A Compositional Model for Multidimensional Data Visualisation

Rajehdra Nagappan†
Department of Computer Science, The Australian National University

ABSTRACT
This paper presents a model for constructing complex visualisation instances by describing them as a set of smaller interconnected modules. The module-relationship structure of the model allows users to explore a given visualisation instance a piece at a time, and then to relate modules to one another to explore the data more deeply. This is known as compositional visualisation. Each module is a visualisation in itself and represents some aspect or view of the data. When a number of such smaller visualisation views are considered conjunctively the result is a broader view of the data that includes the aspects provided by each module. Compositional visualisation is one technique for dealing with data that is too large or complex to visualise using a single visualisation. The model first decomposes data into a collection of simpler data modules. The data modules are then mapped to simple visualisation modules. The visualisation modules are combined to form larger visualisation instances. However, the decomposition of data is not necessarily equivalent to the composition of visualisation, thus a visualisation may give a false impression of data if poorly constructed. The reasons for this and ways to overcome it are presented.

Keywords: Information visualisation, data visualisation, multivariate, multidimensional, visualisation theory, visualisation model, data mining.

1. INTRODUCTION
Data mining is the process of searching for valid, novel, potentially useful and ultimately understandable patterns from a given data collection. Patterns are parts of the data that are interesting, e.g. clusters, outliers or dependency relationships. Visualisation is a technique that can be used for data mining; a visualisation tool represents data in a graphical fashion and the user searches for interesting graphical regions. However, contemporary data collections are large and complex, visualisations need to be quite sophisticated to adequately present such data. For example, multidimensional data consists of many variables or attributes but humans can only perceive three spatial dimensions, so how can multidimensional data be visualised?

One approach that has received attention in recent years is the use of multiple simple visualisations which are related to each other. Given that data is too complex to be visualised in a single go the approach breaks up the data into smaller manageable pieces. Each piece is visualised separately and the resulting visualisations are compared with each other. Roberts18, 19, 20 extended Haber & McNab’s visualisation pipeline model9 to allow for multiple visualisations or views. Figure 1 shows the model. Different visualisations can be created by applying different processes at any of the process stages of the original pipeline model. The same raw data can be filtered to differing derived data sets. The same derived data can be mapped to differing Abstract Visualisation Objects (AVO). The same AVOs can be rendered differently to produce different images.

North presents an interaction model that explicitly links interactions between multiple views15. The model is based on a system of relations and actions. Each object in any view is a entry of a relation; the relation links the object between different views. An interaction with an object in one view causes the object to be looked up in the relation. The object’s entry will stipulate a set of actions and targets for those actions. The actions are carried out on their respective targets by the system. Thus affecting a change to an object in one view updates it in all other views. For example, consider a frame-based HTML page in a web browser. Two frames exist, one is a table of contents and the other contains a page of text. Clicking on an item in the table of contents causes text related to that item to be displayed in the other frame. The ‘click’ interaction on the contents object has caused a ‘load’ action on some specific text in the target frame. User trials have shown that these tightly-coupled coordinated visualisation methods allow users to perform a variety of elementary data analysis tasks in 30-80% less time than when using only a single view. Additionally, users reported a satisfaction rating for the coordinated visualisations approximately double that of the single view methods.

† Correspondence: R. Nagappan, Dept. of Computer Science, CSIT Building 108, The Australian National University,
CANBERRA ACT 0200, Australia. Email: Raj.Nagappan@cs.anu.edu.au
Recently (within the last twelve months) Baldonado et. al. noted that developing models for the use of multiple visualisation units is an area yet to be explored. This paper presents such a model. This model builds on previous research into formal models of compositional visualisation. 

2. THE COMPOSITIONAL MODEL OF VISUALISATION

The Compositional Model of visualisation describes a complex visualisation instance as a set of related visualisation modules. It describes visualisation instances as clustered graphs and data as clustered directional graphs. For clarity visualisation and data are first presented as a tree-like hierarchical structure which possesses parent-child and sibling relationships between modules. Later the graph view is described, where nodes are modules and edges are relationships between modules. The model supports both: combination of visualisation modules into larger compositional visualisation modules; and navigation between visualisation modules. The model relies on relations as its basic building block.

Figure 2 shows an overview of the model. Data is decomposed from its original state to a collection of simpler components. Each of these are then mapped to visualisations. The individual visualisations are composed together to form a larger, more complex visualisation instance. However, the final visualisation instance is not necessarily an accurate depiction of the original data. The reasons for it and strategies to overcome it will be described in this paper.

3. RELATIONS

This section defines relations as used in this paper. This model is based on Ullman, Humphrey and Aho & Ullman. Relational data is observation based and is typical of many multidimensional data collections.

3.1. Relations Defined

A domain d is a set of values. An attribute a is a label that names d. A value for an attribute a actually refers to the value in its corresponding domain d, whilst the name of a refers to the label. Attributes give context to domains. For example, the names ‘price’ and ‘quantity’ are attributes that give context to the domains they refer to. Attribute names form a basis for many operations in data analysis.

A tuple t in domains d₁, d₂, …, dₙ is an element of their Cartesian product, i.e. \( t \in d₁ \times d₂ \times \ldots \times dₙ \iff t = (tᵢ| tᵢ \in dᵢ)ᵢ=1^n \). t is also known as a datum or entry.

A relation R between domains d₁, d₂, …, dₙ is a set of tuples in those domains, i.e. \( R = \{ t | t \in d₁ \times d₂ \times \ldots \times dₙ \} \). Formally, \( R \subseteq d₁ \times d₂ \times \ldots \times dₙ \). The set of attributes \( \{ aᵢ | aᵢ = attr(dᵢ) \}ᵢ=1^n \) is the relation scheme of R and each \( aᵢ \) is referred to as an attribute of R. n is the arity or degree of R. Each domain-attribute pair (dᵢ, aᵢ) is a degree of freedom and is referred to as a dimension of R. A relational database is a set of relations. A key of R is a set of attributes \( \{ aᵦ \}ᵦ=1^k \) in R such that: (a) no two distinct tuples in R can both have the same values for every attribute in \( \{ aᵦ \}ᵦ=1^k \); and (b) no proper subset of \( \{ aᵦ \}ᵦ=1^k \) satisfies (a). Relations are often thought of as tables; each attribute defines a column and each tuple fills a row.
3.2. Operations on Relations

There is a wide variety of operations for relational databases using relational algebra and relational calculus. The operations required by the Compositional Model are shown here.

The projection of a relation $R$ removes or rearranges particular attributes of all tuples. Projection is given by $\text{proj}_{i_1, i_2, \ldots, i_m}(R)$ where $i_1, i_2, \ldots, i_m$ is the ordered list of attributes from $R$ to preserve. Each tuple $t$ in $R$ is converted to a new tuple $t'$ in a new relation $R'$ where $t' = (t[j \in d_j]_1^n)$ and each $d_j$ is the domain corresponding to attribute $a_j$. $R' = \text{proj}_{i_1, i_2, \ldots, i_m}(R) \subseteq d_1 \times d_2 \times \ldots \times d_m$. Projection can be used in many ways. Consider a relation $R$ with attributes $a_1, a_2, \ldots, a_m$. $\text{proj}_{a_1, a_2}(R)$ preserves the first two attributes and discards the rest. $\text{proj}_{a_m, \ldots, a_2, a_1}(R)$ reverses the order of attributes but preserves all of them.

Selection from a relation $R$ preserves all tuples satisfying a logical condition and discards all others. Selection is given by $\text{sel}_{\text{cond}}(R)$ where $\text{cond}$ is the test condition. Selection is used to partition a relation. For example, $\text{sel}_{(a_1 < 5) \land (a_4 > 7.6)}(R)$ selects from relation $R$ all tuples who possess both a value for attribute $a_2$ which is less than 5 and a value for attribute $a_4$ which is greater than 7.6.

Let $R$ and $S$ be relations of degree $m$ and $n$ respectively. The Cartesian product $R \times S$ is the set of all tuples of degree $m + n$ whose first $m$ attributes form a tuple in $R$ and whose last $n$ attributes form a tuple in $S$.

The join operation between relations $R$ and $S$, $\text{join}_{\text{cond}}(R, S)$, is equivalent to $\text{sel}_{\text{cond}}(R \times S)$. It selects tuples satisfying $\text{cond}$ from the Cartesian product of $R$ and $S$. The natural join, $\text{join}(R, S)$, operates as follows:

1. Compute $R \times S$.
2. For each attribute $a$ occurring in both $R$ and $S$ select those tuples whose values agree for both $R.a$ and $S.a$, where $R.a$ and $S.a$ are the attributes in $R \times S$ corresponding to attribute $a$ in relations $R$ and $S$ respectively.
3. For each above attribute \( a \) form a projection which removes \( S.a \) but preserves all other attributes. Call the remaining attribute \( R.a \) by the common name \( a \).

Formally, if \( a_1, a_2, \ldots, a_k \) are all the attributes occurring in both \( R \) and \( S \), then

\[
\text{join}(R, S) = \text{proj}_{a_1, a_2, \ldots, a_k}(\text{sel}_{(\exists x_k \equiv s_{x_k}) \land \ldots \land (\exists x_1 \equiv s_{x_1})}(R \times S))
\]

where \( i_1, i_2, \ldots, i_m \) is the ordered list of all attributes in \( R \times S \) except \( S.a_1, \ldots, S.a_k \). The join operation is useful for combining a number of relations into a single relation. For example, consider a number of relations which each contain different types of measurements at unique locations in the earth’s crust where the set of locations is common to all relations. The group of relations could be joined to form a single relation that contains every measurement at every location.

4. DATA MODULES

Data is a collection of datums. A data relation is a relation between domain-attribute pairs that have some meaning in the problem domain, e.g. price, quantity or date. A data set is a set whose members are data relations or other data sets, also known as a data module. Data relations and sets are not static; they can be iteratively or interactively altered. New relations can be created or existing relations can be changed using operations such as those described in the previous section. The sets can be altered using set algebra. Figure 3 shows some data modelled as a collection of modules.

![Figure 3. Data collection as modules. Typical data collection is a set of both relations and sets of relations nested within it.](image)

4.1. Decomposing Data

Typically an initial data collection will be too large, complex or otherwise unwieldy to visualise directly. It must be transformed into a collection of data modules that are each somehow simpler than the original data. The transformation into smaller or simpler modules is called decomposition. Figure 4 shows a typical decomposition operation; a set of complex relations at the top of the figure are transformed to a collection of sets of simpler relations below. Each of the leaf relations in the tree are sufficiently simple that they can be readily visualised.

5. VISUALISATION MODULES

A visual relation is a relation between visually perceivable domain-attribute pairs, e.g. size, colour, location or texture. Figure 5 shows that the logical and visual representation of a visual relation are equivalent. Indeed, when a user views a visualisation they see the visual representation but they interpret the logical representation.

Visual infrastructure that provides context for a visual domain, e.g. a legend or positional axis, is considered to be part of the attribute for that domain. Subsequently it forms part of the schema of the visual relation. Other types of visual infrastructure such as window decorations are not considered within this paper. There may be attributes of a visualisation instance that do not encode any information. For example, a scatterplot that encodes attributes using only horizontal and vertical position requires a visual mark for every entry. The mark only encodes data using its vertical and horizontal positions; its other properties such as size, shape and colour are visual artefacts. A simple way to distinguish artefacts from ‘true’ attributes (in the sense that a true attribute encodes legitimate data) is that an artefact’s domain has a size of one, e.g. \( \text{markSize} = 1 \). Observation of such artefacts reveals that their purpose is only for indication. Any visual artefact is a piece of visual infrastructure of a visualisation instance and is not dealt with further.

A visualisation module is either a visual relation or a set of visualisation modules; the former is an elementary visualisation module while the latter is a compositional one. Figure 6 shows a number of visualisation modules; each view window is an
elementary module while the set of three windows form a compositional module. An elementary visualisation module is any visualisation that sufficiently simple and understandable that decomposing it into a set of smaller visualisations would be unnatural or useless. Each view window in Figure 6 is not easily decomposed into a set of simpler but still useful visualisations, hence each is elementary. Guidelines for forming effective elementary modules remain outside the scope of this paper. Examples of effective construction techniques include work by Bertin\(^3\), Mackinlay\(^5\), Casner\(^5\), Cleveland & McGill\(^6\) and Senay & Ignatius\(^7\).

A compositional visualisation module binds together a number of other visualisation modules into a larger, more complex, single unit. The composed module can be used either as a set of (possibly related) parts or as a whole. Section 7 discusses the mechanics of composition. The elements of a compositional visualisation are either relations or relations nested inside sets (where the set is another compositional module). Thus a compositional module is a hierarchically nested set of relations. Figure 7 shows two levels of composition into larger modules.

A visualisation module forms a frame of reference. A frame of reference is a cohesive unit that contains enough information and is sufficiently separable from its surroundings that it can independently exist. A group of visualisation modules formed from some common data, e.g. the view windows in Figure 6, can provide a variety of perspectives of that data. Salzmann et. al. found that multiple frames of reference assisted users to understand complex data, since each frame could be used to highlight different properties\(^1\). A compositional visualisation module encapsulates other frames of reference (where each such frame is a child module), an elementary module does not.

**Figure 4.** Data decomposition.
Data is typically transformed from collections of complex relations to collections of simpler relations.

**Figure 5.** The logical and visual representations of a visual relation are equivalent.
The left side shows a relational view of visualisation, the right side shows a semiological view. The two views are equivalent.
Figure 6. Multiple frames of reference.
Each view window is a frame of reference and is an elementary visualisation module. The group of three windows is a larger frame of reference that encapsulates the former three, and is a compositional visualisation module. This is an AutoCAD example.

Figure 7. Compositional visualisation module.
Visualisation modules are composed into larger compositional visualisation modules.

6. MAPPING DATA MODULES TO VISUALISATION MODULES
A mapping \( f: d_S \rightarrow d_T \) is a function from a source domain \( d_S \) to a target domain \( d_T \). The function is a relation \( F \subseteq d_S \times d_T \) such that for all \( x \in d_S \) there exists one and only one \( y \in d_T \) such that \( (x, y) \in F \).11.

Data modules are mapped to visualisation modules by mapping each data relation in the data module to a corresponding visual relation in the visualisation module. Each element in a data relation is mapped to an element in the corresponding visual relation using the mappings defined between them, as shown in Figure 8. Commonly separate mappings operate on each attribute of each tuple independently, but it is possible that mappings operate on combinations of attributes. However, this raises the possibility that two or more mappings of combinations of attributes which possess intersecting attribute sets may produce different values for the same attribute. Such cases are outside the scope of this paper.

7. CORRELATING VISUALISATION MODULES
All the operations defined over relations and sets can be used when required. For example, if two visual relations contain the same attributes of two types of products those two relations can be unioned to produce a single visualisation that shows the said attributes of both products concurrently. Alternatively if a region of interesting points is present in a visualisation we can
Figure 8. Mapping data elements to visual elements.
Each entry in a data relation is mapped to an entry in a visual relation using the mappings defined between the data attributes and visual attributes. Select those points and discard the rest. Mackinlay\textsuperscript{10} and Senay & Ignatius\textsuperscript{17} defined a specific set of composition operations, e.g. single axis composition and mark composition. The Compositional Model, however, is much more flexible since it supports all relational algebra.

Correlation between relations is achieved by a natural join operation between them. Typically in multidimensional data visualisation a group of visual relations will each encode different projections of multidimensional data; those projections need to be correlated with each other in order to understand the data. However, the correlations between visual relations may be ambiguous — this ambiguity will produce a different result to what was desired. This can have a profound impact on the design of a visualisation instance and on our understanding of data; what we expected is not necessarily what turns up. The following subsections discuss correlations in greater detail.

7.1. Full Commonality Between Attributes
In this subsection we consider cases where visualisations have common visual attributes, common mappings and common data attributes, i.e. the visual attributes are equivalent. Equivalence of visual attributes departs from the usual notions of relational algebra. Consider two attributes \( v_1 \) and \( v_2 \) of visual relations, where \( v_1 \) and \( v_2 \) are mapped by \( f_1 : a_1 \rightarrow v_1 \) and \( f_2 : a_2 \rightarrow v_2 \) respectively and \( a_1 \) and \( a_2 \) are attributes of data relations. \( v_1 = v_2 \) if: \( v_1 \) and \( v_2 \) represent the same visual attribute of the display (e.g. colour, size or position); \( f_1 = f_2 \); and \( a_1 = a_2 \). This definition has a significant impact on a number of relational operations, especially natural join. Attributes are not considered to be common if they merely give the same visual appearance; rather they must encode the same attribute of the source data.

Figure 9 shows a pair of scatterplots with no common attributes. When we correlate the two we form a natural join between them; this creates multiple connections between tuples in each scatterplot. Which mark in the left scatterplot corresponds to which in the right? There is no way of telling, the relation for the join between these would show tuples of degree four such that every tuple in the left scatterplot is combined with every tuple in the right. Subsequently the visualisation shows every such combination ‘superimposed’ on each other.

Figure 9. Visualisations with no common attributes.
Which mark in the left visualisation corresponds to which in the right? There is no way to tell as there is no commonality between the two.
However, if there were some common attributes, as in Figure 10 which shows a common attribute on the vertical scale, much of this multiplicity is removed. The natural join operation selects only those tuples whose values for the common attribute in both scatterplots agree. The common attribute is a correspondence between the two scatterplots. Note however that two marks in each scatterplot possess identical values for this common attribute and their correspondences between scatterplots are still not distinguishable from each other.

![Figure 10. Visualisations with a common attribute.](image)

The vertical attribute of the two visualisations is common. It can be used to correlate marks between them. However two marks near to the top of the figure contain identical values for the common attribute and thus their correlations are not distinguishable.

The observation that correspondence between modules relies on a natural join between stems from the following task: “for each tuple in module $x$ containing value $m$ for attribute $a$, find the tuple or tuples in module $y$ that also contain value $m$ for attribute $a$, where $a$ is common to $x$ and $y.”$ This task attempts correlate tuples between modules using common properties. To perform this operation combine all the candidate tuples, select those for which the values in the common attribute agree and finally remove the duplicity of the common attribute via projection — i.e. a natural join operation.

More common attributes between visual relations leads to clearer correspondences between them. In particular if a common key is present in the set of common attributes then a one-to-one correspondence between tuples in each relation is guaranteed. The tricky part when forming visualisations consisting of multiple modules is to construct them such that when they are joined they faithfully reproduce the intended relation rather than a new ‘false’ relation through ambiguous correlations.

This does not take into account the effects of different placement patterns in the display. For example, given the scatterplots in Figure 10, there may be a perceptual difference if the common attribute between them were not aligned. Consideration of such perceptual factors in not within the scope of this paper.

7.2. Partial Commonality Between Attributes

In this subsection we consider cases where visualisations have different visual attributes that stem from common data attributes or different mappings between common data attributes and common visual attributes. These two cases are the same, since the mapping is where the differences arise; in the first case the mapping goes to a different destination while in the second case the mapping mechanics perform differently.

To combine visual relations with attributes that are partially common first perform a natural join on the relations as given in the previous subsection, during this stage only attributes with total commonality are considered. Next form an equivalence relation for the attributes that are partially common. This will allow correlation between values that are different in the visualisation instance but stem from the same source data values. Next select from the joined relation those tuples whose values for the partially common attributes agree using the equivalence relation as a legend. These are partial correlations.

Partial correlations induce a greater cognitive load on the user as they must do significant work to match values. This is manifest by the creation of the equivalence relation which is required for coordination between the source visual relations. Additionally equivalences may be present that are mathematically detectable using the relation but not perceptually detectable using the visualisation instance. This can lead to error or ambiguity. Hence partial correlation is generally not as favourable as total correlation (notwithstanding specific task requirements in particular instances).
7.3. Brushing between Modules

Brushing is commonly used to link entries in visualisations. A brush is a tool used to select data points of interest. The selected points are highlighted in all visualisation modules concurrently. The brush highlight correlates the satisfying points between modules. The correspondences inferred by the brush are still natural joins, but the brush has the effect of an interactive selection system. It is easy to tell that brushed values in each module correspond and that non-brushed values correspond. However, as was pointed out for the natural join operation, there is no way to tell which brushed value in one visualisation module corresponds to which brushed value in another (unless other correspondences make this clear). This is because the brush is just another attribute — it reduces the multiplicity in the natural join but because it is binary it rarely forms a key (the exception being when only one entry is brushed). Thus brushes are not perfect correlators; they do help but some ambiguity may still remain.

8. DATA AND VISUALISATION INSTANCES ARE CLUSTERS OF GRAPHS

So far the data and visualisation structures have been presented as trees. However, two or more visualisation modules may have common attributes regardless of whether or not they are siblings in a compositional module. The correlation relationships described in the previous section can occur between different parts of the hierarchy.

The visualisation side of the Compositional Model thus is a clustered directed graph. A clustered graph is a graph whose nodes are recursively partitioned into groups\(^5\). Each group of nodes is a subgraph. Each subgraph may have both edges within itself and edges connecting it to other subgraphs. A node in the graph is an elementary visualisation module. An edge is the set of common attributes between the nodes it connects. A compositional visualisation module is a subgraph whose nodes are its child modules. Figure 11 shows a graph view of a number of visual relations. These could be combined into a compositional module being this subgraph.

![Figure 11. Graph view of the Compositional Model. Visual relations are nodes. Common attributes are edges.](image)

The data side of the Compositional Model is a clustered directed graph. New data relations are formed by applying operations to existing relations; thus there is only a feed-forward relationship. Nodes in the graph are data relations and directed edges are the operations between relations. Subgraphs are data sets (sets of data relations).

The graph shows the path that must be travelled to move attention from one module to another, and thus for navigating around the entire visualisation instance. Edges containing many attributes or attributes of high correlation quality are ‘mentally’ easy transitions. Edges containing few or ambiguous attributes are ‘mentally’ hard transitions. Mental in this sense refers to relative difficulty of shifting focus using a given strength of correlation.

The graph allows focus and movement. The module that the user is currently engaged with is the region of focus in the visualisation instance. Navigating through to a different module is an act of movement through the visualisation structure. A user can only interpret a module of finite size or complexity at any given time, thus limiting the scope of their focus. Additionally, they can only interpret new information at a finite rate, thus limiting the speed of transition from one module to the next and hence limiting the speed of movement throughout the graph. The aim in creating an effective visualisation instance is thus to
encapsulate a number of visualisation modules which possess the most effective common attribute sets and maximal coverage of the overall data (notwithstanding that a particular task may be the primary driver of a visualisation instance.)

9. TASK GUIDES VISUALISATION DESIGN

The tasks that a user needs to perform guide visualisation design. Effective visualisations are those that are focused on performing the task or tasks that the user requires. Unlike Casner’s task description language, task in the Compositional Model is defined as those things that the user is interested in, and exists only in their mind.

Tasks exhibit a hierarchical structure like that of data or visualisation, this structure was described by Bowman & Hodges. For example, to find to a multidimensional cluster first find a cluster in each dimension separately, then correlate those points in each such cluster together to determine whether or not they stem from the same source data points. A multidimensional cluster is present only if the same points are clustered in all required dimensions.

Thus task structure mirrors visualisation structure, as shown in Figure 12. Tasks are bound to visualisation modules. More than one task can be bound to a given module. If a task can be broken into subtasks then it is likely to require a compositional visualisation module; the subtasks are bound to each of the children of the module. A successful visualisation strategy will compose visualisation modules such that each child of a compositional module satisfies a separable subtask bound to it and interaction between the children satisfies the task bound the module.

![Figure 12. Task structure mirrors visualisation structure. Task structure mirrors the structure of the visualisation instance. Tasks that can be broken into subtasks are bound to compositional visualisation modules. Tasks that are elementary are bound to elementary visualisation modules.](image)

Tasks change — requirements change, perceptions change, interests change, and so on. Therefore a visualisation instance needs to adapt to this change. In the Compositional Model this occurs via interaction with a visualisation instance or iteration of instance design. Change produces a different task structure and hence a different visualisation structure. Interaction alters an existing instance to meet new tasks, whilst iteration produces a new instance that conforms to new tasks. Frequently in data mining users will not know their specific tasks apriori, so a number of iterations of design or interaction will be executed as the tasks are identified and shaped.

Since visualisation composition is not equivalent to data decomposition users must choose the tasks that they want to complete and design a visualisation instance accordingly. Each visualisation module will emphasise some properties and neglect others — the task or tasks bound to it determines which are emphasised and which aren’t. This is significant for both elementary and compositional visualisation modules. In elementary modules the task will determine how data attributes are mapped to visual attributes. In compositional modules the task will determine how children must relate to each other. For example, if it is important that tuples are individually correlated between modules then they will need to be structured so that their join allows clear correlations of each entry. However, if the individual tuples are not important but a binary separation into clustered and not clustered is needed then the same strength of association is not required; if we can correlate a group of tuples in one module with a corresponding group in the others, e.g. by brushing, this is sufficient.
The example above illustrates the trade-off in compositional visualisation: clarity of correlations versus breadth of information. Bertin noted that any visualisation module is limited to a fixed number of effectively discernable attributes\textsuperscript{12}, but earlier we noted that more common attributes leads to stronger correlations. If we chose to clearly correlate the modules then the total number of attributes visualisable is relatively small; however if we choose to visualise many attributes concurrently then the correlations between modules are weak. The task is the deciding factor: for tasks which require strong correlations use the former arrangement whilst for tasks which require a great breadth of information use the latter.

10. CONCLUSION

This paper presented a model for visualisation by composition. Data is decomposed into smaller data relations, the data relations are mapped to visual relations and the visual relations are composed into larger visualisation modules. The composition of visualisation modules may not necessarily reproduce specific properties in the data; relationships of common attributes between modules play a critical role in determining the quality of composition and hence whether these properties are detectable. It is generally not possible to faithfully reconstruct all properties of multidimensional data; the task at hand decides which properties are preserved and which are not.

There are three principles uses for such a model. First, to analyse existing visualisation tools, this will show what types of explorations can be effectively performed. Second, to develop new visualisation tools, e.g. the Multidimensional Data Orb\textsuperscript{14}. Third, to develop visualisation construction toolkits that model visualisation structure. Due to space limitations evaluation and application of the model were not presented here (they have been performed and will be published in the future).

ACKNOWLEDGEMENTS

Thanks to Mark Grundy and Matthew Hutchins for their comments and contributions. Thanks to the Cooperative Research Centre for Advanced Computational Systems under which this work is funded.

REFERENCES