Sonified Motion Flow Fields as a Means of Musical Expression

Jean-Marc Pelletier

International Academy of Media Arts & Sciences
3-95 Ryoke-cho, Ogaki-shi, Gifu
503-0014, Japan
+81-584-75-6600

Graduate School of Media and Governance,
Keio University
5322 Endo, Fujisawa-shi, Kanagawa
252-8520, Japan
+81-0466-47-5111

jmp@iamas.ac.jp

ABSTRACT
This paper describes a generalized motion-based framework for the generation of large musical control fields from imaging data. The framework is general in the sense that it does not depend on a particular source of sensing data. Real-time images of stage performers, pre-recorded and live video, as well as more exotic data from imaging systems such as thermography, pressure sensor arrays, etc. can be used as a source of control. Feature points are extracted from the candidate images, from which motion vector fields are calculated. After some processing, these motion vectors are mapped individually to sound synthesis parameters. Suitable synthesis techniques include granular and microsonic algorithms, additive synthesis and micro-polyphonic orchestration. Implementation details of this framework is discussed, as well as suitable creative and artistic uses and approaches.

Keywords
Computer vision, control field, image analysis, imaging, mapping, microsound, motion flow, sonification, synthesis

1.INTRODUCTION
Since the pioneering work of Erkki Kurenniemi [17] and David Rokeby [20], a great number of musical interfaces using moving images as a source of control have been put forward. While the simple design of Kurenniemi's *DIMI-O* and Rokeby's *VNS* allowed a great degree of freedom as to what their cameras could shoot, it limited at the same time the variety of output of which they were capable. More recent work, notably using the EyesWeb platform, has focused on extracting higher-level information from images, such as expressive content [3]. These advances allow a richer form of interaction between performers and musical system but to achieve this, they require the assumption that there is a performer in the first place.

Despite severe limitations like poor time-wise resolution and latency, image-based interfaces have important merits. One, highlighted above, is that a great amount of information can be extracted from them. Another important characteristic is that they typically require no physical contact between the sensor and the object. While this is certainly useful for stage performers, it also allows a great degree of freedom as to the nature of the objects being sensed. Smoke, clouds, traffic, crowds and countless other phenomena can only be sensed remotely. Such phenomena exhibit complex natural patterns that may prove interesting when they are translated to music. Nature, after all, has been a continuing source of inspiration for countless artists.

Generating the large numbers of control parameters frequently required to generate complex and organic sound structures is a recurring problem. Granular synthesis typically requires the generation of a large number of grains per second [19] each one having its own set of parameters and additive synthesis requires dozens if not hundreds of amplitude envelopes to be generated simultaneously. Several approaches have been proposed over the years to address this issue. Several of these involve one-to-many mappings or stochastic processes [19] to generate large control fields.

Thus, on one hand, imaging technologies give us access to potentially large amounts of information originating from varied sources that often possess inherent complex structures. On the other hand, sound synthesis algorithms like granular and additive synthesis require a large number of control parameters to create complex sounds. It follows that moving images should be well-suited to control such algorithms.

As has been pointed out in previous research [16][26], motion provides a link between images and sound. Hence, in order to access and sonify the complex structures that can be found in imaging data, a *motion flow field* describing the motion at several key points in the image is called upon. Instead of deriving high-level descriptors from this field, its individual components will be used to control matching components of dense synthesis techniques such as granular and additive synthesis.

2.PREVIOUS WORK
2.1 Sonification
There has been surprisingly little research done towards the automatic sonification of generic video sequences for artistic purposes. Some systems have been proposed for automatically generating soundtracks for existing movies [18] but these are typically not aimed at musicians or composers. Some more artistically-relevant work has been done with still images:

sonArt [30], for instance, is a system for creating music from information contained in still pictures.

Some research has been done from a more scientific perspective towards the sonification of vector fields, which at the mapping stage shares some similarities with the framework presented here.

Funk, Kuwabara and Lyons used an optical flow field in conjunction with face detection and zones to devise a musical interface that can be played with the muscles of the face. [7] Jakovich and Beilharz used a dense optical flow field (one computed at every pixel in the image) to alter the cells of a cellular automaton running a “game of life”, which in turn controlled a granular synthesizer [10].

The most similar research to date is that of Kapur et al. who used motion data from a VICON system to control parameters of various synthesis algorithms [11]. While their direct mapping (for instance using *n* motion vectors to control *n* sinusoids of an additive synthesiser) closely mirrors that of the framework presented here, the VICON system, with its six cameras and physical markers imposes great physical and technological constraints that limit the range of its practical uses. Furthermore, the authors focused their research on human gestures, whereas this research aims towards the use of arbitrary imaging data.

3.FEATURES AND FLOW

3.1 Image Features

Raw images contain a vast amount of information: a single-channel 320 by 240 pixel 8-bit image contains 76,800 bytes, which translates to 2,304,000 bytes per second. By contrast, a stereo 16-bit audio stream at 44.1 kHz yields only 176,400 bytes per second. In order to limit the amount of data available, salient image features must first be identified.

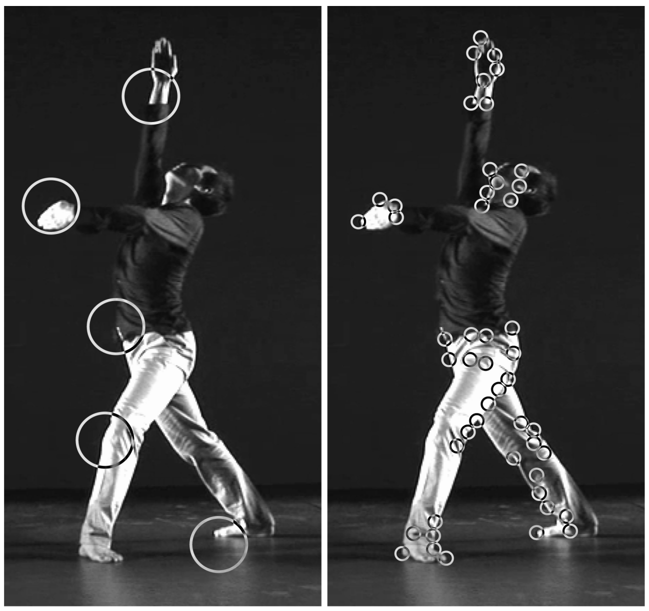

Figure 1: Features computed at two different scales

While the term “feature” is used extensively in the computer vision literature, its definition remains somewhat vague. A feature can be seen as “an interesting image structure that could arise from a corresponding interesting scene structure. Features can be single points such as interest points, curve vertices, image edges, lines or curves or surfaces, etc.” [5] For the purpose of this paper, however, features can be seen as having the following properties: 1) They are local, that is, they have a specific *(x,y)* position. 2) They exist at a given scale. For example, a square can either yield a single large-scale feature or four small scale features at each corner. 3) Features are the local maximum of some image intensity variation metric. The features that match these properties are often referred to as “corners”.

As feature detection is now one of the most fundamental processes in computer vision, several algorithms have been put forward [15]. The Harris detector [8] and its multi-scale variant [14], and the very closely related Shi-Tomasi detector[22], which are based on the partial derivatives of the image intensity, are some of the most commonly used algorithms. Other detectors include the difference of gaussian (DoG), the SUSAN corner detector [25] and the FAST corner detector [21]. If we limit our search to smallest-scale features those occurring in a 9 pixel by 9 pixel neighborhood the machine-learning-based FAST detector is well-suited due to its rapid execution time. However, the Shi-Tomasi detector and the DoG detector may prove better choices in certain situations.

3.2 Motion Flow

As a result of performing feature detection, the image is described as a field of image coordinates (and optionally scale values) corresponding to the features in the image. While it would be possible to use this information as it is, in order to perform more significant mappings, it is important to find out how these features move from frame to frame. The techniques to achieve this can be broadly classified in two categories: feature matching techniques and optical flow-based techniques.

Feature matching techniques [25] involve finding the features in two different frames and matching each feature in one frame to the most similar feature in the second. A number of statistical metrics can be used to measure the similarity of two features based on the values of its pixel neighborhoods. The sum of squared differences and the earth mover distance are two such metrics that perform well [25]. It should be noted that feature matching is an asymmetric process: not all features in both images can be matched into pairs. Some features in the first image will be lost, some in the second will appear and some, outliers, will be mismatched.

Instead of computing features for each frame and finding matching pairs, it is also possible to start with a given set of features and calculate the optical flow at each of these points. The optical flow is a *(Δx,Δy)* vector expressing apparent motion at a point. Common optical flow estimation algorithms can be classified between block-matching methods [1] (which are computationally similar to feature matching algorithms) and differential-based methods such as the Lucas-Kanade algorithm and its more robust pyramidal implementation [2]. Knowing the displacement value, it is possible to compute the new position for every feature at each frame. Because in most cases features will be lost, for example by moving outside the image bounds, and new features are bound to appear, it is necessary to update the feature list in parallel with the optical flow calculation. Hence, the image is processed in this way: optical flow is calculated for existing features and their position is updated → features that could not be successfully tracked are removed from the list → new features are searched in image areas where there are currently no features.

Regardless of the combination of feature detection and tracking algorithm used, the result is conceptually the same: a field of motion vectors either in the format ($x,y,\Delta x,\Delta y$) or ($x,y,s,\Delta x,\Delta y,\Delta s$) where x and y denote position and s denotes scale.

3.3 The Flow Field

In its raw state, the motion vector field computed above is not usable for musical purposes. It will typically contain a certain number of outlier vectors, which will tend to produce jarring and unpredictable results when mapped to sound synthesis parameters. It is thus necessary to run a rather strict filter on the motion field to get rid of these outliers. This filter can be implemented several different ways, including the median flow technique described by Smith et al., in which "each vector in turn is compared with its neighbours. If it points in a similar direction or is a similar, small, length when compared to the 'median flow' in that area, then it is classified as an inlier, otherwise it is discarded as an outlier." [25]

While the motion flow field is expressed using cartesian coordinates and deltas, for later mapping purposes it is useful at this stage to translate at least the displacement values to a polar coordinate system. This yields the following motion vector: $(\alpha,\theta,s,\Delta\alpha,\Delta\theta,\Delta s)$ or in hybrid form: $(x,y,s,\Delta\alpha,\Delta\theta,\Delta s)$. (Here also, the scale dimension is optional.)

It would be possible at this stage to perform further analysis on the motion field. 3D reconstruction algorithms would allow us to recover some form of depth information, either in the form of camera ego-motion or scene structure. More general algorithms can quantify certain types of macroscopic motion such as contractions and expansions, as well as perform object segmentation. However, in this framework, this step is skipped in favor of using the vectors directly.

4.MAPPING

4.1 Time

Depending on the type of synthesis technique used, it may be necessary to process the motion vectors temporally. Using current hardware, frame rates of 30 Hz are typical for camera-based systems. More specialized cameras can image at up to 120 Hz but processing these images in real-time becomes problematic. For additive synthesis and similar generators a control rate as low as 30 Hz may not always be a significant problem, however, the time quantization artifacts that results when motion vectors are used to generate sonic grains are quite noticeable and likely undesirable. The solution to this problem is to smooth out the vector field temporally by delaying each vector individually by some random value normalized between 0 and the projected time until the next frame is processed.

4.2 Space

Since images are inherently spatial, the most natural and motivated mapping possible is that of vector position to sound position. As a matter of fact, the framework outlined in this paper is particularly apt at creating complex spatial trajectories.

The simplest type of spatial mapping is to assign the normalized x values of each vector to the stereo pan position of the sonic component it corresponds to. There is typically more freedom as to how the y axis can be interpreted: in a planar surround playback environment, it can be mapped to the front-back axis although in some setups, it could also be assigned to the up-down axis.

It is also possible to generate positional vectors for the various audio spatialization methods available. The scale dimension (if it is calculated) can be mapped to the z or y axes, with larger features being mapped to closer positions. While this would correctly translate features becoming smaller and larger to sounds moving further and closer, this is a rather naïve mapping that can often lead to undesirable results. Large features do not necessarily correspond to closer objects. In this case, one dimension must be assumed to be constant: the vectors are assumed to move along a plane, though in some cases this is not an accurate representation of the motion of the object being sensed. In some situations it should also be possible to extrapolate the z axis displacement of motion vectors using 3D reconstruction algorithms.

One of the great advantages of using motion vectors for spatialization is that since we know not only where a feature lies but also how fast and in what direction it is moving, it is also possible to use this information to control doppler shift simulations.

4.3 Amplitude

An often convincing approach to controlling amplitude parameters of synthesis components is to assign the length of the displacement vector ($\Delta\alpha$) to amplitude. As $\Delta\alpha$ is directly related to motion velocity, this means that faster objects will sound louder. This relationship is somewhat metaphorically grounded: if the sound is thought to be generated through friction, then indeed faster gestures will produce louder sounds. Hence, the velocity→amplitude mapping is to an extent perceptually motivated.

Overall amplitude is also indirectly controlled via vector density. As has already been mentioned, motion flow over a given area exhibits smooth transitions. This means that areas with a high density of features will tend to produce several similar sound components, which adding up, result in greater overall amplitude.

Lastly, when scale is taken into consideration, it can make sense to use it to control amplitude, with larger features sounding louder. Note that since spatialization, beyond simple linear panning, also affects amplitude it might not always be necessary to control amplitude directly.

4.4 Frequency and Timbre

The most difficult mapping to motivate is that of parameters that affect the pitch and timbre of the sound. That is not to say that such mappings must always be arbitrary, but they largely depend on the nature of the image used, the type of synthesis technique employed and most importantly the intent of the composer or performer.

Even in situations where there is no spatialization, mapping vector displacement in a pseudo-doppler fashion can often result in interesting sound textures. Here, frequency is a function of the displacement relative to the image origin.

In some cases, where the image is to be controlled by a musical performer, simply assigning a given axis value to pitch can be a convincing and easy to understand approach.

Other possible mappings for frequency include distance from origin (α), displacement direction (similar conceptually to accordions and harmonicas) or displacement amplitude (related to the pseudo-doppler approach).

Timbre control in this framework is achieved by the superposition of a great number of sound components and by altering the pitch and amplitude of these components. It is also possible to affect the timbre through the number of features present in the image. This can be done either by changing the input image so that it is less complex or by changing the threshold of the feature detector. A greater number of features directly translates to a greater number of synthesis components.

When using granular processing of recorded sound, it is also possible to control the timbre of the resulting sound by assigning vector position to sound file position. For example, a vector moving from the left edge of the image to the right edge might trigger a sound to be played back from start to end (or vice-versa.)

5. AESTHETIC ISSUES

The framework presented here is meant to be general in nature and adaptable to many different situations. As a performance tool, it offers a natural method of controlling sound clouds and dense textures. Two usage examples highlight an important aesthetic aspect, that of *control gestalt*. This *control gestalt* acts as a binding agent between perceptual groups, or clusters, in both the source image and its sonified form [28].

Recent portable computers often come equipped with a camera mounted somewhere above the screen. With this camera, we can control the parameters of a sound mass generated through additive synthesis. If frequency is a function of the motion vector's position, then head movements towards and away the screen will result in sonic expansion and contraction, as each of the components' frequencies more towards and away from each other. The image features are also contracting and expanding away from each other. However, we do not need to actually measure this change. By virtue of direct sonification, the global characteristics of the motion flow field are expressed in the sound output.

The first use of a system based on this framework by the author occurred in January 2007 for an improvised dance performance held at the Hyōgo Performing Arts Center. The sounds were generated by granular processing of sound files, with each grain's spacial location mapped so that it would sound to the audience as though it was coming from where the dancer was. If there were two dancers standing at opposing sides of the stage, two different sound clusters could be heard in those positions. When a third dancer raced across the stage, yet another sound followed him. However, while it *sounded* as though there were three voices to the music corresponding to each dancer, they were never explicitly identified by the system. The polyphonic aspect of the music was a direct translation of the "polyphonic" nature of the action on stage.

To a limited extent, this form of *control gestalt,* where global control structures implicitly result in similar global output was already present in early zone-based systems like the VNS. However, the greater amount of information contained in motion vector fields coupled with microsonic sound generation means that these relationships occur at a much finer degree.

The performance scenarios outlined above use image analysis in a traditional fashion. Some more exotic approaches include the use of pre-recorded video as a composition tool. Translating visual structures and movements to musical forms can be a very efficient and rewarding method of generating musical material that can be further edited or processed as part of a composition. The motion flow field lends itself especially well to the generation of dense, micro-polyphonic scores.

Returning to the realm of performance, the framework has also been used in conjunction with video content generated in real-time by a VJ, in order to have the visuals linked to part of the music. The possibility of robustly coping with a vast range of possible input structure is a great asset in this scenario.

In a somewhat less musical vein, it is also possible to use systems based on motion flow fields to perform automated foley tasks. While some research has been done in this direction in the past [17], it assumed that the motion of objects in the scene was already known. With some adjustments and proper sound generation algorithms, it is possible to create convincing sound effects, especially considering the spatial *gestalts* outlined above.

6. IMPLEMENTATION

While the general concepts of how the framework can be implemented are presented in earlier sections, the current system implementation will be described in greater detail.

Despite its usefulness, the computation of the motion flow field remains somewhat intensive, limiting both the maximum frame rate, minimum latency, image size and CPU cycles left for sound generation. This is especially problematic since the sound synthesis algorithms used tend to also be rather taxing. In the earliest implementation, the solution was to use two computers with one dedicated to image analysis and the other to sound generation. This solution worked well but it is bulky and costly. In recent years, much attention has been directed towards general processing on graphical processors (GPGPU) [13]. Already a number of libraries, such as OpenVIDIA [6], implement some computer vision task on the GPU, freeing the CPU for other tasks and sometimes yielding improvement in performance of an order of magnitude [23].

The system is currently implemented as an external object for Cycling'74's Jitter system. Standard Jitter functionality is used for image input but all further processing is carried out internally. While this most recent implementation of the framework uses the GPU to perform the image analysis, it is independent of existing software libraries. When GPU processing is available, features are identified using the Shi-Tomasi method and matching is performed using the sum of squared differences. If the computation cannot be performed on the GPU, it reverts to the previous CPU-based algorithm, where features are selected using the FAST method and are then tracked using pyramidal Lucas-Kanade optical flow estimation. It should be noted that since different feature detection and tracking algorithms are used the vector fields generated by GPU and CPU implementations will differ. In practice, however, they will display similar characteristics that will result in very similar sound output.

After the motion flow field has been processed to remove the noise and make adjustments to its coordinates it is sent to the sound synthesizer via OSC [29]. OSC is used to decouple the analysis module from the synthesis module, which is meant to be implemented by the user. Temporal smoothing through random delay can be electively performed prior to output.

7.CONCLUSION

Motion flow fields are not a perfect method of controlling musical parameters. As outlined above, the temporal resolution is comparatively poor. The biggest flaw is probably that feature detection and tracking algorithms are not perfectly robust. When used as an instrument, it is often very difficult to finely control individual components, as one cannot know with certainty where precisely features will be identified in real-world situations. However, motion flow fields are better suited for control of dense masses of sound which in practice alleviates the problem. Its main merits lies in the generality of the approach, the possibility of using natural structures as a source of sonic complexity and the control *gestalts* outlined above.

8.REFERENCES

[1] S. S. Beauchemin and J. L. Barron, "The Computation of Optical Flow," *ACM Computing* Surveys, vol. 27, no. 3, pp. 433-367, 1995

[2] J.-Y. Bouguet, "Pyramidal implementation of the Lucas-Kanade feature tracker," Intel Corporation Microprocessor Research Labs, 1999.

[3] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe, "EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems," *Computer Music Journal*, vol. 24, no. 1, pp. 57-69, Apr. 2000.

[4] M. Cardle, S. Brooks, Z. Bar-Joseph, and P. Robinson, "Sound-by-numbers: motion-driven sound synthesis," in *Proceedings of the 2003 ACM Siggraph/Eurographics Symposium on Computer* Animation, pp. 349-356, 2003.

[5] R. Fischer, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, and E. Trucco, *Dictionary of Computer Vision and Image Processing,* New York: Wiley, 2005.

[6] J. Fung and S. Mann, "OpenVIDIA: parallel GPU computer vision," in *Proceedings of the 13th Annual ACM international Conference on Multimedia,* pp. 849-852, 2005.

[7] M. Funk, K. Kuwabara, and M. J. Lyons, "Sonification of facial actions for musical expression," in *Proceedings of the 2005 Conference on New interfaces For Musical Expression*, pp. 127-131, 2005.

[8] C. G. Harris, "Determination of ego-motion from matched points," in *Proceedings of the 3rd Alvey Vision Conference,* pp. 189-192, 1987.

[9] A. Hunt, M. Wanderley, and R. Kirk, "Towards a Model for Instrumental mapping in Expert Musical Interaction," in *Proceedings of the 2000 International Computer Music Conference*, pp. 209-212, 2000.

[10] J. Jakovich and K. Beilharz, "ParticleTecture: interactive granular soundspaces for architectural design," in *Proceedings of the 2007 international Conference on New interfaces For Musical Expression,* pp. 185-190, 2007.

[11] A. Kapur, G. Tzanetakis, N. Virji-Babul, G. Wang, and P. Cook, "A Framework for Sonification of Vicon Motion Capture Data," in *Proceedings of the 8th International Conference on Digital Audio Effects*, 2005.

[12] E. Klein and O. Staadt, "Sonification of Three-Dimensional Vector Fields," in *Proceedings of the SCS High Performance Computing Symposium,* pp 115-121, 2004.

[13] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, M. Segal, M. Papakipos, and I. Buck, "GPGPU: general-purpose computation on graphics hardware," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing,* p. 208, 2006.

[14] K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision,* vol. 60, no. 1, pp. 63-86, Oct. 2004.

[15] F. Mokhtarian and F. Mohanna, "Performance evaluation of corner detectors using consistency and accuracy measures," *Computer Vision and Image Understanding,* vol. 102, no. 1, pp. 81-94, Apr. 2006.

[16] N. Moody, N. Fells, and N. Bailey, "Ashitaka: an audiovisual instrument," in *Proceedings of the 2007 international Conference on New interfaces For Musical Expression,* pp. 148-153, 2007.

[17] M. Nayak, S. H. Srinivasan, and M. S. Kankanhalli, "Music synthesis for home videos: an analogy based approach," in *Proceedings IEEE Pacific-Rim Conference On Multimedia,* pp. 1556- 1560, 2003.

[18] M. Ojanen, J. Suominen, T. Kallio, and K. Lassfolk, "Design principles and user interfaces of Erkki Kurenniemi's electronic musical instruments of the 1960's and 1970's," in *Proceedings of the 2007 International Conference on New interfaces For Musical Expression,* pp. 88-93, 2007.

[19] C. Roads, *Microsound,* Cambridge, Mass., USA: MIT Press, 2001.

[20] D. Rokeby, "Very Nervous System," Nov. 2000. [Online]. Available: http://homepage.mac.com/davidrokeby/vns.html [Accessed Apr. 15, 2008].

[21] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection." in *Proceedings of the 9th European Conference on Computer Vision,* pp. 430-443, 2006.

[22] J. Shi and C. Tomasi, "Good Features to Track." in Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.

[23] S. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching." presented at *Workshop on Edge Computing Using New Commodity Architectures*, Chapel Hill, North Carolina, USA, May 2006.

[24] P. Smith, D. Sinclair, R. Cipolla, and K. Wood, "Effective Corner Matching," *in Proceedings of the 9th British Machine Vision Conference,* pp. 545–556, 1998.

[25] S. M. Smith and J. M. Brady, "SUSAN—A New Approach to Low Level Image Processing," *International Journal of Computer Vision,* vol. 23, no. 1, pp. 45-78, May 1997.

[26] S. Soto-Faraco and A. Kingstone, "Multisensory Integration of Dynamic Information," *The Handbook of Multisensory Processe*s, G. A. Calvert, C. Spence, and B. E. Stein, Eds. Cambridge, Mass., USA: MIT Press, pp. 49-68, 2004.

[27] B. Truax, "Real-Time Granular Synthesis with a Digital Signal Processor," *Computer Music Journal*, vol. 12, no. 2, pp. 14-26, Summer 1988.

[28] S. Williams "Perceptual Principles in Sound Grouping," in *Auditory Display: Sonification, Audification and Auditory Interfaces*, G. Kramer, ed. Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVIII, Reading MA: Addison Wesley, pp. 95-125, 1994

[29] M. Wright, A. Freed, "Open Sound Control : A New Protocol for Communicating with sound Synthesizers," in *Proceedings of the 1997 International Computer Music Conference,* pp. 101-104, 1997.

[30] W. Yeo, J. Berger, "Application of Image Sonification Methods to Music," in *Proceedings of the 2005 International Computer Music Conference,* 2005.