This section develops two techniques that have a wide applicability to a range of problems that agents face when trying to understand complex worlds. These techniques find a crucial place at the core of the perception systems of many of the agents discussed in this thesis — Loops Score, how long...? *and* 22.

Chapter 5 — The b-tracker framework & distance mapping

In the previously discussed decomposition of the agent — into "perception", "action" and "motor" systems — the perception system holds the privileged place as the point of entry of the external world into the agent. As we take the agent metaphor, or just an agent toolkit into new arenas — choreography, music, visual art — unsurprisingly, the perception systems of our agents require significant attention.

For the perceptual worlds inhabited by the agents are broad ranging: the agents in this thesis perceive the details and the gross aspects of human movement (motion-capture data), of human musical performance (data from an instrumented piano), of human speech (material from a narration) and of the sound of music itself (from live microphones in a gallery). One test of the agent metaphor is to organize these disparate domains without trying to unify them. Indeed in this chapter and those that follow we will see that agent metaphor offers the ability to construct a small set of principles and technologies that span this range — allowing, perhaps even provoking, new relationships between computer, movement, space and sound.

It is in this section that I articulate two such organizing perceptual frameworks — these are the "open forms" of the agent perception system, that help organize,

or indeed provoke, these relationships. It is also in this section that we can formulate most sharply the analysis and critique of the methodologies and results of the traditional digital artist's "mappings".

1. The perception system

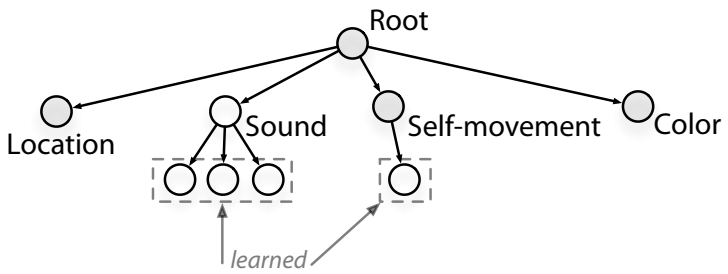


figure 42. A c5 agent "percept tree" classifies and extracts information from the perceptual world.

Often the perception system of a c43/c5 toolkit based creature follows a tree structure. The *percepts* at each node in this tree possess the ability to extract information from the rawest providers of sense information to the creature. This is a hierarchical decomposition of the *state* of the world, as sensed by the agent, into a set of categories, or at the very least, carefully treated responses to it.

Throughout this thesis there have been agents that have grown their percept trees to dynamically extend and tune the way that they decompose the world. In *The Music Creatures' exchange* agent, sub-models of the "sound" percept are dynamically added in response to hearing segmented audio in the gallery; in *network* these sub-models carry transition information that is used to create the body of the creature; in *tile* there is a population of rhythm models, not sonic models. *Loops*, for the matter of a little simplicity and computational efficiency amongst its numerous creatures has a fixed number of percepts looking for the "unexpected" in the colony's signaling environment. Finally, it is by dynamically monitoring and populating sub-levels of this hierarchical structure that *Dobie* is able to construct new states in the world from which to explore new (state,action) pairings, as described previously, *page 61*.

This hierarchical structure works well for these creatures and for a number of others, particularly in simple virtual worlds. But that this is a hierarchical *decomposition* of the state of the world needs to be made clear: a positive response from the "sound" percept of *Dobie*, the interactive, trainable dog indicates the

presence of sound of the environment; a positive response from the "black" percept of a wolf pup in *alphaWolf* indicates the presence of a another black pup in its field of vision; yet the co-activation of "sound" and "black" does not indicate the presence of a single "howling-black-wolf-object". Simply that there is howling and there is blackness somewhere.

Thus, we might say that "objects" in the world, should they exist at any level of description in the virtual environment, are broken up upon entry to the c43/c5 perception system. And that any perceptual fusion that occurs is up to the agent to perform.

This is only the first half of the perceptual fusion problem. Having completed this there exists, in some c5-based creatures, parallel perception trees that take fused packages of percept-tree response and constructs higher level recognizers that have categories for such things as "howling-black-wolf". However, even with these second order trees, there is a need to fuse the information from these perceived objects with objects previously perceived. And this *tracking* of objects is the other half of the perceptual fusion problem.

One can construct agents and interactive worlds that do not require any solution to any tracking or fusion problem: *Dobie*, for example, cared about a reward marker and the position of the interactor's avatar in the world, but cared not for any representation of their common origin; *alphaWolf* could in most cases, like many agent to agent perception problems, simply "cheat" and remember to package up all of the perceptions that came from the same particular agent together, and having done so, no ambiguity remained; the creatures of *Loops* never needed an object model, rather they sensed the average of all the creatures' effects on the surrounding signaling-fluid.

There are three possible reasons for not "cheating" or at least constructing an agent tool-kit such that cheating is not mandatory. The first is computational efficiency — complex perception for the 42 creatures of *Loops* was, at the time , out of the question. The second is what one might call perceptual honesty — that by demanding that our agent synthesize its own object-level models rather than obtaining them directly from the world, the mistakes that the creature makes concerning objects will be believable and, ultimately support an assignation, by an observer, of the agent of consistent knowledge and intentions. There is a substantial pay-off in realism and behavior for what might seem like substantial unnecessary busy-work.

Compelling evidence for this payoff is presented in D. Isla, *The Virtual Hippocampus: Spatial Common Sense for Synthetic Creatures* S.M. Thesis, MIT. 2001.

The third reason for this decision is that when one makes the connection between the virtual agent and the real world more porous, there are simply no object models to be found and the agent *must* synthesize its own. This is the problem we face in interactions less structured than in *Dobie* and *alphaWolf* and, indeed, in domains closer to sensing the real world directly such as robotics. Our agents, when they enter the context of dance theater, must synthesize and maintain models of moving dancers; when they enter a gallery they must construct and monitor models of sonic material; and when they listen to the performance of music they must create and track the location of the performance in the "world" of the score themselves. These things are simply not directly available from the "sensors" that we known how to make.

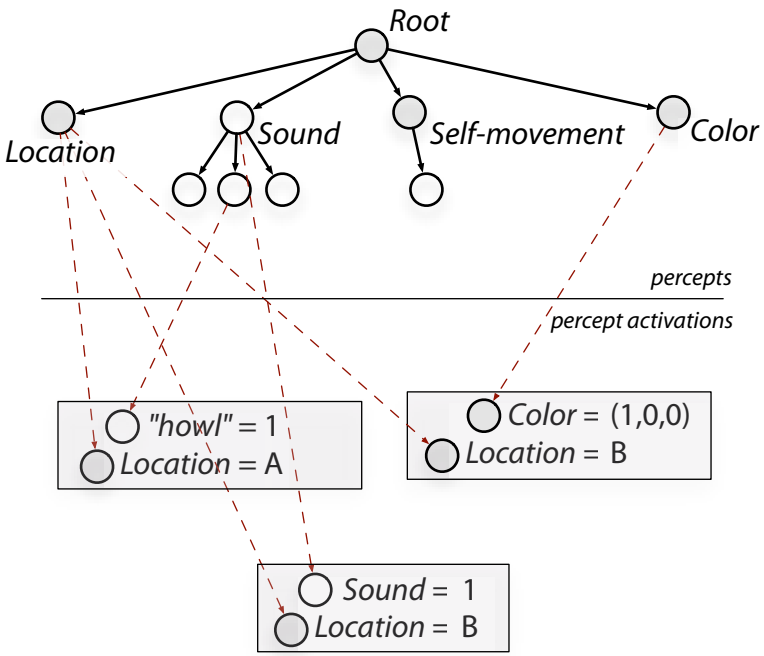
Thus in agent worlds that are more strongly coupled to our worlds, tracking and fusion problems are much less avoidable, since we seldom get an opportunity to control — as we wander as artists from domain to domain — both the sensing and the subject being sensed.

Therefore throughout this work a broad range of algorithms are located in places that act as "perception systems" for the rest of the interactive artwork —

in particular those that build and sustain a set of"object" hypotheses. I shall use as examples in this thesis: **score followers** — that match live musical performance to a pre-arranged score; **choreographic trackers** — that match live movement to open or closed versions of previous rehearsals; **rhythm finders** — that look for repeated gestures and form more complex ideas of the speed of a gesture; **movement trackers** — that rework synthesized movement into new sequences; **speech recognizers** — that identify snippets of sound as similar or different to previously important sounds.

2. The b-tracker"design pattern"

If the perception system is where the uncontrollable organization of the world repeatedly meets the internal author-able organization of the agent, a tracking problem occurs when the agent needs to form perceptual structures that exist longer than a single perceptual snapshot, where new information needs to be matched and incorporated into older structures, when ongoing structures become repositories for learnt information, and when old structures require maintenance and extrapolation. Problems in this task range widely: short term, classic object persistence problems — is this object the same object that I saw some moments ago (from *Dobie*) ?; medium term support for ongoing actions — I have drawn a line from this object to this place, where now is this object (from *how long...?*) ?; long term memory problems — is this previously encountered dominant towards me (from *alphaWolf*) ?, is this sound like a previous sound (from *The Music Creatures*) ?



To develop a general, unifying, reusable framework for solving these tracking problems we decompose the issue into three main stages that take place over two pools of data (we'll see cases where more complicated structures are built up by layering trackers constructed in this fashion). The data pools are the **incoming elements** and the **ongoing models** and the stages are **incoming element fusion**, **incoming->ongoing prediction / match / update**, and **ongoing cleanup**. We'll first develop each stage of the framework before specifying these stages precisely.

figure 43. A hypothetical example consists of object tracking in a multi-object world. "Incoming elements" in this case are various continuous or categorized perceptions that are presumable located in the world.

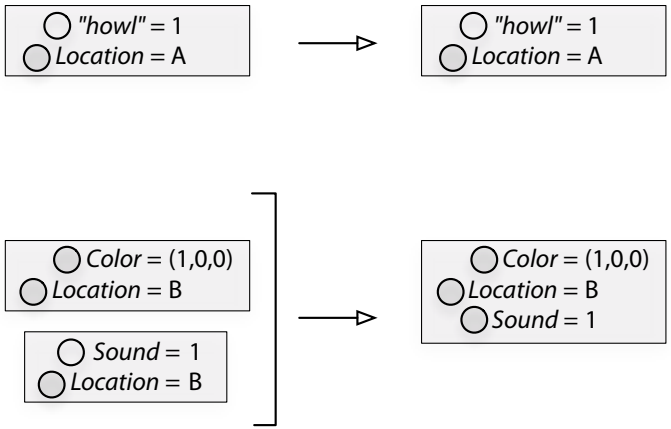


figure 44. These can be fused together on the basis that perceptions that come from the same location come from the same object.

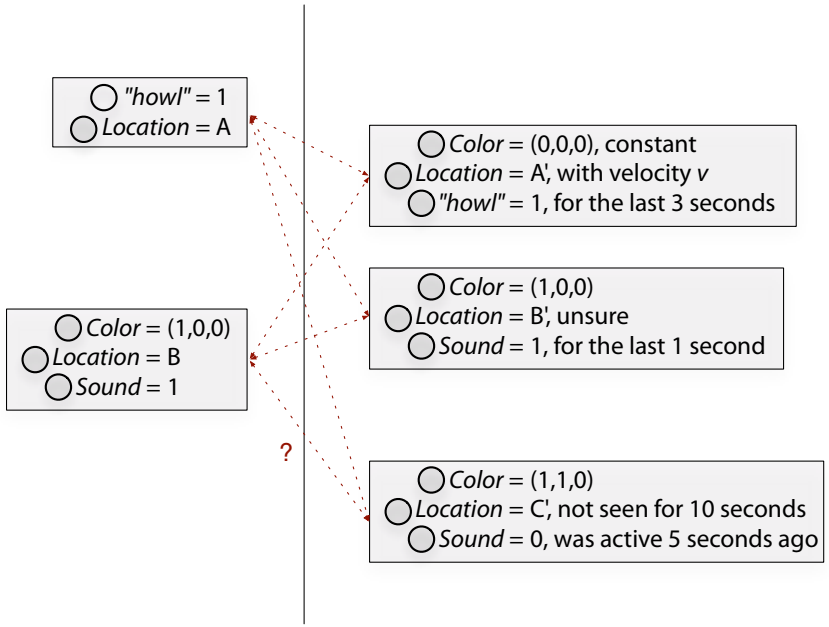


figure 45. Previously the agent has encountered a number of objects, we need to match these older objects with the new, fused sense data.

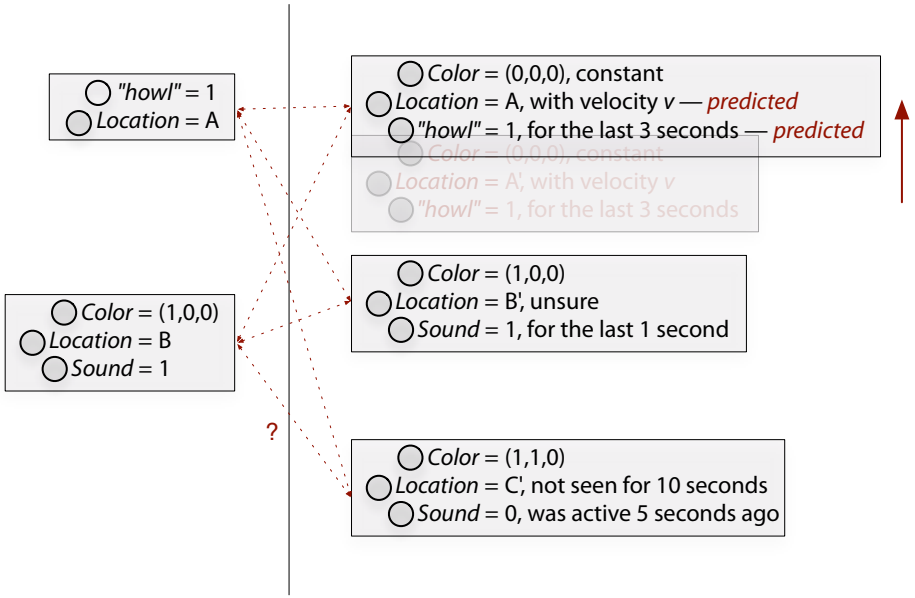


figure 46. In many cases it makes for a more robust perception system to match these new sense data with *predictions* of what these older objects should be now. This prediction process might change the apparent contents of the older-objects, or it may form and add new hypothesized descendants and nominate these as new ongoing models.

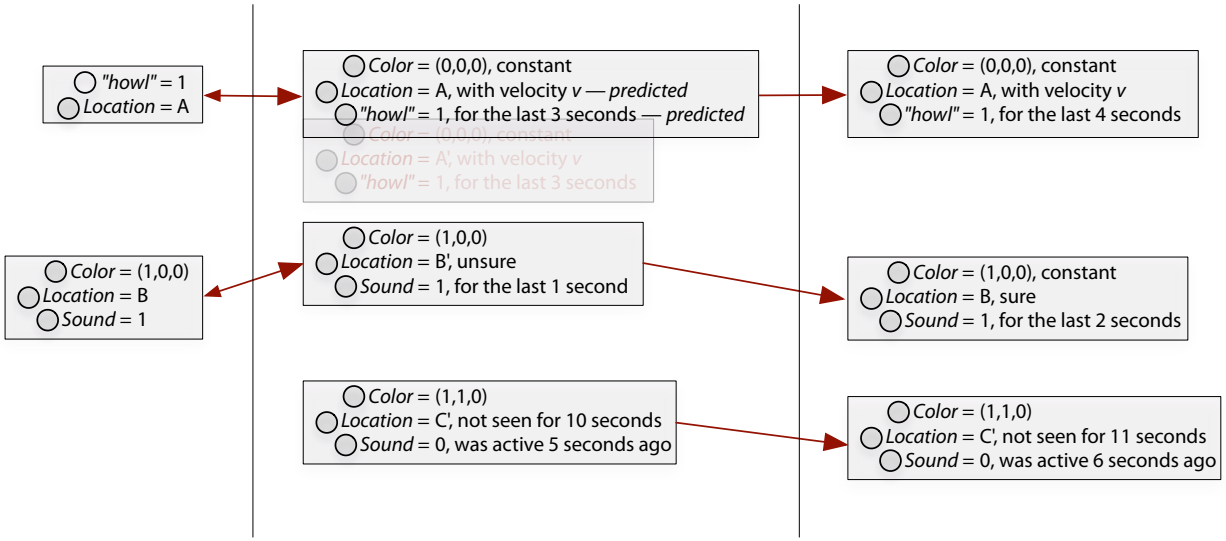


figure 47. In either case, once the new data has been **matched** with some of the older object-models, this new data is **merged** with the older object models. In particular, the agent's confidence in on object model will change, and the agent's confidence in the very existence of this ongoing object will change. Match algorithms vary in the deployment of the b-tracker framework — the two most commonly used are a greedy merge up until a certain threshold and a Hungarian assignment solver, *page 291*. We'll see a selection of simple algorithms below.

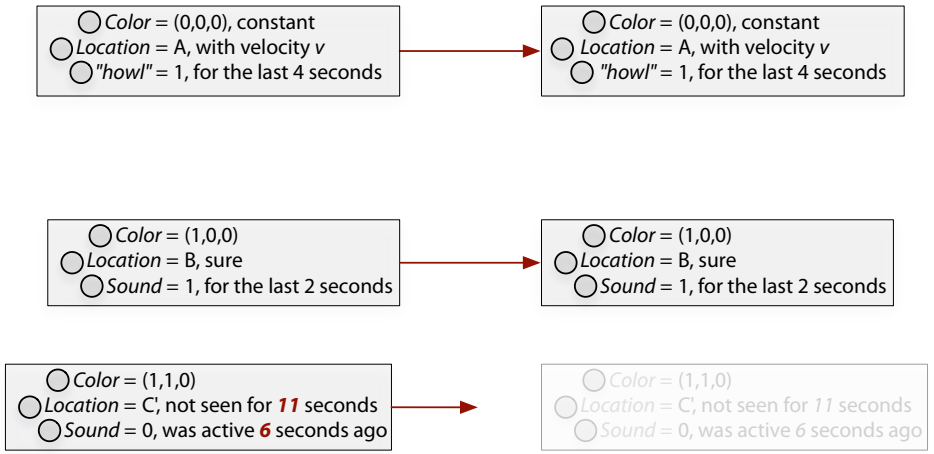


figure 48. Confidence in the existence of some objects might be so low that they are **culled**, stable confusion in the value of some data inside an object may result in ongoing model **fission** or two ongoing object models might be seen to be really the same object and **fused**.

Finally, we note that **top-down** influences may be exerted on the contents of this perceptual structure by injecting particular elements into the incoming set, or more interestingly, speculatively injecting objects into the ongoing model set (and waiting to see if any data "sticks" and increases the confidence associated with this model). In the above, diagrammed example, a creature might hypothesize the existence of an object in the world (perhaps food) with an unknown location. This ongoing model can be used as a placeholder object for action.

Of course this is all extremely general, but it's worth listing the data-structures and algorithms that need to be added to this framework, to provide a kind of template that we can fill in when we come to deploy this framework. To make concrete these stages, I'll sketch the structures and algorithms needed to construct four of the example uses of this framework that were deployed in artworks discussed in this thesis.

They are: a **Hough tracker** — given some movement (here, of dancers in *how long...?*, and the source video of *Imagery for Jeux Deux*) it tries to hypothesize and maintain straight lines that explain the movement or images, named after the Hough transform in image processing that finds straight lines in an image; **score follower** — given live performance data (notes played in *Jeux Deux*) this tracker works out where on a known musical score we currently are; **marker tracker** — given noisy, unlabeled, untracked motion-capture data tries to compute marker assignments, and smooth positions and velocities for these points while ignoring transient ghost markers; **dancer tracker** — given good marker data tries to cluster these locations into isolated areas and thus, without matching skeletons or using any other kinematic knowledge, tries to find clusters of points that are likely dancers.

The Hough Transform — **P.V.C. Hough,** *Machine Analysis of Bubble Chamber Pictures*, International Conference on High Energy Accelerators and Instrumentation, CERN, 1959.

The seminal **score following** work: **B. Vercoe, M. Puckette.** *Synthetic Rehearsal: Training the Synthetic Performer*. Proceedings of the 1985 International Computer Music Conference. San Francisco: Computer Music Association, 1985.

Imagery for Jeux Duex was made to accompiany composer Tod Machover's concerto for "hyperpiano" : **T. Machover,** *Jeux Duex for Hyperpiano and Orchestra*, Musical Score, Boosey & Hawkes, New York. 2005.

The marker and dancer trackers are further discussed on page 286. A less direct tracker is found on page 348.

These problems are of roughly increasing complexity: the *Hough transform* is relatively solved problem, although I am unaware of any interest in solving it incrementally; many have written *score followers*, a fundamental if dangerous building block of interactive computer music for at least decade, and around for two, but our implementation gives us a few novel uses; in tracking *markers* it transpires that it is more important to have a solution based in this perceptual framework than it is to have a more accurate but proprietary black box solution; similarly with the *dancer tracker*, for a number of reasons we can exploit access to the specifics of this solution stratagem during the imagery for *how long...?*

Firstly, the data structures:

incoming element: has some, perhaps fragmentary, labeled piece of data associated with it. In a *Hough tracker* example this will be a short line segment, in a *score follower* example this will be a time-stamped note, in a *marker tracker* example this will be the position of a marker of unknown origin, in a *dancer tracker* this will a set of tracked marker positions.

ongoing model: has space for a complete object model, together with enough element history, use history and merge history to participate in this agent. *score follower* — hypothesized score position and tempo; *marker tracker* — Kalman filter model of marker position, velocity and acceleration; *dancer tracker* — k-means-based clusterer of marker positions.

And at each stage there are algorithms that might be provided:

incoming element fusion: spots that some elements should be pieced together into intermediate packages of data in order to make the matching easier. *Hough tracker* — successive line elements that are too short to be reliable are pieced together into longer elements with a more definite direction; *score follower* — no fusion takes place; *marker tracker* — markers that are too close together are merged; *dancer tracker* — no fusion takes place.

ongoing model prediction: prepares the outgoing models for matching by speculatively, and reversibly, updating them with the current "time". *Hough tracker* — no prediction; *score follower* — predicts where we would be in the score right now if we continued at the hypothesized tempo; *marker tracker* — the kalman filter prediction cycle; *dancer tracker* — the k-means update cycle on the most recent data.

incoming → ongoing match: matches the incoming elements and element packages with the ongoing models often using a distance metric between elements and models. *Hough tracker* — nearest neighbor search using a line-segment to line-segment distance; *score follower* — all incoming data "matches" all models, every hypothesized score position and tempo has to explain the incoming notes; *marker tracker* — an implementation of the "Hungarian algorithm" linear programming method produces unique par-

ings between markers and predicted marker positions, some markers will be new, some ongoing models will be unmatched; *dancer tracker* — all incoming data "matches" all models, all markers have to be explained by each hypothesized dancer configuration.

incoming → ongoing merge: having made a match (or made no match) the ongoing model needs to be updated. Often a global confidence score is associated with a model. *Hough tracker* — a line segment model is rotated and translated toward the new line segment data; *score follower* — each matched model builds some good hypotheses as to what note in the score that incoming note corresponds to, and what that does to the current tempo; *marker tracker* — marker positions are added to the ongoing kalman filter as an observation stage; *dancer tracker* — potential k-means cluster fission and fusion are evaluated in the light of the new data; fitting scores are calculated.

ongoing model cleanup: culls, fuses, fissions and injects ongoing models into the pool. *Hough tracker* — poorly scoring models are dropped, good scoring models duplicated as the number of hypotheses are kept near a particular target number; *score follower* — poor hypotheses are dropped, nearly identical hypotheses are merged, good hypotheses that have multiple explanations of the most recent data split; *marker tracker* — poorly performing, lost markers are dropped; *dancer tracker* — poor models are dropped, good models that wish to offer versions of themselves with greater or fewer active clusters (dancers) do so.

top-down influences: That systems typically post-perception can offer top-down influences on the perception system is also of considerable interest, particularly in a space where an agent might commit to acting upon a model: in a *hough-tracker* — we might need to maintain lines that have been drawn or are being drawn or are the lattice for an ongoing move-

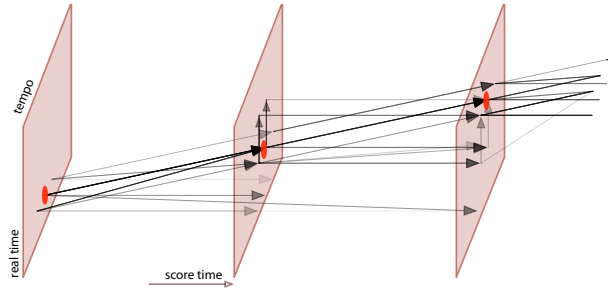


figure 49. In a score follower the ongoing model is a pair (score position, tempo). The b-tracker population of models attempt to predict the next notes and compete to explain the data as it arrives in a live setting.

ment; in a *score follower* — we might have alternative means for guessing the current position, during rehearsal or performance; in a *marker-* or *dancer-tracker* the agent again may already have started to act upon a marker and require that such a position is maintained and updated.

In its chameleon-like configurations the b-tracker framework relates to other work that has been used in the fields that border on that of making agents' perception systems. Maintaining a population of likely hypotheses (ongoing models) as one scans some data piece by piece (incoming elements) is very similar to a beam-search, a general purpose heuristic search technique used, for example, in planners. A beam search is a different (a more general, but often less complete) but related way of solving problems typically solved by dynamic programming.

And much has been written about dynamic programming as a framework for understanding, or at least building, musical perception, for example the work of David Temperly and Roger Dannenberg. But a careful reading of this work will show that once the initial excitement surrounding the dynamic programming trick — the spectacular apparent efficiency of dynamic programming over complete search, converting exponential time algorithms into polynomial time — it becomes increasingly hard to formulate perceptual frameworks inside the limits of the dynamic programing *per se*. Indeed, as Temperly is forced to add optimizations in his monograph on the use of dynamic programming in music and to look to fusing dynamic programming processes together, it looks more and more like heuristic search.

R. Dannenberg, *Dynamic Programming for Interactive Music Systems*, in Readings in Music and Artificial Intelligence, **E. R. Miranda**, (ed.), Contemporary Music Studies series, Vol. 20, Harwood Academic Publishers, 2000.

D. Temperly, *The Cognition of Basic Musical Structures*, MIT Press, 2001.

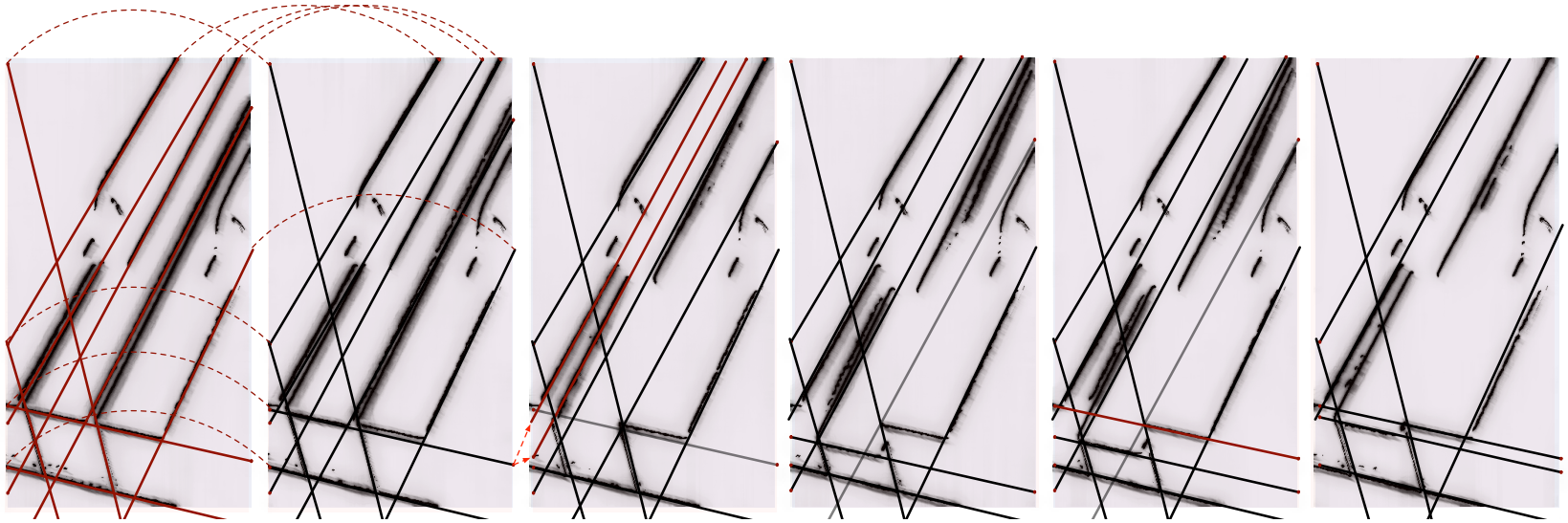


figure 50. In *Imagery for Jeux Deux* video of piano key depresses was integrated into the piece's network of points and lines by annotating the video with straight lines. These lines, identified by a Hough transform on each video frame were then tracked using the b-tracker framework. This yields lines that follow follow the underlying animation of the key press and can be connected to other material in the work.

Especially in the case of the marker tracker there is a clear a relationship between the b-tracker as described here with the Conditional Density Propagation algorithms first described in:

M. Isard and **A. Blake,** *Condensation — conditional density propagation for visual tracking*. International Journal Computer Vision, 29, 1, 1998

A. J. Viterbi. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* IEEE Transactions on Information Theory 13(2):260–267, April 1967.

With application to hidden Markov models, **L. R. Rabiner.** *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE 77(2):257–286, February 1989.

Frames: **M. Minsky,** *A Framework for Representing Knowledge*. MIT AI Lab, Memo 360, June 1974.

Of course, the most important example of dynamic programming is arguably be the Viterbi"search" of hidden Markov models. Indeed, we'll see a gesture recognition task posed in the b-tracker framework, *page 296*. A population of simple predicting trackers that explain incoming data looks a lot like the condensation framework used in computer-vision tracking problems. Further afield, the top-down injection of ongoing models reminds one of symbolic AI's frame structure — when actions want to hypothesize the existence, perhaps of an hidden object, they might instantiate an ongoing model that will act as both a repository for information, should this object become, visible and as a token for other actions to use (for example to provoke and guide search behavior) based on the uncertainty of various"slots" in that"frame".

It is not then that the b-tracker framework necessarily opens up previously in-

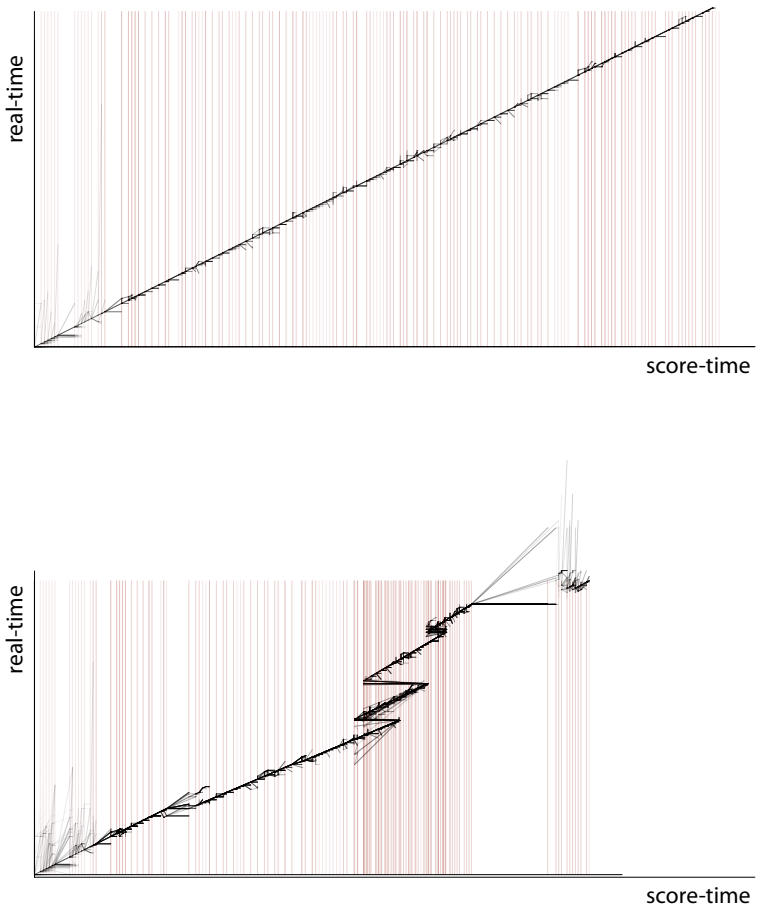


figure 51. The b-tracker framework allows a robust score follower to be quickly created from reusable parts. The above diagram shows a tracking of score position using a synthetic performance of Tod Machover's *Jeux Deux* that has been artificially corrupted with 10% of the notes played wrongly. Despite the noise the tempo is clearly constant and accurate. The lower diagram shows a snapshot from a rehearsal — where at one point a few measures of music are repeated.

tractable problem domains — in terms of analysis — but rather that is is a single core framework for thinking about, and implementing, many kinds of things that intelligent systems end up needing to do. The motivation for and the success of this framework comes from two places: firstly that its broad applicability will allow a great many agent perception systems to be quickly and robustly considered and assembled, exploiting a common set of code and visualization tools; secondly, that as a way of allowing an agent to see the world it is not only an open or "white" box but better — it offers the right kind of openness and the right sort of partial inner structure for other systems to communicate with, on the level not just of "output" or "results" but of inner dynamics as well.

It is worth pausing to reflect upon the openness of this structure compared to other approaches — since the b-tracker framework offers a concrete way to talk about some perception problems, its i worth taking stock and comparing this framework to other "perceptual frameworks" used in digital art. While some incredibly well-written analysis might offer a single "answer", a single "perception" of, say, our current position in a musical score, or the position of a dancer on a stage, the b-tracker offers a small, trackable population of scored hypotheses with histories and uncertainties. Which would we rather work with as artists?

I suggest that tracking problems occur in exactly the places where a mapping approach would fail to gain traction. In lieu of the perfect answer — the exact score position, the precise dynamics of the stage, a stock function-like transformation offers to *flatten* the information present in the perceptual world into a single quantity, which in turn allows subsequent simple transformation. It makes no difference if this quantity is of high dimension or a single number, the function-like core of mapping and transformation promotes a *constant* data dimension.

Tracking problems occur in places where simple averages may capture nothing,

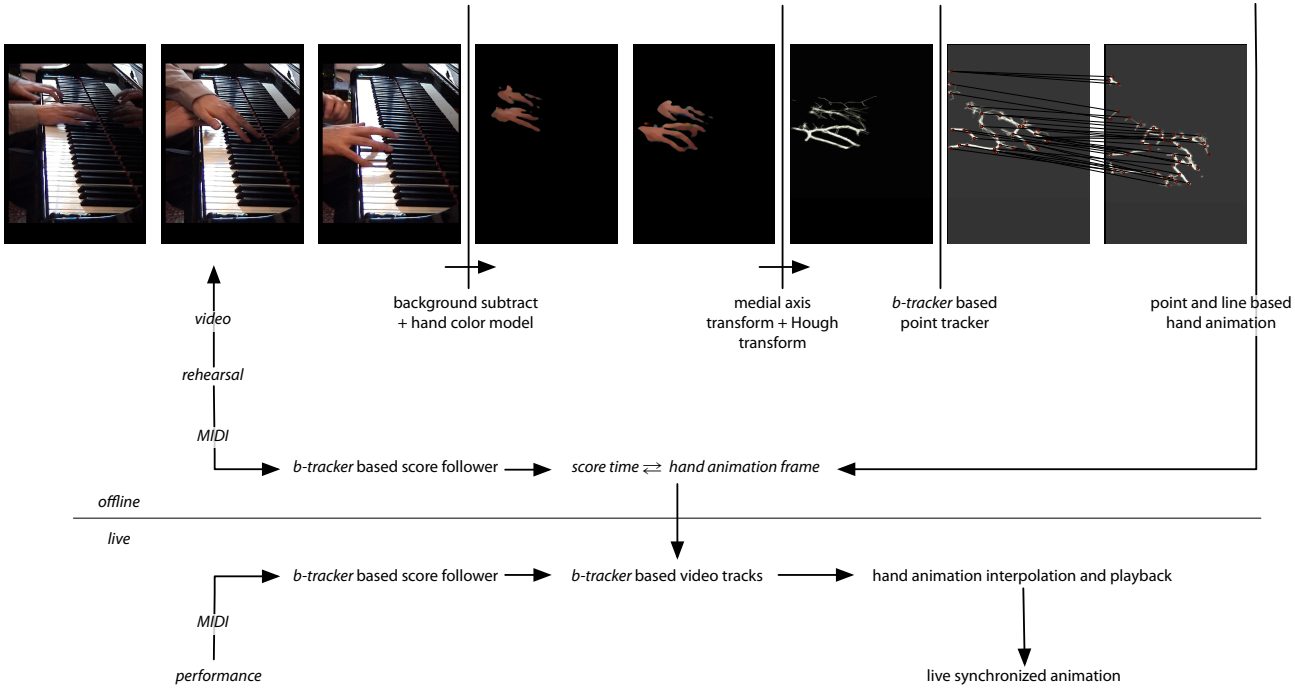


figure 52. *Imagery for Jeux Deux* couples two b-trackers together to synchronize animation (derived from video footage of a rehearsal) to a live performance. First, the note data (MIDI) of the rehearsal is converted to a score-time (in quarter notes) using a score follower. This relationship can be inverted to provide a video time-code and playback rate per quarter note of the music. Then the score is tracker live. As the live b-tracker follows the score, animation material from the rehearsal tracks each score hypothesis, fading in and out with the ongoing model confidences.

where what is being perceived is intrinsically multi-modal (in a statistical, rather than media sense). The *average* score position when there is uncertainty over whether a performer is repeating a section of music is worse than most other guesses one could make; the *average* position of a dancer when there is uncertainty over whether there is one or two dancers can be arbitrarily poor. A motion capture marker-mean-position might be so noisy as to be useless (consider the case where a dancer lingers on the edge of the motion capture volume), and might be arbitrarily far away from the dancers (consider the case of two, opposing, lingering dancers). When confronted with such "noise" mapping tool-kits offer to bury these measurements in increasing levels of filtration. But if there is

insufficient information present in this signal to start with, if the perceptual world of the autonomous artwork is already aliased so severely, to look for the solution by eliminating even more information from this signal seems perverse, *page 288*.

In some cases an agent can work with these potential hypotheses without flattening them in any way: in *Imagery for Jeux Deux*, multiple video streams synchronized with individual score-tracking hypotheses fade in and out with the confidences of these models — the resulting perception is of a continuously synchronized visual performance, that waxes and wanes with the certainty of the tracking, which in turn is affected, on a different level, by the very clarity of the music at the point. Even simpler, lines drawn to moving points on the stage of *how long...?* commit the b-tracker to maintaining an active hypothesis for the end marker while the line exists, allowing the linear form to unfold gesturally, rather than appearing in a single frame.

In other cases, of course, an agent has to pick one hypothesis and stay with it. What constitutes a good decision-making technique? — an action selection strategy. We have already seen that such algorithms are judged by their relevance (they pick good models), their coherence (they stay with these models long enough and no longer).

By using an action-selection framework, our agents can then become extremely conservative concerning the deletion of information. And this aspect of this work — and it really is an aspect of the agent approach that we are building here — is reflected directly in the artwork. It is the difference between being able to stably form a dancer-like cluster of points, maintaining a top-down influence over that cluster and performing a visual operation on their position — one that is coherent over seconds or minutes — versus being able to "visualize" the mean of all the markers on the stage. The former is tentative but specific,

Of course, this flavor of explicit *representation* — our population of hypotheses about the world — appears to go against the spirit at least of the early agent-based research in the field. However, we can avoid some of the dissonance with our historical narrative by realizing that the actions that our agents make are not dictated by or dedicated towards maintaining this model of the world, nor is this model a singular and totalizing end in itself.

possibly fleeting, but a particular segmentation of the perceptual world; the latter is an affirmative, constant, broad averaging over the whole environment. The former meets the world on its own terms; the latter slices the world in a predefined manner regardless of the specifics of what it happens to slice.

The b-tracker framework offers a fundamentally different authorial position on the perceptual world occupied by an interactive artwork. Constant dimensional, pre-arranged slices of a perceptual world that do not segment the perceptual world are of limited use in complex worlds; in fact this might be the very hallmark of complexity itself. By using, instead, these techniques, our agents are more open to the interactive possibilities of those complexities.

3. The distance mapping algorithm

This is not to say that the action *selection* begins and ends the *use* of perceptual information —the properties of the thing perceived of course leak (or perhaps are even mapped) into the action that demonstrates that the perception has occurred (move *toward* an object; reconfigure *toward* a new musical measure, begin breaking down a "scene" of the choreography over *there*, manipulate the temporal flow of a graphical "score" based on *this moment* occurring now). But an agent framework allows the complexity, the multi- and variable dimensionality of the world, into the agents that exist inside it.

This said, we have seen problems and solutions to problems within this agent perspective that have the flavor of mapping to them — the long-term learning database of *The Music Creatures, page 127*, learns simple scalings from one domain to another; the motor learning of music creatures makes small self-organizing maps in order to understand the effects of its own motor control. Of course these are "small numbers" inside large systems — rather than large systems made up of small numbers. But what makes these different from the ex-

cruciatingly hand-made signal manipulations of classical mapping is the technologies that surround these magic numbers and transformations — technologies that allow these numbers to be indirectly and automatically set by a more "human" description or process of what those numbers ought to be.

Our simple, often one-dimensional self-organizing maps can learn a scaling from one domain into a particular, fixed range, while removing some of the static statistical features of the input domain, decoupling the *consumer* of this signal from some of the specifics of the *producer*. This technique, as much as it is useful, does not care about the temporal qualities of the signal. It does nothing with the temporal statistics of the signal (indeed, the first step in the learning algorithm for these maps is to randomize the order of and often down-sample the input signal) and it gives no interesting control over how the the output distribution changes. Perhaps, and especially as we move towards sharing a time and space with live dance, there is a role for indirectly specified maps that do propagate some temporal information.

As work for *how long...?* and *22* progressed, it became increasingly apparent that we needed to build our own layered structure of perceptions of the movements of the dancers, stacking primitive upon primitive perhaps with parameters that could be quickly learned or reconfigured, rather than approaching the problem armed solely with a detailed foreknowledge of the choreography — foreknowledge that was impossible to obtain given the choreographic work schedule of *how long...?* and with the working practices and improvisatory nature of *22*. Early on, as we began to sharpen the marker- and dancer-trackers, we began to look at other simple measures that we could derive from of, a set of markers moving in space that would be robust to both noise and choreographic decisions. The ultimate payoff for this work comes with a description of the works themselves, but the approach is so general as to merit a separate discussion here.

In *22* there are several properties of the imagery that become, at times, connected to motion on the stage — the obvious way to do this is to couple speed (of dancer) to speed (of playback of video, of movement of infinite lines). In *how long...?* there was a perceptible and yet ungraspable, unlocatable, vanishing rhythm to the movement that stood in defiance of the frame-rate and resolution of the motion-capture cameras. Immediately clear was that speed as "distance divided by time", as one would write it in high school, captured little of the motion of modern dance;with its dizzying curves and recursive foldings back on itself, this rhythm in Brown's motion simply was not to be found in such trivial coarse velocities.

Consider the problem, then, of *automatically* mapping the movement of a dancer to the movement of the virtual animation. Specifically, mapping the movement of a motion-capture marker-set to the movement through a particular, pre-made animation. We would like to specify as little foreknowledge to this "motion-scrubbing" problem as possible — because everything might be different in the next rehearsal or performance — and yet relate the complex temporal qualities of the dancer to the complex temporal qualities of the pre-made animation.

A simpler sub-problem is the mapping of a marker set animation to the "movement" of a single number. More precisely, can we find the motion of a single scalar quantity that most succinctly captures the qualities of that marker-set movement? This problem leads to the classic definition of *multi-dimensional scaling problem*, which is in this case equivalent to computing the principle component or leading eigenvector of the self-distance matrix. All multi-dimensional scaling techniques seek to find low dimensional spaces to embed high dimensional data-points such that **distance relationships** between the points are retained in the lower dimensional space. We could treat such an embedding algorithm as a black box, simply reading the literature and implementing one of the well known versions of the technique. However, a reinterpretation of the principle component analysis that underlies this technique will provide us with an algorithmic formulation that is more efficient for our purposes and much more flexible.

For a brief overview of multi-dimensional scaling: **J. B. Kruskal,** and **M. Wish.** *Multi-dimensional Scaling.* Sage Publications. Beverly Hills. CA. 1977

Our approach here is to take the **input signal** (the marker movement) I_t and a **distance-metric** (that gives a distance from any particular configuration of markers to any other) $\mathrm{d}(I_{t_0}, I_{t_1})$ over a range of time $t = a \ldots b$ sampled by N samples. This distance metric is the foreknowledge that we add.

We can then compute the $N \times N$ matrix of distances D^I. The goal is to transfer this distance matrix over to a single output scalar quantity that is also defined over the interval $t = a \ldots b$ and also sampled by N samples. We do this by iteratively making the $N \times N$ matrix of self-distances for the output signal D^O increasingly like that of D^I. For scalars, where the distance metric is simply $\|x - y\|$ the iteration is:

for each element O_t; $t = a \ldots b$,

$$O_t \leftarrow O_t + \alpha \sum_{q \neq t} \left[(O_t - O_q) \cdot \left(D^I_{qt} / D^O_{qt} - 1 \right) \right]$$

By starting with a motion signal that shows some variation (even if it is simply some noise added to the signal) and repeatedly applying the above equation to each element of the signal N for some small α we will decrease the difference between D^I and D^O.

For a description of the power-method, and its convergence properties: **G H. Golub** and **C F. Van Loan,** *Matrix computations, second edition*, The Johns Hopkins University Press, 1989.

also **G. H. Golub, P. Comon,** *Tracking a few extreme singular values and vectors in signal processing* Proceedings of the IEEE Volume 78, Issue 8, Aug., 1990.

The above formulation of the problem is equivalent to an algorithm known as the power-method of finding the largest eigenvector of a matrix. For non-degenerate starting signal, it is extremely likely to converge and converges with a rate proportional to $\alpha\lambda_1/\lambda_2$ — the ratio of the first two eigenvalues of the distance matrix.

A related, more principled, but less general iterative work is: **T. Morita, T. Kanade,** *A Sequential Factorization Method for Recovering Shape and Motion from Image Streams.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (8), 1997.

However, our almost pictorial interpretation given here seems to offer opportunities for special control that the text-book leading-eigenvector formulation does not possess:

Firstly, we might want to find a **constrained solution** where a few particular O_n are fixed, or are less able to move — this is an opportunity for "top-down" control over the answer, perhaps the agent has already committed its body to some part of the solution and thus this part of our re-scaling cannot change.

Secondly it's easy to see how to **iteratively update** this system when a new piece of data arrives — turning an iterative algorithm into an incremental one — we simply need to pick a single new output scalar that minimizes:

$$O_{N+1} \leftarrow \arg\min_y \sum_{n=1\ldots N} \left| |y - O_n| - D^I_{n,N+1} \right|$$

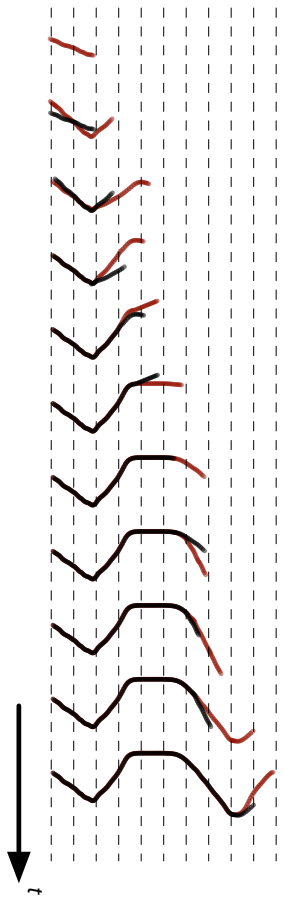


figure 53.
The distance mapping algorithm can be executed iteratively. Here, the solution to the problem given on the next page, *figure 55*, is slowly developed over time.

Since O_N is a good starting guess for O_{N+1}, this is an $O(N)$ operation rather than $O(N^2)$ and also allows us to shape the preference for output distribution.

We can go further than this and, introducing a little latency in the output mapping, update not just one but the last few recent output samples, increasing the stiffness of this update as we go further back in time. This might reflect an increasing commitment to older values, perhaps because some other system has acted upon them.

Thirdly, and perhaps most interestingly, we can redefine the above picture in more abstract terms and operate on **non-scalar output spaces**. We define an operation blendedDistanceNorm(O_a, O_b, d, α) that takes two elements of O — O_a and O_b and returns a version of O_b, O'_b such that the distance between O'_b and O_a is α closer to d. Of course, for such a general function we can say much less about the global convergence properties of this algorithm, but if this function always does what it claims to do, and never goes backwards, we know that we will always converge to a solution, if not the optimal solution. Finding optimality, in the face of many local solutions, is within the purview of the b-tracker framework which will meet this representation shortly.

We are now in a position to define a class of automatic, iterative, temporally aware mapping algorithms that are defined in indirect terms: specifically an **input distance metric**, and **output distance metric** and an **output blend function**. In certain cases where the input space has a particular topology we can automatically generate an input distance metric using the self-organizing map techniques previously described; otherwise there usually remains as a degree of freedom to pick a scaling between the distances of one space and the other.

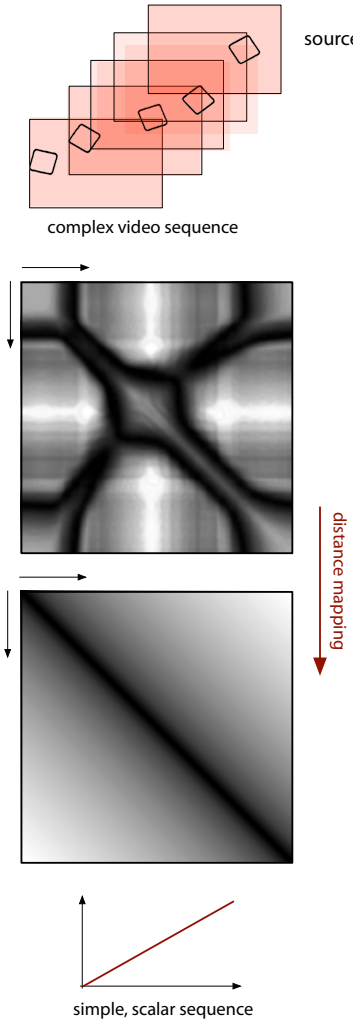


figure 54. The distance mapping algorithm takes one matrix of distances (here formed from a complex video sequence) and tries to the distance matrix of some other signal, in some other mathematical space, look just like it, by modifying that signal.

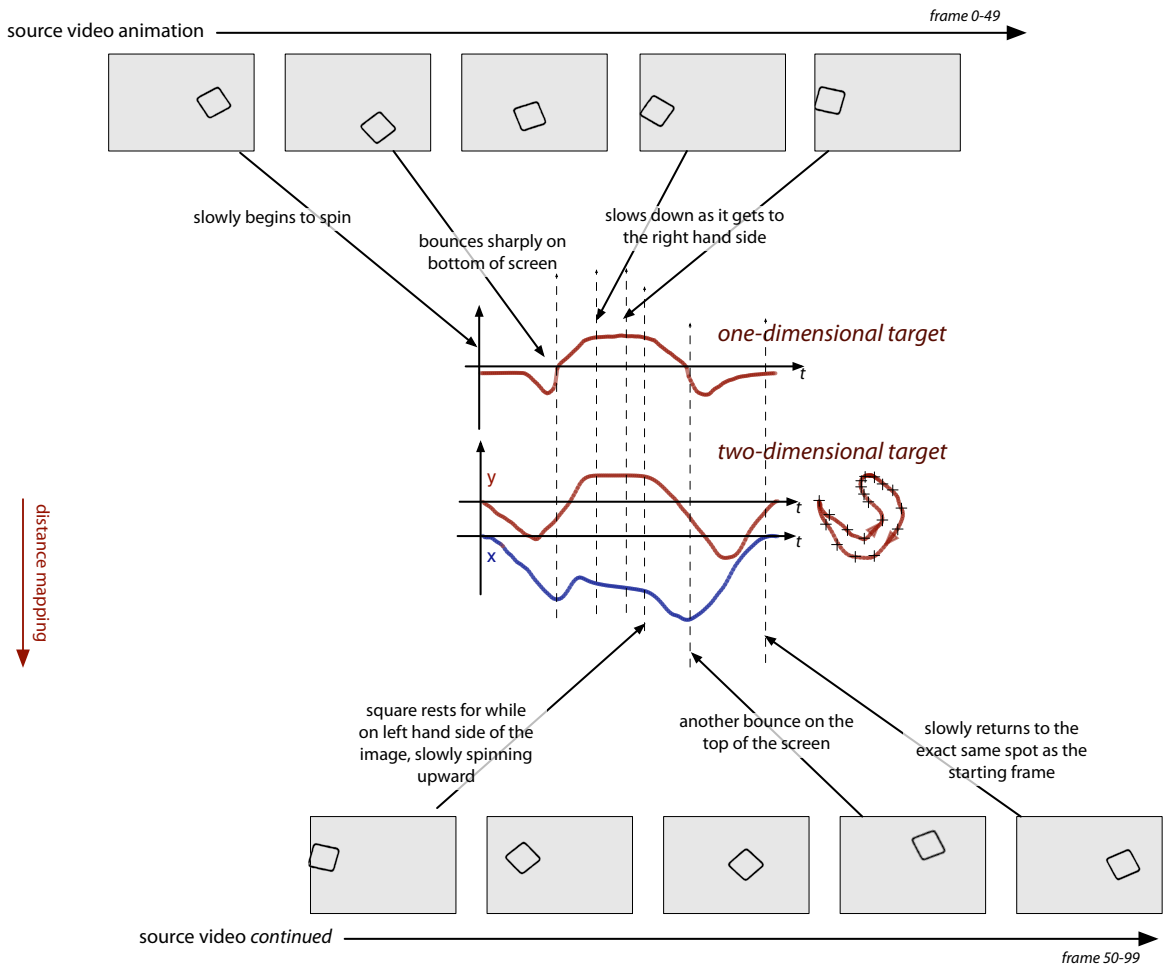


figure 55. Automatically mapping the video sequence of a rotating, bouncing square to a scalar or a 2-vector captures many of the aspects of the source video — the general symmetry, the slight asymmetry, the pause in the middle, the "rebound" upon the bounce. Further distance mapping analysis of the residual distance matrix yields a noisy, but informative, oscillatory signal that corresponds to the wavelength of the square's rotation.

We also note in passing that we have seen other representations that can satisfy these criteria — pose-graph motor systems have distance metrics, and if needs be we can compute a constrained blendedDistanceNorm() function by allowing movement along graph edges; generic radial-basis channels' value representation can be used to formulate a blendedDistanceNorm() function if supplied with a distance-metric (the exact same interface is required for the competitive basis channels, *page 153*).

The motion-scrubbing solution

Thus we can formulate a solution to the motion-scrubbing problem with almost no tuning on our part. We need: a **distance metric** for motion capture marker data — for the case of tracked data, *page 286* this is a trivial sum of squared distances, for untracked, data we use the Hausdorff distance metric; a **distance metric** on the output space — the sum of squared vertex-position differences for the vertex animation data suffices; and a **blendedDistanceNorm()** function — we use an iterative algorithm that searches forwards or backwards in time along the fixed animation, is generic to any interpolatable time series (including the pose-graph representations) that already has a distance metric. In this case the inter-frame distances of the animation that the output distance calculation uses can be pre-computed and the whole iterative system runs in real time with a negligible computation burden; the core iteration is easily accelerated by modern vector processing techniques.

For a formal definition of the Hausdorff distance: **E. W. Weisstein.** *Hausdorff Measure*. From *MathWorld—A Wolfram Web Resource*.

http://mathworld.wolfram.com/HausdorffMeasure.html

Unlike the simple approach of mapping the speed (as in distance divided by time) of dance to the speed (as in frames per second) of video this approach has the following advantages:

it allows **backward motion** by the performer to change the direction of

movement of the video and similarly repetitive movement by the dancer repeats segments of video all the way out to the duration of the time-series windows. No amount of filtering a measured "dancer speed" could ever achieve the same effect for the information is simply not present in the instantaneous velocity of the dancer, but in the relationship to the body now with the body long past;

it is **less sensitive to noise** on the input signal than a first derivative calculation would, be while adding no additional latency;

it is **bidirectional** — the distance metric of the target representation also factors into how quickly we move through it. Rather than just playing out at a variable rate, should the video do something like "reverse" direction, our motion-scrubbing performer will have a much harder time keeping the video moving forward.

Mapping the moving of the dancer to the movement of the fiducial, "infinite" lines that mark the scrim in *22* is accomplished in a similar way — only here the distance-mapping algorithm is located inside a generic radial-basis channel and its output is blended with the influences of nearby geometry on those lines and the line's own momentum. It is useful to use the distance-mapping algorithm within such a process; in fact we can fade the distance mapping layer in and out depending on how good a job it finds itself doing at matching the output distance matrix with the input. This allows not just a mapping to occur, but the system to seize correspondences as the opportunity arises.

Finally, we can use incremental mappings into low=dimensional spaces as an input to higher level perceptual primitives. In particular we can open up the workings of the distance mapping algorithm to the view of b-tracker hypotheses that track a range of output signal trajectories, scoring them on how well their distance matrices correspond to the target input, creating new hypotheses by

perturbing the output signal — helping the system as a whole out of local minima caused by degeneracies of the output space. Alternatively, we can track individual extrema as the move through low-dimensional output signals on the basis that the input signals correspond to at least transiently interesting "poses". We shall see an example of these very algorithms in the *memory score* agent of *how long...?, page 348*.

Concluding remarks

In summarizing the distance mapping approach we might try to find a little space between it and more conventional slices through perceptual worlds. Although the distance "mapping" algorithm began as a direct approach to the mapping problem, with a goal of finding new ways of explicitly yet indirectly specifying mappings, in this formulation information is *compressed* into low dimensional signals but not necessarily *deleted*.

Indeed, we might note, in searching for a definition that separates the "analytical" of mapping from the analytical of more broadly used computer scientific representations is this compression. If we think of the analytical representations that have truly widespread use in synthetic arenas — for example, in computer music we have the short-time Fourier transform, linear predictive coding, the wavelet transform, or even just the radial-basis functions and neural networks of some of the more advanced mapping techniques — each of these representations began or were quickly adapted for the use of signal compression and reconstruction. Multi-dimensional scaling, related as it is to adaptive vector quantization and self-organizing maps, also shares this compression-aspect. And it is this, not any comparative complexity, that indicates a separation between such techniques and the simple averages and global measurements that visual, interactive artists have typically tended to gravitate toward.

This is, of course, a wide cross-section through computer music, for an introduction to most of these techniques as they apply to computer music, **C. Roads,** *The Computer Music Tutorial*. MIT Press, 1996.

For wavelets: **G. Evangelista,** *Flexible Wavelets for Music Signal Processing*. Journal of New Music Research, 30 (1). 2001.

Taken together — the broad, generic and adaptable b-tracker framework and the very specific, analytic distance-mapping technique have a number of features of critical importance to the working artist. They are general purpose: and thus lasting, and worth investing tools and visualizations on; they are generic: armed with these techniques I can build and test algorithms using simple data-sets, perhaps scalars, before exposing agents to the complexities, of, say modern dance; and they are open — they offer structure for thinking about perceptual problems and a variety of detail levels for interaction. These are the two approaches that allow my agents to enter into a whole variety of perceptual worlds, quickly and adaptively.