SurfaceConstellations: A Modular Hardware Platform for Ad-Hoc Reconfigurable Cross-Device Workspaces

Nicolai Marquardt[1], **Frederik Brudy**[1], **Can Liu**[2], **Benedikt Bengler**[2,3], **Christian Holz**[4]
[1]University College London, UK [2]ICRI Cities, UCL [3]IXDS [4]Microsoft Research, USA
n.marquardt@ucl.ac.uk, f.brudy@cs.ucl.ac.uk, c.liu@ucl.ac.uk, ben.bengler@ixds.com, cholz@microsoft.com

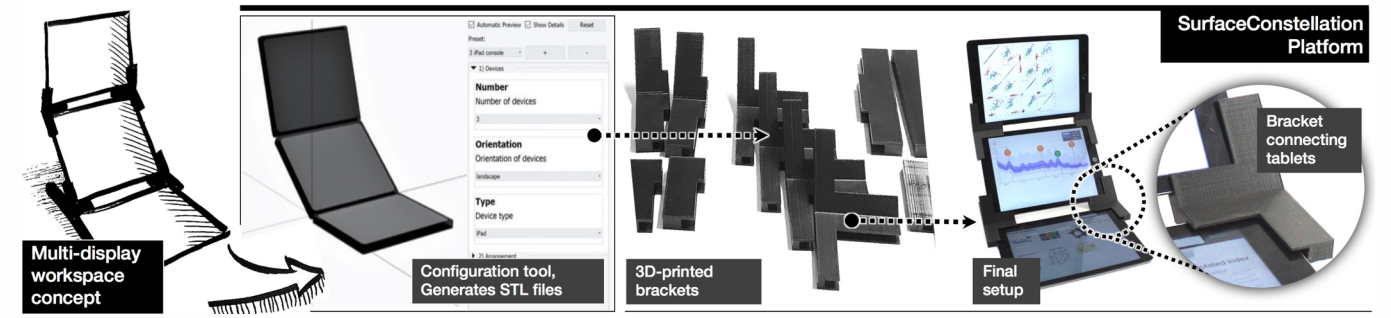

Figure 1. Modular SurfaceConstellations platform enables creation of spatial cross-device workstation setups.

ABSTRACT
We contribute SurfaceConstellations, a modular hardware platform for linking multiple mobile devices to easily create novel cross-device workspace environments. Our platform combines the advantages of multi-monitor workspaces and multi-surface environments with the flexibility and extensibility of more recent cross-device setups. The SurfaceConstellations platform includes a comprehensive library of 3D-printed link modules to connect and arrange tablets into new workspaces, several strategies for designing setups, and a visual configuration tool for automatically generating link modules. We contribute a detailed design space of cross-device workspaces, a technique for capacitive links between tablets for automatic recognition of connected devices, designs of flexible joint connections, detailed explanations of the physical design of 3D printed brackets and support structures, and the design of a web-based tool for creating new SurfaceConstellation setups.

Author Keywords
Cross-device interactions; reconfigurable workspaces; multi surfaces; multi-display environment

ACM Classification Keywords
H.5.2. User Interfaces

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ACM 978-1-4503-5620-6/18/04…$15.00
https://doi.org/10.1145/3173574.3173928

INTRODUCTION
Since early visions such as Vannevar Bush's Memex [4], multi-display setups have been used to effectively support a variety of desktop computing activities: from visual analytics, to financial computing [3], control rooms, business analytics dashboards, or video and audio editing. The main advantage of these setups is not only the larger available screen real estate (i.e. more pixels to display information and interact with), but also the benefit of effectively distributing information across the distinct inter-connected displays. Grudin notes that multi-monitor setups provide "*space with a dedicated purpose, always accessible with a glance*" and "*can facilitate versatility in use*" [13].

This expressive power of multi-display setups has also inspired work in *cross-device interaction*. Cross-device setups allow people to use interfaces that span across several interconnected tablets, phones and other devices. Like multi-monitor desktop setups, these systems provide a larger interaction space (e.g. more content displayed simultaneously, additional input space for gesture input) to interact with applications, whilst enabling one to dynamically add or remove devices from such a device ecology. Most of this work is designed around two primary usage scenarios: mobile, ad-hoc setups for collaborations (e.g. [5,30,37,38,44]); and interactive environments with a variety of mobile and large interactive surfaces (e.g. [47,53]).

Our goal with *SurfaceConstellations* (Figure 1) is to bridge the gap between the power and effectiveness of multi-monitor workstations with the flexibility and ad-hoc configurability of cross-device computing. To implement this vision, we designed a novel modular platform that enables users to easily assemble a large variety of spatial multi-surface arrangements. Our SurfaceConstellations brackets thereby physically connect tablets and phones to create larger dedicated workstation setups. The modularity of our platform enables

a large spectrum of possible multi-surface setups that are easily reconfigurable and can support diverse working styles and applications.

In particular, we contribute:

- The SurfaceConstellations hardware design and a design space taxonomy for the workspace setups it affords;
- Specifications for the physical design of 3D-printed brackets, two designs for flexible joint connections, techniques for including weight-balancing support structures, and configuration tools for setting up new workspaces;
- The capacitive link technique for automatically recognizing connected touch-screen devices;
- Demonstration of the versatility of our platform through four use case applications, implemented using different cross-devices computing frameworks.

To facilitate the adoption of SurfaceConstellations in future applications, all hardware designs, 3D-print STL files, 3D model source files and the software are released as open-hardware and open-software[1].

RELATED WORK

SurfaceConstellations are related to multi-monitor environments, connected-display devices, and more generally cross-device interaction on recent mobile devices.

From Multi-monitor setups towards Multi-Display: Environments with Interactive Walls and Tabletops

Inspired by early visions [4], multi-monitor setups allow the distribution of visual interfaces across two or more screens [19]. Such setups can help to better support interactions with resource-intensive applications such as visual analytics, multi-channel audio editing, or financial computing and trading desks (e.g. [3]).

The same goal – facilitating interaction with large information spaces – is also one of the driving factors in advancing multi-display environments. These setups often include several interactive screens, whiteboards, tabletops, and mobile devices. For example, both iLand [47] and Augmented Surfaces [41] envisioned interaction landscapes spanning across a variety of inter-connected devices. Later, WeSpace [53], ARIS [2] and Dynamo [20] further investigated the design space of multi-display environments. Similar multi-display setups have then been applied to specific use cases, for example, oil and gas exploration [45], visual analytics [10], collaborative sense making [56], and emergency response scenarios [9]. Often, novel interaction techniques had to be designed to manage such environments: for instance, strategies for application relocation [2], perspective correction for cursor manipulation [32], hyper-dragging [41], directing content between devices [14], ad-hoc sharing gestures [30], and transfer with pick-and-drop [39].

With SurfaceConstellations, we are interested in combining the effective workspaces introduced by work in multi-display environments, with strategies from cross-device computing working on ad-hoc reconfigurable device setups.

Rigid and Reconfigurable Joint Surfaces

Related projects have generally tracked grouped devices and screens through rigid or flexible links between them, or through sensing docking events across devices.

Rigid assemblies make for sturdy constructions, which are well-suited for large stationary devices to support task separation and interaction across screens. For example, BendDesk [52] and Curve [54] mimic a traditional PC workstation, but replace the table and keyboard area with an extended, continuous touchscreen. Dell's SmartDesk reduces the form factor and weight to two connected screens whilst maintaining the reconfigurable use [1]. In the mobile space, Codex resembles a notebook, but consists entirely of touchscreens [17]. Codex thoroughly explored the design space of dual-screen mobile devices, spanning interaction techniques, multitasking across applications and screens, and pen and touch input, which has inspired work on dual-display ebook interactions (e.g. [7,8]) and hybrid approaches augmenting physical paper [55].

To omit the necessity of a physical connection between screens, researchers have enabled devices to detect docking events during runtime. Connectables consisted of movable screens for interaction across a larger area after docking [48]. An induction-based tracking mechanism mounted to the displays identified adjacent displays to detect layouts. Hinckley used Synchronous Gestures to detect when users dock tablets together through simultaneous accelerations [16]. Similarly, PhoneTouch requires users to touch their phones to large stationary displays [42], which detect the touch location and then establish a common interaction space. Both Siftables [31] and Droppio [46] further miniaturized this concept to tangible objects and watches, respectively, that sense the docking of independent screen units to produce a larger area. Device configurations can also be obtained from an input gesture spanning both screens (e.g. Stitching across [18] or synchronous swipes [27]).

Dynamic Tracking of Cross-Device Formations

Once device tracking is handled by a sensor external to the involved devices, device constellations afford more flexibility. A common approach for detecting device locations is a general-purpose tracker (e.g. Optitrack [33]) to prototype applications, such as Thaddeus that designed dual-device systems using external tracking to conduct two design studies [57]. Propelled by the availability of commodity depth cameras, many projects have integrated outside-in tracking to dynamically detect device layouts and support cross-device scenarios, such as HuddleLamp [37] and Dippon et al.'s work [12]. Phone as a Pixel acts without a depth camera and instead displays visual codes on all screens, from which an

[1] https://github.com/nicmarquardt/surfaceconstellations

external camera identifies device IDs and locations [44]. As an alternative to optical tracking, GroupTogether's radio beacons provide the 3D positions of moving devices [30].

Conversely, inside-out tracking requires no tracking infrastructure, though typically integrates alternative sensors into mobile devices or provides reduced tracking quality. Pass-them-around achieves this with radio tracking integrated into mobile phones [27], whereas Tracko plays ultrasound signals to establish 3D locations across devices [21]. Tiling Displays [26] and Orienteer [11] both use the cameras of all mobile devices to detect common features and reconstruct device positions.

In SurfaceConstellations, devices can be grouped together in physically stable configurations that may still be flexibly re-configured. Once devices' positions are configured, either manually or through our *CapacitiveLink* modules, no further tracking is needed, which enables spatially aware connections without the need for any tracking infrastructure.

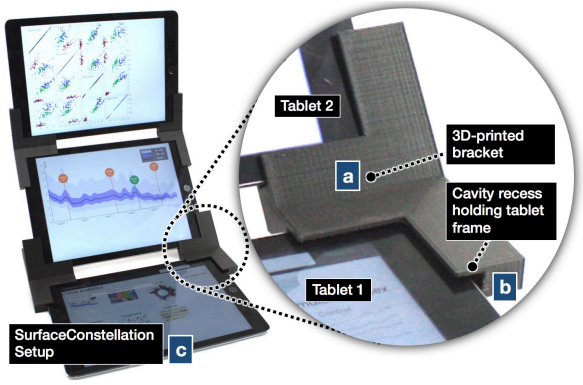

Figure 2. Basic SurfaceConstellation bracket design.

SURFACE CONSTELLATIONS

In this section, we introduce our modular SurfaceConstellations platform and present the design space taxonomy, mapping out possible configurations. We then discuss technical details of the hardware components, in particular: (1) the design of 3D printed modules, (2) flexible-joint brackets, (3) the capacitive link technique, and (4) strategies for adding support extensions.

Concept

The general concept of the SurfaceConstellations platform is to use modular, 3D-printed hardware brackets that physically connect mobile devices (e.g. tablets and phones) to new workstation setups. As described shortly in our design space taxonomy, adding these physical connections enables new spatial arrangements of devices, forming a variety of different workspaces, such as the three-tablet configuration shown in Figure 2. Whilst there are many different types of brackets as we discuss in the 3D design section, they all work the same way: each bracket (Figure 2a) has a cavity recess that holds a part of the frame of at least two devices (2b); after sliding the mobile devices into the bracket's cavities, it physically holds the two (or more) devices in place (2c). The brackets create a permanent, but easily re-configurable physical connection between devices. By combining multiple brackets connecting devices together, one can easily create more advanced setups, affording diverse individual work and collaborative multi-user applications.

Design Space Taxonomy

In our design space taxonomy (extending the taxonomy introduced in Codex [17]) we categorize the principal surface setups supported by the SurfaceConstellations platform (Figure 3). The primary dimension depends on the relative angle between devices (and the second dimension classifies the symmetry of setups):

(a) **Flat:** a flat surface with no angle into the 3D space between tablets. Examples from the design space include a flat book on a table, a larger interaction canvas with 4 or more tablets [37], game board setups with a central shared device, as well as wall/line/fan setups.

(b) **Convex:** the angle between the screen surfaces is larger than 180 degrees. Examples are: the two-sided sign setup [17] and the bridge setup with three tablets. Because convex shapes are outward facing, they better afford collaborative use.

(c) **Concave:** the angle between the screen surfaces is smaller than 180 degrees. Examples for these designs are: the laptop setup, curved design [52,54], dual-screen laptop, and a financial trading-desk inspired setup/wall [3].

(d) **Closed:** the surfaces connect into a 360-degree chained screen. Examples are: a double-sided screen, a cube-like connection of screens, and circular setups.

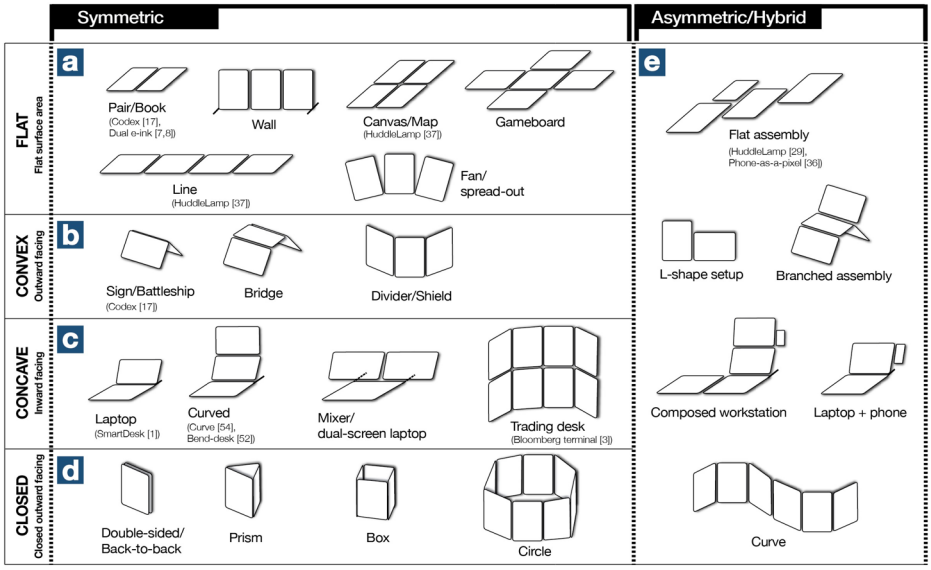

Figure 3. SurfaceConstellation design space taxonomy.

Combining the dimensions/categories above results in hybrid structures (Figure 3e). The dimensions of our design space taxonomy can serve as an inspiration for what can be achieved with SurfaceConstellations, and we will illustrate different use cases across this design space in our application scenarios. We expect that this taxonomy has potential to be extended in the future by adding new designs enabled through our platform.

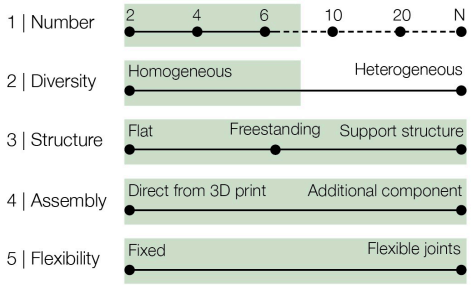

Figure 4. System design parameters (green shades cover the cases supported by our implementation).

Design Parameters

There are further parameters to consider when designing and setting up new SurfaceConstellation setups (Figure 4).

First, the *number of devices*: Whilst technically there is no upper limit for the number of devices that can be connected, we see most practical setups using between 2 and 8 devices.

Second, the *diversity of devices* (e.g. only connecting similar devices vs. connecting different hardware, such as iPads together with Android tablets). Heterogeneous setups require a customized bracket design, as every side of the bracket now needs to be customized for the device size and thickness of the individual tablets. Heterogeneous setups also affect the way software needs to be designed to work across the different OS platforms. In a single setup, we can combine different kinds of devices, such as multi-touch tablets, phones and e-ink displays [55]. Besides tablet-like devices, we can also create SurfaceConstellations connected to desktop monitors or laptops (e.g. adding additional surfaces to the top or the side of the monitor). To increase the stability of setups connected to a laptop, we need to add reinforcement brackets fixing the angle of the hinge between the laptop base and screen (otherwise the weight of additional tablets would cause the laptop screen to fold flat because most laptop hinges do not provide sufficient friction to hold the weight of the full setup in place).

The third parameter considers the *structural setup*: are all devices placed flat on a desk (e.g. the game board design), is the setup freestanding (e.g. the trading desk setup, where the curvature can provide enough support), or are additional support extensions for the brackets needed to balance the weight and make a stable setup (more on this later).

The fourth design parameter accounts for the *complexity of the assembly*: most of the link brackets we designed can be directly 3D-printed (or manufactured through other techniques, like injection moulding). However, some bracket designs might require additional assembly steps, for example to combine different hardware components and materials (examples are our flexible joint brackets).

The fifth parameter is *flexibility*, which we discuss in-depth in a later section: the fixed vs. flexible joint brackets.

Scenario

The following scenario describes an example of how we envision the use of SurfaceConstellations:

> *A financial analyst starts investigating a new data set about pension investments. To better compare and interpret the data, she decides to use multiple tablets and link brackets provided by her employer, linking three tablets into a workstation setup ('Curved') and opening multiple views of the data. As the analysis continues over the day, she finds additional government data she would like to correlate to her earlier data sets, and adds a second 3-screen setup to her desk (linking now six tablets together). In the evening when she has completed the analysis, she takes the setup apart (for another person to use) as she will not need it over the next couple of days when writing the report on her laptop.*

As illustrated in this scenario, SurfaceConstellations are at the sweet spot between ad-hoc, loose multi-tablet setups and fixed multi-monitor desktops: re-configurations are made possible and easy, but we expect these to be only sporadic (e.g. adding tablets to visualize additional data). There are many possible workstation setups in our design space (Figure 3) that – once configured – would not necessarily require any (or only minimal) changes. Importantly, our proposed designs are not intended to replace existing work-station setups (e.g. financial trading desks), but provide more flexible options and new possible workstation designs.

Creating 3D-Printed Modular Brackets

Next, we will explain the details of how to design, build, and manufacture the brackets holding devices for SurfaceConstellation setups.

Whilst there are many possible options for holding a tablet device in place (for example, an all surrounding case, or a mount at the back of the device), we opted for a design that holds the devices in place by clipping a bracket onto each connected corner. Typically, a single bracket connects 2 to 4 devices, directly relating to the number of L-shaped sides of the bracket (Figure 5 top). A bracket can be flat (for example, to connect devices on a desk in Figure 3a) or angular (for most other designs in the design space, such as Figure 3b-e).

Besides the number of connecting sides for the tablet, there are two key hardware parameters. First, the *width of the area covering the tablet case* (Figure 5w); in our designs, this parameter is usually between 8-12mm. Ideally it does not cover any part of the screen; though it needs to be wide enough to hold the tablet in place, which is more difficult with rounded tablet designs. Second, *the thickness of the space holding the*

tablet (Figure 5t). This thickness needs to be slightly smaller than the device's own thickness, so it applies enough pressure onto the case to hold the device in place (with our PLA/ABS prints, decreasing thickness by 0.5mm results in good tension of the bracket onto the device to hold it in place).

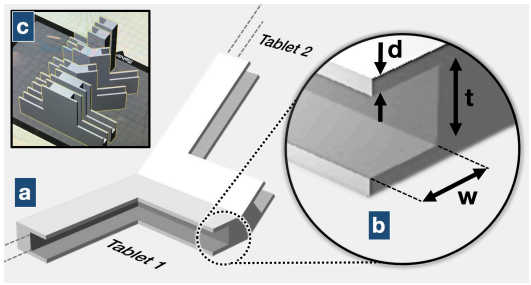

Figure 5. 3D-printed bracket design and design parameters.

Brackets can be manufactured from different materials (e.g. ABS, wood, acrylic) with a variety of techniques (e.g. CNC milling, glued acrylic laser-cut layers). We create most of our brackets using 3D-printing (using Makerbot Replicator 2X, Ultimaker 2+, and Objet Connex 3). We achieved robust brackets with both PLA and ABS prints, with infill of 20% up to 100%. A third parameter of our bracket design to adjust is the *thickness* of the bracket's top and bottom layers (Figure 5d). We have found a thickness of ≥1mm to provide sufficiently robust bracket walls to hold the weight of the tablet.

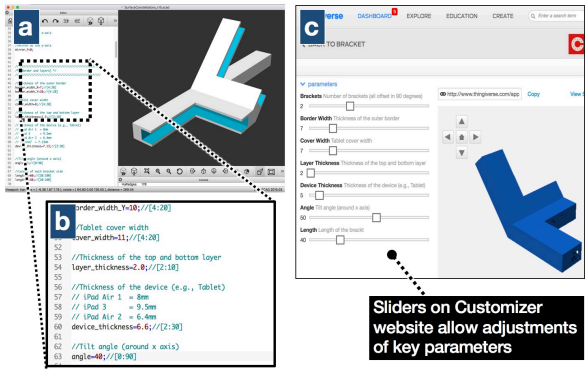

Figure 6. (a) OpenSCAD parametric bracket design and (b) code view, (c) *MakerBot Customizer* view, with sliders on the left side to adjust parameters (e.g. device thickness, angle).

All our bracket designs are modelled in OpenSCAD [24] (Figure 6a), which renders 3D models based on parametric script files and directly creates STL files for 3D printing software. Our modelling files (source code: [29]) include variables for all design parameters for each bracket, such as device thickness, number of L-shaped links, and angle between devices (Figure 6b). Using the OpenSCAD script file syntax also allows us to use these files directly as input for the Thingiverse MakerBot Customizer [28] website (Figure 6c): this website interprets the modelling script files and creates an interface frontend that allows us to use sliders and dropdown menus to adjust any of the parameters for the 3D model. All tagged variables in the file are added as interface elements (e.g. sliders in Figure 6c), and meta comments in the script file can be used to specify minimum/maximum values and other conditions [28].

Using the parametric OpenSCAD language, we designed a basic set of brackets with different parameters: number of links, angles, and device size and thickness (Figure 7). Choosing brackets from these base designs allows one to assemble most of our design space examples (Figure 3): for instance, the curved setup requires 2xE and 2xH, whilst the 6-tablet trading desk setup needs 2xB, 4xE, and 2xG. The STL files of all base designs are included in our SurfaceConstellation library [29] (size for 5 different device sizes: iPad 3, iPad Air, iPad Air 2, and Microsoft Surface 3/4; and presets for 5 angles, see Figure 7).

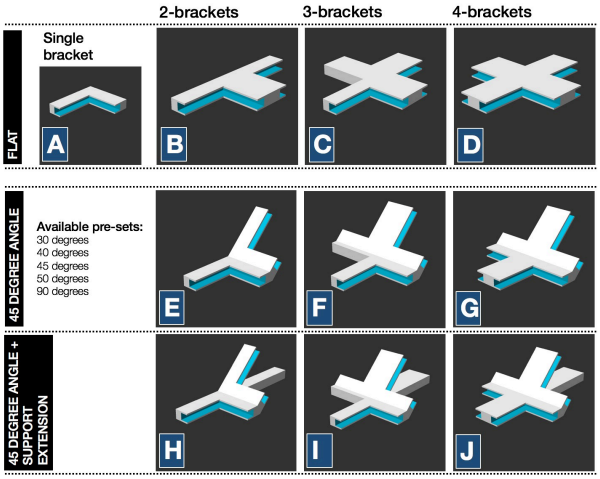

Figure 7. Library of bracket base designs.

Weight-balancing and Support Extensions

Next, we address the need for balancing weight to allow for device setups with free-standing elements. If the weight distribution of a device setup is unbalanced, additional support extensions need to complement a bracket to support the weight, and hold the complete SurfaceConstellation setup in a stable position without tipping over.

Determining the need for support extensions can be done as follows: first, we determine the projection of the centre of mass of the whole setup on the ground. In most cases, we can assume that the weight distribution of each single device is even (i.e. the centre of mass of a single device is in the centre of the volume). We then calculate the centre of mass of the combined setup and project the coordinates back to the ground plane (e.g. onto the surface the setup is standing on). A structure is stable if the projection point is within the base of support, which is the polygon composed of all the touching points between the built structure and the ground (Figure 8a). If the projection point is outside that area (8b), we extend the 3D printed brackets on the bottom with additional parts to increase the base of support area (8c).

In practice, the structure should also be able to stand external forces such as user's touch input, which has different leverage depending on the surface orientation and height. To

achieve this, we extend the minimum length calculated as above with 20% of the height of the entire SurfaceConstellation (this makes sure that the higher the overall setup, the longer the extensions, thus increasing stability). Figure 9 shows example extensions that were added to the brackets. To simplify the use of weigh-balancing extensions, we integrated the calculation of support extension into our GUI tool (explained shortly) for creating new device setups.

For asymmetric and hybrid structures, we need to calculate the projection calculation of centre of mass in two dimensions. For example, if we attach another tablet on the right side of the top tablet of the Bend Desk, the mass of centre will move towards the right side, then the structure could possibly tip over both backwards and to the right.

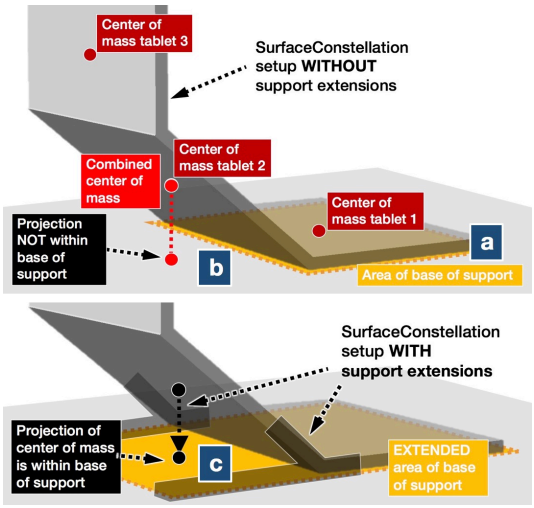

Figure 8. Weight balancing and support extensions.

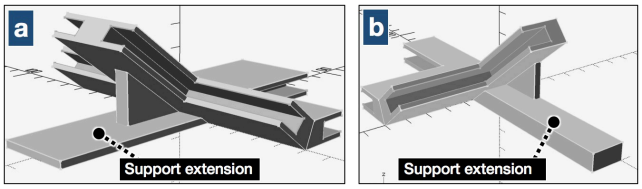

Figure 9. Examples of support extensions for brackets.

Flexible Joint and Variable Thickness Brackets

All bracket designs we have introduced so far are static and fixed objects with no moving parts. We investigated the possible designs of flexible joint brackets, allowing adjustments of the angle between surfaces. For example, when connecting two tablets with such a flexible joint bracket, we can create a setup similar to the book design in Codex [17], and going beyond, adding flexible joints to any part of a SurfaceConstellation. In particular, we explored two techniques for adding such adjustable joints:

Flexible Joints 1: Ratchet-hinged brackets. We developed flexible brackets that feature repositionable ratchet joints (Figure 10a). These can be moved by 90 degrees to both sides (direction switchable) with detent stops each 22.5 degrees. Once a final position is found, the hinge can be locked in place, which quickly sets up workspaces on the fly and allows users to adapt them later. We can also use the flexible brackets for quick physical prototyping: a person can use the flexible joint bracket to find the preferred angle in a console design before 3D-printing a set of solid brackets.

Flexible Joints 2: Dynamic friction-multiplying brackets. In our second design of a flexible joint bracket, each bracket comprises two parts that pivot around an axle (Fig. 10b). When tightening the nut, the resulting friction between the interleaved extensions produces a rigid but reversible lock-in-place mechanism. We incorporate a recess in each bracket, such that the opposite nut locks and disappears inside the assembly without visually standing out.

The gain of flexibility of the ratchet-hinged and friction-bracket designs comes at the cost of increased technical complexity, possibly a slight reduction in stability, and an overall larger space required for the joint part of the bracket. Whilst all of them can be locked in place, they might be less resistant to long-term use compared to rigid brackets.

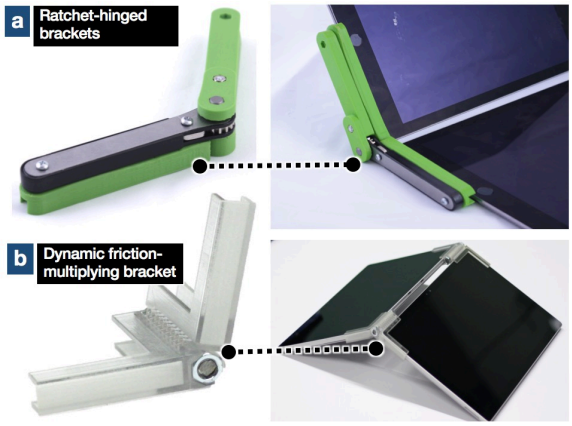

Figure 10. Flexible joint brackets.

Variable Thickness: Our current bracket design is specific to a given device thickness as configured through a parameter. It is possible to modify this design by including padding inside the bracket so devices of varying thickness can be held securely in place (similar to press-fit 3D-printed designs). There are different ways to achieve this, for example by adding padding material inside the bracket (Figure 11a, this would require assembly), or by printing a bending extrusion inside the bracket that flexes in the range of a few millimetres to adapt for varying device thickness (Figure 11 shows the cut profile of a bracket with padding material and flexible bending extrusion).

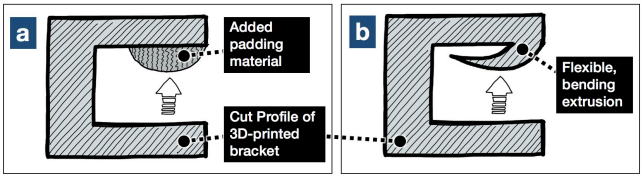

Figure 11. Cut profile of variable thickness brackets.

CapacitiveLink: Recognizing Connected Devices

Once we have assembled a new SurfaceConstellations setup, the software running on the tablets might require information about which tablet is located where (i.e. adjacent devices).

One strategy to configure the location of devices requires *manual setup*, where a user either selects the position of each device in a visual interface, or links devices by performing synchronous gestures [16] (e.g. stitching [18]). Another strategy is to use *camera-based computer vision techniques* (e.g. the outside-in or inside-out tracking approaches we summarized in related work) to determine the relative location between tablets. For example, we could use external RGB cameras (e.g. Phone as a Pixel [44], back-facing cameras [11]), or depth-sensing cameras (e.g. HuddleLamp [37]) to recognize device positions.

We investigated a third option to determine the connection between tablets. Our *CapacitiveLink* approach does not require manual setup or external or internal tracking devices for positioning (e.g. cameras, RF radios), and relies entirely on the hardware design of the brackets. Our approach is that we add a second, conductive material to the 3D-printed brackets, which overlaps with a small section of the device's touchscreen to be recognized as a unique touchpoint. To build this design, we leverage Rekimoto's approach of capacitance tags [40], often used to recognize tangible objects through triggered capacitive touch points on the screen [6,50], or re-directing input to tangible controls (e.g. Clip-on-Gadgets [59]). Unlike PERCs [49] and TUIC [58] tags, our capacitive link brackets do not require additional electronics or batteries.

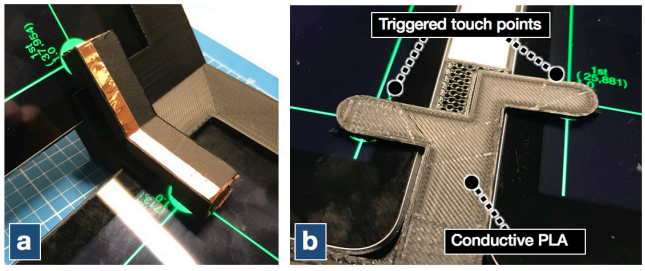

Figure 12. (a) *CapacitiveLink* using conductive copper material as connection between tablets and (b) conductive PLA.

Similar to Extension Sticker [22] and 3D-printed tangibles [23], we use conductive 3D-printed material (*Conductive PLA* [36]) triggering touch events on the screen. Similar to capacitive widgets [50], our brackets are recognized without the need for a person to touch the conductive material. Our 3D-printed bracket design (Figure 12b), which includes an inner core of conductive 3D-print material, overlaps with the touchscreens of connected tablets and triggers a touch contact on each screen (we tested this design first using copper tape connecting touch screens, Figure 12a). We use the 2D position of this contact to uniquely identify the bracket the tablet is connected to. By using a look-up table, we then determine the location of each connected tablet. As seen in the design of the overlapping areas in Figure 12, the bracket triggers two touch points (marked on screen in the centre of the green lines), one on the left and one on the right tablet. Triggering these additional touch points does not interfere with the recognition of other touch screen events, though it does permanently block one of the multi-touch points of the screen (e.g. Apple iPad 2/3 recognizes 11 touchpoints).

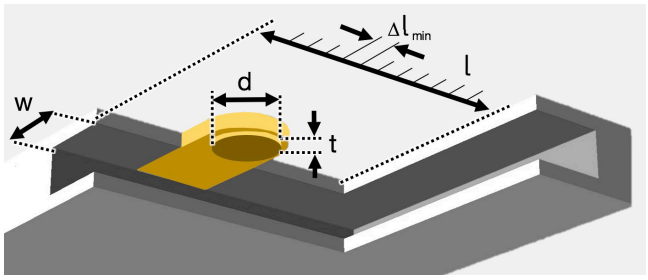

Figure 13. One side of a *CapacitiveLink* bracket design.

To make sure the conductive link reliably triggers touch events when connected to a tablet, we increased the overall volume of the printed conductive material (and use infill >50%). We increased the extrusion of the contact point by 0.1-0.2mm (t in Figure 13) ensuring close connection between conductive material and touch screen surface. To avoid contact points disappearing due to adaptation of the touch screen detection threshold, our CapacitiveLinks are connected to the frame/back of the device [50]. To simulate the size of a finger we set the diameter d of the contact point with the touch screen to 8-10mm. Two parameters are important for the positioning of the ID touchpoint: l is the length of one inner side of the bracket, and identical to the maximum range where we can re-position the contact point (in most of our brackets this length is ~40-60mm); and Δl_{MIN} is the minimum difference between the position of two touchpoints so that they can be uniquely identified. Our tests with the PLA printed conductive brackets (d=10mm, t=0.1, *infill*=50%, for iPad Air 2) showed that due to jitter of the recognized touch contact, Δl_{MIN} would need to be larger than 2mm. This means that with an inner bracket length l=60mm, we would get approximately 30 unique ID positions. With a minimum of two unique IDs necessary per bracket (one for each side), this would allow the use of 15 CapacitiveLink brackets. We can increase this number by extending the length of the bracket, or adding more than one touch point per touchscreen.

One limitation of using CapacitiveLinks is that the conductive material occludes a small part of the screen. The ideal use case for using CapacitiveLinks are setups that are frequently reconfigured and where devices often change their position. The brackets would then help to automatically recognize each of these changes immediately. In many other workstation setups, however, CapacitiveLinks might not be required and a person can do a one-time configuration of the device positions instead.

SETTING UP SURFACECONSTELLATION WORKSPACES: CONFIGURATOR GUI TOOL

So far, we have mentioned four different ways in which a user can set up and use a new SurfaceConstellation workspace: first, a person can directly use one of our *existing complete sets* of link brackets for pre-defined setup of devices (e.g. examples from the design space in Figure 3). Each package includes the designs of all required brackets (STL files for direct 3D-print), rendered for different device types. These packages are included in our SurfaceConstellation library [29]. Second, a user can choose from the collection of existing *base brackets* (Figure 7) and combine them into new setups of connected devices. Third, if none of the existing brackets in the library is appropriate, a user can customize brackets manually with the *MakerBot Customizer* using our source file (specifying different angles or device thickness). Fourth, the most flexible method for creating new brackets is by using the OpenSCAD script directly. However, this does require knowledge of the OpenSCAD scripting language.

Whilst the first two options are the easiest to use, the latter two allow the most flexible customizations (many parameters that can be changed), but are also more complex to use. To bridge this gap between easy-to-use and powerful options, we designed a web-based GUI tool (Figure 14) that allows the configuration of entire workspaces via a parameter menu. Users can choose the number (14a) and type (14b) of devices they want to use, and then define orientation and angles between them (14c). The resulting workspace is visualized in real-time as an interactive 3D model (14d). The tool also provides a selection of typical presets to be used as-is (e.g. 3 tablet console) or as a starting point for new designs by adjusting the parameter set (14e). Each design can be saved as a new preset. Furthermore, the tool automatically calculates the weight distribution and centre of mass of the constellation setup, and adds any necessary support structure extensions to the brackets (14f). Finally, the tool renders all the STL models for any required brackets (14g), and provides a single link to a ZIP file including all files (14h). To further simplify the specification of the actual angle between tablets, we added an additional configuration-by-demonstration feature: one (or multiple) devices can stream their angle and orientation (measured by the internal IMU) to the web-based GUI tool, which then automatically uses this current angle to modify the setting in the web interface.

Figure 14. GUI tool for creating customized workspaces.

DEVELOPING SOFTWARE AND APPLICATIONS

Once a SurfaceConstellations hardware setup has been created and the tablets are physically connected, the next question we need to address is how to use existing software with the setup, or how to develop new applications. Because SurfaceConstellation setups are fundamentally similar to cross-device applications, it is possible to leverage existing toolkits that facilitate the development of multi-surface applications, such as *Webstrates* [25] for dynamically shared media webpages; *XDBrowser* [34], which allows adapting websites for cross-device use; *XDSession* for testing [35]; or *Connichiwa* [43] for local hosted, ad-hoc cross-device applications. We decided to demonstrate the SurfaceConstellation platform's versatility with four example applications, built with different frameworks and tools, whilst at the same time illustrating diverse setups across the design space taxonomy. Before we go through these use cases, we describe four development strategies for using software with or programming software for SurfaceConstellation setups.

Software Connectivity and Interaction

Applications running on touch-screen devices that are connected in a SurfaceConstellation workstation can be designed in four different ways (summarized in Figure 15):

METHOD 1 | No connection: In this configuration, existing applications can be used side-by-side without any direct communication between them (Figure 15a). Examples for this scenario are an email client on one device and a calendar application on the other, or a word processor next to a dictionary.

METHOD 2 | Indirect connection: The software running on each of the devices are communicating indirectly with each

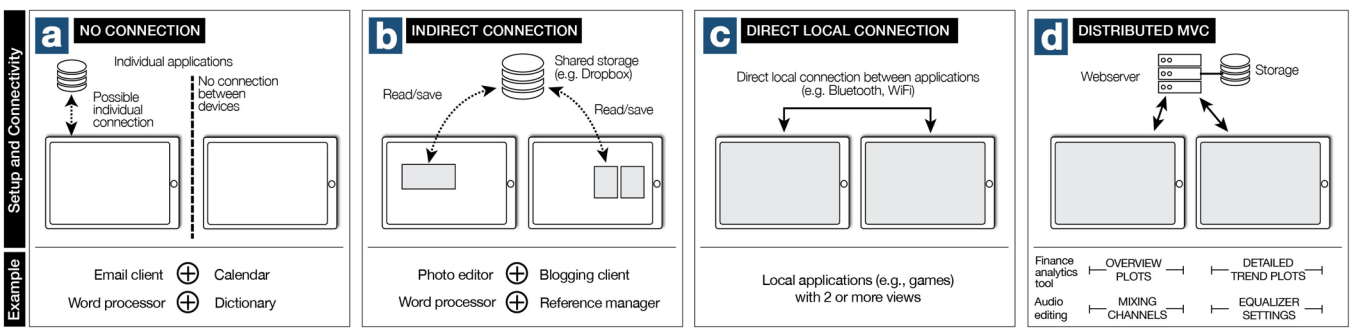

Figure 15. Design characteristics for software running on SurfaceConstellation setups.

other, e.g. through explicit read and write operations on a backend server or shared storage (Figure 15b). An example of this method is using a photo editing software on one device and saving the results to a cloud storage where a blogging platform can access and use the edited photos.

METHOD 3 | Direct local connection: We can leverage direct, local cross-device communication between devices (e.g. with Wi-Fi, Bluetooth, Figure 15c). Direct connections allow for network independence and low communication latency [43]. Examples for such software setups are multiplayer games with different views for each player on the connected devices.

METHOD 4 | Distributed MVC: Lastly, the devices can display views of distributed interface as part of a distributed model-view-controller (dMVC) architecture. The view is individual to each device and a backend server controls the application logic (Figure 15d). An example of this category is a web-based visual analytics tool.

These four methods differ in the degree of how closely coupled the software connection is implemented. Whilst for some use cases it is feasible to set up SurfaceConstellation with multiple devices that have no direct software-side connectivity (Method 1), adding connectivity can allow more fluid and seamless interactions with the cross-device applications (Methods 2-4).

Available Information about Setup and Devices

When designing applications for SurfaceConstellation setups, it can be useful to address the following 4 parameters:

(i) *presence, of devices, number of devices, and the identification of devices*. Most cross-device development frameworks [25,35,43] incorporate a device registration and discovery mechanism.

(ii) *device capabilities*, in particular the resolution of the displays (e.g. pixels height, width, ppi). This information can be shared through the network connection between devices by using a development framework or web-based connections (for example, WebSockets).

(iii) *orientation of devices in space*, e.g. are the devices laid out flat on a table, or are they positioned standing up vertically? Integrated IMU sensors can provide this information automatically.

(iv) *relative position of devices in a SurfaceConstellation setup*. This information can be established either via manual configuration, sensor readings, or through the CapacitiveLink brackets as described above.

Four Example Applications Across the SurfaceConstellations Design Space

To best illustrate the flexibility and expressiveness of the SurfaceConstellations setups across our design space (Figure 3), we describe four use case applications (three custom-built and one commercial software). Our applications also demonstrate how to use existing cross-device development frameworks (in particular, Connichiwa [43] and Webstrates [25]) when developing SurfaceConstellation applications (overview of the four use cases in Table 1).

Application	Design space	Implementation
1. Audio-channel mixing	'Dual-screen laptop'	*Commercial application*
2. Board game	'Bridge'	*WebSockets + nodeJS*
3. Financial Computing	'Trading desk'	*Connichiwa* [43]
4. Visual Analytics	'Bend-desk'	*Webstrates* [25]

Table 1. Overview of use-case applications.

APPLICATION 1: Audio-Track Mixing

Multi-touch tablets are increasingly used to control professional live digital audio mixers (e.g. to control level faders, gain and tone controls). Providing a flexible spatial arrangement for these tablet devices is an ideal use case for SurfaceConstellation setups. We designed a mixing table hardware setup supporting three control tablets (Figure 16). Using the commercial Soundcraft *Ui24* mixing system [15] we can control 24 audio channels and settings. Each of the tablets can provide access to a different subpage of the mixer's software control interface (hosted on a server in the rack unit).

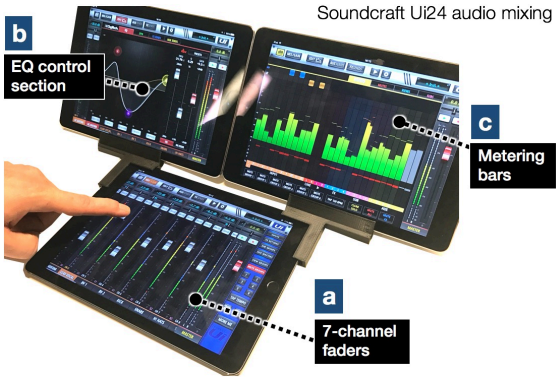

Figure 16. Audio-track mixing setup: (a) control 7-channel level fader bank, (b) equalizer, and (c) metering.

APPLICATION 2: Bridge Setup for Two-player Board Games

This example implements a board game (like the Scrabble™ word game) using the 'bridge' setup (Figure 17a). The shared tablet in the centre shows the playing field, whilst the user-facing devices show a private view of each user's letter rack. On their turn, a user selects a combination of letters, which is then shown in the central shared playing field. From there, the person can drag the letters to position them on the playing board. This example is implemented using *WebSocket* connections between the devices and a central *nodeJS* server managing the shared game state.

APPLICATION 3: Trading Desk for Financial Computing

Trading desks often consist of multiple screens and many different views of related data. As an example, we built a financial trading workspace which consists of stock trading widgets (Figure 17b). We developed this application with the Connichiwa [43] framework to run a local server instance on one of the tablets. This tablet functions as a master device to which clients can connect, to select which financial data widget should be displayed on each device.

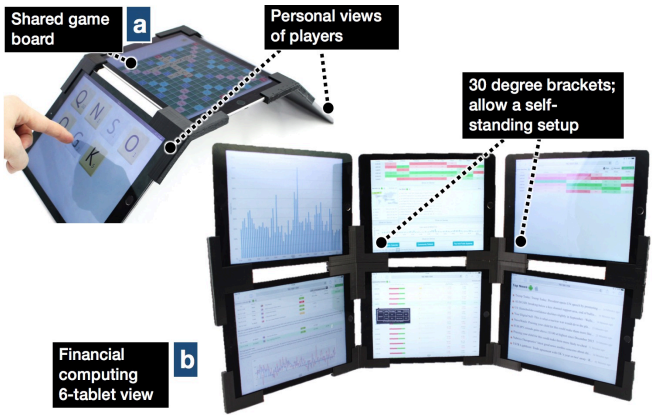

Figure 17. (a) Multi-player board games; (b) trading-desk.

APPLICATION 4: Hybrid Setup for Visual Analytics

Visual analytics often requires analysts to distribute information items across different screens, allowing them to compare different views of the same data. We built a distributed web application (based on the WebStrates platform [25]) in which analysts are presented with an overview of available visualizations for a dataset (Figure 18a). By using additional brackets, analysts can dynamically add devices to the setup. Figure 18b shows how a phone is added to the workstation setup, displaying a control selection interface for the visualization presented on a tablet next to it (18c).

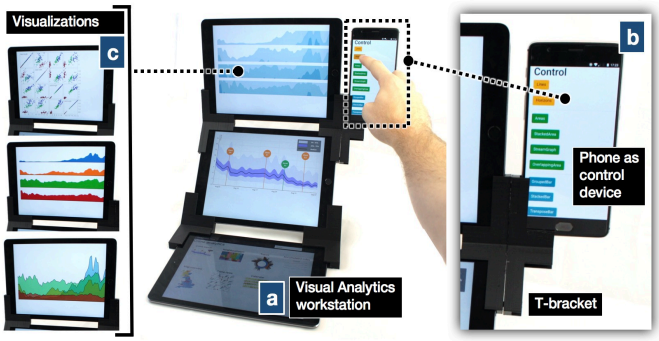

Figure 18. Setup for visual analytics application.

DISCUSSION

Reflecting on SurfaceConstellations designs, we briefly discuss characteristics of semi-fixed setups, roles of devices, and possible interaction techniques.

Semi-Fixed vs. Mobile Cross-Device Setups

SurfaceConstellations strike a balance between the traditional desktop screen arrangements found in expert environments as well as ad-hoc mobile device groupings. The former is a specialized arrangement of screens, typically to support a set of specific tasks (e.g. air traffic control, finance applications, etc.). Ad-hoc mobile scenarios, in contrast, typically involve fluid arrangements of devices, quickly positioned usually by hand to support a certain task (e.g. exchanging or comparing information) and often span multiple users. Importantly, such scenarios involve *personal* mobile devices – the devices we always carry with us.

SurfaceConstellations offer a framework for semi-rigid arrangements of mobile devices. Whilst such arrangements are more rigid than the fluid gathering of mobile devices, they can be made *persistent* if desired and redeem many of the benefits of traditional multi-monitor setups. At the same time, they can be *reconfigured* easily by rearranging device positions and orientations through different brackets.

Roles of Devices

Closely related to the setups of devices are the roles different devices may take. From a technical perspective, all connected tablets in a constellation might be of the same kind (for instance, using only Microsoft Surface tablets), but they might take very different roles depending on their placement, the application, and the task at hand. For example, some devices might become primary interaction devices (for example, a touch keyboard for rich input) whilst others become secondary devices (e.g. a drag-and-drop clipboard, or a touch-enabled thumbnail overview) or even passive, viewing-only devices (e.g. a large zoomed-in view of content). Further investigating the roles (and possibly fluid changes of roles) of devices remains part of future work.

Interaction Techniques

For the scope of this paper we focused on the hardware designs of the platform. For any SurfaceConstellation workstation, there is a design opportunity to tailor interaction techniques to best support interaction in each particular setup and application. For example, we can develop techniques to better support cross-device interaction such as dragging objects across surfaces [18]. Indirect manipulations could be used when the configured workstation setup has devices or areas that are inconvenient to reach. Existing *overview+detail* techniques could be integrated, such as having one surface showing a data map overview and multiple *DragMag* views to show details of particular regions [51].

Cross-Device Applications for the Masses

The SurfaceConstellation platform enables anyone with access to a 3D printer and multiple tablets/phones to design and construct one's own multi-surface workspace. Similar to the research field of cross-device interactions, we anticipate that an increasing number of available touch-screen devices will soon allow people to *use their devices in concert* – and that SurfaceConstellations arrangements can help to facilitate people's interaction with this larger number of devices. Importantly, we made the SurfaceConstellations designs available as open hardware and open software [29]. With our designs, taxonomy and examples, we aim to inspire users' creativity to build, use and re-appropriate such environments for various scenarios of use, which we hope takes us one step closer to making cross-device applications available to the masses.

ACKNOWLEDGEMENTS

We thank Soundcraft, HARMAN and Scott Wood for their support of this research project. Frederik Brudy has been supported by Microsoft Research through the PhD Scholarship Programme. Part of this research funded by ICRI Cities.

REFERENCES

1. Rocco Ancona. 2014. The Future of Workplace Productivity: Smart Desks and UltraSharp Monitors. Retrieved September 7, 2017 from https://blog.dell.com/en-us/smart-desks-and-ultrasharp-monitors-are-the-future-of-workplace-productivity/
2. Jacob T. Biehl and Brian P. Bailey. 2004. ARIS: An Interface for Application Relocation in an Interactive Space. In *Proceedings of Graphics Interface 2004* (GI '04), 107–116.
3. Bloomberg L.P. Bloomberg Professional Services: The Terminal | Hardware. Retrieved from https://www.bloomberg.com/professional/solution/bloomberg-terminal
4. Vannevar Bush. 1945. As We May Think. *The Atlantic*. Retrieved September 4, 2017 from https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/
5. Jessica R. Cauchard, Markus Löchtefeld, Pourang Irani, Johannes Schoening, Antonio Krüger, Mike Fraser, and Sriram Subramanian. 2011. Visual Separation in Mobile Multi-display Environments. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (UIST '11), 451–460. https://doi.org/10.1145/2047196.2047256
6. Liwei Chan, Stefanie Müller, Anne Roudaut, and Patrick Baudisch. 2012. CapStones and ZebraWidgets: Sensing Stacks of Building Blocks, Dials and Sliders on Capacitive Touch Screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2189–2192. https://doi.org/10.1145/2207676.2208371
7. Nicholas Chen, Francois Guimbretiere, Morgan Dixon, Cassandra Lewis, and Maneesh Agrawala. 2008. Navigation Techniques for Dual-display e-Book Readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08), 1779–1788. https://doi.org/10.1145/1357054.1357331
8. Nicholas Chen, François Guimbretière, and Abigail Sellen. 2013. Graduate Student Use of a Multi-slate Reading System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 1799–1808. https://doi.org/10.1145/2470654.2466237
9. Apoorve Chokshi, Teddy Seyed, Francisco Marinho Rodrigues, and Frank Maurer. 2014. ePlan Multi-Surface: A Multi-Surface Environment for Emergency Response Planning Exercises. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14), 219–228. https://doi.org/10.1145/2669485.2669520
10. Haeyong Chung, Chris North, Jessica Zeitz Self, Sharon Chu, and Francis Quek. 2014. VisPorter: Facilitating Information Sharing for Collaborative Sensemaking on Multiple Displays. *Personal Ubiquitous Comput.* 18, 5: 1169–1186. https://doi.org/10.1007/s00779-013-0727-2
11. David Dearman, Richard Guy, and Khai Truong. 2012. Determining the Orientation of Proximate Mobile Devices Using Their Back Facing Camera. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2231–2234. https://doi.org/10.1145/2207676.2208377
12. Andreas Dippon, Norbert Wiedermann, and Gudrun Klinker. 2012. Seamless Integration of Mobile Devices into Interactive Surface Environments. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces* (ITS '12), 331–334. https://doi.org/10.1145/2396636.2396693
13. Jonathan Grudin. 2001. Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '01), 458–465. https://doi.org/10.1145/365024.365312
14. Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: Enabling and Understanding Cross-device Interaction. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14), 2773–2782. https://doi.org/10.1145/2556288.2557170
15. Harman. Soundcraft Ui24R. *Soundcraft - Professional Audio Mixers*. Retrieved September 15, 2017 from http://www.soundcraft.com/products/ui24r
16. Ken Hinckley. 2003. Synchronous Gestures for Multiple Persons and Computers. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (UIST '03), 149–158. https://doi.org/10.1145/964696.964713
17. Ken Hinckley, Morgan Dixon, Raman Sarin, Francois Guimbretiere, and Ravin Balakrishnan. 2009. Codex: A Dual Screen Tablet Computer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1933–1942. https://doi.org/10.1145/1518701.1518996
18. Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. 2004. Stitching: Pen Gestures That Span Multiple Displays. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (AVI '04), 23–31. https://doi.org/10.1145/989863.989866
19. Dugald Ralph Hutchings, John Stasko, and Mary Czerwinski. 2005. Distributed Display Environments. *interactions* 12, 6: 50–53. https://doi.org/10.1145/1096554.1096592

20. Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers, and Mia Underwood. 2003. Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (UIST '03), 159–168. https://doi.org/10.1145/964696.964714
21. Haojian Jin, Christian Holz, and Kasper Hornbaek. 2015. Tracko: Ad-hoc Mobile 3D Tracking Using Bluetooth Low Energy and Inaudible Signals for Cross-Device Interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15), 147–156. https://doi.org/10.1145/2807442.2807475
22. Kunihiro Kato and Homei Miyashita. 2015. ExtensionSticker: A Proposal for a Striped Pattern Sticker to Extend Touch Interfaces and Its Assessment. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), 1851–1854. https://doi.org/10.1145/2702123.2702500
23. Kunihiro Kato and Homei Miyashita. 2016. 3D Printed Physical Interfaces That Can Extend Touch Devices. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (UIST '16 Adjunct), 47–49. https://doi.org/10.1145/2984751.2985700
24. Marius Kintel. OpenSCAD. Software for creating solid 3D CAD objects (under GPL v2, https://github.com/openscad/openscad/). Retrieved June 2, 2017 from http://openscad.org
25. Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*, 280–290. https://doi.org/10.1145/2807442.2807446
26. Ming Li and Leif Kobbelt. 2012. Dynamic Tiling Display: Building an Interactive Display Surface Using Multiple Mobile Devices. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia* (MUM '12), 24:1–24:4. https://doi.org/10.1145/2406367.2406397
27. Andrés Lucero, Jussi Holopainen, and Tero Jokela. 2011. Pass-them-around: Collaborative Use of Mobile Phones for Photo Sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11), 1787–1796. https://doi.org/10.1145/1978942.1979201
28. MakerBot Industries LLC. Developer Documentation MakerBot Customizer. Retrieved August 22, 2017 from http://customizer.makerbot.com/docs
29. Nicolai Marquardt. SurfaceConstellations Github Repository. Retrieved January 1, 2018 from https://github.com/nicmarquardt/surfaceconstellations
30. Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. Cross-device Interaction via Micro-mobility and F-formations. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (UIST '12), 13–22. https://doi.org/10.1145/2380116.2380121
31. David Merrill, Jeevan Kalanithi, and Pattie Maes. 2007. Siftables: towards sensor network user interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (TEI '07), 75–78. http://dx.doi.org/10.1145/1226969.1226984
32. Miguel A. Nacenta, Samer Sallam, Bernard Champoux, Sriram Subramanian, and Carl Gutwin. 2006. Perspective Cursor: Perspective-based Interaction for Multi-display Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06), 289–298. https://doi.org/10.1145/1124772.1124817
33. NaturalPoint. 2013. Motion Capture Systems by OptiTrack. *http://www.optitrack.com/hardware/*. Retrieved from http://www.optitrack.com/hardware/
34. Michael Nebeling. 2017. XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17), 4574–4584. https://doi.org/10.1145/3025453.3025547
35. Michael Nebeling, Maria Husmann, Christoph Zimmerli, Giulio Valente, and Moira C. Norrie. 2015. XDSession: Integrated Development and Testing of Cross-device Applications. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (EICS '15), 22–27. https://doi.org/10.1145/2774225.2775075
36. ProtoPlant. Conductive PLA. Retrieved September 10, 2017 from https://www.proto-pasta.com/pages/conductive-pla
37. Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (ITS '14), 45–54. https://doi.org/10.1145/2669485.2669500
38. Roman Rädle, Hans-Christian Jetter, Mario Schreiner, Zhihao Lu, Harald Reiterer, and Yvonne Rogers. 2015. Spatially-aware or Spatially-agnostic?: Elicitation and Evaluation of User-Defined Cross-Device Interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), 3913–3922. https://doi.org/10.1145/2702123.2702287

39. Jun Rekimoto. 1997. Pick-and-drop: A Direct Manipulation Technique For Multiple Computer Environments. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology* (UIST '97), 31–39. http://dx.doi.org/10.1145/263407.263505

40. Jun Rekimoto. 2002. SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02), 113–120. https://doi.org/10.1145/503376.503397

41. Jun Rekimoto and Masanori Saitoh. 1999. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '99), 378–385. https://doi.org/10.1145/302979.303113

42. Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM Symposium on User Interface Software and Technology* (UIST '10), 13–16. https://doi.org/10.1145/1866029.1866034

43. Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. 2015. Connichiwa: A Framework for Cross-Device Web Applications. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '15), 2163–2168. https://doi.org/10.1145/2702613.2732909

44. Julia Schwarz, David Klionsky, Chris Harrison, Paul Dietz, and Andrew Wilson. 2012. Phone As a Pixel: Enabling Ad-hoc, Large-scale Displays Using Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2235–2238. https://doi.org/10.1145/2207676.2208378

45. Teddy Seyed, Mario Costa Sousa, Frank Maurer, and Anthony Tang. 2013. SkyHunter: A Multi-surface Environment for Supporting Oil and Gas Exploration. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces* (ITS '13), 15–22. https://doi.org/10.1145/2512349.2512798

46. Teddy Seyed, Xing-Dong Yang, and Daniel Vogel. 2016. Doppio: A Reconfigurable Dual-Face Smartwatch for Tangible Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16), 4675–4686. https://doi.org/10.1145/2858036.2858256

47. Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. 1999. i-LAND: An Interactive Landscape for Creativity and Innovation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '99), 120–127. https://doi.org/10.1145/302979.303010

48. Peter Tandler, Thorsten Prante, Christian Müller-Tomfelde, Norbert Streitz, and Ralf Steinmetz. 2001. Connectables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (UIST '01), 11–20. https://doi.org/10.1145/502348.502351

49. Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergaard, and Jan Borchers. 2015. PERCs: Persistently Trackable Tangibles on Capacitive Multi-Touch Displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15), 351–356. https://doi.org/10.1145/2807442.2807466

50. Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergaard, and Jan Borchers. 2013. PUCs: Detecting Transparent, Passive Untouched Capacitive Widgets on Unmodified Multi-touch Displays. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces* (ITS '13), 101–104. https://doi.org/10.1145/2512349.2512791

51. Colin Ware and Marlon Lewis. 1995. The DragMag Image Magnifier. In *Conference Companion on Human Factors in Computing Systems* (CHI '95), 407–408. https://doi.org/10.1145/223355.223749

52. Malte Weiss, Simon Voelker, Christine Sutter, and Jan Borchers. 2010. BendDesk: Dragging Across the Curve. In *ACM International Conference on Interactive Tabletops and Surfaces* (ITS '10), 1–10. https://doi.org/10.1145/1936652.1936654

53. Daniel Wigdor, Hao Jiang, Clifton Forlines, Michelle Borkin, and Chia Shen. 2009. WeSpace: The Design Development and Deployment of a Walk-up and Share Multi-surface Visual Collaboration System. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1237–1246. https://doi.org/10.1145/1518701.1518886

54. Raphael Wimmer, Fabian Hennecke, Florian Schulz, Sebastian Boring, Andreas Butz, and Heinrich Hußmann. 2010. Curve: Revisiting the Digital Desk. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* (NordiCHI '10), 561–570. https://doi.org/10.1145/1868914.1868977

55. Christian Winkler, Julian Seifert, Christian Reinartz, Pascal Krahmer, and Enrico Rukzio. 2013. Penbook: Bringing Pen+Paper Interaction to a Tablet Device to Facilitate Paper-based Workflows in the Hospital Domain. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*

(ITS ’13), 283–286. https://doi.org/10.1145/2512349.2512797

56. Pawel Wozniak, Nitesh Goyal, Przemyslaw Kucharski, Lars Lischke, Sven Mayer, and Morten Fjeld. 2016. RAMPARTS: Supporting Sensemaking with Spatially-Aware Mobile Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI ’16), 2447–2460. https://doi.org/10.1145/2858036.2858491

57. Pawel Woźniak, Lars Lischke, Benjamin Schmidt, Shengdong Zhao, and Morten Fjeld. 2014. Thaddeus: A Dual Device Interaction Space for Exploring Information Visualisation. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational* (NordiCHI ’14), 41–50. https://doi.org/10.1145/2639189.2639237

58. Neng-Hao Yu, Li-Wei Chan, Seng Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Fang-I Hsiao, Lung-Pan Cheng, Mike Chen, Polly Huang, and Yi-Ping Hung. 2011. TUIC: Enabling Tangible Interaction on Capacitive Multi-touch Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI ’11), 2995–3004. https://doi.org/10.1145/1978942.1979386

59. Neng-Hao Yu, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Meng-Han Lee, Mike Y. Chen, and Yi-Ping Hung. 2011. Clip-on Gadgets: Expanding Multi-touch Interaction Area with Unpowered Tactile Controls. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (UIST ’11), 367–372. https://doi.org/10.1145/2047196.2047243