TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/08 pp157-170
Volume 18, Number 2, April 2013

# A Study of Animated Transition in Similarity-Based Tiled Image Layout

Huan Zhang, Jun Tao, Fang Ruan, and Chaoli Wang*

**Abstract:** For many information visualization applications, showing the transition when interacting with the data is critically important as it can help users better perceive the changes and understand the underlying data. In this paper, we investigate the effectiveness of animated transition in a tiled image layout where the spiral arrangement of the images is based on their similarity. Three aspects of animated transition are considered, including animation steps, animation actions, and flying paths. Exploring and weighting the advantages and disadvantages of different methods for each aspect and in conjunction with the characteristics of the spiral image layout, we present an integrated solution, called AniMap, for animating the transition from an old layout to a new layout when a different image is selected as the query image. We show the effectiveness of our animated transition solution by demonstrating experimental results and conducting a comparative user study.

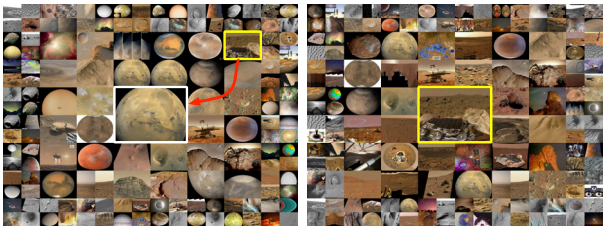
**Key words:** animated transition; image layout; large image collections; user study

## 1 Introduction

Animation is a promising approach to facilitate the perception of changes when transforming from one layout to another. Previous research has found that animated transition can significantly improve graphical perception of changes between statistical data graphics[1], spatial relationship perception[2,3], and decision making[4]. In dynamic graphic drawing, it was also suggested that tasks related to dynamic evolution might be better handled via animation compared to small multiples[5]. However, others have also warned that animation could become problematic if not used appropriately. Compared to static depiction of trends, animation is the least effective in trend analysis[6]. Besides, animation shows no significant improvement in window navigation[7]. For browsing image collections, although researchers have shown that arranging a set of thumbnail images in a layout according to their similarity does help viewers narrow down their search and identify images of interest[8], whether or not animation can facilitate browsing and querying requires a thorough study.

In this paper, we study animated transition for iMap that tiles images in a collection based on their visual and textual similarity[9]. As shown in Fig. 1a, within a rectangular display area, iMap shows the query image at the center as the focus and arranges other images with decreasing similarity ranks as the context following a spiral pattern. By default, the focus image is displayed in a normal size and the width and height are reduced by half for the successive layers. The user can adjust the number of repetition layers needed to keep the current width and height. Although effective, a key challenge of using iMap lies in the difficulty to follow and track the changes when updating image arrangement as the

• Huan Zhang is with Dematic Corp, Grand Rapids, MI 49505, USA. This work was performed when she studied at Michigan Technological University. E-mail: huan.zhang@mtu.edu.
• Jun Tao and Chaoli Wang are with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931, USA. E-mail: {junt, chaoliw}@mtu.edu.
• Fang Ruan is with Google Inc, Mountain View, CA 94043, USA. This work was performed when she studied at Michigan Technological University. E-mail: fangruan@google.com.
* To whom correspondence should be addressed.
Manuscript received: 2013-03-05; accepted: 2013-03-06



(a) (b)

**Fig. 1 iMap query without animated transition. In the old layout (a), an image highlighted in yellow boundary is selected as the new query image. The new layout after the selection is shown in (b).**

query image changes. Figure 1 shows such an example.

Tversky et al.[10] suggested two high-level principles for effective animation. One is *congruence principle*, which states that "the structure and content of the external representation should correspond to the desired structure and content of the internal representation." In our case, the external animation should reflect the internal similarity rank changes. The other is *apprehension principle*, which states that "the structure and content of the external representation should be readily and accurately perceived and comprehended." That is, our animation should be understandable from the viewer's perspective and the changes of image similarity ranking should be accurately perceived during the transition. Our animation design should conform to these two principles. It should also abide by the characteristics of the iMap layout by allowing the user to focus on the centered query image and similar images retrieved for the display. Furthermore, our animation should strive to avoid distraction while maintaining the efficiency.

In our work, we explore three aspects, animation steps, animation actions, and flying paths, to design effective animated transition. The new AniMap we develop is an enhanced version of iMap, introducing animated transition from an old layout to a new layout when a different image is selected as the query image. We demonstrate that by breaking down a complex animated transition into several logical stages and making each stage simple enough to comprehend, we can improve the understanding of image changes. By exploring different factors that might have an impact on the animation, we propose our solution for smoothing the animation and reducing the overlap among images during the transition. To verify the effectiveness of our animated transition solution, we perform a user study to evaluate the three animation conditions: no-animation, one-step animation, and multi-step animation (used in the AniMap). Our results show that multi-step animation is significantly more accurate than no-animation and one-step animation, and is noticeably faster than one-step animation. Due to the nature of animated transition, we refer readers to the accompanying video for the best evaluation of AniMap.

## 2 Related Work

Animation has long been a topic of research in information visualization, focusing on user interface for a variety of purposes[11]. Hudson and Stasko[12] introduced toolkit support for animation. The Information Visualizer[13] utilizes 2-D and 3-D animation with cognitive coprocessor to explore information and its structure. Other researchers have concentrated on designing animations to facilitate human perception and transition understanding. The cone tree[14] and perspective wall[15] use 3-D animations to help keep viewers oriented. DynaVis[1] supports animated transitions between statistical data graphics backed by a shared data table. Yee et al.[16] applied animated transition to achieve smooth transition in dynamic graphs with a radial layout. van Wijk and Nuij[17] introduced a generic model for smooth and efficient animation with simultaneous zooming and panning for image viewing. Elmqvist et al.[18] leveraged animated 3-D rotations to show the transition among a matrix of scatterplots. Gapminder's Trendalyzer uses an animated bubble chart to show trends over time, which can be effectively utilized in analysis, exploration or presentation. Microsoft's Pivot demonstrates great aesthetic and practical values of animated transition in handling large image databases displayed in regular row-and-column layouts. In contrast, our work particularly focuses on a spiral image layout which brings new challenges such as reducing image overlap during animated transition.

## 3 Animation Factors

For iMap, while it is sufficient to show the final layout from the perspective of the new focus, simply switching to this new view can cause a highly disorienting rearrangement especially when the number of images displayed is large. To reduce this disorientation and improve visual comprehension, we consider our animation from a *global* view and a *local* view. The global view focuses on the overall animation

structure and refers to *animation steps* such as one-step and multi-step animations. The local view considers animation details, including *animation actions* such as fading and flying, and *flying paths* such as straight paths and curved paths.

### 3.1 Animation steps

An animation usually consists of multiple sub-animations. To complete an animation, all steps could be finished simultaneously, or each successive step is accomplished before the next step starts. We propose two categories of animation steps: *one-step* and *multi-step*. For one-step animation, after a new query image is selected, all images are synchronously changed with one kind of animation action, such as fading, flying or size change. While it is easy to implement and indeed informs the viewers that the change is happening, one-step animation only provides fairly limited clue for tracking images and their similarity rank changes.

To facilitate the tracking, we can break down the animation into multiple steps, i.e., staging the animation. Lasseter[19] pointed out it is important, when staging an action, that only one idea is seen by the audience at a time. If many actions are happening at once, the eyes do not know where to look at and the main idea of the action will be "upstaged" and overlooked. More importantly, the object of interest should distinguish itself from the rest of the scene. Since one is unable to perceive many different actions at one time, every key action must be staged in the strongest and simplest way before moving to the next one. Based on these guidelines, only a few different actions should be played simultaneously in one step, and only those actions that will not confuse the viewers can be issued together. Furthermore, the actions that are totally contrasted with each other will be performed separately, because it is very important for the viewers to look in the right place at the right time.

### 3.2 Animation actions

Animation actions deal with how an image should be brought in or out of the current layout. We implement four representative animation actions: *fade-out/fade-in*, *fly-out/fly-in*, *size-out/size-in*, and *card-flip*. Here, "out" means image disappearance, and "in" means image appearance. The user is allowed to adjust the animation time and the number of animation steps for each kind of action. The first three animation actions can be applied to both one-step and multi-step animations and mixed together (e.g., fade-out/fly-in). The card-flip only works for one-step animation due to its nature. The first three actions we discuss in this section apply to one-step animation. In multi-step animation, these three actions will be utilized as well. However, at the same time, the action taken by each image may be different (refer to Section 4).

**Fade-out/fade-in** This action is the most popular way to show the transition. In our scenario, all images shown in the old layout fade out first and then all images in the new layout fade in.

**Fly-out/fly-in** For this action, images in the old layout fly out of the display area first and then images in the new layout fly in (see Fig. 2). The center of an image's position before flying is the source $p$. For one-step animation, the "virtual" destination $q$ of the image is located along the ray that connects the center of the entire display area $c$ to $p$. Assuming the distance between $c$ and $p$ is $d$, we assign the distance from $c$ to $q$ as $d + \Delta d$, where $\Delta d$ is some constant distance. For multi-step animation, the center of the image's new position after flying is the "actual" destination. By flying each image from its source to destination, we can detect if the image's similarity rank increases or decreases. As a matter of fact, one of the greatest challenges for us is to design an optimal way of flying to reduce image overlap. We will analyze this and present our solution in Section 3.3. Another important factor of the fly-out/fly-in action is timing control. Even though the user is able to adjust the total flying time, we apply slow-in/slow-out timing instead of straightforward linear timing for better motion perception. We compute the velocity at each flying step as

$$v_t = \mathrm{e}^{-\left(\frac{t}{T-1}-0.5\right)^2 \times s} \tag{1}$$

where $t \in [0, T-1]$ is the $t$-th flying step, $T$ is the total number of flying steps, and $s$ is a speed factor. Both $T$ and $s$ can be adjusted by the user. When $s$ is zero, there is a constant velocity of image flying. When $s$ is

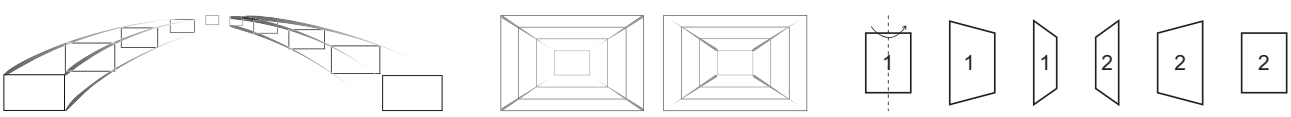



**Fig. 2 Left: fly-out/fly-in action; Middle: size-out/size-in action; Right: card-flip action.**

set to a larger value, the difference between the speed at the middle of flying and the speed at both ends will be larger, i.e., the animation begins slowly, smoothly accelerates, and then decelerates in the end. Therefore, most of the movement occurs during the middle one-third of the given flying time, which provides a good visual cue to help the user anticipate image movement.

**Size-out/size-in** For this action, we first gradually shrink the size of an image into a point, and then grow the size of the new image from a point to its target size (see Fig. 2). The reason for us to take the size factor into consideration is mainly due to its role in the multi-step animation for overlap reduction, which we will discuss in Section 4.

**Card-flip** Unlike all previous three actions, the card-flip creates a 3-D effect for the transition. Each image quad in the display area is now regarded as a slab where the front face is mapped with an image in the old layout. When a new focus is selected, images shown in the new layout are first mapped to the back faces of the slabs. Then in its local coordinate frame, we simultaneously rotate each slab along the $y$ direction for 180° in an animated fashion, so that images on the front and back faces are flipped (see Fig. 2).

**Overlapping actions** Under the condition that two actions can not influence each other and make either of them unreadable, overlapping actions is helpful for conveying the main ideas of the transformation while maintaining the efficiency. Lasseter[19] stated that an action should never be brought to a complete stop before another action starts, and the second action should overlap the first. In this way, the overlapping maintains a continuous flow among different phrases of actions. We apply overlapping actions to both one-step and multi-step animations. The user has the freedom to adjust the degree of overlapping, from non-overlapping to fully-overlapping.

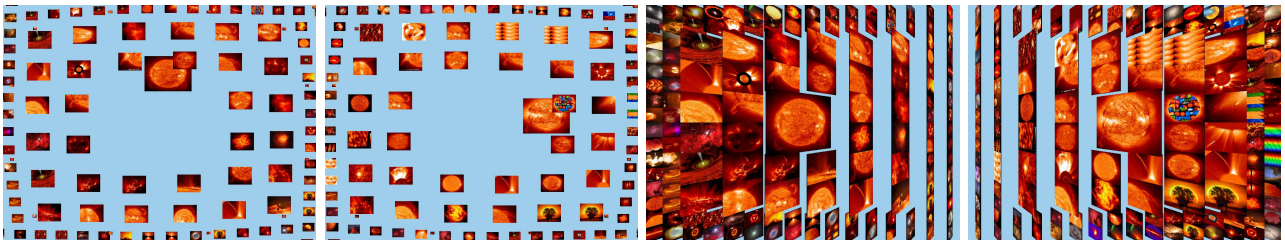
Figure 3 shows the snapshots of two animation actions, fly-out/fly-in and card-flip, for one-step animation. With fly-out and fly-in overlap, images in the old layout fly out while images in the new layout fly in. Compared with straight paths, using curved paths (refer to Section 3.3) makes the animated transition more attractive. For the card-flip action, the flipping effect incurs image overlapping during the transition. However, since the images do not fly around, it is still easy to follow even though many images are being flipped simultaneously. The video clips also include the results for the other two animation actions, fade-out/fade-in and size-out/size-in.

### 3.3 Flying paths

For the fly-out/fly-in action, many images will fly from their sources to destinations at the same time, which easily leads to image overlapping and visual cluttering. Unlike overlapping actions which we leverage, overlapping images should be reduced as it would confuse the user and make it difficult to track the transition. To this end, we consider curved paths with different radians and clockwise/counterclockwise flying directions. We first present a naïve solution that optimizes each flying path separately without considering their relationships. Then, we propose a greedy approach that determines the flying paths on an incremental basis. That is, for the current path to be determined, our goal is to reduce its overlap with all previously determined flying paths.

**Measuring image overlapping** We propose an exact solution to quantitatively compare the percentage of image overlapping before and after path optimization. Since each image is loaded as a texture and mapped to a quad for display, we compute the amount of quad overlap (in pixels) and derive the *average* percentage of overlapping during the flying process as follows.

$$o = \frac{\sum_{i\in N, j\in N, i<j} \sum_{t=1}^{T} Q_{i,t} \cap Q_{j,t}}{T \times A} \tag{2}$$



(a) (b) (c) (d)

**Fig. 3 One-step animation. (a) and (b) show two snapshots of the fly-out/fly-in action with curved paths and a 70% of overlap in the action. (c) and (d) show two snapshots of the card-flip action.**

where $Q_{i,t}$ and $Q_{j,t}$ denote the positions of image quads $Q_i$ and $Q_j$ at flying step $t$ and $\cap$ denotes the number of pixels overlapped by the two quads; $N$ is the total number of images considered; $T$ is the total number of flying steps; and $A$ is the entire display area.

To actually show pixel overlap in an animated fashion, we draw all quads in red without texture with the same amount of transparency at the beginning (the background color is light blue). During the flying process, we blend together overlapped quads. At each flying step, the amount of opacity accumulated for each pixel is in proportion to the number of overlaps it has.

**Straight vs. curved paths** The simplest flying path between a source and its destination is to directly follow the straight line connecting them. This solution works well when the transition only involves a few images. As illustrated in Fig. 4, moving along straight lines more likely leads to image crossover, especially around the center of the display area. Furthermore, in our spiral image layout, using a straight-line pattern would give the viewers a wrong impression. Since a new query image could be quite similar to the old one, in a typical transition, there could be many images staying on the same or adjacent layer (i.e., their rankings do not change dramatically). However, moving images only along straight lines could force these images to leave further away from its layer before turning back (shown in Fig. 4).

For our spiral layout, we draw a curved path based on the following parameters: the two endpoints (source and destination) and radian (the higher the value, the more curvy the path). We derive the radian of the curved path as follows.

$$r = \min\left\{r_0 \times d \times \left(1.0 - \frac{|\pi/4 - \theta'|}{\pi/4}\right), r_{\max}\right\} \quad (3)$$

and

$$\theta' = \begin{cases} \pi - \theta, & \theta > \pi/2; \\ \theta, & \theta \leqslant \pi/2 \end{cases} \quad (4)$$

where $r_0$ is a base radian which is a constant, $d$ is the distance between the two endpoints, $\theta$ is the angle formed between the straight path connecting the two endpoints and the $+x$ axis, and $r_{\max}$ is the maximum radian allowed. The intuition is that when the distance is large and the angle is close to 45° or 135°, then the straight path is likely to cross the center of the display area. Therefore, the radian should be large so that the curved path could be effectively deviated from the center of the display area. On the other hand, $r$ is bounded by $r_{\max}$ since a too large radian could force the flying path mostly out of the display area which is not desirable either. A naïve solution is to follow Eq. (3) for each flying path separately and determine its radian accordingly. Besides the two endpoints and radian, we also consider adding the flying direction for the following greedy optimization where we take into account the relationship among paths for occlusion reduction.

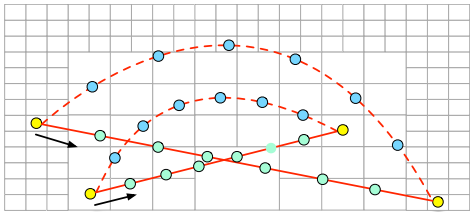


**Fig. 4 Straight vs. curved paths. Curved paths work better for the spiral layout and reduce the possibility of overlap during the flying.**

**Greedy optimization** We further present a greedy optimization method to reduce image overlapping. This greedy approach is a grid-based approximation, where each cell in the grid has the same size as the smallest remaining image displayed in the layout for flying (we assume that all these images will be shrunk to this smallest size before flying). We create an *occupancy buffer*, in which a total of $n_c \times n_t$ counters are used to keep track of the number of paths in each cell for each time step, where $n_c$ and $n_t$ are the total numbers of cells and time steps, respectively. Assuming the number of existing paths in cell $i$ at time step $j$ is $n_{i,j}$, the occlusion among these existing paths in cell $i$ at time step $j$ is $n_{i,j} - 1$ if $n_{i,j} > 0$; otherwise the occlusion is zero. Therefore, the occlusion of a newly added path will be $\sum_{i,j} n_{i,j}$, where the new path goes through cell $i$ at time step $j$. Since the paths are more likely to occlude each other around the center of the display area, we can also calculate the occlusion value as a weighted sum $\sum_{i,j} w_i n_{i,j}$, where the closer a cell to the center, the higher the weight $w_i$.

Our approach starts from an empty set, and adds the path which has the shortest Euclidean distance between the two end points at each step, since there is less space for the shorter paths to bypass the previous ones. For each path, we compute the occlusion values for several candidates, which include the straight line, clockwise and counterclockwise curvy lines with radian as 0.5, 1.0, 1.5, 2.0, and 2.5, respectively. From these

candidates, the one with the smallest value of occlusion will be selected. This process terminates when all paths are added.

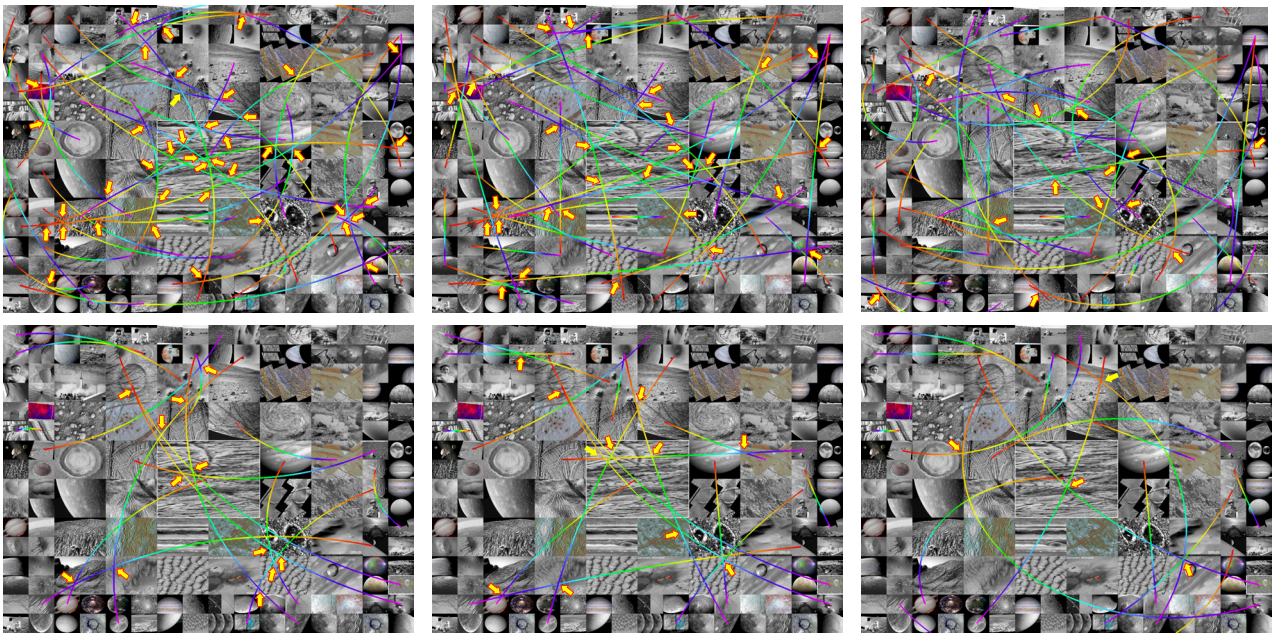
In Fig. 5, we show an example that compares different ways to determine curved flying paths. Actual image overlaps at the same time steps during the flying are highlighted with arrows. For the flying paths with the same radian, there are 41 overlaps for images decreasing their ranks and 12 overlaps for images increasing their ranks. These two numbers drop down to 30 and 11 for the naïve improvement, and further down to 15 and 4 for the greedy optimization. It is clear that our greedy solution is able to well reduce image overlapping. This improvement has been consistently observed in many of our trials.

## 4 Multi-Step Animation

Decomposing the transition into several steps facilitates the understanding of layout changes. We propose several guidelines for designing our multi-step animation. First, the animation should be always directing the viewers' attention, leading them to look at where they should focus on at the right moment. Second, we should consider action overlapping for efficiency while avoiding issuing too many different kinds of actions at the same time for clarity. Third, the animation should be as simple as possible, conveying the main idea without distracting the viewers.

### 4.1 Stage design

Following the above guidelines, we design our multi-step animation with three stages: *preparation*, *reorganization*, and *finalization*. At the preparation stage, we swap the new query image with the old one. Meanwhile, we fade out those images that will not be shown in the new layout. Removing these images as early as possible helps the viewers pay more attention to the images in the new layout. We know that when staging an action, only one idea should be seen by the audience at a time. Nevertheless, performing swapping and fading actions simultaneously does not have a negative impact on our animation. In a still scene, the eye will be attracted to movement[19]. Thus, the swapping of the old and new query images is the key action which takes the priority to attract viewers' attention, while the fading evolves gradually without distracting the viewers. After this stage, our new query image is located at the center of the display area. Other images remaining in the new layout stay put. At the reorganization stage, we reorganize the positions of images remaining from the old layout. For these images, their positions in the new layout will likely



(a) (b) (c)

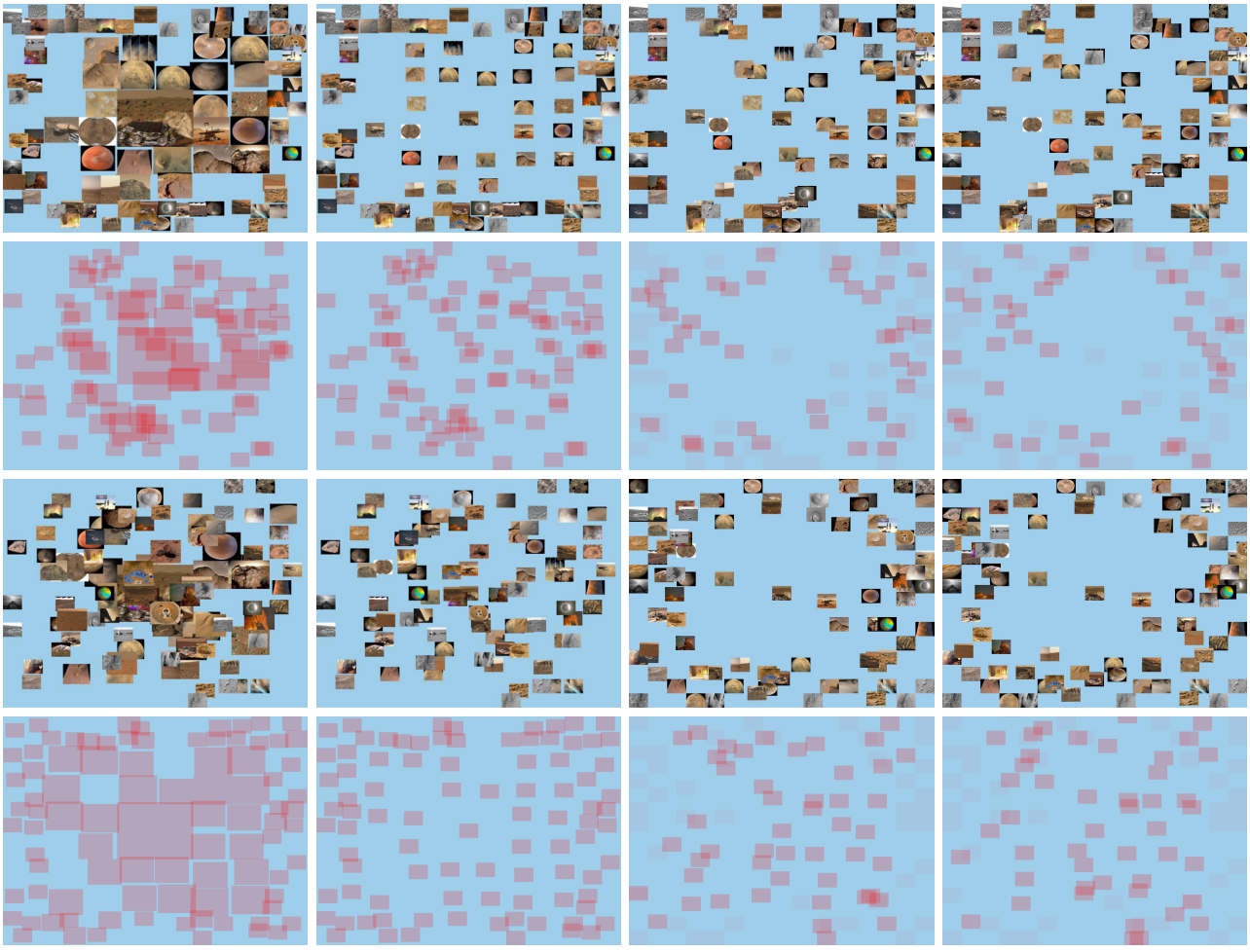
**Fig. 5 Comparison of (a) flying paths with the same radian, (b) flying paths with naïve improvement, and (c) flying paths with greedy optimization. First row: flying paths for images with decreasing ranks. Second row: flying path for images with increasing ranks. The paths are colored using a rainbow color map where the purple and red colors correspond to the beginning and end of the flying, respectively. Arrows indicate image overlaps at the same time steps during the actual flying.**

change due to the change of their similarity ranking. Each image flies to their new position following the solutions proposed in Section 3.3. After this stage, the remaining images are now located at their final positions in the new layout. At the finalization stage, we bring in newly added images by applying one of the "in" processes discussed in Section 3.2.

### 4.2 Further consideration

Besides the three stages outlined above, there are some other issues we need to consider for multi-step animation. We find that at the reorganization stage, even though we have already applied our greedy optimization scheme for flying paths, image overlapping and visual cluttering could still be rather severe. Therefore, we seek two further improvements to reduce the overlap. First, at the preparation stage, in addition to image swapping and fading, we also shrink all remaining images to the same size as the smallest remaining image in the layout. Second, at the reorganization stage, we further split the flying process into two steps: flying images with decreasing ranks first and then flying images with increasing ranks. To avoid possible distraction, we will fade still images to the background when other images are flying. Combining size shrinking and fly splitting, we are able to largely reduce image overlapping.

In Fig. 6, we show the effectiveness of reducing image overlap by adding one strategy at a time to the original flying solution where curved flying paths are used. Shrinking the size of images contributes to the most overlap reduction, from 14.2% to 3.58% for this



(a) original curved paths (b) (a) + size shrinking (c) (b) + fly splitting (d) (c) + greedy optimization

**Fig. 6 Comparison of the original flying solution with the addition of each new strategy to reduce image overlap. The transition is for the two layouts shown in Fig. 1. In (a) and (b), solutions from top to bottom are selected at 20%, 40%, 60%, and 80% of the entire flying process. In (c) and (d), the top/bottom two are selected at 33% and 66% of the flying process with flying images with decreasing/increasing ranks. The average percentages of overlapping for the entire flying process are 14.2%, 3.58%, 0.869%, and 0.375% for (a) to (d), respectively.**

example. Furthermore, when we split the flying process into two steps, the average percentage of overlap drops to 0.869%. Finally, we optimize the flying paths with our greedy strategy, further dragging down the average percentage of overlap to 0.375%. Side-by-side visual comparison also shows the gradual reduction in image overlap, especially for the images in the second and fourth rows where the bounding quads instead of the actual images are drawn. Such a trend in overlap reduction has been consistently observed and verified in many of our trials with varying numbers of images in the layout and different query images chosen.

### 4.3 AniMap — Put it all together

We present AniMap, a visualization framework that builds on top of iMap and supports animated transition. As shown in Fig. 7, AniMap utilizes multi-step animation, decomposing the animation into the preparation, reorganization, and finalization stages. At the first preparation stage, the new focus image swaps with the old focus image, and at the same time, the images that will not be shown in the new layout will be faded out and all remaining images will be shrunk to the same size as the smallest remaining image in the layout. At the second reorganization stage, images with decreasing ranks will fly to their new destinations first, and then images with increasing ranks will fly. We leverage curved flying paths for overlap reduction and apply the slow-in/slow-out timing technique for visual consistency. We also use the greedy path optimization method to further reduce the overlap and improve the readability. At the last finalization stage, all new images will fade in and at the same time, all images (including those remaining ones) will be enlarged to their target sizes.

## 5 User Study

To evaluate the effectiveness of AniMap, we performed a 3 conditions (no-animation vs. one-step animation vs. multi-step animation) $\times$ 2 input image numbers (small vs. large) $\times$ 2 remaining image percentages (small vs. large) $\times$ 2 playback speeds (normal vs. slow) $\times$ 5 questions within-subject experiment. Before the actual user study, we also conducted a pilot study on six users to identify the appropriate parameter values for the image number and percentage, and playback speed. After that, 24 new users were recruited to participate in the actual experiments. All of them are undergraduate and graduate students and each student was paid \$10 in return. Each time, up to four users conducted the user study concurrently with one of our student researchers present. All experiments were conducted in our graphics and visualization laboratory using four standard desktop PCs with the same configuration. The experiment interface consists of a single window showing the image layout and its transition with one of the three conditions (no-animation, one-step animation, or multi-step animation). The users were seated in front of 27 inch monitors with 1920 $\times$ 1080 screen resolution

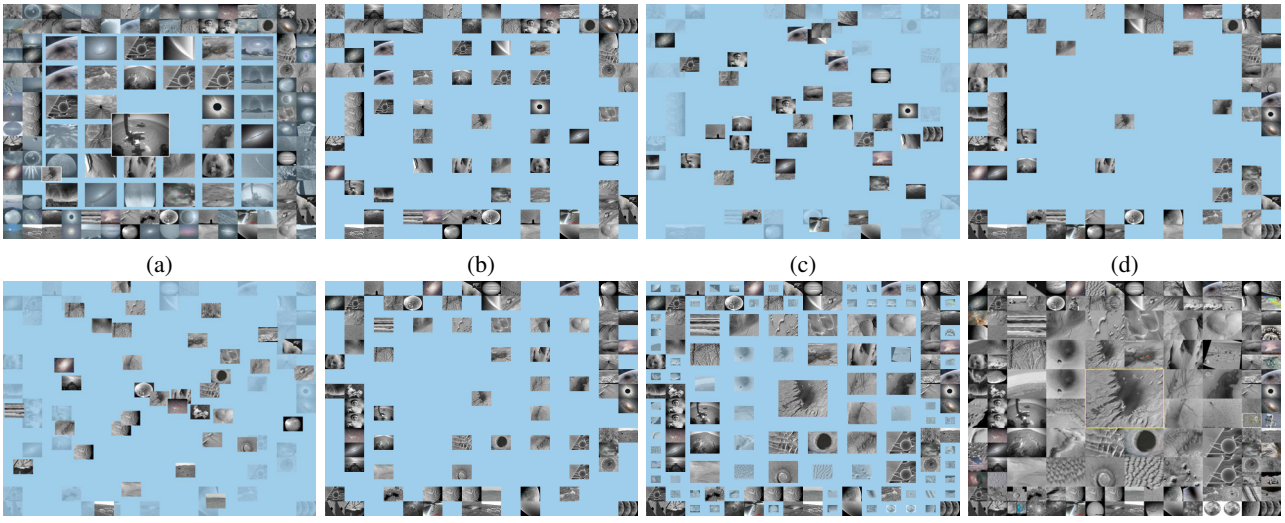



**Fig. 7 Snapshots of the AniMap. Stage 1: swapping the old and new focus images, fading out images that are not in the new layout and shrinking all remaining images (a). Stage 1 finishes (b). Stage 2: images with decreasing ranks fly (c) and the fly finishes (d). Images with increasing ranks fly (e) and the fly finishes (f). Stage 3: fading in new images while enlarging their sizes (g). The entire animation finishes (h).**

while the image layout occupied $1400 \times 1050$ pixels. They could sit in any fashion they found comfortable and were asked to answer a question for each task given. A question and its answer options with radio buttons appear on the right-hand side of the window, which were visible for the duration of each task. The users selected their answer by clicking the corresponding radio button and then clicked the "Next" button to move on to the next question.

The tasks were drawn from screen shots of our AniMap program running the Astronomy Picture of the Day (APOD), a popular online astronomy archive with thousands of handpicked pictures. Playing back the screen shots yielded the animation. For the input image number, we considered two cases: small (49 images with four layers and one repetition level) and large (145 images with five layers and two repetition levels). For the remaining image percentage, we considered two cases: small ($\leqslant 30\%$ of the input image number) and large ($\geqslant 40\%$ of the input image number). For the playback speed, our pilot study result showed that 10 seconds of animation for each task was comfortable, which we set it for the normal speed case. We doubled the playback time to 20 seconds for the slow speed case.

For no-animation, an old layout was first displayed together with the question and answer options. There was no time limit for the user to read and understand the question. Once the user clicked the "Start" button, "Backward" and "Forward" buttons could be clicked to see the old and new layouts back and forth within the time limit, which is the same as the animation time given. After that, no further interaction was allowed and the user must answer the question and click the "Next" button to move on to the next question.

For one-step animation, a repeated animation was shown. The animation started playing automatically after the user read the question and clicked the "Start" button. Within the time limit, the animation automatically repeated itself with an interval of two seconds between two playbacks. The users were instructed that they could select the answer and moved on to the next question even when the animation did not finish the playing.

For multi-step animation, the entire process of animation played only once. Similar to one-step animation, the user could select the answer and moved on to the next question before the animation finished. Except for the no-animation condition which allowed forward and backward interaction, no other forms of interaction on the animation were allowed for all three conditions.

### 5.1 Tasks

We designed the following five different questions in order to evaluate whether or not the users comprehend both the local and global layout structures under the three animation conditions. The users were asked to answer each question from multiple choices, making their best effort to answer correctly. Nevertheless, if there was no ground to make a choice, they were suggested to select "I don't know." as the answer.

Our first question considers the local evolution of image similarity rank. One image in the old layout was highlighted with a yellow boundary, and the user was asked: **Q1**. *How does the similarity rank of the image with yellow boundary change (increase, decrease or keep the same)?* We chose this question because the essential difference between the old and new layouts was the changes of image similarity ranks. The ability to track such a change was one of the key criteria in our evaluation.

The second question explores image disappearance. Two images were highlighted in different colors, and the user was asked to select the image as the answer to: **Q2**. *Given two images highlighted with yellow and green boundaries respectively, which image disappears in the new layout?* We chose this question as image appearance/disappearance is one of the most basic results of the transition. This question is also locally structured.

In our scenario, images are arranged according to their degrees of similarity along a spiral pattern from inside out. The third question measures if the animation allows the user to detect the degree of change for image similarity ranks. Two images were highlighted in different colors, and the user was asked to select the image as the answer to: **Q3**. *Given two images highlighted with yellow and green boundaries respectively, which image's similarity rank has a larger (or smaller) degree of change?*

In our fourth question, we test if the user is able to notice more global trends. Specifically, the question tests if an overall update in the number of images in the new layout is perceived. No images were highlighted and the user was asked: **Q4**. *Given the total number of images in the old layout, please estimate the number of images that remain from the old layout.* We selected this question because it is globally structured. Since

the number of newly appeared images equals the total number of images minus the number of remaining images, we asked the user to estimate the number of remaining images so that they did not need to make an extra subtraction by themselves.

Finally, we tested whether the total number of images that increase or decrease their similarity ranks could be perceived or not. No images were highlighted and the user was asked: **Q5**. *Given the number of remaining images, please estimate the number of images that increase (or decrease) their similarity ranks.* This question is more of a global question as it requires the user to read both image updates and their similarity rank changes.

### 5.2 Experiment design

The interface for each of the three conditions was demonstrated to the users before the test. They could ask questions, figure out the tasks, and see how the answers to the questions could be inferred. The experimental procedure required that the users answered all questions under each condition in order. Therefore, any cognitive shift required to move from one interface to another only occurred twice. We counterbalanced between users by presenting six different orders of three conditions with four users following one of the orders. However, within 40 tasks for each condition, the order of tasks was not counterbalanced, but rather given in the order from simple to complex so that the users could better prepare for the more difficult tasks. The user could take a short break between experimental conditions if preferred.

To help overcome the learning effect, each condition block was preceded by a practice block of five questions, selected randomly from the set of experimental tasks. The users were not aware that this initial block of five questions did not form part of the experimental data collection. Each user, therefore, completed a total of 135 tasks. To avoid possible rushing toward the end of the tasks, we did not show the question number or the overall progress to the users. The time to read questions before the start of animation was not recorded. The completion time was recorded to include the animation time and the time to answer questions. The users were instructed to focus more on the accuracy rather than the completion time. On average, it took about 90 minutes for a user to complete all the tasks, which includes the pre-experiment training, practice tasks, experimental tasks for all three animation conditions and a post-experiment questionnaire.

### 5.3 Results

Based on our experience in the pilot study, we anticipated that using multi-step animation would be faster in completion time and more accurate than using no-animation and one-step animation, but we were not sure whether or not the input image number and playback speed would have a significant effect on the performance.

Since the data we collected do not form a normal distribution for most cases, instead of using ANOVA, we used a nonparametric Kruskal-Wallis (KW) one-way analysis of variance by ranks with a standard significance level $\alpha = 0.05$ to determine statistical significance between conditions. Using correlation coefficients to analyze user behaviors, we removed three users from further analysis due to their large negative coefficient values comparing against other users. Therefore, the data collected from 21 users were used in the statistical analysis.

**Accuracy** As shown in Figs. 8a and 8c, the results for different animation conditions indicated a strong advantage for multi-step animation. KW analysis found a significant difference among the three animation conditions ($H(2) = 24.69, p \ll 0.0001$) and post-hoc analysis found that multi-step animation was significantly more accurate than no-animation ($H(1) = 17.89, p = 0.0002$) and one-step animation ($H(1) = 21.01, p \ll 0.0001$). No-animation was not significantly distinguishable from one-step animation.

Post-hoc analysis using Bonferroni corrections showed that, for Q4 and Q5, multi-step animation was significantly more accurate than both no-animation (Q4: $H(1) = 5.84, p = 0.015$, Q5:$H(1) = 32.25, p \ll 0.0001$) and one-step animation (Q4: $H(1) = 14.92$, $p = 0.0001$, Q5: $H(1) = 34.53, p \ll 0.0001$). No significant difference was found in terms of accuracy for Q1, Q2, and Q3.

No significant difference was found under different input image numbers for all three animation conditions and all five categories of question except that for Q3, one-step animation was significantly more accurate with the small image number ($H(1) = 5.37, p = 0.0205$) but multi-step animation was significantly more accurate with the large image number ($H(1) = 10.30, p = 0.0013$).

Considering the playback speeds, KW analysis

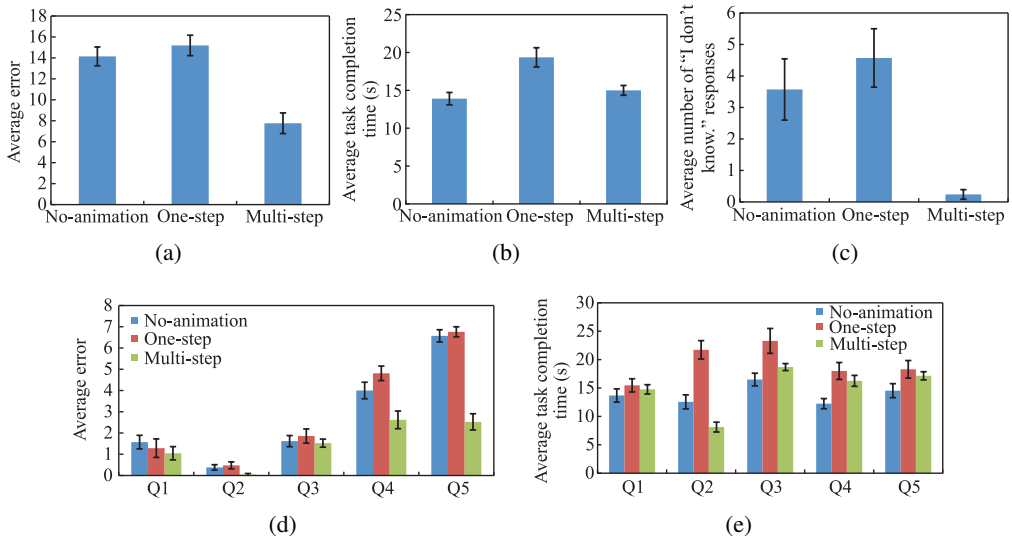



**Fig. 8 Mean values and standard errors of (a) the average number of task errors, (b) the average task completion time, (c) the average number of "I don't know." responses, (d) the average number of task errors for each category of questions, and (e) the average task completion time for each category of questions.**

results were significant at the 0.01 level under the multi-step animation condition ($H(1) = 14.36$, $p = 0.0002$) for Q3, for which the average number of errors was significantly higher with the slow playback speed.

Figure 8c shows the average number of "I don't know." responses for the three animation conditions, where users were unwilling to make an estimate. A significant difference was found among the three animation conditions ($H(2) = 25.09$, $p \ll 0.0001$), with post-hoc tests showing that multi-step animation was the fewest in the number of "I don't know." responses. Considering task variants, "I don't know." responses were all from Q4 and Q5, i.e., questions on the global layout structure.

**Task completion time** The time to complete a task was measured from when the user clicked the "Start" button to when the user clicked the "Next" button. Figures 8b and 8e show the average task completion time. We analyzed all tasks combined as well as tasks in each category of questions. In terms of task completion time, a significant difference was found among the three animation conditions ($H(2) = 13.61$, $p = 0.0001$), with post-hoc tests showing a significant interaction effect between multi-step animation and one-step animation ($H(1) = 8.01$, $p = 0.0047$). The mean time for completion for multi-step animation was 15.00 s compared to 19.37 s for one-step animation (about 29% faster). Multi-step animation and no-animation were not statistically distinguishable. For all animation conditions, the input image numbers did not have a significant effect in terms of task completion time. Considering the playback speed, KW analysis results were significant at the 0.01 level for both one-step animation ($H(1) = 8.30$, $p = 0.0040$) and multi-step animation ($H(1) = 18.08$, $p \ll 0.0001$). Decreasing the playback speed noticeably increased the task completion time.

**Subjective preferences** After the experiments, users completed a survey with six statements as listed in Table 1. Each was answered with a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree). KW analysis was conducted on ratings for each condition. Table 1 gives the pair-wise comparison results.

KW analysis results found a significant difference in rating among the three animation conditions ($H(2) = 39.32$, $p \ll 0.0001$), with post-hoc analysis showing that multi-step animation obtained significantly higher grades than no-animation ($H(1) = 29.88$, $p \ll 0.0001$) and one-step animation ($H(1) = 32.16$, $p \ll 0.0001$). The users' subjective feeling that multi-step animation was much easier for all tasks was also consistent with the previous result that multi-step animation got the fewest "I don't know." responses, i.e., the users were more willing to estimate the answer.

At the end of the survey, many users indicated in the open comments section that the movement of multi-step animation was understandable and clear. One user reported that "*It's very easy to track specific picture and a limited number of pictures.*" Many users commented in a similar way that "*It's detectable and easy to track*

**Table 1 Average ratings for six statements for each animation condition. * indicates significant differences ($p < 0.01$).**

| Statement | Multi-step animation | One-step animation | No-animation |
|---|---|---|---|
| **S1.** *It was easy to estimate the number of images remaining from the old layout.* | 4.43, *one-step, *no | 1.76 | 2.00 |
| **S2.** *It was easy to estimate the number of images that increase or decrease their similarity ranks.* | 4.38, *one-step, *no | 1.90 | 2.10 |
| **S3.** *It was easy to track the changes of image similarity ranking.* | 4.71, *one-step, *no | 2.43 | 2.43 |
| **S4.** *It was easy to detect the appearance or disappearance of images.* | 4.86, *one-step, *no | 3.14 | 3.24 |
| **S5.** *It was easy to estimate the degree of changes of image similarity ranking.* | 4.38, *one-step, *no | 2.48 | 2.81 |
| **S6.** *Overall, the solution was effective.* | 4.67, *one-step, *no | 2.38 | 2.52 |

*and relocate desired objects.*" but some users also pointed out that "*It's much slower than no-animation.*" and "*It could be more than perfect if the speed of animation could be controlled.*"

For one-step animation, users complained that "*The shift between old layout and new layout totally confuses me and destroys my impression on the old layout if I have any.*" and "*It was difficult to track changes and estimate the numbers.*" Furthermore, some users pointed out that one-step animation was time consuming.

The users also expressed their preferences for no-animation because it was easy to focus and would not be interrupted by animations. Another important advantage of no-animation was the control of switching between the old and new layouts. However, in terms of estimating the numbers in Q4 and Q5, there was no clue for guessing under the no-animation condition.

**Discussion of user study results** Overall, we have compelling evidence that using multi-step animation was significantly more accurate than the other two conditions in terms of answering questions and was significantly faster than one-step animation in terms of completion time.

In terms of accuracy, we found significant differences among the three animation conditions between questions on local and global structures. No significant difference was found for questions on local structures, which asked about the change of image similarity ranks and the disappearance of images. We discovered that it was not difficult to obtain the right answer as long as the users were able to identify highlighted images in the new layout. However, multi-step animation was significantly more accurate than the other two conditions when answering questions on global structures such as estimating the number of remaining images. Different image sizes and limited comparison time added the difficulty to estimate under no-animation and one-step animation conditions. Some users might try their best to get an estimated answer, while many other users selected "I don't know." as the answer without much thinking or reasoning, increasing the errors in these two conditions. This was why the most number of "I don't know." answers came from Q4 and Q5.

In terms of task completion time, one-step animation was significantly slower than multi-step animation and no-animation, and no significant difference was found between multi-step animation and no-animation. Post-hoc analysis found a significant difference between multi-step animation and no-animation at the slow playback speed ($H(1) = 8.59$, $p = 0.0034$), but not at the normal playback speed. Besides, no significant difference was found in playback speed for the no-animation condition, but multi-step animation was significantly slower in completion time at the slow playback speed ($H(1) = 18.18$, $p \ll 0.0001$). Therefore, it is very likely that, using multi-step animation at the slow playback speed, more task completion time needed was simply due to the longer animation time itself. The users could not answer the questions without waiting for the animation playing to that specific step. For example, if the users want to estimate the number of images that increase their similarity ranks, they must wait until the second flying process begins at the reorganization stage.

Our results indicated that the input image number did not have a significant effect on the accuracy and task completion time. Only one exception need to be noticed, that is, multi-step animation was significantly more accurate with a large number of input images for Q3, which asked the users to track images with larger or smaller similarity rank changes. Since the large number of image input consists of five layers with two repetition levels, it might be easier to identify the difference between each layer, so as to identify the degree of image similarity rank change.

No significant difference was found in terms of accuracy under different playback speeds except that, for Q3, multi-step animation generates significantly

more errors at the slow playback speed. It could be possible that, given the slow playback speed, the users would be more likely to forget the positions of images in the old layout after the animation finished, leading to wrong answers to the questions on similarity rank change.

## 6 Conclusions and Future Work

Animated transition plays an important role in helping the viewers grasp the changes of data, both locally and globally. We have presented the AniMap, an animated transition solution specifically designed for similarity-based tiled image layouts. The image and video results demonstrate the effectiveness of AniMap in terms of improved understanding and increased engagement. We have also conducted a user study to compare our multi-step animated transition against no-animation and one-step animation solutions. The results show that multi-step animation significantly improves the accuracy (especially for questions on global layout structure) and the overall ratings. We plan to further investigate flying paths by staggering the start times for images going different distances and incorporating temporal distortion factors[20]. We would also apply the general approach presented in this work to other visualization applications where such a transition is not inherently given.

### Acknowledgements

This work was supported in part by the US National Science Foundation (Nos. IIS-1017935 and CNS-1229297). We would like to thank Dr. Robert Nemiroff for providing the Astronomy Picture of the Day (APOD) image collection to us. All the images at the APOD website are credited to the owners or institutions where they originated.

### References

[1] J. Heer and G. G. Robertson, Animated transition in statistical data graphics, *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1240-1247, 2007.

[2] B. B. Bederson and A. Boltman, Does animation help users build mental maps of spatial information? in *Proceedings of IEEE Symposium on Information Visualization*, 1999, pp. 28-35.

[3] T. Bladh, D. A. Carr, and M. Kljun, The effect of animated transitions on user navigation in 3D treemaps, in *Proceedings of IEEE Symposium on Information Visualization*, 2005, pp. 297-305.

[4] C. Gonzales, Does animation in user interfaces improve decision making? in *Proceedings of ACM SIGCHI Conference*, 1996, pp. 27-34.

[5] D. Curvehambault, H. C. Purchase, and B. Pinaud, Animation, small multiples, and the effect of mental map preservation in dynamic graphs, *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 539-552, 2011.

[6] G. G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. T. Stasko, Effectiveness of animation in trend visualization, *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1325-1332, 2008.

[7] M. Donskoy, Window navigation with and without animation: A comparison of scroll bars, zoom and fisheye view, in *Proceedings of ACM SIGCHI Extended Abstracts*, 1996, pp. 279-280.

[8] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, Does organization by similarity assist image browsing? in *Proceedings of ACM SIGCHI Conference*, 2001, pp. 190-197.

[9] C. Wang, J. P. Reese, H. Zhang, J. Tao, and R. J. Nemiroff, iMap — A stable layout for navigating large image collections with embedded search, in *Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis*, 2013.

[10] B. Tversky, J. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, vol. 57, no. 4, pp. 247-262, 2002.

[11] R. Baecker and I. Small, Animation at the interface, in *The Art of Human-Computer Interface Design*, B. Laurel, Ed. Addison-Wesley, 1990, pp. 251-267.

[12] S. E. Hudson and J. T. Stasko, Animation support in a user interface toolkit: Flexible, robust, and reusable abstractions, in *Proceedings ACM Symposium on User Interface Software and Technology*, 1993, pp. 57-67.

[13] G. G. Robertson, S. K. Card, and J. D. Mackinlay, The cognitive coprocessor architecture for interactive user interfaces, in *Proceedings ACM Symposium on User Interface Software and Technology*, 1989, pp. 10-18.

[14] G. G. Robertson, J. D. Mackinlay, and S. K. Card, Cone trees: Animated 3D visualizations of hierarchical information, in *Proceedings of the ACM SIGCHI Conference*, 1991, pp. 189-194.

[15] J. D. Mackinlay, G. G. Robertson, and S. K. Card, The perspective wall: Detail and context smoothly integrated, in *Proceedings of ACM SIGCHI Conference*, 1991, pp. 173-176.

[16] K. Yee, D. Fisher, R. Dhamija, and M. Hearst, Animated exploration of dynamic graphs with radial layout, in *Proceedings of IEEE Symposium on Information Visualization*, 2001, pp. 43-50.

[17] J. J. van Wijk and W. A. A. Nuij, A model for smooth viewing and navigation of large 2D information spaces, *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 447-458, 2004.

[18] 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*, vol. 14, no. 6, pp. 1539-1148, 2008.

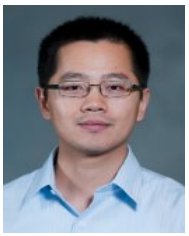
[19] J. Lasseter, Principles of traditional animation applied to 3D computer animation, in *Proceedings of ACM SIGGRAPH Conference*, 1987, pp. 35-44.

[20] P. Dragicevic, A. Bezerianos, W. Javed, N. Elmqvist, and J.-D. Fekete, Temporal distortion for animated transitions, in *Proceedings of ACM SIGCHI Conference*, 2011, pp. 2009-2018.

**Huan Zhang** is a software engineer in Dematic Corp. She received her BS degree in Computer Science from Hunan University of Commerce, China, in 2010, and her MS degree in Computer Science from Michigan Technological University in 2012. Her research interests include animation, data visualization, and user interfaces.

**Jun Tao** is a PhD student of computer science at Michigan Technological University. His research interests include flow visualization, image resizing, and mesh editing. He received his BS degree in software engineering from Sun Yat-sen University, China, in 2008, and his MS degree in computer science from Michigan Technological University in 2010.

**Fang Ruan** is a software engineer in Google Inc. She received her MS degree in Electrical Engineering from Huazhong University of Science and Technology, China, in 2004. From 2004 to 2009, she was a software engineer in Guangdong Nortel, China. She received her MS degree in computer science from Michigan Technological University in 2011. Her research interests include multimedia system, animation, and large-scale data analysis.



**Chaoli Wang** is an assistant professor of computer science at Michigan Technological University. His research focuses on large-scale data analysis and visualization, high-performance computing, and user interfaces and interaction. He received his BEng and MEng degrees in computer science from Fuzhou University, China, in 1998 and 2001, respectively, and a PhD degree in computer and information science from the Ohio State University in 2006. From 2007 to 2009, he was a postdoctoral researcher at the University of California, Davis. He has served on the program committees of the IEEE SciVis, EuroVis, and IEEE PacificVis.