SurfaceStreams: A Content-Agnostic Streaming Toolkit for Interactive Surfaces

Florian Echtler
Bauhaus-Universität Weimar
Weimar, Germany
florian.echtler@uni-weimar.de

ABSTRACT

We present SurfaceStreams, an open-source toolkit for recording and sharing visual content among multiple heterogeneous display-camera systems. SurfaceStreams clients support on-the-fly background removal and rectification on a range of different capture devices (Kinect & RealSense depth cameras, SUR40 sensor, plain webcam). After preprocessing, the raw data is compressed and sent to the SurfaceStreams server, which can dynamically receive streams from multiple clients, overlay them using the removed background as mask, and deliver the merged result back to the clients for display. We discuss an exemplary usage scenario (3-way shared interactive tabletop surface) and present results from a preliminary performance evaluation.

INTRODUCTION & RELATED WORK

Large-scale interactive surfaces hold promise for many collaborative scenarios, particularly those involving remote work. However, most existing systems in this space have been designed for one specific application scenario running on one specific type of device.

We present SurfaceStreams, a content- and device-agnostic toolkit for rapidly assembling shared interactive surface applications. SurfaceStreams builds on widely used libraries such as OpenCV and GStreamer, and supports a variety of input devices, including the RealSense and Kinect depth cameras, the SUR40 tabletop, or a plain webcam. Example usage scenarios for SurfaceStreams include remotely shared whiteboards [15, 5, 1], ad-hoc projected interactive surfaces [8, 11], or shared remote tabletop settings [14, 13].

A major design goal of SurfaceStreams was the capability to avoid "video loops" which can easily occur as soon as two or more display-camera systems are linked together. Therefore, we support dynamic background removal by detecting and subtracting the background plane when depth data is available.

UIST '18 Adjunct October 14–17, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5949-8/18/10…$15.00
DOI: https://doi.org/10.1145/3266037.3266085

Figure 1. Remote board gaming scenario, implemented with SurfaceStreams (left: remote user 2, right: local user). [Video]

Our work generalizes from a large body of existing research into shared interactive surfaces. In addition to the works mentioned above, we draw inspiration from multi-user collaborative virtual environments [12, 4], projector-based telepresence applications [3, 6], and real-world deployments of interactive surfaces [7, 2].

SURFACESTREAMS ARCHITECTURE

SurfaceStreams uses the GStreamer library; below, we use its notation "`component1 optX=123 ! component2 ! ...`" to describe a dataflow pipeline in which data is processed first by component 1 with option X, then component 2, and so on.

The client component of SurfaceStreams continuously performs the following tasks:

1) acquire RGB+depth video stream from a capture device,

2) detect major plane in the depth data using RANSAC,

3) set key color (bright green) on pixels near or below the plane,

4) extract and rectify a quadrilateral area from the image, and

5) pass result on to a GStreamer pipeline.

If no depth data is available, e.g. when using an infrared camera or a regular webcam, the data can either be passed on to the rectification component as-is, or an intensity-based heuristic (Otsu threshold) is used to classify pixels into background and foreground.

This approach offers maximum flexibility: e.g. for debugging and configuration purposes, the pipeline can contain just a local video sink to display the output ("`videoconvert ! fpsdisplaysink`"). For connecting

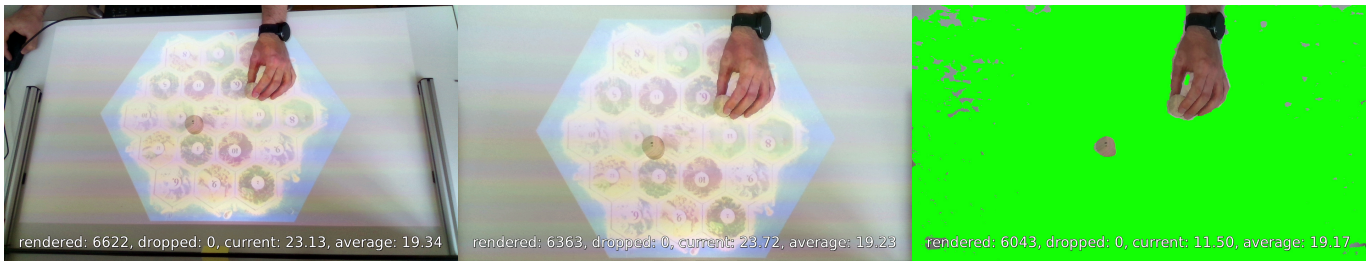

Figure 2. SurfaceStreams sends rectified and background-subtracted depth camera video via GStreamer (left: raw video with projected game board and physical tokens, center: rectified to projection area, right: background replaced by chroma-key).

to the server (see below), the pipeline uses a JPEG encoder, RTP (Realtime Transport Protocol) payloader, and UDP network sink ("`jpegenc quality=75 ! rtpgstpay ! udpsink host=1.2.3.4`"). For saving the resulting stream to a file, the pipeline contains a H.264 encoder and file sink ("`x264enc ! mp4mux ! filesink name=out.mp4`").

The server component receives multiple RTP streams, uses the color-key information to overlay these into one result stream, and sends the result back to clients. The incoming RTP buffers are expected to contain JPEG-compressed image data at HD resolution (1280x720), and are dynamically "cross-mixed" across all incoming streams (see figure 3). The mixing is performed using a chroma-key approach, replacing bright green areas with image data from the other streams. Alternatively, the image data could also directly contain an alpha channel which is used for mixing the different streams.

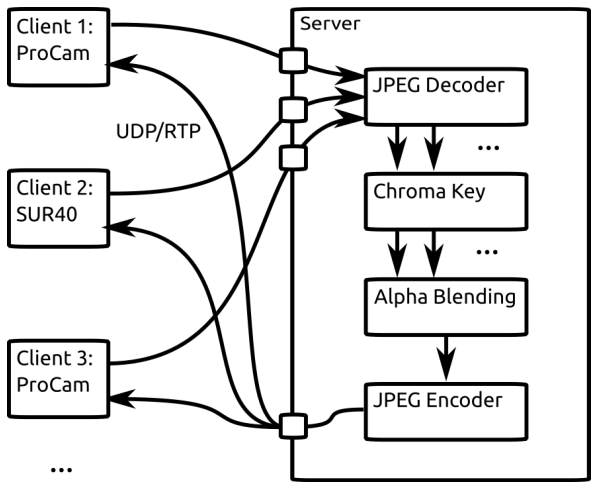

Figure 3. SurfaceStreams dataflow for example scenario.

However, none of the compressed video formats currently provided by GStreamer offer support for including an alpha channel, and the bandwidth requirements of uncompressed HD video (roughly 55 MB/s at 15 FPS) would place significant strain on the network infrastructure, especially if multiple clients are in use. Consequently, a video compression method needs to be used. Even the relatively simple MJPEG compression already reduces the required bandwidth for one stream to about 2.8 MB/s at 15 FPS. Using MJPEG instead of a more complex encoding scheme such as H.264 or VP9 also has the advantages that no additional metadata needs to be transferred, that the computational requirements for encoding and decoding are low, and that every frame can be decoded separately, thereby increasing robustness to network dropouts.

The end result is a new stream containing non-background data from all incoming streams stacked on top of each other, which is then re-encoded and delivered back to all clients simultaneously using the same data format.

Although the server component adds extra complexity, it allows to reduce the total required network bandwidth considerably, especially in a scenario with 3 or more clients such as the following one.

EXAMPLE SCENARIO

As an example scenario, we implement a heterogeneous 3-way shared tabletop system with SurfaceStreams (see also figure 1). To the best of our knowledge, this is also the first system which remotely integrates more than two tabletop devices simultaneously in a telepresence scenario.

Our setup consists of one projector-camera (ProCam) system with a Realsense D415 depth camera, one ProCam system with a regular Logitech C920 webcam, and a Samsung SUR40 (PixelSense) tabletop device. All devices are connected to a 100 MBit Ethernet wired network. The size of the shared display area is defined by the fixed 40" screen of the SUR40 and is therefore 89x50 cm. The projection on both ProCam systems is adjusted to cover the same size. As there is an inevitable mounting offset between each camera and the projected display area, the resulting perspective transformation between the two image spaces needs to be compensated. To this end, a four-point calibration is built into SurfaceStreams which exactly aligns the outgoing stream with the projected area.

Due to the simple encoding, the total load on the clients is moderate: on an Intel Core i5-4310M with 2.7 GHz, the entire depth segmentation and encoding process of a HD stream at 15 FPS results in approximately 32-33% of CPU utilization. Decoding the incoming stream adds another 6-7%, resulting in a total of at most 40% of CPU load. As stated above, the network load of the incoming stream is approximately 2.8 MB/s in the worst case, which can easily be handled even by today's domestic network connections. Due to its large amount of chroma-key content, the outgoing stream can generally be compressed better and will consume about 500-700 kB/s of upstream bandwidth.

CONCLUSION & FUTURE WORK

We present SurfaceStreams, a toolkit for rapidly prototyping interactive surface applications using a variety of image sensor and display options. Currently, SurfaceStreams is entirely content-agnostic and will only transport visual information between surfaces without any interpretation beyond removing the background plane. In the future, we plan to include an optional GStreamer connector to the widely used tracking software package reacTIVision [9] to detect fiducials, touch events, or arbitrary objects in the incoming image data, and deliver synchronized TUIO 2.0 [10] data back to the clients, which can then be used to trigger content-specific interaction events. In addition, we will evaluate how "virtual" clients which only create synthetic image data can be used to insert additional content into a surface application scenario via the standardized GStreamer interfaces.

SurfaceStreams is available as open source under GNU LGPL 3.0 license at https://github.com/floe/surface-streams.

REFERENCES

1. Ignacio Avellino, Cédric Fleury, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2017. CamRay: Camera Arrays Support Remote Collaboration on Wall-Sized Displays. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6718–6729. DOI: http://dx.doi.org/10.1145/3025453.3025604
2. Florian Echtler and Raphael Wimmer. The Interactive Dining Table, or Pass the Weather Widget, Please. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (2014) *(ITS '14)*. ACM, 419–422. DOI: http://dx.doi.org/10.1145/2669485.2669525
3. Andreas Rene Fender, Hrvoje Benko, and Andy Wilson. 2017. MeetAlive: Room-Scale Omni-Directional Display System for Multi-User Content and Control Sharing. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 106–115. DOI: http://dx.doi.org/10.1145/3132272.3134117
4. Leif Handberg, Charlie Gullstrom, Joke Kort, and Jimmy Nyström. SharedSpaces Mingle. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (2016) *(CHI EA '16)*. ACM, 269–272. DOI: http://dx.doi.org/10.1145/2851581.2889469
5. Keita Higuchi, Yinpeng Chen, Philip A. Chou, Zhengyou Zhang, and Zicheng Liu. 2015. ImmerseBoard: Immersive Telepresence Experience Using a Digital Whiteboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2383–2392. DOI: http://dx.doi.org/10.1145/2702123.2702160
6. Brett R. Jones, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. IllumiRoom: Peripheral Projected Illusions for Interactive Experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013) *(CHI '13)*. ACM, 869–878. DOI: http://dx.doi.org/10.1145/2470654.2466112
7. Tejinder K. Judge, Carman Neustaedter, Steve Harrison, and Andrew Blose. 2011. Family Portals: Connecting Families Through a Multifamily Media Space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1205–1214. DOI: http://dx.doi.org/10.1145/1978942.1979122
8. Sasa Junuzovic, Kori Inkpen, Tom Blank, and Anoop Gupta. 2012. IllumiShare: Sharing Any Surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1919–1928. DOI: http://dx.doi.org/10.1145/2207676.2208333
9. Martin Kaltenbrunner. 2009. reacTIVision and TUIO: A Tangible Tabletop Toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*. ACM, New York, NY, USA, 9–16. DOI:http://dx.doi.org/10.1145/1731903.1731906
10. Martin Kaltenbrunner and Florian Echtler. 2018. The TUIO 2.0 Protocol: An Abstraction Framework for Tangible Interactive Surfaces. *Proceedings of the ACM on Human-Computer Interaction* 2, EICS, Article 8 (June 2018), 35 pages. DOI:http://dx.doi.org/10.1145/3229090
11. Natan Linder and Pattie Maes. LuminAR: Portable Robotic Augmented Reality Interface Design and Prototype. In *Adjunct Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (2010) *(UIST '10)*. ACM, 395–396. DOI: http://dx.doi.org/10.1145/1866218.1866237
12. Holger Regenbrecht, Michael Haller, Joerg Hauber, and Mark Billinghurst. 2006. Carpeno: Interfacing Remote Collaborative Virtual Environments with Table-Top Interaction. *Virtual Real.* 10, 2 (Sept. 2006), 95–107. DOI:http://dx.doi.org/10.1007/s10055-006-0045-3
13. Baris Unver, Sarah A. McRoberts, Sabirat Rubya, Haiwei Ma, Zuoyi Zhang, and Svetlana Yarosh. 2016. ShareTable Application for HP Sprout. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 3784–3787. DOI: http://dx.doi.org/10.1145/2851581.2890252
14. Andrew Wilson and Daniel Robbins. PlayTogether: Playing Games across Multiple Interactive Tabletops. In *IUI Workshop on Tangible Play: Research and Design for Tangible and Tabletop Games* (2007) *(IUI '07)*. http://research.microsoft.com/en-us/um/people/awilson/publications/WilsonIUI2007/WilsonIUI2007.html
15. Jakob Zillner, Christoph Rhemann, Shahram Izadi, and Michael Haller. 2014. 3D-Board: A Whole-Body Remote Collaborative Whiteboard. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 471–479. DOI: http://dx.doi.org/10.1145/2642918.2647393