ABSTRACT
The Table Lens, focus+context visualization for large data tables, allows users to see 100 times as many data values as a spreadsheet in the same screen space in a manner that enables an extremely immediate form of exploratory data analysis. In the original Table Lens design, data are shown in the context area using graphical representations in a single pixel row. Scaling up the Table Lens technique beyond approximately 500 cases (rows) by 40 variables (columns) requires not showing every value individually and thus raises challenges for preserving the exploratory and navigational ease and power of the original design. We describe two design enhancements for introducing regions of less than a pixel row for each data value and discuss the issues raised by each.

Keywords
Focus+Context, Fisheye, Information visualization, Table Lens

INTRODUCTION
Focus+Context views use some technique for distorting the visual space so that a portion of the view (the focus) has more space to display information while a larger portion of surrounding areas (the context) receive enough space to render relevant properties of the data. The Table Lens [3] uses a two level discrete mapping for assigning space to focus and context in each of two dimensions. Context data elements are represented graphically which allows data rows to be as little as one pixel high. Focus data elements, on the other hand, are represented textually as well as graphically, which allows their values to be read exactly.

By using a two levels of space assignment to data rows, the original Table Lens increased the viewable portion of a table by 100 times compared to a purely spreadsheet-like view. The scale advantage of the Table Lens was achieved while also gaining two other advantages:

• "Ease of Navigation": The entire data set could fit on the screen all at once, thus minimizing the mechanical overheads of navigation.

• "Ease of Exploration": Graphical representation of data allowed the user to quickly spot trends and outliers.

The main limitation is that the original Table Lens design only works for data tables up to the size of data table that allows making each row a pixel high and each column enough pixels wide to allow distinguishing values in the column.

In this paper, we discuss two ways of increasing the reach of Table Lens to arbitrarily large data tables. Both techniques start from the premise that there is significant value in being able to access directly the entire data set. It may be sufficient in many applications to sample the entire data set according to some predicate (i.e. query) such as the top or bottom 500 rows using one of the variables or a random collection of rows. In such cases, the simplicity of using a query front end and the original Table Lens as a view could be the best approach.

To allow access to all data rows, both designs introduce a third level of focus in which multiple data rows are mapped to each pixel row. This choice threatens the navigational and exploratory advantages of the Table Lens. For example, the user is no longer able to see or point at individual data elements in the third focal level. Moreover, any method of selecting or averaging the multiple data elements mapped to a single element may hide singularities or even overall patterns. The impact of these necessary consequences can be minimized by careful interaction design.

The two designs provide the users with different means for controlling what portion of the data table the representation of individuals and thus the original Table Lens' exploratory and navigational advantages are preserved. Each attempts to do this while trying to minimize the complexity of provided controls. The first design nests one or more 2-level focus inside a larger background of the third level and provides controls to the user for manipulating the 2-level areas. The second design allows the user to manipulate directly the space allocated to different regions of the table.

For Color Plate See Page 122

0-8186-8189-6/97 $10.00 © 1997 IEEE
TABLE LENS MAPPING
The original and the two augmented designs of the Table Lens share a common model of mapping a table to the screen. The abstract grid of the table is mapped onto the screen such that some cells are allocated more space than others are, which leads to the effect of multiple focal levels. This mapping function is referred to here as a spatial map, which can be thought of as a function that maps from abstract grid coordinates into screen coordinates. Our framework is similar to the one introduced in [3,4].

To preserve the user’s ability to interpret relationships, the Table Lens preserves the tabular arrangement of the grid elements. If two data elements occupy the same row or column in grid space, they must be aligned with the same vertical or horizontal coordinate in screen space. This constraint is satisfied as long as two independent maps are used along each of the two dimensions.

All interactions with the Table Lens view, such as creation, deletion, and manipulation of focus determine the spatial map. The challenge is to define a set of constraints on the spatial map and a set of controls that allow users to explore the table and efficiently focus on what they want to. For each of the two designs that we have developed to handle larger tables, we describe the constraints on the spatial map and the controls that are provided which manipulate the spatial map.

DESIGN I: NESTING FOCAL LEVELS

![Figure 1 Design I: Three Focal Levels](image)

**Focal Constraints**
In this design, the space allocated to each element is directly dependent on the focal level of that element. Thus along the row dimension, the heights of primary and secondary focus rows are constants, e.g. 21 (text height + extra) and 1 screen units respectively.

A primary focus is always nested inside a region of secondary focus. In this way, the secondary focus functions as a context for the primary one. The visual effect is that of an original Table Lens inside a larger space of many rows mapped to single row (see Figure 1).

The secondary focus is used for coarse navigation, whereas the primary focus is used for finer navigation within the secondary focus.

Given these constraints, the spatial map is calculated by first allocating space to the primary and secondary foci and then evenly distributing the remaining of screen space amongst tertiary foci (see Figure 2). When the user interacts with the view, either by changing the focus boundaries or by creating new focus, the same rules are used to recalculate the spatial map.

![Figure 2 Spatial Map and Focal Levels](image)

**Focal Controls**
Focal controls allow the user to create, delete, resize and move focus. The user creates a 1x1 primary focus by double clicking at the desired location. The primary focus is created nested inside a secondary focus. The initial size of the secondary focus is determined in the following way. Pretend that the space assigned to tertiary focus is such that n tertiary data units must be mapped onto one pixel. In such case, the initial size of the secondary focus is \((n+1)\) data units on each side of the newly created primary focus. The idea is that the denser the tertiary focus is, the less accuracy the user has when pointing at the desired location for the new focus. That is why by creating a secondary focus of size \((n+1)\) on each side of the primary one, the user can make fine adjustment of the primary focus location within nearby region.

In addition to creating new foci, the user can move or resize existing ones. The block arrows on Figure 2 illustrate the effect of these actions on the spatial map. The actual UI controls directly correspond to this kind of
manipulation of the focal level map. For example, to resize the secondary focus, the user can either grab and drag its edges or the control points on the right and bottom sides of the table body (see Figure 1). The primary focus can be resized by dragging its edges or the control points on the left and top sides of the table body. When doing so, the primary focus cannot be resized beyond the bounds of the secondary one and the secondary focus cannot be resized into the primary one. Instead, when one of these two foci is about to cross the bounds of the other focus, the two bounds merge and they are resized simultaneously.

The user can move the primary focus within the secondary one by dragging it with the mouse. If the SHIFT key is pressed while dragging, then the two foci move together with respect to the table body. The effect is that of sliding a lens.

As the secondary focus slides into each unit of tertiary screen units, the user is able to see individual data elements in the context of the trends visible across even the tertiary level. Any outliers are easily spotted within the secondary focus. Thus, this design preserves the ability of Table Lens to represent trends and outliers at the same time despite not having the space to represent all values individually.

Open Questions
This design has a number of issues that still need further design exploration. One such issue is about the default size of the secondary focus when a new primary focus is created. The suggested value of \((n+1)\) data units on each side of the primary focus (see above) is chosen from to support fine navigation. To support rapid exploration, one might choose a larger default value in order for the secondary focus to function adequately as the context for the primary focus. This only begins to hint at the issues of how fast a secondary focus can be used to explore a large data set.

Another open question concerns the behavior of the primary-secondary foci after permuting the data elements. For example, in the bi-focal Table Lens sorting a column causes focal rows to permute with the data, thus splitting the focus into several smaller foci. If the same policy were to apply to a tri-focal Table Lens, the result would split the secondary focus away from the primary focus, which violates the constraint of nesting secondary and primary foci. The current design solves this problem by “turning off” all secondary foci while data and primary foci are being permuted. After the permutation, new secondary foci are re-created around the primary foci by applying the rule for calculating the default size of secondary foci.

DESIGN II: CONTROLLING FOCAL SPANS

In Design II, the space allocated per data element is no longer solely dependent on the focus level of that element. Instead, it also depends on a size parameter specified by the user. Depending on the focal level, the size parameter may affect either a single data element or a group of contiguous elements. In the latter case, we refer to that group of data elements (e.g. a group of rows or columns) as focal span.

Focal Constraints
The user can control the size of individual data elements only in primary focus areas. That is why primary focus elements may vary in size. On the other hand, the user has no control over the size of individual secondary and tertiary focus elements. Therefore, all secondary and tertiary focus elements within the same focal span are assigned equal size (see Figure 4).

Figure 3 Design II: Three Focal Levels

Figure 4 Spatial Map, Focal Spans and Focal Levels
Another focal constraint is a precedence order amongst size requirements specified by the user. The specified size of primary focus elements has highest precedence.
Next is the size of the focal span that has been resized last. All remaining focal spans have the lowest precedence.

Based on these constraints, the spatial map is computed in the following way. First space is allocated for primary focus elements. Next, space is allocated to the most-recently-resized focal span if it exists. The remaining space is divided proportionally among the remaining focal spans. The spatial map is recalculated in this manner after every user interaction with the view.

One property of this design is that the spatial map at any moment depends on the history of user interaction up to that moment. This is unlike Design I, where knowledge of the focal levels is sufficient to reconstruct the spatial map, regardless of previous interactions.

**Focal Controls**

In design I, all operations that could be performed on a focus are done in grid space. For example, resizing a focus means changing its dimensions in grid space such that it begins to encompass more or fewer rows or columns. In design II, focal elements or focal spans can be resized in grid space as well as in screen space.

In Figure 4, the block arrows designate actions similar to those in Design I, for moving and resizing a focus in grid space. In addition to these, the figure also illustrates the actions for resizing a focus in screen space. These are designated as vertical arrows operating on the knees of the spatial map curve. To get a feeling of how the spatial map is calculated, imagine fixing the two ends of the spatial map curve and dragging one of its knees up and down.

The circles in Figure 4 correspond to the control points for resizing individual rows and columns. In the user interface, they are located at the boundaries of the row and column tiles. When the user resizes individual elements (rows or columns), the size that he or she specifies for that element is stored for the element. Even if the element is not at the primary focal level, the size specification is preserved and is later used when the element regains primary focus level.

The diamonds in Figure 4 correspond to the control points for resizing focal spans. In the actual user interface, these are located on the table border. The control points for resizing focal spans are located at the ends of those spans; hence, we refer to them as span breaks. When a primary focus is created, there are four span breaks automatically established at the vertical and horizontal limits of the focus. By dragging these span breaks, the user is able to resize any secondary or tertiary focal spans that surround the primary focus. Apart from the span breaks introduced with the creation of a primary foci, the user can establish additional ones by double clicking on the table border, which results in splitting an existing span into two.

Double clicking on a previously existing span break removes it and the two adjacent spans are merged into one.

By including an extra parameter in the calculation of the spatial map, this design allows for greater flexibility of space allocation. Surprisingly, the added degree of freedom does not lead to a more complex set of UI controls. One reason is that the user is free to introduce or remove UI controls on demand. Consequently, extra controls will be introduced only in areas in which the user wants control.

**Open Questions**

In Design II, the user has great control over the allocation of space to regions of the table. Therefore, this design does not have some of the problems of Design I in having to design in choices about how secondary focal areas are created and maintained across operations (e.g., sorting). On the other hand, it has introduced other issues.

One such problem is the issue of *pixel precision* versus *visual stability*. Pixel precision is a constraint that requires spans to map the same number of pixels to each data element or the same number of data elements to each pixel. If this constraint is violated, artifacts of pixels round off may lead to false impressions.

Enforcing the requirement of pixel precision implies that focal spans cannot be resized in a continuous manner, because they can only assume discrete lengths. Consequently, the user will see noticeable jumps as he or she resizes a focal span. Very similar jumps will be observed when a primary focus is moved across the boundaries of focal spans whose spatial mapping differs from one another.

Thus, there may be a tradeoff between accuracy of interpretability and visual stability. In our implementation, we have allowed the user to turn on or off pixel precision. The intuition is that for browsing and navigation inside the Table Lens, visual stability is more important than accuracy of representation, but occasionally an analysis may require greater control over exact rendering.

In Design I, satisfying the constraint of pixel precision is not a problem, because the spatial mapping within a focal level is the same for all elements with that focal level. Furthermore, Design I does not allow resizing of foci in screen space.

**CONCLUSION**

We have implemented both designs, and our preliminary conclusion is that the second design offers a more versatile and simple mechanism for dealing with very large tables. Though we were initially quite attracted by the "automatic-ness" of space assignment in Design I, we were drawn to Design II by the need to get greater control.
over the actual sizes of visual elements. For example, it was important to be able to control column widths as we started considering real requirements for heavy-duty use. Eventually that lead to giving greater direct control over space allocation. We continue our investigation of Design II in a commercial implementation.

REFERENCES:


<table>
<thead>
<tr>
<th>Name</th>
<th>Value1</th>
<th>Value2</th>
<th>Value3</th>
<th>Value4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack Clark</td>
<td>123</td>
<td>456</td>
<td>789</td>
<td>123</td>
</tr>
<tr>
<td>Mike Smith</td>
<td>234</td>
<td>567</td>
<td>890</td>
<td>234</td>
</tr>
<tr>
<td>Alice W.</td>
<td>345</td>
<td>678</td>
<td>901</td>
<td>345</td>
</tr>
<tr>
<td>Sarah J.</td>
<td>456</td>
<td>789</td>
<td>123</td>
<td>456</td>
</tr>
<tr>
<td>Bob E.</td>
<td>567</td>
<td>890</td>
<td>234</td>
<td>567</td>
</tr>
<tr>
<td>Linda L.</td>
<td>678</td>
<td>901</td>
<td>345</td>
<td>678</td>
</tr>
<tr>
<td>David M.</td>
<td>789</td>
<td>123</td>
<td>456</td>
<td>789</td>
</tr>
<tr>
<td>Emily F.</td>
<td>890</td>
<td>234</td>
<td>567</td>
<td>890</td>
</tr>
<tr>
<td>Frank N.</td>
<td>901</td>
<td>345</td>
<td>678</td>
<td>901</td>
</tr>
<tr>
<td>Tom P.</td>
<td>345</td>
<td>678</td>
<td>901</td>
<td>345</td>
</tr>
</tbody>
</table>