SUR40 Linux: Reanimating an Obsolete Tangible Interaction Platform

Florian Echtler
Mobile Media Group
Bauhaus University
Weimar, Germany
florian.echtler@uni-weimar.de

Martin Kaltenbrunner
Interface Culture Lab
University of Art and Design
Linz, Austria
martin.kaltenbrunnner@ufg.at

ISS '16, November 06-09, 2016, Niagara Falls, ON, Canada
ACM 978-1-4503-4248-3/16/11.
http://dx.doi.org/10.1145/2992154.2996778

Abstract

Optical sensing technologies are among the most versatile hardware solutions for interactive surfaces, as they are capable of recognizing touch as well as (limited) hover state in addition to printed tokens. One widely used system is the Pixelsense/SUR40, currently one of very few devices which provides these capabilities in the form factor of a regular table, thereby allowing working at the device in a sitting position. Unfortunately, the device has been discontinued by the manufacturer, provides only an unsupported SDK on an outdated operating system, and has a gathered a reputation for high latency as well as sensitivity to environment light.

In this paper, we present our research into modernizing and extending the SUR40 system. By switching to a Linux operating system running a custom video driver, we are able to provide lower latency, support other types of optical tags and improve the system's robustness, particularly regarding external lighting conditions. We present an analysis of the device's internals, a comparison of quantitative performance measurements, and an outlook into extending the tangible interaction capabilities with an improved cross-platform development framework.

ACM Classification Keywords

H.5.m. [Information Interfaces and Presentation (e.g. HCI)]: Miscellaneous

Author Keywords
SUR40; Linux; computer vision; touch sensing; tangible interaction

Introduction & Related Work
After a decade of fundamental tangible interaction research initiated by Ishii et.al. [3], large-scale interactive surfaces became popular in 2005 following the seminal publication by Han [2]. Initially, most research setups were custom-built constructions based on infrared cameras and consequently employed optical sensing using techniques such as FTIR, diffuse illumination or combinations thereof [6].

Although capacitive sensing has been widely used for small-scale touch sensors, large-scale capacitive surfaces have only recently started to become available. Even if capacitive surfaces are generally perceived to have more robust touch detection than optical systems, tangible interaction with, e.g., fiducial tokens is still a domain where optical sensing has distinct advantages.

One major drawback of many existing optical systems, however, is their bulkiness. Camera-based systems require a certain amount of distance and space for the light path, and therefore often take the form of large boxes with closed sides. Very few systems with optical sensing exist which don't have this limitation, either by arranging a large number of cameras close to the surface (e.g. MultiTaction[1] systems) or by integrating the sensor directly into the display.

In this paper, we present our results from reverse engineering one of the latter systems, the SUR40 tabletop system. This was the only commercially available system with optical sensing that has directly been integrated into the display, and therefore provides an unique combination of features, such as the ability for the user to work on the system like at a regular table, and the ability to detect optical tags and hovering hands. As this system is still very popular in a research context, even though it has been discontinued by the manufacturer, we believe that the background information we are able to provide on the system's hardware and software will help other researchers to optimize their applications and to continue using the device for their research.

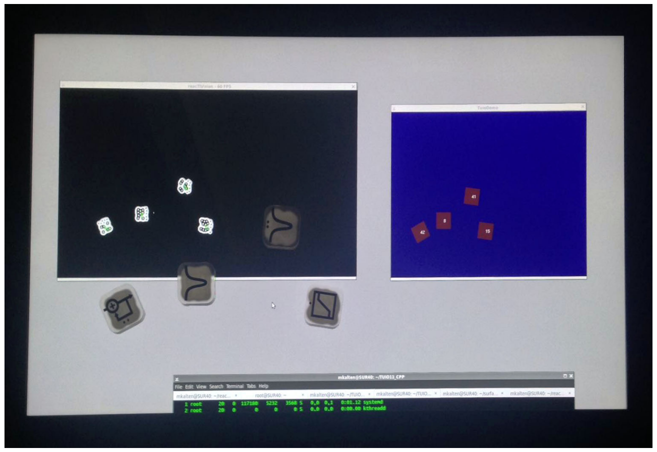
Figure 1: reacTIVision running natively under SUR40 Linux.

Hardware Description
The SUR40 is one of the few commercial examples of an optical in-cell sensor, i.e. an infrared sensor which is directly integrated with a flat-panel display. The LCD screen of the SUR40 has a standard resolution of 1920x1080 pixels at a diagonal size of 40 inch, while the sensor covers the same screen area at half resolution, i.e. 960x540 pixels. The screen dimensions are approximately 89x50 cm, resulting in

[1] https://www.multitaction.com/

a sensor resolution of 0.93 mm. The sensor is connected to the integrated host computer via an USB 2.0 connection and delivers 8-bit greyscale images at a constant rate of 60 Hz.

One interesting aspect of the SUR40 hardware is that blob/touch detection is performed directly in the panel controller through a custom FPGA chip. This is in contrast to most custom-built systems, where only a raw camera stream is provided and then interpreted by computer-vision software. On the SUR40, however, detected blobs are sent to the host system over a separate USB channel and can directly be interpreted as touch events without having to expend significant CPU power for image-processing tasks.

Reverse Engineering

Most of the presented information was gathered by booting the SUR40 with a Linux operating system, running the native Windows environment inside a virtual machine and observing the USB traffic between the hardware and the driver running inside the VM. As a result of this process[2], we were also able to write a native Linux driver providing standard system interface access to both the processed touch data (evdev) and the raw video stream (V4L2). This driver has been included into the official mainline Linux kernel since version 4.1 and enables users to run an unmodified Linux OS directly on the SUR40 hardware with full access to all features of the device (including native touch support). A standalone and continuously improved version of this driver is available on Github[3].

Performance Evaluation

As noted earlier, in contrast to its innovative hardware design the SUR40 is not without performance issues. In particular, it has developed a bad reputation for its high sensitivity to environmental light and its significant latency. While those issues are common to most optical tabletop setups, current camera based systems provide lower latency and can be better shielded from environmental light through optical filtering.

[2]Details at http://floe.butterbrot.org/matrix/hacking/surface/
[3]https://github.com/floe/surface-2.0/

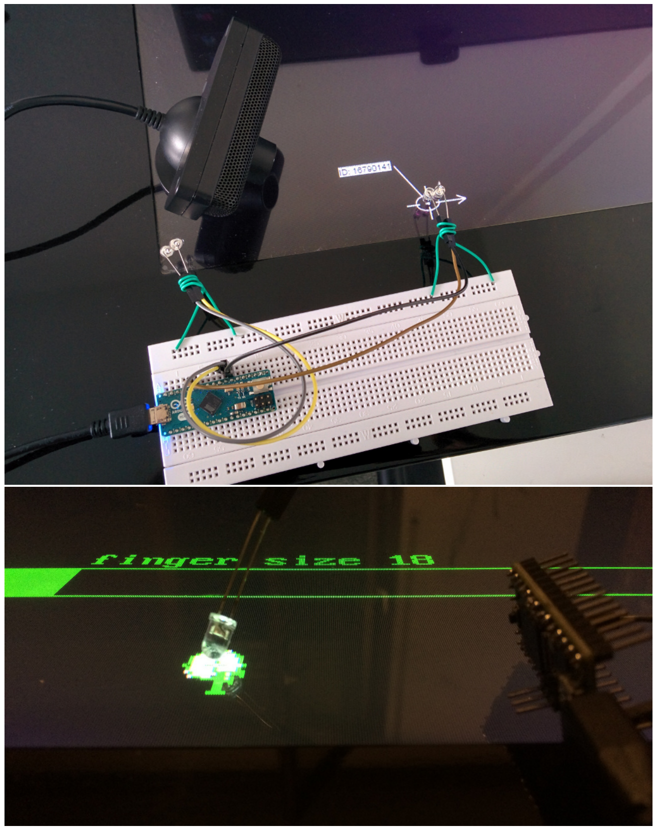

Figure 2: top: End-to-end optical latency measurement tool. bottom: USB-based optical latency measurement tool.

Touchscreen latency is difficult to measure directly, as it is hard to determine the exact moment of contact between the touching object and the screen. Deber et al. presented he "Latency Hammer", a mechanical measurement device for capacitive touch screens[1]. However, this approach is not directly applicable to optical screens such as the SUR40.

To obtain objective measurements of the perceived overall system latency, we created an optical measurement device based on an Arduino and infrared LEDs. This device alternatingly illuminates two spots on the SUR40 surface, thereby triggering a simulated touch event. We run the standard software designed to visualize incoming touch events. The entire setup is recorded with a PS3Eye infrared camera at 187 frames per second, which then allows a custom OpenCV application to compute the delay between lighting up the IR LED and its on-screen visual response. This enables us to measure the entire end-to-end latency with a temporal resolution of 5.3 ms (see also figure 2).

Under the original Windows environment, we used the provided "Touch Input Visualizer" and adjusted our software to detect the appearance of a touch label after the LEDs had been turned on. We repeated this process for 10 measurements and calculated a total end-to-end latency of 153 ms (+- 10 ms). This is noticeably higher than the generally agreed-upon upper latency bound of 100 ms and provides one explanation for the high perceived latency.

Repeating the same experiment under Linux, we measured the total latency of our video-based driver implementation, based on the raw video data provided by the SUR40 device and its processing by a local reacTIVision installation. In this case, we achieve a comparable result of 163 ms (+- 10 ms), although the video data is now already being processed on the host computer.

In a second experimental setup we have created a similar device comprised of a Teensy Board and a single LED simulating an alternating touch event triggered every second (see also figure 2). Before the LED is illuminated, the device sends a simultaneous event through a USB/serial connection. We chose the Teensy board for its real-time capabilities and low serial latency of around 1ms, allowing the precise measurement of the delay until the first raw image showing a bright spot. For comparison we conducted a similar experiment with a PS3Eye camera connected through a standard USB port. While the USB camera showed a constant delay of roughly one frame, our SUR40 measurements revealed a minimum delay of roughly three frames (50ms) before the touch point became visible. Due to the asynchronous signalling, the maximum measured latency was gradually shifting towards four frames (66ms) as expected. The total acquisition latency can be considered as the sum of sensor latency, image processing and USB transport of the raw image data.

We subsequently also measured the "headless" latency with a modified TUIO client, which provided the total delay for the above V4L2 image acquisition, plus the reacTIVision image analysis and TUIO transport without display. This measurement exposed an expected latency of roughly one additional frame, resulting in an average minimum latency of 66 ms and a maximum of 82 ms by average.

LCD panel refresh (sensor readout) at 60 Hz	1 frame
internal image buffer + processing	1 frame
USB transfer (506 kB/frame at 480 MBps)	1 frame
reacTIVision and TUIO	1 frame
66ms Total Latency (before display)	4 frames

Table 1: Latency analysis (average case)

Based on our experiments the perceived overall latency can be interpreted as a combination of individual delays accumulated by data acquisition, image processing, API communication and finally the display. We can also conclude that the majority of the total latency is caused by the sensor processing itself as well as a considerable delay of the display, which both eventually could be improved with an extended knowledge of the hardware internals. Table1, analyzes the average latency of these individual sources and how much they contribute to the observed total latency.

User Space Interaction

By having direct access to the raw video stream over the standard V4L2 interface, it is now possible to run vision-based interaction frameworks such as reacTIVision [4] directly on the SUR40. In turn, this enables the detection of alternative fiducial markers in addition to the original byte tags. It is now also possible to adjust parameters of the software-based touch detection algorithm on the fly and thereby react to changing external lighting conditions. Finally, this more modular and open software stack allows researchers to experiment with different touch, hover and token detection algorithms on a high-performance platform.

Integrating reacTIVision and TUIO

Due to our standard V4L2 driver, the current reacTIVision 1.5.1 release can be run the SUR40 Linux platform. The image processing in user space allows a much more flexible and expandable analysis of the tangible interaction on the tabletop surface, without adding significant overhead to the overall system performance as our above measurements have shown. reacTIVision currently consumes roughly 30-40 percent of the dual core 2.9GHz AMD Athlon CPU, easily processing the provided 960x540 raw image frames at 60fps. The current development version 1.6 now also implements the full TUIO 1.1 specification, providing raw blob analysis in addition to the standard fiducial and finger tracking. In controlled lighting condition the overall tracking performance is considerably more robust and easier to calibrate than compared to camera based systems. Compared to the limited possibilities of the original Windows .NET SDK, this native implementation of the TUIO framework[5] also provides a widely extended choice of programming languages and environments that can now be used on the SUR40 platform.

Environment Light

The original Windows environment provides a calibration tool for the SUR40 which requires alternate placement of a black and a white board on the surface. This establishes minimum and maximum values for the brightness and then uses these values to configure a simple normalization step in the device hardware. However, large changes in the dynamic brightness of the environment (e.g. sudden appearance of sunlight in the room) will usually cause this static threshold to result in very noisy touch data, thereby sometimes rendering the device unusable.

The realtime frame-equalization and adaptive threshold of the reacTIVision engine generally provides a more flexible and robust approach to changing light conditions, which is still able to track fiducial markers in brighter environments. Therefore we observed a more robust tracking performance even with an uncalibrated raw camera image.

Pending Improvements

Due to our reverse-engineering approach, we still have limited access to some of the internal configuration, features and processing details of the SUR40 sensor component, which has not been sufficiently exposed by the manufacturer. Based on the provided functionality and observed latencies, we assume that the overall system performance could be significantly improved by a deactivation of the hardware image processing, resulting in an earlier delivery of the raw sensor data for user-space processing. This step could potentially reduce the acquisition latency by roughly one frame (16ms), in equivalent to our USB camera measurements. In addition to a desirable reduction of the system latency, we are also looking in the possible adjustment of the "sensor camera" parameters, such as exposure, brightness and contrast if possible, as well as the implementation of a user-space tool for the overall device configuration and calibration.

Conclusion & Outlook

In this paper, we have presented an in-depth look at the hardware and software of the widely installed SUR40 device in order to provide previously undocumented background information on its inner workings. We also present a production-ready Linux kernel driver for the device, thereby enabling a contemporary operating system to run on the hardware. This, in turn, enables a wider range of alternative interaction frameworks to run on the device as well as the development of custom experiments, thereby greatly expanding the functionality, flexibility and future sustainability of the device.

As we will continue the ongoing development of the kernel-level drivers as well as the user-space TUI tools, we are also looking for collaborators, testers and users on this platform. At this occasion we'd also like to encourage the original Samsung and Microsoft developers to support the completion of this open source effort.

Acknowledgments

We'd like to thank the Linux-Media kernel developers and especially Hans Verkuil for their help in testing, debugging and improving the V4L2 driver for publication.

REFERENCES

1. Jonathan Deber, Bruno Araujo, Ricardo Jota, Clifton Forlines, Darren Leigh, Steven Sanders, and Daniel Wigdor. 2016. Hammer Time!: A Low-Cost, High Precision, High Accuracy Tool to Measure the Latency of Touchscreen Devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2857–2868. DOI: http://dx.doi.org/10.1145/2858036.2858394

2. Jefferson Y. Han. 2005. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In *UIST '05: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. 115–118.

3. H. Ishii and B. Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *CHI '97: Proceedings of the Conference on Human Factors in Computing Systems*. 234–241. citeseer.ist.psu.edu/ishii97tangible.html

4. M. Kaltenbrunner and R. Bencina. 2007. reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction. In *TEI '07: Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. 69–74.

5. M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. 2005. TUIO: A Protocol for Table-Top Tangible User Interfaces. In *Proceedings of Gesture Workshop 2005 (GW '05)*.

6. Johannes Schöning, Peter Brandl, Florian Daiber, Florian Echtler, Otmar Hilliges, Jonathan Hook, Markus Löchtefeld, Nima Motamedi, Laurence Muller, Patrick Olivier, Tim Roth, and Ulrich von Zadow. 2008. *Multi-Touch Surfaces: A Technical Guide*. Techreport. Technische Universität München.