— Metamorphers —
A Formalization of Transitions in Illustrative Molecular Visualization

A, B, and C

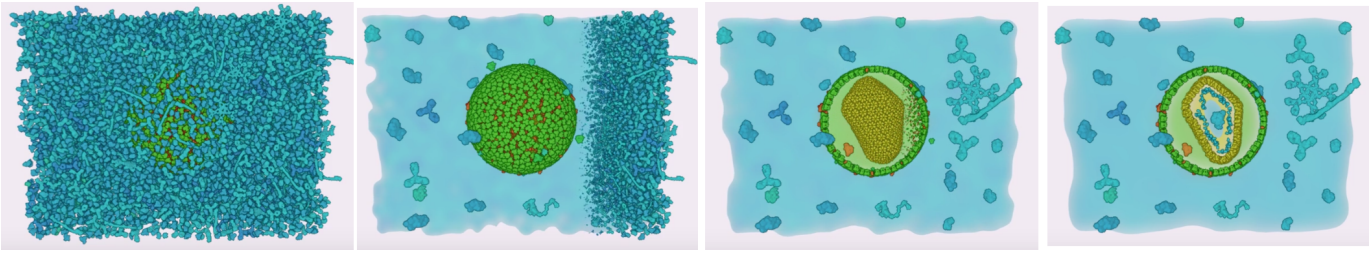


Fig. 1: Metamorphosis to a schematic representation of an HIV particle in blood serum.

Abstract— Illustrators in molecular biology often use animated transitions as a means to communicate complex phenomena. While these animations are popular, they still have to be created manually in 3D modeling software. We therefore provide illustrators with a more streamlined way for creating such transitions between various representations of molecular data.
We propose a formalization of the process of specifying transitions to arbitrary representations in the form of a conceptual pipeline. Data representations are specified in such a way that they form a continuous space. The first three stages of the pipeline are responsible for creating this continuum. The final four stages of the pipeline are responsible for presenting the continuum. This continuum can be arbitrarily sampled in order to create a transition between two representations, e.g., densely, in the form of animation, or sparsely, in the form of small multiples.
We demonstrate our concept on exemplary use cases by creating multiple operators, so called metamorphers, for each pipeline stage. These metamorphers can be combined to achieve different types of transitions for arbitrary molecular data sets. We present three transitions that we designed in collaboration with illustrators, based on the data set of HIV: the explosion of inner and outer cellular compartments, a novel illustrative rendering approach for automatic schematization of mesoscopic data, and a transition to an infographic view, conveying quantitative relationships. In a qualitative discussion of our results, we received positive feedback from domain experts in illustration.

Index Terms—Molecular Visualization, Illustrative Rendering, Animation.

1 Introduction

The traditional approach for the visual communication of scientific insights is handled by illustrators and animators. Their goal is to convey complex phenomena in an accessible way, and to attract the interest of broad audiences. In recent years, we have seen a rapid increase in the communication of topics from molecular biology **cite[drew berry, gj, david bolinsky, Gal McGill, janett iwasa]**, such as the splitting of DNA, the transport of oxygen in the the blood stream, or the comparison of molecular structures within a cell. Such animations are used in teaching undergraduate level biology worldwide.

When illustrators want to convey, for instance, the inner and outer structures of a cell, they face the challenge that the structures of their data are very densely packed. The dense data makes it very difficult to convey inner and outer structures at the same time. Therefore they employ means for occlusion handling, such as exploded views, in order to make all important aspects of the data visible. Illustrators often want to convey the entire variety of different molecules within a cell in form of an atlas-based view [cite some atlas]. However, a cell contains in reality many thousand instances of the same molecular structures. To achieve such an atlas based view, an illustrator does not want to display every single molecule but would rather draw various structures at sufficiently large scale so that the viewer can understand the structural details. In these examples, the illustrator represents abstracted spatial information within same spatial frame of reference. We denote this as a transformation *within* the same visualization space (Figure 2a). In other cases, illustrators want to convey quantitative instead of spatial information, e.g., how many different molecules of each type a biological entity contains, or how a functional relationship between these molecules would look like. In the former case, a bar chart could be used as representation of the quantities. A node-link diagram could be chosen in the latter case to convey the relationships. In such cases, the illustrator would abstract the biological structure from a spatial representation to a quantitative or relational one, which we denote as a transformation *across* visualization spaces (Figure 2b).

For biological entities, a "canonical" representation, especially for non-experts, is the one closest to their mental model that is formed by images from photography, microscopy, and XXX(could not read). The mental model is therefore anchored within the same visualization space as the actual biological entity. If no obvious relation between a mental model and a particular representation exists, the viewer needs guidance on how to relate their mental model to this transformed representation, in order to comprehend the other (transformed) representation form.

In scenarios where interaction is possible or desired, e.g., in an interactive analysis scenario, approaches such as coordinated multiple view (CMV) visualizations can be used for conveying the relations between two representations, e.g., via brushing and linking.

In cases, where interaction is not feasible, such as in presentation scenarios, this guidance is often realized by depicting a transition from a representation A to a representation B, e.g., in the form of static se-

- *Author A is with Institute X. E-mail: a@x.com.*
- *Author B is with Institute Y. E-mail: b@Y.com.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

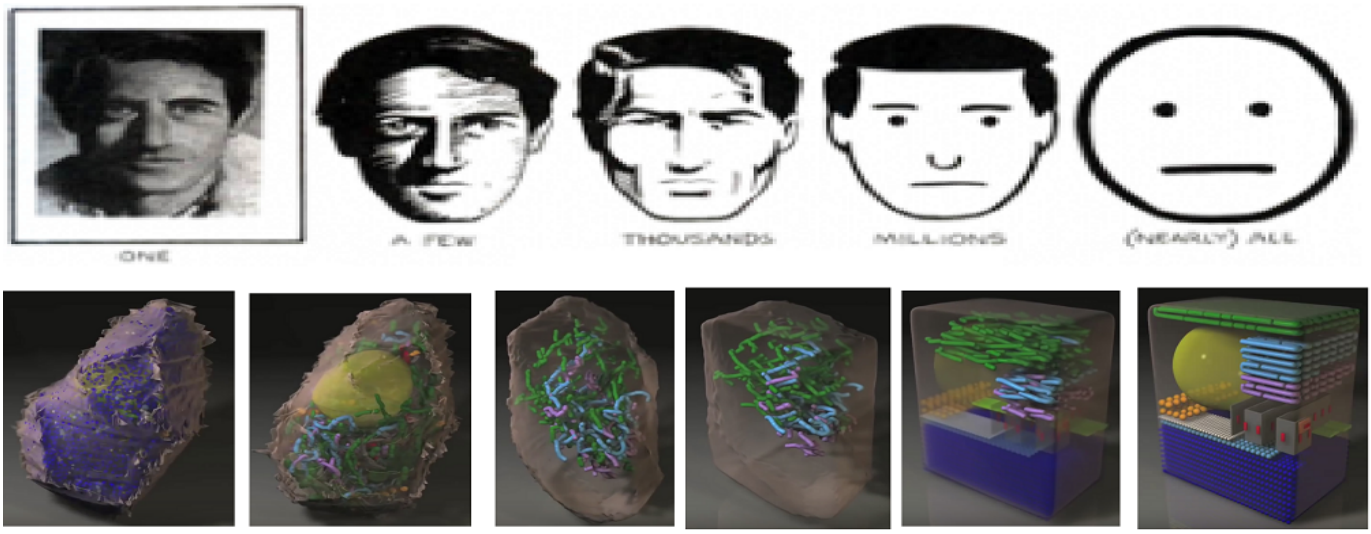


Fig. 2: Examples of transitions within the same visualization space (a), across different visualization spaces (b).

quence of snapshots/small multiples, or an animation/animated transition. We denote such a transition as a metamorphosis and the individual building blocks of the metamorphosis as metamorphers.

Animations are a powerful tool for conveying these metamorphoses, since they are capable of displaying such a transition between two representations in a continuous way. The continuous presentation supports an intuitive understanding of the relationship between representation states. The presentation is both self-explanatory, and enjoyable to audiences [13]. Despite their immense popularity in the visual communication ... (cannot read), animations are also critically regarded by the visualization community. Tversky describes the fleetingness of the displayed information and the potential sensory overload and distracting nature of badly designed animations.

Alternative solutions for conveying transitions alleviate the fleetingness of the displayed information by sacrificing the continuous aspect of the representation. Examples are narrative sequences of small multiples or static images that supplement the missing information of the continuity with glyphs. While solving the problem of fleetingness, they have to be well designed to be understandable.

Once a transition from one representation to another one is planned/conceptualized, there is little aid in modelling tools to realize it. Animators employ key frame animation and create these keyframes manually which is a laborious and time-consuming task. Considering that new discoveries are made frequently in molecular biology, these representations and their transitions have to be re-modeled when new insights are found, which makes the effort for maintaining them even higher.

We therefore set our goal to assist illustrators by eliminating the need to manually author their target representations and the transitions to them. We supply illustrators with a pipeline for creating continuous target representation as well presenting the transitions to them.

Our primary contribution is a formal description of a uniform method for creating continuous target representations for spatial biological data sets, as well as presenting the transitions to these representations - describing the complete metamorphosis of the data representation, so to say.

Data representations are specified in such a way that they form a continuous space. The first three stages of the pipeline are responsible for creating this continuum. The final four stages of the pipeline are responsible for presenting the continuum. This continuum can be arbitrarily sampled in order to create a transition between two representations, e.g., densely, in the form of animation, or sparsely, in the form of small multiples. Each stage supports operators, so called metamorphers, that are responsible for one aspect of the metamorphosis.

As our secondary contribution, we demonstrate a proof-of-concept implementation of this pipeline based on three use cases that we developed in collaboration with illustrators. One of these results, a novel illustrative rendering approach for automatic schematization of mesoscopic data, is our third and final contribution.

2 Related Work

Animation and transitions between visual and data representations have been employed in many contexts. They serve as powerful tools for the dissemination of complex relations in space, time and abstract dimensions. They are frequently used in visual story telling and for the depiction of correspondences between data representations as well as visual representations.

2.1 Visual story telling

Kosara and Mackinlay [9] note that for a long time the focus of visualization research was on exploration and analysis but that presentation of findings especially using elements of story telling should be a research focus of equal importance. They list prominent examples of story telling that include animated transitions for trend analysis and dissemination.

Robertson et al. [13] have looked at trend visualizations in depth. Three methods including animated transitions were analyzed. They found that animated transitions of data was reported to be more enjoyable and exciting to users. They also found that it was significantly faster when used for presentation but less exact and less effective for analyzing data. These results encouraged us to use animated transitions to facilitate presentation of complex molecular data.

Segel and Heer [14] analyze story telling with narrative visualization and point out different approaches. They distinguish between author- and reader-driven elements in a narrative and speculate that ”exploration of transitions between them presents an exciting area for researchers and practitioners”. Along these lines Wohlfart and Hauser [17] presented a guided interactive volume visualization approach. Their system enables authoring and editing of visualization stories that give the user partial control over the exploration.

Ma et al. [12] describe the need of new scientific visualization methods that can be used for scientific story telling. Among other issues they suggest that novel transitions between different coordinate systems are needed to address the overwhelming complexity involved in today’s scientific story telling.

2.2 Transitions between visual representations

Kosara et al. [10] use 3D scatter plot matrices to establish a link between physical layout and the abstract dimensions of the data. Their

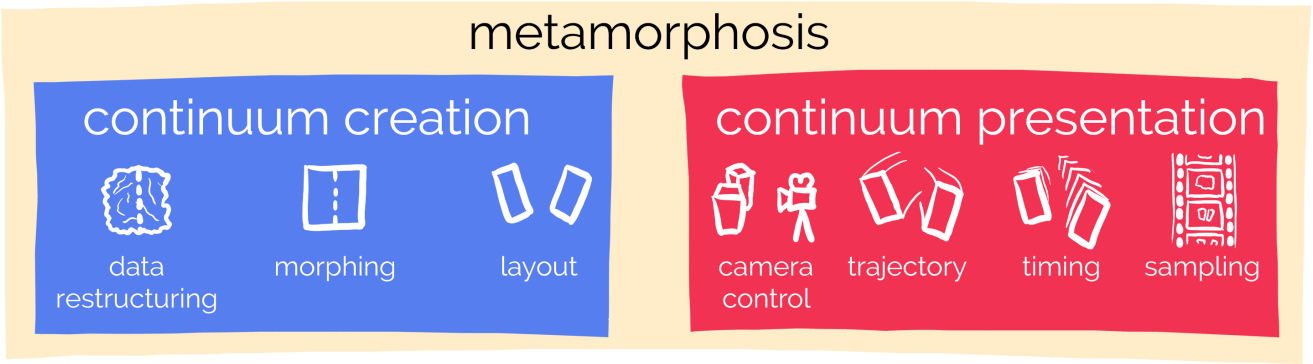


Fig. 3: The metamorphosis pipeline. Continuum creation: data restructuring, morphing, layouting. Continuum presentation: camera control, trajectory, timing, sampling.

work is an early attempt to connect information visualization and scientific visualization using points as data primitives. Elmquist et al. [3] present an exploration technique for multi-dimensional data. They place scatter plots on the sides of a dice and animate the transitions when the dice is rotated.

Guilmaine et al. [4] compare different animated transitions of tree structures. They find that hierarchical animation is better suited for tracking of changes.

Basch [1] describes possibilities for animated transitions between volumetric rendering and abstract views like histograms and scatter plots. They use staggered animation to reduce occlusion. Hurter et al. [7] present a more general technique that interpolates data points between different views which are projections of the original data dimensions. They demonstrate applications for volume data as well as for images. Their approach addresses occlusion with interaction methods like brushing and locking of data points. Heer and Robertson [5] show animated transitions between different visual representations of statistical data. Their work makes extensive use of staging to reduce occlusion and clutter. Our approach is based on several findings of these previous works and formalizes animated transitions between spatial and abstract attributes.

Huron et al. [6] present the visual sedimentation metaphor for the animation of data streams. This metaphor allows to transition from discrete visual elements to a continuous representation. We employ a similar metaphor for the transition of spatial objects to abstract charts.

Exploded views [2, 11, 8] have been used in different contexts as a technique for illustrating spatial or hierarchical relations of parts of an object. They are frequently used in animations or interactive systems and reported to be effective in the dissemination of complex layouts. Our approach is similar to exploded views, since it animates the decomposition of the object into its parts. However, the spatial displacement is not only driven by the visibility of the individual parts, but also by additional data attributes.

2.3 Alternatives to animated transitions

Small multiples [15], coordinated multiple views, and static images in conjunction with annotations, traces and glyphs are alternatives that are typically used to visualize transitions, trends, correspondences or sequences [13]. Tversky et al. [16] summarize cognitive studies on the benefits of animation. Although, they conclude that animation alone has not been convincingly demonstrated to be superior to static illustrations, other findings of their study suggest a direction for research on animations in visualization. They report that animation together with basic interaction methods like pausing, partial re-playing, zooming and change of perspective might be the key to enhance the effectiveness of animation. Further, they acknowledge that the alternatives to animation (such as sequences of static illustration) are surprisingly hard to design and are therefore not an easy target for computer automation.

3 Metamorphosis Formalization

The context of our research is an explanatory visualization of large-scale structural models of viruses and bacteri that conveys relationships between various abstracted or transformed visual representations of the models without any need for user interaction. Such visualization techniques can then be easily employed by scientific illustrators and animators for easily accessible and engaging content authoring for popularising and conveying science to non-expert audiences.

Each illustration which communicates a relationship starts with the two representations of the model. The first, source representation, is typically close to the mental model of the audience. The second, target representation illustrates different aspects of the presented model, but its relationship to the source representation might not be obvious. In that case, the illustrator needs to manually create the visual transition between the two representations. We present a method for creating continuum between two arbitrary model representations, or metamorphosis, and using it to illustrate the relationship between these representations. The metamorphosis is created automatically through describing the target representation, thus alleviating the illustrator from creating it manually.

Figure 4a shows the source and the target representations. They both communicate certain aspects of the model, but they are disconnected, their relationship is not immediately clear. Our pipeline consists two steps. In the first step, we create a continuous space between these representations. However, this low-level connection is not yet ready for illustrating the relationship between the representations (Figure 4b). Therefore, the second step of the pipeline prepares this continuum for the presentation to the intended audience (Figure 4c). In this step, the metamorphosis is finalized so that it illustrates the relationship between the source and the target representations.

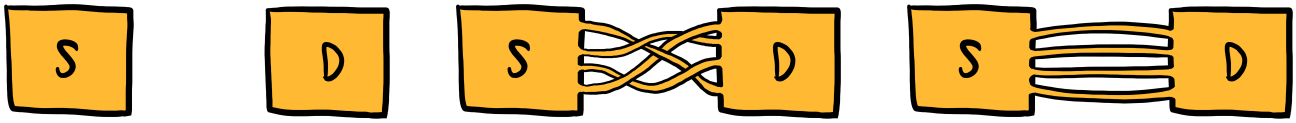
The metamorphosis is carried out by applying a sequence of operators, which we refer to as *metamorphers*.

3.1 Continuum Creation

In the first step of the pipeline, the continuum between two model representations is created. In this section, we describe the three stages of the continuum creation, with examples of metamorphers which can be used in these stages.

3.1.1 Data restructuring

Biological data models are organized in a hierarchy, which corresponds with the semantics of abstraction of the modelled phenomena. This hierarchy often differ in both representations. Therefore, the first step of connecting the representations is to match their semantic hierarchy. In this step of the pipeline, the illustrator defines data subsets within the source model representation, which match the semantics of the target representation, thus restructuring the data. Additionally, some of the data subsets can be duplicated, if the target representation requires it. All the data subsets are then organized in a hierarchy with



(a) (b) (c)

Fig. 4: The continuum between two representations describes the metamorphosis from one representation state to another: a) source and target representation. b) the continuum creation connects both representations. c) the continuum is restructured for presentation.

a tree structure, or a scene graph. All the subsequent pipeline stages can operate on any node of the scene graph, giving the illustrators the flexibility they are used to from the 3D modelling software.

Examples of metamorphers used in this stages are implicitly defined cutaway objects, such as those proposed by Le Muzic at al. **??**, which specify spatial data subsets, or various data clustering algorithms.

3.1.2 Morphing

An important aspect that might obfuscate the relationship between different representations of the biological data are the shapes, or visual representations, of the individual data elements. For instance, molecules in the mesoscale biological models can be displayed as space-filling models, stick models, or one of many different representations commonly used in molecular graphics. Alternatively, molecules can be represented by abstract shapes, such as balls or cubes illustrating their volume.

In this stage of the pipeline, the relationships between the shapes of the corresponding data elements are specified. For this purpose, two types of metamorphers can be applied.

Object-space morphing metamorphers are used to transition between 3D shapes of the visual representations of the data elements. For instance, in the molecular data, this is achieved by continuously repositioning the individual atoms of a molecule to form the shape of a different molecule. If the atom counts of the two molecules differ, multiple atoms are placed at the same spatial location to match these counts before the object-space morphing is applied.

Image-space morphing metamorphers transforms the visual representations of the data elements in the image space, for instance through alpha blending or other image-processing operations. These metamorphers are used when the two representations cannot be adequately matched in the object space.

3.1.3 Layout

Depending on the intended message, the illustrators often need to reposition data elements within the visualization. This might be done as means for occlusion handling, e.g., by exploding dense data to reveal the internal structures, or to simplify noisy or complex spatial arrangement of the data elements to better fit the viewer's mental model. Elements could be positioned so that they represent an altogether different visualization space, such as a network that shows interactions between molecules.

To allow the illustrators to create transitions between different data representations, the spatial arrangements of the data elements in both representations need to be matched. This is achieved by transforming the positions, rotations, and scaling of the data elements through *layout* metamorphers. For instance, such metamorphers can arrange the data elements into various 3D volumes, such as cubes or spheres in order to illustrate their relative counts, or align them side by side for comparison.

The layout metamorphers operate on the individual nodes of the scene graph down to the individual data elements (e.g., molecules), so that whole data subsets can be transformed at once. Additionally, the layout metamorphers can be arbitrarily stacked to achieve wide range of spatial transformations.

3.2 Continuum Presentation

After the continuum between the two data representations is created, it needs to be processed so that it adequately communicates the intended message. This processing takes place within the second step of the pipeline, consisting of four stages.

3.2.1 Trajectory

While the layout stage of the pipeline defines the relationship between data elements in different data representations, the actual transition between them is not yet specified. It is necessary to define the trajectory along which the data elements move in order to transit from the source representation to the target one. This is done in the first stage of the continuum presentation step.

The trajectories are defined as arbitrary sets of control points between the source and target transformations of the data elements. These control points can form various curves along which the data elements move during the metamorphosis. The continuity of the spatial transformations is achieved through linear interpolation of the translations and scaling, and spherical linear interpolation of the rotations.

The metamorphers of the trajectory stage can be used to structure the transition, such as through edge-various bundling methods. For instance, the molecules of the same type can move to their target positions along similar paths in order to minimize the visual clutter created by the transition.

3.2.2 Timing

In order to present the transitions specified in the morphing and the layout stages to the viewers, illustrators need to create appropriate temporal arrangement for them. Such arrangement encodes the information about the chronology of the illustrated events. Additionally, illustrators use the speed at which individual transitions occur as means to suggest their importance for the communicated message, as well as to draw viewers attention to different parts of the illustration.

The temporal arrangement is designed through metamorphers of the timing stage. They operate on so called *time curves* associated with each node in the scene graph. These curves are used as transformations of the data elements' positions within their specified trajectories as functions of time.

The metamorphers are used to specify the starting time of each transformation, as well as its speed. As such, they can be used to create various temporal effects, e.g., ease-in or ease-out curves for the movement of the data elements or entire data subsets. The time curves can be also modified to make some of the elements stop in the middle of the specified trajectory, or to reverse their movement.

3.2.3 Camera Control

To adequately follow the development of continuous transitions, especially those between different visualization spaces, camera steering is a necessary component of a non-interactive explanatory visualization. To assure visibility of all data elements at each timestep, illustrators manually specify camera settings and how they change over time.

In our pipeline, this task is carried out through the metamorphers in the *camera control* stage. These metamorphers access the temporally and spatially arranged data elements and use this information to automatically modify the position and the look-at vector of the camera. An example of camera control metamorpher sets the look-at vector of the camera to point in the centroid of the data elements, while it moves the

camera far enough to capture the entire dataset for the given field-of-view.

3.2.4 Sampling

To present the created transition to the viewers, it is necessary to select a set of timesteps which are going to tell the story intended by the illustrators. In the simplest case, the transition is sampled densely enough to form a seamless animation, given certain FPS and a desired length of the animation. However, due to various disadvantages of animation, as discussed in Section , it might be necessary to select only few representative timesteps which show the essential aspects of the transition.

Metamorphers of the *sampling* stage implement various strategies how these timesteps are chosen. While the previous stages of the pipeline can be called multiple times in order to create more complex animations/transitions/stories, the sampling stage is final, as it ties together the information from the previous stages to sample the visual result.

For this purpose, the scene graph information is propagated through the hierarchy down to the individual data elements, to determine their absolute positions and orientations. The trajectory and the timing stages of the pipeline provide the information about the intermediate transformations of the data elements in an arbitrary timestep, which is in turn acquired by the sampling metamorphers.

Since the operators applied in the previous stages of the pipeline are continuous, there are no restrictions on how the sampling is performed. For instance, it is possible to create a hybrid visualization that displays a sparsely sampled narrative sequence of small multiples. Upon request, a continuous animated transition between two consecutive small multiples can be shown to provide additional information on how do they relate to each other.

The sampling stage can also determine if the transition is pre-baked or sampled on the fly in real-time. This makes it possible to use a continuous pre-baked transition to create, for instance, a single result image that encodes the missing transition information within glyphs. These glyphs can be calculated from the continuous pre-baked information.

4 USE CASES

After the description of the theoretical pipeline, we demonstrate its utility on several use cases. We show these examples on the structural model of the human immunodeficiency virus which is built-up from thousands of macromolecules. The HIV particle is contained within blood serum **(see Figure XXX)**. The source visual representation is the direct rendering of the structural data set, which is then continuously transformed into a particular destination visualization space. Each example highlights a different aspect of the structure: 1) what does the three dimensional structure of the virus and its inner compartments actually look like and how do they relate to each other hierarchically? 2) what types of molecules are contained in which compartment of virus, how do they look like, and in which approximate quantity are they present? 3) how large is the volume of each compartment and the molecules contained within in respect to each other?

4.1 Explosion of Molecular Structures

Since molecular structures of the HIV data are very densely packed, it is impossible to inspect the outer and inner structures of the virus at the same time. The four structures that we want to show are assembled in an onion-like structure where one compartment is contained in another one. We therefore chose to convey the encapsulation of compartments in form of an exploded view (Figure **??**), to answer our first question.

The specific pipeline for creating the continuum between the original visualization and the exploded view starts with restructuring. In order to create the relation of the original data to the four compartment structures of the target representation, we create subsets for all four compartments in the restructuring stage. To create the compartment subsets, we exploit the fact that each compartment contains molecules of different types, and implemented a *type range* metamorpher. The subsets crated by the type range metamorpher contain all given molecules that match a specified type range. If each molecule type could exist within multiple compartments, we could achieve the creation of the compartment subsets, by spatially grouping the molecules with a *spatial splitting* metamorpher. A spatial splitting metamorpher checks the position of a given list of molecules in respect to a spatial object, like a plane, cube, or sphere, that the user can position freely in the scene. Molecules are then assigned to two new subsets, depending on which side of the object they are situated. To create the seam at which compartments are split for the explosion, we actually apply a spatial splitting metamorpher to spatially subdivide the three outer of the four subsets that we just created along a plane. At the end of the restructuring stage, the root node of the scene hierarchy has four child nodes for the compartment subsets. The nodes of the three compartments that are exploded in the final representation, have two additional child nodes containing the spatial subdivision.

Since the visual representation of individual molecules does not change across the source and the target representation, we do not need to apply any metamorphers in the morphing stage of our pipeline.

In order to relate the original unexploded data to the final exploded representation, we need to call *translation* and *rotation* metamorphers in the layouting stage that create the translation (and slight rotation) of the explosion for each pair of spatially split subsets (Figure **??**b). An additional translation moves each inner compartment to the right of its previously encapsulating compartment (Figure **??**c).

We now created the continuum between the original unexploded and the exploded target representation. In order to present the transition in a pleasant way, we follow the remaining steps of our pipeline.

Since both, source and target representation, remain within the same view frustum, and we do not explicitly point the camera on any specific details, we apply no metamorpher in the camera control stage.

Our original data differs from the exploded target representation only in position and topology (the split compartments). To present the transition, we move all compartment subsets as a unit so that they keep their shape during the transition. In the trajectory stage, we therefore apply the *linear interpolation* metamorpher to achieve a linear transition along the continuum between source and target representation.

In the layouting stage, we apply a total of XXX layout metamorphers/operations to create the relation to the exploded representation. If all these operations would be executed at the same time, the transition could be hard to follow [cite Heer paper here]. In the timing stage, we can make the presentation of the transition more pleasant, by applying a *delay* metamorpher. The delay metamorpher adjusts the time curve of each compartment subset so that each subsequent explosion starts only after the previous is finished.

In order to also support understanding for the structures of individual molecules/proteins and to convey the opening motion of each exploding compartment in the final representation, we show representatives of the proteins separated from the compartments in which they are contained. These representatives are molecules that simply do not complete their transition. To end the transition for these molecules before they reach their final position, we manipulate their timecurves with a *[need fitting name]* metamorpher. During the transition, the opening compartments thus leaves a motion trail of molecules behind that makes individual molecules discernible.

The only thing that is left to accomplish now, is to sample the continuum representation that we created. We chose a *dense/continuous sampling* metamorpher to create a ten second animation with twenty four frames per second for the supplemental video. For the Figure **??** we applied a *sparse sampling* metamorpher [or we took stills from this video?][or we have one sampling metamorpher that we used for video and figures but with different parametrizations - how to spin/sell this?].

With the transition that we just created, we can explain to a viewer, how the four compartments are actually enclosed within each other, as the final representation alone does not convey that one object is enclosed in another, especifally for a viewer who has not seen the dataset before. Furthermore the transition explains where the isolated molecules originate from.

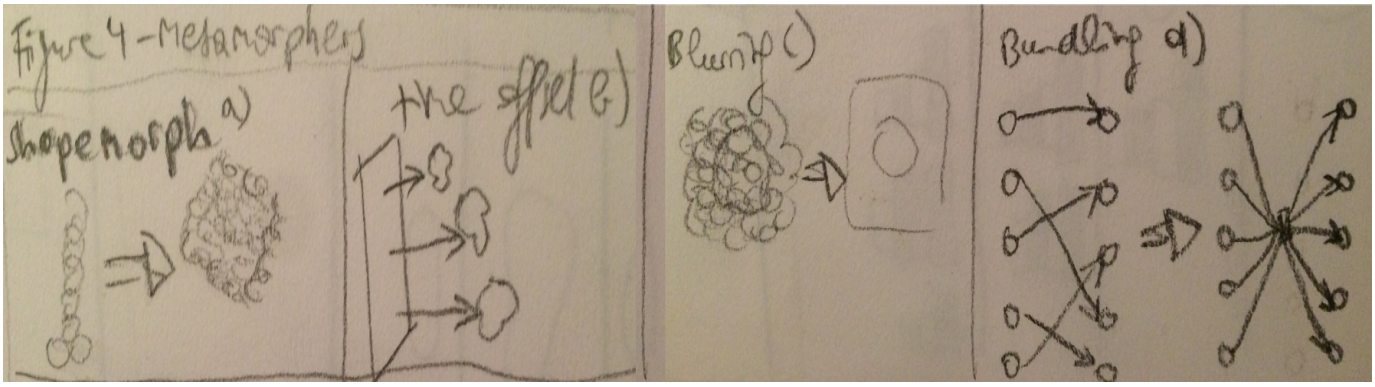


Fig. 5: Examples of different metamorphers: a) shape morphing, b) blurring, c) time offset, d) some layout

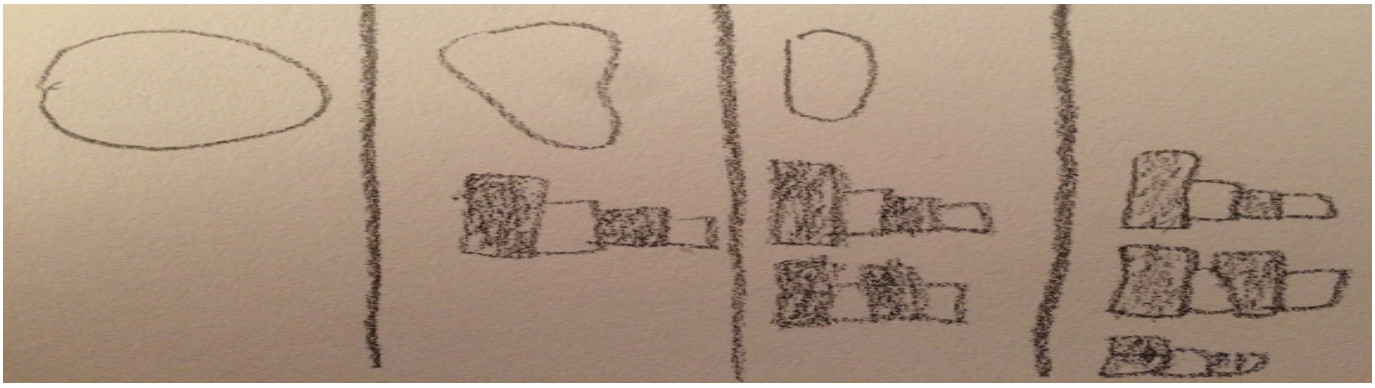


Fig. 6: Infovis

4.2 Schematization of Molecular Structures

Besides the structural composition of a cell, biologists are often also interested in the cell's molecular composition, i.e., what types of molecules are present in the data, in which compartment are they contained, what do the molecules look like? The HIV data, for instances, contains 42 different molecules but due to their high count (thousands of molecule instances across these 42 types), their mostly random distribution and the dense packing of the data, the detailed information about the number of types and their individual shape is impossible to deduce from the original data. Illustrators therefore create atlas based views (such as in Figure 1d) that show a simplified schematic representation of a cell to highlight the shapes of different molecules, as well, as the shape and size of the structures that they collectively compose and are contained in. Low level details of the noisy molecule distribution, i.e., the exact count, position and shape of individual molecules are suppressed for this schematic representation.

We achieve such a transition to an atlas based view with our pipeline in the following way: Similar to the first step in the exploded view use case, we restructure the data with the type range metamorpher to create the relation to the four subsets that represent the compartments of our schematic target representation. Technically, we need each subset twice. Since the four subsets will be blended in image space to create the uniform compartment color, we need a copy of each one for the layouting stage. The *copy* metamorpher creates copies of the supplied subsets, resulting in copies A and B for each subset.

In the morphing stage, we now create the relation of our original noisy data representation to the uniform color that marks each compartment's area by applying a *blur* metamorpher. The blur operator creates an additional render pass for each given subset where the respective subset is rendered and blurred individually. By increasing the contrast of the alpha channel for the blurred image, the sharp edges of each compartment are created.

In the layouting stage, we create the relation of our dense and noisy original data to the sparse enlarged rendering of representative molecules and compartment shells in the final representation. For a given subset, the *schema* layout metamorpher differentiates between membrane molecule types, i.e., molecules that compose the shell of a compartment, and non-membrane types, i.e., the ones that are floating within a compartment, in order to create the final schematic representation. Since illustrators know, which molecule types the membranes of compartments are composed of, they can specify the membrane type so that they are treated differently than the molecules within compartments. For membrane molecules, the schema layout metamorpher picks the ones that are closest to the border of the compartment, from the viewing position. The picked membrane molecules remain their original size and rotation. The non-picked membrane molecules are scaled to zero to be invisible in the final representation, in order to reveal the compartments and molecules within. Alternatively, instead of scaling, transparency could be applied to membrane molecules in the morphing stage. For membranes, the schema layout operator thus preserves the membranes shells of compartments while revealing the inner structures of a compartment.

For non-membrane molecules, the illustrator has control over the total count of representatives that should be displayed in a compartment. Based on the specified number, the schema layout metamorpher calculates how many representatives of each molecule type of the given subset should be kept. The number of kept representatives depends on the ratio between the total number of molecules of each type within the subset, and the maximum number of allowed representatives. Each molecule has one guaranteed representative. Once the number of representatives for each molecule type is calculated, the schema layout metamorpher creates a uniform grid for the available space with as many grid points as there are representatives. For each grid point, the metamorpher then pics

to specify a percentage of how much of the available space should be covered with molecule representatives. This percentage is used to

the schema layout metamorpher calculates a count of how many molecules can be rendered on counts the instances of each type within the subset, calculates a number that represents

To achieve this schematic representation therefore, on the one side, specific molecules have to be chosen as representatives of their type to be displayed at a sufficiently large scale (focus), while suppressing the noisy distribution of a vast amount of other molecules (context).

molecules that do not have a noisy distribution but form coherent structures, such as cell membranes, have to be preserved.

timing: distance

because this type of automatic illustrative schematization has not been presented/solved yet in literature, we present it as our tertiary contribution...

discussion: @discussion of results: transition = more intuitive to follow. This metamorphosis corresponds with a transformation within the same visualization space.

4.3 Representation of Quantitative Relations

For the analysis and explanation of molecular data often also the comparison of quantitative aspects is important for its understanding. Such quantification is necessary, for instance, when analyzing different life cycle stages of a cell. In our example, we want to convey how many different molecule types the HIV particle contains, and how much of the volume within the particle each type occupies.

The direct comparison of such information on the original 3D models of the cells can deceive due to the complexity of the molecular structures. Illustrators therefore often supply quantitative information in the form of infographics, which are based in a different visualization space than the original biological data. In this space, spatial dimensions are not used to convey spatial dimensions but rather abstract dimensions, such as atom counts[cite tory/mller?]. The relation between the original spatial and the transformed abstract dimensions is even harder to grasp. Continuous transitions are especially suitable for depicting such complex relations, as was successfuly demonstrated in the award-winning animation of the decomposition of a cell (Figure **??**b). In this video, scientific illustrator Graham Johnson used an animated transition to convey the relation between the 3D model of a pancreas cell and an abstract depiction of volumetric relations between the elements of the cell.

In this final use case, we demonstrate how we can achieve such a transition to an abstract visualization space with our pipeline. We chose to convey the quantitative information about the number of molecule types in the HIV particle and the volumetric relation between types with a histogram (Figure **??**d). The number of histogram bars corresponds to the number of molecule types. The height of each bar corresponds to the volume that the respective molecule type occupies in the HIV particle. Since we are concentrating on displaying relations within the HIV particle, we remove the blood serum from our original representation before we restructure our data hierarchy to match the intended target representation.

Since each histogram bar has to represent a single molecule type, we use a *group-by-type* metamorpher to create a subset for each type.

The histograms in the final representation are constructed from the individual molecules in our original data. Each molecule is represented as a slice with given base area and a height that represents the molecule's volume. By stacking the molecule slices, we receive a height that is representative of the total volume of all molecules of the respective type. In the morphing stage, we therefore apply an *object space* metamorpher on each type subset, in order to morph the shape of each molecule to the afore mentioned slice. The morphing is achieved by repositioning the atoms of a molecule into a slice. The slice height depends on the number of atoms that the molecule is composed of.

In order to layout the created slices into the shape of a bar, we use a *bar layout* metamorpher on each type subset. The bar layout simply stacks the molecule slices in a subset on top of each other. The bars are then positioned side-by-side along a line with a *line layout* metamorpher.

Since we position the histogram bars in the final representation below the HIV particle, the molecules on top of the particle would, with a linear trajectory between their origin and destination point, fly through the particle, and exit the particle on the bottom side, before reaching their final position in the histogram. To present the transition in a manner where molecules do not cross the HIV particle on their way to their destination, we introduce additional control points to the path of the molecules in the trajectory stage of our pipeline. **[TODO: description of first (sideways) CP]**

Further, we add a control point for each molecule type above its respective histogram bar to create a trajectory that looks like the molecules are filling the bar by pouring into it. We therefore bundle the trajectories of the molecules above each bar with a *trajectory bundling* metamorpher. We place the control point at the same height for all bars. To calculate the x/y position for a control point, we use the center of the bounding box of the respective molecule type subset. Each subset in the scene hierarchy has a bounding box. Bounding boxes are calculated when a subset is created and updated when a layout is applied.

Due to the onion like structure of the HIV particle, inner structures are occluded by outer structures. Inner structures should therefore not start their transition before the outer structures, as they would never be visible in their original representation. Further, if the inner structures would fly through the outer structures to their target representation, a viewer would not understand, from which part of the virus exactly they would be coming from. To present the transition in a manner, that reveals each original structure before it starts its transition, we apply the *staged timing* metamorpher to peel HIV particle like an onion, starting with its outer layers. In contrast to the previous two use cases, here, we stage the timing by molecule type to transition each molecular type individually into its histogram bar representation. The transition of each type starts after a certain delay / when the previous transition is finished. To create the order of the staging that results in the peeling effect, we use a *distance offset* metamorpher. The distance of each molecule of a specific type to the center of the particle is calculated and averaged. The average distance per type is then used for the ordering of the staging. As the center of the particle we assume the center of the bounding box of the root node of the scene hierarchy.

In contrast to the previous use case, the position of the final representation (the histograms) differs from the position of the original representation (the HIV particle). A camera position that shows both representations at the same time would use the screen space inefficiently, since only during the transition both representations are visible. The original and the target representation would be shown with unused screen space where the respective other representation would be positioned. We therefore implemented a simple *guided navigation* metamorpher to use the available screen space efficiently by zooming in as closely as possible to the displayed data, while assuring that entire scene (all molecules) are visible at all times during the transition. The metamorpher checks if molecule instances are culled from the view. If yes, then the camera is zoomed out until no more instances are culled. If no, the camera zooms in until instances are culled. A constant zooming in and out at the optimal position is avoided by smoothing the camera transition. The camera look-at vector is updated to look at the center of our scene, where our HIV particle is positioned.

Finally, to produce the animated transition to the histogram bars, why apply the same continuous sampling as in the previous two use cases.

The transition conveys two aspects of the data that a simple side-by-side presentation of original and target state would not. One the one side, the relation between the histogram bars and the molecular structures that they represent is established. On the other side, the inner structures of the HIV particle that are not visible in the original representation and not present in the target representation are revealed in the course of the transition.

[this is the old usecase section]

4.3.1 Applied Metamorphers

–data transform: 1) [type range grouper] create three high lvl subsets: the blood serum, the membrane, the capsid 2) [copy] the entire partitioned data set because we will render it in two different representations =¿ results in copy A & B

–annotations: do the circular annotations?

–visual transform: A: blurrin & increasin contrast of alpha channel for nice charp edges of each subset individually B: scalin

–layouting (spatial transform): A: renderin them on top of each other (occlusion management) B: rotatin

–navigation: not necessary, because we stay in the same visualization space / spatial extents

–trajectory: B: linear interpolation between rotations

–timing: staged per compartment distance field left to right for each

–sampling: continuous

4.4 Metamorphers

[just for reference] [describes PoC components that we implemented per pipeline stage]
[should we describe beforehand what (which representations) we try to achieve with these implementations?]

4.4.1 Partitioning

spatial subselection by clipping plane/object,
splitting into types,
splitting into type ranges
cloning

—not showcased but implemented: *spatial splitting operations into equal parts (pie splitter),

4.4.2 Annotations

text labels, icons (textures), maybe: data representatives (for each type), optional addon to each operator: connectors between annotation & assigned subset

4.4.3 Blending / Morphing

blur into collective shape, morph,

—not showcased but implemented: blending to texture

—other not implemented & not used examples: unfolding

4.4.4 Layouting

bar, line, translate, rotate, scale, schematization layout = combi of scale & rotate but selection of elements where what op is applied is very specific

—not showcased but implemented: slice, sphere, circle

4.4.5 Camera / Guided Navi

[whole-data-in-view-keeper] update the camera pos & lookat to keep all molecules in the view frustum.

other examples:

4.4.6 Trajectory

linear: from A straight to B, bundling/transit points

—other not implemented & not used examples: curved

4.4.7 Timing

offset function: delay timing by distance to plane & staging by type ... anything else?

—other not implemented & not used examples: offset according to: inherent spatial parameters (position, distance to other entities, density)

4.4.8 Sampling

continuous sampling,

—other not implemented & not used examples: sparse (comic strip / narrative sequence),

post-processing operators: motion blur,

5 Implementation

5.1 PoC Pipeline

[contains specifics about our pipeline implementation] **[move to implementation section]** implemented in cellVIEW: give some details about cellVIEW): unity, supports loading &rendering of cellPACK data, renders molecules as point clouds

desc was jeder component beisteuert um die beiden reps zu erzielen

6 Discussion

6.1 Discussion of Results

6.1.1 infovis

shapes and positions of compartments and molecules are sacrificed. @discussion: here, large distance between representation forms in visualization space - a continuous transition is therefore especially suitable for conveying the transition..

6.2 Expert Feedback

6.3 General Applicability of Our Approach

compatible data formats: point cloud, volume, polygonal?

compatible data types: dense hierarchical data with lots of repeating entities of a few dozen types

6.4 general discussion

6.4.1 additional outcomes

side products of our pipeline - free of charge / easily achievable with minimal additional effort: sequence: comic strips, static image with glyph

6.4.2 animation

tversky: congruence principle: the structure and content of an external representation should match the structure and content of the desired mental representation.

apprehension principle: the structure and content of the external representation should be readily and accurately perceived and comprehended.

6.4.3 things that should be preserved:

depending on the usecase / representation - not all properties are essential for each representation form.

- instance level: position of instances
- type level: shape of types, relative size of types, color of types (?), number relation between types
- compartment level: shape/pos/size/color of compartments / structures
- global level: volumetric dimensions of the data
- todo: what else?

6.5 occlusion

is of course a central factor that could hinder a user's understanding of a transition. can be dealt with at various stages of the pipeline. grouping: choose groups so that they dont interfere with each other (depending on how you plan to transform the data) TODO: come up with examples that describe which kind of grouping would be beneficial for which kind of target state e.g., group the data first into layers parallel to the viewing plane that basically peel the data layouts inner bar: within the bar occlusion does not matter, as long as the bar represents entities of the exact same type todo: other inner layouts & general outer general: the placement of entities/subests should be chosen so that they dont occlude each other from the viewing angle. maybe we should not distinguish between inner and outer layout since this kind of restricts us to having two layouts instead of n.

then again, we could argue that there can be n-outer layouts, and one inner layout morphing: same as inner layout: within the shape occlusion is not an issue

trajectory: the trajectories offer two of techniques for avoiding occlusion.

edge bundling: setting control points so that paths don't occlude each other todo: anything else?

timing: coordination of timing and speed of elements, e.g., staging

7 Conclusion

7.1 future work

implementation of comic strip / animation hybrid

Acknowledgments

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

References

[1] C. Basch. Animated transitions across multiple dimensions for volumetric data. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, oct 2011.

[2] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, Sept. 2006.

[3] N. Elmqvist, P. Dragicevic, and J. D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, Nov 2008.

[4] D. Guilmaine, C. Viau, and M. J. McGuffin. Hierarchically animated transitions in visualizations of tree structures. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 514–521, New York, NY, USA, 2012. ACM.

[5] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, Nov 2007.

[6] S. Huron, R. Vuillemot, and J.-D. Fekete. Visual Sedimentation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2446–2455, 2013.

[7] C. Hurter, R. Taylor, S. Carpendale, and A. Telea. Color tunneling: Interactive exploration and selection in volumetric datasets. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis 2014)*, pages 225–232, March 2014.

[8] O. Karpenko, W. Li, N. Mitra, and M. Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, Nov. 2010.

[9] R. Kosara and J. Mackinlay. Storytelling: The next step for visualization. *IEEE Computer*, 46:44–50, 2013.

[10] R. Kosara, G. N. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3d scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 133–140, 2004.

[11] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Transaction on Graphics*, 27(3):101:1–101:7, 2008.

[12] K. L. Ma, I. Liao, J. Frazier, H. Hauser, and H. N. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, Jan 2012.

[13] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, Nov 2008.

[14] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, Nov 2010.

[15] E. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, USA, 1990.

[16] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, Oct. 2002.

[17] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07, pages 91–98. Eurographics Association, 2007.