frequency of the gait due to either oscillatory motion in the background, or noise. Two things can happen.

1. **Oscillatory background motion:** The configuration $\mathbf{z}$ can be partitioned into two parts, $\mathbf{z} = [\mathbf{z}_{fg}^T \mathbf{z}_{bg}^T]^T$, corresponding to foreground and background. The eigenvalue decomposition of $\mathbf{S}_u$ produces two large eigenvalues, one corresponding to $\mathbf{z}_{fg}$ and the other to $\mathbf{z}_{bg}$. There relative strengths of the eigenvalues depends on the number of pixels in each partition.
2. **Background noise:** The configuration $\mathbf{z}$ can be partitioned into two parts, $\mathbf{z} = [\mathbf{z}_{fg}^T \mathbf{z}_{noise}^T]^T$, corresponding to foreground and noise. The eigenvalue decomposition of $\mathbf{S}_u$ produces one large eigenvalue corresponding to $\mathbf{z}_{fg}$. The magnitude of the eigenvalue is smaller due to the smaller number of contributing pixels.

Thus, Procrustes analysis gives us some immunity to background oscillations and noise.

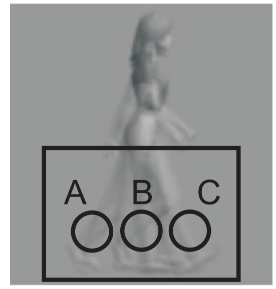


Figure 5. Regions sampled for Procrustes shape analysis.

4.3. Results

Following the above procedure for computing the Procrustes mean shape of a sequence we select k pixels by coarsely sub-sampling, within a box that covers either the pendulum or the legs of the pedestrians, as illustrated in Figure 5. After computing the Procrustes mean, we face the dilemma of how to view an image of phasor values. Our approach is to reconstruct a new motion sequence from the phasors by spinning them and extracting the real component, i.e.,

$$\mathbf{x}_j = \mathrm{real}(e^{i\omega}\hat{\mu}),$$

where $\mathbf{x}_j$ is the j^{th} image in the generated sequence and ω is the angular velocity of the phasors. This should create an image equivalent to the video PLL input (band-pass filter output) with the advantage of the PLL and subsequent Procrustes averaging. Note that we are free to choose ω and can effectively change the frequency of the gait.

The results are in Figure 6. Each shows a single frame from the reconstructed sequences.

We currently lack an automated and efficient method to register two regions of oscillating motion. Therefore, to find the full Procrustes distance between the four sequences, we manually select three pixel sites in each sequence corresponding to regions A, B, and C in Figure 5. The phasors corresponding to these sites are shown in Figure 7.

Having extracted these three phasors, we can compare the configurations for the four sample sequences using the full Procrustes distance measure, Equation 1. The results are shown in Table 2. Values for d_F are in the range $0 \leq d_F \leq 1$ with 0 indicating a perfect match and 1 indicating no match. The pendulum is very different from the gait sequences. Sequence three matches the other gaits well, while the match between sequences two and four is ambiguous.

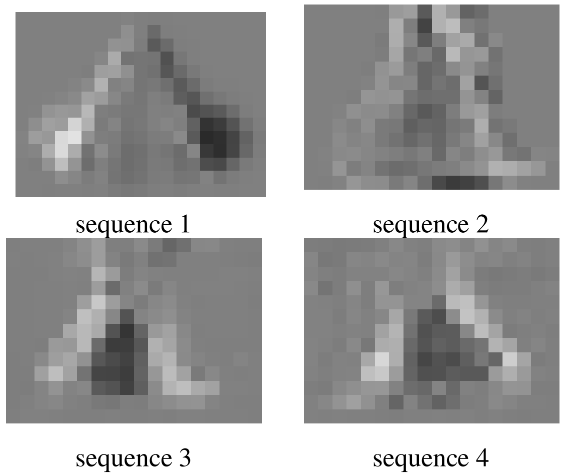


Figure 6. Procrustes mean shapes for sample sequences, projected onto real axis.

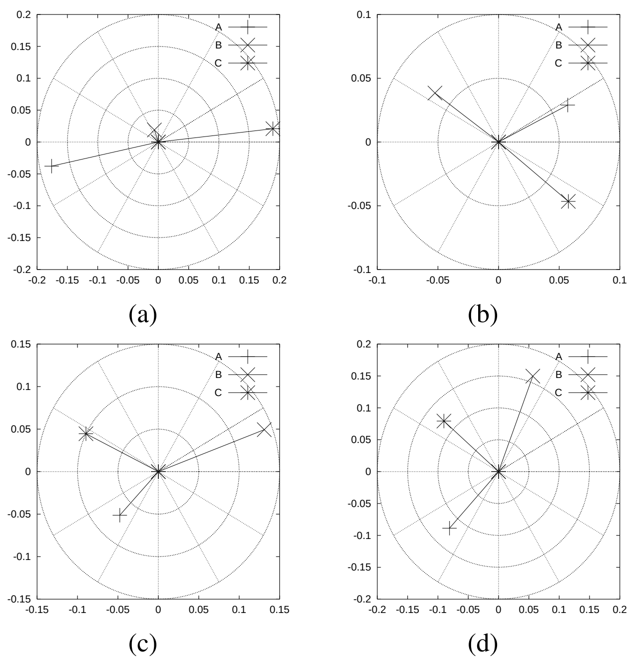


Figure 7. Phasor for sample sequences.

	1	2	3	4
1	0	0.84	0.90	0.79
2		0	0.19	0.56
3			0	0.26
4				0

Table 2. Full Procrustes distances between phasor configurations for sample sequences.

5. Discussion

Does the video PLL compute optical flow? No, it does not. Optical flow considers both temporal as spatial variations in image intensities. Video PLLs examine oscillations at a single pixel site only. Spatial factors only come in to play when the phasor from several pixel sites are combined to form phase configurations.

The illusion of motion created by a string of lights flashing in sequence comes to mind. Video PLLs and Procrustes analysis suggests that optical flow is not necessary to perceive the motion. The video PLL will lock to the oscillations of the individual lights and establish the timing/phase patterns between them. The resulting phase configuration indicates the perceived motion.

Other inputs: As presented here, the video PLLs use only raw pixel intensity values as input. Although this has worked well so far, we can imagine situations in which images must be processed before going into the video PLL. For example, suppose an object with a pattern that oscillates in the spatial dimension (a repeating pattern or texture) translates across an image. This will produce oscillations at pixel sites and may confound recognition from video PLL generated phasors. One alternative would be to use optical flow. The hypothetical translating textured object would then produce a constant optical flow with no oscillations.

Computational issues: PLLs are well known and are currently used in communications devices that operate at frequencies many orders of magnitude faster than video rates. Therefore, for video PLLs to run in real time requires that we have N very slow PLLs operating concurrently, where N is the number of pixels in an image. This is certainly within reach of current workstation technology. The full Procrustes mean distance requires computation of three inner products, something that is not computationally demanding. The Procrustes mean shape requires computing the $k \times k$ matrix $\mathbf{S}_u$ and its eigenvectors, where k is the number of pixels used in the configuration. If k is large, this can be very time consuming. Therefore, it is desirable to keep k small by using only relevant points in an image. Also, mean shapes can be computed off-line to find exemplar patterns for recognition, and only the full Procrustes distance to these exemplars need be calculated on line.

A complete gait recognition system. We have presented the bulk of what is required to build a gait recognition system. The method to acquire features for recognition, phasor configurations, is defined as well as a mechanism for computing exemplar configurations and matches to the exemplars. To complete the system we need to register the pixels in different video sequences. The first step is to track a moving object. Our sample sequences one through three do not require this either by virtue of being synthetic or by using a treadmill. Sequence four required tracking the subject manually. Once tracking is done we get a nearly stationary oscillating motion. The registration is completed by selecting a configuration of k pixels to use for recognition. These must be somehow distributed about an oscillating region so that the correct correspondence between phasors results. Methods for tracking moving objects are well-known. Part of our future work will be to define a method for selecting pixels in the phasor configuration.

What can the system recognize? We have yet to collect enough data to answer this questions. The difference between the pendulum and pedestrians is large and we expect, therefore, that the system will distinguish among a variety of oscillatory motions such as gaits, waving, a pendulum, a wheel rotating, or a cyclist. It may be possible, although we do not yet know, to distinguish different types of gaits (e.g., running versus walking), or the gaits of different people.

Phase recognition using models: Although, this work has focused on model-free recognition, Procrustes shape analysis is equally useful for model-based methods. One can form a phase configuration using the oscillatory trajectories of joints or limbs as determined by model parameters. The phase configuration and Procrustes analysis can then be used as demonstrated here, to form exemplars and compare configurations.

6. Conclusions

We have presented the innovation in motion recognition of the video phase-locked loop. While the innovation is simple, the resulting system is uniquely well suited to the task of recognizing oscillatory motions such as gaits.

Video PLLs have the ability to perform frequency entrainment and phase locking, two things that are required for the perception of gaits. The result of the entrainment and locking is a configuration of phasors over the oscillating regions in an image. These configurations can be combined to give a mean configuration (Procrustes mean shape) over a set of images, and be compared for similarities and differences (full Procrustes distance). The combination of the video PLL and Procrustes shape analysis forms the cornerstone of a gait and other oscillatory motion recognition system.

References

[1] K. Amaya, A. Bruderlin, and T. Calvert. Emotion from motion. In *Graphics Interface 96*, 1996.

[2] A. Antoniou. *Digital filter analysis and design*. McGraw–Hill, New York, 1979.

[3] B. I. Bertenthal and J. Pinto. Complementary processes in the perception and production of human movements. In L. B. Smith and E. Thelen, editors, *A Dynamic Systems Approach to Development: Applications*, pages 209–239. MIT Press, Cambridge, MA, 1993.

[4] R. E. Best. *Phase-locked loops design, simulation and applications*. McGraw–Hill, New York, 1999.

[5] J. E. Boyd and J. J. Little. Global versus structured interpretation of motion: moving light displays. In *IEEE Nonrigid and Articulated Motion Workshop, CVPR 97*, pages 18–25, 1997.

[6] J. E. Boyd and J. J. Little. Phase in model-free perception of gait. In *IEEE Workshop on Human Motion*, Austin, TX, December 2000.

[7] R. Cutler and L. Davis. Robust periodic motion and motion symmetry detection. In *Computer Vision and Pattern Recognition 2000*, 2000.

[8] J. T. Kent. New directions in shape analysis. In K. V. Mardia, editor, *The art of statistical science: a tribute to g. s. watson*, pages 115–127. Wiley, Chichester, 1992.

[9] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *SIGGRAPH 96*, pages 155–162, 1996.

[10] J. J. Little and J. E. Boyd. Recognizing people by their gait: the shape of motion. *Videre*, 1(2):1–32, 1998.

[11] F. Liu and R. W. Picard. Finding periodicity in space and time. In *International Conference on Computer Vision*, 1998.

[12] K. V. Mardia and P. E. Jupp. *Directional statistics*. Wiley, Chichester, 2000.

[13] MetaCreations Corporation. *Poser 4 user guide*, 1999.

[14] M. P. Murray. Gait as a total pattern of movement. *American Journal of Physical Medicine*, 16(1):290–332, 1967.

[15] M. P. Murray, A. Bernard, and R. C. Kory. Walking patterns of normal men. *The Journal of Bone and Joint Surgery*, 46A(2):335–359, March 1964.

[16] R. Polana and R. Nelson. Detection and recognition of periodic, nonrigid motion. *International Journal of Computer Vision*, 23(3):261–282, June 1997.

[17] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *SIGGRAPH 95*, pages 91–96, 1995.