Music Moves

Wim Looman
Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand
wgl18@uclive.ac.nz

Richard Green
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

Abstract

This report details the design and implementation of an "air drum" interactive music program. Various methods for implementing this on consumer hardware utilising a simple webcam are explored. The results from

1. Introduction

The goal of this project is to create an interactive music program that can generate music based by analysing the movement of a human body or hand position. This will be capable of running in real time on a consumer pc utilising a single webcam.

A lot of different methods for producing movement from sound have been researched. Some of these involve using the hands to "shape" the music, for example having a 2d pointer (which could easily be generated from a hand tracking algorithm) that affects the music based on a set of simple gestures [1], [2].

Others attempt to provide a more traditional musical interface, albeit without the actual instrument. This can be a simple interface such as a DJ's deck [3], or a more complex interface like a full piano [4] or guitar [5].

Even more forgo any direct relationship between the users movement and the type of music produced, instead relying on a simple mapping of specific posture to a specific sound to be produced [6].

Others still use a rhythmic approach, the rhythm from a user dancing [7] or conducting [8], [9] is used to affect the music being played.

Whilst the first approach will probably lead to a more novel category of research, it would also require a deep understanding of music theory. Therefore the second approach was the one chosen for investigation.

One instrument that hasn't been widely studied for video based playing is the air drum. It would have to be wondered why; the drum simultaneously provides both a complex interface, the 3d position of the imaginary sticks, and a simple interface, only around 3-10 different drums/cymbals to hit depending on the drum set.

To enable the user to play the air drum just via a normal webcam a robust algorithm for turning their motions into sound will be needed.

2. Implementation

The method decided on is based on the pipelining used in standard Unix systems, this will allow for multi-threading to be utilised in the future. As seen in Figure 1 this system will utilise three processing stages. The first is hand tracking, this will identify the position of the users hands and track them through subsequent frames. Next the hand positions will enter the strike detector, this will determine when the hands hit one of the virtual drums. Finally the sound production stage will take the strikes and output the sound.

2.1. Hand Tracking

There are many possibilities for hand tracking, both marker-based tracking where the user has something like a pair of brightly coloured gloves for the computer to track and marker-less tracking where the computer tracks the users hands without any additional marker [10].

There were two major methods looked at for this system; feature tracking and skin colour masking with blob tracking.

Feature tracking would likely be the most effective tracking method when used with a high quality camera and fast processing speeds. This system was limited to run on a mid-range laptop, however, and the lower quality camera and slow processor produced significant blurring in the images taken, see Figure 2.

Because of this blurring feature detection would be almost useless, the features that get detected would disappear into the noise whenever the hands move too quickly. Even using something like

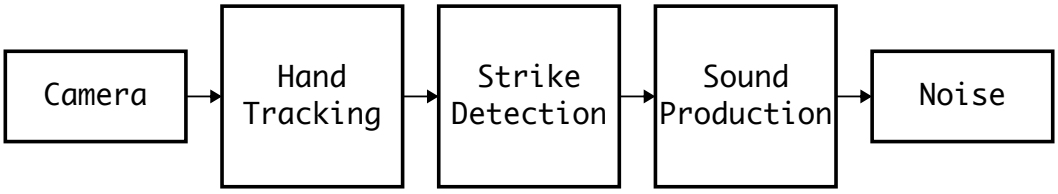


Figure 1. The pipelining approach.

Figure 2. The blurring produced by the low quality camera.

Kalman filtering the absolute destruction of the features would be too much.

Skin colour masking with blob tracking would be more resilient to the blurring occurring with the camera; the blurred hands will move the outsides of the blob, but the centroid will remain in approximately the right position.

2.1.1. Skin colour masking. Again there are various possibilities for the skin colour masking. Some methods that have been employed are Gaussian chromatic color segmentation [11], more complex YCbCr/HSV color space division [12] or just simple linear HSV color space division [13].

The Gaussian chromatic color segmentation and complex YCbCr/HSV color space division provide more accurate results than the simple linear HSV color space division, however they are more complex to implement and designed for the specific datasets used in their studies.

The chosen algorithm was simple HSV colour masking for the reasons listed above. The implementation was a simple range check, if a pixel's HSV value fell within the range (0, 0.12, 0.31) -> (0.08, 0.59, 1.00) then it was counted as skin coloured. The result of this check was stored as a mask with skin-coloured pixels fully white and non-skin coloured pixels fully black.

To make the masking more resilient to noise such as specular reflection spots the colour image was smoothed before the conversion to HSV colour space. This increased the mis-classification of other colours as being skin, but it was felt that the trade-off was worth it in terms of better tracking of real skin.

Once the mask was produced it was dilated then eroded to close up the blobs. Regions such as the lines between fingers on a closed fist had too much shadow to detect as skin coloured so this helped to form the hand into a single blob for the tracking stage.

2.1.2. Blob tracking. Once the skin regions have been detected a simple algorithm that tracks the hands via centroid, size and movement can be used effectively [14]. First the contours of the skin mask are found using the Suzuki85 algorithm [15]. This uses a border following method to find all edges between black and white and returns a set of contours.

These contours then have their spatial and central moments calculated and used to determine the centroid and area covered by the contour. These are used to determine a score for the contour, this is a simple heuristic score proportional to the size of the contour (it is assumed that the hands will be the largest blobs in the frame) and the distance from the contour's centroid to the last centroid for that hand. The score is calculated separately for each contour for both hands and the two highest scoring contours are selected as the new contours representing that hand for this frame.

This will give as a final value the current centroid of each hand, ready to be used for the strike detection.

2.2. Strike Detection

The next step in the pipeline is detecting when the user strikes the virtual drum. Possible options explored for this was training up a neural network that took a few frames of (x,y) locations of the hands to determine when a drum was struck, attempting to derive a hand model based system for detecting the drum strike and using a simple kinematic first and second order derivative detection system [16].

The problem with using a neural network based solution would be the large amount of training required. Building the set of training data would involve recording a lot of video then painstakingly annotating where it was believed the drum strike sound should occur. This would introduce a human error element in that if the annotation of the training set was off the neural network would be incorrectly trained.

Using a hand model would also require a lot of working in deriving a valid model, there is previous work in this area that could be used as a starting point [17], [18], [19], [20].

The simple kinematic system for detecting the strike would be a lot simpler to implement, the hand co-ordinates captured by the hand tracking can be used to directly generate first and second order derivatives. Once these values are found a strike can easily be detected via its kinematic properties, when the hand hits a drum its movement abruptly changes from downwards to upwards. This abrupt change can be approximated by the velocity of the hand following a sigmoid function (*Figure 3a*). Taking the derivative of this function we can see that there is a large spike when this strike happens (*Figure 3b*).

2.3. Sound Output

Two methods were explored for outputting the sound. The first was using OpenAL and manually outputting the sound from the program developed. The second was using MIDI to send the sound output to a synthesiser and let that handle the actual sound generation.

Using OpenAL was very quickly discarded as the effort required was simply too great. The focus of this project was on the computer vision, not the sound production.

For MIDI a very easy to use library known as PortMIDI was found. This allows a stream based approach to MIDI production, it simply needs to be given an output stream to write to and it will send the MIDI commands required to make sound. The synthesiser used is known as Renoise, this was the only soft-synth found that provided a stream to be written to; all the others wanted to be given an input stream that they would then read from and play, which was not supported by PortMIDI.

3. Results

3.1. Hand tracking

The hand tracking was the least reliable part of the system. Under ideal conditions it worked almost perfectly, as long as the hands were the only skin coloured objects and they didn't occlude one another there was no problem tracking them.

However, under non-ideal conditions the tracking deteriorated rapidly. If there was another patch of skin colour larger than one of the hands then the simple blob tracking in use would prefer to follow that instead, and there was no way to take the marker back from the patch. If there were smaller patches of skin colour then the centroid would jump around a lot as the hand joined on to different patches.

3.2. Strike Detection

When the hand tracking was working well the strike detection was very good. After extensive testing the results shown below were found. These results were gathered under near perfect conditions to purely test the strike detection, there was only a few small patches of non skin that was being detected by the hand tracking algorithm.

Test	Strikes	Correct +ve	False +ve
S1 (1 bps)	100	92	13
S2 (2 bps)	100	87	19
F1 (4 bps)	200	184	40
F2 (8 bps)	200	175	60

Key: S1 = Slow single handed, S2 = Slow double handed, F1 = Fast single handed, F2 = Fast double handed.

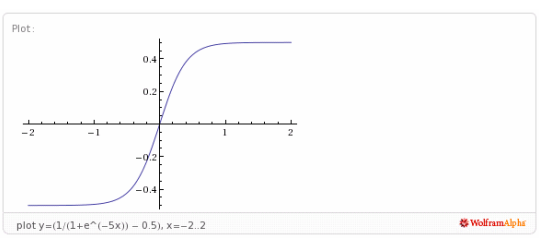
These results show that the strike detection is 87-92% accurate, with a false positive rate of 13-30%. Most of the false positives were occuring as double hits, this could be because of a series of five frames showing the hand moving down, the hand stopping, then the hand heading back up. Both the hand stopping and the hand heading back up could have high enough second derivatives to register as strikes under the current system.

3.3. Sound Playing

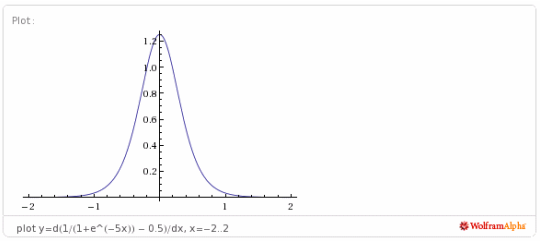
The playing of sounds worked perfectly, thanks to such an easy to use MIDI library. There was no noticeable delay between when the program detected a strike and when the sound from the strike was played.

4. Future Work

Because of the pipelined architecture it will be very simple to replace parts of the system with



(a) Velocity



(b) Acceleration

Figure 3. The approximation of the hands kinematic motions.

different implementations. This means it can make a good testbed for future research into hand tracking and strike detection algorithms.

5. Conclusion

References

[1] K. Jensen, "Aspects of the multiple musical gestures," *Journal on data semantics*, vol. 3902, pp. 140–148, Jan. 2006.

[2] T. Winkler, "Making motion musical: Gesture mapping strategies for interactive computer music," *Computer*, 1995.

[3] B. Yetton and R. Green, "Virtualmix - virtual dj interface using computer vision," 2010, produced for COSC428.

[4] R. I. Godøy, E. Haga, and A. R. Jensenius, "Playing 'air instruments': Mimicry of sound-producing gestures by novices and experts," *Journal on data semantics*, vol. 3881, pp. 256–267, Jan. 2006.

[5] Y. Pan and R. Green, "A mark-less air guitar game through computer vision," 2010, produced for COSC428.

[6] B. Chong and R. Green, "Music moves," 2010, produced for COSC428.

[7] G. Castellano, R. Bresin, A. Camurri, and G. Volpe, "User-centered control of audio and visual expressive feedback by full-body movements," in *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ser. ACII '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 501–510. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74889-2_44

[8] A. Salgian, M. Pfirrmann, and T. Nakra, "Follow the beat? understanding conducting gestures from video," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, N. Paragios, S.-M. Tanveer, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Müller, and T. Malzbender, Eds. Springer Berlin / Heidelberg, 2007, vol. 4841, pp. 414–423. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-76858-6_41

[9] R. Behringer, "Gesture interaction for electronic music performance," in *Proceedings of the 12th international conference on Human-computer interaction: intelligent multimodal interaction environments*, ser. HCI'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 564–572. [Online]. Available: http://portal.acm.org/citation.cfm?id=1769590.1769654

[10] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, December 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[11] H. Chang and U. Robles, "Face detection," May 2000, eE368 Final Project Report. [Online]. Available: http://www-cs-students.stanford.edu/~robles/ee368/main.html

[12] D. Wang, J. Ren, J. Jiang, and S. S. Ipson, "Skin detection from different color spaces for model-based face detection," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*, ser. Communications in Computer and Information Science, D.-S. Huang, D. C. Wunsch, D. S. Levine, and K.-H. Jo, Eds. Springer Berlin Heidelberg, 2008, vol. 15, pp. 487–494. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85930-7_62

[13] A. X. Li, "Hand detection using opencv," March 2009. [Online]. Available: http://www.andol.info/hci/830.htm

[14] L. Tarabella, "Handel, a free-hands gesture recognition system," in *Proceedings of the 2004 Second Internaional Symposium Computer Music Modeling and Retrieval*. University Esbjerg, Denmark, 2004.

[15] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 – 46, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X85900167

[16] D. Popa, G. Simion, V. Gui, and M. Otesteanu, "Real time trajectory based hand gesture recognition," *WSEAS Trans. Info. Sci. and App.*, vol. 5, pp. 532–546, April 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1481952.1481972

[17] F. L. Buczek, E. W. Sinsel, D. S. Gloekler, B. M. Wimer, C. M. Warren, and J. Z. Wu, "Kinematic performance of a six degree-of-freedom hand model (6dhand) for use in occupational biomechanics," *Journal of Biomechanics*, vol. 44, no. 9, pp. 1805 – 1809, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021929011003083

[18] P. Cerveri, N. Lopomo, A. Pedotti, and G. Ferrigno, "Derivation of centers and axes of rotation for wrist and fingers in a hand kinematic model: Methods and reliability results," *Annals of Biomedical Engineering*, vol. 33, no. 3, pp. 402–12, 2005, 756754739. [Online]. Available: http://search.proquest.com/docview/756754739?accountid=14499

[19] C. Kerdvibulvech and H. Saito, "Model-based hand tracking by chamfer distance and adaptive color learning using particle filter," *J. Image Video Process.*, vol. 2009, pp. 6:2–6:2, January 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/724947

[20] S. Cobos, M. Ferre, M. Angel Sanchez-Uran, J. Ortego, and R. Aracil, "Human hand descriptions and gesture recognition for object manipulation." *Comput Methods Biomech Biomed Engin*, 2010.