Outlining temporally parameterised visualisation in support of information fusion

Roland Jesse

University of Magdeburg
Department of Simulation and Graphics
jesse@isg.cs.uni-magdeburg.de

Abstract. The process known as information fusion faces the extraction and combination of information from multiple, heterogeneous sources to derive information of a new quality or abstraction level. Though parts of this process can be automated it is heavily influenced by user interaction. New approaches in visualisation are required to support the user in interacting with constructed information spaces. Motion enhanced visualisation is one appropriate way to express fusion characteristics. Its temporal nature meets temporal relations of the information fusion process. This paper discusses methods and an architecture to represent temporal behaviour throughout the visualisation process.

1 Introduction

Information fusion is seen as a process of integrating heterogeneous data sources, examining stored data, and extracting possibly hidden information [6]. It helps to handle and access potentially large data sets. For that purpose, the fusion process is defined as a concatenation of various fusion operators. These operators include but are not limited to database integration, machine learning, and data mining techniques. The execution of the fusion process is of iterative nature and controlled by user interaction. Behaviour of information fusion as an interactive process is characterised by temporal parameters regarding two aspects: user interaction and process execution.

User interaction is expressed by events which occur consecutively. Each interaction event might cause some action that changes the current state of the system. This change is valid until another user event occurs or until it is outdated by system progress. The effect of any user based event is therefore valid for a specific time interval. Any supporting visualisation technique therefore needs to be temporally parameterised to ease interaction and follow its time constraints.

The temporal nature of the fusion process execution itself results out of the execution of fusion operators. Two operators with a dependency between them are always executed consecutively. Independent operators can possibly be executed in parallel. The order of operator execution results as an acyclic graph. At every execution step, intermediate results are gained. These may be visualised in order to support the user in interacting with the fusion process execution. The temporal component of each result set closely correlates to the time dependend position of its causal operator in the execution graph. Some results are gained earlier than others. Some results are kept valid for longer periods whereas others need to be revised as parameterisation of their corresponding operators is to be changed depending on user decissions. The completeness of a result set needs to be expressable by an accompanying visualisation. Temporal influence on presentation variables is suitable for complying to this requirement.

The next section introduces a motion enhanced visualisation toolkit that is temporally parameterisable. Section 3 outlines an abstract temporal mapping architecture to be used for controlling the toolkit in order to employ it as time related expression media for an information fusion environment. A formal model for temporal mapping is developed in section 4 and section 5 presents some concluding remarks.

2 Temporally parameterised visualisation toolkit

A library of visualisation techniques is currently under development. Special emphasis is thereby put on representation of temporal information. Motion is thereby regarded as an appropriate display dimension of its own. A set of motion techniques have already been implemented. A common technique for mapping visualisation data onto graphical representations are glyphs. These are graphical objects whose display variables are bound to data characteristics. Various techniques exist for constructing glyphs [4, 7, 11, 16, 17, 25].

2.1 Motion as presentation variable

Early studies express that the visual gathering of groups of semantically connected data is eased in case the groups are expressed by similar motion patterns instead of static displays [14]. Ware presents another survey by Johansson [15] revealing that even abstract motion patterns are conceived as representations of concrete facts [24, p. 237f]. Visual metaphors can even be resolved in case the motion is only visible for parts of a second. Likewise, the causal connection of data is easily perceivable by users if represented by either the same motion patterns or patterns that are temporally connected [24, p. 235f]. Bartram et al. show that object movements are recognised faster and less error-prone than changes in colour or shape of an object [2].

Any use of motion as expression dimension is to be parameterised in order to comply with the cognitive potential of the user. Pylyshyn et al. show that motion is an expressive presentation variable with a wide bandwidth for information communication [18]. Admittedly, the authors express that no more than five distinct motion patterns can be perceived independently. Accordingly used, motion is capable of supporting information transmission with special regard to temporal constraints. For complex user interaction tasks, a display parameterisation is to be found that possibly falls back to using classic presentation variables.

2.2 Motion techniques

Using motion for visualisation of object representations is done on the basis of three fundamental motion types: translation, rotation, distortion [13]. Their principle object manipulation is expressed in Fig. 1.

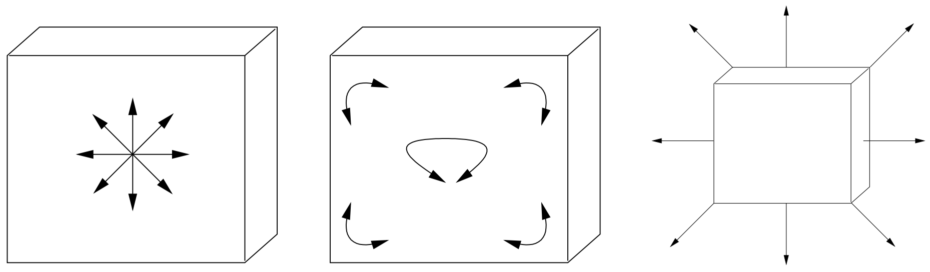


Fig. 1. Basic types of motion library: translations, rotations, distortions.

Translation The glyph's position is constantly changed. Thereby, care should be taken to ensure that the object's location does not change significantly in relation to the remaining scene. An appropriate way to conform to this restriction is oscillation. In case translation modification affects multiple scene objects, their influence on scene coherence is regarded as a group. That allows for these objects to change their spatial structure without influencing other (possibly unrelated) parts of the display.

Rotation A glyph is to be rotated around a predefined rotation centre. An advantage of using the objects central point is that the scene's coherence is not considerably invalidated. However, spotting this motion might require an intense cognitive load by the user. To avoid this, the rotation centre might vary from the object's centre. It does not necessarily need to be a point inside the volume of the glyph. This flexibility is of course paid for by the potential of invalidating the scene coherence.

Distortion Steady changes of shape can be divided in two distinct types: (a) changes of an individual glyph's shape, and (b) changes of the relative positions of a group of related glyphs.

Changing only a single object provides a means of motion without manipulating the overall scene construction. The objects spatial correlation to other objects is not affected. This technique is therefore to be used preferably when the frame coherence of a specific object is of minor importance compared to its influence on the scene context. The change in shape can be done either by repetitive scaling or switching the shape's kind.

Changing the relative positions of a group of respective glyphs allows to express inherent relation characteristics of these objects. The glyphs can be manipulated by similar changes in shape. Additionally, translation changes as mentioned above help to create explosion like distortions.

These techniques can of course be combined to create sophisticated motion patterns. In addition, multiple methods can be used simultaneously in a scene to express different data characteristics by means of different motion application. For the parameterisation of complex patterns, intervals of technique validity are to be specified.

2.3 Scene graph for time integration

Fig. 2 shows the structure of the motion library's implementation as a matter of principle. As expressed in Fig. 2(a) the library is build on top of Open Inventor™. Its structure allows for embedding other libraries of the same abstraction level as well as an integration into these. An example for such a cooperation is a library providing implementations of non-photorealistic (NPR) computer graphics algorithms [23]. As long as this library is also build on top of Open Inventor™, techniques of both toolkits can easily be interchanged and combined. Such an NPR toolkit is currently under development and noted in the figure as OpenNPAR.

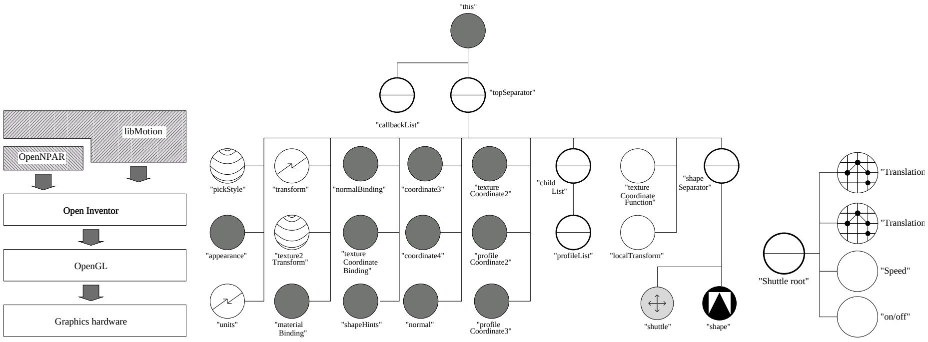
The implementation of individual motion techniques is expressed by Fig. 2(b) and 2(c). Each oscillating motion is implemented as a *nodekit*, which is a sub scene graph representing a geometric object (i.e. a glyph). Coding of the control parameters for motion in this nodekit is integrated in the list of classic presentation variables of the object. An example is given for a *shuttle node* representing oscillation. Motion is controlled by a line with two endpoints, which are specified by translations. The oscillation frequency is noted as a float value for the number of cycles per second. Accelerating motion is easily done by connecting this speed field to an *engine*. Such an engine interpolates between two given float values and has a float output for the discretised intermediate points between these two values. Other motion definitions are integrated in the nodekit structure analogous.

A *separator* node is used for encapsulation of the motion in regard to the remaining scene. This node saves all configuration of the rendering system's status machine. After successful traversal of the nodekit scene graph, this configuration is restored. The overall scene coherence is therefore not influenced and kept valid.

2.4 Exemplary use

For the purpose of representing fusion results, a wide range of visualisation techniques is available. An extensive overview of a toolkit providing these techniques is given in [22]. Fig. 3(a) shows a scatterplot representing results of a fusion process on data from the field of bioinformatics [6].

Especially for large data sets such a scatterplot includes the potential of hiding relevant information by overlapping glyph location. A motion enhanced scatterplot supports in finding such overloaded plot areas. For detailed user investigation such an area might be presented by an exploded presentation. An example of such explosion is presented in Fig. 3(b) and 3(c). The images were

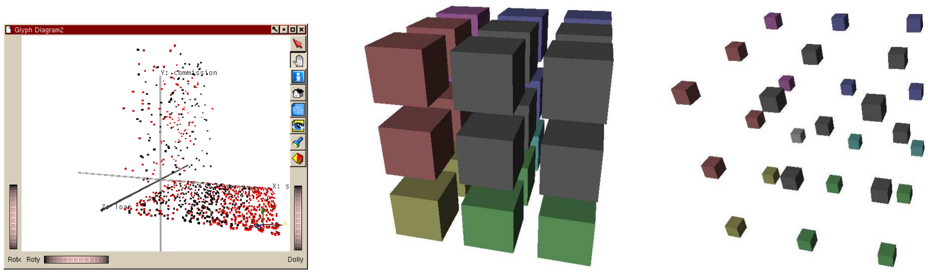


(a) Abstraction layer (b) Glyph nodekit (c) Shuttle node

Fig. 2. Structure of the motion libraray

generated by using the temporally parameterised visualisation toolkit as introduced above.

Assumed the glyphs as shown in Fig. 3(b) represent a subset of a scatterplot presentation. The objects are located in a narrow grid and overlap each other. This same set of glyphs is displayed in Fig. 3(c) after applying an explosion method. The relative positions of all objects in regard to each other are preserved whereas the absolute distances are changed in order to allow for user investigation. The explosion effect is of course reversible. Context information is therefore not lost.



(a) Scatterplot from [6] (b) Unexploded (c) Exploded

Fig. 3. Use of the motion toolkit for presentation

3 Architecture

The principle architecture of a temporal mapping component is shown in Fig. 4. For simplicity, this component is referred to as *Timemachine*. It basically consists of mappers that convert inputs into a visual output description. This output is used to control the visualisation toolkit as introduced in the previous section.

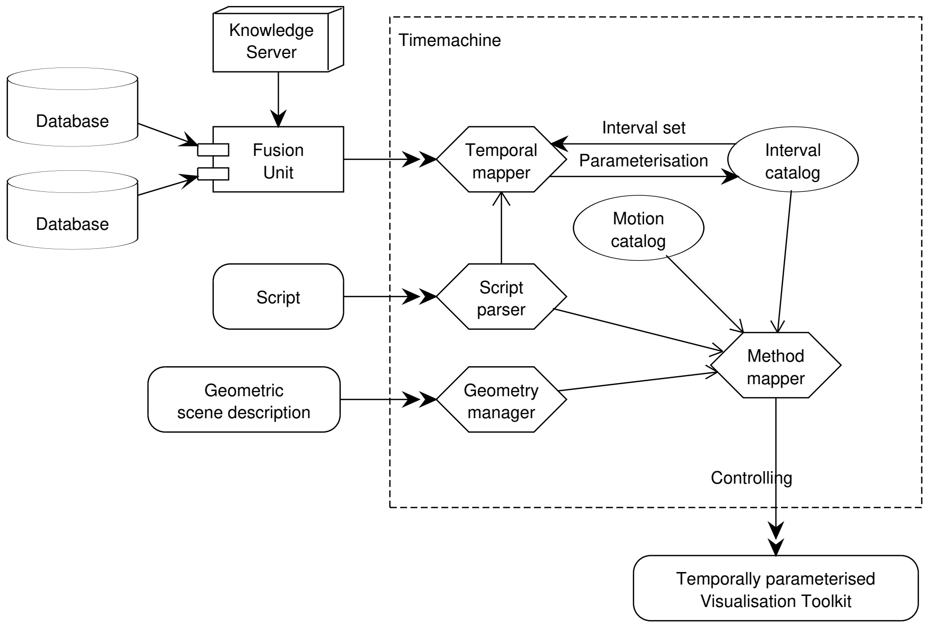


Fig. 4. The abstract temporal mapping architecture

3.1 Inputs

The Timemachine uses a framework for an information fusion workbench [5, 6] as one of its input prerequisites. This is expressed by the *Fusion Unit* module in the figure. Because of its component architecture, the Timemachine can easily be integrated in the workbench framework or similar architectures. Therein it works as integral component and can be used as one of the controllers for fusion process execution.

As further input, the Timemachine accepts a *geometric scene description*. By putting the geometric model into a separate input component, the generality of the architecture is preserved. Thereby the Timemachine can be integrated into a wider range of application frameworks, such as tools for interactive exploration of geometric models [21]. In that case the geometry model would possibly be constructed with an external modelling application. For the purpose of visualising in support of information fusion, the geometry description can automatically be extracted from fusion results. This is done as part of the visualisation pipeline [22, p. 83ff].

The *Script* input represents an implementation of the formal model presented in section 4. Every given script specifies concrete interval relations of the model's temporal relation algebra. Network based component creation is ensured as long as the employed scripting language provides streaming support. In this case the Timemachine is usable as binding component for distributed fusion operation.

3.2 Mappers

Mappers are the central components of the Timemachine architecture. They control the temporal parameterisation of any visualisation output. This is done on two levels: (a) according visualisation *methods* are to be specified, (b) their *temporal* parameters are to be determined.

The basis for method mapping is the method catalog. For the concrete toolkit as presented in the previous section this catalog consists of translations (T), rotations (R), and distortions (D): $cat = \{T, R, D\}$. To determine a motion method, the individual parts of the catalog can be combined. Furthermore, the parameter space $P = \{p_1, \ldots, p_m\}$ is to be defined. A concrete motion method results as $method = \{C, P\}$ with $C = \{c_1, \ldots, c_n\}; c_i \in cat, 1 \leq i \leq n$.

Temporal mapping is done based on an analysis of given interval relations. These are expressed by the scripted part of the input model. Interval consistency is checked based on the model specification (see next section). The constructed interval catalog is to be used as one input for assignment of the parameter space P of the above mentioned method manager. An example is in finding a visual correspondent for completeness of fusion results expressed by motion. It is possibly gained by according adaption of the active amplitude setting. Results gained early throughout the fusion process move less than those gained later. Analogous parameterisation is usable for expressing specific fusion analysis methods as source of concrete result sets.

4 Model

4.1 Model basis

An interactive system which is supported by a visually enriched user interface reflects multi layered representation models. The highest abstraction level acts as *represented world*, which reflects entirely the system's underlying semantics. This level is named $world_{rep} = \{user, scene, relations, operations\}$ and includes the information space constructed by executing a fusion process. The represented world describes a set consisting of the user model, the scene model, relations between different components of the scene, and operations defined in the represented world.

A typical user model is determined by $user = \{percept, actions, scope\}$. The perception of the user as well as her or his actions are represented by the sets *percept* and *actions*. The *scope* describes a set of parameters of the user model that determines the individual target function of a user with regard to the system. It is therefore given as $scope = \{s_1, \ldots, s_n\}$. Depending on the possible interpretations of all parameters, this target function demands the motivation and necessity for parameterisation of motion techniques which are used as separate expression variable.

Allen points out various ways of representing *time* in a model [1]. The presented set of model principles has been extended for specific contexts since then [3, 8–10, 12]. As pointed out in section 3, a temporal mapper component to be used for support of an information fusion framework deals with a combination of time related inputs: interaction events, fusion events, and pre-modelled scripting events. These events are constrained by interval borders. Therefore, a point based representation model is used. This allows for the creation of a complete linear order of all events [12]. The overall set of intervals ($\mathcal{I}$) results from the sum of all modelled intervals ($\mathcal{I}_{model}$) and all intervals gained at runtime ($\mathcal{I}_{runtime}$), whereby $\mathcal{I}_{runtime} = \mathcal{I}_{script} + \mathcal{I}_{user}$. Overall, $\mathcal{I} = \mathcal{I}_{model} + \mathcal{I}_{runtime} = \{I_1, \ldots, I_n\}; n \in \mathbb{N}$ holds.

4.2 Mapping of temporal items onto parameterisation

An interval $I = \{e_{begin}, e_{end}\}$ with an implicit interval duration of $e_{end} - e_{begin}$ is given as a model primitive. For easier notation, the interval borders can be specified by I^b and I^e directly. The set of known events for the overall system runtime is named E. To determine the correlation of any given time event $e \in E$ to an interval $I \in \mathcal{I}$, a function $\psi : E \rightarrow \mathcal{I}$ is defined as noted in equation (1).

$$\psi(e) = \begin{cases} I_0 & & \text{single interval} \\ I_i \iff I_i^b \leq e \leq I_i^e & |i = [1, 2] & \text{double interval} \\ I_j \iff I_j^b \leq e \leq I_j^e & |1 \leq j \leq n & \text{otherwise} \end{cases} \tag{1}$$

Thereby, $\psi(e)$ maps the event's point in time onto its corresponding interval by using plain comparison operations. According to section 4.1 denotes n the number of currently defined intervals and thereby the dimension of $\mathcal{I}$.

An interval can be discretised by use of an interpolation function $\tau : I \rightarrow \mathcal{I}$. Thereby, a change between different instances of the parameter space can be accomplished fluently. This allows for acceleration and deceleration as well as for fading individual motion techniques.

The function δ for determination of a concrete motion parameterisation results as tuple $(\mathcal{M}, \mathcal{I}, \psi, \tau)$, whereby $\mathcal{M}$ names the set of all available motion techniques according to section 2.2, $\mathcal{I}$ names the set of all currently defined time intervals, ψ and τ correspond to the above mentioned mapping functions.

Any evaluation of the user's target function as defined in 4.1 is done analogical to Reddy [20]. Individual parameters of the user model are evaluated by Reddy in order to determine an area of interest. This area is used to limit the effects of some rendering aspects for the purpose of runtime optimisation. For the purpose of motion parameterisation the user's target function is to be evaluated in order to allow for the best possible interaction support. A main interaction requirement is the determination of object location [19]. The restriction of motion in accordance with the remaining system state is explicitly required.

4.3 Exemplary interval set

Fig. 5 shows an exemplary set of presentation methods and their scheduling order throughout the time of a session of an information fusion process execution. The bars represent several used motion techniques, noted as function $m(cat)$ with cat representing the motion catalog as introduced in section 3.2. Any interpretation of the parameter space P is left out of the figure for brevity.

After a preparation period of the session, the execution of the fusion process is started at time $t_0 = start$ and finished at time t_{10} because of a terminating user interaction event. Beginning at t_1, the user is presented with visual representations of gained results of executed fusion operators. Some results might not be represented visually but only be used internally. An example for such case is result set r_3.

As time advances and process execution progresses, the parameterisation of any used method m is subject to change. As an example changes the visualisation of result set r_1 at time t_2 because of internal process events. User interaction is thereby not involved. On the other hand leads an interaction event at $t_7 = act_2$ to the extraction of result set r_4 and its immediate presentation. More process execution time is obviously needed in order to gain r_5 after user action happening at $t_5 = act_1$. The visualisation of result set r_n changes multiple times throughout the whole session depending on both—user interaction and fusion process events.

All of r_1, r_2, r_4, and r_5 represent intermediate result sets that are discarded and not used for final information gathering. Only r_2 is discarded explicitly by the user at time t_{10} (which is equivalent to act_3).

4.4 Specification

In classical notation a signature is specified as tuple $SIG =< S, OP, REL >$ consisting of the set S, the set family $OP = (OP_{\omega,s})_{\omega \in S^*, s \in S}$, as well as the set family $REL = (REL_\omega)_{\omega \in S^*}$. The elements of S specify sorts (types). S^* names the set of all sort chains. For each $\omega \in S^*$ and each $s \in S$ is $OP_{\omega,s}$ the set, whose elements op name the possible operations. These are therefore mappings $op : S^* \times S \rightarrow S$. For each $\omega \in S^*$ the set of possible relations is given by REL_ω.

To ease the notation, this syntax is modified to use a schematic presentation instead. All sorts are listed after the key word *sorts*. Operation declarations follow the key word *opns*, relations follow the key word *rels*. In case λ names the empty word the term $c \in OP_{\lambda,s}$ is reduced to $c :\rightarrow s$. Similarly is $op : \omega \rightarrow s$ used instead of $op \in OP_{\omega,s}$ (with $\omega \neq \lambda$) and $rel : \omega$ instead of $rel \in REL_\omega$.

The specification of the model is given as tuple $SPEC =< SIG, X, REQ >$. It consists of a signature SIG, a set of variables X, and a set of formulas $REQ \subset$

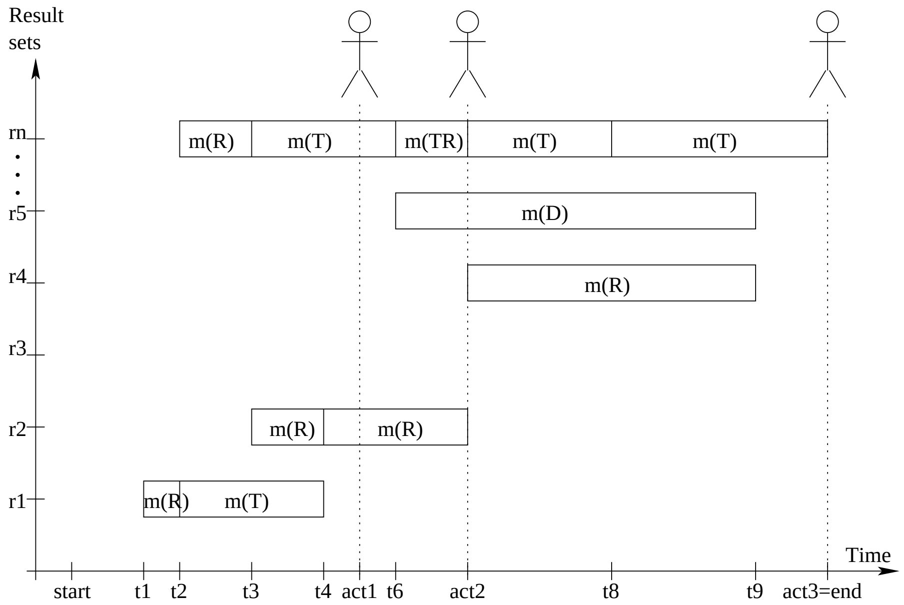


Fig. 5. Scheduling of presentation methods during fusion process execution. Time points t_5, t_7, and t_{10} are defined by user interaction events and are noted as act_1, act_2, and act_3 respectively. Note that result set r_3 does not lead to any visual display and that only one result set (r_n) is valid until the final user event.

$F_{SIG}(X)$ that denote the algebra of the specification. For the schematic notation of the specification, the syntactical representation of the signature is continued. The set of variables follows the key word *vars*, the requirements of the algebra follow *reqs*.

Temporal motion parameterisation $\delta(\mathcal{M}, \mathcal{I}, \psi, \tau)$
sorts: I, $\mathcal{I}$, E, $\mathcal{M}$, $\mathcal{W}_{rep}$
opns: $\psi : E \rightarrow I$
$\tau : I \rightarrow \mathcal{I}$
$+ : \mathcal{I} \rightarrow \mathcal{I}$
rels: $<, =, >: I\ I$
$\subset, \subseteq, =: \mathcal{I}\ \mathcal{I}$
vars: $e_1, \ldots, e_n : E$
$I_1, \ldots, I_n : \mathcal{I}$
$m_1, \ldots, m_n : \mathcal{M}$
$world^1_{rep}, \ldots, world^n_{rep} : \mathcal{W}_{rep}$
reqs: $\psi(e_n) < \psi(e_m) \Longleftrightarrow e_n < e_m$
$\tau : \forall I_i \in \mathcal{I} \Longrightarrow I^b_1 \leq I^b_i \wedge I^e_n \geq I^e_i$ (limit of interval borders)
$I_a < I_b \Longleftrightarrow I^b_a \geq I^b_b \wedge I^e_a < I^b_b$
$I_a = I_b \Longleftrightarrow I^b_a = I^b_b \wedge I^e_a = I^e_b$

$$I_a > I_b \iff I_a^b \leq I_b^b \wedge I_a^e > I_b^e$$
$$\mathcal{I} + \lambda = \mathcal{I}$$
$$\mathcal{I}_a = \{I_1, \ldots, I_a\} \wedge \mathcal{I}_b = \{I_{a+1}, \ldots, I_b\} \Longrightarrow \mathcal{I}_a + \mathcal{I}_b = \{I_1, \ldots, I_b\}$$
(continuity of intervals)
$$m_a \neq m_b \iff world_{rep}^a \neq world_{rep}^b$$

5 Conclusion

A vital requirement for interactive systems supporting information fusion is the representation of time. This paper motivates the use of a motion enhanced visualisation toolkit for this purpose. A formal model is presented for mapping temporal events onto motion parameterisation. Using the presented architecture of a temporal mapping component allows to bridge the gap between the model, an implementation of its algebra, the toolkit, and an information fusion environment. A throughout implementation of all mapper components remains as challenge for the ongoing work.

6 Acknowledgements

This work is supported by the German Research Council (Research Grant No. STR 398/4). Special thanks are due to Axel Panning for parts of the implementation work.

References

1. James F. Allen. Time and Time again: The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4):341–355, 1991.
2. Lyn Bartram, Colin Ware, and Tom Calvert. Moving Icons: Detection And Distraction. In *Proceedings of Interact 2001*, Tokyo, 2001.
3. Claudio Bettini, Sushil Jajodia, and Sean X. Wang. *Time Granularities*. Springer-Verlag, Berlin Heidelberg New York, 2000.
4. Herman Chernoff. The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of American Statistical Association*, 68:361–368, 1973.
5. Oliver Dunemann, Ingolf Geist, Roland Jesse, Gunter Saake, and Kai-Uwe Sattler. InFuse – Eine datenbankbasierte Plattform für die Informationsfusion. In Andreas Heuer, Frank Leymann, and Denny Priebe, editors, *Datenbanksysteme in Büro, Technik und Wissenschaft, BTW 2001*, pages 9–25, Oldenburg, 2001. Springer.
6. Oliver Dunemann, Ingolf Geist, Roland Jesse, Kai-Uwe Sattler, and Andreas Stephanik. A Database-Supported Workbench for Information Fusion: Infuse. In Christian S. Jensen, Keith G. Jeffery, Jaroslav Pokorny, Simonas Šaltenis, Elisa Bertino, Klemens Böhm, and Matthias Jarke, editors, 8^{th} *International Conference on Extending Database Technology (EDBT) Proceedings*, Lecture Notes in Computer Science, pages 756–758. Springer, March 2002.
7. Jim Foley and Bill Ribarsky. Next-generation data visualization tools. In L. Rosenblum, R. A. Earnshaw, J. Encarnação, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization*, chapter 7, pages 103–127. Academic Press, 1994.

8. Christian Freksa. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 52(1-2):199–227, 1992.
9. Christian Freksa. Using Orientation Information for Qualitative Spatial Reasoning. In A. U. Frank, I. Campari, and U. Formentini, editors, *Proceedings Int. Conference on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Berlin, 1992. Springer.
10. R. Galetto and F. Viola. Time as the fourth dimension of the topographic databases. In *Mapping and Geographic Information Systms*. International Society for Photogrammetry and Remote Sensing, 1994.
11. Donald Hearn and Polly Baker. Scientific Visualization Tutorial Notes. In *Eurographics '91*, Eurographics Technical Report Series, 1991.
12. Chung-Ming Huang and Ming-Yuhe Jang. Interactive Temporal Behaviours and Modelling for Multimedia Presentations in the WWW Environment. *The Computer Journal*, 42(2):112–128, 1999.
13. Roland Jesse and Thomas Strothotte. Motion Enhanced Visualization in Support of Information Fusion. In Hamid R. Arabnia, editor, *Proceedings of International Conference on Imaging Science, Systems, and Technology (CISST'2001)*, pages 492–497. CSREA Press, June 2001.
14. Gunnar Johansson. Perception of motion and changing form. *Scandinavian Journal of Psychology*, 5(3):181–208, 1964.
15. Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):201–211, 1973.
16. Haim Levkowitz. Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Parameters. In *Proceedings Visualization '91*, pages 164–170, Los Alamitos, 1991. IEEE Computer Society, IEEE Computer Society Press.
17. R. M. Picket and G. Grinstein. Iconographics Displays for Visualizing Multidimensional Data. In *Proceedings IEEE Conference on Systems, Man and Cybernetics*, pages 514–519, Beijing and Shenyang, 1998.
18. Zenon W. Pylyshyn, J. Burkell, B. Fisher, C. Sears, W. Schmidt, and L. Trick. Multiple parallel access in visual attention. *Canadian Journal of Experimental Psychology*, 1993.
19. Andreas Raab and Thomas Strothotte. Interactively Exploring Geometric Models. In *Proc. 8th International Conference on Engineering Computer Graphics and Descriptive Geometry*, pages 745–749, Austin, TX, July 31 - August 3 1998.
20. Martin Reddy. Perceptually Optimized 3D Graphics. *IEEE Computer Graphics and Applications*, 21(5):68–75, September/October 2001.
21. Felix Ritter, Thomas Strothotte, Oliver Deussen, and Bernhard Preim. Virtual 3D Puzzles: A New Method for Exploring Geometric Models in VR. *IEEE Computer Graphics and Applications*, 21(5):11–13, September 2001.
22. Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, 2. edition, 1998.
23. Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics. Modeling, Rendering, and Animation*. Morgan Kaufmann, San Francisco, 2002.
24. Colin Ware. *Information Visualization: Perception for Design*. Interactive Technologies. Morgan Kaufmann Publishers, 2000.
25. Pak Chung Wong and R. D. Bergeron. 30 Years of Multidimensional Multivariate Visualization. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 3–33. IEEE Computer Society Press, Los Alamitos, 1997.