Phone as a Pixel: Enabling Ad-Hoc, Large-Scale Displays Using Mobile Devices

Julia Schwarz[1] **David Klionsky**[1] **Chris Harrison**[1] **Paul Dietz**[2] **Andy Wilson**[3]

[1]Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, 15213 USA
{julia.schwarz, dklionsk, chris.harrison}@cs.cmu.edu

[2]Microsoft Applied Sciences Group, [3]Microsoft Research
One Microsoft Way, Redmond, WA 98052
{paul.dietz, awilson}@microsoft.com

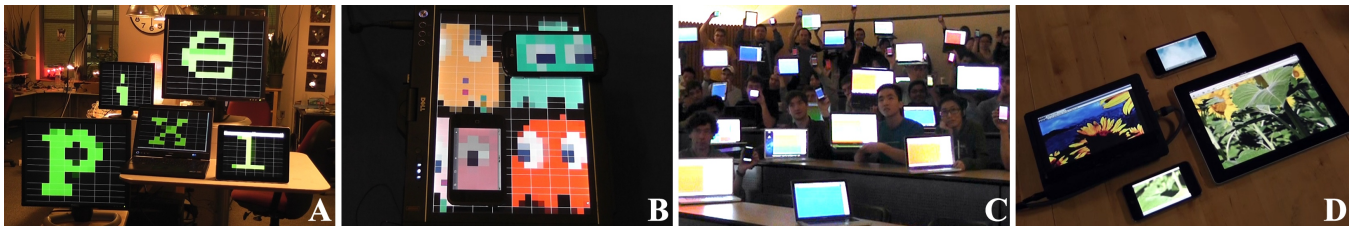

Figure 1. *Phone as a pixel* allows images to be rendered onto ad-hoc collections of small displays. From left to right: desktop monitors at arbitrary positions, tablet with smartphones laid on top, a crowd of people with devices, smartphones arranged on a table.

ABSTRACT

We present *Phone as a Pixel*: a scalable, synchronization-free, platform-independent system for creating large, ad-hoc displays from a collection of smaller devices. In contrast to most tiled-display systems, the only requirement for participation is for devices to have an internet connection and a web browser. Thus, most smartphones, tablets, laptops and similar devices can be used. *Phone as a Pixel* uses a color-transition encoding scheme to identify and locate displays. This approach has several advantages: devices can be arbitrarily arranged (i.e., not in a grid) and infrastructure consists of a single conventional camera. Further, additional devices can join at any time without re-calibration. These are desirable properties to enable collective displays in contexts like sporting events, concerts and political rallies. In this paper we describe our system, show results from proof-of-concept setups, and quantify the performance of our approach on hundreds of displays.

Author Keywords: Crowd-computer interaction, ubiquitous computing, distributed screens, computer vision, devices.

ACM Classification Keywords: H.5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces. B.4.2 [Input/Output and Data Communications]: Input/Output Devices - Image Display.

INTRODUCTION

Large-scale displays are compelling, but are generally immobile, expensive, and time-consuming to set up. However, most people have easy access to personal digital devices such as laptops, tablets and mobile phones. Collectively, these small displays can be used to create compelling, large-scale displays in a crowd or at a large gathering, such as at concerts, political rallies, or sporting events. Unfortunately, crowds are highly dynamic. Users are unlikely to form perfect grids, space out with uniform density, and will come and go as they please. Current systems for generating mosaics from smaller displays generally require a static arrangement of devices and calibration – almost always with special hardware and/or software.

This paper presents an approach for enabling potentially thousands of devices to be aggregated together as a collective display. Devices can have varying size and shape (Figure 1), and can join the display at any time in any location. Further, the only requirement is that participating devices have an internet connection and web browser. After presenting related work, we describe our system and evaluate the performance of our approach on hundreds of displays, showing that *Phone as a Pixel* is robust and immediately feasible.

RELATED WORK

Combining (or "stitching") many small displays to create a larger display has been of interest to the research community for some time (using e.g., manual alignment [3,9,18], structured light [8,12], fiducial markers [4,15], device-crossing pen strokes [6] and short-range infrared [9]). There has also been significant interest in crowd interaction [1,2,5, 11]. Combining the two is an emergent domain due to the recent proliferation of mobile devices with connectivity (see e.g., [13,17] for extended discussion).

Of particular note is Blinkendroid [3], an open source Android application that uses cross-device synchronization to generate tiled displays and animations. However, devices must be arranged in a strict grid, and are required to run

CHI'12, May 5–10, 2012, Austin, Texas, USA.

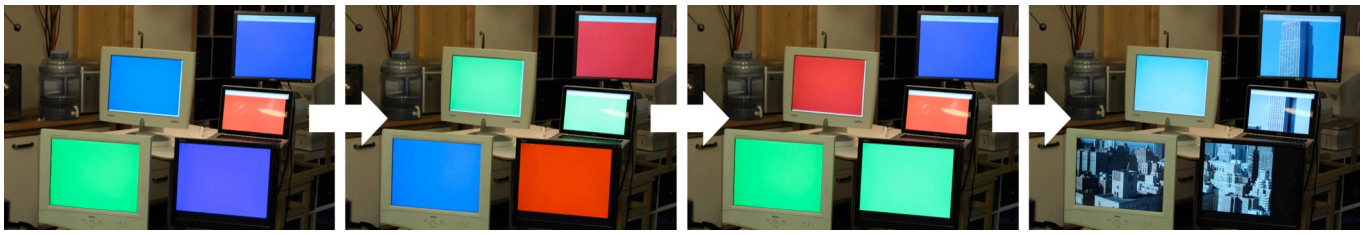

Figure 2: Three frames on left: devices outputting their IDs over a one-second period. Right frame, tiled image output.

special software. Blinkendroid's 2010 Guinness world record of 72 phones in an animated mosaic took over an hour to coordinate and setup (personal correspondence, Benjamin Weiss, 9/14/11).

Junkyard Jumbotron [4] allows for arbitrary layouts of devices, and, like our approach, also only requires devices have a web browser. Schmitz et al. [15] also provide a framework for multi-device displays and interaction using either marker-based or manual calibration. The key difference is that these approaches use fiducial markers or manual calibration. This provides a highly accurate registration of each device, but requires a high-resolution camera if more than a few dozen devices are to be used. As we will discuss, our approach allows for devices to appear as small as single pixels in our camera's view, which could allow for tens of thousands of devices to be used.

Most recent and similar is PixelPhones [8]. Devices are synchronized with the camera (using the persistent WebSocket API available in HTML5), and flash a sequence of black/white frames encoding their ID. This is feasible because devices connect to a dedicated webserver on the local WiFi subset, affording low and consistent round trip times.

An alternative to using dense collections of high-resolution displays is to use sparse collections of pixel-like devices [7, 14]. Although the expressive power of a single pixel is low, when viewed collectively, they can form compelling ambient or aesthetic displays, and even interactive experiences.

APPROACH AND IMPLEMENTATION

Our system consists of a target image, a collection of client display devices, a web server, and a camera. Each client device is first navigated to a web page containing a JavaScript application that controls all further client activity. The client then requests a unique ID from the web server. Once the client has received an ID, it flashes a color sequence on its screen, which encodes its ID (Figure 2, Left).

The camera tracks flashes emanating from each display (operating in its field of view). From this, it can determine

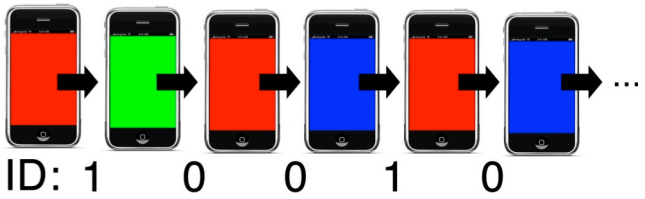

Figure 3: A device's ID is encoded using color transitions.

IDs for all active devices in parallel, along with their camera coordinates (Figure 2). For each device, a single color value (RGB triplet) or region of a larger image (X/Y offset and scale) is assigned. This data is sent to the web server and saved. After each device finishes displaying its encoded ID, it requests its data from the web server. If content has been set, the flashing ID sequence ends and the desired output is displayed. When using multiple devices, the target image is rendered across many devices and on a larger scale (Figure 2, Right).

So far, we have described color flashes that occupy the entire screen. However, a client device can break up its display into multiple "virtual" screens, requesting a unique ID for each of them (they, in turn, each flash a unique color sequence). This effectively increases the resolution if using single-color pixels. Also, since we did not have hundreds of devices available for testing, this is also how we simulated large-scale uses of our system.

Resolving IDs

We use the technique proposed in [10] to identify and localize mobile phones. Our visual encoding scheme uses a device's screen to flash a unique pattern of red, green, and blue. Transitions from red to green, green to blue, and blue to red denote a 1 in the bit sequence, while transitions in the opposite direction denote a 0. Figure 3 offers an example sequence. Color is determined using RGB Euclidian distance. Critically, because transitions are used, the devices do not have to be synchronized among themselves or with the camera, and can output their IDs at different rates.

The maximum frequency at which a device can make the above color transitions depends on the available frame rate of the camera. To guarantee a transition is not missed, the ID transition rate must be no more than half the camera frame rate according to the Nyquist Frequency [16]. Our system, including some preliminary image processing, achieves approximately 55 FPS. Thus, our device ID display rate has a theoretical maximum of 27.5 FPS (i.e., ID bits/sec). We used 20 FPS during testing.

The camera treats every pixel in its field of view as though it could belong to a device. Whenever one of the above color transitions occurs in a pixel, the camera records the encoded bit in a sliding buffer for that pixel. Every frame, the buffer for each pixel in the camera's view is checked to see if it fits the criteria for a valid device ID. If a valid ID is detected for a pixel, the camera marks that pixel as belonging to the device with that ID.

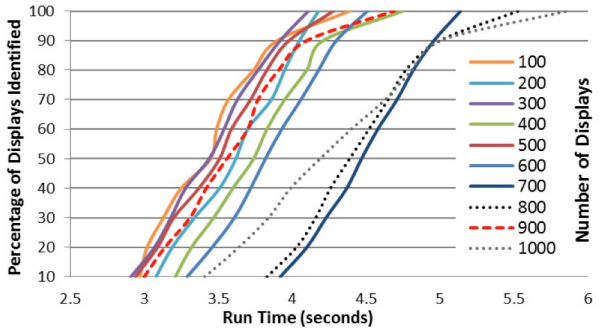

Figure 4: Average time to find a percentage of total displays.

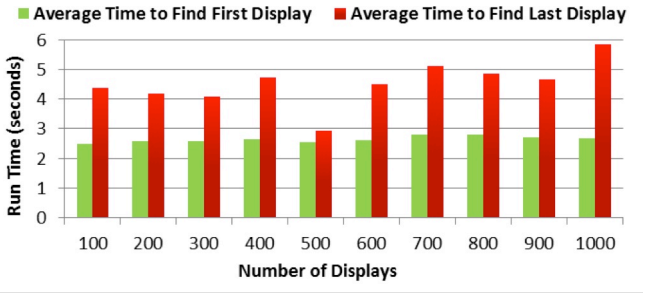

Figure 5: Average time (over three runs) to find the first and last display, starting from the camera turning on.

Due to noise in both the camera and the environment (e.g., pixels on color boundaries, occlusion by people walking by, glare from indoor lighting, etc.), it is possible for the camera to incorrectly capture or miss color transitions. In theory, this could yield a false positive ID. To improve robustness, the ID sequence is given a header. If there are *n* bits in the ID, the header contains *n* zeroes followed by a 1. This method eliminates nearly all-false positives ([10] found similar performance). Additionally, there has been tremendous work on bit sequence error correction over noisy channels that apply to our approach (e.g., parity check, turbo coding).

Assigning Content to Devices

Once the above approach has identified and localized a device, it needs to assign a color or image region. For single color values, this is done by taking all pixels in the camera image that match an ID, calculating the centroid, and using the color at that location in the target image (Figure 1A, B).

Alternatively, part of a high-resolution image can be displayed (Figure 1D and 2). To do this, the camera's program estimates how large the device screen is within the camera's field of view. Since both the device screen size and the camera screen size are known, this allows the camera to calculate a scale at which that device should display its image. The device's centroid is used as the image offset.

Advantage of Communicating over Light

We chose to encode IDs using color transitions since, when devices are very small or very far away, transitions will still be visible to a camera. Indeed, devices could be as small as one pixel, and still be identified and localized. A single 640x480 webcam could potentially digitize thousands of devices. This scale is not possible with most other visual encoding schemes such as fiducial markers [4,15].

Animation

Our technique can also be used to display animations, which even when devices are sparse, appear as movement due to *apparent motion* perceptual effects [19]. Animations would be particularly compelling in a context like a stadium, where a "wave" could sweep over thousands of devices.

To implement this, we send a sequence of colors to the devices instead of a single color. The primary challenge for animation is synchronizing individual devices, which we accomplish by synchronizing all devices with our server. This is done by averaging the offset between the server and device clock at ten different points in time.

SYSTEM EVALUATION

To better understand the effectiveness of *Phone as a Pixel*, we instrumented our code to measure how quickly our system could identify individual displays. The setup for the evaluation consisted of a single computer monitor with 100 to 1000 simulated device screens (increments of 100). Each simulated display was a div element on an HTML page that was assigned a unique ID and performed the same function as an independent device would in the real world. The camera code was instrumented to record the time at which each ID was identified and a color calculated.

The results are shown in Figures 4 and 5. In nearly all cases, the camera takes between 3 and 6 seconds to identify all device IDs and calculate a color or image region for each screen. Since the camera maintains a buffer of decoded bits for every pixel in its field of view, we expected identification time to be roughly constant as the number of simulated screens increased, and the findings support this prediction.

The time between the camera program sending content assignments to the server and devices retrieving them is much more variable. Some devices retrieve content data almost instantly, while others can take upwards of 10 seconds for the same task. We hypothesize these performance differences are caused by varying CPU speeds and wireless network latency. These factors, while troublesome, lie outside of our control. Further, they are the circumstances one would encounter in real world deployments. Nonetheless, our approach appears robust.

EXAMPLE APPLICATIONS

We built several example setups to demonstrate the feasibility of our system (see also Video Figure). Figure 1A illustrates rendering text across five desktop LCD displays, each of which contained 100 simulated devices. From our system's perspective, this was no different than having 500 mobile devices. Figure 2 (right) shows a different set of LCD displays, this time with a high-resolution image tiled across them. Figure 1B illustrates rendering a low-resolution image across 816 simulated screens on a table, with two smartphones lying on top. Figure ID shows a high-resolution image tiled over four devices resting on a table.

We were limited by what displays and devices we had available in our lab. However, there are many interesting real world uses for such a crowd display technology [1,2,5, 11,14,17]. As already discussed, this could allow for huge distributed displays at events like company retreats, Olympic events, and music performances.

Also interesting to consider is the fact that each person is holding a highly capable mobile device (with connectivity). Thus, users could not only "participate as a pixel", but also collectively control interactive elements. For example, imagine thousands of people tilting their devices (captured by onboard accelerometers) to steer a digital "wave" across the stadium. Or perhaps each person could vote for best performance, the results of which could be displayed as a crowd-sized histogram. Finally, because our approach resolves the location of each device (much higher resolution than e.g., GPS, and works inside), people could be given location specific information, such as nearby concession stands or the fastest path to an exit.

LIMITATIONS

Phone as a Pixel has potential to bring collective displays to reality, but it is important to note several factors that limit the efficacy of our system in real-world environments.

Light: As commonly experienced, using LCD screens in sunny, outdoor settings is challenging. Contemporary display technologies simply do not output enough lumens to compete with the sun. This lack of contrast makes capturing screen color-transitions difficult, even with high quality camera equipment. As a result, for the time being, *Phone as a Pixel* is best used indoors or in lower light conditions.

Glare: Light bouncing off the device's screen can also overwhelm the color transitions. Glossy screens, commonplace on mobile devices, have compounded this problem. Fortunately, if a device has severe glare, it does not affect the identification of other devices. Thus, in the worst case, a device with glare simply doesn't get utilized for the display.

User movement during recognition: As described previously, our system takes between 3 and 6 seconds to identify all devices. If a user moves his or her device too much (an entire device-width over) during the ID transmission period, the camera will fail to identify a valid ID for the device. In this case, the device will need to go through another round (another ~3 seconds) before being identified. As a consequence, users must be fairly still while registering their device. Recognition only fails if a user continuously moves during several recognition periods. This rarely occurred during our tests with multiple sets of users.

User movement after recognition: If people move after being assigned a color, the image or animation will begin to break down. The only option in this case is to periodically re-acquire device locations, which we implement through a "reset" pushed from the server. However, with collective displays, the focus is on the aggregate. Thus if a few people wander, the overall image is not affected since hundreds of others still make up the bulk of the display.

CONCLUSION

Phone as a Pixel is a technique for creating large, ad-hoc displays using a collection or arbitrarily positioned, commonly available, consumer devices. Using a web server and a camera, *Phone as a Pixel* can identify and locate displays to create compelling, large-scale mosaics of pixels or images. Our system is scalable and cross-platform, allowing for displays to be made up of thousands of individuals and opening up opportunities for creating crowd-based displays.

REFERENCES

1. Barker, T., Haeusler, M.H., Maguire, F., and McDermott, J. Investigating political and demographic factors in crowd based interfaces. In *Proc. OZCHI '09*. 413-416.
2. Barkhuus,L. and Jørgensen, T. Engaging the crowd: studies of audience-performer interaction. In *Proc. CHI EA '08*. 2925-2930.
3. Blinkendroid Project. http://code.google.com/p/blinkendroid
4. Borovoy, R. and Knep, B. Junkyard Jumbotron. http://jumbotron.media.mit.edu
5. Brown, B., O'Hara, L., Kindberg, T., and Williams, A. Crowd computer interaction. In *Proc. CHI EA '09*. 4755-4758.
6. Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P. and Smith, M. Stitching: pen gestures that span multiple displays. In *Proc. AVI '04*. 23-31.
7. LED Throwies. http://graffitiresearchlab.com/projects/led-throwies
8. Lee-Delisle, S. PixelPhones. http://sebleedelisle.com/2011/09/pixelphones-a-huge-display-made-with-smart-phones/.
9. Merrill, D., Kalanithi, J., and Maes, P. Siftables: towards sensor network user interfaces. In *Proc. TEI '07*. 75-78.
10. Miyaoku, K., Higashino, S., and Tonomura, Y. C-blink: a hue-difference-based light signal marker for large screen interaction via any mobile terminal. In *Proc. UIST '04*. 147-156.
11. O'Hara, K., Glancy, M., and Robertshaw, S. Understanding collective play in an urban screen game. In *Proc. CSCW '08*. 67-76.
12. Okatani,T. and Deguchi, K. Easy Calibration of a Multi-projector Display System. *Int. J. Comput. Vision,* 85, 1 (Oct. 2009), pp. 1-18.
13. Reeves, S., Sherwood, S., and Brown, B. Designing for crowds. In *Proc. NordiCHI '10*. 393-402.
14. Sato, M., Hiyama, A., Tanikawa, T. and Hirose, M. Particle Display System - Virtually Perceivable Pixels with Randomly Distributed Physical Pixels. *Journal of Information Processing*, Vol. 17, 2009, pp. 280-291.
15. Schmitz, A., Li., M., Schonefeld, V., and Kobbelt, L. Ad-Hoc Multi-Displays for Mobile Interactive Applications. In *Proc. Eurographics '10*. 45-52.
16. Shannon, C.E. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37, 1 (Jan. 1949), pp. 10–21.
17. Terrenghi, L., Quigley, A., and Dix, A. A taxonomy for and analysis of multi-person-display ecosystems. *Personal Ubiquitous Computing,* 13, 8 (Nov. 2009), pp. 583-598.
18. Welikesmall, Inc. iPod Wall. http://vimeo.com/13404489
19. Wolfe, J.M., Kluender, K.R., Levi, D.M., Bartoshuk, L.M, Herz, R.S., Klatzky, R.L., and Lederman, S.J. 2006. Sensation and Perception. Sinauer Associates, Sunderland, MA.