conga: A Framework for Adaptive Conducting Gesture Analysis

Eric Lee, Ingo Grüll, Henning Kiel, and Jan Borchers
Media Computing Group
RWTH Aachen University
52056 Aachen, Germany
{eric, gruell, kiel, borchers}@cs.rwth-aachen.de

ABSTRACT

Designing a conducting gesture analysis system for public spaces poses unique challenges. We present *conga*, a software framework that enables automatic recognition and interpretation of conducting gestures. *conga* is able to recognize multiple types of gestures with varying levels of difficulty for the user to perform, from a standard four-beat pattern, to simplified up-down conducting movements, to no pattern at all. *conga* provides an extendable library of *feature detectors* linked together into a directed acyclic graph; these graphs represent the various conducting patterns as gesture *profiles*. At run-time, *conga* searches for the best profile to match a user's gestures in real-time, and uses a beat prediction algorithm to provide results at the sub-beat level, in addition to output values such as tempo, gesture size, and the gesture's geometric center. Unlike some previous approaches, *conga* does not need to be trained with sample data before use. Our preliminary user tests show that *conga* has a beat recognition rate of over 90%. *conga* is deployed as the gesture recognition system for *Maestro!*, an interactive conducting exhibit that opened in the Betty Brinn Children's Museum in Milwaukee, USA in March 2006.

Keywords

gesture recognition, conducting, software gesture frameworks

1. INTRODUCTION

Orchestral conducting has a long history in music, with historical sources going back as far as the middle ages; it has also become an oft-explored area of computer music research. Conducting is fascinating as an interaction metaphor, because of the high "bandwidth" of information that flows between the conductor and the orchestra. A conductor's gestures communicate beat, tempo, dynamics, expression, and even entries/exits of specific instrument sections. Numerous researchers have examined computer interpretation of conducting gestures, and approaches ranging from basic threshold monitoring of a digital baton's vertical position, to more sophisticated approaches involving artificial neural networks and Hidden Markov Models, and even analyzing data from multiple sensors placed on the torso, have been proposed.

NIME 06, June 4-8, 2006, Paris, France

Figure 1: *Maestro!*, an interactive conducting exhibit for children that we developed, at the Betty Brinn Children's Museum in Milwaukee, USA. *Photo appears courtesy of the Betty Brinn Children's Museum in Milwaukee, WI, USA.*

Our work is motivated by research on novel computer music and multimedia interfaces for public spaces such as museums (see Figure 1), and *conga* builds on our prior experience with designing interactive orchestral conducting exhibits, including *Personal Orchestra*, an exhibit for the HOUSE OF MUSIC in Vienna (coordinated by Max Mühlhäuser, now at Darmstadt University) [1] and a collaboration with Teresa Marrin Nakra on *You're the Conductor*, a children's exhibit for the Boston Children's Museum [9]. Our systems allow the user control over tempo, by making faster or slower gestures; volume, by making larger or smaller gestures; and instrument emphasis, by directing the gestures towards specific areas of a video of the orchestra on a large display (instrument emphasis is not supported in *You're the Conductor*). Designing a gesture recognition system for a museum-type environment poses unique and interesting challenges, primarily because museum visitors have a wide range of experience with conducting. Moreover, there is little to no opportunity to either train a user to use the system, or to train the system to a specific user; a museum on a busy day may see over 1000 visitors, and so a visitor will spend, on average, less than one minute at an exhibit.

In this paper, we present *conga*, a system for **con**ducting **g**esture **a**nalysis. Unlike current systems, *conga* does not restrict the user to conduct in a specific way, nor does the system itself require training to tune itself to a user's specific movements; instead, it continuously evaluates user gestures against a set of gesture *profiles*, which are encoded with the characteristic features of particular types of gestures, and uses the best-matching profile to extract information such as beat, tempo, size, and center from

the user's gestures.

We begin with a more detailed description of the scope and requirements for *conga*, followed by a quick survey of existing work in the area of conducting gesture recognition. Then, we describe our design of *conga*, and provide some implementation-specific details and challenges. We conclude with a discussion of some preliminary results obtained by testing *conga* with users.

2. REQUIREMENTS AND SCOPE

Our target user group for this work is museum visitors, and thus, one of our primary goals was to build a gesture recognition system that works for a wide spectrum of users. We also wanted to accommodate people with a wide variety of musical/conducting knowledge. This led to requirements that *conga* be able to:

- recognize gestures from a user without any prior training (either for the user or for the system).
- recognize a variety of gestures to accommodate different types of conducting styles.

One of our goals was also to design *conga* as a reusable component of a larger system that requires gesture recognition; thus, we also required *conga* to:

- integrate well with a computationally-expensive rendering engine for digital audio and video.
- not depend on the specific characteristics of any particular input device.

While conducting is an activity that typically involves the entire body [11], it is generally agreed that the most important information is communicated through the hands [6, 17]. Since we also intended *conga* for use in a public exhibit, we have thus far limited our gesture analysis with *conga* to input from the user's dominant hand. The output of the gesture analysis consists of four parameters: rate (tempo), position (beat), size (volume), and center (instrument emphasis). It is important to note, however, that the design of *conga* itself does not place any restrictions on the types of inputs or outputs, although we leave the implementation of such extensions for future work.

3. RELATED WORK

Gesture-controlled conducting systems have a long history in computer music research. Mathews' early work on the *Radio Baton* [12], which triggers a beat when the baton goes below a certain vertical position, has inspired a number of researchers to study conducting as an interface to computer music systems.

Ilmonen and Takala's *DIVA* system [5] features a conductor follower that is capable of classifying and predicting beats, and even sub-beats, to control tempo and dynamics. The system uses artificial neural networks, and needs to be trained with user data prior to use.

Usa and Mochida's *Multi-modal Conducting Simulator* [18] analyzes two-dimensional accelerometer data using Hidden Markov Models and fuzzy logic to detect beats in gestures. The system features beat recognition rates of 98.95–99.74%, although it also needs to be trained with sample data sets prior to use.

Murphy *et al.*'s work on conducting audio files uses computer vision techniques to track tempo and volume of conducting gestures [14]. Users' movements are fitted to one of several possible conducting *templates*, described in [13]. While the system does not require any training, the user must be familiar with the gesture templates. Murphy used a combination of C code and EyesWeb [3], a library for gesture processing.

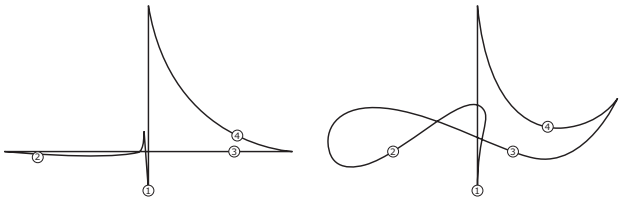

Figure 2: Beat patterns for the four-beat neutral-legato (left) and expressive-legato (right), as described by Rudolf. The numbers indicate where beats are marked in the gestures.

Marrin's *Conductor's Jacket* [11] collects data from sensors along the arms and upper torso, measuring parameters such as muscle tension and respiration. She was primarily interested in mapping expressive features to sections in the music score, rather than obtaining measurements on how movements map to rhythm and beats. In her later collaboration with us on *You're the Conductor* [9], she developed a gesture recognition system that mapped gesture velocity and size to music tempo and dynamics. Her systems were built using LabVIEW [15], a graphical development software for measurement and control systems.

Kolesnik's work also uses Hidden Markov Models for recognizing conducting gestures [6], although the focus of this work was on expressive gestures with the off-hand rather than beat recognition with the dominant hand. His conducting system was built using a combination of EyesWeb and Max/MSP [16].

Our system is thus unique in the following ways:

- the system does not need to be trained prior to use, unlike those that use artificial neural networks and Hidden Markov Models.
- users are not required to learn or be proficient with specific gestures before using the system.
- the system interprets multiple types of gestures, allowing it to respond to the precise gestures of a conductor's four-beat conducting pattern as well as the potentially erratic movements of a child.

4. DESIGN

The design of *conga* is inspired by Max Rudolf's work on the grammar of conducting [17]. In his book, he models conducting gestures as two-dimensional beat patterns traced by the tip of a baton held by the conductor's right hand (see Figure 2). Conducting, then, is composed of repeating cycles of these patterns (assuming the user keeps to the same beat pattern), with beats corresponding to specific points in a cycle. By analyzing certain features of the baton's trajectory over time, such as trajectory shape or baton movement direction, we can identify both the specific pattern, and the position inside the pattern, as it is traced by the user.

Unlike Murphy's work on interpreting conductors' beat patterns [13], we do not try to fit the user's baton trajectory to scaled and translated versions of the patterns shown in Figure 2; as a majority of our target user base are not proficient conductors, such a scheme would most probably not work very well for them; in fact, we have found in previous work that even after *explicitly* instructing users to conduct in an up-down fashion, the resulting gestures are still surprisingly diverse [10]. Murphy also makes use of the dynamics encoded in the music that the user is conducting to differentiate between unintentional deviation from the pattern and intentional deviation to change dynamics; the ability to make this distinction requires one to assume that the user is already familiar with the music (an assumption we are unable to make).

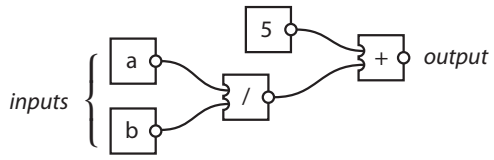

Figure 3: A basic *conga* graph to compute $5 + \frac{a}{b}$. The addition triggers a division, which then in turn pulls data from the inputs a and b.

Our general approach is to instead identify specific characteristics (*features*) in various types of gestures, such as turning points in the baton position with a certain direction or speed. These features are encoded into gesture *profiles*, and the features are triggered in sequence as the user moves the baton in a specific pattern. The advantage of this approach is that the system does not require the user to perform the gesture too exactly; as long as the specific features of the gesture can be identified in the resulting movements, the overall shape of the gesture is unimportant.

conga, as a software framework, allows a developer to work at several layers of abstraction; at the most basic level, it provides a library of feature detectors. These feature detectors can then be linked together into a more complex graph to identify specific gesture profiles, and to date, we have encoded three types of gesture profiles into *conga*, with increasing levels of complexity: wiggle (for erratic movements), up-down (for an inexperienced conductor, but one who moves predictably), and the four-beat neutral-legato (for the more experienced conductor). Finally, we have developed a profile selector that evaluates which of these profiles best matches the user's baton movements at any given time, and returns the results from that profile.

4.1 Feature Detectors

conga's library of feature detectors offers basic building blocks that provide a specific function; for example a bounce detector may detect a change in the baton's direction. Each feature detector *node* has one or more input ports and at least one output port. It takes, as input, a continuous stream of data (e.g., two-dimensional position of a baton). The output is a "trigger", a Boolean value that is true when the feature is detected, and false otherwise. There may be other outputs from the feature detector, so that any nodes that use the output from the feature detector can obtain more information regarding what caused the feature detector to trigger. Other types of nodes also exist to manipulate data, such as rotating the data about an axis, applying various types of filters, etc.

These nodes are connected into directed, acyclic graphs. The graph is evaluated using a pull model, where the output requests data which then pulls on its input nodes to perform the necessary computation (see Figure 3).

This graph-based approach has been used successfully in a number of existing frameworks, including LabVIEW, EyesWeb, and Max/MSP. While it would have been possible to build *conga* as a layer on top of one of these systems, we decided against such a solution after evaluating each of these three systems. *conga* was envisioned from the beginning as a component of a larger system for conducting digital audio and video streams running on Mac OS X [7]; of the three aforementioned frameworks, only Max/MSP runs on the Mac, but, unfortunately, Max/MSP does not provide all of the basic building blocks that we needed to implement *conga*. An alternative would be to use a two-machine solution, such as in [6, 1, 9], although we have learned from prior experience that such setups are awkward to maintain in a museum setting. Nevertheless, we feel these are implementation-specific details, and we emphasize that *conga*'s contribution to the research community is not just the feature detector framework, but our design of a system for conducting gesture analysis based on such a framework.

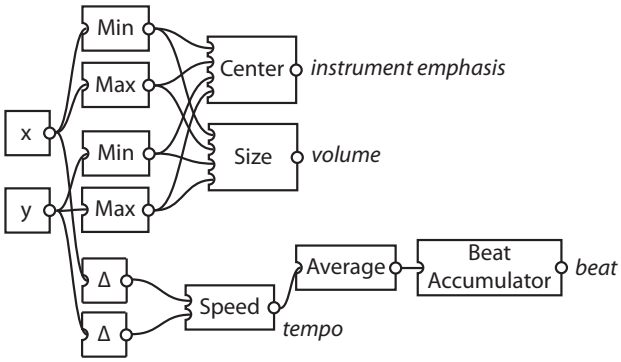

Figure 4: The *conga* graph for the Wiggle gesture profile. The gesture speed determines tempo, gesture size determines volume, and the gesture's geometric center determines instrument emphasis.

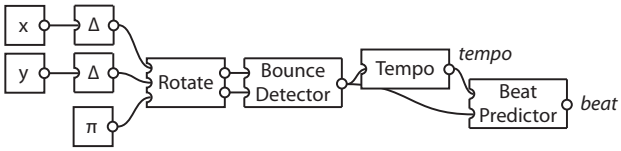

Figure 5: The *conga* graph for the Up-Down gesture profile. The downwards turning points of the gestures correspond to beats; a beat predictor beat values in between these values.

Further details of the feature detector framework and types of nodes it provides are given in [4]; we will discuss only the feature detectors of relevant interest to our discussion here.

The next three subsections describe the three profiles that we have built for *conga* using our feature detector library.

4.2 Wiggle Profile

Figure 4 shows the *conga* graph for the Wiggle profile, which is the most fundamental of the three gesture profiles that *conga* recognizes. Inspired by Teresa Marrin Nakra's work on *You're the Conductor* [9], gesture speed is mapped to tempo, gesture size is mapped to volume, and the geometric center of the gesture determines instrument emphasis (see Figure 4). *conga* falls back to this profile when it cannot use any other means to interpret the user's gestures.

The "x" and "y" nodes hold time-stamped positional data from the baton that has been preprocessed to remove noise. *conga* assumes the origin is at the lower left of the coordinate system. From there, "min" and "max" nodes store the most recent minimum and maximum values of the baton position; these are then used to determine the gesture size and center.

The gesture speed is computed by taking a numerical time derivative of the baton position, followed by a moving average of this derivative. Since the gestures themselves are not synchronized to the music beat, a numeric integral of the speed is used to arbitrarily derive beat information from the gesture speed.

4.3 Up-Down Profile

The Up-Down profile tracks the vertical movement of the user to determine beat and tempo (the method for deriving gesture size and geometric center remain the same as Wiggle, and will not be repeated here). Figure 5 shows the *conga* graph for the Up-Down profile.

The primary feature that is detected is the downwards turning

point, using the "bounce detector" node. The bounce detector node takes, as input, the current velocity of the baton, and looks for a positive to negative zero crossing in the y component of the velocity (i.e., an upside-down "U" shape). Since such a detector would normally track the *upwards* turning point, the data from the baton must first be rotated by 180 degrees. To prevent false triggers, the bounce detector imposes a criterion that the magnitude of the vertical movement over the last few samples be some multiple of the magnitude of the horizontal movement (set as optional parameters in the bounce detector node).

The triggers sent by the bounce detector mark whole beats, and so the tempo can be derived by taking the numerical time derivative of these beat positions over time. This tempo is then used to predict the current fractional beat value until the next trigger occurs. If r is the current tempo in beats per minute, and t_0 is the time of beat b_0 in seconds, then our predicted fractional beat value b for time t is computed using $b = b_0 + \frac{r}{60}(t - t_0)$. We also impose the additional constraint that $b < b_0 + 1$ until the next trigger occurs, to ensure that beats are always monotonically increasing.

We found this simple beat prediction algorithm to work well for estimating the fractional beat values between beats in early prototypes of *conga*. While the beat prediction could be improved if we detected more features in the gesture (e.g., detecting the upper turning point to mark the halfway point into the beat, in addition to the lower turning point), doing so would also place more constraints on the types of movements that would fit the profile. For example, we found that many users naturally tend to conduct "pendulum-style", rather than in strictly vertical up-down movements.

4.4 Four-Beat Neutral-Legato Profile

The Four-Beat Neutral-Legato profile is the most complex, and unsurprisingly, the most challenging beat pattern to detect. Multiple features are detected in parallel, which then drive a probabilistic state machine to track where in the four-beat pattern the user currently is at (see Figure 6).

The features that are detected are: the downwards turning point at beat 1; the upper turning point just after beat 1; the change in horizontal direction just after beat 2; the change in horizontal direction just after beat 3; and the upper turning point after beat 4. Note that the features detected do not necessarily correspond to the beats themselves (see Figure 6).

The first and third features are very distinct sharp turns, and so the bounce detector is again used to track these features. The second feature tends to be more subtle, and thus we look only for a zero crossing in the baton's vertical velocity at that point, without the additional constraint that the bounce detector imposes, as described earlier. Finally, the fourth and fifth features also have a softer curvature, and are also tracked with a zero crossing node. Since zero crossing nodes trigger on both positive to negative, and negative to positive transitions, the undesired trigger is filtered out before sending it to the state node. The state machine node tracks the progress through the beat cycle; it also detects and compensates for missed or false beats using a probability estimation based on the current tempo and time in which the last trigger was received. For example, if we are currently in state 4, and the state machine receives a trigger for state 1, it checks to see how much time has elapsed, and together with the current tempo, guesses what the correct state should be. If it appears that the feature for state 5 was just simply not detected, the state machine will jump directly to state 1. Otherwise, it will assume the trigger for state 1 was simply a falsely detected trigger and ignore it.

The state machine node also acts as a beat predictor; however, unlike the beat predictor in the Up-Down profile, which receives whole beat information and predicts beat values in between the whole beats, the state machine receives *fractional* beat information – this is to compensate for the phase shift between the beats and features in the gesture cycle. For the four-beat pattern, beats 1 to 4 are at 0, 0.25, 0.5 and 0.75 (percentage of one whole cycle), respectively, while the features occur at values of 0, 0.12, 0.31, and 0.63 (see Figure 6).

```
CONGANode *a, *b, *div, *five, *plus;

// Inputs nodes.  Their values will be set externally.
a = [[CONGAPassiveValueNode alloc] initWithTime:0.0f];
b = [[CONGAPassiveValueNode alloc] initWithTime:0.0f];

// Division node.
div = [[CONGADivisionNode alloc] initWithTime:0.0f];
[div setInputPorts:[NSArray arrayWithObjects:a, b]];

// A node with a constant value.
five = [[CONGAPassiveValueNode alloc] initWithTime:0.0f];
[five setValue:5.0f];

// Addition node.
plus = [[CONGAAdditionNode alloc] initWithTime:0.0f];
[plus setInptPorts[NSArray arrayWithObjects:five, div]];
```

Figure 7: The source code corresponding to the basic *conga* graph shown in Figure 3, which computes $5 + \frac{a}{b}$.

4.5 Profile Selection

The three gesture profiles described above run concurrently in *conga*, and the final step in interpreting the user's gestures is a profile selection scheme that decides which of the profiles is returning the most reasonable data. Our algorithm for performing this selection is based on the assumption that the user does not make erratic changes to the tempo; our informal observations of users using our prior systems have confirmed that users moving in an up-down gesture, or a four-beat neutral-legato pattern must exert considerable effort to make relatively sudden changes to the conducting pattern, and thus, the conducting pattern is usually quite regular.

At each regular update cycle, each of the profile graphs is evaluated to determine the current beat. A threshold value is computed based on the standard deviation of the last four calculated beat values, and a confidence value returned by the beat predictor for each of the profile graphs. If this value falls below a certain threshold, we conclude that the profile is returning a sensible result. Profiles are also given a precedence order, so that if more than one profile falls below the given threshold, the one with the highest precedence wins. Our order of precedence from highest to lowest is: four-beat neutral-legato, up-down, and wiggle.

5. IMPLEMENTATION

conga was implemented using the Objective-C programming language under Mac OS 10.4 with the support of Apple's AppKit libraries. Nodes and graphs are created programmatically rather than graphically, a departure from systems such as LabVIEW, EyesWeb, and Max/MSP (see Figure 7). While it is possible to build a graphical editor to create *conga* graphs, such an editor was beyond the scope of this work. *conga* builds into a standard Mac OS X framework, making it easy to include it as part of a larger system, as we have done with *Maestro!*, our latest interactive conducting system [8].

conga graphs are evaluated at 9 millisecond intervals; computation occurs on a separate, fixed-priority thread of execution.

We tested early prototypes of *conga* with a Wacom tablet. Our current implementation uses a Buchla Lightning II infrared baton system [2] as the input device. While the design of *conga* itself is device agnostic, the specific characteristics of the Buchla did

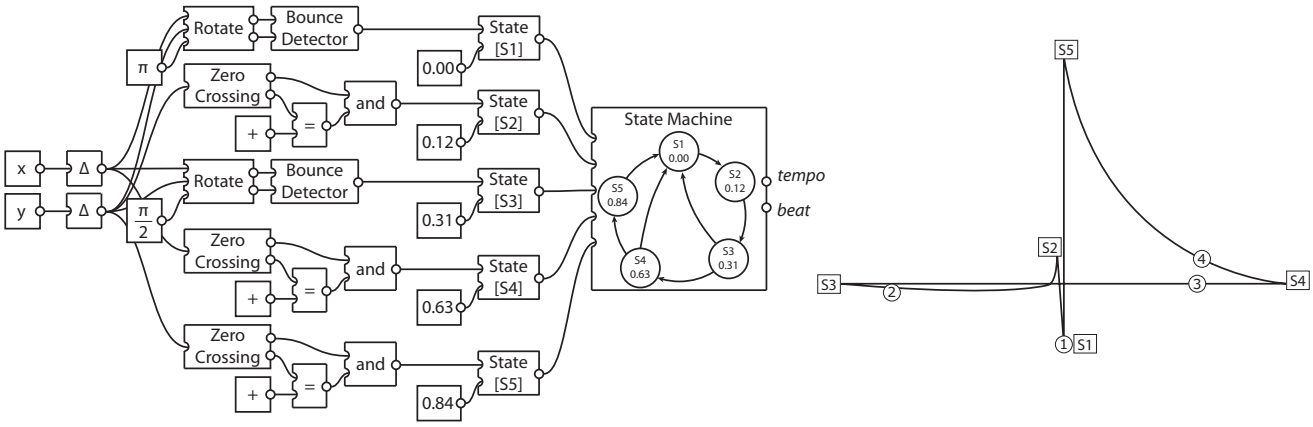

Figure 6: The left figure shows the *conga* graph for the Four-Beat Neutral-Legato gesture profile. Five features are detected, which are used to trigger the progress of a state machine that also acts as a beat predictor. The input to the state machine is the current progress (0 to 1) of the baton as it moves through one complete cycle of the gesture, starting at the first beat. The right figure shows the corresponding beat pattern that is tracked; numbered circles indicate beats, squared labels indicate the features that are tracked and the state that they correspond to.

Table 1: Summary of latency results for the Up-Down profile.

User	Avg Tempo [bpm]	Latency [ms]		
		Min	Max	Avg
1	54	36	117	96
2	58	63	117	86
3	96	81	144	113
4	114	81	144	118
5	130	108	135	122

Table 2: Summary of latency results for the Four-Beat Neutral-Legato profile.

User	Avg Tempo [bpm]	Latency [ms]		
		Min	Max	Avg
1	104	27	675	175
2	96	72	666	203
3	98	18	225	107

present some challenges during implementation. For example, data from a Wacom tablet is relatively high resolution and noise-free, compared to the Buchla Lightning II, which has a resolution of only 128 in both width and height. Data from the Buchla can also be quite noisy, and we experimented with different types of filtering to compensate. Based on these experiments, we found a combination of hysteresis filtering and a 32 point Hanning filter to denoise the data gives the best results. Unfortunately, this Hanning filter also adds between 4 to 10 samples of latency to the overall system (36 to 90 ms), and we are looking into alternative methods to reduce this latency without compromising overall accuracy.

6. EVALUATION

We conducted some preliminary testing with users to evaluate *conga*'s accuracy and response. We asked five users (four male, one female) to conduct using up-down movements, and three users (all male) to conduct using the four-beat neutral-legato pattern. The users conducted for approximately 30 seconds each. The three users conducting the four-beat pattern were already somewhat proficient with the gesture prior to the experiment.

The system starts by default using the Wiggle gesture profile; for all five users, the system switched to the Up-Down profile within the first two beats. After that, *conga* did not falsely detect any beats, nor miss any beats, in the user's gestures (100% recognition rate). We also measured *conga*'s overall latency by measuring the time difference between when the user marks a beat, and when it is detected by *conga*; the results are summarized in Table 1.

Since the smoothing that we apply to the Buchla baton data introduces an average 63 ms delay, we estimate *conga*'s latency to be between 23 and 59 ms. There also appears to be a correlation between tempo and latency, although more data points would be required to make a conclusive statement.

For the Four-Beat Neutral-Legato pattern, we found that for one user, *conga* fell back to the Up-Down profile 8% of the time, and failed to detect his beats 6% of the time. For the other two users, *conga* stayed in the Four-Beat profile 100% of the time, and did not fail to detect any of their beats. The latency results are summarized in Table 2.

The maximum latencies for users 1 and 2 were particularly high; a closer analysis of the data showed that these high latencies occurred consistently on beat 3, and sometimes for beat 4. One possible explanation is that the users' unfamiliarity with the four-beat gesture confused the beat predictor, resulting in *conga* behaving unpredictably. For user 3, who is the most familiar with the gesture, *conga* fared significantly better. Again, we believe that more data points and detailed analysis would be required to make a conclusive statement.

7. FUTURE WORK

We have identified a number of areas that we are actively pursuing to further the development of *conga*:

More gesture profiles: We have currently implemented three gesture profiles in *conga*, which already illustrate the capabilities and potential for the framework. However, only one of these is actually a real conducting gesture, and thus we would like to incorporate more professional conducting styles, such as the four-beat expressive-legato pattern shown in Figure 2.

Improved profile selector: As we incorporate more gesture profiles, we will naturally have to improve the profile selector as well. For example, the four-beat neutral-legato and the four-beat full-staccato have very similar shapes, but their placement of the beat is different, as is the way in which they are executed.

Lower latency: While the current latency introduced by *conga*

is acceptable for non-professionals, professionals will find the latency much more disturbing. One way to reduce the latency is to implement a better beat predictor, especially for the four-beat profile. Another method to reduce latency is to realize that *conga* can only detect a feature *after* it has occurred; thus, by the time the trigger is sent, we are already at some future point in time. *conga* nonetheless reports the feature as having triggered "now", and by compensating for this time delay from when the feature actually happened to when it is detected in the beat predictor, we can reduce the perceived latency further. Again, this would require a more sophisticated beat predictor.

Graphical *conga* editor: *conga* graphs are currently created programmatically rather than graphically, like in Max/MSP. A graphical editor for creating *conga* graphs would make our framework more approachable to a wider range of potential developers.

8. CONCLUSIONS

We presented *conga*, an analysis framework for conducting gestures. *conga* distinguishes itself from current approaches to conducting gesture recognition in that it uses a feature detector approach, which allows the user's data to be fitted to multiple gesture profiles. These gesture profiles represent the key characteristics of a particular beating pattern. The advantage of this approach is that *conga* does not need to be trained with sample data sets, nor does it require users to conduct in specific patterns. As long as their movements trigger the features of a particular profile, the precision by which they are executed is unimportant. A profile selector decides which profile best matches the user's movements, in order to maximize the user's communication bandwidth with the virtual orchestra. We showed our design for three gesture profiles of varying quality and difficulty for the user: four-beat neutral-legato, up-down, and wiggle. Our preliminary evaluation of *conga* showed that it has a remarkably high beat recognition rate, although the latency can be quite high, making our current implementation a little premature for professional use.

Nonetheless, *conga* is both a significant improvement over our previous work, and a novel approach to a well-studied problem, and we hope to continue its development to further advance conducting as an interface to computer music.

9. ACKNOWLEDGEMENTS

The authors would like to thank Teresa Marrin Nakra for early pointers and advice on this project, and Saskia Dedenbach for her assistance with the preliminary user evaluation of *conga*. Work on *conga* was sponsored by the Betty Brinn Children's Museum in Milwaukee, USA as part of the *Maestro!* interactive conducting exhibit.

10. REFERENCES

[1] J. Borchers, E. Lee, W. Samminger, and M. Mühlhäuser. Personal Orchestra: A real-time audio/video system for interactive conducting. *ACM Multimedia Systems Journal Special Issue on Multimedia Software Engineering*, 9(5):458–465, March 2004. Errata published in ACM Multimedia Systems Journal 9(6):594.

[2] D. Buchla. Lightning II MIDI controller. http://www.buchla.com/.

[3] A. Camurri, B. Mazzarino, and G. Volpe. Analysis of expressive gesture: The EyesWeb expressive gesture processing library. In *Gesture Workshop 2003*, volume 2915 of *Lecture Notes in Computer Science*, Genova, 2003. Springer.

[4] I. Grüll. conga: A conducting gesture analysis framework. Diploma Thesis, University of Ulm, April 2005.

[5] T. Ilmonen and T. Takala. Conductor following with artificial neural networks. In *Proceedings of the ICMC 1999 International Computer Music Conference*, pages 367–370, Beijing, October 1999. ICMA.

[6] P. Kolesnik. Conducting gesture recognition, analysis and performance system. Master's thesis, McGill University, June 2004.

[7] E. Lee, T. Karrer, and J. Borchers. Toward a framework for interactive systems to conduct digital audio and video streams. *Computer Music Journal*, 30(1):21–36, 2006.

[8] E. Lee, H. Kiel, S. Dedenbach, I. Grüll, T. Karrer, M. Wolf, and J. Borchers. iSymphony: An adaptive interactive orchestral conducting system for conducting digital audio and video streams. In *Extended Abstracts of the CHI 2006 Conference on Human Factors in Computing Systems*, Montréal, Canada, April 2006. ACM Press.

[9] E. Lee, T. Marrin Nakra, and J. Borchers. You're the conductor: A realistic interactive conducting system for children. In *Proceedings of the NIME 2004 Conference on New Interfaces for Musical Expression*, pages 68–73, Hamamatsu, Japan, June 2004.

[10] E. Lee, M. Wolf, and J. Borchers. Improving orchestral conducting systems in public spaces: examining the temporal characteristics and conceptual models of conducting gestures. In *Proceedings of the CHI 2005 Conference on Human Factors in Computing Systems*, pages 731–740, Portland, USA, April 2005. ACM Press.

[11] T. Marrin Nakra. *Inside the Conductor's Jacket: Analysis, interpretation and musical synthesis of expressive gesture*. PhD thesis, Massachusetts Institute of Technology, 2000.

[12] M. V. Mathews. *Current Directions in Computer Music Research*, chapter The Conductor Program and Mechanical Baton, pages 263–282. MIT Press, Cambridge, 1991.

[13] D. Murphy. Live interpretation of conductors' beat patterns. In *13th Danish Conference on Pattern Recognition and Image Analysis*, pages 111–120, Copenhagen, 2004.

[14] D. Murphy, T. H. Andersen, and K. Jensen. Conducting audio files via computer vision. In *Gesture Workshop 2003*, volume 2915 of *Lecture Notes in Computer Science*, pages 529–540, Genova, 2003. Springer.

[15] National Instruments. LabVIEW. http://www.ni.com/labview/.

[16] M. Puckette. Max at seventeen. *Computer Music Journal*, 26(4):31–43, 2002.

[17] M. Rudolf. *The Grammar of Conducting: A Comprehensive Guide to Baton Technique and Interpretation*. Schirmer Books, 3rd edition, June 1995.

[18] S. Usa and Y. Mochida. A multi-modal conducting simulator. In *Proceedings of the ICMC 1998 International Computer Music Conference*, pages 25–32. ICMA, 1998.