Messy Tabletops: Clearing Up the Occlusion Problem

Euan Freeman and Stephen Brewster
Glasgow Interactive Systems Group
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ UK
e.freeman.1@research.gla.ac.uk,
stephen.brewster@glasgow.ac.uk

CHI 2013 Extended Abstracts, April 27–May 2, 2013, Paris, France.
ACM 978-1-4503-1952-2/13/04.

Abstract
When introducing interactive tabletops into the home and office, lack of space will often mean that these devices play two roles: interactive display and a place for putting things. Clutter on the table surface may occlude information on the display, preventing the user from noticing it or interacting with it. We present a technique for dealing with clutter on tabletops which finds a suitable unoccluded area of the display in which to show content. We discuss the implementation of this technique and some design issues which arose during implementation.

Author Keywords
Tabletop; occlusion; clutter; algorithm

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Algorithms; Design

Introduction
When introduced into environments such as the home or workplace where space is often limited, interactive tabletops will begin to compete for space with existing furniture. In the home, for example, an interactive

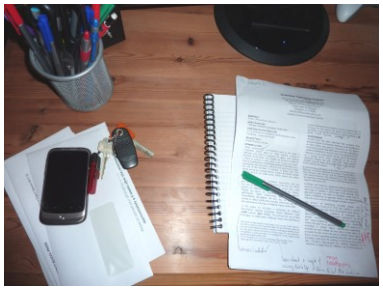

Figure 1. Messy table surfaces - a familiar sight!

tabletop is likely to replace a coffee table in the living room rather than be introduced as an additional piece of furniture. As a consequence the interactive tabletop will also have to fulfill the role of the furniture it replaces. In many cases this means acting as a storage device; a place for books, letters and the many small items which clutter our coffee tables (e.g. Figure 1).

This clutter inevitably constrains interaction and can prevent information from being observed by the user. When the user is actively engaged with the system this could simply be addressed by moving items off the display, freeing up space for interaction. We feel that interactive tabletops, the coffee table of the future, will also be ideal for more passive applications because of their prominent placement within the home. Such applications may display information intermittently throughout the day. Reminders about appointments, new email notifications and information about the home, e.g. warnings that a tap has been left running, could be delivered to a prominent display in the living room. However, users may not know that information is being occluded by items on the table.

The work reported here is part of an ongoing study into the use of tabletops in the home. In this paper we introduce a technique which efficiently finds unoccluded regions of the display. Using this technique, virtual content can be moved to a visible region of the display. We discuss an efficient algorithm for finding visible regions and discuss some design issues we encountered when developing our technique. Future work which fully addresses these issues will allow tabletops to be used effectively in the home, both as a display and place for storage.

Background

O'Hara [5] noted that although an interactive tabletop in a bar provided a dual purpose as both interactive device and normal tabletop, use of the tabletop as a storage device often constrained its use as an interactive display and *vice versa*. Some patrons avoided using the device as a normal table as they did not wish to obstruct the display.

To allow tabletop computers to be effectively used as both interactive display and normal piece of furniture, occlusion management is required. Javed *et al.* [3] proposed several techniques for making the user aware of occluded content. They presented three categories of techniques for identifying occluded objects: awareness-supporting, identification-supporting and access-supporting.

To identify and manage occlusion on interactive surfaces projected from above, Cotting and Gross [1] used patterns projected onto the surface to identify areas which were not suitable for display. Their bubble-based user interface would warp around areas deemed unsuitable for display. This approach also addressed 3D occlusion, where content is not directly covered by an object but is not visible to the user as a protruding object on the table surface blocks the line of sight of the user.

Khalilbeigi *et al.* [4] presented a novel awareness-supporting occlusion management technique which used pressure input to provide a more detailed overview of occluded objects. Applying pressure to the occluding object on the tabletop caused more detailed information to be displayed. Users could drag the item into an un-occluded area of the tabletop to gain access.

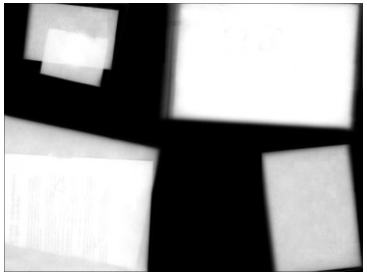

Figure 2. Reflected IR light.

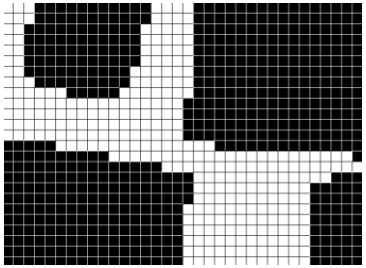

Figure 3. Matrix constructed for the image in Figure 2.

Furumi, Sakamoto and Igarashi [2] created *SnapRail*, an access-supporting widget which moved occluded virtual items onto a rail surrounding the occluding object. Users can then access the virtual objects either by touching them, or touching the rail to move all attached virtual objects.

Another form of occlusion when interacting with touch devices is that caused by the hand of the user. Vogel and Balakrishnan [7] used bubble-like callouts in a similar manner as *Shift* [8] to display important information which was occluded by the hand and forearm of the user when interacting with a tabletop.

Occlusion Management

We chose to implement an access-supporting occlusion management technique as we believed this would be the most effective way of dealing with occlusion when parts of the display are still available. Such techniques make occluded content visible in an unoccluded part of the display. Whilst awareness-supporting techniques could be used, our approach (motivated by Javed *et al.'s* Move method [3]) makes the most of the available display space. Our approach differs from *SnapRail* [2] in that we move occluded content to a clear location on the table rather than around the occluding object. We believe this approach is more appropriate when there is a lot of clutter on the tabletop.

This paper discusses the design and implementation of an occlusion management technique for tabletops which use reflected infrared (IR) light for detecting input. This approach is commonly used in interactive tabletops. IR light is shone towards the rear side of the table surface. Anything in contact with the tabletop (items, hands, etc.) reflects light which is captured by cameras (diffused illumination tabletops) or sensors (e.g. Microsoft's PixelSense) inside the device.

Figure 2 shows an example of an image captured by the cameras inside a diffused illumination tabletop. White areas of the image represent reflected infrared light. Nothing is known about these objects other than their 2D footprint. An access-supporting occlusion management technique must display occluded information in an unobstructed area of the tabletop. There are two parts to this problem: (1) detect items on the tabletop and (2) find a visible area for display.

Tabletop Item Detection

Images taken directly from the cameras inside the tabletop, such as that shown in Figure 2, can be used to detect objects on the table surface using computer vision techniques. To reduce the noise present in the image, a simple threshold can be applied. A blob detection algorithm can then be used to detect objects in the image. Taking the convex hull of each blob gives the footprints of all items atop the table. To avoid classifying hands or fingers as items on the surface, blobs corresponding to touch points should be ignored.

Finding Visible Areas

Having discovered items which are occluding the display, the next step is to find a suitable space for displaying information. As a geometric problem this would be complex and computationally expensive. Instead, we opt for efficiency and choose to approach this as a matrix problem.

Using the 2D footprints of items, a binary matrix is constructed which shows occluded areas of the display. In this matrix, a value of *1* represents an area of the

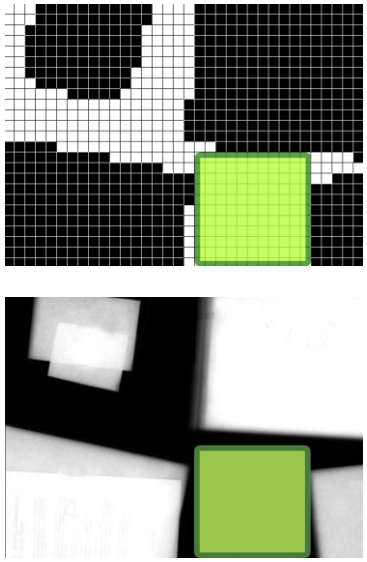

Figure 4. The largest unoccluded area found in Figure 3 and this area shown on the original infrared image.

image which is occluded and *0* represents an area of free space. This matrix can be updated to show the position of unoccluded content, preventing occluded content being moved to the same location. Figure 3 shows the 40x30 matrix constructed for the image shown in Figure 2. In this image, a black cell represents a value of *1* (i.e. an item) and a white cell represents a visible area of the tabletop. The size of the binary matrix is arbitrary; we found that a size of 40x30 offers a balanced trade-off between speed of computation and the accuracy with which item edges are represented.

Using this matrix it is then possible to identify regions of the table surface which are suitable for presenting information. A dynamic programming algorithm was implemented which finds all maximal zero rectangles within the matrix. A zero rectangle is a rectangular area containing only *0* values (i.e. an unoccluded area).

This algorithm, shown in Listing 1, iterates over each cell in the matrix and determines how far to the top, left, and right a border can be extended such that each border marks the first boundary between an occluded and unoccluded cell. For unoccluded cells, these borders represent the edges of the maximum area rectangle, up to the current row, which contains the current cell. Using the border positions *L*, *R* and *T* (for the left, right and top, respectively), the corners of the largest zero rectangle at row *r*, column *c* can be found.

A stack, *s*, is used to determine the position of the left and right borders. Borders will remain on the stack until the next is encountered. Previously computed values for *T* are also reused. This dynamic-programming approach improves the efficiency of this algorithm. Processing each cell has complexity *O(m)* in the worst case, resulting in an overall complexity of $O(m^2n)$ for *m*, the number of columns and *n*, the number of rows. Each of the resulting zero rectangles represent a suitable area of the surface for displaying information. Different heuristics could then be used to select the "best" area for display, considering properties such as size and location. Figure 4 shows the largest area found in the infrared image shown in Figure 2.

```
for r in range(0, n):
    # 1. Compute T
    for c in range(0, m):
        if matrix[c, r] == 1:
            T[c] := r

    # 2. Compute L
    s.clear()
    for c in range(0, m):
        while (|s| > 0 and T[s.peek()] <= T[c]):
            s.pop()

        L[c] := |s| == 0 ? -1 : s.peek()
        s.push(c)

    # 3. Compute R
    s.clear()
    for c in range(0, m).reverse():
        while (|s| > 0 and T[s.peek()] <= T[c]):
            s.pop()

        R[c] := |s| == 0 ? m : s.peek()
        s.push(c)

    # 4. Zero rectangles at each column
    for c in range(0, m):
        topLeft := (L[c] + 1, T[c] + 1)
        bottomRight := (R[c] - 1, r)
```

Listing 1. Algorithm to calculate *L*, *R* and *T*.

Discussion

This access-supporting occlusion management technique allows tabletops to locate a suitable unoccluded area of space for displaying information. The efficiency of this approach allows applications to

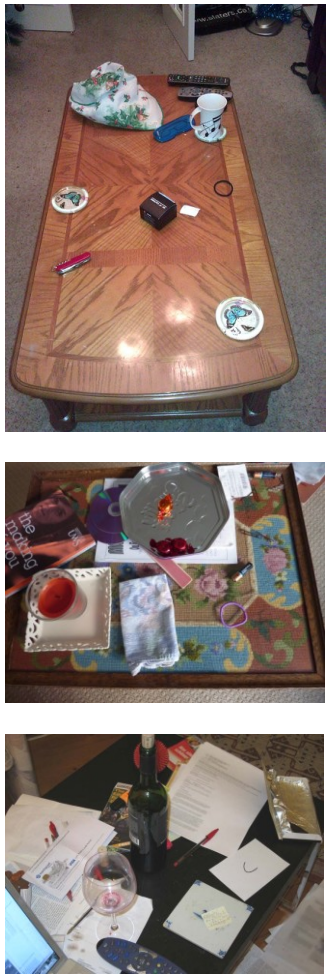

Figure 5. Some of the photos received during the study.

respond dynamically to changes on the table surface and move content as the state of the tabletop changes.

A limitation of our approach to occlusion management is that it searches for unoccluded rectangular areas perpendicular to the table edges. To maximize utilization of the available space, different orientations of content would have to be considered. Our approach only considers the 2D footprint of items on the tabletop. A limitation of 2D occlusion management is that it does not consider occlusion caused by the height of objects. Whilst content on the display may not be directly under an object on the table, that content may not be visible to the user because items on the table protrude into the user's line of sight.

To further explore potential occlusion from clutter in the home we invited people to send us photos of tabletops in their homes, e.g. coffee tables in the living room. We received photos of 11 coffee tables during this study, some of which are shown in Figure 5. The photos showed that most clutter consisted of small items which were unlikely to cause significant 3D occlusion. Books, paper and crockery were the most common tabletop items. The height of these items combined with the low height of the table relative to someone sitting nearby would suggest that 3D occlusion is not likely to be a big problem in this case.

Design Issues

During the design and implementation of our occlusion management technique we encountered some interesting problems. Some of these issues relate to the implementation of our occlusion management approach, others to the general problem of dealing with occlusion on the tabletop.

Positioning content appropriately

We tested the effectiveness of our technique under a variety of conditions, from a small amount of clutter to large amounts of clutter almost entirely covering the display. The largest suitable area for displaying content was chosen. This heuristic was unsuitable with low levels of clutter because it would often result in content moving a lot in response to changes on the table surface. This is a limitation of our algorithm; the largest rectangular space is chosen, when other spaces on the display may also be suitable. The "best" area on the table surface for presenting information may also be the position closest to the user. Future work will investigate better heuristics for positioning content.

Distinguishing between moving and new content

Our technique responds dynamically to objects moving on the table surface. As items are moved on the surface, new visible areas are chosen for content display. In our initial implementation we instantly moved occluded content to a new position on the tabletop. We found that it was not clear what was new content and what was content just being moved on the display. We made the distinction between existing and new content through the use of animation. As virtual items were moved on the display, they followed a smooth, curved path to their destination. This animation showed the user that the virtual item was just being rearranged on the display. New items were faded in gradually such that they appeared to be arriving on the display for the first time.

Responding to touch appropriately

When identifying items atop the table it is important that hands are not incorrectly classified as occluding objects. This allows users to interact with widgets

without them moving in response to a finger upon the table. Occlusion caused by the user's body should still be addressed using techniques such as callout bubbles [8]. We used information provided by the Microsoft Surface SDK to ignore fingers.

Respecting intentional occlusion
Occlusion management is not appropriate for all contexts of tabletop use. Steimle *et al.* [6] suggested that occlusion can be meaningful, e.g. when the user logically groups paper and virtual documents together. Our initial implementation has not considered intentional occlusion. Future work will investigate when occlusion management is necessary and how best to toggle occlusion management, to prevent meaningful piles of virtual and physical objects from being rearranged.

Dealing with very messy tables
In situations where the table surface is heavily cluttered other information delivery modalities may have to be used. Ambient lights placed in the bezel around the display, for example, could be used to make users aware that new content on the display is being completely occluded. Microsoft Surface 1.0 uses lights behind its glass bezel to communicate hardware status. Using ambient lighting to create a glow around the tabletop is similar to the Glow technique suggested by Javed *et al.* [3], who rendered a glowing outline around occluding objects on the display.

Conclusion

In this paper we introduced an access-supporting occlusion management technique for tabletops. This technique detects items atop the table and finds a visible region of the display suitable for showing content. We also discussed issues which arose during implementation. Our future work will address these design issues and investigate interesting applications of our technique.

Acknowledgements

This work was supported by the SFC funded MATCH project (HR04016).

References

[1] Cotting, D. and Gross, M. Interactive Environment-Aware Display Bubbles. *Proc. UIST 2006*, ACM Press (2006), 245-254.

[2] Furumi, G., Sakamoto, D. and Igarashi, T. SnapRail: A Tabletop User Interface Widget for Addressing Occlusion by Physical Objects. *Proc. ITS 2012*, ACM Press (2012), 193-196.

[3] Javed, W., Kim, K., Ghani, S. and Elmqvist, N. Evaluating Physical/Virtual Occlusion Management Techniques for Horizontal Displays. *Proc. Interact 2011*, (2011), 391-408.

[4] Khalilbeigi, M., Schmittat, P., Mühlhaüser, M. and Steimle, J. Occlusion-Aware Interaction Techniques for Tabletop Systems. *Ext. Abstracts CHI 2012*, ACM Press (2012), 2531-2536.

[5] O'Hara, K. Interactivity and Non-Interactivity on Tabletops. *Proc. CHI 2010*, ACM Press (2010), 2611-2614.

[6] Steimle, J., Khalilbeigi, M., Mühlhaüser, M. and Hollan, J.D. Physical and Digital Media Usage Patterns on Interactive Tabletop Surfaces. *Proc. ITS 2010*, ACM Press (2010), 167-176.

[7] Vogel, D. and Balakrishnan, R. Occlusion-aware interfaces. *Proc. CHI 2010*, ACM Press (2010), 263-272

[8] Vogel, D. and Baudisch, P. Shift: A Technique for Operating Pen-Based Interfaces Using Touch. *Proc. CHI 2007*, ACM Press (2007), 657-666.