

# SplatOverflow: Asynchronous Hardware Troubleshooting

Amritansh Kwatra  
Cornell Tech  
New York, USA  
ak2244@cornell.edu

Tobias Weinberg  
Cornell Tech  
New York, USA  
tmw88@cornell.edu

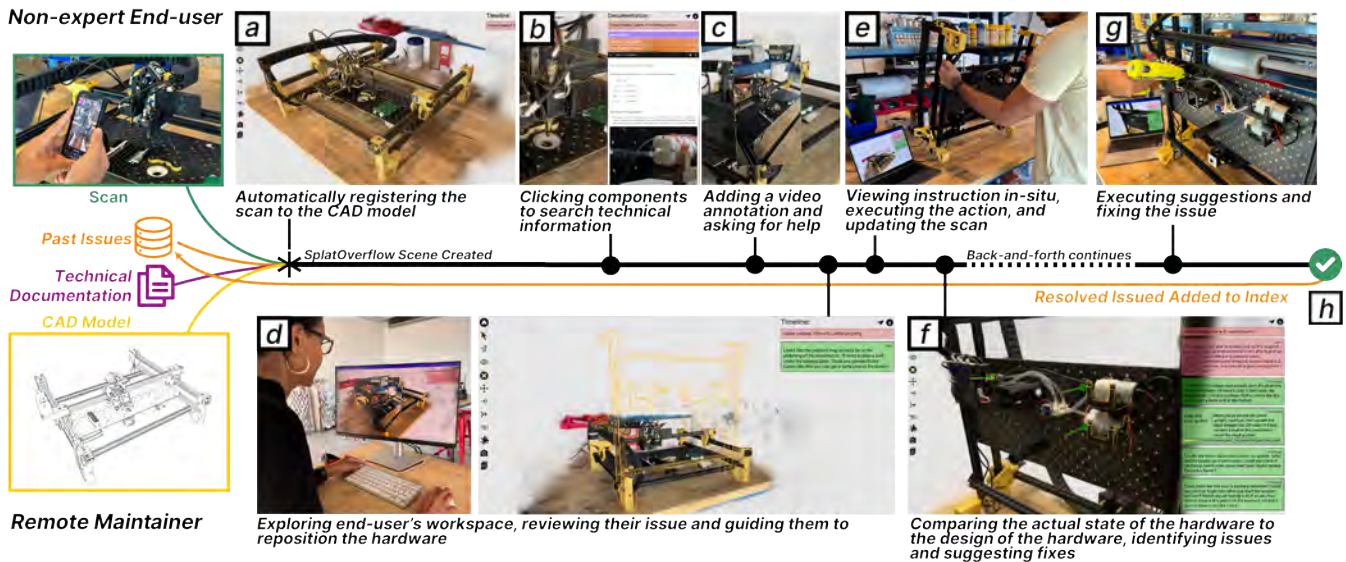
Ilan Mandel  
Cornell Tech  
New York, USA  
im334@cornell.edu

Ritik Batra  
Cornell Tech  
New York, USA  
rb887@cornell.edu

Peter He  
Cornell University  
Ithaca, USA  
ph475@cornell.edu

François Guimbretière  
Cornell University  
Ithaca, USA  
fvg3@cornell.edu

Thijs Roumen  
Cornell Tech  
New York, USA  
tjr92@cornell.edu



**Figure 1:** A workflow illustrating a non-expert end-user using SplatOverflow to troubleshoot an issue with their hardware. (a) A SplatOverflow scene consisting of a user-captured scan of their hardware registered to the hardware’s CAD model. (b) The user queries technical documentation and past issues associated with the hardware by clicking on components in the scene. (c) They request help from maintainers by sharing their SplatOverflow scene. (d) A maintainer explores the SplatOverflow scene and instructs the end-user to move their machine. (e) The local user sees instructions rendered as an overlay on their workspace and repositions their machine. (f) A maintainer compares the as-built hardware to the as-designed CAD model and suggests a solution. (g) The local user executes the suggestion and fixes the issue. (h) Once resolved, the issue, instructions, and deliberation are indexed back into a database of past issues for future users to reference.

## ABSTRACT

As tools for designing and manufacturing hardware become more accessible, smaller producers can develop and distribute novel hardware. However, processes for supporting end-user hardware troubleshooting or routine maintenance aren’t well defined. As a result, providing technical support for hardware remains ad-hoc and challenging to scale. Inspired by patterns that helped scale software troubleshooting, we propose a workflow for asynchronous hardware troubleshooting: SplatOverflow.



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

CHI '25, April 26-May 1, 2025, Yokohama, Japan

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1394-1/25/04.

<https://doi.org/10.1145/3706598.3714129>

SplatOverflow creates a novel boundary object, the SplatOverflow scene, that users reference to communicate about hardware. A scene comprises a 3D Gaussian Splat of the user’s hardware registered onto the hardware’s CAD model. The splat captures the current state of the hardware, and the registered CAD model acts as a referential anchor for troubleshooting instructions. With SplatOverflow, remote maintainers can directly address issues and author instructions in the user’s workspace. Workflows containing multiple instructions can easily be shared between users and recontextualized in new environments.

In this paper, we describe the design of SplatOverflow, the workflows it enables, and its utility to different kinds of users. We also validate that non-experts can use SplatOverflow to troubleshoot common problems with a 3D printer in a usability study.

Project Page: <https://amritkwatra.com/research/splatoverflow>.

## CCS CONCEPTS

• **Human-centered computing** → *Interactive systems and tools*.

## KEYWORDS

Hardware Maintenance, Repair, Troubleshooting

### ACM Reference Format:

Amritansh Kwatra, Tobias Weinberg, Ilan Mandel, Ritik Batra, Peter He, François Guimbretière, and Thijs Roumen. 2025. SplatOverflow: Asynchronous Hardware Troubleshooting. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 1, 2025, Yokohama, Japan*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3706598.3714129>

## 1 INTRODUCTION

Hardware design tools have enabled small teams to develop and distribute novel machines and devices for niche applications. Examples of this type of hardware range from automation equipment for mid-scale manufacturing, such as the Lumen pick-and-place<sup>1</sup>, to personal fabrication machines, like the Prusa MK3S<sup>2</sup>, to do-it-yourself gadgets like the Open Book<sup>3</sup> e-reader. We refer to such products generally as *hardware*. For hardware producers, turning a prototype into a viable product entails addressing challenges that stem from supporting a growing user base with varying technical expertise [35]. This paper focuses on one aspect of producing hardware: supporting end-user troubleshooting and maintenance.

Detailed and thorough documentation is an essential part of developing a hardware product [3, 11, 35]. However, documentation alone is insufficient to support the long tail of niche hardware issues consumers may encounter [56]. Instead, Subbaraman and Peek [66] argue that maintenance should be considered a core part of owning this kind of hardware, and systems should be designed to support end-user troubleshooting and maintenance. We take inspiration from the infrastructure that supports software maintenance and troubleshooting workflows to examine how to create similar systems for hardware.

Platforms such as GitHub and StackOverflow have supported communities of software users by allowing them to help one another troubleshoot issues and by cataloging a history of past issues

for anyone to reference. Crucially, these platforms rely on *asynchronous* communication between users. This allows users to seek out help or provide suggestions without coordinating availability or scheduling meetings with one another. Eliminating this barrier has allowed distributed communities of users to flourish and made asynchronous modes the norm in software development [2, 77]. This contrasts the workflows HCI researchers have proposed for remote expert guidance and hardware troubleshooting, which are primarily designed for synchronous modes of communication [29, 30, 54].

The utility of asynchronous workflows for troubleshooting is due, in part, to how they facilitate communication between users: via references to shared digital artifacts. These artifacts resemble boundary objects [2, 46, 65] that serve as mechanisms for communicating context and ideas online. In StackOverflow, for example, issues are accompanied by segments of code that the user is writing. Suggestions are then made as references to lines of code the user shared. The boundary object (in this case, lines of code) captures the user’s context and scaffolds the asynchronous communication with others. Importantly, this context is identical for those contributing to troubleshooting the issue and future users referencing the issue. As a result, a larger group of current and future users benefit from the knowledge gleaned in the interaction [77].

Hardware needs a robust boundary object to support the asynchronous modes of communication necessary to scale maintenance and troubleshooting infrastructure. We present SplatOverflow, a system that enables asynchronous hardware troubleshooting through a novel boundary object: a SplatOverflow scene. A SplatOverflow scene captures the as-built hardware through a scan (a 3D Gaussian Splat [41]) and renders as-designed details by aligning the scan to a shared CAD model of the hardware.

The construction of a SplatOverflow scene presents distinct advantages to local users (who have the hardware in front of them), remote maintainers, and future users who may encounter similar hardware issues. Local users benefit from access to technical documentation linked to the CAD model and the ability to describe issues on their hardware by pointing and clicking on parts rather than knowing hardware-specific vocabulary. Remote maintainers benefit from seeing the local user’s issue registered onto the familiar CAD model and being able to freely move through the scene to inspect the hardware in the local user’s environment. Finally, future users benefit from being able to retrieve and replay solutions to past issues without seeking out support. Instead, SplatOverflow re-contextualizes instructions from a past issue by overlaying them onto the current user’s environment. This ability to accommodate multiple users and maintainers is essential to how SplatOverflow can help support scaling the maintenance effort for hardware.

In this paper, we present the design of SplatOverflow, demonstrate how it can be used to troubleshoot issues on different kinds of hardware, and validate that non-expert users can use SplatOverflow to troubleshoot hardware issues.

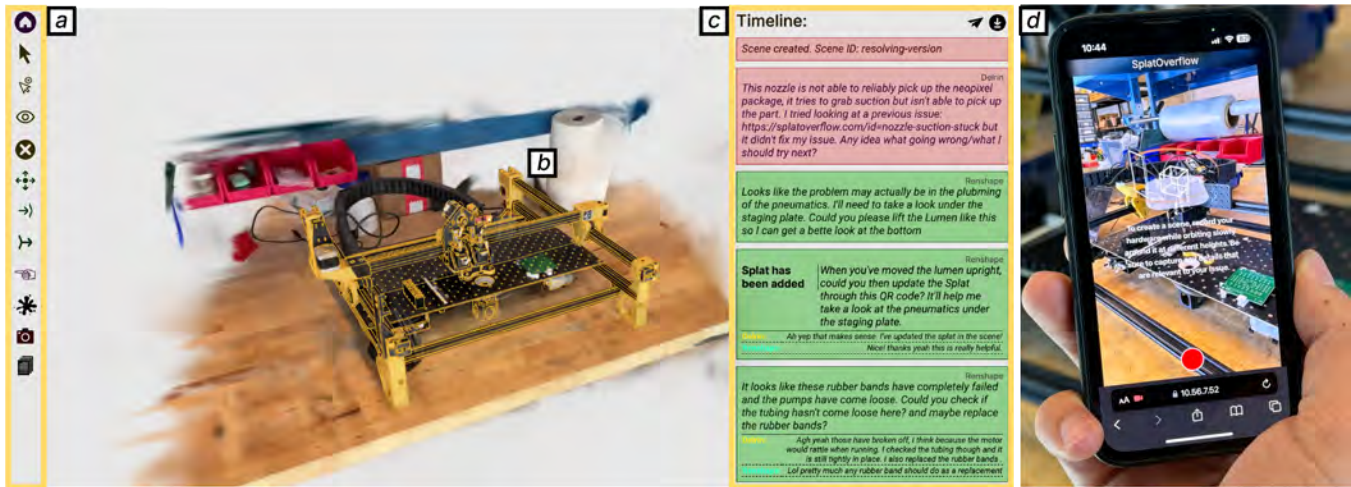
## 2 WALKTHROUGH

We demonstrate how a non-expert local end-user can troubleshoot a complex issue on the Lumen v3 Pick-and-Place machine using SplatOverflow. The Lumen is an automation machine designed to assemble PCBs by picking up surface-mount electrical components

<sup>1</sup><https://www.opulo.io/products/lumenpnp>

<sup>2</sup><https://help.prusa3d.com/tag/mk3s-2>

<sup>3</sup><https://www.oddllyspecificobjects.com/projects/openbook/>



**Figure 2: The components of SplatOverflow’s web and mobile interface. (a) The palette of gestures SplatOverflow offers for selecting parts, guiding attention, and communicating actions. (b) The 3D Gaussian Splat of the hardware is aligned and registered onto the CAD model. (c) The timeline captures the troubleshooting interaction as a sequence of instructions and responses between users. (d) SplatOverflow’s mobile interface allows local users to scan their hardware using a smartphone.**

with a vacuum nozzle and placing them on circuit boards. In this scenario, the local user’s machine can not reliably pick up parts with one of the two vacuum nozzles. They use SplatOverflow to capture the issue they are facing and receive suggestions from a remote maintainer. Figure 1 depicts a high-level visual illustration of this process.

## 2.1 Creating a SplatOverflow Scene

The user opens SplatOverflow on their smartphone to capture the issue they are facing. Figure 2(d) shows SplatOverflow’s capture interface. They record a roughly one-minute-long video, filming the Lumen from multiple angles. After recording, they upload the video, and SplatOverflow generates a scene.

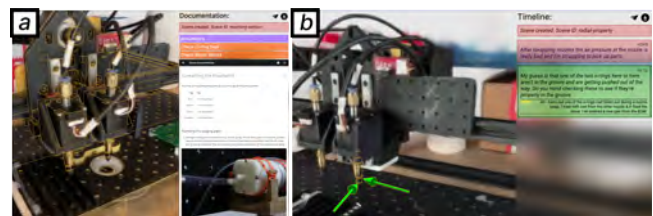
After two minutes, SplatOverflow completes generating a *low-resolution* scene. This scene is generated using down-sampled video, which yields a lower quality splat. However, this lower-resolution scene allows the user to quickly begin using SplatOverflow. After roughly four minutes from the original upload, SplatOverflow finishes generating the *full-resolution* scene containing the user’s Lumen registered onto its CAD model. SplatOverflow removes the background regions of the splat by default to preserve the local user’s privacy. What’s left is a splat of the user’s hardware and parts of the immediate work surface on which it is placed. Figure 2(a-c) shows the scene as it is rendered in the browser.

## 2.2 Visually Querying Documentation and Past Issues

First, the local user examines any technical documentation for Lumen to find a solution. They select SplatOverflow’s documentation tool and click on the nozzle in the scene. Figure 3(a) shows how SplatOverflow then displays sections of the technical documentation and past SplatOverflow issues referencing the nozzle assembly.

The user scrolls through the retrieved documentation but does not find any suggestions about troubleshooting poor suction.

Next, the user sees a past issue that references a faulty nozzle. They open the issue, and SplatOverflow re-contextualizes the instructions from that scene onto their hardware. Figure 3(b) shows the solution to a past issue recontextualized in the local user’s SplatOverflow scene. The user walks through the solution, which had to do with fixing the O-rings that seal the vacuum nozzle. They check that their O-rings are not dislodged and confirm this is not the source of their problem. They then decide to capture more details about the issue and ask for help.



**Figure 3: (a) Using the CAD model to visually query technical documentation and past issues. When the user clicks on the nozzle, SplatOverflow retrieves relevant sections from the assembly documentation and past issues referencing the part. Links to documentation are rendered in purple and previewed in the timeline as an iframe. Past issues are rendered in orange and can be recontextualized within the scene. (b) The local user recontextualizes a previous issue referencing the nozzle’s poor suction; viewing the suggestions from a prior overlaid onto their machine.**

### 2.3 Asking for Help

To share their issue with a remote maintainer, the local user adds a description of the problem they are facing. They also add a video annotation of the nozzle failing to pick up a component. SplatOverflow places the video into the scene to match the filming perspective, as shown in Figure 4(a). The video describes the dynamic aspects of the issue that cannot be captured in the scan of the workspace. The local user then posts the issue and waits for a remote maintainer to view it and offer suggestions.

### 2.4 Providing Guidance

After a few hours, the remote maintainer reviews the SplatOverflow issue and offers feedback. Figure 4(b) shows the maintainer reviewing the issue and inspecting the nozzle assembly in the scene. The maintainer cannot see any problems with the nozzle assembly and suspects that the source of the issue may be the pneumatics under Lumen’s staging plate. However, the underside of the staging plate is not visible in the machine’s current orientation.

The maintainer directs the local user to reorient the machine by manipulating the CAD model in SplatOverflow and requests that they update the captured splat after moving it. The move instruction is visualized by animating the CAD model to move from the original position to a new target position. The request to update the splat includes a QR code pointing the user to SplatOverflow’s mobile capture interface for adding a splat to an existing scene. Figure 5 shows the instructions authored by the maintainer in the timeline and how each is rendered in SplatOverflow.

### 2.5 Understanding Suggestions

The local user reviews the suggestions made by the remote maintainer. They see that the maintainer has asked them to move their machine. Clicking the instructions in the SplatOverflow timeline shows the user that they need to stand the machine on its back legs. Figure 5(d) shows the user following the maintainer’s instructions by moving the machine and updating their splat. The user scans the QR code in the timeline event and captures a new splat showing the underside of the staging plate. Once the splat is updated, the scene contains two splats, with the machine in different orientations. Figure 6(a) shows the updated scene with the Lumen standing upright and the pneumatics easily visible for the remote maintainer to inspect. The splat captures the state of the wiring and routed tubes that are not included in the machine’s CAD model.



Figure 4: (a) The local user’s video is placed as a floating screen in the SplatOverflow scene and aligned to the same perspective it was filmed in. (b) The remote maintainer reviews the local user’s issue by inspecting the vacuum nozzle in the splat and CAD model.

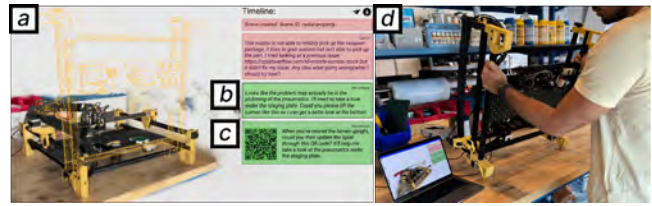


Figure 5: A reply from the remote maintainer asking the user to re-orient the machine to a new position and then update the scene with footage of the underside of the machine. (a) The wireframe corresponding to the final position the machine should be in. (b) The timeline element explaining why the movement needs to be made. When the local user clicks this element, the CAD model will animate to show how the machine should be moved. (c) The request for a new splat is rendered for the local user. The QR code links them to the mobile capture interface, where they can update the scene the QR code links to. (d) The local user performing the action specified by the maintainer and preparing to update the splat.

### 2.6 Back and Forth Communication

The maintainer sees the updated splat and examines the pneumatics under the staging plate. Using the splat, they inspect the routing of tubes (not modeled in the CAD) and assess the condition of components that may be degrading. After orbiting around the model, the maintainer feels confident that the wiring is correct but notices two concerns. First, the rubber bands holding the pumps have wholly degraded, as shown in Figure 6(c). Second, the blue fitting on one of the pneumatic fittings has come loose and will likely not provide an adequate seal, as shown in Figure 6(d). Both are potential sources of the issue, so the remote maintainer instructs the user to fix both. They use SplatOverflow’s *pointing* gesture to attach comments to the pneumatic fitting and the vacuum pumps. They instruct the

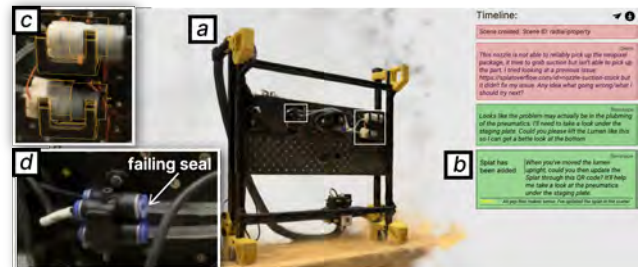
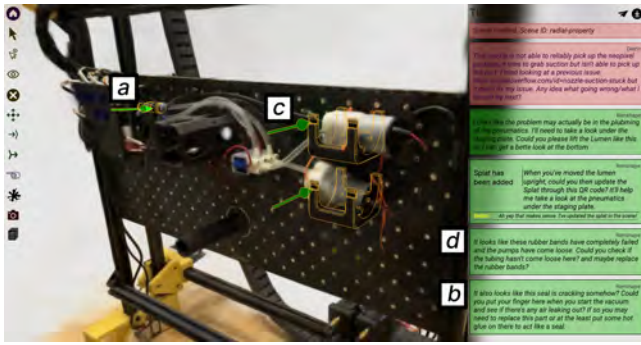


Figure 6: (a) The updated splat rendered in the browser. The machine is now standing upright as the maintainer requested, and the pneumatics in the staging plate are visible. (b) The timeline is updated to indicate that a new splat has been added, replacing the QR code with a success message. (c) A close-up of the vacuum pumps dislodged with their positions in the CAD model overlaid. (d) A close-up of the pneumatic seal that the maintainer is concerned about.



**Figure 7: The remote maintainer makes two sets of suggestions. First, they indicate that based on the splat, one of the fittings appears to be failing. They instruct the user on how to test the fitting and offer a stop-gap solution if it has failed. (a) The visualization indicates which part the remote maintainer references. (b) The timeline element with the relevant instruction. Next, the remote maintainer notices that the pumps have come loose and asks the local user to check if the fitting is still secure. They also indicate that the local user should replace the deteriorated rubber bands. (c) The visualization overlaid onto the pump mounts. (d) The timeline element with the relevant instruction.**

local user to test if the tubes leaving the air pump are still securely attached and to replace the rubber bands when they have the chance. Next, the maintainer guides the user through testing if air escapes the fitting when the pump is turned on. The maintainer asks the user to place their finger on the seal and feel for air when the pump is on. In case the seal leaks air, the maintainer suggests that the local user apply some hot glue as a stop-gap solution while waiting for a replacement part. Figure 7 shows how these instructions are rendered in the timeline and within the scene.

The local user reviews the suggestion and performs the test as described by the maintainer, as shown in Figure 8. When they turn on the pump, they feel air escaping the fitting. They apply some hot glue, and the vacuum pressure becomes more substantial. The local user orders a replacement fitting and replies to the suggestion indicating that a poor fitting was the culprit. Once the issue is resolved, the timeline is indexed to all the parts referenced in the back-and-forth exchange.

### 3 UTILITY OF SPLATOVERFLOW

This section outlines the capabilities made possible by SplatOverflow. We discuss the scenarios in which SplatOverflow addresses the limitations of existing troubleshooting artifacts, highlight the novel capabilities SplatOverflow enables, and outline the utility it creates for different users.

#### 3.1 Existing Troubleshooting Artifacts

Existing hardware troubleshooting workflows use a variety of artifacts to facilitate back-and-forth communication between parties, each with its own drawbacks. We highlight how SplatOverflow

mitigates these drawbacks by complementing each kind of artifact with additional information and context within a scene.

**3.1.1 Text Posts.** Text is commonly used to describe the physical state of hardware, how it may be erroring, or what action must be taken on the hardware. However, text descriptions can create confusion if parties do not share a common vocabulary. This is especially true when non-expert users are troubleshooting hardware with maintainers. Non-expert users may not know how to refer to specific parts or which part a provided instruction refers to. SplatOverflow explicitly links text to the referent parts from the CAD model and visually highlights corresponding parts when the text is selected. This preserves the flexibility of text while addressing issues of referential uncertainty and ambiguity.

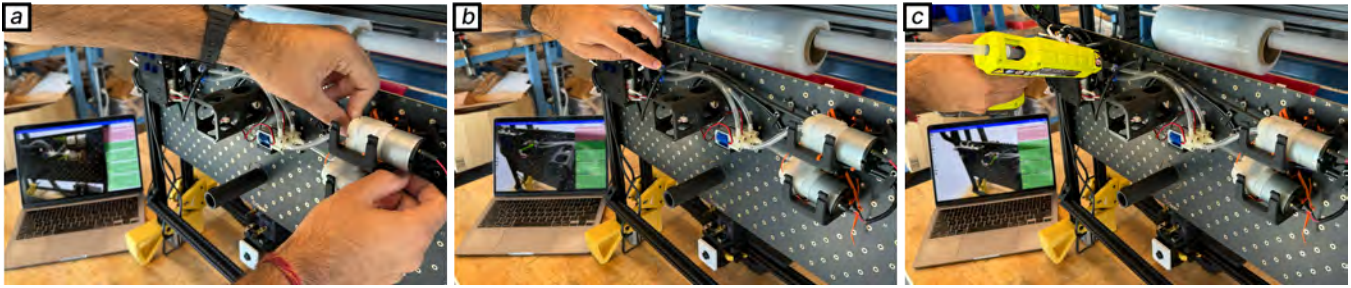
**3.1.2 Annotated Images.** Annotated images direct attention to a specific area or part of the hardware. Moreover, annotations allow maintainers to specify instructions through deictic references on the local user’s hardware, e.g., tighten this bolt (where *this bolt* is circled in an image). This lets local users reason about instructions without changing contexts, as the annotation is authored by referencing their hardware. However, image annotations can only reference what the local user captured in the image. When the image does not capture the appropriate details, maintainers must guide the user without context cues from the local user’s hardware. SplatOverflow extends the utility of image annotation for maintainers by allowing them to annotate the scene from various arbitrary viewpoints.

**3.1.3 Video Tutorials.** Video tutorials can capture detailed multi-step processes on hardware and show local users how to manipulate their hardware through demonstration. Video tutorials contain a large amount of *detail*, but can demand significant time and planning to be effective. As a result, video documentation is often used for build instructions or set-up guides, the utility of which can be pre-empted and planned for. However, troubleshooting is often about problems that are hard to preempt [56]. Creating detailed video tutorials for each of these issues as they arise is demanding for maintainers. In contrast, impromptu and unedited videos can be difficult for users to parse, leading to more miscommunication. SplatOverflow improves the utility of *impromptu video* filmed by non-expert users by situating the filming perspective of the video within the scene.

### 3.2 Novel Capabilities

In addition to addressing some of the limitations of existing artifacts, SplatOverflow enables novel capabilities that are not supported by existing asynchronous hardware troubleshooting artifacts.

**3.2.1 Indexing Issues.** In SplatOverflow, resolved issues are indexed in a database based on the parts they reference. Future users can subsequently query this database by selecting the parts they are experiencing issues with. This streamlines accessing the evolving technical information related to the hardware. Similar to the pools of technical knowledge created by the social network of copier repair technicians described by Orr [56], the indexed corpus of hardware issues created by SplatOverflow is a valuable reference for other maintainers and future users [10].



**Figure 8: The local user examines the instructions left by the remote maintainer. (a) They test whether the tubing is still firmly attached to the vacuum pump. They feel that it is, so they move to the next suggestion. (b) They test whether they can feel any air escaping the fitting that the maintainer referenced in their suggestion. They can feel air leaking out. (c) The local user tries the stop-gap solution of applying some hot glue to the fitting to improve the seal.**

**3.2.2 Independent Asynchronous Navigation.** With SplatOverflow, maintainers can independently navigate and author instructions from perspectives and on parts not intentionally captured by the local user. This adds flexibility and enables maintainers to inspect and investigate the actual state of the local user’s hardware.

**3.2.3 Issue Re-contextualization.** By defining annotations in reference to CAD data, SplatOverflow can re-contextualize them into new scenes containing the same hardware. This allows local users to view suggestions from past issues overlaid onto their own hardware and reduces the amount of context shifting needed to make sense of instructions from past issues.

### 3.3 Utility to Users

SplatOverflow mediates an interaction between two kinds of users, non-expert *local* end-users and expert *remote* maintainers. This section underscores the utility to each type of user and the conditions in which SplatOverflow is most beneficial.

**3.3.1 For End-Users.** SplatOverflow provides the most utility to users whose expertise is not in the hardware itself but in what they use it for. These users depend on the hardware for productivity but are not familiar with the intricacies of its design. As such, maintenance and troubleshooting are simultaneously critical and challenging tasks for these users to carry out. SplatOverflow benefits these end-users in three ways: (1) By indexing past issues and relevant technical information, SplatOverflow helps end-users search for workflows to remedy issues they are facing. (2) By re-contextualizing past issues, SplatOverflow allows end-users to reason about and follow existing instructions within the context of their own hardware. (3) By capturing the hardware context in a scan, SplatOverflow allows end-users to seek help without knowing hardware-specific terminology. These benefits are accessed by capturing a brief video of their hardware.

**3.3.2 For Maintainers.** As hardware develops, end-user maintenance and troubleshooting is essential to support [25, 26, 35, 56]. This entails hardware designers, producers, or enthusiasts taking on the role of maintainers to support the growing number of non-expert end-users. SplatOverflow helps maintainers scale support for hardware in three ways. (1) By creating an asynchronous troubleshooting workflow, SplatOverflow allows maintainers to address

end-user issues at their own pace. (2) By registering an immersive scan to a CAD model, SplatOverflow allows maintainers to compare the actual state of the hardware to its design without context shifting across artifacts. (3) By indexing resolved issues, SplatOverflow allows maintainers to inspect and share links to related issues and relevant solutions.

These features help maintainers attend to individual issues. However, as more issues are addressed and indexed through SplatOverflow, the benefits to maintainers expand. For example, the maintainer’s effort in troubleshooting issues is multiplied when future end-users can reference the solution without seeking out help. Moreover, when multiple people are responsible for maintenance and troubleshooting, they can reference the pool of advice offered by others in the past while diagnosing new issues. As such, SplatOverflow supports maintainers in scaling hardware support.

## 4 RELATED WORK

SplatOverflow builds on studies of hardware maintenance practices, documentation systems, telepresence systems for expert support, and spatial rendering techniques.

### 4.1 Hardware Maintenance Practice

SplatOverflow draws on the findings of Julian Orr’s ethnographic work, which studies the sharing practices of Xerox repair technicians in the field [56]. Orr’s work found that such troubleshooting was improvisational and centered collaborative sensemaking between a technician, a client, and a machine [56]. This form of collaboration between different parties remains essential for effective maintenance in modern, distributed communities of hardware users. Dunn et al. [25] study one such distributed community: users of the Jubilee motion platform, an open-source hardware project. They find that encouraging community-driven maintenance enabled the hardware to evolve to address design issues that would likely have gone unaddressed otherwise [25]. Orr emphasizes how hardware maintenance and troubleshooting practice cannot exhaustively be preempted through documentation [56]. The practical nature of hardware issues is that they are non-canonical and can only be addressed through pools of shared knowledge that continue to evolve over time [10, 12].

More recently, Subbaraman and Peek [67] argue that maintenance is a core component of using digital fabrication workflows in 3D printing communities. Like Orr, they caution against addressing maintenance needs with fully automated procedures and instead advocate for systems to be designed with maintenance in mind. This resembles Jackson’s [37] concept of “broken world thinking,” which argues for foregrounding repair and maintenance practices when considering technological progress. SplatOverflow revisits these ideas to examine how such troubleshooting could be coordinated asynchronously and how systems supporting hardware maintenance can contribute workflows to pools of practical knowledge in hardware communities.

## 4.2 Documentation Systems

Documentation for assembly instructions, usage guides, and maintenance tasks is integral to how users interact with and navigate hardware. Such documentation is distinct from the design files that define the hardware, but researchers have argued that they contribute just as significantly to its success and adoption [11, 56]. To this end, HCI researchers have examined how hardware designers can efficiently create documentation that is useful to end-users. Mariscal-Melgar et al. [48] propose a semi-automated method for generating and updating assembly documentation by leveraging data from the hardware’s CAD model. This lets designers compile new assembly instructions as designs change and keep documentation in sync with updates to the hardware. Milara et al. [50] propose software tools for makers to create documentation as they work on projects to ensure that essential design decisions, lessons from failures, and the process narrative are not lost. Tran O’Leary et al. [72] demonstrate how to create fabrication workflows in a manner that foregrounds reproducibility by others. By interweaving digitally controlled processes and physical interventions when defining a fabrication workflow, *Tandem* guides users through the manual steps critical to successfully reproducing a workflow [72].

Researchers have also explored how to make it easier for end-users to explore existing technical documentation. For instance, *MagicNeRF* [43] allows users to navigate a NeRF scene in virtual reality, with additional documentation overlaid onto the virtual scene. *ARDW* [17] is an augmented reality projection workbench that overlays documentation onto a PCB to support debugging. These projects present overlay and interaction techniques that address challenges associated with context shifting between design, documentation, and physical instantiation of hardware. This research underscores that the successful documentation, distribution, and reproduction of hardware and hardware workflows extends far beyond sharing digital design files. However, these projects demonstrate that there is significant value in incorporating connections between digital design files, technical documentation, and physical instantiations of hardware. SplatOverflow builds on this work by leveraging existing CAD model information to author, share, visualize, and index hardware troubleshooting workflows.

## 4.3 Telepresence for Expert Support

In software development [5] and document editing [57], all collaborators typically access the same source code instead of compiled

or high-level artifacts. Direct access to the collaborative artifact enables asynchronous support across the web [5].

HCI researchers examining synchronous support have proposed multiple theories and developed speculative systems for how remote experts can virtually occupy a shared task space with non-expert users to provide guidance [14, 27, 29, 54]. Telepresence solutions often try to establish a shared reference space, in which participants can communicate with one another with deictic expressions and gestures as they would if they were in person [24, 32, 38]. Such a reference space depends on both participants being able to communicate within a common context. In a survey on augmented reality systems for collocated experiences, Radu et al. [58] found that collaborators need a shared environment to ground their communication. Moreover, they find that collaborators need tools to guide attention and give instructions by referencing objects in the shared task space. Chastine et al. [16] develop a formal model of such inter-referential awareness, providing a detailed framework for how designers can develop referential systems.

Advances in augmented and virtual reality have led to work studying how experts can remotely collaborate on physical tasks [75]. Researchers have proposed systems for ‘mentoring’ tasks, where one less experienced user seeks out guidance from an expert user who views the scene remotely using mobile AR to bring the physical environment of users to remote collaboration [28, 29, 38]. Some systems let experts view the perspective of remote users in VR and communicate guidance through overlaid annotations [18, 71, 79].

Alternatively, systems also capture 3D reconstructions [36] and allow remote collaborators to interact by combining with teleoperated robots as in *VRoxy* [61] or by integrating live 360° video as Teo et al. [69] demonstrated.

In these systems, the interaction is synchronous, and the guidance provided is ephemeral, making it difficult for third parties to replay and observe at a later time. *Heimdall* [39] is an alternative approach to remote collaboration for prototyping electrical circuits featuring a bespoke scanning system. Instructors navigate around the prototype circuit and inspect its schematics through an instrumented breadboard. This enables instructors to remotely examine and debug student circuits without relying on a student to navigate around their breadboard. SplatOverflow marries insights from synchronous remote expert support systems with the flexibility and observability enabled by asynchronous troubleshooting.

## 4.4 Scanning and Spatial Rendering

To troubleshoot hardware issues, SplatOverflow must be able to capture a local user’s hardware, ideally without specialized equipment. To do this, it leverages research in scanning and scene representation techniques. Recently, there has been substantial progress in novel-view synthesis techniques that can construct a 3D scene from collections of posed 2D images [41, 51, 53]. These approaches build on COLMAP, a structure-from-motion (SfM) tool that constructs a coarse point cloud and estimates camera pose from image frames [62–64]. These tools and techniques allow SplatOverflow users to capture high-fidelity scans using commodity smartphone cameras they likely already own.

Although SplatOverflow is generally agnostic to scanning technology, our implementation leverages 3D Gaussian Splatting [41]

to capture a workspace due to its fast training time and real-time rendering capabilities. A splat represents the scene as a collection of 3D Gaussian distributions and can be rendered in real-time using traditional graphics pipelines. Notably, there is considerable progress in reducing training times and improving the visual definition of Gaussian splats [33, 44, 76]. These methods enable fast 3D scene reconstruction with accurate geometric information from limited data, lowering the overhead of capturing a scene for users.

## 5 SPLATOVERFLOW

This section describes the design of SplatOverflow. We discuss the constituent artifacts that comprise a SplatOverflow scene, including the benefits and shortcomings of each artifact. Then, we explain the construction of the SplatOverflow scenes, how it facilitates asynchronous communication using *gestures*, and how data from issues can be indexed and retrieved using a CAD model.

### 5.1 Constituent Artifacts

At a minimum, SplatOverflow requires a scan of the hardware and a CAD model that describes the hardware’s design. SplatOverflow combines these artifacts to construct a single boundary object that amplifies the benefits and minimizes the drawbacks of each constituent artifact, as well as addressing the limitations of traditional asynchronous troubleshooting artifacts described in Section 3.

**5.1.1 CAD Models.** CAD models can precisely and accurately describe the design of hardware. They also offer various viewing, selection, and manipulation tools to help make sense of hardware issues. For example, CAD models allow designers to inspect internal mechanisms that are not visible in the assembled hardware. This ability to peer through parts can help designers reason about mechanical issues without taking apart the hardware. These models can also be the basis of generating visual assembly instructions that are easy for users to follow [4, 48].

The primary drawback of CAD models is that they do not reflect the host of ways hardware can err. This is because CAD models capture the hardware in an idealized state. In practice, however, hardware (especially malfunctioning hardware) does not perfectly resemble the CAD model. This is additionally problematic as some deviations from the CAD model are perfectly acceptable, while others can introduce errors. CAD models offer the tools and interactions to visualize, explore, and reason about a hardware design but do not capture the multitude of ways that each hardware instance is unique.

**5.1.2 Scans.** Scans, however, can capture the unique details of hardware instances. Dedicated mobile scanners using stereo vision, structured light, or novel-view synthesis techniques can accurately capture the geometry of hardware at different scales. This data can be used to measure and inspect hardware as it has been built and assembled, which is essential when diagnosing what may be going *wrong* with the hardware.

The drawbacks of relying on scans are that they only capture visible geometry and do not know the semantic meaning of what they are capturing. Most scanning methods cannot safely penetrate materials to scan internal components, so scans capture only the

outer shell of the hardware geometry. Moreover, as scanning methods often do not know what they are scanning, isolating different parts in an assembly is challenging. These drawbacks are related in that they highlight how references are hard to anchor to scan data.

### 5.2 A SplatOverflow Scene

SplatOverflow scenes comprise two essential artifacts: a scan of the user’s workspace aligned and registered onto a CAD model of the hardware. Our implementation generates the scans using 3D Gaussian Splatting [41]. We chose Gaussian Splatting as it can generate high-quality scans rapidly using only user-captured video data as input. We use open-source CAD models saved as .glb files with additional data to capture the assembly constraints in the model. The 3D Gaussian Splat (referred to here as splat) is aligned and registered to the CAD model using ArUco markers that are placed on the hardware at known locations [52, 55].

The aligned artifacts support selecting and manipulating individual components on the hardware segmented using the CAD model and inspecting the physical state of the hardware as captured in the splat. SplatOverflow scenes are implemented in WebGL and run in the browser. As a result, they can be shared with anyone online, requiring no installation or OS-specific set-up. Scenes can be viewed on personal computers, mobile devices, and virtual reality headsets that support WebXR.

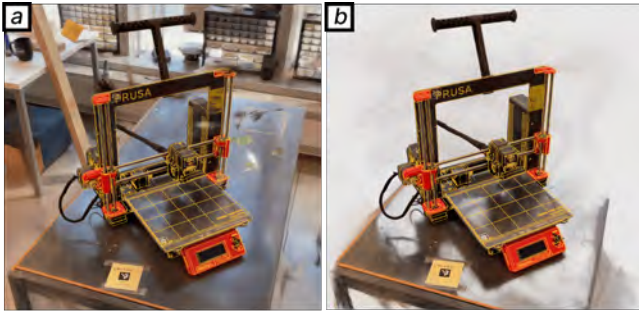
**5.2.1 Scanning a Workspace.** Scanning the hardware and surrounding workspace captures the as-built state of the hardware. This includes modifications, job configurations, or parts not traditionally included in CAD, such as wires or cables.

The input to the scan is a video of the user’s workspace captured through SplatOverflow’s mobile interface, as described in Section 2. The video is then sampled in two passes. First, at a fixed frame rate (4 FPS in our examples) and then again to more densely sample frames containing ArUco tags to generate a set of images. SplatOverflow feeds this set of images to COLMAP [62, 63] to generate an intermediate Structure-from-Motion model of the local users’ workspace. SplatOverflow uses this model to generate a splat of the recorded scene.

SplatOverflow takes roughly four minutes to generate a 3D Gaussian Splat from a video filmed in 1080p resolution and roughly two minutes from a 360p video. Generating the splat is the primary bottleneck for developing a SplatOverflow scene. Processing times could be reduced by using more efficient splat generation techniques from recent research, such as MVSGaussian [44].

**5.2.2 Registering a Gaussian Splat to a 3D CAD Model.** SplatOverflow automatically aligns and registers a splat of the local user’s hardware to the 3D CAD model in two steps using ArUco tags placed in known locations on the hardware [52]. SplatOverflow first uses the SfM model generated by COLMAP [62, 63] to compute a three-dimensional coordinate of every corner of each ArUco tag [49]. These corners are used to rescale the splat to be the same size as the CAD model. Next, SplatOverflow uses Arun’s method [6] to compute a rigid transformation going from corners in the SfM model’s coordinate space to CAD coordinate space. If there are no moving parts in the CAD assembly, this transformation yields an aligned and registered splat.





**Figure 9:** (a) An unaltered 3D Gaussian Splat as returned by our system. (b) A pruned 3D Gaussian splat is used to preserve the user’s privacy and not share background details with a community of users.

However, hardware will often have moving parts. These degrees of freedom mean that the hardware can be in one of many states, which are unlikely to match the static state of the CAD model. To address this, SplatOverflow recommends placing at least one *constraint tag* on each degree of freedom and one *grounding tag* on a stationary part of the hardware. When a CAD model is being prepared, the model’s maintainer indicates which tag ID maps to which degree of freedom. With this precondition, SplatOverflow uses the same detection method to find the centers of each *constraint tag* and computes an offset from the *grounding tag* to the *constraint tag*. This offset is then used to calculate a transformation that automatically aligns the CAD component to the position of the part in real life. Following these two steps, the hardware parts are aligned to their corresponding parts in the CAD model.

The tags used for alignment, registration, and constraint satisfaction can be applied manually onto existing hardware, placed during manufacturing, or designed into the hardware [21, 23]. In all of our examples, tags were manually placed onto existing hardware.

**5.2.3 Post Processing a Splat Using Features of the CAD Model.** Once the splat has been aligned and registered, SplatOverflow leverages information from the CAD model to post-process the splat. SplatOverflow uses a signed distance function from points in the splat to CAD meshes to identify what region of the splat contains relevant information about the hardware. By default, SplatOverflow uses this to remove the background details from a scene to preserve the local user’s privacy and share only parts of the splat that correspond to the hardware or the workspace the hardware rests on. Figure 9 shows a splat as it is generated and the same splat after SplatOverflow’s privacy filter.

**5.2.4 Localizing Video Feeds.** SplatOverflow allows users to capture dynamic details of their issue as short video clips and aligns these clips in the SplatOverflow scene using an estimate of the camera pose at each frame. This allows remote maintainers to review the perspective from which a video was shot as well as the content of the video. Figure 4 shows the video feed in the SplatOverflow scene. SplatOverflow estimates the pose of a video feed by localizing individual frames into an existing COLMAP model [62, 63].

### 5.3 Interacting with a SplatOverflow Scene

SplatOverflow offers tools to explore a scene, communicate actions, and facilitate discussion. This section describes each of these tools and the interactions they support.



**Figure 10:** (a) A sub-assembly on the Lumen v3 with occluded components visible to the user. (b) A gesture indicating that the user tightens a bolt on the sub-assembly. (c) Querying technical documentation referencing a specific sub-assembly.

**5.3.1 Selection and View Control.** SplatOverflow allows users to select parts and navigate subassemblies of the CAD model to compare the as-designed to the as-built and make sense of occluded parts of the hardware not visible in the splat. When a part from the CAD model is selected, it is rendered to the user with a yellow wireframe. This allows the user to compare details from the splat to the geometry of an individual CAD component. Moreover, the visibility of CAD parts in a subassembly can be controlled, allowing users to “peer through” sub-assemblies and visualize internal components that are not visible, as shown in Figure 10(a).

Remote users can use SplatOverflow scenes to independently explore novel viewpoints within the local user’s environment. This is enabled by the splat component of a SplatOverflow scene. Once the splat is aligned onto the CAD model, it occupies the same coordinate space and can be navigated with the same controls as a CAD environment.

**5.3.2 Gestures.** Users communicate in SplatOverflow via *gestures*. A gesture is an instruction or request that references an element in the SplatOverflow scene. Gestures are organized chronologically in an interactive timeline. The timeline elements trigger visualizations of the gesture in the SplatOverflow scene and house user discussions about the gesture.

Maintainers can use gestures to request more information from the user. This can be done by requesting video annotations or updated splats using the *request* gesture. When a request gesture is authored, a timeline element appears with a QR code. The QR code lets the user populate the request via SplatOverflow’s mobile interface and appends the requested information to the scene. Figure 5 shows an example of the request gesture.

Gestures can also communicate instructions or actions a user should take on their hardware. These gestures are all anchored to parts in the CAD model, which are highlighted whenever a gesture is selected in the timeline alongside each gesture’s animation. SplatOverflow’s *pointing* gestures indicate where on a part in CAD a user should inspect or attend to. When selected, the referenced CAD part is highlighted, and the viewport shifts to re-orient the local user. SplatOverflow’s *move* gesture indicates to local users how parts should be manipulated. When selected, the referenced CAD part animates to tween between its original position and the final position defined by the remote maintainer. Finally, SplatOverflow includes a set of action gestures for common troubleshooting

operations, such as tightening or loosening specific bolts and probing particular pads on a PCB. When selected, the gestures overlay arrows onto the scene to visualize the operation.

**5.3.3 Timeline Elements and Discussions.** Timeline elements refer to gestures authored by different users. The timeline element contains additional text to clarify what a gesture entails. Users can ask for clarification or follow up on a gesture by replying to the corresponding element in the timeline. This facilitates the asynchronous back-and-forth communication to help make sense of a given gesture.

## 5.4 Indexing Data to a CAD Model

SplatOverflow uses the CAD model to index and query data. This includes technical documentation referencing parts in the CAD model and past SplatOverflow issues. For existing technical documentation, SplatOverflow renders web pages that reference CAD components, as illustrated in Figure 10(c). Similarly, SplatOverflow can query the history of past issues to retrieve troubleshooting instructions that reference specific CAD components. This ability to feed structured data from troubleshooting issues into a database that can be queried allows SplatOverflow to become increasingly useful to hardware communities over time.

## 6 IMPLEMENTATION

Our implementation of SplatOverflow is divided into four sections: Scene Capture, Scene Generation, the SplatOverflow front-end, and the back-end Coordination Server.

The mobile capture interface runs in a browser and is written in JavaScript using the MediaStream Recording API. It is designed to be accessible from modern smartphone browsers. On the server, SplatOverflow uses a C++ 3D Gaussian Splatting implementation by MrNeRF [41] and the COLMAP package for Structure-from-Motion (SfM) to generate the scene. To align a CAD model to the scan, SplatOverflow uses the method from ArUco SfM Scale Adjustment [49] and an implementation of Arun’s method [6].

The back-end coordination server is a Flask application and SQL database that manages the data associated with a SplatOverflow scene. The SfM [62] and Gaussian Splatting [41] pipelines run on a PC equipped with an Intel i7-14700K, 32GB RAM DDR5 and Nvidia 4080S GPU with 24GB of VRAM. On average, a 60-second video takes 122 seconds to generate a scene trained on 360p footage and 250 seconds to generate a scene trained on 1080p footage.

We use Mark Kellogg’s 3D Gaussian Splatting Renderer [40] to render the 3D Gaussian Splats and develop the interface elements in Javascript, using the Three.js library [20].

## 7 DEMONSTRATIVE EXAMPLES

Figure 11 demonstrates the utility of SplatOverflow in three workflows that extend beyond troubleshooting of machines: (a) verifying assembly of the PCB in the *Open Book* e-reader by Oddly Specific Objects, (b) performing routine maintenance on the *Prusa MK3S* 3D printer, and (c) following disassembly instructions for a flat-pack bookshelf using the *Billy* bookshelf by IKEA. These workflows intend to capture a variety of scales, applications, and user roles that SplatOverflow can support.

### 7.1 Verifying the Correct Assembly of the *Open Book* E-Reader

SplatOverflow is compatible with non-mechanical CAD designs. The *Open Book* [15] e-reader is an open-source e-reader that consumers assemble on their own, with the goal of demystifying consumer electronics. Successful assembly is a function of correct and precise soldering. In this example, we show how the maintainer of the *Open Book* can help users verify correct assembly and track down potential errors that inhibit successful booting using a workflow authored and shared in SplatOverflow. Specifically, the maintainer defines a set of probe points to be checked with a multimeter that can indicate the exact probe points on the PCB the user must test, along with the expected outcome of each test.

The local user starts by creating a SplatOverflow scene to follow the assembly verification guide created by a maintainer. The *Open Book* has a 20mm tag that aligns the CAD model of the PCB generated from KiCAD [68]. In this example, we applied the tag to the PCB, but in practice, such a tag could be printed as part of the PCB’s silkscreen. Once the SplatOverflow scene is generated, the local user loads the instructions authored by the maintainer into their scene and sees them projected onto their hardware. Figure 11(a) shows the instructions rendered onto their PCB.

The local user references verification instructions loaded into their SplatOverflow scene. As shown in Figure 11(a), instructions are overlaid onto the pins the user must probe. The user follows probing instructions to check for continuity, voltage, and resistance at points across the PCB. The guide authored in a SplatOverflow scene can be recontextualized to each local user’s hardware. This contrasts most online guides, which are rendered using virtual objects or images and videos of a different hardware instance.

### 7.2 Sharing and Updating Routine Maintenance Workflows on the *Prusa MK3S* 3D Printer

Asynchronous sharing enables feedback loops, leading to improved workflows. The *Prusa MK3S* 3D Printer is a popular personal fabrication machine targeting hobbyists. Like all 3D printers and fabrication machines, the printer requires routine maintenance to ensure proper function. A typical maintenance task is tightening the belts that drive two axes of motion. Poorly tightened belts can lead to failed prints or unintended artifacts in otherwise successful prints. Learning to carry out such tasks is essential to owning and operating personal fabrication machines [66]. We illustrate how SplatOverflow can be used to share maintenance workflows and how end-user feedback can improve the quality of such workflows.

A local user sees ghosting artifacts on their print and is pointed to a SplatOverflow issue explaining how to adjust the machine’s belt tension. Figure 11(b) shows the workflow rendered in the local user’s workspace. The user follows the instructions but is unsure how much to tighten the belts. They leave a comment asking the maintainer how they know when to stop tightening the belt. The maintainer realizes that the instruction is underspecified and shares an online resource that uses a microphone to validate belt tension. Here, we show a simple example of how the open and asynchronous nature of SplatOverflow issues allows communities of users to iterate on workflows together.



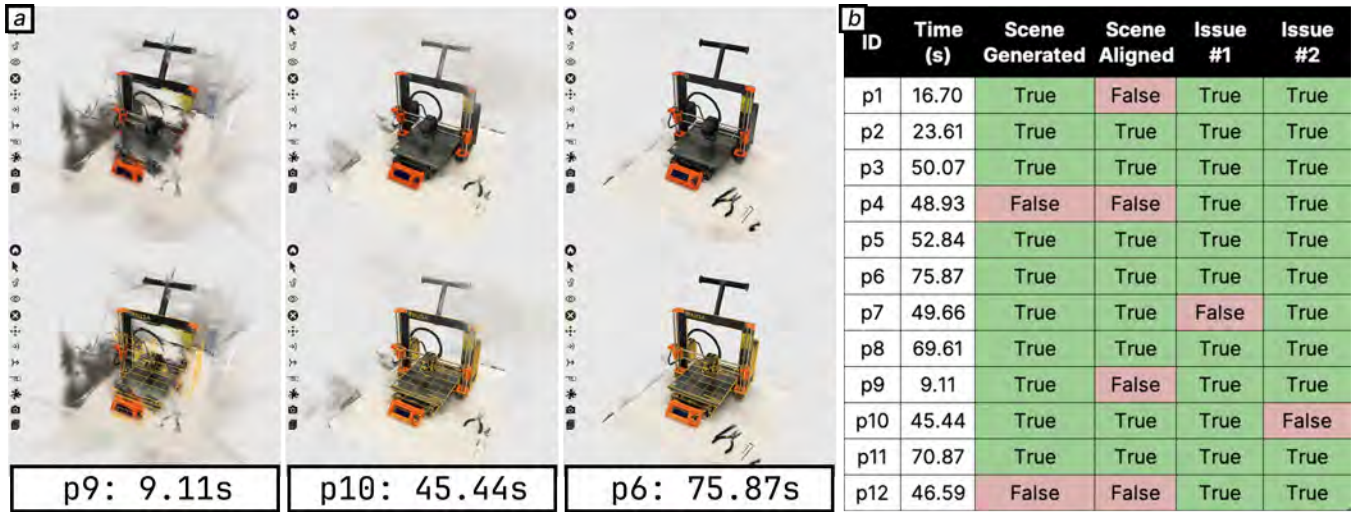


Figure 12: We conducted a usability study with 12 participants to evaluate whether they could generate a SplatOverflow scene and successfully fix the issue. (a) Sample scenes generated by study participants. (b) A table showing each participant’s outcomes in our study and the length of the recording used to generate the scene.

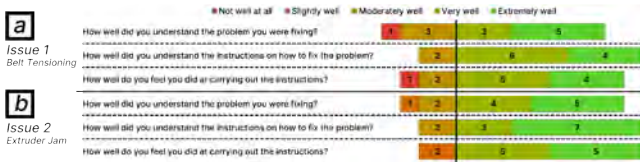


Figure 13: Participants’ responses to post-task questions regarding their confidence while troubleshooting the two issues. (a) Shows responses for Issue 1: adjusting belt tension. (b) Shows responses for Issue 2: clearing an extruder jam.

to answer questions about their experience using SplatOverflow to troubleshoot the issue. Figure 13 shows participants’ responses on a Likert scale. Overall, users found that instructions rendered in SplatOverflow were easy to follow and well-aligned with the scanned hardware.

#### 8.4 System Usability

After completing both parts of the study, participants evaluated the overall usability of the system through a survey. We use the System Usability Scale [13] and our system scored 86.25 / 100.

In addition, participants shared insights on how SplatOverflow could be expanded to assist with troubleshooting in other contexts. Participants felt that SplatOverflow would be helpful when facing issues with unfamiliar hardware: *“It is very helpful for me to use and repair, especially when I’m not familiar with the machine itself”* (Participant 1). They also compared our system to existing methods of troubleshooting: *“I was working with robots (Turtlebot 4 setup). Their documentation was too wordy, and it was hard to locate the solution from that. So, we had to post GitHub issues and wait for a response. It was sometime hard to explain them my specific problem. Of course, if there was a system that I could have scanned my problem and send them, it would have saved a lot of time”* (Participant 3).

## 9 DISCUSSION

We discuss the implementation and application of SplatOverflow. We first highlight aspects of the experience for different users and then discuss technical features and future implementation.

### 9.1 SplatOverflow Users

**9.1.1 Hardware Support for End-Users.** SplatOverflow offers a new way to collaborate on hardware. For workflows such as assembly, routine maintenance, operating instructions, and troubleshooting, SplatOverflow provides a novel way for users to receive guidance from collaborators or retrieve associated documentation. By connecting physical instances of hardware to their CAD model and associated documentation, end-users can inspect the ‘source’ of the hardware they are working with. This direct connection between the physical part and documentation provides a more direct way to navigate documentation that can exist in disparate locations.

Moreover, users can seek help through SplatOverflow, analogous to asking for help in a forum. In software, platforms such as GitHub [19] and StackOverflow [5] have been essential for helping users learn about new software, troubleshoot issues, and scale maintenance efforts to support distributed projects. By supporting asynchronous communication about physical things, SplatOverflow presents a similar workflow for hardware.

**9.1.2 Hardware Support for Maintainers.** Through SplatOverflow, remote maintainers gain insight into the local user’s workspace and the actual state of the hardware they are working on. Moreover, maintainers do not have to rely on the local user to move a camera precisely to inspect hardware. They can examine novel viewpoints independently and use gestures to guide the local user to move to the exact location.

This is a longstanding problem in collaborative workflows previously addressed with teleoperated robotic systems [39, 61]. SplatOverflow achieves a similar outcome without the need for bespoke

hardware or additional instrumentation of a space. This, paired with SplatOverflow’s web-based interface, expands the number of users who can access remote support.

*9.1.3 Reducing Barriers to Distribute Hardware.* Tech support is a core component of a viable hardware product but requires significant effort for hardware companies to scale. Asynchronous support infrastructure could reduce this effort by (a) providing communities of users the means of querying past issues and examining solutions and (b) allowing maintainers to provide support in an open format without coordinating with end-users. This approach would support maintainers, producers, and hardware communities in scaling technical support for users. To better evaluate this, we plan to engage with open-source hardware producers and community maker spaces to study longer-term deployments of SplatOverflow.

## 9.2 Extending Multi-media Workflows

Many tasks start with the assumption that the user is seeking guidance and has a screen available in front of them. For example, tools such as Interactive HTML BOM [34] allow users to correlate components and their placements visually. The tool *assumes* users will be using a screen to aid in assembly and debugging.

Similarly, YouTube and other video hosting platforms [9, 42, 73] are used extensively for step-by-step instructions in cooking, assembly, and DIY Home repair. Documentation often has directly embedded video [31, p. 121] and manufacturers such as IKEA [74] produced videos to demonstrate operations clearly. While these videos can be valuable resources, they are often expensive and time-consuming to create. As a result, video is not always the ideal choice for end-users to articulate issues or seek guidance.

Our work extends multi-media hardware support by allowing users to rapidly author and share workflows for acting upon hardware amongst themselves. With SplatOverflow, bidirectional communication becomes easier instead of being solely broadcast from experts or designers.

## 9.3 Cross-Referencing, Inspectability and Information Sharing

SplatOverflow enables the ability to link across hardware and data. This interaction mode is similar to *View Source* or Right Click to *Inspect Element* that is available in browsers [78]. These features were originally intended to help developers debug their own software [45, 59]. However, they rapidly became tools for education and exploration where “Every single web page you visited contained the code showing you how it was created. The entire internet became a library of how-to guides on programming” [70]. While we leave the specific application of SplatOverflow in education contexts to future work, we believe that the ability to inspect, discuss, and cross-reference physical and digital representations is vital to sharing hardware designs and know-how. SplatOverflow provides a set of primitives to support a known challenge for commons-based peer production [7, 8] of physical systems [66], primarily that of scalable instruction, support, and collaboration.

## 9.4 Technical Features

After implementing and testing SplatOverflow in a variety of contexts, we distilled a set of features to support it as we move towards a full deployment of the system.

*9.4.1 Guided Splat Capture.* In our evaluation, some user captures failed due to insufficient data to align the model. Given that proper alignment and registration are essential for creating a successful SplatOverflow scene, we plan to improve feedback in SplatOverflow’s mobile interface. The capture process could also be enhanced by allowing users to specify the hardware they intend to scan. With this information, the interface can guide users through capturing enough frames to align the splat to the CAD model and indicate which *grounding* and *constraint* tags require more footage.

*9.4.2 Automatically Preparing CAD Models.* To make a CAD model compatible with SplatOverflow, the producers of the hardware need to determine optimal locations to place the *grounding* and *constraint* tags. The choice of location for these tags can affect how difficult it is to align the captured splat to the CAD model. To support onboarding hardware into SplatOverflow, we plan to automate the tag placement process to optimize for visibility using a process similar to *BrightMarker* [22].

*9.4.3 IP of CAD models.* In some cases, CAD models are highly protected by hardware maintainers, who may not want to share these files with users. Currently, SplatOverflow uses a mesh representation of the CAD model and only requires information about the position of alignment tags on the model. For maintainers concerned about sharing mesh models, we plan to build an import pipeline that strips only the relevant data for our workflows without saving the actual CAD model on our server.

*9.4.4 Mixed Reality.* SplatOverflow is built in the browser and could be extended to fully immersive Augmented and Virtual Reality contexts by leveraging the WebXR Devices API [47]. While our implementation focuses on screen-based interaction, future work may examine how authoring SplatOverflow gestures could be extended to mixed reality.

## 10 LIMITATIONS

This section details some limitations we have observed over the course of designing and implementing SplatOverflow.

### 10.1 Acquiring CAD Models

SplatOverflow requires access to the hardware’s CAD model to populate the scene. For SplatOverflow scenes to be useful, the CAD models must also be *complete*. CAD models like this can be challenging to obtain outside of open-source hardware projects, which can limit the adoption of SplatOverflow. Moreover, even when CAD files are available, the structure of sub-assemblies, parts, and components is not standardized. As a result, preparing CAD files for use in SplatOverflow can require coercing the structure of the CAD assembly into a compatible shape.

## 10.2 Structure References in Technical Documentation

Another limitation of SplatOverflow lies in its ability to reference a corpus of existing technical documentation. With the Lumen v3, each technical documentation section contained links referencing the related parts. This greatly simplified populating the documentation interface with instructions to search through. However, not all hardware will share technical documentation in the same structured fields as Opulo does with Lumen v3. As a result, maintainers must post-process existing technical documentation to add the relevant structured data indicating which CAD components are referenced.

## 10.3 Errors in Tag Placement

In all our examples, *grounding* and *constraint* tags were manually placed on the hardware. In doing so, we realized that while getting precisely aligned CAD models using manually placed tags is possible, the process is also prone to errors. Specifically, if a tag is misplaced, the amount SplatOverflow's alignment deviates increases proportionate to the distance from the tag. As a result, placing SplatOverflow tags manually is a challenging, error-prone task and may not be approachable for novice users.

## 11 CONCLUSION

We introduced SplatOverflow, a novel workflow for asynchronous hardware troubleshooting. SplatOverflow constructs a boundary object that can capture the physical details of a user's hardware and communicate instructions as actions on their hardware. We demonstrate SplatOverflow through a series of examples with different kinds of hardware. We show how these workflows can support complex troubleshooting workflows that require multiple exchanges between users and physical manipulation of the hardware. Moreover, we illustrate how a SplatOverflow scene can index technical knowledge about hardware to the CAD model. As a result, local users searching multiple disparate sources for relevant information can now access documentation and past solutions directly through the physical hardware.

We plan to distribute and deploy SplatOverflow with a broader user base through hardware maintainers and study our proposed forms of collaboration *in the wild*. Besides direct utility for hardware users and maintainers, we believe tools for communicating about physical hardware can reduce barriers to entry for smaller producers distributing niche hardware and supporting community development amongst their users.

## ACKNOWLEDGMENTS

We thank the Digital Life Initiative at Cornell Tech for supporting this work through a doctoral fellowship. We thank the Bowers CIS Undergraduate Research Experience for supporting this work through their summer research program. We thank Roy Zunder for helping review this manuscript. Finally, we thank Joey Castillo, Frank Bu, and Stephen Hawes for participating in preliminary discussions that helped motivate this work.

## REFERENCES

- [1] Prusa 3D. 2023. Adjusting the belt tension on the Original Prusa MK4 - Belt Tuner App. [https://youtu.be/oeq2MVxE\\_H8?feature=shared&t=56](https://youtu.be/oeq2MVxE_H8?feature=shared&t=56)

- [2] Mark S. Ackerman, Juri Dachtera, Volkmar Pipek, and Volker Wulf. 2013. Sharing Knowledge and Expertise: The CSCW View of Knowledge Management. *Computer Supported Cooperative Work (CSCW)* 22, 4-6 (Aug. 2013), 531–573. <https://doi.org/10.1007/s10606-013-9192-8>
- [3] John R Ackermann. 2008. Toward Open Source Hardware. (2008).
- [4] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. 2003. Designing Effective Step-by-Step Assembly Instructions. *ACM Trans. Graph.* 22, 3 (jul 2003), 828–837. <https://doi.org/10.1145/882262.882352>
- [5] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2012. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Beijing China, 850–858. <https://doi.org/10.1145/2339530.2339665>
- [6] K. S. Arun, T. S. Huang, and S. D. Blostein. 1987. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9, 5 (Sept. 1987), 698–700. <https://doi.org/10.1109/TPAMI.1987.4767965>
- [7] Yochai Benkler. 2002. Coase's penguin, or, linux and "the nature of the firm". *Yale law journal* (2002), 369–446.
- [8] Yochai Benkler and Helen Nissenbaum. 2006. Commons-based peer production and virtue. *Journal of political philosophy* 14, 4 (2006).
- [9] Aditi Bhatia. 2018. Interdiscursive performance in digital professions: The case of YouTube tutorials. *Journal of Pragmatics* 124 (2018), 106–120.
- [10] Daniel G. Bobrow and Jack Whalen. 2002. Community Knowledge Sharing in Practice: The Eureka Story. *Reflections: The SoL Journal* 4, 2 (Dec. 2002), 47–59. <https://doi.org/10.1162/152417302762251336>
- [11] Jérémy Bonvoisin, Robert Mies, Jean-François Boujut, and Rainer Stark. 2017. What is the "Source" of Open Source Hardware? *Journal of Open Hardware* 1, 1 (Sept. 2017), 5. <https://doi.org/10.5334/joh.7>
- [12] Stewart Brand. 2025. *Maintenance: Of everything: Part one*. Stripe Press.
- [13] John Brooke. [n. d.]. SUS - A quick and dirty usability scale. ([n. d.]).
- [14] William Buxton. 1992. Telepresence: Integrating shared task and person spaces. In *Proceedings of graphics interface*, Vol. 92. Canadian Information Processing Society Toronto, Canada, 123–129.
- [15] Joey Castillo. [n. d.]. Open Book Project. <https://github.com/joeycastillo/The-Open-Book/tree/reboot#state-of-the-book>
- [16] Jeffrey W. Chastine, Ying Zhu, and Jon A. Preston. 2006. A Framework for Inter-referential Awareness in Collaborative Environments. In *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. 1–5. <https://doi.org/10.1109/COLCOM.2006.361859>
- [17] Ishan Chatterjee, Tadeusz Pforte, Aspen Tng, Farshid Salemi Parizi, Chaoran Chen, and Shwetak Patel. 2022. ARDW: An Augmented Reality Workbench for Printed Circuit Board Debugging. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. ACM, Bend OR USA, 1–16. <https://doi.org/10.1145/3526113.3545684>
- [18] Lei Chen, Yilin Liu, Yue Li, Lingyun Yu, BoYu Gao, Maurizio Caon, Yong Yue, and Hai-Ning Liang. 2021. Effect of Visual Cues on Pointing Tasks in Co-Located Augmented Reality Collaboration. In *Proceedings of the 2021 ACM Symposium on Spatial User Interaction (Virtual Event, USA) (SUI '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 12 pages. <https://doi.org/10.1145/3485279.3485297>
- [19] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 1277–1286.
- [20] Brian Danchilla. 2012. Three.js Framework. In *Beginning WebGL for HTML5*. Apress, Berkeley, CA, 173–203. [https://doi.org/10.1007/978-1-4302-3997-0\\_7](https://doi.org/10.1007/978-1-4302-3997-0_7)
- [21] Mustafa Doga Dogan, Vivian Hsinyueh Chan, Richard Qi, Grace Tang, Thijs Roumen, and Stefanie Mueller. 2023. StructCode: Leveraging Fabrication Artifacts to Store Data in Laser-Cut Objects. In *Proceedings of the 8th ACM Symposium on Computational Fabrication (SCF '23)*. Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3623263.3623353>
- [22] Mustafa Doga Dogan, Raul Garcia-Martin, Patrick William Haertel, Jamison John O'Keefe, Ahmad Taka, Akarsh Aurora, Raul Sanchez-Reillo, and Stefanie Mueller. 2023. BrightMarker: 3D Printed Fluorescent Markers for Object Tracking. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, San Francisco CA USA, 1–13. <https://doi.org/10.1145/3586183.3606758>
- [23] Mustafa Doga Dogan, Ahmad Taka, Michael Lu, Yunyi Zhu, Akshat Kumar, Aakar Gupta, and Stefanie Mueller. 2022. InfraredTags: Embedding Invisible AR Markers and Barcodes Using Low-Cost, Infrared-Based 3D Printing and Imaging Tools. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/3491102.3501951> event-place: New Orleans, LA, USA.
- [24] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work - CSCW '92*. ACM Press, Toronto, Ontario, Canada, 107–114. <https://doi.org/10.1145/143457.143468>

- [25] Kellie Dunn, Cynthia Feng, and Nadya Peek. 2023. Jubilee: A Case Study of Distributed Manufacturing in an Open Source Hardware Project. *Journal of Open Hardware* 7, 1 (May 2023), 4. <https://doi.org/10.5334/joh.51>
- [26] Nadia Eghbal. 2020. *Working in public: the making and maintenance of open source software*. Stripe Press.
- [27] Susan R. Fussell, Robert E. Kraut, and Jane Siegel. 2000. Coordination of Communication: Effects of Shared Visual Context on Collaborative Work. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (Philadelphia, Pennsylvania, USA) (CSCW '00). Association for Computing Machinery, New York, NY, USA, 21–30. <https://doi.org/10.1145/358916.358947>
- [28] Susan R. Fussell, Robert E. Kraut, and Jane Siegel. 2000. Coordination of communication: effects of shared visual context on collaborative work. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, Philadelphia Pennsylvania USA, 21–30. <https://doi.org/10.1145/358916.358947>
- [29] Steffen Gauglitz, Cha Lee, Matthew Turk, and Tobias Höllerer. 2012. Integrating the Physical Environment into Mobile Remote Collaboration. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 241–250. <https://doi.org/10.1145/2371574.2371610> event-place: San Francisco, California, USA.
- [30] Steffen Gauglitz, Benjamin Nuernberger, Matthew Turk, and Tobias Höllerer. 2014. World-stabilized annotations and virtual scene navigation for remote collaboration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, Honolulu Hawaii USA, 449–459. <https://doi.org/10.1145/2642918.2647372>
- [31] Alicia Gibb. 2015. *Building open source hardware: DIY manufacturing for hackers and makers*. Pearson Education.
- [32] Carl Gutwin and Saul Greenberg. 2002. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3–4 (Sept. 2002), 411–446. <https://doi.org/10.1023/A:1021271517844>
- [33] Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. (2023). <https://doi.org/10.48550/ARXIV.2311.12775> Publisher: [object Object] Version Number: 3.
- [34] Hackaday. 2018. Interactive KiCAD BOMs Make Hand Assembly A Breeze. <https://hackaday.com/2018/09/04/interactive-kicad-boms-make-hand-assembly-a-breeze/>
- [35] Steve Hodges and Nicholas Chen. 2019. Long Tail Hardware: Turning Device Concepts Into Viable Low Volume Products. *IEEE Pervasive Computing* 18, 4 (Oct. 2019), 51–59. <https://doi.org/10.1109/MPRV.2019.2947966> Conference Name: IEEE Pervasive Computing.
- [36] Shahram Izadi, David Kim, Otmarr Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, Santa Barbara California USA, 559–568. <https://doi.org/10.1145/2047196.2047270>
- [37] Steven J. Jackson. 2014. Rethinking Repair. In *Media Technologies: Essays on Communication, Materiality, and Society*, Tarleton Gillespie, Pablo J. Boczkowski, and Kirsten A. Foot (Eds.). The MIT Press, 0. <https://doi.org/10.7551/mitpress/9780262525374.003.0011>
- [38] Janet G Johnson, Danilo Gasques, Tommy Sharkey, Evan Schmitz, and Nadir Weibel. 2021. Do You Really Need to Know Where “That” Is? Enhancing Support for Referencing in Collaborative Mixed Reality Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 514, 14 pages. <https://doi.org/10.1145/3411764.3445246>
- [39] Mitchell Karchemsky, J.D. Zamfirescu-Pereira, Kuan-Ju Wu, François Guimbretière, and Bjoern Hartmann. 2019. Heimdall: A Remotely Controlled Inspection Workbench For Debugging Microcontroller Projects. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) (CHI '19). Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3290605.3300728>
- [40] Mark Kellog. 2024. 3D Gaussian splatting for Three.js. <https://github.com/mkkellogg/GaussianSplats3D>
- [41] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (Aug. 2023), 1–14. <https://doi.org/10.1145/3592433>
- [42] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J Guo, Robert C Miller, and Krzysztof Z Gajos. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 4017–4026.
- [43] Ke Li, Susanne Schmidt, Tim Rolf, Reinhard Bacher, Wim Leemans, and Frank Steinicke. 2023. Magic NeRF Lens: Interactive Fusion of Neural Radiance Fields for Virtual Facility Inspection. <https://doi.org/10.48550/arXiv.2307.09860> [cs].
- [44] Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. 2024. MVSGaussian: Fast Generalizable Gaussian Splatting Reconstruction from Multi-View Stereo. <http://arxiv.org/abs/2405.12218> arXiv:2405.12218.
- [45] Chandan Luthra and Deepak Mittal. 2010. *Firebug 1.5: Editing, Debugging, and Monitoring Web Pages*. Packt Publishing.
- [46] Wayne G. Lutters and Mark S. Ackerman. 2007. Beyond Boundary Objects: Collaborative Reuse in Aircraft Technical Support. *Computer Supported Cooperative Work (CSCW)* 16, 3 (June 2007), 341–372. <https://doi.org/10.1007/s10606-006-9036-x>
- [47] Blair MacIntyre and Trevor F Smith. 2018. Thoughts on the Future of WebXR and the Immersive Web. In *2018 IEEE international symposium on mixed and augmented reality adjunct (ISMAR-Adjunct)*. IEEE, 338–342.
- [48] J. C. Mariscal-Melgar, Pieter Hijma, Manuel Moritz, and Tobias Redlich. 2023. Semi-Automatic Generation of Assembly Instructions for Open Source Hardware. *Journal of Open Hardware* 7, 1 (Aug. 2023), 6. <https://doi.org/10.5334/joh.56>
- [49] Lukas Meyer. 2023. Aruco Scale factor Estimation for COLMAP. <https://pypi.org/project/aruco-estimator/>
- [50] Iván Sánchez Milara, Georgi V. Georgiev, Jani Ylioja, Onnur Özürü, and Jukka Riekkki. 2019. “Documentation-while-doing”: a documentation tool for Fab Lab environments. *The Design Journal* 22, sup1 (April 2019), 2019–2030. <https://doi.org/10.1080/14606925.2019.1594926>
- [51] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. <http://arxiv.org/abs/2003.08934> arXiv:2003.08934 [cs].
- [52] Rafael Munoz-Salinas. 2012. Aruco: a minimal library for augmented reality applications based on opencv. *Universidad de Córdoba* 386 (2012).
- [53] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant NGP. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–15. <https://doi.org/10.1145/3528223.3530127>
- [54] Ohan Oda, Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. 2015. Virtual Replicas for Remote Assistance in Virtual and Augmented Reality. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) (UIST '15). Association for Computing Machinery, New York, NY, USA, 405–415. <https://doi.org/10.1145/2807442.2807497>
- [55] Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, Shanghai, China, 3400–3407. <https://doi.org/10.1109/ICRA.2011.5979561>
- [56] JULIAN E. ORR. 1996. *Talking about Machines: An Ethnography of a Modern Job*. Cornell University Press. <http://www.jstor.org/stable/10.7591/j.ctt1hhfnkz>
- [57] Ilona R Posner, Ronald M Baecker, and M Mantei. 1993. How people write together. In *Proceedings of the Hawaii International Conference on System Sciences*, Vol. 25. IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, 127–127.
- [58] Iulian Rade, Tugce Joy, Yiran Bowman, Ian Bott, and Bertrand Schneider. 2021. A Survey of Needs and Features for Augmented Reality Collaborations in Collocated Spaces. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 169 (apr 2021), 21 pages. <https://doi.org/10.1145/34449243>
- [59] Mike Ratcliffe. 2013. The History of Firebug. <https://flailingmonkey.com/the-history-of-firebug>
- [60] LA 3D Printer Repair. 2019. Prusa MK3S fixing stuck filament or bad unload. <https://www.youtube.com/watch?v=i5xnAQ5dHV8>
- [61] Mose Sakashita, Hyunju Kim, Brandon Woodard, Ruidong Zhang, and François Guimbretière. 2023. VRoxy: Wide-Area Collaboration From an Office Using a VR-Driven Robotic Proxy. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, San Francisco CA USA, 1–13. <https://doi.org/10.1145/3586183.3606743>
- [62] Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [63] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.
- [64] Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*. ACM Press, Boston, Massachusetts, 835. <https://doi.org/10.1145/1179352.1141964>
- [65] Susan Leigh Star and James R. Griesemer. [n. d.]. Institutional Ecology, ‘Translations’ and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907–39. <https://doi.org/10.1177/030631289019003001>
- [66] Blair Subbaraman and Nadya Peek. 2023. 3D Printers Don’t Fix Themselves: How Maintenance is Part of Digital Fabrication. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*. ACM, Pittsburgh PA USA, 2050–2065. <https://doi.org/10.1145/3563657.3595991>
- [67] Blair Subbaraman and Nadya Peek. 2023. 3D Printers Don’t Fix Themselves: How Maintenance is Part of Digital Fabrication. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (DIS '23). Association for Computing Machinery, New York, NY, USA, 2050–2065. <https://doi.org/10.1145/3563657.3595991>
- [68] KiCad Development Team. [n. d.]. KiCad EDA - Schematic Capture & PCB Design Software. <https://www.kicad.org/>
- [69] Theophilus Teo, Louise Lawrence, Gun A. Lee, Mark Billinghurst, and Matt Adcock. 2019. Mixed Reality Remote Collaboration Combining 360 Video and 3D

- Reconstruction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) (CHI '19). Association for Computing Machinery, New York, NY, USA, 1–14. <https://doi.org/10.1145/3290605.3300431>
- [70] Clive Thompson. 2020. *Coders: The making of a new tribe and the remaking of the world*. Penguin.
- [71] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Bjoern Hartmann. 2020. TransciVR: Bridging Asymmetrical Communication Between VR Users and External Collaborators. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (UIST '20). Association for Computing Machinery, New York, NY, USA, 182–195. <https://doi.org/10.1145/3379337.3415827>
- [72] Jasper Tran O'Leary, Thrisha Ramesh, Octi Zhang, and Nadya Peek. 2024. Tandem: Reproducible Digital Fabrication Workflows as Multimodal Programs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–16. <https://doi.org/10.1145/3613904.3642751>
- [73] Sonja Utz and Lara N Wolfers. 2022. How-to videos on YouTube: The role of the instructor. *Information, Communication & Society* 25, 7 (2022), 959–974.
- [74] Keith Wagstaff. 2012. IKEA Starts 'How to Build' YouTube Channel to Help Frustrated Customers | TIME.com. <https://techland.time.com/2012/02/24/ikea-starts-how-to-build-youtube-channel-to-help-frustrated-customers/>
- [75] Peng Wang, Xiaoliang Bai, Mark Billingham, Shusheng Zhang, Xiangyu Zhang, Shuxia Wang, Weiping He, Yuxiang Yan, and Hongyu Ji. 2021. AR/MR remote collaboration on physical tasks: A review. *Robotics and Computer-Integrated Manufacturing* 72 (2021), 102071.
- [76] Shuzhe Wang, Vincent Leroy, Johann Cabon, Boris Chidlovskii, and Jerome Revaud. [n. d.]. DUS3R: Geometric 3D Vision Made Easy. ([n. d.]).
- [77] Yutaka Yamauchi, Makoto Yokozawa, Takeshi Shinohara, and Toru Ishida. 2000. Collaboration with Lean Media: how open-source software succeeds. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, Philadelphia Pennsylvania USA, 329–338. <https://doi.org/10.1145/358916.359004>
- [78] Mykyta Yevstifeyev. 2011. *The 'view-source' URI Scheme*. Internet-Draft draft-yevstifeyev-view-source-uri-01. Internet Engineering Task Force. <https://datatracker.ietf.org/doc/draft-yevstifeyev-view-source-uri/01/> Work in Progress.
- [79] Kevin Yu, Ulrich Eck, Frieder Pankratz, Marc Lazarovici, Dirk Wilhelm, and Nassir Navab. 2022. Duplicated reality for co-located augmented reality collaboration. *IEEE Transactions on Visualization and Computer Graphics* 28, 5 (2022), 2190–2200.