Body LayARs: A Toolkit for Body-Based Augmented Reality

Henning Pohl
Henning@di.ku.dk
University of Copenhagen
Copenhagen, Denmark

Tor-Salve Dalsgaard
torsalve@di.ku.dk
University of Copenhagen
Copenhagen, Denmark

Vesa Krasniqi
wqm875@alumni.ku.dk
University of Copenhagen
Copenhagen, Denmark

Kasper Hornbæk
kash@di.ku.dk
University of Copenhagen
Copenhagen, Denmark

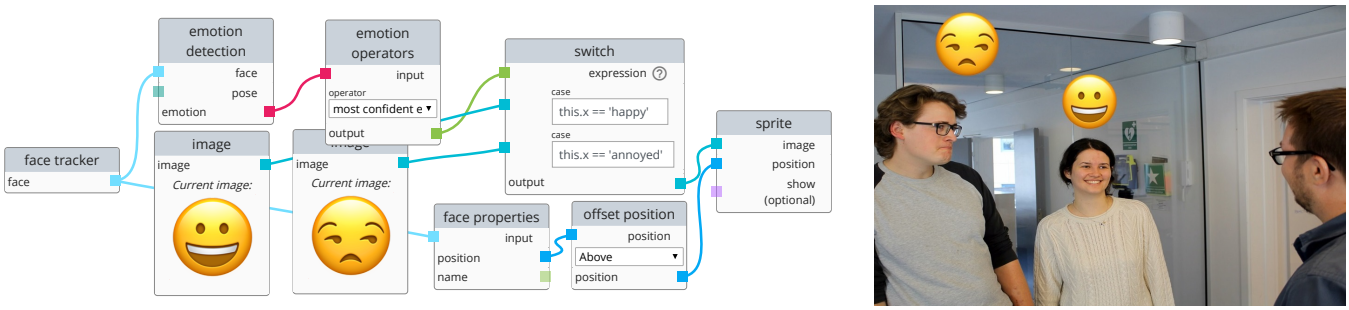

Figure 1: Body LayARs is a toolkit for prototyping body-based AR applications. It offers node-based visual programming, but can also be customized and extended via JavaScript. Applications can be started on connected devices and edited while running. Shown on the right is the output of the application to the left, which tracks faces, detects emotions, and visualizes them as floating emoji. The output was generated with a photo as input to illustrate future AR device's fidelity.

ABSTRACT

Technological advances are enabling a new class of augmented reality (AR) applications that use bodies as substrates for input and output. In contrast to sensing and augmenting objects, body-based AR applications track people around the user and layer information on them. However, prototyping such applications is complex, time-consuming, and cumbersome, due to a lack of easily accessible tooling and infrastructure. We present Body LayARs, a toolkit for fast development of body-based AR prototypes. Instead of directly programming for a device, Body LayARs provides an extensible graphical programming environment with a device-independent runtime abstraction. We focus on face-based experiences for headset AR, and show how Body LayARs makes a range of body-based AR applications fast and easy to prototype.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; *User interface toolkits*; *Mobile devices*; • **Software and its engineering** → *Software prototyping*.

KEYWORDS

Augmented reality, toolkit, body-based augmentation

VRST '20, November 1–4, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-7619-8/20/11…$15.00
https://doi.org/10.1145/3385956.3418946

ACM Reference Format:
Henning Pohl, Tor-Salve Dalsgaard, Vesa Krasniqi, and Kasper Hornbæk. 2020. Body LayARs: A Toolkit for Body-Based Augmented Reality. In *26th ACM Symposium on Virtual Reality Software and Technology (VRST '20), November 1–4, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3385956.3418946

1 INTRODUCTION

Augmented reality (AR) promises ubiquitous enhancement of the world. While this commonly focuses on objects and spaces, AR also can be used to layer information on *people*. Consider, for example, an AR application that tracks people around you, determines their emotional state, and shows this information on their faces. As we start *"wearing"* AR [39], this technology has the potential to impact interpersonal communication [40] and interaction. We call this kind of AR "body-based", and define it as follows:

> **Body-Based AR** is augmented reality that uses bodies as substrate, deriving information from and layering output on the user themselves and others. Instead of interacting with virtual objects, the focus is on augmenting interaction with people.

Exploration of body-based AR is hard because of a lack of infrastructure [10]. In general, prototyping for AR can be challenging [5] and this is exacerbated in body-based AR. For example, while there is mature support in AR tooling for marker tracking, occlusion detection, and rendering, support for tracking and augmentation of people is less sophisticated. When building body-based AR applications, designers and developers could, for example, be interested in who is around the user, how they behave, or how they move. Tracking data can be transformed and aggregated to derive measures, such as participants' share of talking time during a meeting or the path somebody moved along. Subsequently, this data needs to be visualized, such as with annotations around or overlays on people.

With Body LayARs, we present an open-source toolkit that facilitates the creation of body-based AR prototypes. Users of the toolkit get access to tracking information on nearby people and can link it with outputs to create applications. The web-based visual programming environment enables quick iteration and experimentation, as well as easy collaboration. We provide a large set of built-in capabilities, but users can also extend the toolkit to add functionality or target new devices. Development in Body LayARs is device independent and allows for quick execution on any connected device that implements the Body LayARs runtime.

Figure 1 shows an example of Body LayARs in use. The application here is meant to support users who have trouble recognizing facial expressions of emotion (often impaired in persons with Alzheimer's disease [17]). With Body LayARs, developers can easily track faces, get data on the corresponding emotions, and surface this to the user in a convenient way. In this example, emoji sprites are used to adorn each tracked person and provide additional emotion cues to the user.

In summary, our contributions are:

- a description of body-based AR
- a toolkit for easy prototyping of body-based AR apps
- a runtime for the Microsoft HoloLens to run these apps
- a demonstration of the utility and expressiveness of the toolkit through a set of example applications.

2 DESCRIBING BODY-BASED AR

Historically, AR has been focused on object-based interactions, such as in gaming, assembly, or training. In contrast, the body has always played a role in more performative AR experiences[1]. Julie Martin's *Dancing in Cyberspace* [24] is an early example, where acrobats interacted on stage with virtual objects. Another example is *DanceSpace* [47], where music and graphics are generated based on dancers' movements. Yet, such experiences have mostly been restricted to instrumented rooms (e.g., with multi-camera coverage for marker tracking). Mobile AR experiences have not generally had information on bodies. Today, with the development of more powerful computer vision methods, *bodies* are becoming fully available as material to use in AR. In this section, we describe the technologies that body-based AR is building on and provide examples of use.

2.1 Recent Advances in Body Tracking

Tracking is a core requirement for AR; in body-based AR, this means tracking of people. Classic techniques like marker-based tracking [25] are not suitable for this and thus a different set of methods is required. In addition to positional tracking, body-based AR will often also require information on the *state*, *actions*, and *identity* of others as well as users themselves.

Image-based techniques for detecting faces are mature and have been extended to full bodies. This spans from detecting the presence of a single body to full pose tracking of multiple bodies. For example, the *OpenPose* project enables pose tracking of any number of people [7]. Other examples are *HoloPose* [15], *DensePose* [4], *VNect* [32], *PoseNet* [35, 36], or *SMPL-X* [37]. They differ in the fidelity of the tracked skeleton, whether they provide a 2D or 3D skeleton, or a full body mesh, and whether they work with one or more people. Most also provide face tracking.

2.1.1 State. Once bodies are detected, additional properties of them can be inferred. For example, facial expressions can be derived from tracked faces. An AR application might detect if a person in front of the user is smiling, winking, or frowning. Tracking of facial expressions can also be used to infer the emotional state of others. A recent survey by Mehta et al. [31] detailed many techniques for this and demonstrated emotion detection with a HoloLens camera. Similar to identification, emotion can also be recognized based on audio features of speech [6].

Another set of properties can be derived from pose data. For example, such data can be used to determine whether a teacher has a 'confident' stance [43]. Some other possibilities are slouching or sitting detection, whether somebody is within an area of interest, or whether two people are sitting close to each other.

2.1.2 Action. Many properties of tracked bodies are time related. For example, dynamic gestures and movements, such as waving, nodding, shaking, and their properties, such as walking speed. Instead of just detecting movements, recent work also has shown the prediction of short-term future motions [21, 51].

There is also active research in detection of higher-level actions from video. For example, HMDB51 [23] and UCF101 [46]—two common video datasets used for this research—contain actions such as eating, brushing teeth, or shaking hands. These actions can already be recognized well from video [26] and, given further improvements, at some point likely will also work in realtime on AR devices.

2.1.3 Identity. With tracked bodies, their identity can be important additional information. An established way to do this is by using facial data. Face detection and tracking comes built into many platforms and is fast enough for realtime processing of camera data. However, people are also identifiable through properties such as their gait [20], voice [42], and overall look [2].

2.2 Body-Based Output

In body-based AR, bodies should not just be tracked, but it should also be possible to have output relative to or added to a body. With pose information, rendering content around a body is straightforward. For example, instead of rendering a model on top of a tracking marker, the model could be rendered above a person's head.

However, body-based AR also brings in new ways to visualize information in AR. A popular body-based visualization are face filters (Instagram) and lenses (Snapchat). Both layer visual content on top of and around faces, such as adding animal ears, fake glasses, or artificial makeup. These effects can be dynamic and, for example, also react to head pose and facial expressions. The term 'filter' is also used for image-based effects. For example, the view can be altered to appear like a film noir, or color graded to be more vivid. With machine learning methods, more elaborate manipulations of people's faces in images have become feasible. An example of this are generated artificial hair colors [38] and makeup [19]. A common goal behind such methods is beautification [9, 27].

[1] See http://www.augmentedperformance.com/ for a sample selection

Table 1: Comparison of common toolkits and SDKs with respect to their suitability for body-based AR prototyping.

Toolkit/API	Target	Availability	Development	Body Tracking	Body-Based Augmentation§
ARCore	📱📱	free	compiled code	1 face†	n/a
ARKit	📱📱	free	compiled code	1–3 faces† + 1 person	n/a
Vuforia	📱📱👓	$	compiled code	n/a	n/a
Maxst	📱📱👓	$	compiled code	n/a	n/a
EasyAR	📱📱💻	$	compiled code	n/a	n/a
ARToolKit	💻	free	compiled code	n/a	n/a
Wikitude	📱📱👓	$	compiled code + GUI	n/a	n/a
Torch	📱📱	$	GUI	n/a	n/a
ZapWorks	📱📱	$	GUI	1 face†	face paint
HoloJS	👓	free	scripting	n/a	n/a
MagicScript	👓	free	scripting	n/a	n/a
buildwagon	👓	$	scripting	n/a	n/a
DeepAR	⚙	$	compiled code + GUI	1 face‡	"face filters, lenses, and masks"
Lens Studio	⚙*	free	GUI	multiple faces	face lenses
Xzimg	⚙	$	compiled code	faces + emotions	n/a
Spark AR	⚙*	free	GUI	multiple faces and hands	face masks, filters, and "people effects"
Face AR	⚙	$	compiled code	multiple faces	filters, makeup, lenses, beautification
SentiMask	⚙	$	compiled code	faces + attributes (e.g., age, beard)	n/a
visage	⚙	$	compiled code	faces + attributes (gender, age, emotion)	n/a
Makeup AR-tist	⚙	$	compiled code	faces	virtual makeup

* For use in instagram/snapchat. Does not allow development of stand-alone applications. † Only works with front camera and hence cannot be used to track others. ‡ Default, multi-face support is only available from company on individual request. § We only consider "out-of-the-box" support. Developers are commonly able to render custom content. 📱 = phone, 📱 = tablet, 👓 = glasses, and ⚙ = library.

2.3 Examples of Existing Body-Based AR

There are a number of existing systems within the space of body-based AR. A common use of body-based AR is overlays of anatomical and medical data, such as in physiotherapy education [11, 18]. Similarly, the *AnatOnMe* system demonstrated how such visualization could be used to improve doctor-patient communication [34]. *Labella* is designed to augment and to promote self-discovery of the user's vagina [3].

Instead of anatomical data, the *LightGuide* system overlays directional information on the user's body in order to guide them through movements [45]. In *LumiWatch* a graphical user interface is projected on the user's arm [49]. Visual output that is linked to the body is also enabled by the *MultiFi* system, which extends the screen space of a user's wearables [14].

Saquib et al. built a presentation system that allows for flexible coupling of AR content to bodies and movement [44]. This allows users to, for example, attach icons to their hand and then gesture to switch between states. Performances, such as guitar playing, can also be augmented with spatially coherent visual effects.

Body-based AR has particular promise where it augments human-human interaction. For example, the *LittleHelper* system supports users with autism during job interviews [50]. One component of this system tracks the face of the interviewer and guides the user back to it, should they be looking away. *Superpower Glass* [8] and *Brain Power System* [28] also aim to support people with autism. Both systems are designed for therapy support and share modes aimed at training emotion recognition. Here the systems detect the emotional state of a person the user is interacting with. This information is then either surfaced to the user or used to quiz them—in either case in order to help them train their own emotion recognition abilities.

2.4 Developing Body-Based AR Applications

There are many tools available to prototype and develop AR applications. On top of the general challenges in prototyping AR, this section shows that the support for body-based AR is scarce.

As noted by Ashtari et al. in a recent paper, the entry barriers for AR development are high [5]. Domain experts do not commonly have the skillset to develop AR projects "from scratch." This can be especially daunting in the more technical parts of an AR application. Ashtari et al. cited a participant who remarked that they had no idea how computer vision works and consider those parts a "black box." Unfortunately, this exacerbates the challenges when developing body-based AR, as face and body tracking commonly is not available as a component out of the box.

There are several AR prototyping tools aimed at non-developer audiences. For example, the *DART* toolkit, was built for designers [29]. Non-programmers were also targeted by Güven and Feiner with *MARS* [16]. Recently, there has also been more work on phone-based AR prototyping. For example, *ProtoAR* enables creation of prototypes from sketches and captured clay models [33]. However, none of these systems offer the capabilities required for body-based AR.

Table 1 shows an overview of common toolkits for AR development as well as of libraries relevant for body-based AR. While many options exist, none are suitable for fast body-based AR prototyping. For example, many toolkits, such as *ARToolKit* [25] only handle tracking of the camera (e.g., with visual markers). While face tracking now also is a commonly available component, it is regularly restricted to only the front camera. This allows for selfie apps, but not for creation of applications that work with the faces of others.

For body-based AR, an additional third-party face tracking library thus would need to be included—something that is difficult for non-developers. We also see that support for body augmentation is mostly limited to external libraries as well as the filter editors from Instagram and Snapchat. Yet, while Instagram's filter development tool, Spark AR, for example, does allow for visual programming (as well as scripting) of face and body effects, these can only be used in their phone apps. Working with external libraries that bring in advanced face tracking and augmentation features also comes at a prize. Instead of fast prototyping in a graphical environment, tying in these libraries requires writing code and working with the system on a comparably low level. Hence, there is a gap in the AR development ecosystem, where no solution allows for fast and easy prototyping of body-based experiences.

2.5 Motivation for Body LayARs

With BodyLayARs, we address this gap and present an environment that brings together easy visual programming and body tracking as well as augmentation. By enabling prototyping with a visual programming approach, we cater to domain experts and other people not trained in software development, as there is evidence that visual programming can improve the performance of such users [41]. They can quickly assemble systems from a set of building blocks, yet Body LayARs also allows for extensive scripting and customization. Hence, developers with more advanced expertise are able to leverage it as well. Applications aimed at augmenting interpersonal interactions benefit especially from headset AR. Hence, prototypes developed with Body LayARs can be executed on the Microsoft Hololens (yet are fundamentally device agnostic). Furthermore, Body LayARs makes information on people available to built-in components, requiring no inclusion of additional libraries and thus allowing easy access to these features.

In the remainder of the paper, we describe the design and capabilities of Body LayARs in detail. We also show a range of examples of simple prototypes that would be complicated or impossible to built with existing prototyping solutions.

3 THE BODY LAYARS TOOLKIT

As we have described earlier, existing environments for AR development do not adequately address the requirements for body-based AR prototyping. Our Body LayARs toolkit is specifically designed to address these shortcomings. Specifically, we designed the toolkit around five goals:

Low barrier of use to enable people without expert knowledge in computer graphics, computer vision, networking, or machine learning to prototype body-based AR experiences. MacIntyre et al. pointed out that "non-technologists" find it hard to build prototypes, "due to both their lack of expertise in areas such as tracking, and to the complicated software development that would be required" [29]. While their and other people's software have made this easier for AR in general, body-based AR faces similar issues.

Fast iteration to encourage experimentation with minimal delay between changing a project and seeing that change in a running application. In addition to compile time costs, AR prototyping commonly requires an additional deployment step to the target device.

Device independence to enable project development compatible with several different AR devices. The AR landscape is changing rapidly which, as Ashtari et al. noted, "can make end-user developers feel especially left behind and struggle to keep up" [5]. An abstraction from specific hardware can help reduce the complexity users have to deal with.

Collaboration to allow multiple people to work on a prototype at the same time. Collaborative coding tools, such as *Collabode* [13], have shown the potential of this approach.

Extensibility to allow users to add functionality and share it with others. This is a common goals shared with many other toolkits, and AR prototyping—with a diverse device landscape—can particularly benefit from this.

3.1 Considerations

To achieve fast iteration times and low entry barriers, we opted for a web-based solution where projects are deployed to a host application already running on a target device. Similarly to HoloJS or buildwagon, this eliminates the need for users to have a compiler toolchain installed for the target device. Furthermore, deployment of projects to the target device becomes faster as no restart of the application is needed if projects are executed inside a host application. A browser-based editor also enables easier collaboration with project and code sharing, as well as simultaneous editing.

To further make development device-agnostic, we decided to develop most of Body LayARs in JavaScript and only have a small API to actual devices. Devices then only need to implement a small set of functions (e.g., rendering a model, returning the current user position, finding faces in the scene) to be able to run Body LayARs applications. Using JavaScript for the majority of the code also makes it easier to customize and extend Body LayARs.

3.2 Overview

Users work with the Body LayARs toolkit (see Figure 2 for a conceptual overview) via a web application. The application server holds all project files and enables project management, versioning, editing, and sharing. Prototype development primarily happens within a visual programming editor. In addition to the visual flow-based editing, users can also write JavaScript inside scripting nodes. Assets (such as 3d models or audio files) can be uploaded and then used within a project.

While development happens inside the web application, projects run on AR devices or in a standalone desktop application. In either case, after starting the Body LayARs application on a device, it automatically registers with the webserver. Users can see currently connected devices and their status at the bottom of the project editor. Once ready to test a project, users only need to click on one of the available devices to run their project on it.

To run a project, the webserver transforms the flow-based representation of the project into a JavaScript application package. Each node is translated into a corresponding JavaScript object and the resulting node graph linearized. Assets are stored on the server, and then referenced from the application package so applications

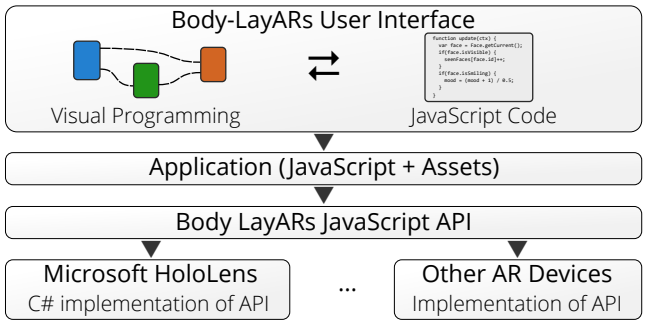

Figure 2: With Body LayARs, users built apps in a visual programming environment running in their browser. For execution, projects are transformed into JavaScript and send to a connected device. The same project can run on different devices as the Body LayARs API provides an abstraction layer.

can fetch them later. When starting a project, the host application on the selected device receives this package and runs the contained project. During execution, host devices are asked to call an UPDATE function of the packaged application every time a frame is rendered. While an application is executing, users can still make changes in the editor and update the application state. For example, they can change the color of a label that is being shown at runtime.

Because of the differences between AR devices, each device currently requires its own implementation of the host application. For example, there are different SDKs for the Microsoft HoloLens and the Magic Leap. While environments like Unity or the Unreal Engine provide some abstraction, there remain some fundamental architectural differences (such as the HoloLens only running UWP applications). We envision that the future *OpenXR* standard will soon enable more device-agnostic development.

We built host applications for the Microsoft HoloLens as well as for the Windows desktop. The latter allows for convenient prototyping but is limited in its capabilities due to running on a desktop. However, both implement the full Body LayARs JavaScript API and thus can run the same application bundles.

All parts of Body LayARs are open source[2]. We hope that this will result in further development of and interest in body-based AR. We are especially keen on widening access to experimentation with body-based AR from AR experts to a broader audience.

3.3 Project Server and Editor

The server contains all projects and offers management, editing, and deployment capabilities. This approach allows users to work on their body-based AR projects without setting up a development environment on their own machine. Instead of requiring a powerful development setup, they can work on any device with a browser.

Figure 3 shows the project editor in action. Users can instantiate nodes from a categorized drawer on the left. They can freely move nodes on the main canvas and drag between node attributes to connect them. The canvas can be panned and zoomed, which enables working with larger node layouts than fit on one screen.

[2]Available at https://github.com/henningpohl/body-based-ar-tk.

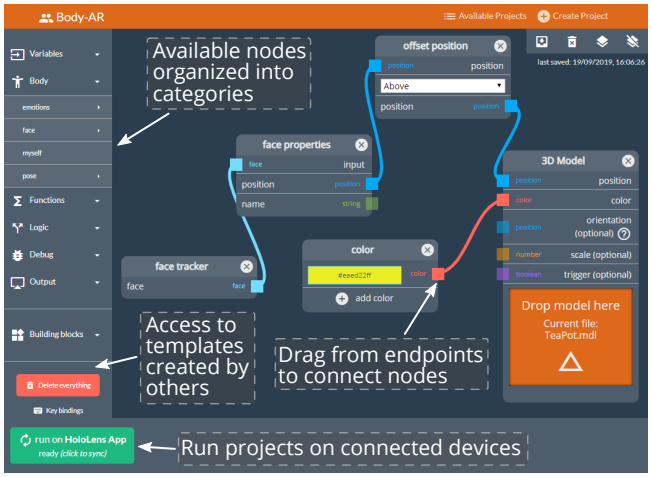

Figure 3: The web-based Body LayARs editor enables users to build applications by connecting nodes. Devices also connect to the editor, and users can run their applications on a connected device by selecting it from the device list. Once a project is running, users can continue to make changes to nodes, which propagate to the executing device.

Apart from connecting nodes, some nodes offer additional embedded functionality. For example, some allow for asset upload and storage, while others can be configured into different modes. Node inputs and outputs are typed (shown in color coding) to only allow suitable connections. However, there is some flexibility and there are nodes that accept input of a range of types. Where possible, connection points change their type as nodes are configured or inputs are set. Node connections can pass single items, but also multiple instances (e.g., when multiple faces are detected). This simplifies connecting nodes as users do not need to handle each instance separately.

The editor also shows a list of connected devices and allows users to start execution of the current project on any of them that are idle. Web server, devices, and browsers are all connected via a WebSocket communication layer build with *socket.io*[3]. This enables users to send execution commands to devices, but also to share node updates. Hence, while a Body LayARs application is already running, it can still be changed. However, when changes alter the structure of an application (e.g., when deleting nodes), device and editor desynchronize. In this case, the application continues running on the device, but will no longer update with changes from the editor.

3.3.1 Available Nodes. We provide nodes for body-specific functionality as well as more general processing, debugging, and output capabilities. Examples of the former are the FACE TRACKER and POSE TRACKER nodes. These provide the capability to receive a list of faces and poses visible to the device respectively. The output from these nodes can be routed through additional nodes to further augment the data. For example, a face can be given to a FACE RECOGNIZER node to retrieve a name for that face.

[3]https://socket.io/

General flow control is available through nodes like FILTER, CONDITIONAL, or LOOP. For example, the FILTER node can be used to reduce a bundle of all detected faces to only the closest one. The CONDITIONAL node works similarly, but also allows for branching to, for example, show different outputs depending on a currently visible face. For more complex logic or data handling, we provide the SCRIPT node, that allows users to embed arbitrary JavaScript code into their application. We describe this node in the next section on customizing and extending Body LayARs.

Users can use a set of output nodes to show the results of their applications. A basic example is the SOUND node which plays back a sample when triggered. With a LABEL node, floating text can be shown anchored within the scene, while the SPRITE and MODEL nodes do the same but with a sprite and full model respectively. While the anchor used can be static it can also be dynamically tied to a tracked scene feature, such as a person's head. For display of movement data we provide a PATH node, while a BARGRAPH node can be used to put a corresponding visualization into the world.

Finally, we also provide two kinds of debugging nodes. The APP DEBUG node enables textual output to an overlay inside a device. On the other side, the GRAPH DEBUG node only surfaces debug information inside the project editor.

3.3.2 Customizing, Extending, and Sharing. Users can modify Body LayARs in multiple ways. First, the SCRIPT node can be used for operations not supported by the visual programming environment. For example, users could use it to keep a face history (e.g., to trigger output based on when a person was last seen). In a SCRIPT node, any standard JavaScript language feature and type can be used. We also provide a few custom types specific to AR, such as COLOR, VECTOR, MATRIX, FACE, and POSE. Additionally, user scripts can make calls to the underlying Body LayARs JavaScript API (see below).

Second, node groups in a project can be shared by saving them as a named building block. These blocks are available to all other users in an extra menu at the bottom of the node drawer. This makes it easy to share common node combinations but also to share custom logic in SCRIPT nodes.

Third, all nodes are editable on the server. Nodes consist of at least an interface definition in JSON format and runtime JavaScript code. By editing a node's interface, users can add, change, or remove connection points available on that node. For example, they might want to add an input to the FACE RECOGNIZER node to be able to activate or deactivate it at runtime. Changes to the runtime code get deployed to devices and can substantially alter the behavior of a node.

More complex nodes also have custom styling and code for the editor. For example, the color node contains a color picker that shows up when the color value is selected. This is also editable by users so they can make improvements to existing widgets as well as add new ones.

Finally, users are able to apply the node editing capabilities to create entirely new nodes from scratch or based on existing nodes. In this way large changes to Body LayARs are possible. Users might want to create a new node from a script they have used or to make a common design easier to build. As with built-in nodes, how these new nodes show up and behave in the editor is also fully customizable.

Nodes on a project server are shared with all users. Changes made by one automatically manifest in everybody's projects. Similarly, if one user adds a new node or saves a building block, this is also available to all. We opted for this open design as we assume collaborating users working on prototypes. For this scenario, we value flexibility and collaboration higher than stability.

3.4 Body LayARs JavaScript API

While all application logic is handled via nodes and scripts, these have no way of reading inputs or effecting any outputs on their own. To interface the application logic with actual devices, we provide an API layer. The API is designed to be stateful and to mostly work asynchronously. Sounds, models, and labels are identified by handles that are passed to the API. This does not expose the actual objects to the runtime and allows devices to implement the API in a way that suits them most. Any input is received by callback functions that are registered for events, such as user movement or face tracking. In addition to device capabilities, it also provides access to state, such as the current time or frame number.

In addition to the Body LayARs API, host applications are also required to provide two extensions to the JavaScript runtime: (1) logging, and (2) networking. For logging, a CONSOLE.LOG function needs to be available. This is an especially useful functionality when debugging applications in IDEs. For networking, we require implementations to provide the XMLHTTPREQUEST API. This allows users to move networking code from the browser directly to Body LayARs. Prototypes can use this functionality to make requests to servers to, for example, fetch additional resources at runtime or to save tracking data online.

3.5 Host Applications

Applications written with Body LayARs are not directly executable on a device. Instead, they require a host application to be written for each device that can then run the packaged JavaScript bundle. An important requirement is hence that such applications need to be able to interface with JavaScript code. However, JavaScript engines are available for all relevant platforms for integration into applications. Hence, enabling running of Body LayARs application boils down to implementing the about 20 functions that form the JavaScript API.

We built an example host application for the Microsoft HoloLens, which we describe in this section. We chose to focus on AR headsets, in particular the HoloLens, as body-based AR experiences especially benefit from hands-free and see-through kinds of AR. Having to hold a phone in front of them would make testing of many scenarios (e.g., augmenting conversation) awkward and unnatural. However, note that the HoloLens also is not ideal for this purpose and does inhibit eye contact between participants.

We also compile a variant of the application that instead of on the HoloLens runs in a local window (enabling faster local testing). It shares all the Body LayARs relevant code with the HoloLens version and hence we will not describe it here.

3.5.1 Microsoft HoloLens. We built our host application for the HoloLens around the *UrhoSharp* engine. This provides a graphics layer abstraction, asset loading, an audio system, as well as integration of the HoloLens tracking. To run Body LayARs applications,

we embedded the *ChakraCore* JavaScript engine. The application itself is small and primarily translates calls to the Body LayARs JavaScript API into UrhoSharp calls. For example, when a model is loaded, the resource is fetched from the project server and just added to the engine's resource cache.

While the HoloLens is fast enough to run all logic and rendering locally, some operations require additional processing capabilities. For example, while we run face tracking on the HoloLens directly, this was not feasible for the more advanced person-centered detection and tracking. We hence offload this work to an external server. Correspondingly, these parts of Body LayARs are device-agnostic and other devices could make use of this.

3.5.2 Remote Services. Our external server provides services for face recognition, emotion classification, and pose tracking. As we run face detection locally, we only need to involve this server if (1) people are present, and (2) face or emotion recognition are actually required. For pose recognition we always need to send whole video frames to the server. We use *PoseNet* [48] for pose recognition, the *Face Recognition* library [12] for face recognition, and *FER* [30] for facial expression recognition, which we use to classify emotional state.

More advanced detection, tracking, and recognition is available, however we chose a set of models that still allowed us for close to realtime (we do not run the models on every frame from the HoloLens camera) execution. Furthermore, a limitation of the models we used is that they only provide 2D results. Hence, while we estimate the depth of recognized faces and joints, this is less accurate than full 3D model fitting. We will return to this and other limitations below.

3.6 Comparison to Other AR Tools

As we have described earlier, current AR development environments (see Table 1) do not adequately support prototyping of body-based AR. While there are some toolkits that allow for easy prototyping with an editor, such as Wikitude Studio or Torch, these do not track bodies. Google's ARCore and Apple's ARKit both support some tracking of faces and poses. However, this only works with the front facing camera of phones and tablets, prohibiting prototyping of applications that augment interaction with others—a core aspect of body-based AR—as well as immersive experiences. Similarly, while Spark AR enables building of some body-based experiences, it can only be used for filters running inside of the instagram phone app.

Body LayARs is similar to Micosoft's HoloJS and Magic Leap's MagicScript, in that all three are built on top of a JavaScript stack, which enables faster prototyping. Our visual programming environment is a further abstraction on top of this. Furthermore, HoloJS and MagicScript both are comparatively low-level and, for example, require developers to program in WebGL for graphics. Like Body LayARs, HoloJS applications can also be deployed quickly to an app running on a target device with *Spin*.

Table 1 also showed several libraries that can be used to add similar functionality as in Body LayARs to applications. However, all these are costly and require expertise in software development.

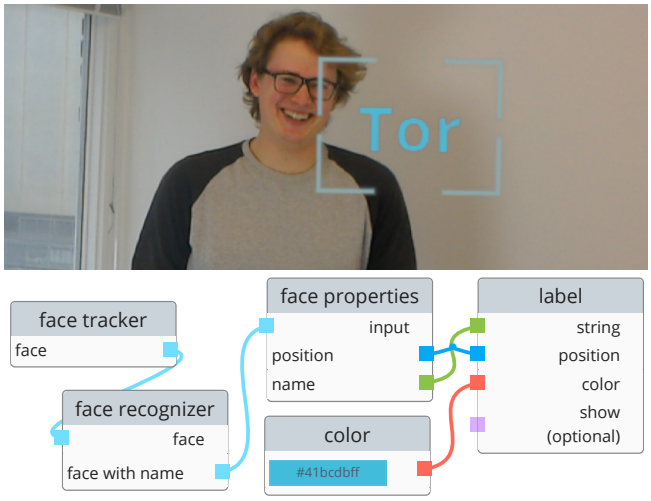

Figure 4: In this example, Body LayARs is used to detect and recognize people and then display nametags on top of them.

4 EXAMPLE PROJECTS

To show how Body LayARs enables easy development of body-based AR prototypes we have created a set of example applications. We have aimed to (1) cover a range of use cases, and (2) demonstrate the development capabilities provided by Body LayARs. For each of these examples, we show the corresponding Body LayARs application as well as captures of these applications running on the HoloLens. Note that we have reduced the fidelity Body LayARs nodes are shown at in the corresponding figures to make them more readable.

4.1 Placing Nametags on Recognized People

The first example application demonstrates a basic use of Body LayARs. To help people remember names, this application attaches nametags to them. As shown in Figure 4, this application only requires a few nodes, primarily: (1) a FACE TRACKER node to find faces in front of the user, (2) a FACE RECOGNIZER node to associate a name with each face, and (3) a LABEL node that places the name next to each face.

This application could be extended in many ways. For example, additional information for each recognized person could be retrieved from a web service with a SCRIPT node. The label could then show a combination of name and position, or name and last time that person was met.

4.2 Tracking Student Responses in Classrooms

Our second example application prototypes an application for teachers and instructors, working in classrooms. When one-on-one engagement is not feasible, they commonly just put problems before the students and ask them to vote on potential answers. While there is tooling support for this activity, it usually requires students to vote on a website using their laptop or mobile, instead of directly responding to the instructor. Traditionally, students could also just raise their hand to show their agreement with an answer.

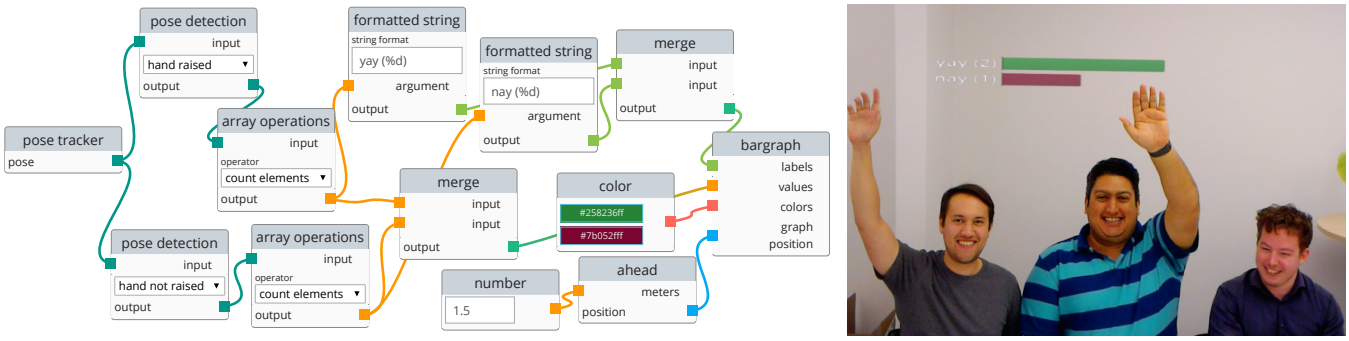

Figure 5: The project shown here tracks people's poses and determines how many raised their hand and how many did not. This is combined with some color coding and labels to create a bar chart visualizing the result of this show of hands.

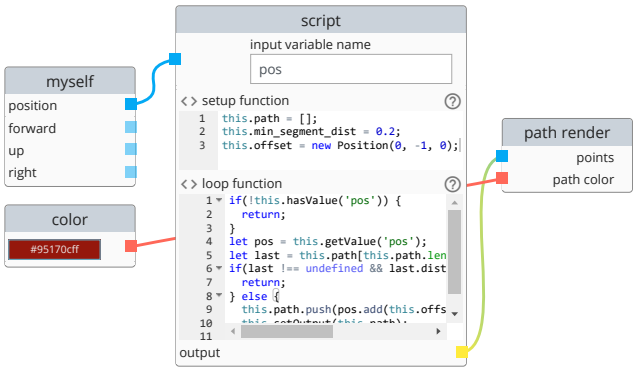

Figure 6: This projects shows how a script node can be used to aggregate incoming data into more complex structures. Here, a path is assembled from the movement data of the user and shown in the world.

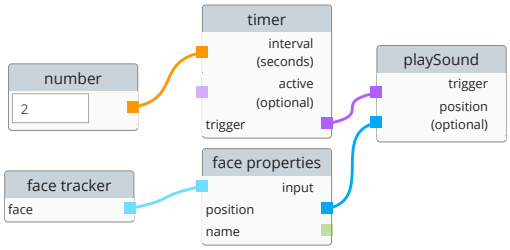

Figure 7: In this project, every two seconds a sound is played at the location of every person around the user.

However, keeping track of students and getting a decent tally of the room can be challenging, especially in large rooms and small differences in voting behavior. Yet, where it is hard for people to keep track of a large number of people, this is not the case for computers. Figure 5 shows a Body LayARs prototype that works with a small group of people. Their poses are tracked and then the instances where the hands are raised or lowered are counted up respectively. These counts are then visualized in a bar graph—color coded and labeled.

4.3 Emotion Annotation

Our third example shows how body-based AR can help users to better notice the emotional state of people around them. For people that have trouble reading faces, this can be a conversational aid, but it could also be seen as a form of expression. In the project shown in Figure 8, a FACE TRACKER works in combination with an EMOTION DETECTOR to infer emotional state from faces in view of the user. Here, we are only interested in *sad* faces, which we detect with a string comparison against the dominant detected emotion. If a person is found to be sad, a model is added to hover just above their face. In this case, the project includes the model of a cartoon-like cloud that is colored in a dark gray.

4.4 Visualizing Self-Tracking Data

The earlier examples all demonstrate prototypes that work with other people. To show that Body LayARs can also be used to work with one's own body and movements we included this fourth example (shown in Figure 6). This application aggregates the user's movement through space and visualizes it using a path.

The example shows the most reduced instance of self tracking and we can envision multiple ways this can be extended to prototype more intricate applications. Instead of showing movement as a path, the aggregating function could instead discretize location data and build a spatial heatmap for a room.

4.5 Sonification

Finally, with our fifth example we demonstrate that Body LayARs can also be used to build non-visual applications. The project shown in Figure 7 again tracks people around the user, but this time uses the positional information to anchors sounds in the scene. Every two seconds the sounds effect are triggered.

Instead of playing sound samples, this could be extended by recognizing people and then speaking out their names with a SPEAK TEXT node. With an additional script, it could be detected when a person first appears and instead of continuously sounding out their names, they would only be announced once. While emotion can also be gleaned from voice, additional sonification of visual information could aid blind people in conversations.

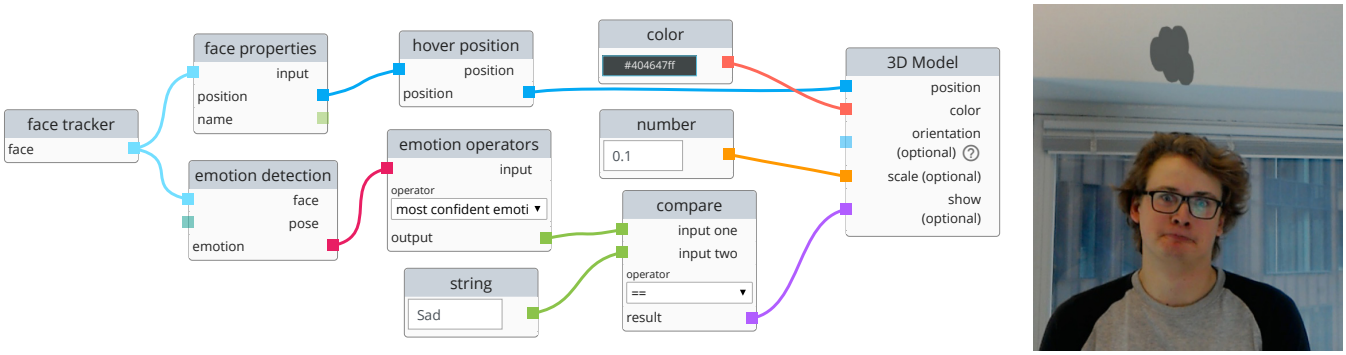

Figure 8: In this example people's emotions are inferred and a dark cloud is rendered above people found to be sad.

5 LIMITATIONS

Body LayARs has a number of limitations on the editor and runtime side as well as in the HoloLens application. While the editor and runtime allow for extension and customization, the set of nodes available out of the box is still limited. We believe that the set we provide allows for an interesting set of initial explorations, but for more complex kinds of prototypes likely will need to be extended.

Use of JavaScript allows for faster iteration (nothing needs to be compiled) and broad accessibility. However, the lack of static typing also means that runtime behavior of prototypes can be more fragile. Once it has become clearer what kind of functionality is required and which features are superfluous for body-based AR, it would be sensible to put more constraints on the development.

We have already mentioned above that the tracking capabilities on the device are currently limited. This is to strike a balance between the desire for realtime performance and the fidelity of tracking. For example, more advanced face tracking on the HoloLens is possible (as demonstrated by the HoloFace project [22]), yet comes at additional computational cost. As we do not have full 3d data available on tracked faces and bodies, some augmentation can be expressed in Body LayARs, yet not rendered by the HoloLens application. For example, the API allows users to specify that they would like to add an eye shadow to a tracked face, yet this is not currently rendered. A host application for future, more powerful, AR headsets could then enable this kind of augmentation.

While Body LayARs supports playback of spatial audio, the audio functionality in general is comparably limited. For example, in addition to tracking people around oneself using cameras, it should also be possible to do the same with microphones. Yet, while some headsets come with microphone arrays built-in, we have found access to these to be too restricted. Hence, Body LayARs currently does not include any capability for using audio as an input.

In Body LayARs we currently also do not address the privacy issues that body-based AR is fraught with. As shown by Acquisti et al., face recognition in an AR context can easily be abused [1]. Hence, while body-based AR can be beneficial for users (e.g., by aiding them in social situations), the cost of it can be felt more by other people. Social acceptability will depend on negotiating a balance between privacy and utility. As Body LayARs is a prototyping tool, we took a non-restrictive approach. However, we do limit all recognition to people who are explicitly added to the system.

6 CONCLUSION

Advances in computer vision are enabling a new kind of AR experience: body-based AR, where the augmentation is focused on adding to interaction with people. Prototyping this kind of AR experience, however, is currently complicated, effectively limiting who can explore the space of body-based AR. To alleviate this issue, we have presented the open source Body LayARs toolkit, which enables users to rapidly prototype body-based AR experiences.

Body LayARs provides a graphical flow-based programming environment, but also allows users to deeply customize and extend the toolkit. Where the graphical programming is not sufficient, users can integrated chunks of JavaScript directly or use JavaScript to write entirely new components. With a set of example applications, we have shown that Body LayARs enables easy augmentation of interactions based on identity, emotional state, and movement of oneself and others. We have kept these simple but have outlined throughout the paper how the capabilities of Body LayARs can support more complex developments. For example, as networking is available, users can tie body-based AR prototypes to more powerful web backends. And as a full JavaScript engine forms the underlying execution environment, users are free to pull in any of the plethora of JavaScript modules available from others.

We believe that body-based AR offers exciting possibilities. With Body LayARs many experiences can be prototyped today. But ongoing development in the underlying technologies will open up more possibilities in the future. In particular, we can expect full 3d face and pose tracking to mature and for AR devices to incorporate hardware for faster execution of neural network models. A benefit of developing with Body LayARs is that, because these applications build on higher level abstractions, improvements in the underlying tracking will directly benefit existing projects. Furthermore, these technological advances will enable more complex applications, such as visually augmenting other's bodies in realtime or tracking larger groups (such as in classrooms).

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement 648785).

REFERENCES

[1] Alessandro Acquisti, Ralph Gross, and Fred Stutzman. 2014. Face Recognition and Privacy in the Age of Augmented Reality. *Journal of Privacy and Confidentiality* 6, 2 (2014), 1.

[2] Ejaz Ahmed, Michael Jones, and Tim K. Marks. 2015. An Improved Deep Learning Architecture for Person Re-Identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Teresa Almeida, Rob Comber, Gavin Wood, Dean Saraf, and Madeline Balaam. 2016. On Looking at the Vagina Through Labella. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. ACM, New York, NY, USA, 1810–1821. https://doi.org/10.1145/2858036.2858119

[4] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. DensePose: Dense Human Pose Estimation in the Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K. Chilana. 2020. Creating Augmented and Virtual Reality Applications: Current Practices, Challenges, and Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, Article 593, 13 pages.

[6] Moataz El Ayadi, Mohamed S. Kamel, and Fakhri Karray. 2011. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition* 44, 3 (2011), 572 – 587. https://doi.org/10.1016/j.patcog.2010.09.020

[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*.

[8] Jena Daniels, Jessey N. Schwartz, Catalin Voss, Nick Haber, Azar Fazel, Aaron Kline, Peter Washington, Carl Feinstein, Terry Winograd, and Dennis P. Wall. 2018. Exploratory study examining the at-home feasibility of a wearable tool for social-affective learning in children with autism. *npj Digital Med* 1 (2018), 32:1–32:10. https://doi.org/10.1038/s41746-018-0035-3

[9] Nir Diamant, Dean Zadok, Chaim Baskin, Eli Schwartz, and Alex M. Bronstein. 2019. Beholder-Gan: Generation and Beautification of Facial Images with Conditioning on Their Beauty Level. In *2019 IEEE International Conference on Image Processing (ICIP)*. 739–743. https://doi.org/10.1109/ICIP.2019.8803807

[10] W. Keith Edwards, Mark W. Newman, and Erika Shehan Poole. 2010. The Infrastructure Problem in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. ACM, New York, NY, USA, 423–432. https://doi.org/10.1145/1753326.1753390

[11] Hasan Shahid Ferdous, Thuong Hoang, Zaher Joukhadar, Martin N. Reinoso, Frank Vetere, David Kelly, and Louisa Remedios. 2019. "What's Happening at That Hip?": Evaluating an On-body Projection Based Augmented Reality System for Physiotherapy Classroom. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 234, 12 pages. https://doi.org/10.1145/3290605.3300464

[12] Adam Geitgey. 2019. Face Recognition. Retrieved 19-Sep-2019 from https://github.com/ageitgey/face_recognition

[13] Max Goldman, Greg Little, and Robert C. Miller. 2011. Real-Time Collaborative Coding in a Web IDE. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 155–164. https://doi.org/10.1145/2047196.2047215

[14] Jens Grubert, Matthias Heinisch, Aaron Quigley, and Dieter Schmalstieg. 2015. MultiFi: Multi Fidelity Interaction with Displays On and Around the Body. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. ACM, New York, NY, USA, 3933–3942. https://doi.org/10.1145/2702123.2702331

[15] Riza Alp Güler and Iasonas Kokkinos. 2019. HoloPose: Holistic 3D Human Reconstruction In-The-Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Sinem Güven and Steven Feiner. 2003. A hypermedia authoring tool for augmented and virtual reality. *New Review of Hypermedia and Multimedia* 9, 1 (2003), 89–116. https://doi.org/10.1080/13614560410001725329 arXiv:https://doi.org/10.1080/13614560410001725329

[17] Rita Hargrave, Richard J. Maddock, and Valerie Stone. 2002. Impaired Recognition of Facial Expressions of Emotion in Alzheimer's Disease. *The Journal of Neuropsychiatry and Clinical Neurosciences* 14, 1 (2002), 64–71. https://doi.org/10.1176/jnp.14.1.64 PMID: 11884657.

[18] Thuong Hoang, Martin Reinoso, Zaher Joukhadar, Frank Vetere, and David Kelly. 2017. Augmented Studio: Projection Mapping on Moving Body for Physiotherapy Education. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 1419–1430. https://doi.org/10.1145/3025453.3025860

[19] Xin Jin, Rui Han, Ning Ning, Xiaodong Li, and Xiaokun Zhang. 2019. Facial Makeup Transfer Combining Illumination Transfer. *IEEE Access* 7 (2019), 80928–80936. https://doi.org/10.1109/ACCESS.2019.2923116

[20] Amit Kale, Aravind Sundaresan, A. N. Rajagopalan, Naresh P. Cuntoor, Amit K. Roy-Chowdhury, Volker Krüger, and Rama Chellappa. 2004. Identification of Humans Using Gait. *IEEE Transactions on Image Processing* 13, 9 (Sep. 2004), 1163–1173. https://doi.org/10.1109/TIP.2004.832865

[21] Angjoo Kanazawa, Jason Y. Zhang, Panna Felsen, and Jitendra Malik. 2019. Learning 3D Human Dynamics from Video. In *Computer Vision and Pattern Recognition (CVPR)*.

[22] Marek Kowalski, Zbigniew Nasarzewski, Grzegorz Galinski, and Piotr Garbat. 2018. HoloFace: Augmenting Human-to-Human Interactions on HoloLens. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 141–149. https://doi.org/10.1109/WACV.2018.00022

[23] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. 2011. HMDB: A Large Video Database for Human Motion Recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[24] European Theater Lab. 2017. The history of augmented reality and how theatre may benefit from it. Retrieved 25-Aug-2019 from https://www.europeantheatrelab.eu/history-augmented-reality-theatre-may-benefit/

[25] Philip Lamb. 2004. ARToolKit. Retrieved 25-Aug-2019 from http://www.hitl.washington.edu/artoolkit/

[26] Zhenzhong Lan, Yi Zhu, Alexander G. Hauptmann, and Shawn Newsam. 2017. Deep Local Video Feature for Action Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[27] Ji Liu, Shuai Li, Wenfeng Song, Liang Liu, Hong Qin, and Aimin Hao. 2018. Automatic Beautification for Group-Photo Facial Expressions Using Novel Bayesian GANs. In *Artificial Neural Networks and Machine Learning – ICANN 2018*. Springer International Publishing, 760–770.

[28] Runpeng Liu, Joseph P. Salisbury, Arshya Vahabzadeh, and Ned T. Sahin. 2017. Feasibility of an Autism-Focused Augmented Reality Smartglasses System for Social Communication and Behavioral Coaching. *Frontiers in Pediatrics* 5 (2017), 145:1–145:8. https://doi.org/10.3389/fped.2017.00145

[29] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 197–206. https://doi.org/10.1145/1029632.1029669

[30] Mayur Madnani. 2018. FER — Facial Expression Recognition. Retrieved 20-Sep-2019 from https://github.com/mayurmadnani/fer

[31] Dhwani Mehta, Mohammad Faridul Haque Siddiqui, and Ahmad Y. Javaid. 2018. Facial Emotion Recognition: A Survey and Real-World User Experiences in Mixed Reality. *Sensors* 18, 2 (2018). https://doi.org/10.3390/s18020416

[32] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics* 36, 4, 14. https://doi.org/10.1145/3072959.3073596

[33] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173927

[34] Tao Ni, Amy K. Karlson, and Daniel Wigdor. 2011. AnatOnMe: Facilitating Doctor-patient Communication Using a Projection-based Handheld Device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 3333–3342. https://doi.org/10.1145/1978942.1979437

[35] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. 2018. PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model. In *The European Conference on Computer Vision (ECCV)*.

[36] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. 2017. Towards Accurate Multi-Person Pose Estimation in the Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[37] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[38] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. 2016. Invertible Conditional GANs for image editing. arXiv:1611.06355 [cs.CV]

[39] Ken Perlin. 2015. Eccescopy: To look, is to see. *XRDS* 22, 1 (Nov. 2015), 36–39. https://doi.org/10.1145/2810052

[40] Ken Perlin. 2016. Future Reality: How Emerging Technologies Will Change Language Itself. *IEEE Computer Graphics and Applications* 36, 3 (May 2016), 84–89. https://doi.org/10.1109/MCG.2016.56

[41] Thomas W. Price and Tiffany Barnes. 2015. Comparing Textual and Block Interfaces in a Novice Programming Environment. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*

(Omaha, Nebraska, USA) *(ICER '15)*. Association for Computing Machinery, New York, NY, USA, 91–99. https://doi.org/10.1145/2787622.2787712

[42] Douglas A. Reynolds and Richard C. Rose. 1995. Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE Transactions on Speech and Audio Processing* 3, 1 (Jan 1995), 72–83. https://doi.org/10.1109/89.365379

[43] Rui Sacchetti, Tiago Teixeira, Bruno Barbosa, António Neves, Sandra Soares, and Isabel Dimas. 2018. Human Body Posture Detection in Context: The Case of Teaching and Learning Environments. In *Proceedings of the Third International Conference on Advances in Signal, Image and Video Processing (SIGNAL '18)*. 79–84.

[44] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive Body-Driven Graphics for Augmented Video Performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 622, 12 pages. https://doi.org/10.1145/3290605.3300852

[45] Rajinder Sodhi, Hrvoje Benko, and Andrew Wilson. 2012. LightGuide: Projected Visualizations for Hand Movement Guidance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. ACM, New York, NY, USA, 179–188. https://doi.org/10.1145/2207676.2207702

[46] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild.* Technical Report CRCV-TR-12-01. Center for Research in Computer Vision, University of Central Florida.

[47] Flavia Sparacino, Christopher Wren, Glorianna Davenport, and Alex Pentland. 1999. Augmented Performance in Dance and Theater. In *International Dance and Technology 99*.

[48] TensorFlow. 2019. Pose Detection in the Browser: PoseNet Model. Retrieved 19-Sep-2019 from https://github.com/tensorflow/tfjs-models/tree/master/posenet

[49] Robert Xiao, Teng Cao, Ning Guo, Jun Zhuo, Yang Zhang, and Chris Harrison. 2018. LumiWatch: On-Arm Projected Graphics and Touch Input. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, Article 95, 11 pages. https://doi.org/10.1145/3173574.3173669

[50] Q. Xu, S. S. Cheung, and N. Soares. 2015. LittleHelper: An augmented reality glass application to assist individuals with autism in job interview. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. 1276–1279. https://doi.org/10.1109/APSIPA.2015.7415480

[51] Jason Y. Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. 2019. Predicting 3D Human Dynamics from Video. In *The IEEE International Conference on Computer Vision (ICCV)*.