

---

# Toward Rich, Continuous Representation and Interaction

**Joseph Malloch**

Graphics and Experiential Media  
(GEM) Lab  
Dalhousie University  
Halifax, NS B3H 1W5 Canada  
joseph.malloch@dal.ca

**Abstract**

I present a suggestion for rethinking some common assumptions when designing interactive systems. In brief, I advocate that developers exercise restraint before performing classification or identification steps when processing human input to a system, and consider carefully a) whether classification is necessary or desirable for the interaction; b) whether a continuous representation might be preferable in terms of computational efficiency, user efficiency, or user experience; and finally c) in cases where classification/identification is the appropriate approach, whether it might be useful to augment and enrich the discrete, classified objects with continuous data representations.

**Author Keywords**

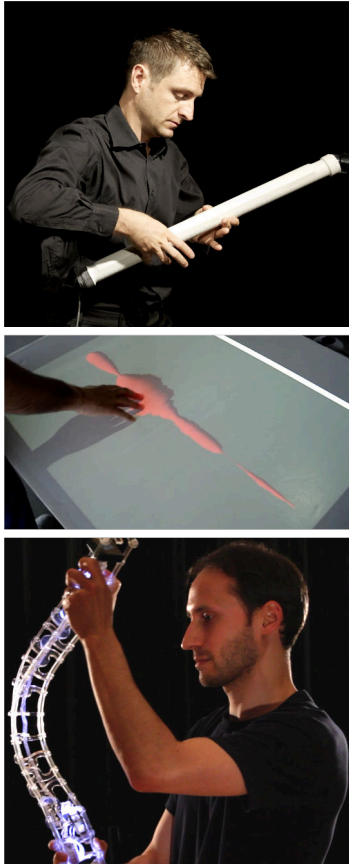
continuous interaction, continuous representation, input devices, gesture input, design principles

**ACM Classification Keywords**

H.5.2 [User Interfaces]: Interaction Styles

**Introduction**

When interacting with modern computer systems, it is common for human input, internal system representations, and generated output to be considered in simple discrete units. Much of our “natural” language communication is in the form of machine-encoded text, and most of the possible



**Figure 1:** Some of the digital musical instruments created by the author. From top: the T-Stick (performer: D. Andrew Stewart, photograph: Vanessa Yaremchuk); the IDMIL interactive table; the Spine.

user actions in the context of a software application are in form of discrete, non-parametric commands displayed as buttons to be pressed or menu items to be selected. Even when the input device used provides fine-grained resolution (such as a laptop touchpad) the input data is typically abstracted into discrete *touch events*—usually represented as ellipses with attached identifiers—before they are made available to software developers (let alone end-users). Machine learning systems mostly focus on classification of input into discrete categories, and even “fuzzy” input modalities such as gesture controls usually still focus on recognising discrete gestures from noisy input rather than leveraging the continuous nature<sup>1</sup> of the raw data.

Hardware drivers and application programming interfaces (APIs) don’t only influence our implementations of interactive systems, they also shape our conceptualisation of the interactions themselves. We should not be content with APIs or input device drivers that arbitrarily prevent access to low-level user input.

In the following sections I will present several examples of this approach to representation and interaction from my own work, and endeavour to show that continuous representations and interactions can be (sometimes surprisingly) interesting and useful. By simply waiting before jumping to recognition or identification of “objects”—and often no additional design or implementation effort—this approach can offer increased redundancy, robustness, and scope for personalisation, expression and rich communication.

<sup>1</sup>In this paper I use the term “continuous” somewhat loosely, and use it to refer to both actually-continuous phenomena, and digitised stream representations of such phenomena provided that they are presented as a sampled stream rather than discrete events. The main topic here concerns perception and system conceptualisation.

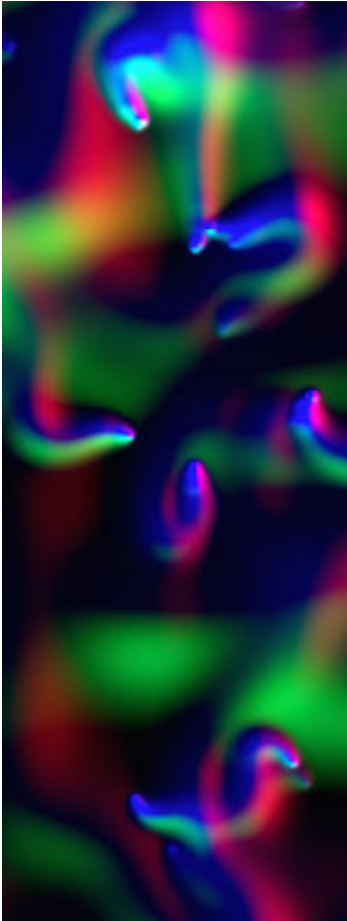
## Continuous Representations

### *Digital Musical Instruments*

Digital Musical Instruments (DMIs) are commonly described as a combination of *input device* and *sound synthesiser* linked by a layer of *mapping* connections [10]. While many commercial DMIs have “keyboard” (piano-like) interfaces with discrete notes, the academic and experimental DMI communities have created a wild array of fascinating instruments and interactive systems. My own contributions to this world of new instruments include those depicted in figure 1; the experience of developing these and other music systems has had an important role in shaping my perspective on human-computer interaction and interaction design. The T-Stick [9] and Spine [5] DMIs (figure 1, top and bottom) are designed for professional musicians and dancers to perform, but as experimental instruments they are also designed for composers and software developers, and rely on open-source distributed connection middleware [7] to publish control signals that can be freely connected to various software synthesisers. This set of control signals form an informational interface of their own, defining the ways that a composer, synth developer, or hacker-musician could make use of the system. While designing this space of controls, we developed a number of ad hoc guidelines:

- always provide access to raw sensor data—don’t assume you know what will be interesting to users,
- also provide access to the “cooked” (conditioned, pre-processed) version of this data, scaled to real units,
- for higher-level extracted features, support and provide multiple parallel (and even redundant) representations to support different users, perspectives, or scenarios.

One of the functions of the connection middleware system *libmapper* is to support publication of large numbers of sig-



**Figure 2:** Distributed software agents connected to the *Influence* environment can move around and read from and write to their local position, while 2D convolution propagates information across the space.

nals without flooding the network with unused data, with the intention that DMI designers could then publish multiple interesting signal representations without worrying about efficiency. Another function is to automatically perform linear scaling between the published ranges of data signals, so that basic compatibility can be achieved without requiring either normalisation or “shoehorning” the representation of a novel DMI into a predefined schema.

#### *IDMIL Interactive Table and Influence*

This *IDMIL Interactive Table* was created in approximately one month, for a one-day installation in a children’s science museum. It was designed to be inexpensive—actually free, since it used only discarded hardware—with an angled mirror mounted above a regular table so that a video projector and a webcam mounted behind the table both “see” the table’s surface. Rather than using traditional video object or blob tracking, video convolution was used to generate a vector field from the webcam input, by which the dynamics of virtual objects or systems projected on the table could be influenced. In some modes the vector field was used to simulate physical dynamics in which virtual particles were attracted to and projected onto objects or body parts on the table. In another mode it animated a control-rate digital waveguide system used for scanned synthesis (figure 1, middle), allowing users to metaphorically “inject” energy into the waveform with their hands.

For a later project exploring emergent phenomena in the context of the media arts, we decided to use this approach again for propagating information between distributed software agents. This had several advantages for our application: first, the N-body solution enabling all agents to “see” all other agents is spread across time, so that it becomes a linear problem related to the number of pixels rather than agents. Although this is a considerable

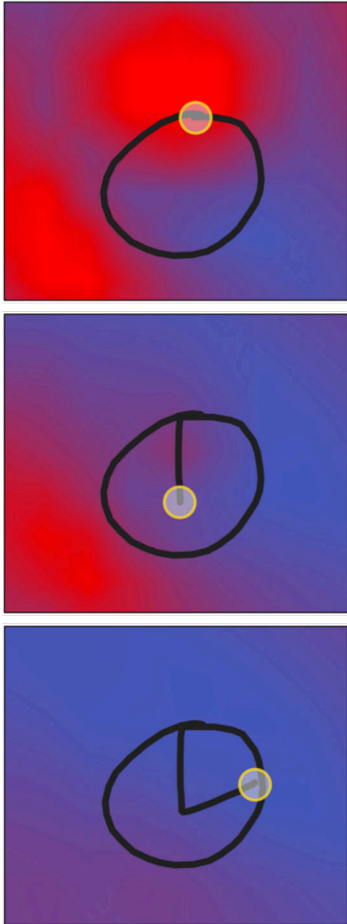
amount of processing, the convolution process is highly parallel and runs in a shader program on the GPU. Secondly, since the inter-agent information sharing takes place in a 2-D bitmap, the approach lends itself to other methods of interaction, such as drawing directly into the space using user-controlled mouse input, multi-touch, or data from a depth camera such as the Microsoft Kinect.

The current implementation features 2-D vector fields that support directionality, enabling effects such as spin and flow. A user can draw flows directly into the space that can be observed by agents. As can be seen in Figure 2, the agents leave decaying trails of written data as they move around the environment.

Tests with large numbers of local agents during the planning stages showed that interesting emergent behaviour—including waves, flows, quasi-stable clustering, and cell-like membrane structures—could arise with only one or two types of passive (non-learning) agents and simple sets of rules mapping their observations to actions. The distributed nature of the final system supports much more complex scenarios: agents are not constrained to behave similarly, or to interpret observations in the same way; and multiple Influence environments can be “stacked” to increase the dimensionality of observations, or tiled to represent adjacent interaction spaces.

#### *Fieldward*

*Fieldward* is a technique for interactively exploring the negative space of a gesture recogniser for the purpose of defining new unambiguously-recognisable gestures [6]. Inspired by the system Influence, Fieldward generates a dynamic heatmap-like display showing *progressive feedforward* of the potential effects on recognisability if the gesture is continued to each point on a mobile touchscreen (figure 3). While Influence used the field-based represen-



**Figure 3:** Progressive definition of a new gesture within the system *Fieldward*.

tation mostly for computation and conceptualisation of information-sharing, *Fieldward* builds the 2-D representation from repeated queries to a gesture recognition engine and presents it directly to the user. In laboratory experiments, participants found that *Fieldward* successfully supported the task of defining new gestures that were both *memorable* (for the user) and *recognisable* by the system, and that *Fieldward* was preferred by users to another representation involving discrete trajectories.

### Exceptions; Hybridisation

This paper is not intended to be a polemic against classification or discrete representations. In fact I have spent considerable time and effort working on solutions for distributed, asynchronous, arbitrarily-connected flows of control associated with discrete *instances* (of objects, blobs, trajectories, gestures, etc.) [8]. The DMIs mentioned above publish discrete gesture events (e.g. *jabbing*, *shaking*) alongside raw and preprocessed sensor data and more continuous features related to movement energy.

So when are discrete objects, gestures, or representations necessary and when should they be preferred? I am not suggesting that such representations shouldn't exist, but merely that interaction designers should resist the temptation to “recognise first, ask questions later” since this will frame the conceptualisation of subsequent design decisions. When recognition is preferred, we should also consider keeping the raw input as an alternative representation, either published as a parallel signal as discussed above, or for reverting the recognition step for potential modification and re-interpretation.

*Expressive Keyboards* [2] are an example of this sort of hybridisation, extending identification of discrete words to support continuous modifications of the output (font, colour,

etc.) by mapping continuous features of the performed user input. The extra dimensions of output are controlled in an *integral* rather than *separable* way [4], and potentially enrich textual communication with implicit communication of emotion or mood, environmental context, personal style or expressive emphasis. In fact, *Expressive Keyboards* is a sort of double hybrid, since the underlying gesture-typing system itself leverages rich user input (i.e. gestured word shapes rather than virtual button presses), albeit for providing redundancy for the word recognition system [11].

### Conclusion

In this paper I have suggested some benefits to using rich continuous representations directly in the design of interactive systems. The HCI research community has already clearly shown the benefits of some techniques and approaches that align with this approach, such as *crossing* instead of clicking [1], continuous *multitouch gestures* for scrolling, panning, and zooming, and *demonstrating* rather than specifying parameters [3].

This discussion of continuous representations and interactions is of course not complete. While there is not space here to consider additional factors in depth, I believe strongly that the continuous streams should be freely connectable and reconfigurable, forming a distributed graph system topology rather than a hierarchical, application-siloed approach in which input and output devices and data “belong” to a particular software application or computing device. Distributed connection middleware such as *libmapper* [7] supports this kind of flexibility, hopefully making it easy for developers to use rich, continuous representations by default rather than reducing human input to small, static, discrete units.

## REFERENCES

1. Johnny Accot and Shumin Zhai. 2002. More than dotting the i's — Foundations for crossing-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, New York, NY, USA, 73–80. DOI : <http://dx.doi.org/10.1145/503376.503390>
2. Jessalyn Alvina, Joseph Malloch, and Wendy E. Mackay. 2016. Expressive Keyboards: Enriching Gesture-Typing on Mobile Devices. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 583–593. DOI : <http://dx.doi.org/10.1145/2984511.2984560>
3. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 145–154. DOI : <http://dx.doi.org/10.1145/1240624.1240646>
4. Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and Jr. M. Preston Mullen. 1994. Integrality and separability of input devices. *ACM Transactions on Computer-Human Interaction* 1, 1 (1994), 3–26. DOI : <http://dx.doi.org/10.1145/174630.174631>
5. Joseph Malloch. 2013. *A Framework and Tools for Mapping of Digital Musical Instruments*. Ph.D. Dissertation. McGill University.
6. Joseph Malloch, Carla F. Griggio, Joanna McGrenere, and Wendy E. Mackay. 2017. Fieldward and Pathward: Dynamic Guides for Defining Your Own Gestures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4266–4277. DOI : <http://dx.doi.org/10.1145/3025453.3025764>
7. Joseph Malloch, Stephen Sinclair, and Marcelo M. Wanderley. 2014. Distributed Tools for Interactive Design of Heterogeneous Signal Networks. *Multimedia Tools and Applications* 74, 15 (February 2014), 5683–5707. DOI : <http://dx.doi.org/10.1007/s11042-014-1878-5>
8. Joseph Malloch, Stephen Sinclair, and Marcelo M. Wanderley. 2018. Generalized Multi-Instance Control Mapping for Interactive Media Systems. *IEEE MultiMedia* PP, 99 (2018), 1–13.
9. Joseph Malloch and Marcelo M. Wanderley. 2007. The T-Stick: From Musical Interface to Musical Instrument. In *Proceedings of the 2007 International Conference on New Interfaces for Musical Expression (NIME2007)*. New York City, USA, 66–69.
10. Eduardo Reck Miranda and Marcelo M. Wanderley (Eds.). 2006. *New Digital Instruments: Control and Interaction Beyond the Keyboard*. A-R Publications, Middleton, Wisconsin. <http://www.areditions.com/miranda-new-digital-musical-instruments.html>
11. Shumin Zhai and Per Ola Kristensson. 2012. The Word-gesture Keyboard: Reimagining Keyboard Interaction. *Communications of the ACM* 55, 9 (Sept. 2012), 91–101. DOI : <http://dx.doi.org/10.1145/2330667.2330689>