GACA: Group-Aware Command-based Arrangement of Graphic Elements

Pengfei Xu[1] **Hongbo Fu**[2] **Chiew-Lan Tai**[1] **Takeo Igarashi**[3]
[1]HKUST [2]City University of Hong Kong [3]The University of Tokyo

ABSTRACT

Many graphic applications rely on command-based arrangement tools to achieve precise layouts. Traditional tools are designed to operate on a single group of elements that are distributed consistently with the arrangement axis implied by a command. This often demands a process with repeated element selections and arrangement commands to achieve 2D layouts involving multiple rows and/or columns of well aligned and/or distributed elements. Our work aims to reduce the numbers of selection operation and command invocation, since such reductions are particularly beneficial to professional designers who design lots of layouts. Our key idea is that an issued arrangement command is in fact very informative, instructing how to automatically decompose a 2D layout into multiple 1D groups, each of which is compatible with the command. We present a parameter-free, command-driven grouping approach so that users can easily predict our grouping results. We also design a simple user interface with pushpins to enable explicit control of grouping and arrangement. Our user study confirms the intuitiveness of our technique and its performance improvement over traditional command-based arrangement tools.

Author Keywords

Layout editing; alignment; arrangement commands; group-aware

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces – Graphical user interfaces (GUI)

INTRODUCTION

Creating accurate layouts of graphic elements is vital for many graphic applications. Almost all modern graphic editors such as Adobe Illustrator and Microsoft PowerPoint support command-based arrangement operations for precise alignment and/or equal spacing of graphic elements. An arrangement command is essentially a 1D operation associated with an arrangement axis and applied to a set of user-selected elements. For example, the command "align top" is associated with a horizontal arrangement axis and changes only the vertical coordinates of the selected elements.

CHI 2015, April 18 - 23 2015, Seoul, Republic of Korea.

http://dx.doi.org/10.1145/2702123.2702198

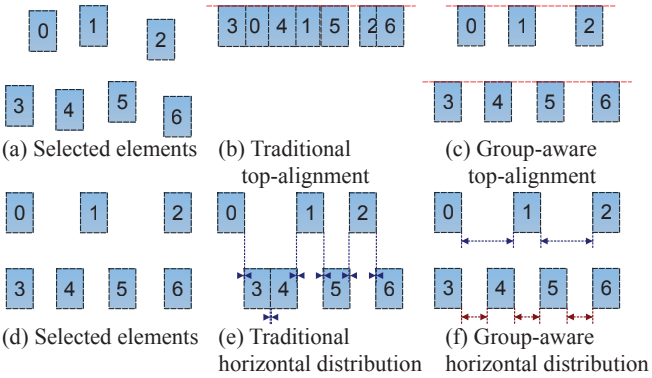


Figure 1. When applied to the same sets of selected elements, our group-aware technique, GACA, achieves more desired results (right) than traditional command-based arrangement (middle).

We observe that an arrangement command works well for a group of elements that are distributed along the corresponding axis, but might cause unexpected results when the selected elements exhibit a 2D pattern instead, i.e., there are multiple element groups distributed perpendicular to the arrangement axis. Figure 1 (a) shows a simple top alignment example. It is easy to see that there are two vertically distributed groups of elements: one containing elements $\{e_0, e_1, e_2\}$ and the other containing elements $\{e_3, e_4, e_5, e_6\}$. A simple top-alignment of all elements $\{e_0, ..., e_6\}$ thus would cause a typically undesired result in Figure 1 (b) instead of a more desired one in Figure 1 (c). To circumvent this problem, the user has to select each group of elements and apply the command to individual groups. Similarly, a *group-oblivious* horizontal distribution of all elements $\{e_0, ..., e_6\}$ in Figure 1 (d) causes an undesired result in Figure 1 (e). In short, 2D layout editing through traditional arrangement commands often involves many steps of element selection and command invocation, which could be very tedious and error-prone [17].

We aim to improve the performance of command-based arrangement tools for achieving 2D layouts, involving multiple rows and/or columns of well aligned and/or distributed elements. Motivated by the above observation, we intend to automatically derive desired groups present in the selected elements, when an arrangement command is issued. The issued command suggests the search for such groups perpendicular to the arrangement axis, reducing to a one-dimensional grouping problem. We present a parameter-free grouping approach to produce easily predictable results. We also introduce a simple but effective user interface with pushpins to enable explicit user control (Figure 5).

Our technique, which we call GACA, is able to significantly reduce the number of selection operations and arrangement

commands for achieving 2D layouts, which are commonplace in desktop publishing, graphic design and preparation of presentations. For instance, to turn the layout in Figure 1 (d) into the one in Figure 1 (f), two selection operations and two group-oblivious horizontal distribution commands are reduced to a single selection operation and a single group-aware horizontal distribution command. The intuitiveness of GACA and its performance improvement over traditional command-based arrangement tools are confirmed by our user study. We expected our tool to be particularly useful for professional designers who often need to design lots of layouts.

RELATED WORK

Various layout editing techniques like constraint-based techniques [3, 11, 15] have been proposed and even integrated into commercial graphic editors like OmniGraffle, Visio, and InDesign. Since our goal is to improve arrangement commands, which are fundamental in these editors and can coexist with their other arrangement tools, below we focus on reviewing those existing tools that are closely relevant to arrangement commands only.

It is hard to track down the history of command-based arrangement tools [13, 4]. Although the underlying technology of such tools has advanced very little since day one, due to their simplicity and power they are still being offered by almost all modern graphic editors. Such commands are very useful for improving geometric relationships between selected elements in terms of both alignment and distribution. However, as discussed previously, due to the 1D nature, an arrangement command often gets confused by the existence of multiple elements groups distributed perpendicular to the arrangement axis.

Unlike arrangement commands, which achieve alignment tasks in an indirect way, snapping techniques [2, 1, 5, 7] allow precise positioning in a more direct way. They are more suitable for aligning individual elements one by one but provide little control over precise spacing of elements. Thus snapping and command-based arrangement complement each other, and they are often used together for complex layout editing tasks. We expect that GACA can and should be used together with snapping and other arrangement techniques in practical applications. Therefore, our evaluation focused on the performance of GACA compared to traditional arrangement commands.

There are several attempts to beautify rough layouts of elements either locally [9] or globally [12, 6]. Like snapping, which can be considered as the simplest local beautifier, the local beautifiers [9] are often direct and more suitable for alignment but not distribution tasks. In contrast, the global beautifiers [20] are essentially indirect, often invoked by a single "beautification" command, and thus akin to command-based arrangement. However, unlike traditional command-based arrangement, global beautification requires an input layout to be very close to a desired layout in a user's mind. For example, given a layout in Figure 3 (a), it is very likely that beautification solutions would automatically decide to

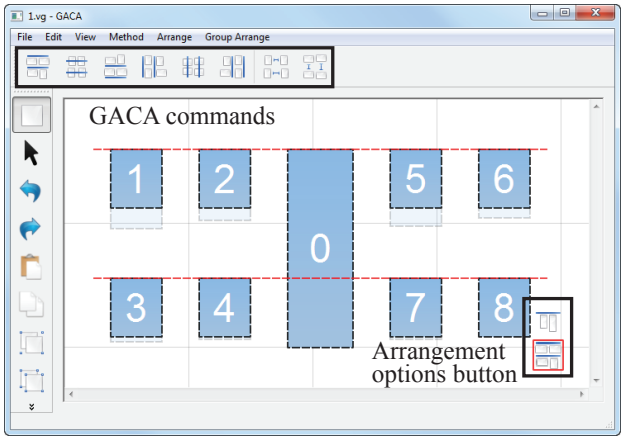


Figure 2. Main user interface of our system. Our current system supports 8 group-aware arrangement commands.

bottom-align all the elements, rather than the result in Figure 3 (b). In contrast, our goal is to enhance the arrangement commands while still allowing the user to explicitly control the layout by manually issuing proper commands.

On the other hand, GACA is inspired by such layout beautification techniques by inferring relationships among selected elements, and thus requires these elements to roughly form rows or columns. However, while the global beautifiers have a higher degree of intelligence, they can easily cause unwanted results because of ambiguity in the input layout [9]. In contrast, our technique inherits the explicit control feature of command-based arrangement tools and exploits the information carried in the commands, which greatly reduces the level of ambiguity.

There exist many advanced graph layout algorithms (e.g., [18, 14]), which typically take a large set of graphic elements and turn them into visually or artistically pleasing layouts, either automatically or with little intervention. They have very different goals from ours. For example, the technique proposed by Reinert et al. [14] evenly distributes graphical primitives to reflect the artistic goals inferred from several interactively placed primitives. In other words, their focus is to let the computers design layouts for the users, with no or little user intervention. In contrast, we aim at designing a better tool to help users design layouts themselves. With our tool, the user explicitly edits the layout by a series of manual selection operation and command invocation.

Our work is also related to perceptual grouping of elements, which aims to group elements with certain common characteristics. For instance, Nan et al. [10] integrate the well-known Gestalt principles, including similarity, proximity, continuity, closure, and regularity, to detect groups of objects in architectural drawings. Xu et al. [19] identify groups of elements covered by a user-specified scribble using proximity, similarity, and common region in visual perception. It is possible to adapt these techniques for our 1D grouping problem. However, these algorithms are often sensitive to the choice of parameters, possibly causing unpredictable results, as illustrated in Figure 6. Instead, we take a more partitioning-based solution and introduce a parameter-free approach.

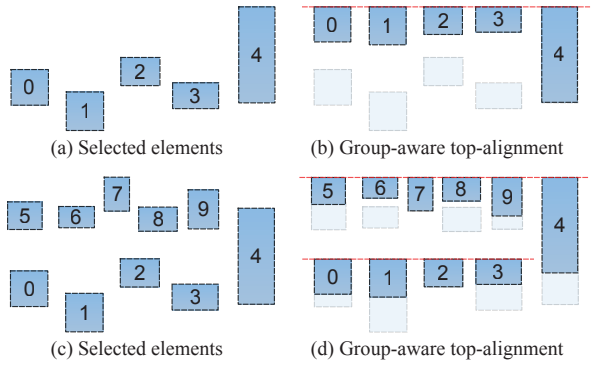


Figure 3. Group-aware top-alignment. (b) and (d) are group-aware alignment results of (a) and (c), respectively.

USER INTERFACE

As shown in Figure 2, our current system supports commonly used commands for vertical alignment ("align left", "align center", "align right"), horizontal alignment ("align top", "align middle", "align bottom"), horizontal distribution ("distribute horizontally") and vertical distribution ("distribute vertically"). Each command is associated with either a horizontal or a vertical arrangement axis. In our prototype, we simply use rectangles to represent elements, since their arrangement is often operated at the bounding-box level.

We will describe our main user interface using top-alignment as an example. The idea can be easily generalized to other arrangement commands. Since the top-alignment command has a horizontal arrangement axis, to get desired results the user needs to roughly place elements of interest to form a pattern consisting of one (Figure 3 (a)) or multiple (Figure 1 (a)) horizontal rows of elements. It is possible to have elements (e.g., element e_4 in Figure 3 (c)) spanning multiple rows. To top-align the elements of interest, the user first selects them and then issues the "align top" command from a toolbar (in our current implementation) or menu to the selected elements.

If there is only a single row of elements to be top-aligned, our group-aware top-alignment produces the same result (Figure 3 (b)) as group-oblivious top-alignment. For simplicity, our tool aligns the top edges of the selected elements with the top edge of the topmost element (i.e., element e_4 in this example). In case there are multiple rows of elements, our tool automatically identifies such rows and then applies the top-alignment command to the individual rows of elements, often leading to more desired results (Figure 1 (c) and Figure 3 (d)).

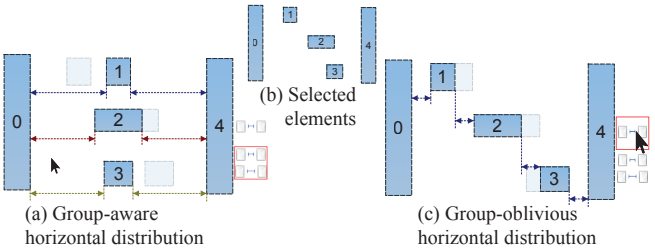


Figure 4. The user may easily switch between the results of group-aware (a) and group-oblivious (c) arrangement (horizontal distribution in this example) using the "Arrangement Options" button. The input selected elements are shown in (b).

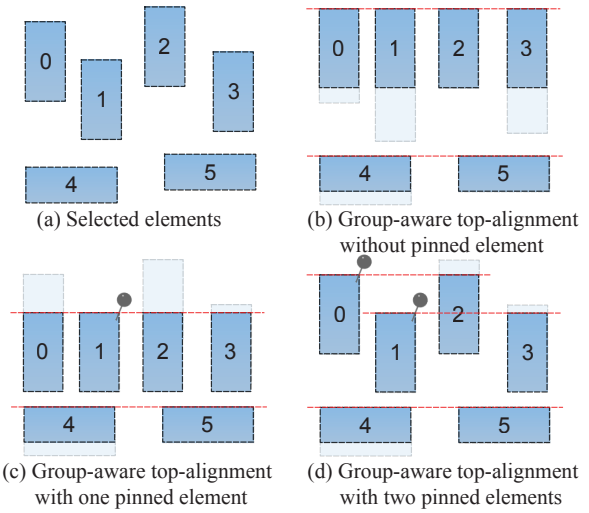


Figure 5. The user may use pushpins to specify reference elements (c) and additional arrangement groups (d). (a) is the input selected elements and (b) is the default result by group-aware top-alignment.

The user may use the provided visual cues (Figure 4 (a)) to better understand how the selected elements are moved by a certain command. First, after issuing a command, the elements before movement are displayed semi-transparently. This visual effect disappears if the user deselects the elements or issues another arrangement command. Second, to emphasize the achieved arrangement result, the alignment line(s) or the equal-spacing arrow(s) appears after the user issues an alignment or distribution command, respectively. Such lines or arrows fade away within seconds to avoid visual clutter. Please see the accompanying video for live demos.

It is not always the case that group-aware arrangement produces more user-desired results than group-oblivious arrangement. The user may easily switch between the results by group-aware and group-oblivious arrangement using a simple "Arrangement Options" button (Figures 2 and 4(b)), which is similar to the "Paste Options" button in Microsoft Office products. This button appears next to the selected elements and does not show up when group-aware and group-oblivious results are the same. It disappears after a deselection operation.

The user may use the pushpin tool to explicitly specify reference elements or determine arrangement groups. *A single pushpin* can be employed to indicate the reference element for alignment. The user can place a virtual pushpin at a selected element by right-click to fix its position during subsequent arrangement. A second right-click on the same element removes the associated pushpin. As shown in Figure 5, without such a pushpin the top-alignment command uses the topmost element, i.e., element e_2, as the reference element to align elements $\{e_0, e_1, e_2, e_3\}$ (Figure 5 (b)). If e_1 is pinned, it becomes the reference element, leading to a different alignment result (Figure 5 (c)). The user may use *multiple pushpins* to control the determination of arrangement groups. For

example, by default, elements $\{e_0, e_1, e_2, e_3\}$ are considered within a single arrangement group. A single pushpin does not change the grouping behavior. But if pins are placed on both e_0 and e_1, this group breaks into two groups (e_0 and e_2; e_1 and e_3), producing the alignment result as shown in Figure 5 (d).

METHODOLOGY

Our core problem is how to automatically identify the arrangement groups distributed along the direction perpendicular to a given arrangement axis. The design of our solution is driven by a key guideline: the technique should be very intuitive to use. In other words, after getting familiar with GACA, the user should be able to completely predict the result of a given arrangement task. This motivates us to design a *parameter-free* approach.

Group-Aware Alignment

Straightforward Approach. Our first attempt was to adopt a perception-based clustering approach, similar to Nan et al. [10]. Given an arrangement axis, the problem here is reduced to a 1D clustering problem, e.g., by clustering the top edges of the selected elements by proximity in the case of top-alignment. This approach is sensitive to the choice of parameters. For example, to turn the layout in Figure 3 (a) into the one in Figure 3 (b), a large threshold is needed to cluster all elements into a single group. However, the same threshold may undesirably group all elements in Figure 3 (c) into a single group. A small clustering threshold would require the input layout to be very close to the desired layout, and thus result in a beautification-like arrangement scheme, whose behavior is different from that of command-based arrangement (see the discussion in related work). Moreover, such a beautification-based approach is still parameter-sensitive. For example in Figure 6 (a), it is unclear which group element e_4 belongs to. Hence such a clustering approach easily leads to multiple possible results (Figure 6 (b)–(d)), which are unpredictable to the user. A suggestion-based interface (e.g., [9, 8]) might be adapted for ambiguity resolution. However, combinations of local ambiguous relationships may easily lead to a large set of global suggestions. How to effectively present such suggestions to the user is not obvious.

Our Approach. We aim to retain the power of traditional arrangement commands but avoid their undesired results of overlaps after alignment (Figure 1 (b)). This motivates us to design a two-step approach: first group elements in a relaxed setting (similar to perception-based clustering, using a large clustering threshold) and then refine the groups by examining undesired overlaps between elements if they get aligned in the same group. The two steps can be achieved by *graph construction* and *graph partitioning*, respectively.

First, our approach *dynamically* constructs an undirected graph (Figure 7) after an alignment command is issued. Each node in the graph represents an element in the selection. An edge is added to connect a pair of nodes if their corresponding elements *potentially* belong to the same group. Each connected component of the graph thus corresponds to a potential alignment group.

The key to graph construction is to determine if an edge is needed between a pair of nodes. Again we use top-alignment to illustrate the idea. We assume that the selected elements are defined in a 2D Cartesian coordinate system, with the x-axis pointing rightward and the y-axis downward. We project the vertical edges of their bounding boxes to the y-axis. If these edges overlap, we consider the corresponding elements as in a single row. This is largely consistent with perception-based clustering, but does not involve any parameter. For example, the projected vertical edges of elements $\{e_0, e_1\}$ in Figure 7 (a) overlap and form a group, and similarly, elements $\{e_2, e_3, e_4\}$ form another group. Requiring all pairs of elements to overlap each other at the y-axis is too stringent a requirement, making it difficult to handle, for example, the selected elements in Figure 7 (b), where the projected vertical edges of elements e_1 and e_2 do not overlap. Our relaxed requirement thus only requires the projected vertical edge of an element to overlap that of at least one of the elements in the same group. This implies that the presence of an element of large height (e.g., element e_3 in Figure 7 (c)) would allow a large height deviation of elements in the same group. However this rule alone would misidentify elements $\{e_0, ..., e_5\}$ in Figure 7 (c) as a single group. This happens when an element spans across multiple desired rows of elements. Fortunately, this problem can be easily solved by our next graph partitioning step.

Second, we perform a simple graph partitioning procedure on each connected graph component *if necessary* (to avoid serious overlaps between alignment elements). We check if any pair of elements have their projected horizontal edges to the

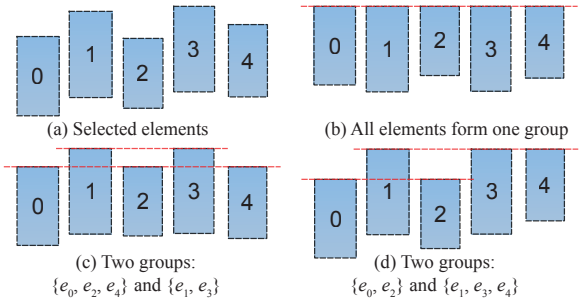


Figure 6. A clustering-based approach may lead to multiple possible results, highly dependent on the clustering threshold.

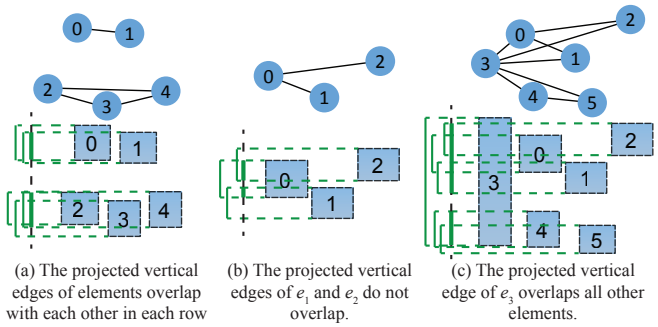


Figure 7. Illustration of graph construction for top-alignment.

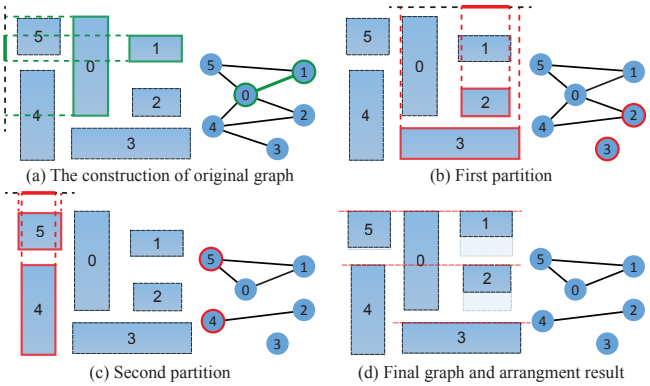


Figure 8. Illustration of graph partitioning for top-alignment.

Algorithm 1 Graph Partitioning

```
Find partition pairs set P
while P is not empty do
    Find partition pair p = (e_i, e_j) with smallest error
    if e_i and e_j belong to the same component G then
        for all e ∈ G do
            if d(e, e_i) < d(e, e_j) then
                Assign e to e_i
            else
                Assign e to e_j
            end if
        end for
        Update graph
    end if
    Remove p from P
end while
```

x-axis (the arrangement axis) overlapped (e.g., elements e_2 and e_3 in Figure 8 (b)). Such a pair of elements (e_i, e_j) may not exist. If it exists, the two elements e_i and e_j are not supposed to be in the same arrangement group and thus induce a graph partitioning on the graph component. The partitioning is done by assigning each of the rest of the elements, denoted as e, to e_i or e_j depending on whether the top edge of e is closer to that of e_i or e_j. For example, element e_4 in Figure 8 (b) is assigned to the group containing element e_2. The edges crossing the resulting two groups are removed (making element e_4 belong to the row of element e_2). We use all the identified element pairs to partition the graph incrementally. The order is according to the error of a pair of elements (e_i, e_j), denoted $d(e_i, e_j)$, defined as the distance between the alignment edges of e_i and e_j (e.g., the top edges for top-alignment). Pairs with smaller errors have higher priority of being used to partition the graph. In Figure 8, the pair (e_2, e_3) has a smaller error compared with (e_4, e_5), so it is used to partition the graph first. Algorithm 1 gives the detail of the partition process. Every pair of user-specified pushpins in the same connected component will also trigger the same graph partitioning step performed on the component.

Group-Aware Distribution

We use a slightly different way to construct the graph for distributing a set of selected elements in a group-aware manner.

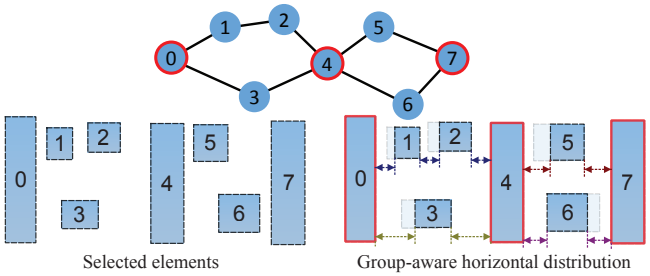


Figure 9. An example of group-aware horizontal distribution. Automatically fixed elements are highlighted in red.

Let us take horizontal distribution as an example. An edge is established between a pair of elements if and only if their projected vertical edges overlap each other *and* they are horizontally adjacent (i.e., no other element in-between them). For example, in Figure 4 (b), elements e_0 and e_2 are connected by an edge, but elements e_0 and e_4 are not. We enforce such a neighborhood condition since equal spacing aims to equalize the spacings between neighboring elements.

Next we identify the horizontal distribution groups in the constructed graph. First, we identify the horizontally outermost elements (e.g., elements e_0 and e_7 in Figure 9) in each connected component and fix their positions. Second, we fix the nodes whose degrees are greater than two. Such nodes correspond to elements that might be shared by multiple distribution groups (e.g., element e_4 in Figure 9). Without fixing those nodes, their positions after distribution would highly depend on the order of applying the distribution command to individual groups, making the distribution results hard to predict. Third, the user-pinned elements, if any, are also fixed as well. Finally, we identify a path starting from one fixed element to another fixed element as a distribution group. For instance, the example in Figure 9 results in four distribution groups (elements $\{e_0, e_1, e_2, e_3\}$; elements $\{e_0, e_3, e_4\}$; elements $\{e_4, e_5, e_7\}$; elements $\{e_4, e_6, e_7\}$).

USER STUDY

To evaluate the effectiveness of GACA we conducted a user study, which consisted of two parts. In Study I we examined the intuitiveness of GACA and in Study II we compared GACA to traditional command-based arrangement tools. The same group of participants helped in both studies.

Study I: Evaluation of Intuitiveness

We recruited twelve university students to help with our study. All of them were familiar with the behavior of traditional command-based arrangement tools. Three participants even considered themselves as experts in using these tools.

There were in total three sessions: tutorial session, practice session and questionnaire session. In the tutorial session, each participant was briefed on how to use GACA by showing them eight simple static examples, with pairs of layouts before and after issuing a certain command. Please refer to the supplementary material for the tutorial examples. In the practice session, we first demonstrated the user interface of GACA, including the use of pushpins and the "Arrangement

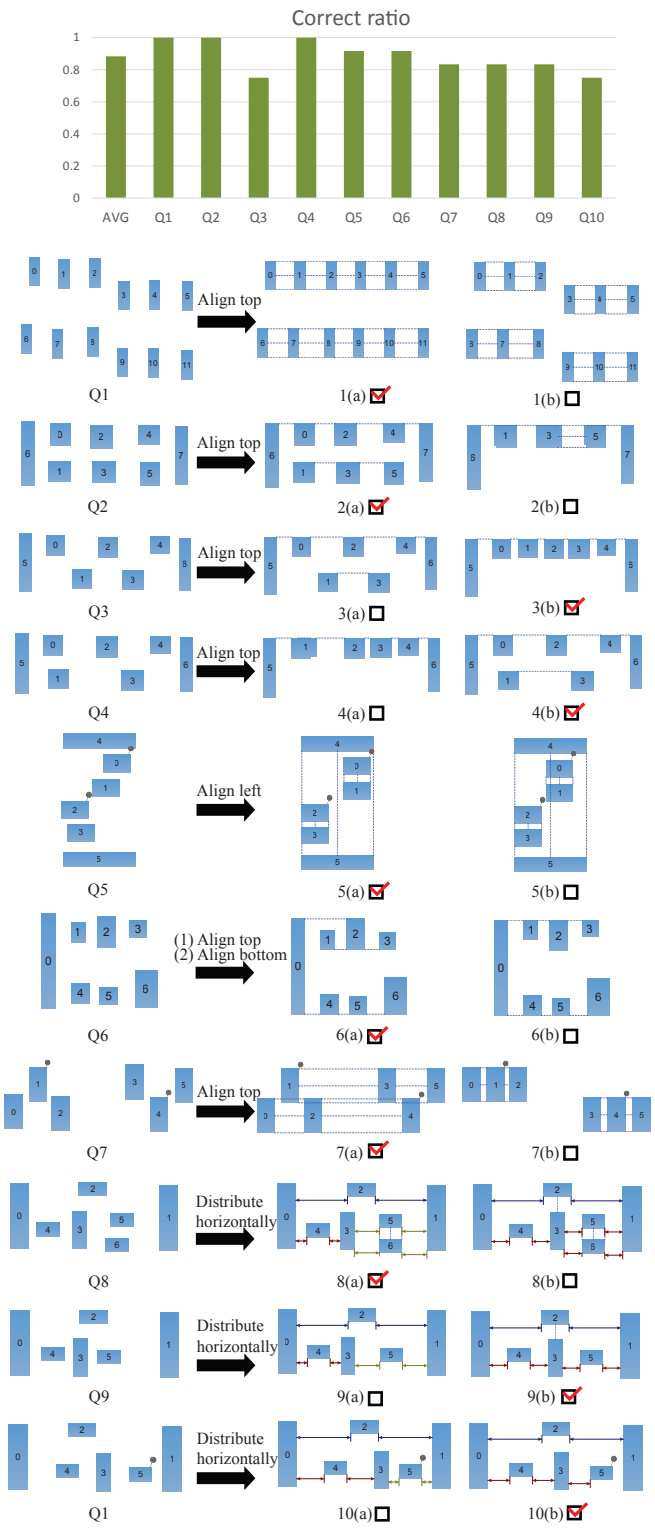


Figure 10. The high correct ratios (top) from the questionnaire (Q1–Q10) indicate the intuitiveness of GACA. The correct choices are checked.

Options" button. The participants then tried GACA. They could try the tutorial examples, or apply different commands to layouts created by themselves. When the participants were first introduced to the pushpin tool, some of them were curious about the results with multiple pushpins. They noticed that conflicts may arise if multiple pinned elements were in a single group. After they were told about another function of the pushpin tool, i.e., forming different groups, they felt comfortable and agreed that the results were logical. When they tried the tool, they quickly understood how it worked and conveyed that the visual feedback helped them to perceive the arrangement results.

After they felt confident with the use of GACA, they were asked to complete a ten-question form (Figure 10). In each question, the participants were given a layout of elements and asked to predict the result of applying a given command of GACA. Four choices were given: two possible arrangement results (Figure 10), "neither", and "not sure". After the questionnaire session, we briefly explained the answer to each question. It took about 30 minutes on average for each participant to complete Study I.

Figure 10 plots the average score of each question over the participants. It shows that the arrangement results were highly predictable to the participants, indicating the intuitiveness of GACA. The performance for Q3, the question we predicted to be the most difficult, was the lowest. This is possibly because some participants might get confused by the existence of the perceptually salient groups (elements e_0, e_2, and e_4; elements e_1 and e_3), which, however, were grouped into a single row by GACA (see more discussion in the "Discussion" section). From the algorithmic point of view, the layouts in Q3 and Figure 3 (a) were treated in a very similar way.

Study II: Comparison with Traditional Tools

To better test the performance of traditional and group-aware arrangement commands, no other arrangement tools including snapping were provided in the study. Traditional arrangement commands were integrated into our software and could be invoked from the toolbar (see the accompanying video). Specifically, we provided six traditional alignment commands (align top/middle/bottom/left/center/right) and two traditional distribution commands (distribute horizontally/vertically), whose implementation was similar to the ones in MS PowerPoint 2013.

Tasks and apparatus. There were in total 7 tasks (Figure 11). The tasks were designed to simulate a layout creation process. In each task, the participants were asked to create a target layout by moving a grid of numbered elements from an input zone to a target zone (see the accompanying video). The general operations supported in the study included selection (by single click or rectangle selection), move, grouping and ungrouping. Element creating, resizing and removal were disabled.

The study was conducted on an ordinary PC (DELL OptiPlex 960), with 3.00 GHz Intel Core 2 Duo CPU and 4.00 GB RAM. The target layout of each task was displayed on a sep-

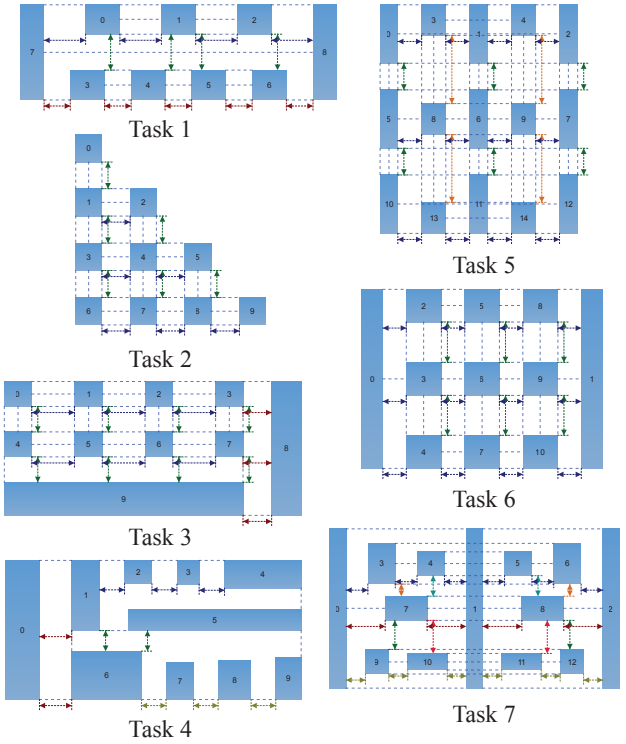


Figure 11. The tasks in Study II. They were designed to simulate a layout creation process.

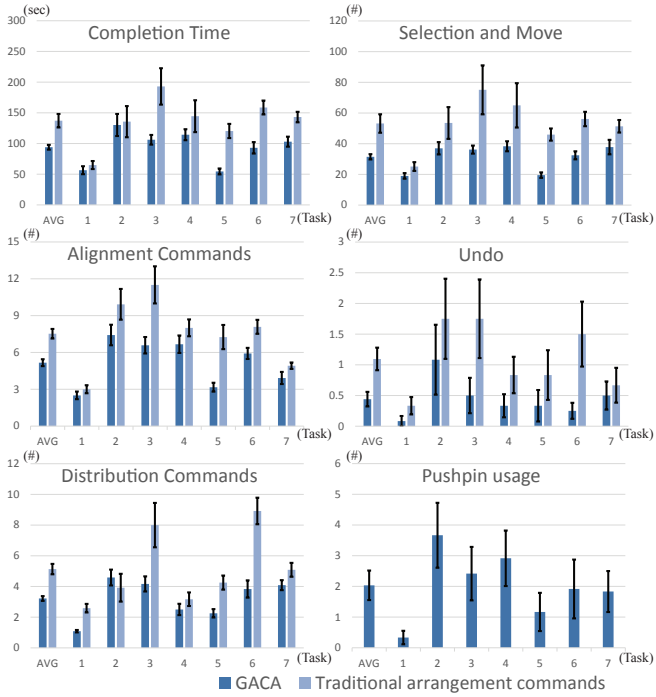


Figure 13. The statistics of Study II. On average GACA needed significantly less completion time, fewer operations of selection and move, fewer undo operations, and fewer arrangement commands per task. The usage of pushpin varied with tasks. Error bars represent the standard error of mean.

arate monitor, with clearly visualized alignment and equal-spacing relationships. The same group of 12 participants from Study I helped with Study II. Each participant was asked to complete each task twice, with our technique and traditional command-based arrangement tools, respectively. The order of the techniques was counter-balanced across participants. In total there were 7 (tasks) $\times$ 2 (techniques) $\times$ 12 (participants) = 168 trials.

Performance measures. During the study, the following information was recorded: the completion time of individual trials, the numbers of general operations (move, undo, grouping, ungrouping), the number of alignment commands, the number of distribution commands. For GACA, two pieces of additional information were recorded: the numbers of pushpin operation and "Arrangement Options" button click.

Results. Figure 13 plots the statistics of the core information. The average completion time per task among the participants was significantly less with GACA (93.9 seconds) than the traditional tools (137.2 seconds). A 2 (TECH) $\times$ 7 (TASK) ANOVA confirmed that there was significant difference in completion time between our technique and traditional tools ($F(1, 154) = 28.025$, $p < 5 \times 10^{-7}$). No significant interaction TECH $\times$ TASK was found ($F(6, 154) = 2.053$, $p = 0.062$), which indicated that the performance improvement of our technique over traditional tools was not affected much by the tasks. However, we noticed that the degree of performance improvement still varied with the tasks. For Task 1, due to its simplicity the performance improvement of our technique was not significant ($p = 0.38$, measured by paired t-test), as expected. For Task 2, the improvement was not significant either ($p = 0.88$). We observed that for this task our group-aware distribution commands might violate the existing alignment relations when the elements distributed diagonally were not equally spaced (see Figure 12). To deal with this problem, the participants either issued extra alignment commands or equally distributed individual rows or columns of elements separately. Some participants also performed pushpin operations to fix the already aligned elements. Since Task 4 contained only few multi-row or multi-column patterns, no significant improvement was found ($p = 0.29$). However, our technique was still slightly better, mainly because the participants took the advantage of the pushpin tool to fix already-aligned elements. For the remaining tasks our technique was significantly faster ($p < 0.05$ for Tasks 3, 5, 6 and 7). This is largely because the participants often issued a single command to multiple groups of elements in these tasks, implying that they were aware of the advantages of our technique and willing to use it when possible.

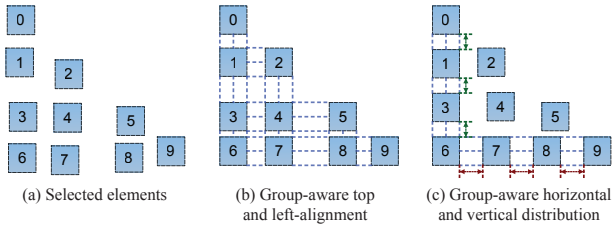


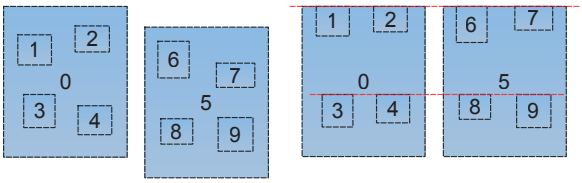
Figure 12. Our group-aware distribution commands might violate the existing alignment relations when the elements distributed diagonally are not equally spaced.

On average, fewer alignment (5.17 vs. 7.52) and distribution (3.21 vs. 5.13) commands per task were issued with GACA. The ANOVA also confirmed that there was statistically significant difference between GACA and traditional tools ($F(1,154) = 33.430$, $p < 4 \times 10^{-8}$ for alignment commands; $F(1,154) = 32.331$, $p < 7 \times 10^{-8}$ for distribution commands). The ANOVA also found significant interactions between techniques and tasks for alignment ($F(6,154) = 2.282$, $p = 0.039$) and distribution ($F(6,154) = 4.811$, $p = 0.0002$) commands, indicating that the degree in which GACA reduced the number of command invocation depended on tasks. It was found that selection and move were the most commonly used general operations. Confirmed by ANOVA, on average our technique needed significantly fewer selection and move operations per task, (31.45 vs. 53.14, $F(1,154) = 32.744$, $p < 6 \times 10^{-8}$), indicating that our technique was less tedious. No significant interaction ($F(6,154) = 1.133$, $p = 0.35$) between techniques and tasks was found for selection and move operations.

The participants understood the pushpin tool quickly and tended to use it when necessary. The usage of pushpin tool depended on the tasks (see Figure 13). For Task 1, most participants did not use the pushpin tool. On average 0.33 pushpin operations were performed for this task. The usage of the pushpin tool was the highest for Task 2: 3.67 pushpin operations on average. This is mainly because of the failure of our group-aware distribution when diagonal elements are not equally spaced (Figure 12). For Task 4, most participants used the traditional tools and ours in a similar way. Our tool was still slightly better, mainly due to the use of the pushpin tool. Although most participants knew how to take advantage of the pushpin tool, some users used pushpins for extra assurances.

During the use of GACA, the "Arrangement Options" button was almost never clicked: only one participant decided to switch to the group-oblivious result only once. This suggests that the results by GACA were almost never worse than those by group-oblivious arrangement. We thus believe that GACA could be used as a better replacement of the traditional command-based arrangement tools. The participants performed undo operations when they were not satisfied with the current results of either group-oblivious or group-aware arrangement tools. On average significantly fewer undo operations were used with GACA (0.44 vs. 1.10, $F(1,154) = 10.210$, $p < 0.002$, confirmed by ANOVA). This reflects that the results by GACA were often expected and desired. However, a significant difference in the number of undo operations was not found for any of the individual tasks.

After completing all the tasks, each participant was asked to rate the two techniques. Ten of them were in favor of GACA, including the three participants who reported themselves as expert users of traditional command-based arrangement tools. However, two participants preferred the traditional tools since they felt more familiar and thus comfortable with them. Most of the participants expressed that at the beginning of Study II they felt a bit unsure of the behavior of GACA. This was not unexpected since after all the logic behind GACA was a bit more complicated than the traditional tools. After completing Study II most of them felt very confident of GACA.



(a) Selected overlapping elements (b) Group-aware top-alignment

Figure 14. Applying GACA to overlapping elements.

DISCUSSION

As described in the "Group-Aware Alignment" section, each pair of elements that would cause overlap between each other after alignment induces a graph partitioning. This rule, however, does not apply to pairs of elements that already intersect with each other before alignment. Figure 14 shows such an example of applying GACA to a set of overlapping elements. A more desired result might be achieved by forming elements e_0 and e_5 as a group and elements e_1, e_2, e_6, and e_7 as another group, since the enclosing relation here implies a two-level organization of elements. This problem is interesting to explore further in the future.

Our grouping strategy is not completely compatible with perceptual grouping, which refers to the human visual ability to extract a meaningful higher-level structure from low-level primitive features [10, 16]. For cases where our approach and perceptual grouping lead to the same grouping results (e.g., Figures 1 (a) and 3 (c)), the user may employ GACA with little thinking. However, when the grouping result by our approach is different from the one in the user's mind (e.g., the input layout in Q3 of Study I), the user might get confused by her perception. Whether the user is still able to predict the result of our approach is highly dependent on her familiarity with GACA. This was confirmed by our user study: although a user could quickly understand how GACA works, it still took time to get used to it. Therefore for novice users, the advantage of our grouping strategy might not be apparent. While for users who are familiar with GACA, it is expected that the performance of our tool would increase. It would be interesting to study the learning process of the users for different grouping strategies.

It is possible to incorporate more perceptual rules for the determination of arrangement groups. However, we have to address two major problems. First, perceptual grouping results can be hierarchical and might not be unique. How to effectively determine the most user-desired result is a challenging problem. Second, perceptual grouping is sensitive to the choice of parameters. Adaptive determination of parameter setting, e.g., to handle the input layouts in Figures 1 (a) and 3 (c), could be intractable. Solving these two problems would further improve the predictability of grouping.

CONCLUSION AND FUTURE WORK

We presented a novel group-aware command-based arrangement technique called GACA. It helps to achieve layouts involving multiple rows and/or columns of well aligned and/or distributed elements. Our user studies confirmed that our technique is intuitive and outperforms the traditional command-based arrangement tools, even for novice users. Still, the user's familiarity with GACA may affect its performance. Therefore, GACA would be particularly useful to professional designers who do lots of layout creating or editing tasks.

Our GACA tool aims to improve the traditional arrangement commands, which arrange the elements defined in a 2D Cartesian coordinate. With coordinate transformation, our algorithm can achieve more complex arrangement tasks, for example, fan-stack arrangements by replacing the Cartesian coordinates with polar coordinates. How to define an intuitive and efficient tool to perform the coordinate transformation would be an interesting problem. Similar to traditional arrangement commands, GACA does not support numeric specification of the arrangement results. It would be useful to provide such a numeric entry after the user issues an arrangement command so that more desired arrangement results can be obtained. To achieve this, it would be necessary to perform a more sophisticated analysis after command invocation. This would be an interesting future direction.

Acknowledgements

We thank the reviewers for their insightful and constructive comments. We also appreciate the user study participants for their time. This work was partially supported by grants from the Research Grants Council of HKSAR, China (Project No. 16209514, 16201315, 113513, 11204014) and JSPS KAKENHI (Grant No. 26240027).

REFERENCES

1. Baudisch, P., Cutrell, E., Hinckley, K., and Eversole, A. Snap-and-go: helping users align objects without the modality of traditional snapping. In *CHI '05* (2005), 301–310.
2. Bier, E. A., and Stone, M. C. Snap-dragging. In *ACM SIGGRAPH Computer Graphics*, vol. 20 (1986), 233–240.
3. Dwyer, T., Marriott, K., and Wybrow, M. Dunnart: A constraint-based network diagram authoring tool. In *Graph Drawing* (2009), 420–431.
4. Frisch, M., Kleinau, S., Langner, R., and Dachselt, R. Grids & guides: multi-touch layout and alignment tools. In *CHI '11* (2011), 1615–1618.
5. Frisch, M., Langner, R., and Dachselt, R. Neat: a set of flexible tools and gestures for layout tasks on interactive displays. In *ITS '11* (2011), 1–10.
6. Galindo, D., and Faure, C. Perceptually-based representation of network diagrams. In *ICDAR '97*, vol. 1 (1997), 352–356.
7. Heo, S., Lee, Y.-K., Yeom, J., and Lee, G. Design of a shape dependent snapping algorithm. In *CHI EA '12* (2012), 2207–2212.
8. Igarashi, T., and Hughes, J. F. A suggestive interface for 3D drawing. In *UIST '01* (2001), 173–181.
9. Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H. Interactive beautification: a technique for rapid geometric design. In *UIST* (1997), 105–114.
10. Nan, L., Sharf, A., Xie, K., Wong, T.-T., Deussen, O., Cohen-Or, D., and Chen, B. Conjoining gestalt rules for abstraction of architectural drawings. *ACM Trans. Graph. 30*, 6 (2011), 185:1–185:10.
11. Nelson, G. Juno, a constraint-based graphics system. In *ACM SIGGRAPH Computer Graphics*, vol. 19 (1985), 235–243.
12. Pavlidis, T., and Van Wyk, C. J. An automatic beautifier for drawings and illustrations. In *ACM SIGGRAPH Computer Graphics*, vol. 19 (1985), 225–234.
13. Raisamo, R., and Raiha, K.-J. A new direct manipulation technique for aligning objects in drawing programs. In *UIST* (1996), 157–164.
14. Reinert, B., Ritschel, T., and Seidel, H.-P. Interactive by-example design of artistic packing layouts. *ACM Trans. Graph. 31*, 6 (2013), Article No. 218.
15. Ryall, K., Marks, J., and Shieber, S. An interactive constraint-based system for drawing graphs. In *UIST '97* (1997), 97–104.
16. Saund, E., Fleet, D., Larner, D., and Mahoney, J. Perceptually-supported image editing of text and graphics. In *UIST '03* (2003), 183–192.
17. Scarr, J., Cockburn, A., Gutwin, C., and Bunt, A. Improving command selection with commandmaps. In *CHI '12* (2012), 257–266.
18. Tollis, I., Eades, P., Di Battista, G., and Tollis, L. *Graph drawing: algorithms for the visualization of graphs*, vol. 1. Prentice Hall New York, 1998.
19. Xu, P., Fu, H., Au, O. K.-C., and Tai, C.-L. Lazy selection: a scribble-based tool for smart shape elements selection. *ACM Transactions on Graphics 31*, 6 (2012), Article No. 142.
20. Xu, P., Fu, H., Igarashi, T., and Tai, C.-L. Global beautification of layouts with interactive ambiguity resolution. In *UIST 14* (2014), 243–252.