SoundThimble: A High Resolution Gesture Sonification Framework

Ben Trovato
Authors hidden for review
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

G.K.M. Tobin
Authors hidden for review
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

ABSTRACT

The installation is based on a state-of-the-art Vicon motion capture system, used alongside a Max-based platform to track, interpret and sonify the movement and gestures of a performer in 3D space.

Author Keywords

sonification, motion tracking, gesture spotting, interactive installation, synthesis

ACM Classification

- Applied computing → Sound and music computing
- Computing methodologies → Motion capture
- Human-centered computing → Gestural input
- Human-centered computing → Auditory feedback

1. INTRODUCTION

High resolution three-dimensional motion capture systems are traditionally used for animation in film and games [**?**], as well as for life sciences research [**?**] and engineering applications [**?**]. This technology has long been mined by the NIME community [3, 9], although in many early cases, technological limitations meant that the motion data transmission and the sound generation processes were not simultaneous [3, 7].

The *SoundThimble* project harnesses current motion tracking technology and gesture detection algorithms to develop new modes of sound exploration in an interactive installation context. Our aim is to push beyond the standard paradigms of isolated body motion audification [3, 7] or parameter mapping-based new instruments [9], towards deeper narrative structures coupled with layered arrangement of music patterns.

Our implementation uses a state-of-the-art Vicon motion capture system[1] based on eight Vantage 5-megapixel infrared cameras and two Bonita video cameras. Since the open-source software developed in this project[2] is built around Vicon's Datastream SDK,[3] the platform can be ported to both older and future Vicon-based systems.

In the remainder of the paper, we review relevant existing projects and technology (section 2), we describe the *SoundThimble* concept and development (section 3), we analyse two practical case studies (section 4), and we conclude with a survey of remaining challenges and future perspectives (section 5).

2. RELATED WORK

- interactive / movement sonification examples[5].
- Vicon & related projects
10+ year history of Vicon+sonification
- micro
[12]
- vicon + OSC de la iem.at

The current decade has seen qualitative advances in the interaction between human gesture and sound behaviour [6]. ... [4].

3. PROJECT DESCRIPTION

3.1 Concept

The sound-thimble, as the basic building block of our framework, is based on the concept of *sound object* in the Schaefferian sense, as a clearly delimited sounding unit, open to manipulation, arrangement and composition [10].

Such an entity, once instanced, can retain an ambiguous nature (spatially and acousmatically) or can switch to a more material state (positioned in space and tied to a causal source) [1]. The duality between the latent positioning of the object (which can be inferred from phenomena other than spatial sound reproduction), and the active sound spatialisation, once tied to motion data, becomes an innovative tool for sonic arts through sound sketching, auditory games and other realtime interaction scenarios.

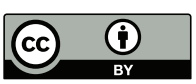
3.1.1 Interaction scenario

SoundThimble can be viewed as both an interactive sound installation and an auditory game, comprising three phases: search, manipulation, arrangement.

The game's narrative starts with a human player attempting to find a sound-thimble (a stationary virtual object, randomly positioned in 3D space), by analysing cues that are constantly shifting in the sonic fabric based on the human's movement relative to the object. Analogously to the traditional game of *Hunt the Thimble* (a.k.a *Hot or Cold*), the space between the human and the virtual object is correlated to sound synthesis and modulation parameters. Briefly, the closer one comes to the object, the more coherent the sound and vice-versa.

Once the object is found, its sonic manifestation gains a richer causal relationship to the human: the player becomes

[1] See `https://www.vicon.com`.
[2] Available at `https://github.com/RVirmoors/viconOSC`.
[3] See `https://www.vicon.com/products/software/datastream-sdk`.



NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.
.

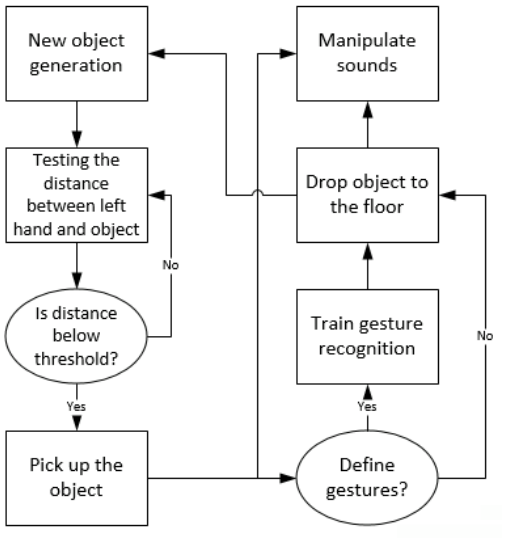


Figure 1: *SoundThimble* interaction workflow.

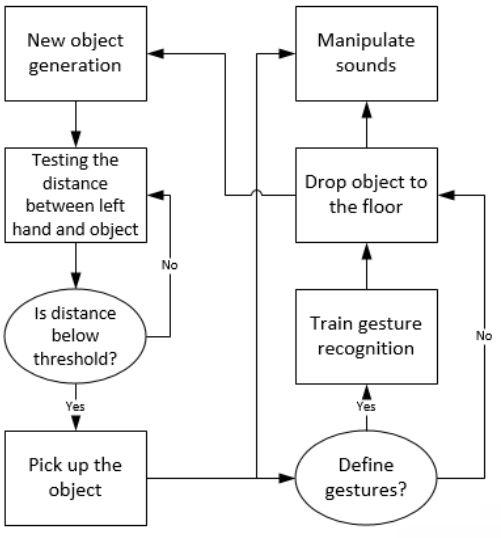


Figure 2: *SoundThimble* framework architecture.

a performer, and is now able to explore the object's sonic palette, and record a number of gestures that can be re-performed later, re-called, and used to trigger or manipulate sonic shifts and events.

Finally, the performer can drop the virtual object to the floor, or discard it by "pushing" it outside of the installation boundaries. This triggers a new object to be randomly generated, while the player retains a degree of control over the initial object via the recorded gestures. Both objects are now in a latent state, with the new one guiding the player's search, and the previous one responding to the learned set of gestures.

This repeating scenario is outlined in Figure 1: objects are randomly generated, the performer finds them, defines gestures and interacts sonically with them before arranging them in a pleasing configuration. With each spawning of a new object or assignment of a new gesture, the game becomes more challenging and complex, but also more flexible and rewarding.

3.1.2 Performance aesthetic

- text Bogdan

3.2 Implementation

=== FIG: Framework Diagram: Senzori, Camere, Nexus, C++ SDK, Max, Speakers ====

The framework architecture diagram is laid out in Figure 2. Three-dimensional sensor data is streamed into the Vicon Nexus software, which is able to reconstruct and label the underlying character skeleton. The gesture recognition and sonification algorithms are programmed in Max[4], which generally receives control data via the OSC[5] protocol. Since Vicon systems do not support OSC out of the box, we used the *oscpack*[6] library to extend the DataStream C++ SDK and send OSC bundles to Max.

3.2.1 Character design

Figure 3 shows a skeletal reconstruction in the Nexus environment. Since our project is intended as a public installation, we pursued a minimal amount of markers, for ease of setup and prototyping. The resulting configuration—sufficient for tracking hand and arm gestures, while ensuring the redundancy needed when one marker is obscured from the cameras—consists in 5 markers: two positioned on the head, one on the forearm, and two on the hand (thumb and index finger).

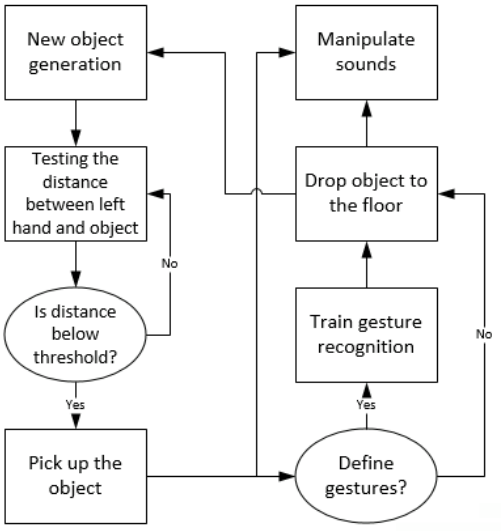


Figure 3: A human performer being tracked in Vicon Nexus.

Each OSC bundle sent through the SDK consists of the following:

- 3D coordinates for the head (averaged from the two head markers);
- 3D coordinates for the hand (averaged from the two hand markers);
- distance between thumb and index finger.

All three items are sent only if non-zero, i.e. for the head and hand coordinates at least one of the respective markers is active, while for the distance computation, both markers need to be visible and correctly labelled. The forearm marker is used only for segment reconstruction, and is not sent via OSC.

[4]Max is a state-of-the-art programming environment for realtime multimedia performance: `http://cycling74.com/`.
[5]OpenSoundControl is a multimedia communication protocol: `http://opensoundcontrol.org/`.
[6]See `http://www.rossbencina.com/code/oscpack`.

The described configuration produces highly stable and responsive inputs into the Max system, with a spatial resolution of 1mm and a latency of ...ms at a frame rate of ...hz.

3.2.2 Object generation & interaction mechanics

The following object-related mechanics are implemented as basic algorithms in Max: generation, detection, dropping.

Object generation is executed randomly within the boundaries of the motion capture field, as defined in the Vicon calibration phase. Detection of the sound-thimble occurs when the distance between hand and object falls below a set threshold. By default this threshold is set at 200mm radial distance; lowering it can make the game considerably more difficult. Once the object is detected, it becomes mobile, its coordinates tracking those of the hand's.

Finally, a simple thresholding of the z-axis (height) value of the hand position serves to put down the object. If the velocity computed on the x or y axes (horizontal plane) is high enough, then the object is pushed in the respective direction, and is able to leave the area of the installation, essentially being removed from the game. At this point, a new object is introduced. The system keeps track of all object coordinates, as they appear and disappear over time.

3.2.3 Gesture recognition

We use the thumb-index finger distance value to enable gesture recording while the two fingers are kept close together. The input data captured into *MuBu* multi-buffer containers [8] consists of pairs of values:

- $\sqrt{\frac{(\Delta x)^2+(\Delta y)^2}{2}}$;
- Δz,

where Δx, Δy, Δz are the respective differences between head and hand coordinates. This feature preprocessing serves two purposes:

Firstly, gestures are recorded based on the position of the hand relative to the head, thus becoming invariable to the performer's absolute *position* within the space. Secondly, by composing the x and y values into a single feature, gestures become invariable to the performer's *orientation* on the horizontal plane. Thus, gestures can be recorded and recalled anywhere within the space, irrespective of the direction the performer is facing.

The input features are fed to one of two gesture recognition algorithms, both part of the *MuBu* package. The first, based on Hierarchical Hidden Markov Models (HHMM), is implemented in the `mubu.hhmm` Max object. HHMMs are a generalization of HMM where each state is considered to be a self-contained probabilistic model [4, 11]. The second is the *Gesture Follower* `gf` Max object, based on a Sequential Monte Carlo inference engine [2].

The first method allows for *gesture spotting*, i.e. it constantly produces likelihood values of a certain gesture being active, together with an approximation of its completion rate. If these are over a certain threshold, the respective gesture is triggered. The second method is more flexible and precise: the detected gesture can be followed at a variable rate or scale, even backwards. The only drawback is that it requires a start trigger, which we send by quickly attaching and separating the thumb and index finger.

Each generated object has a number of gestures associated to it. When an object is dropped to the floor, the classifier is (re)trained with the new data, and consequently gestures can act as on/off switches for a particular sonic behaviour (if they are performed/spotted once at a time), or as continuous controllers (if they are repeated). When several objects exist on the floor, one specific movement might act on one or more objects, depending on which detection likelihoods exceed the threshold.

3.2.4 Sound design

...

In this way, the whole soundscape can be generated in a continuous, organic manner by correlating markers' positions with synthesis parameters.

The interactive experience can be described as having two main paradigms: object finding and object interaction. For object finding we have been experimenting with two straightforward patches in MaxMSP: the first is based on a sawtooth wave that is sent to 8 delays. These output a random signal with a settable range which continuously alter the phase and frequency of the sawtooth iterations. This chorus-inspired algorithm has three controllable parameters: main frequency, range and speed of frequency variation. The farthest someone is from the virtual object the more detuning and phase shifting occurs on each of the eight iterations, while approaching it the effect becomes less pronounced to the point where only slight variations of the signal occur. Also, main frequency is associated with movement in the vertical plane. Further modulation can be used to make the sound more complex. The second patch is somewhat based on the same principle of decorrelation: six sine waves with different frequencies are modulated in amplitude by a random object which generates control waves with a variable degree of complexity (more or less noise-like shapes). As in this case, distance between the performer and the virtual object is associated with the level of randomness and decorrelation: longer distance translates to a higher degree of decorrelation and randomness. What is interesting about this patch is the effect of amplitude modulation (AM) (referinta ce e AM) when the carrier frequency (?) goes beyond 20hz and sidebands occur. By using noise-like carriers, complex sonorities occur with a variable harmonic content. In both cases, the performer tries to find the object by listening to these variations. By correlating small and large variations to its position in the 3d field the performer receives meaningful clues about where the object might be as well as an interesting and engaging soundscape.

An additional granular synthesis (ref) patch that was initially created for object interaction also seems to be effective for object finding. The patch reads short grains from a pre-loaded sound and scatters into "clouds" in either a controlled or random manner. Among the parameters that can be mapped are: grain position, grain size, envelope shape, level of scattering, pitch, stereo width. Not only this, but by using the [pattrstorage] and [pattr] objects we can group multiple parameters' state, save them as presets and interpolate between them. By doing this, we can control an undefined number of parameters by linking only one marker/one gesture to the interpolation amount box. This patch also seems to be effective for finding the object, differentiating the two paradigms by the level of control: less control for object finding and more control for object interaction.

Real-scenario testing and simulations using the Vicon cameras is quite limited because two main reasons: we don't always have access to the space and we would always need a performer that could cope with long hours of code debugging, errors, MaxMSP programming and so on. This is the reason why in order to simulate interactions inside the sound design patches, we also created a basic interface in Jitter that receives data from Vicon Blade via OSC and represents each marker in the 3d space. By using this in-

terface it is possible to move a particular marker anywhere along the XYZ planes, by only using the mouse, and get instant auditory feedback. The downside being that experiments done in virtual space do not always translate well to the real space: calibration by value scaling, experimenting with different function shapes etc. is tedious but absolutely necessary.

So far, all sound-design is based on two channels that can either be routed to multiple pairs of speakers or downmixed to mono and diffused on an arbitrary number of speakers, however multichannel sound is taken into consideration for future improvements.

- Visualisation (jitter)

4. CASE STUDIES

4.1 Interactive Installation

- performance analysis

4.2 Performance

5. CONCLUSIONS AND FUTURE WORK

- Areas of improvement

- Eye tracking?

We have shown ... all software we developed, including... is open source ...

Future work on SoundThimble could/will include: multiple performers, eye tracking, generative visuals, spatial sound, more powerful synthesis and sound manipulation algorithms, different rules added to the narrative of the auditory game.

6. ACKNOWLEDGMENTS

Hidden for review.

7. REFERENCES

[1] BrianKane. *Sound Unseen - Acousmatic Sound in Theory and Practice*. Oxford University Press, New York.

[2] B. Caramiaux, N. Montecchio, A. Tanaka, and F. Bevilacqua. Adaptive gesture recognition with variation estimation for interactive systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 4(4):18, 2015.

[3] C. Dobrian and F. Bevilacqua. Gestural control of music: using the vicon 8 motion capture system. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 161–163. National University of Singapore, 2003.

[4] J. Francoise, N. Schnell, R. Borghesi, and F. Bevilacqua. Probabilistic models for designing motion and sound relationships. *International Conference on New Interfaces for Musical Expression*, pages 287–292, June 2014.

[5] T. Hermann, A. Hunt, and J. G. Neuhoff. *The sonification handbook*. Logos Verlag Berlin, 2011.

[6] K. N. Jorge Solis. *Musical Robots and Interactive Multimodal Systems*. Springer-Verlag Berlin Heidelberg, Berlin, 2011.

[7] A. Kapur, G. Tzanetakis, N. Virji-Babul, G. Wang, and P. R. Cook. A framework for sonification of vicon motion capture data. In *Conference on Digital Audio Effects*, pages 47–52, 2005.

[8] D. S. G. P. R. B. Norbert Schnell, Axel Robel. Mubu and friends assembling tools. *International Computer Music Association*, pages 423–426, August 2009.

[9] K. Nymoen, S. A. v. D. Skogstad, and A. R. Jensenius. Soundsaber-a motion capture instrument. 2011.

[10] P. Schaeffer, G. Reibel, B. Ferreyra, H. Chiarucci, F. Bayle, A. Tanguy, J.-L. Ducarme, J.-F. Pontefract, and J. Schwarz. *Solfège de l'objet sonore*. INA GRM, 1998.

[11] N. T. Shai Fine, Yoram Singer. The hierarchical hidden markov model: Analysis and aplications. *Machine Learning*.

[12] D. Worrall. Understanding the need for micro-gestural inflections in parameter-mapping sonification. Georgia Institute of Technology, 2013.

TO DO: review bib!