The Tactus: a Tangible, Rhythmic Grid Interface Using Found-Objects

Yotam Mann **Jeff Lubow** **Adrian Freed**

Center for New Music and Audio Technologies
Department of Music
University of California at Berkeley
1750 Arch Street, Berkeley CA USA 94720
yotammann@berkeley.edu, jml@dabkitsch.com, adrian@cnmat.berkeley.edu

Abstract

This paper describes the inspiration and implementation of a tactile, tabletop synthesizer/step sequencer. The Tactus is an expandable and inexpensive musical interface for the creation of loop-based music inspired by the *Bubblegum Sequencer* [2]. An optical camera, coupled with a computer running Max/MSP/Jitter can turn almost any matrix-like object into a step sequencer. The empty cells in the gridded object are filled with a fitting, colored object; the placement of which is analogous to adding an instrument or switching on a box in a step sequencer grid. The color and column position of every element in the matrix are used as parameters for a synthesizer while the row position of that element corresponds to the moment within the loop that entry is sounded. The two dimensional array can be positioned anywhere within the camera's visibility. Both the translation and rotation of the physical matrix are assigned to global parameters that affect the music while preserving the color and order of the cells. A rotation of 180 degrees, for example, will not reverse the sequence, but instead change an assigned global parameter.

1. Introduction

Software sequencers, with the advantages of features, price, and upgradeability have all but eliminated their hardware counterparts with the *Tenori-on* [8] being one of a few notable exceptions. A handful of products like the *Bubblegum Sequencer* and *reacTable* [2,5] have strived to add the intuitive, tactile experience of hardware with the functionality of software by using optical cameras to relay data about the object or objects on the work surface to a computer. The advantage over strictly hardware setups is that all of the processing is done on the computer instead of through specialized hardware. The Tactus follows in this line of optically-based tangible sequencers.

The intention of the Tactus is to create an intuitive interface for creating music that does not require expensive, specialized hardware. The nature of the device encourages the user to employ found-objects for the matrix. And since there are no prescribed dimensions for the array, odd sized matrices can be utilized to create unusual metric organizations and deviations from an isochronous pulse. The grid used can range from a wood-lattice to a honeycomb to graph paper. The holes in the grid can be filled with any found-object that would fit in them (Figure 1).

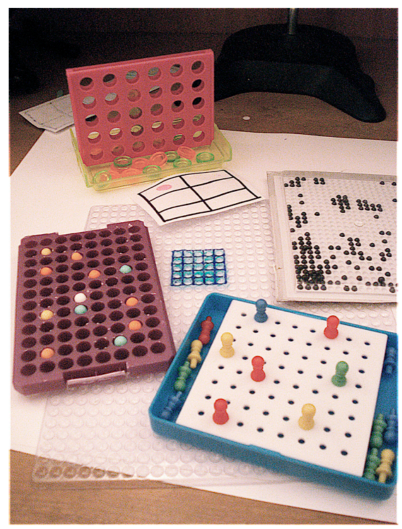

Figure 1. Some found-objects to use with the Tactus including games, magnetic drawing boards, and gridded paper.

The *Bubblegum Sequencer* [2] was the original inspiration for the Tactus. This sequencer uses a video camera underneath a frame with a 16x4 grid of holes that accept colored gumballs and map the color of each ball to a MIDI event. The timing of the MIDI event corresponds to the position of the ball within the grid—each hole is a 1/16th note step, and the 16 columns represent 16 steps of one full 4/4 measure. The Tactus expands on the *Bubblegum Sequencer's* functionality by allowing for any size grid, placing the camera above the surface, and enabling movement anywhere within the sight of the camera. The mobility and interchangeability of the grid allows the performer to prepare a number of different frames and exchange them during performance or to explore one frame's configuration more thoroughly using the rotation and translation parameters.

With these augmentations to the tactile sequencing of the *Bubblegum Sequencer*, the Tactus resembles the *reacTable* [5] where the position, rotation, and distinguishing features of an object are tracked and interpreted by a computer. The Tactus also accepts multiple grids simultaneously, tracking each one's composition and movement like the *reacTable*. The Tactus distinguishes itself from the *reacTable* and *Bubblegum Sequencer* in that it does not use specially-made *tangibles* or structures, making it relatively inexpensive and easily expandable.

2. Hardware and Setup

The Tactus does not need any specialized equipment in order to make music; the requisite hardware is merely a computer, a camera, and any matrix-like object to represent the rhythmic grid. The simplicity and ubiquity of the setup and hardware is inline with the goal of the project: to quickly create a tactile and performable interface for a step sequencer with readily found equipment.

After connecting the computer and camera, the camera is placed so that it faces a blank, well-lit surface. It is important that the surface be cleared of any obstructions and shadows, any of which might be misconstrued by the camera as part of the control matrix.

In this illustration (Figure 2) of the Tactus, the work surface is an 11x17 sheet of white paper with the webcam mounted above it. A webcam is employed because of its decent video quality and affordability. The flexible nature of the Tactus allows for any type of video camera to be situated in different ways—a built-in laptop camera facing a white wall for example.

The physical matrix, in a step sequencer application, has a few requirements. It needs to be rectangular, the inside of the cells must be approximately the same color as the background, and the object must be sufficiently dark in color so that the software can distinguish the elements of the matrix from the background. Any object with evenly spaced holes arranged in a grid would work well. The Tactus tries to make as few prescriptions as possible on the matrix object in order to allow for the greatest range of items that can be employed.

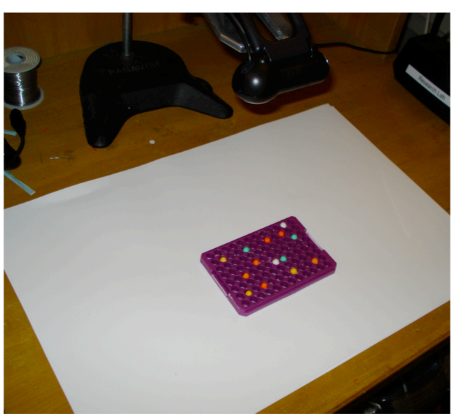

Figure 2. A basic Tactus setup using a webcam, pipette holder and multicolored Tic-Tac confections.

3. Software

The software was created in Cycling '74's graphical programming environment, Max/MSP/Jitter, and relies heavily on the Computer Vision (CV) for Jitter object package [3].

The Tactus software, in essence, performs a translation and downsampling on the input image in order to output the column and row position and color of each entry of the grid. After the software locates the step sequencer matrix in the input video, the CV package's orientation object corrects its rotation so that its vertical and horizontal sides are approximately parallel to the viewing frame (Figure 3). From there, blob detection is employed to locate the physical matrix on a binary image representation of the input image.

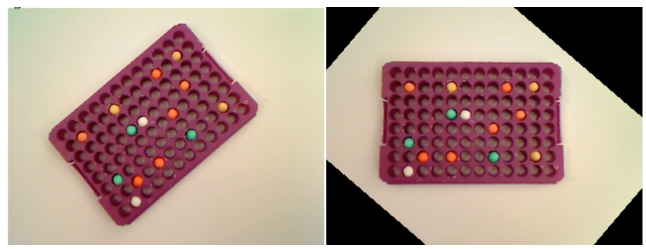

Figure 3. Input after orientation.

3.1 Transformation

A matrix transformation is used to turn the input image of the grid into a rectangular image, because the image of the grid is skewed by the camera lens and viewing angle. The camera input is expressed as a matrix, the basic Jitter data type[1]. This makes it practical to do a matrix transformation from the video input matrix to a matrix representing a regular quadrilateral to correct for skewing [1,7].

A CV object then locates most of the holes of the physical matrix. For this application, it is only crucial that the software reliably finds the corner holes. Using the position of the four corners, along with the user-supplied number of rows and columns, the software computes a homography from the quadrilateral implied by the four corner points to a rectangle represented by a matrix. Once the homography is made, it is possible to then work backwards to find the equivalent of every point on the regular, final matrix in the skewed, input matrix.

[1] Note that the word matrix is used in three different contexts in this section: the physical object, which represents the step sequencer, the input video of the physical matrix which is in the form of a Jitter matrix, and the final output matrix, also a Jitter matrix, which is the result of the matrix transformation.

$$
\begin{aligned}
X &= \frac{ax+by+c}{gx+hy+1} \\
Y &= \frac{dx+ey+f}{gx+hy+1}
\end{aligned} \tag{1}
$$

$$
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -X_1y_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1x_1 & -Y_1y_1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -X_2y_2 \\
0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2x_2 & -Y_2y_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_n & y_n & 1 & 0 & 0 & 0 & -X_nx_n & -X_ny_n \\
0 & 0 & 0 & x_n & y_n & 1 & -Y_nx_n & -Y_ny_n
\end{bmatrix}
\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}
=
\begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ \vdots \\ X_n \\ Y_n \end{bmatrix} \tag{2}
$$

For example, to find the color that would be in position 2,2 in the final matrix, we compute the above equation (1) with x=2, and y=2, and the variables a through h as parameters determined by the linear equation in matrix form (2) where X_1, Y_1, etc. are the corners of the input matrix and x_1, y_1, etc. are the corners of the matrix representing a rectangle. The variables a thought h each affect the perspective in different ways, i.e. rotation, stretch, translation, etc. After they are determined with the linear equation (2), it is possible to work backwards from the regular quadrilateral, the final matrix, to the skewed video of the physical step sequencer grid [1,7]. X and Y from equation (1) are the corresponding coordinates to 2,2 in the input video matrix. The color in position X, Y is then used in the final matrix and for the synthesis portion of the software. The final matrix has the same number of rows and columns as the physical input matrix and its cells contain the same color as the physical matrix (Figure 4).

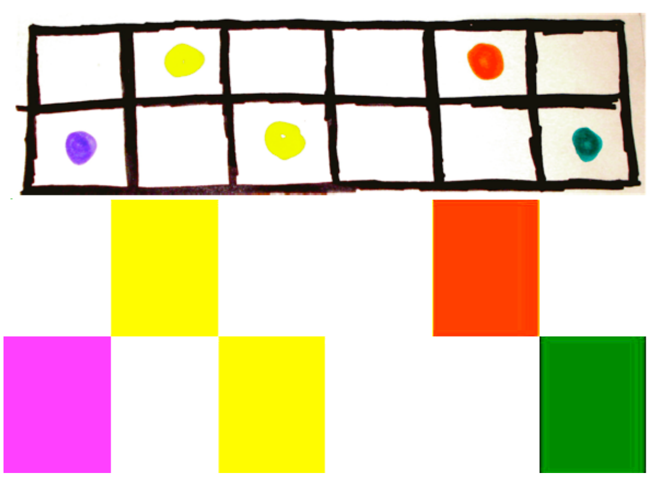

Figure 4. Above, the 6x2 input matrix drawn with markers on paper. Below, the resulting 6x2 Jitter matrix with correspondingly colored cells.

3.2 Synthesis and Sequencing

The final Jitter matrix with each cell's color corresponding to the cells of the physical matrix is fed to the synthesizer. Each row of the matrix drives transients of a unique sound module and the length of the sequence is given by the X dimension of the sequence matrix. In it's current incantation, since there might be more rows in the matrix than available synthesizers, after all of the sound modules have been exhausted, the sound modules are repeated. For example, if there are four sound modules available, a 16x8 matrix would control four sounds (each in two ways) for the duration of 16 steps.

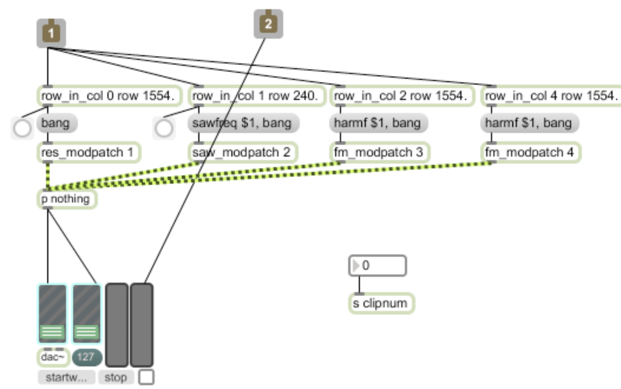

Figure 5. A screenshot of four of the sound modules currently in the Tactus.

The RGB matrix that is input is fed into an algorithm to convert the combined values to luminance. This matrix of luminance values is stored for later use after being converted. To create the sounds, a tempo-based metronome polls a mechanism to update the current step of the sequence. This counter-based column-extraction gets fed and routed to the various tracks based on tagged IDs. Each sound is activated based on the presence or absence of a color within a designated acceptable luminance range. Consequentially, a sound is either generated or left as a rest. The available parameters for a given transient are derived from A) the luminance of the step in the sound-module's sequence, B) the rotation of the physical matrix and C) the translation of said matrix. This is but one possible use of the grid layout of the Tactus.

The Tactus also lends itself well to many of the geometric mappings for intuitive synthesis developed by Ali Momeni and David Wessel [6]. The user can generate and explore a timbre spaces in novel ways. An especially interesting technique for loop-based music outlined in the paper is the probabilistic determination if/how a pattern or sound will occur. The higher the row number, the greater the probability that event will sound, potentially making the loop different through each iteration. Also, rotation can be intuitively mapped to the degree of groove, or *tatum* [4]. Vijay Iyer et al define a *tatum*, after Art Tatum, as the "smallest meaningful subdivision of the main beat". A rotation by 180 degrees would lead the greatest deviation from the regular beat.

4. Music Performance

Performance on the Tactus is fairly intuitive for anyone who has experience with step sequencers. The user can start in a traditional way by placing down the identical color in the same row with equal spacing to produce a steady rhythm and slowly build onto that in subdivisions. Alternatively, one can begin by randomly dropping colored objects on the physical grid—the Tactus lends itself well to creating aleatoric music.

The translation and rotation of the entire grid is also an intuitive way for a user to interact with the instrument. One issue with this level of control is that the user's movements need to be incremented if the intention is to sound at all smooth; this is because, in order to avoid having the software analyze the user's hands as part of the grid or skin pigments as part of the entry's color, analysis is stopped while a hand is detected in the camera's view.

5. Future

There is room for many improvements in the Tactus as it exists in the current iteration. The goal of placing *any* gridded object in front of the camera and instantly making music is close, but not yet in hand. For this to happen, there needs to be greater robustness in the detection of the holes and the colors. For now, many detection problems can be solved by introducing more lighting on the work surface, but this solution is not optimal. In the future, more robust detection will lead to greater accuracy and fewer restrictions placed on the sequencer grid that can be used.

Another step towards this goal is automated row and column counting so that the user can simply place an object in front of the camera and immediately start making music. Currently, the user must manually specify the dimensions of the matrix so that the software knows how many elements to search for. This could be achieved, once again, through better detection. Finally, relating to the synthesizer, one question for the future of the prototype would be how to choose colors to be representative of sounds in order to derive implicit mappings for sonic control.

6. Acknowledgments

The author would like the thank Andy Schmeder and John MacCallum for their technical assistance and advice.

References

[1] A. Criminisi, I. Reid, and A. Zisserman, "A Plane Measuring Device," in *Image and Vision Computing*, v. 17, i. 8, June 1999, pp. 625-634.

[2] H. Hesse and A. McDiarmid, "The Bubblegum Sequencer," in *Proceedings of the Conference on Computer Human Interaction (CHI)*, Apr. 2008. http://www.backin.de/gumball/

[3] IAMIS, "Computer Vision for Jitter", July 2008. [Online]. http://www.iamas.ac.jp/~jovan02/cv/

[4] V. Iyer, M. Wright, D. Wessel, J. Bilmes, "A Novel Representation for Rhythmic Structure," in *International Computer Music Conference*, 1997, pp. 97-100.

[5] M. Kaltenbrunner, S. Jordà, G. Geiger, M. Alonso, "The reacTable: a collaborative musical instrument," in *Proceedings of the Workshop on "Tangible Interaction in Collaborative Environments" (TICE), at the 15th International IEEE Workshops on Enabling Technologies (WETICE),* 2006.

[6] A. Momeni and D. Wessel, "Characterizing and Controlling Musical Material Intuitively with Geometric Models," in *Proceedings of the Conference on New interfaces for Musical Expression (NIME),* 2003, pp. 54-62.

[7] C. R. Wren, "Perspective Transform Estimation." Massachusetts Institute of Technology Media Laboratory, Cambridge, MA, USA, Dec. 2008. [Online]. http://alumni.media.mit.edu/~cwren/interpolator/

[8] Yamaha Global Gateway, "Tenori-on", Jan. 2009. [Online]. http://www.global.yamaha.com/design/tenori-on/