Pose-on-the-Go: Approximating User Pose with Smartphone Sensor Fusion and Inverse Kinematics

Karan Ahuja
Carnegie Mellon University
Pittsburgh, PA, USA
kahuja@cs.cmu.edu

Sven Mayer
Carnegie Mellon University
Pittsburgh, PA, USA
info@sven-mayer.com

Mayank Goel
Carnegie Mellon University
Pittsburgh, PA, USA
mayank@cs.cmu.edu

Chris Harrison
Carnegie Mellon University
Pittsburgh, PA, USA
chris.harrison@cs.cmu.edu

ABSTRACT

We present Pose-on-the-Go, a full-body pose estimation system that uses sensors already found in today's smartphones. This stands in contrast to prior systems, which require worn or external sensors. We achieve this result via extensive sensor fusion, leveraging a phone's front and rear cameras, the user-facing depth camera, touchscreen, and IMU. Even still, we are missing data about a user's body (e.g., angle of the elbow joint), and so we use inverse kinematics to estimate and animate probable body poses. We provide a detailed evaluation of our system, benchmarking it against a professional-grade Vicon tracking system. We conclude with a series of demonstration applications that underscore the unique potential of our approach, which could be enabled on many modern smartphones with a simple software update.

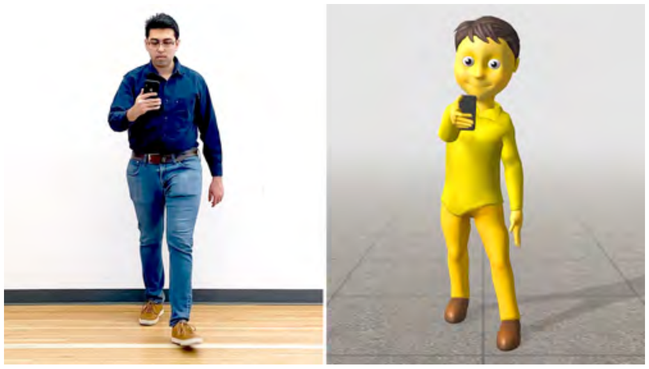

Figure 1: Running on an unmodified smartphone held in the hand (left), Pose-on-the-Go produces an animated, full-body pose estimation (right) through sensor fusion.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Interaction techniques*; Gestural input.

KEYWORDS

Motion capture, Motion tracking, Body pose, Mobile, Smartphone

ACM Reference Format:
Karan Ahuja, Sven Mayer, Mayank Goel, and Chris Harrison. 2021. Pose-on-the-Go: Approximating User Pose with Smartphone Sensor Fusion and Inverse Kinematics. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3411764.3445582

1 INTRODUCTION

The human body is expressive and offers many degrees of freedom for human-computer input purposes. Technologies able to digitize a user's full-body pose could enable new interactive experiences beyond the touch-centric (and occasionally IMU-driven) input that we see on contemporary mobile devices.

Today, full-body motion capture is most closely associated with computer-generated imagery in blockbuster films, using expensive multi-camera rigs and special suits with markers. However, as technologies have improved, consumer-oriented uses have become possible. For instance, there are now several companies offering small sensors, worn on the body, that digitize the wearer's pose for use in more immersive VR experiences [5, 68]. Of course, the bar for consumer acceptance is high, and this highly-instrumented approach seems unlikely to go mainstream in the near future. A decade ago, Microsoft took a different approach with its XBox Kinect sensor [41], a $150 accessory depth camera that could capture users' pose without any worn instrumentation. A variety of interactive, pose-enabled games proliferated, crossing genres including sports, dance, and role-playing games [42].

Regardless of whether the sensors are worn or external, the necessity for extra devices, plus the added cost of that hardware, dampens the likelihood of mass adoption. More importantly, both approaches preclude many interesting uses of body digitization when people mobile and outside of controlled settings. In response, we set out to develop a full-body pose estimation system that could run entirely self-contained on a smartphone held normally in one's hand. Our system can work on the go, offering new avenues of interactivity *anywhere* and without prior setup. For this reason, we call our system *Pose-on-the-Go*.

Achieving this vision required leveraging almost every sensor at our disposal in modern smartphones, including the front and rear cameras, user-facing depth camera, capacitive touchscreen, and IMU. We fuse data from these disparate sensors to rig a real-time, animated skeleton of the user as they operate their phone (see Figure 1). As far as we are aware, Pose-on-the-Go is the first system to demonstrate full-body pose estimation using an unmodified smartphone held in the hand. This affords our approach wide applicability and superior practicality over other methods, which almost all require special instrumentation. An additional contribution is our rigorous study, benchmarking against a true gold standard - a professional-grade Vicon optical tracking system.

We believe exposing live user pose (even a coarse approximation as we demonstrate) as an API on mobile devices could enable some very creative and novel interactive experiences. For example, one of our example applications is a 3/4 perspective space shooter where the user's virtual on-screen character matches their live body pose, offering a unique level of embodiment not previously seen in smartphone gaming. Indeed, a significant benefit of being software-only is that many recent smartphone models could be enabled via an over-the-air update, and our software could run as a background service on top of which developers could build pose-enabled apps.

2 RELATED WORK

Full-body motion capture has a long history, dating back to at least 1878 with Muybridge's "The Horse in Motion" [21]. This film is widely regarded as the start of chronophotography, a photographic technique for the scientific study of movement and especially locomotion. Later pioneers, such as Max Fleischer, used rotoscoping as a way to capture and then transform complex movements, such as locomotion, for use in early 20th century animated films. Today, digital technologies have enabled a wide variety of highly-automated and precise techniques for human motion capture. We now review this literature, paying particular attention to systems that capture continuous full-body pose, as opposed to techniques for tracking just fingers or hands in free space.

2.1 External Body Capture Sensing

The most common way to capture a user's pose is with external sensors fixed in a room. Cameras are by far the most popular sensor type. To facilitate accurate capture, passive (e.g, retroreflective balls [45, 67], fiducials [5]) or active [51, 68] markers can be worn. The advent of depth cameras helped to overcome long standing challenges in user segmentation and enabled some of the first commercial applications offering markerless pose tracking (e.g., Microsoft Kinect [41], OpenNI [48], and Intel RealSense [31]). More recently, advances in computer vision have enabled markerless pose tracking with standard RGB cameras (e.g., OpenPose [16], PoseNet [49], DensePose [4]). Many non-optical body tracking approaches have also been considered, utilizing e.g., mechanical linkages [62], acoustics [3, 20], magnetic fields [47, 53], RF [74], and RFID tags [33].

2.2 Worn Body Capture Sensing

The need for external infrastructure inherently means body capture is limited to a specific location. Additionally, external sensors may be undesirable in settings such as the home for aesthetic and privacy reasons. Finally, sensing a user at a distance (e.g., with a camera) has inherent accuracy drawbacks. For these reasons, worn systems are also popular, offering location flexibility and generally a high degree of accuracy, benefiting from tight coupling to the object they are sensing.

A wide variety of approaches have been developed, including exoskeletons [40], worn ultrasonic beacons [24, 69], and worn magnetic tracking [17]. However, two categories of sensing approaches stand out. First are IMU-equipped, battery-powered and wireless sensor boards, now sufficiently small and light to permit placement on many parts of the body without impeding movement. There are innumerable academic [25, 44, 63] and commercial systems [5, 30, 66, 71] that use this approach, often coupled with an inverse kinematic solver to improve output. Second most common are camera-based methods. Arrangements include cameras worn on the head with markers on the body [73], outward-facing cameras worn on joints capturing optical flow [60] and multi-user co-located systems wherein users digitize each other [1]. There are also systems that fuse data from worn IMUs and cameras [38, 55].

We also note there is an expansive literature on worn systems that recognize static hand poses (thumbs up, fist, etc.) for gestural input purposes (see e.g., Chen et al. [18] for a survey). However, these systems are very different in operation and goal, and do not attempt continuous pose estimation.

2.3 Single-Point Worn Body Capture Sensing

If a body capture technology needs only a single point of instrumentation, and is not reliant on any extra components worn on the body or installed in the environment, it can be considered "self-contained" and "single-point". Such systems are inherently mobile, able to function anywhere in the world. This combination of properties makes it the rarest in the literature, but also often the most practical.

There are several single-point worn systems able to capture arm pose. Among the earliest was Digits [35], which used a wrist-worn infrared camera and line laser to measure finger bend, which was passed to an IK solver to generate a continuous hand pose. ArmTrack [59] demonstrated that an IMU in a smartwatch, in concert with an IK solver, could estimate continuous arm pose (shoulder, elbow and wrist joints). IK has similarly been used to model upper-body pose [46, 50]. Worn depth cameras have also been used to capture a 3D model of the arms and hands [27, 56]. More recently, commercial AR/VR systems such as the Oculus Quest 2 [22] and Microsoft HoloLens [43] include hand and arm pose tracking capabilities. The former uses wide angle cameras, while the latter relies on specialized depth cameras, all of which are integrated into a VR/AR headset. It is also worth mentioning PuppetPhone [6], which uses several smartphone sensors and a "MotionStick" interaction metaphor to realistically control (and animate) a virtual character in a passthrough AR experience.

There are only a handful of self-contained, worn systems able to capture full body pose. The first is EgoCap [54], which used a pair of downward-facing fisheye cameras cantilevered from the head, proving a sufficient view for a computer vision pose model to extract a body skeleton. Mo2Cap2 [72] is virtually identical, except it uses just a single fisheye camera. Instead of requiring extra cameras, MeCap [2] shows that the rear camera of a smartphone placed into a low-cost VR headset can capture full body pose with a clip-on mirror accessory. Finally, and perhaps most practical, is [64], which demonstrates that the wide-angle RGB cameras contained in Facebook's latest VR headsets could be used to capture a wearer's body skeleton. Equally practical is [19], which uses a smartphone's user-facing camera and IMU for 3-DOF localization of the device with respect to the user's body for around body interaction. Later, [36] extended this idea and added a depth camera to improve tracking

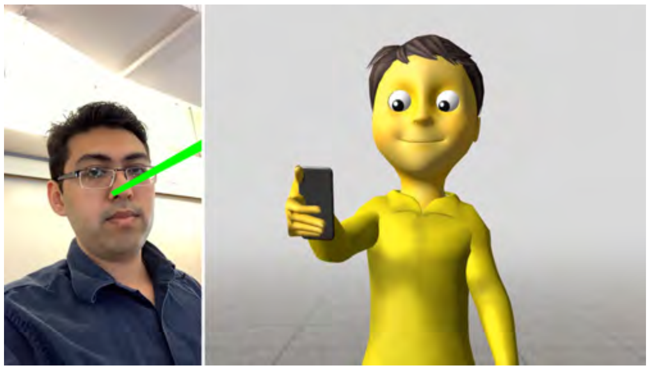

Figure 2: Left: User-facing camera view with head vector shown in green. Right: Avatar with corresponding head pose. Note that eye gaze is also captured and applied to the avatar.

accuracy and capture the user's shoulders. Finally, [11] uses two smartphones in a 3D printed case to provide tracking of a user's hand holding the device, along with the user's head position, relative to a distant display using inside-out tracking. To our knowledge, Pose-on-the-Go is the only system to demonstrate full-body pose tracking using a single unmodified smartphone held regularly in the hand.

3 IMPLEMENTATION

We built our proof-of-concept implementation on iOS, which has an API that allows both the front and rear cameras to be streamed simultaneously (a feature now available on some Android smartphones, and likely to be mainstream soon). iOS also offered a robust set of APIs across all of the features we wished to implement (face tracking, pedometer, locomotion mode prediction, 6-DOF absolute spatial tracking, etc.), and many of the heavier-weight APIs are optimized to take advantage of hardware acceleration, offering us some computational headroom. As a development device, we selected an iPhone XR, which is Apple's mid-tier offering.

3.1 Inverse Kinematic Model & 3D Engine

There are many inverse kinematic (IK) SDKs available, both open source and commercial. We chose to use Root-Motion's VRIK library [58] running on the popular Unity engine [65], which has native iOS support and an extensive set of tools and assets for creating example apps. As with our hardware, Pose-on-the-Go does not have any strong dependencies and should work with practically all human IK packages. To facilitate rapid prototyping, we wrote a thin smartphone app responsible for interfacing with all sensors and streaming data to a MacBook Pro laptop (3.1 GHz Dual-Core Intel i5, 16 GB RAM) over WiFi, where our Unity-based implementation, along with the IK solver ran. With additional engineering, it should be possible to run the entire process on the smartphone.

3.2 Head Position & Orientation

The first step of our process is to establish the position and orientation of the head relative to the phone. For this, we use the user-facing camera and ARKit's ARFaceAnchor API [7], which offers 6-DOF head tracking (see Figure 2 and 3). Note this does not use the iPhone's user-facing depth camera.

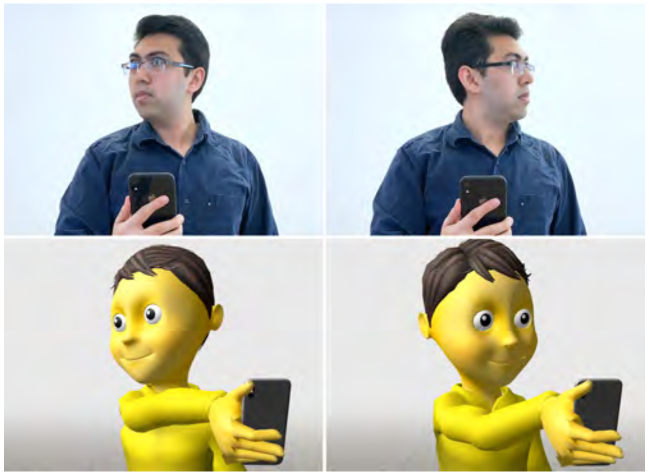

Figure 3: Head rotations are also reflected by the avatar independently from the chest normal.

3.3 Eye Gaze

The same ARFaceAnchor API [7] also provides an estimate of the gaze vectors for each eye. Although too inaccurate to enable gaze targeting on the phone's screen, it is more than sufficient to realistically animate the eyes of our avatar, providing another dimension of expressivity (see Figure 2, 6 and also Video Figure at 0:39; feature disabled in all other clips).

3.4 Torso Orientation

While ARKit comes with advanced, built-in functionality to track heads, it offers no facilities to track the user's lower body. Critically, body orientation needs to be established as it moves independently from the head. To capture chest *yaw*, we use the iPhone XR's user-facing depth camera to exact two bilaterally-symmetric patches on the user's torso (Figure 4) and compute a chest yaw vector. These patches cannot be statically positioned, as the user's head and body move around in the camera's field of view, and sometimes the chest is not visible at all. Instead, we take 85% of the detected head width and place two patches on either side of the head's center line. For vertical positioning, we move down 50% of the detected head height from the bottom of the head. We then use the median depth value from each 5x5 pixel patch, and the distance between the two patches, to estimate torso rotation. If the patches are not in the depth camera's field of view, the IK solver tries its best to animate and orient the torso using other available data.

Estimating chest *pitch* is more challenging, as less of the chest is seen vertically from which to extract reliable patches, and variation in body geometry, such as breasts, can affect the calculated pitch vector. These factors would have to be accounted for in order to produce a reliable estimate. Fortunately, in most cases when one is

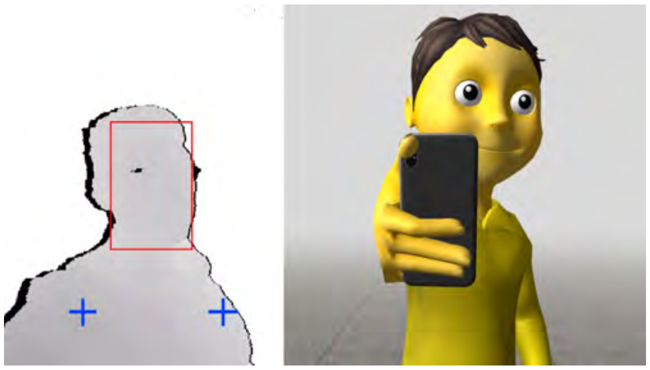

Figure 4: Left: Depth map provided by iPhone XR user-facing TrueDepth camera, with head tracking (red; bounding box extracted from the RGB camera) and torso patch extraction (blue) superimposed. Right: this data allows the torso to rotate independently of the head, as seen in this avatar.

standing, walking or sitting, the chest is generally oriented perpendicular to the ground, without an extreme pitch, and so we found it was not vital to compute this vector to achieve reasonable output.

We also acknowledge that only a handful of Android phones contain user-facing depth cameras at present, but as costs fall, it may become more pervasive in the same way multiple rear RGB cameras have trickled down to mid-tier devices. Regardless, in cases where a depth camera is not available, we found that a rough approximation of the torso yaw is still possible by using the user-facing RGB camera and a 2D pose model (e.g., PoseNet [49]) and utilizing the asymmetry of the two shoulder points relative to the head.

3.5 Phone Orientation

Using its IMU, the phone tracks its absolute 3-DOF orientation, establishing both north and down (i.e., gravity vector). We use this data to animate the avatar's hand holding the phone. More importantly, by combining this 3-DOF data with the aforementioned 6-DOF tracking of the user's head, we can now correctly orient the head with respect to the world, from which we can "hang" the rest of the user's body (see Figure 5).

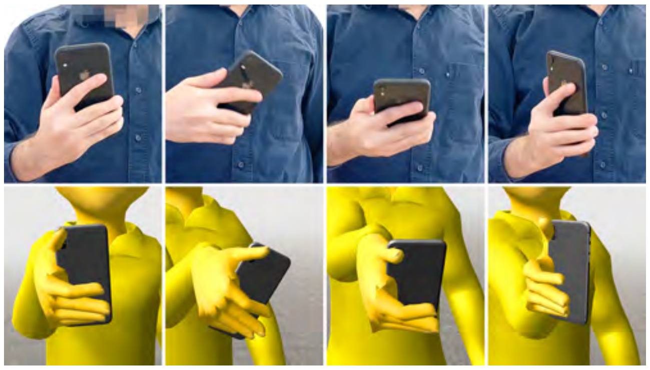

Figure 5: Phone orientation (pitch, yaw, and roll) is used to control the phone prop and better pose the wrist and hand.

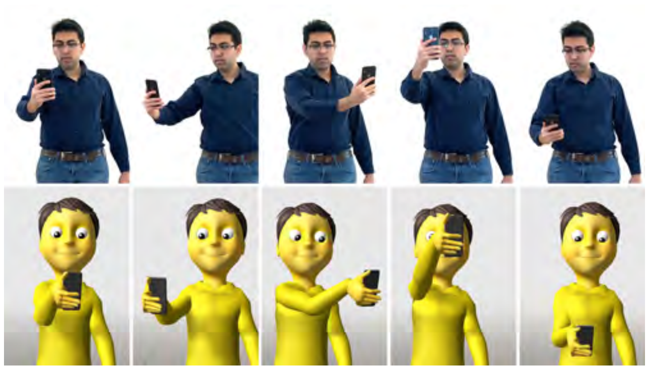

Figure 6: Pose-on-the-Go tracking a user's arm pose and eye gaze.

3.6 Arm & Hand (Holding Phone) Pose

At this stage, we know the relative spatial arrangement between the phone, head and torso. We assume the phone is held in one hand, and that an arm links the two (Figure 6). As the elbow and wrist joints are rarely seen in the user-facing camera view, we instead use the IK solver to generate a likely arm pose, articulating the avatar's elbow to connect the two points (i.e., shoulder to phone). As briefly noted in the last section, we also take into account phone orientation to articulate the wrist joint. Finally, we assume the phone is held in a standard grasp, and so we pose the avatar's fingers to match.

3.7 Arm & Hand (Not Holding Phone) Pose

For the arm not holding the phone, we have very little data to operate on. Although the shoulder is often visible in the user-facing RGB and depth cameras, we almost never see the elbow, wrist, or hand. Fortunately, it is not uncommon for users to employ their other hand for input on the touchscreen. In such cases, we can use the cartesian touch screen location of the finger, in combination with the 6-DOF location of the phone relative to the torso, to pose a plausible arm.

Touchscreen input, such as clicks and swipes, are animated on the virtual phone held in the avatar's hand. Note that we lose tracking the instant the finger leaves the touchscreen. Rather than animating the arm dropping back to the side of the avatar immediately, we apply a small delay, which better simulates sequences of inputs (e.g., typing, repeated scrolling). We note that our proof-of-concept implementation is single touch only, with the avatar's hand posed with a protruding index finger; however, animating multitouch gestures (e.g., pinch-to-zoom) should be possible if desired (see Figure 7).

3.8 Absolute "World" Position

So far, we have only discussed the relative spatial arrangement between body parts and orienting the body with respect to the gravity vector. Such data alone would allow Pose-on-the-Go to provide an

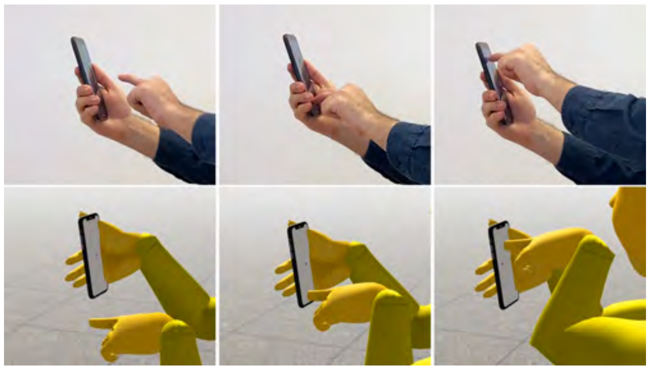
Figure 7: Pose-on-the-Go tracks finger position on the screen, which is reflected on the virtual phone and also poses the joints for the arm not holding the phone.

upper body pose, locked to a frontal view (e.g., fixed to the chest normal, allowing the head to look around). With the addition of absolute 6-DOF "world" tracking, several more expressive dimensions can be enabled, which we discuss in the next two sections.

One of the main reasons we selected iOS as our development platform was ARKit's best-in-class absolute 6-DOF tracking. Apple's hardware-accelerated, inside-out sensing implementation combines both visual odometry using the rear camera and data from the iPhone's IMU. This allows the phone to track its movement and position in 3D space. Since Pose-on-the-Go positions the head relative to the phone, the torso relative to the head, the arms relative to the torso (and so on), we can use the phone's spatial data to not only translate our avatar accurately in space, but also rotate the avatar to match the direction the user is facing.

3.9 Locomotion Mode & Leg Animation

The 6-DOF location of a user's body allows us to animate avatar locomotion, despite having no direct sensor data for the legs. To achieve this, we translate the avatar's upper body and run VRIK's solver, which contains animation support for bipedal locomotion (VRIK.solver.locomotion [57]). Stride length is a static parameter, and so to solve for different movement speeds, the leg animation is simply sped up or down, though this can produce unrealistic results. Additionally, small shifts in the user's posture and occasional position tracking errors even when the user is standing still can cause the IK solver to take errant steps.

To improve pose and animation quality, we leverage iOS's CMMotionActivity API [8], which provides the following locomotion mode predictions: stationary, walking, running, cycling, automotive and unknown, along with a confidence score. We found this prediction to a reliable filter, verses relying on motion alone to animate the legs. When the phone reports that a user is stationary, we do not animate the legs, except when turning the body. When walking, we set the stride length to that of a typical walk, while running requires a longer gait. By setting these parameters appropriately to a user's anthropometrics, a more realistic animation is achieved.

In the future, stride length could be set by the user, or perhaps automatically derived using absolute movement and the pedometer [9] features in iOS. Note that we did not create special kinematic logic for cycling or driving locomotion modes, but these could be implemented in the future to pose the avatar more accurately (e.g., sitting in a car, riding on a bike).

Although this process offers only a very coarse approximation of leg movements, when taken together with the full-body avatar, the output is reasonably convincing. Anecdotally, we found users to be quite forgiving – it may be that people do not attend to their absolute leg position (e.g., while walking) at the same level of scrutiny as their arms or head, where spatial errors are immediately commented upon.

Even with relaxed fidelity constraints, Pose-on-the-Go fails in two key ways. First and foremost, we do not know which leg leads in a bipedal movement; Pose-on-the-Go always steps with the right leg first, which can cause the whole step sequence to be 180° out of phase. Secondly, we cannot handle small leg movements, such as repositioning of the legs during conversations or taking a half step towards an object, as our stride length is fixed per locomotion mode in our current implementation. In a future implementation, this could be improved by using a dynamic stride length based on e.g., user velocity.

3.10 Sittings/Standing & Body Height

Locomotion animation primarily leverages a user's X/Y translation in an environment. However, the Z-axis (e.g., elevation) is useful in detecting and representing other poses, such as sitting and stepping onto objects (Figure 8). Elevation above the ground plane is not automatically provided in ARKit (though there are helper functions in ARWorldTracking). Instead, as a proof-of-concept implementation, we assume the phone starts in the user's hand while standing. As we have absolute 6-DOF position tracking, we can simply calculate if the head is above or below its starting height, and correspondingly rig the avatar with this constraint. In cases where the user is now higher than standing, we can infer the user has stepped up onto an object and we show the legs leaving the floor (Figure 8 far right). If the user is lower, the IK solver bends the legs constrained by the floor plane (Figure 8 center right). In general, this process is pretty crude as there are many ways to e.g., sit, squat, and kneel.

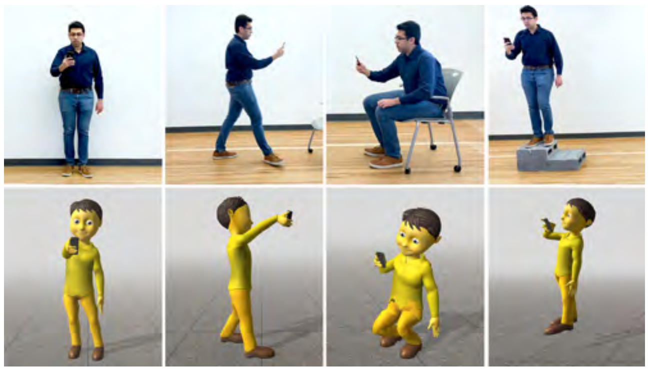
Figure 8: A user standing (far left), walking, (center left), sitting (center right) and standing on a box (far right) is mirrored by the avatar.

3.11 Data Synchronization

Our implementation is highly asynchronous, with a wide variety of sensors and processes running at different framerates. Most notably, our head tracking pipeline using the front camera runs at 15.5FPS, our depth camera-driven torso normal extraction runs at 11.8FPS, and the phone's 6-DOF spatial tracking runs at 19.9FPS. We set other low priority processes, such as reading the finger touchscreen position, to 5FPS. All of this data is passed to our IK solver running in Unity, which is set to run at 60FPS, matching the screen refresh rate.

Fusing many data streams of varying framerates requires some strategy for effective alignment. For our prototype implementation, a pool of threads asynchronously receive and store the most recently received frame of data for each sensor. Our main process loop, which runs at best effort given available CPU and GPU resources, simply uses the most recent data reported by all sensors. Although simple, we found this approach to be sufficient for our proof-of-concept system. An approach taking into account time-since-data-received could potentially extrapolate better live estimates of sensor values, and yield higher quality and lower-latency pose output – an avenue we leave to future work.

3.12 Framerate, Latency and Power Draw

As noted above, Unity (which runs IK, animates and renders the avatar) runs at 60 FPS, though the underlying sensor data arrives at a variety of frequencies. The best measure of end-to-end latency is "motion-to-photon", which is the time taken between a user's motion and when the device's pixels reflect that change. For this, we used a 240 FPS camera to record the user and screen in the same scene. We found an average motion-to-photon latency of 358ms (SD=63ms). Our implementation is very much a proof of concept, with minimal optimization, and thus these numbers should be viewed as a performance upper bound. For reference, Apple's Animoji feature [10], which digitizes people's faces as characters, runs with a latency of around ~150ms on an iPhone XR. We believe comparable performance should be achievable, certainly on future smartphones, which continue to make significant strides in compute power.

Pose-on-the-Go requires many sensors to be running, including three different cameras, which is energy-intensive. To quantify this, we measured the power draw with our iPhone XR on its home screen and also running our Pose-on-the-Go daemon. We found a power consumption delta of ~5.3W, which includes running all sensors and system processes like ARKit and CoreMotion. In practice, we found our iPhone XR could run continuously for ~2h, which closely matches what we would expect from the iPhone XR's 11.12Wh battery rating. Note that this power consumption number does not include running IK (not particularly intensive) or application graphics (highly variable depending on the app), which will add further burden.

4 OPEN SOURCE MODEL AND DATA

To enable other researchers and practitioners to build upon our system and study results, we have made our code and study data freely available at https://github.com/FIGLAB/PoseOnTheGo. We thank our participants for their permission to share this data.

5 ADDITIONAL CAPTURE DIMENSIONS

We note that our proof-of-concept feature set is not exhaustive, and future work could enable several additional dimensions of full-body capture on smartphones, further increasing fidelity and realism. We now briefly describe these avenues. In several cases, existing literature has already demonstrated feasibility.

5.1 Smartwatch for Other Arm Tracking

A natural limitation of Pose-on-the-Go is the inability to track the arm and hand not holding the phone or seen in the user-facing cameras. Instead, our system only animates this arm upon receiving touchscreen events, resulting in a very limited approximation. Fortunately, watches are often worn on the non-dominant arm, opposite to where most users would hold and use a phone one-handed. Thus, there is an opportunity for smartwatches to provide a complimentary stream of IMU data that would permit high-quality, six degree-of-freedom animation of this other arm [59]. In cases where the smartwatch is worn on the same arm that holds the phone, the data would still be useful for improved rigging of the wrist joint.

5.2 Finger Identification & Orientation

There is considerable prior work that has extended the capability of standard smartphone touchscreens, which could be put to use in Pose-on-the-Go. For instance, Le et al. [37] demonstrated finger-identification using the size and shape of finger blobs in the capacitive touch image. Using similar data, Mayer et al. [39] and Xiao et al. [70] demonstrated that finger pitch and yaw can be recovered. In TapSense [28], different parts of the finger, such as the knuckle or fingertip, could be recognized. All of the data above could be fed to Pose-on-the-Go's IK model to more faithfully depict the pose of the hands, which fingers are in use, and their angles of attack.

5.3 Facial Expression & Speech

There are now innumerable software libraries able to track the 2D or 3D geometry of a user's face with a camera [12, 34]. This capability is seen in commercial applications such as Apple's Animoji [10]. Such data could be used to enhance our avatar with realistic mouth and lip movements, both for facial expressions (e.g., smiles) and during speech. It may also be possible to render realistic mouth, lip and tongue movements by using the smartphone's microphone to capture a users speech [23, 29].

5.4 Appearance

As shown in systems such as MeCap [2], user-facing cameras can be used to capture one's personal appearance, such as hair style, glasses and apparel, allowing instant, real-world personalization of avatars, offering an alternative to manual editing through an avatar builder interface.

6 EVALUATION

We performed a series of studies, benchmarking Pose-on-the-Go against a professional-grade optical tracking system, which serves as a gold standard ground truth. Specifically, we used a Vicon system [67], with twelve MX40 cameras and four T160 cameras

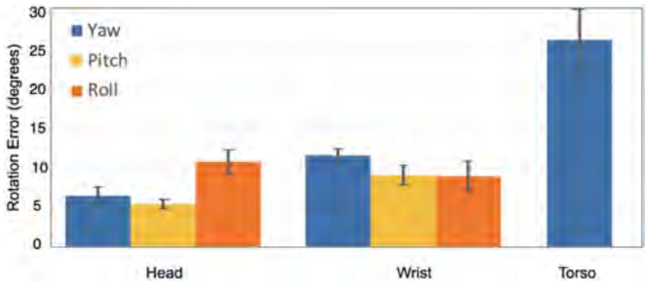

Figure 9: Rotational errors for head, phone and torso in degrees. Error bars are standard error.

capturing at 120FPS. We used Vicon Blade 3.2 for capture and data export and Vicon IQ 2.5 for data cleaning. Vicon's software and Pose-on-the-Go produce equivalent 3D joint data (14 joints: head, torso, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left foot and right foot), but in different coordinate systems. For analysis, we perform an extra post processing alignment step, designating the torso of both pose outputs as the body origin.

We recruited 8 participants for our user studies (2 female) with ages ranging from 18 to 39 years (M=27.6 years); all were right-handed. We asked our participants to complete a series of tasks, described in the rest of this section, in a randomized order. While completing the tasks, we purposely did not display the avatar on the phone or any screen, as participant's may have adapted their movements. The Vicon system and Pose-on-the-Go captured data simultaneously (participants wore MoCap suits while also holding a phone running Pose-on-the-Go), but run on two separate computers.

Lastly, even with synchronized system clocks (via NTP), data still required post hoc fine-grained synchronization. For this, we had users perform a calibration gesture at the start of each study. This calibration gesture had users extend their phone-holding arm directly forwards (parallel to ground), then rotating the arm up 45°, then down (−45°), then returning to straight ahead, and then from left to right (also −45° to +45°). After this, the users tapped the screen three times with the hand not holding the phone. Additionally, to isolate spatial accuracy separately from system latency, we dynamic time warp Pose-on-the-Go's output to the Vicon data stream with a maximum shift of ±300ms.

6.1 Head Orientation

We asked participants to face a wall 1.5m away with four markers arranged in a rectangle. We asked participants to hold the phone in front of them in a natural position and perform the following head movements five times in a row, using the markers as a positioning guide.

- Pan left and then right across the middle of the markers.
- Pan up and then down across the middle of the markers.
- Pan clockwise around the perimeter of the markers.
- Pan counterclockwise around the perimeter of the markers.
- Pan across the markers in a "figure eight" pattern.
- Roll the head to comfortable extremes.

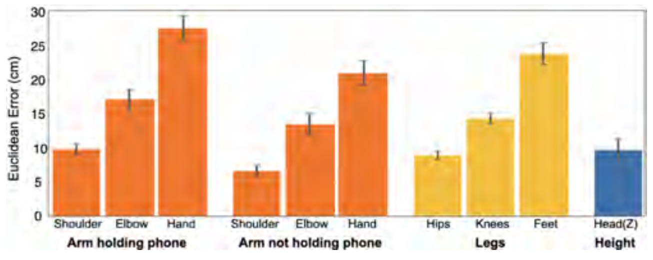

Figure 10: Euclidean distance error of Pose-on-the-Go for Head Height, Leg Pose, Arm (hand not holding phone) and the Arm (hand holding phone). Error bars are standard error.

These motions allowed us to evaluate the performance of Pose-on-the-Go's estimation of head yaw, pitch and roll. Across all participants, we found a mean yaw, pitch and roll angular error of 6.4° (*SD* = 3.0°), 5.4° (*SD* = 1.6°) and 10.7° (*SD* = 4.0°) respectively; see Figure 9.

6.2 Torso Orientation

In this task, we asked participants to rotate their chest left-to-right and back again (i.e., yaw), while trying their best to keep the phone held in the same position. This motion was completed five times per participant. From this data, we found Pose-on-the-Go deviated from our Vicon ground truth by a mean angular error of 26.1° (*SD* = 10.1°), see Figure 9. This poorer result was due to the chest leaving the view of the depth camera, precluding estimation (i.e., relying on IK interpolation alone). In post-hoc analysis, we found that there was a constant offset in the torso position (an unintended bug) that affected our IK solver's output. If we account for this, our mean angular error drops to 15.9° (*SD* = 10.1°).

6.3 Arm Pose (Hand Holding Phone)

To gauge the accuracy of arm joint tracking, we asked our participants to perform the following motions with the smartphone five times each:

- Move the phone away from the body, and then towards.
- Move the phone left-to-right and then right-to-left.
- Move the phone up and then down.
- Move the phone clockwise in a ~50cm circle.
- Move the phone counterclockwise in a ~50cm circle.
- Move the phone in a "figure eight" ~50cm tall.

With the data from this task, we computed the spatial error for participants' wrists, elbows and shoulders. We found that Pose-on-the-Go had a mean 3D euclidean error of 18.0cm (SD=3.0cm) across all joints. Broken out by joint (Figure 10), wrists had the greatest error of 27.4cm (SD=4.7cm), followed by elbows (M=17.0cm, SD=3.9cm), and shoulders (M=9.7cm, SD=2.1cm). Unsurprisingly, error increases as the IK solver tries to estimate joints farther along bone linkages (here, the torso is the body origin).

6.4 Hand Orientation (Hand Holding Phone)

We asked the participants to hold the phone in their dominant hand and rotate it left-to-right (yaw), up-and-down (pitch), and twist it

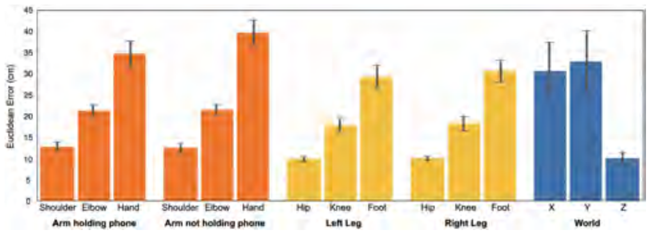

Figure 11: Pose-on-the-Go's euclidean distance error (across participants) in the "obstacle course" study. Error bars are standard error.

while keeping the screen facing them (roll), five time each using only their wrist (as best possible). From this data, we calculated a wrist angular joint error yaw, pitch and roll of 11.5° (*SD* = 2.3°), 9.1° (*SD* = 3.3°), and 8.9° (*SD* = 4.9°), respectively; see Figure 9.

6.5 Arm Pose (Hand Not Holding Phone)

We asked our participants to use their non-dominant hand (in our case, all participants used their left hand) to touch the screen five times in a row, simulating typing, and then drop their arm to a resting position. This process was repeated five times, for a total of 25 screen taps. On this data, Pose-on-the-Go achieved an average 3D euclidean error of 20.9cm (SD=5.1cm), 13.4cm (SD=4.4cm), and 6.6cm (SD=1.2cm) for the wrist, elbow and shoulder joints respectively; see Figure 10.

6.6 Leg Pose (Sitting/Standing)

In this task, we placed a chair and 25cm tall pedestal into the Vicon tracking area, 2.5m apart. We asked participants to sit in the chair for a few seconds, then stand up and walk over to the pedestal, and finally to step onto the pedestal. This procedure was repeated five times and provided a wide variety of poses to evaluate the quality of our leg posing.

There was no significant difference between left and right leg accuracy, and so we combined results. We found a mean 3D euclidean error of 8.9cm (SD=1.6cm), 14.2cm (SD=2.3cm), and 23.7cm (SD=3.1cm) for the hips, knees and feet respectively (see Figure 10). This follows a similar trend as the arms, where the IK solver gets increasingly errorful father along bone linkages. Nonetheless, given we have no direct data for any of these joints, we view this result as very promising.

6.7 World Tracking

All of the analyses discussed so far have reported results in body coordinates (torso as the origin). However, Pose-on-the-Go also tracks and animates avatars moving around the world, and so we can evaluate the fidelity of absolute "world" position tracking. For data collection, we asked participants to walk the following paths continuously for 20 seconds each (with the aid of floor markers).

- Walk forwards 2m, then backwards 2m (repeatedly).
- Sidestep left 2m and then right 2m (repeatedly).
- Walk in a circle (2m diameter).
- Walk the perimeter a 2 × 2 meter square.
- Walk in "figure eight" pattern in a 4 × 2m area.

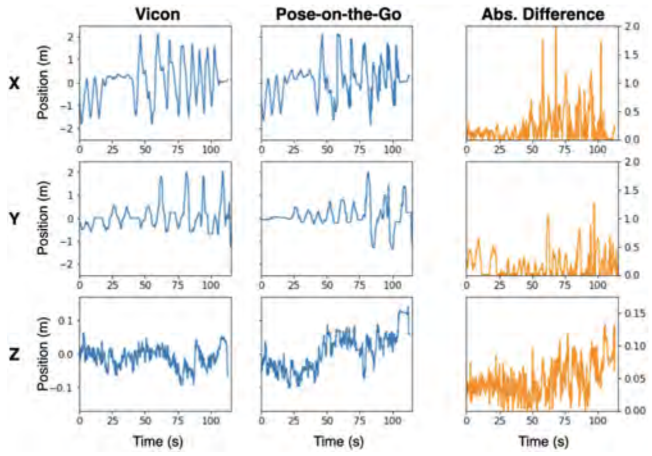

Figure 12: Comparison of Vicon tracking vs. Pose-on-the-Go for one participant's 3-DOF position in the "obstacle course" study. Note that spikes in the absolute difference plots are chiefly due to latency and not true spatial error.

We use our avatar's torso point for analysis. Across all walked paths and participants, we found a mean 3D world euclidean error of 11.2cm (SD=5.7cm). Across X, Y and Z axes individually (Z is up), the mean euclidean errors are 11.6cm (SD=6.0cm), 15.2cm (SD=9.5cm) and 6.9cm (SD=3.9cm) respectively.

6.8 Head Height Tracking (Z)

Our "world" tracking procedure described in the previous section did not vary the height of the user (other than natural variation during locomotion). For this reason, the height of the head varied by ~20cm over the entire data collection period. To more fully evaluate head height, we instead use the data collected in our previous Leg Pose (Sitting/Standing) study, where users were asked to sit and step onto a pedestal. Using this data, but now with absolute tracking, we found a mean head euclidean Z-axis error of 9.6cm (SD=4.6cm); see Figure 10.

6.9 Locomotion

We use the data from our body "world" tracking study to estimate the fidelity of Pose-on-the-Go's locomotion animation. Rather than spatially compare every step taken, which we previously noted can be 180 degrees out of phase, we instead evaluate the movement sequence holistically to see if the avatar is locomoting in a faithful way. For this, we compare the number of steps the user took in reality vs. number of steps taken by the Pose-on-the-Go avatar. Across all participants, our average step count error is 6.3% (SD=2.74%). In other words, if a user takes 100 steps, their avatar will have been animated taking 6.3% more of fewer steps on average.

6.10 Obstacle Course

All of the prior studies focused on different sets of joints in purposely designed tasks. To test the efficacy of our full-body motion capture system, we conducted a less constrained study where we asked participants to walk in a 4 × 2m area in a "figure eight"

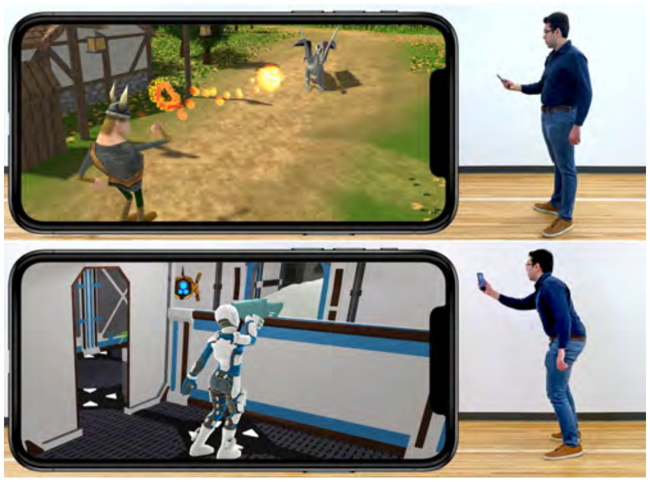
Figure 13: We created two examples games that utilize Pose-on-the-Go. Top: A fantasy game, where users can fight with swords and cast spells using arm movements. Bottom: a futuristic third-person shooter, where a user can physically run and duck, while using their phone to aim and shoot a laser weapon.

repeatedly. During this we asked them to look at different targets randomly and occasionally asked them to sit down, stand on the pedestal and touch their screens (akin to an "obstacle course" methodology, see e.g., [13, 32]. We then calculated participants' full body 3D euclidean error and also their world position error. Since they were walking and the legs were not in sync, we also calculated the number of times their legs were out of phase and corrected for it by matching the appropriate leg.

In this task, our average Euclidean error is 20.9cm (SD=2.6cm) across all participants and all joints. We found that the legs were out of phase 39.8% of the time (which we corrected for when computing leg joint error). Our world tracking accuracy broken out by X, Y and Z axes is 30.5cm (SD=17.7cm), 32.8cm (SD=19.0cm) and 10cm (SD=3.3cm) respectively. Per joint results are provided in Figure 11. Additionally, a representative example of positional data for one participant (captured synchronously by both systems) is shown in Figure 12.

7 EXAMPLE USES

Applications that could utilize information about a user's full-body pose are incredibly diverse, as seen in the HCI literature (see Related Work) and the 100+ Xbox titles that used the Kinect sensor [41]. To illustrate the utility and feasibility of Pose-on-the-Go, we selected three use domains – gaming, social apps, and health – and created two proof-of-concept demo apps for each category. Please also refer to our Video Figure.

7.1 Gaming

The first demo application we created was a third-person fantasy game (Figure 13 top). The user can select among different weapons by tapping on the screen, which are controlled through arm motion. For example, a sword can be swung in the air, using the phone as a proxy for the hilt. We also included a magic wand, where various spells can be cast by tracing different paths in the air (e.g., a circle to cast a fireball). In this example app, walking forwards/backwards and turning left/right could be controlled on the phone using up/down and left/right swipes on the phone's screen, allowing the user to stand or sit without moving their body.

For our second game — a third-person futuristic shooter — we utilize the user's actual absolute body position and pose (Figure 13 bottom), allowing them to physically walk around the virtual environment, and also duck behind cover to avoid enemy fire. In this game, the phone is a proxy for a handheld laser, allowing the user to both aim and shoot with their arm.

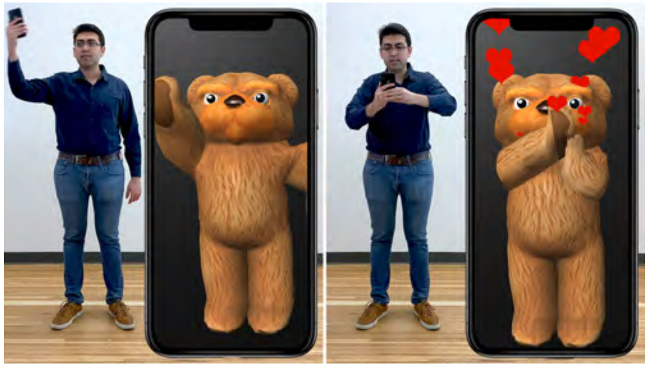
Figure 14: An example of full-body "Animoji" enabled by Pose-on-the-Go, where users can wave (left) and hug (right).

7.2 Social Apps

There are many smartphone apps that capture a user's face and digitally transform or augment it for social purposes, such as Apple's Animoji [10] and Snapchat's Lenses [61]. The latter software is also an example of full-body AR augmentation, through this app augments other users (i.e., captured through the rear facing camera) rather than the holder of the phone. Such AR augmentation and "avatarization" has value in both entertainment and professional collaborative contexts [14, 15, 52].

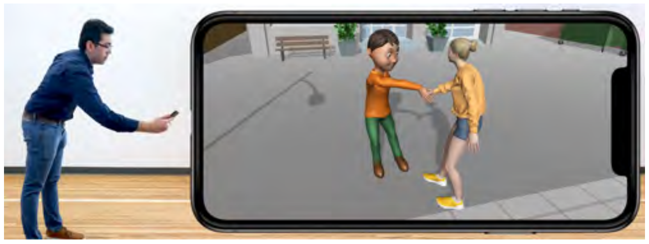
Figure 15: Pose-on-the-Go could also be used to enhance communication expressivity in social and collaborative apps, helping to convey body language and gestures, such as handshakes, seen here.

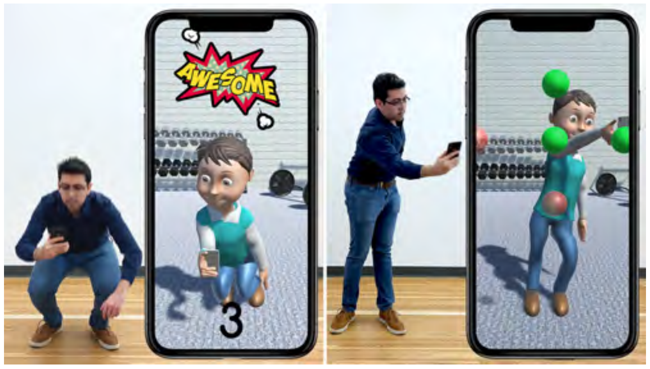

Figure 16: Two health-related example applications, one that counts squats (left) and another that lays out reaching targets (right).

With Pose-on-the-Go, we can immediately extend Animoji-style face capture with full-body versions. As an example, we created a bear avatar that can wave, hug, jump, and make a wide variety of other expressive, full-body motions (Figure 14). Similarly, mobile AR teleconferencing (Figure 15) could be greatly enhanced with avatars that can better convey body language (e.g., looking down at the floor, shrugging) and also allowing for social gestures (e.g., waves and handshakes). Our demo app also allowed users to walk around a shared virtual space, and users could realistically turn their eyes, heads and bodies towards different targets, providing an medium for effects such as proxemics [26] to play out.

7.3 Health

Finally, there are many examples in fitness and rehabilitation where capturing a user's pose could be especially valuable. As a proof of concept, we created two applications. The first is a fitness app (Figure 16 right), where exercises like squats and lunges could be counted or timed, as well as evaluated for quality (e.g., squat height, lunge distance). As a second example, we created an app that presents a series of targets that the user must reach with their hands and maintain balance (Figure 16 left), which could be part of a larger physical therapy regiment.

8 LIMITATIONS

It is important to note that Pose-on-the-Go is only an estimation of full-body pose, and it not meant to compete with high-accuracy tracking systems (e.g. Vicon), which are used in film post-production and similar uses requiring high fidelity. Indeed, this is the inherent tradeoff that comes with building a system that requires no new, special or external sensors, and instead attempts to maximize use of the data already available to it. Nonetheless, Pose-on-the-Go's accuracy should be sufficient for a wide range of casual gaming and social apps. That said, there are several areas where the system falls short, which are worth reviewing, pointing to potential future work.

First is incomplete data about the body, most notably the arm not holding the phone, which is almost never digitized except when the user touches the screen. This potentially could be remedied with the advent of wider field-of-view, user-facing cameras, which might better capture the elbow, or at least the upper arm with respect to the shoulder joint. The hand also might occasionally appear. The other body location we can only loosely approximate are the legs, animated based on absolute body motion and fixed stride lengths derived from a locomotion mode prediction. This is sufficient to animate the avatar, but is not true body capture. For this reason, walking animations can be totally out of leg phase from reality. Other joints, such as the hips and knees are entirely estimated by the IK solver, and are essentially an interpolation between other known pose data.

Latency is another limitation of our current implement. At ~350ms, it starts to degrade the realism of the full-body tracking. Experiences would have to be designed to accommodate such latency, an issue also faced in many Xbox Kinect titles. In particular, the arm not holding the phone can only start animating once a finger touches the screen and has lag closer to one second (when animation is applied) with no immediate avenue for improvement. Of course, even commercial systems have some lag – Apple's Animoji feature, which also fuses data from RGB and depth user-facing cameras, has a latency of ~150ms.

Another limitation is computational complexity and power demand. This will require significant optimization efforts, but we do believe it is possible as demonstrated by Animoji-type features found on many smartphones today, as well as pose tracking of other users (via the rear camera) in e.g., PoseNet [49], SnapChat [61], and ARKit [7]. These processes are also heavyweight, but have been highly engineered and take advantage of hardware acceleration in order to run at interactive speeds. Of course, Pose-on-the-Go has to contend with more sensors, including all three iPhone cameras, which impacts battery life when running applications utilizing full-body pose features. However, as noted earlier, we still get ~2 hours of battery life with our current implementation.

9 CONCLUSION

We have presented our work on Pose-on-the-Go, a sensor fusion approach that allows smartphones to estimate their owners' full-body pose using only internal sensors. We benchmark our pose tracking output against a "hollywood-grade" optical tracking system requiring retroreflective markers. Our results show that we can resolve most joints to within 25cm in 3D space, including the feet, despite typically having no direct sensor data below the chest. While only a coarse estimation of body pose, it nonetheless opens up new and interesting whole-body applications, ranging from mobile AR games to more expressive social and collaborative interactions.

REFERENCES

[1] Karan Ahuja, Mayank Goel, and Chris Harrison. 2020. BodySLAM: Opportunistic User Digitization in Multi-User AR/VR Experiences. In *Symposium on Spatial User Interaction* (Virtual Event, Canada) *(SUI '20)*. Association for Computing Machinery, New York, NY, USA, Article 16, 8 pages. https://doi.org/10.1145/3385959.3418452
[2] Karan Ahuja, Chris Harrison, Mayank Goel, and Robert Xiao. 2019. MeCap: Whole-Body Digitization for Low-Cost VR/AR Headsets. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 453–462. https://doi.org/10.1145/3332165.3347889
[3] Karan Ahuja, Andy Kong, Mayank Goel, and Chris Harrison. 2020. Direction-of-Voice (DoV) Estimation for Intuitive Speech Interaction with Smart Devices

Ecosystems *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 1121–1131. https://doi.org/10.1145/3379337.3415588

[4] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '18)*. IEEE, 7297–7306. https://doi.org/10.1109/CVPR.2018.00762

[5] ALT LLC. 2020. Antilatency. Retrieved 2020 from https://antilatency.com/

[6] Raphael Anderegg, Loïc Ciccone, and Robert W. Sumner. 2018. PuppetPhone: Puppeteering Virtual Characters Using a Smartphone. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/3274247.3274511

[7] Apple Inc. 2020. Apple Developer - ARFaceAnchor. Retrieved 2020 from https://developer.apple.com/documentation/arkit/arfaceanchor

[8] Apple Inc. 2020. Apple Developer - CoreMotion Activity. Retrieved 2020 from https://developer.apple.com/documentation/coremotion/cmmotionactivity

[9] Apple Inc. 2020. Apple Developer - CoreMotion Pedometer. Retrieved 2020 from https://developer.apple.com/documentation/coremotion/cmpedometerdata

[10] Apple Inc. 2020. Support - Animoji. Retrieved 2020 from https://support.apple.com/en-au/HT208190

[11] Teo Babic, Florian Perteneder, Harald Reiterer, and Michael Haller. 2020. Simo: Interactions with Distant Displays by Smartphones with Simultaneous Face and World Tracking. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3334480.3382962

[12] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. 2018. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG '18)*. IEEE, 59–66. https://doi.org/10.1109/FG.2018.00019

[13] Ling Bao and Stephen S Intille. 2004. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*. Springer, 1–17.

[14] Steve Benford, John Bowers, Lennart E. Fahlén, Chris Greenhalgh, and Dave Snowdon. 1995. User Embodiment in Collaborative Virtual Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 242–249. https://doi.org/10.1145/223904.223935

[15] Barry Brown and Marek Bell. 2004. CSCW at Play: 'there' as a Collaborative Virtual Environment. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (Chicago, Illinois, USA) *(CSCW '04)*. Association for Computing Machinery, New York, NY, USA, 350–359. https://doi.org/10.1145/1031607.1031666

[16] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR '17)*. IEEE, 7291–7299. https://doi.org/10.1109/CVPR.2017.143

[17] Ke-Yu Chen, Shwetak N. Patel, and Sean Keller. 2016. Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1504–1514. https://doi.org/10.1145/2858036.2858125

[18] Weiya Chen, Chenchen Yu, Chenyu Tu, Zehua Lyu, Jing Tang, Shiqi Ou, Yan Fu, and Zhidong Xue. 2020. A Survey on Hand Pose Estimation with Wearable Sensors and Computer-Vision-Based Methods. *Sensors* 20, 4 (2020), 1074. https://doi.org/10.3390/s20041074

[19] Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott Hudson. 2014. Around-Body Interaction: Sensing & Interaction Techniques for Proprioception-Enhanced Input with Mobile Devices. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services* (Toronto, ON, Canada) *(MobileHCI '14)*. Association for Computing Machinery, New York, NY, USA, 287–290. https://doi.org/10.1145/2628363.2628402

[20] Amit Das, Ivan Tashev, and Shoaib Mohammed. 2017. Ultrasound based gesture recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '17)*. IEEE, 406–410. https://doi.org/10.1109/ICASSP.2017.7952187

[21] Muybridge Eadweard. 1878. The Horse in Motion.

[22] Facebook Technologies LLC. 2020. Oculus Quest. Retrieved 2020 from https://www.oculus.com/quest

[23] Bo Fan, Lei Xie, Shan Yang, Lijuan Wang, and Frank K Soong. 2016. A deep bidirectional LSTM approach for video-realistic talking head. *Multimedia Tools and Applications* 75, 9 (2016), 5287–5309.

[24] Eric Foxlin and Michael Harrington. 2000. WearTrack: a self-referenced head and hand tracker for wearable computers and portable VR. In *Digest of Papers. Fourth International Symposium on Wearable Computers (ISWC '00)*. IEEE, 155–162. https://doi.org/10.1109/ISWC.2000.888482

[25] Sehoon Ha, Yunfei Bai, and C. Karen Liu. 2011. Human Motion Reconstruction from Force Sensors. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vancouver, British Columbia, Canada) *(SCA '11)*. Association for Computing Machinery, New York, NY, USA, 129–138. https://doi.org/10.1145/2019406.2019424

[26] Edward Twitchell Hall. 1962. *Proxemics: The study of man's spatial relations.*

[27] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. 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, 441–450. https://doi.org/10.1145/2047196.2047255

[28] Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011. TapSense: Enhancing Finger Interaction on Touch Surfaces. 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, 627–636. https://doi.org/10.1145/2047196.2047279

[29] Gregor Hofer, Junichi Yamagishi, and Hiroshi Shimodaira. 2008. Speech-driven lip motion generation with a trajectory HMM. (2008).

[30] Notch Interfaces Inc. 2020. Notch Interfaces. Retrieved 2020 from https://wearnotch.com/

[31] Intel Corporation. 2020. RealSense. Retrieved 2020 from https://www.intelrealsense.com/

[32] Stephen S. Intille, Ling Bao, Emmanuel Munguia Tapia, and John Rondoni. 2004. Acquiring in Situ Training Data for Context-Aware Ubiquitous Computing Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/985692.985693

[33] Haojian Jin, Zhijian Yang, Swarun Kumar, and Jason I. Hong. 2018. Towards Wearable Everyday Body-Frame Tracking Using Passive RFIDs. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 145 (Jan. 2018), 23 pages. https://doi.org/10.1145/3161199

[34] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR '14)*. IEEE Computer Society, USA, 1867–1874. https://doi.org/10.1109/CVPR.2014.241

[35] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: Freehand 3D Interactions Anywhere Using a Wrist-Worn Gloveless Sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 167–176. https://doi.org/10.1145/2380116.2380139

[36] Daehwa Kim, Keunwoo Park, and Geehyuk Lee. 2020. OddEyeCam: A Sensing Technique for Body-Centric Peephole Interaction Using WFoV RGB and NFoV Depth Cameras. 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, 85–97. https://doi.org/10.1145/3379337.3415889

[37] Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the Feasibility of Finger Identification on Capacitive Touchscreens Using Deep Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 637–649. https://doi.org/10.1145/3301275.3302295

[38] Mingyang Li and Anastasios I Mourikis. 2013. 3-D motion estimation and online temporal calibration for camera-IMU systems. In *2013 IEEE International Conference on Robotics and Automation (ICRA '13)*. IEEE, IEEE, 5709–5716. https://doi.org/10.1109/ICRA.2013.6631398

[39] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 220–229. https://doi.org/10.1145/3132272.3134130

[40] Meta Motion. 2018. Gypsy Motion Capture System. Retrieved 2021 from http://metamotion.com/gypsy/gypsy-motion-capture-system.htm

[41] Microsoft Corporation. 2010. Microsoft Kinect. Retrieved 2021 from https://en.wikipedia.org/wiki/Kinect

[42] Microsoft Corporation. 2010. Microsoft Kinect Games. Retrieved 2021 from https://en.wikipedia.org/wiki/Category:Kinect_games

[43] Microsoft Corporation. 2019. HoloLens. Retrieved 2021 from https://www.microsoft.com/en-us/hololens

[44] Nathan Miller, Odest Chadwicke Jenkins, Marcelo Kallmann, and Maja J Mataric. 2004. Motion capture from inertial sensing for untethered humanoid teleoperation. In *4th IEEE/RAS International Conference on Humanoid Robots (ICHR '04, Vol. 2)*. IEEE, 547–565. https://doi.org/10.1109/ICHR.2004.1442670

[45] NaturalPoint Inc. 2020. OptiTrack. Retrieved 2020 from http://optitrack.com

[46] Seungtak Noh, Hui-Shyong Yeo, and Woontack Woo. 2015. An HMD-based Mixed Reality System for Avatar-Mediated Remote Collaboration with Bare-hand Interaction. In *International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments (ICAT-EGVE '15)*. The Eurographics Association, 61–68. https://doi.org/10.2312/egve.20151311

[47] Northern Digital Inc. 2020. trakSTAR. Retrieved 2020 from https://www.ndigital.com/msci/products/drivebay-trakstar/

[48] OpenNI. 2020. OpenNI. Retrieved 2020 from https://structure.io/openni

[49] 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 *Proceedings of the European Conference on Computer Vision (ECCV '18)*. 269–286. https://doi.org/10.1007/978-3-030-01264-9_17

[50] Mathias Parger, Joerg H. Mueller, Dieter Schmalstieg, and Markus Steinberger. 2018. Human Upper-Body Inverse Kinematics for Increased Embodiment in Consumer-Grade Virtual Reality *(VRST '18)*. Association for Computing Machinery, New York, NY, USA, Article 23, 10 pages. https://doi.org/10.1145/3281505.3281529

[51] PhaseSpace Inc. 2020. PhaseSpace. Retrieved 2020 from https://phasespace.com/

[52] Thammathip Piumsomboon, Gun A. Lee, Jonathon D. Hart, Barrett Ens, Robert W. Lindeman, Bruce H. Thomas, and Mark Billinghurst. 2018. Mini-Me: An Adaptive Avatar for Mixed Reality Remote Collaboration. 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, 1–13. https://doi.org/10.1145/3173574.3173620

[53] Polhemus. 2020. Polhemus. Retrieved 2020 from https://polhemus.com/case-study/detail/polhemus-motion-capture-system-is-used-to-measure-real-time-motion-analysis

[54] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. 2016. EgoCap: Egocentric Marker-Less Motion Capture with Two Fisheye Cameras. *ACM Trans. Graph.* 35, 6, Article 162 (Nov. 2016), 11 pages. https://doi.org/10.1145/2980179.2980235

[55] Thiago Braga Rodrigues, Ciarán Ó Catháin, Declan Devine, Kieran Moran, Noel E O'Connor, and Niall Murray. 2019. An Evaluation of a 3D Multimodal Marker-Less Motion Analysis System. In *Proceedings of the 10th ACM Multimedia Systems Conference* (Amherst, Massachusetts) *(MMSys '19)*. Association for Computing Machinery, New York, NY, USA, 213–221. https://doi.org/10.1145/3304109.3306236

[56] Grégory Rogez, Maryam Khademi, JS Supančič III, Jose Maria Martinez Montiel, and Deva Ramanan. 2014. 3D hand pose detection in egocentric RGB-D images. In *European Conference on Computer Vision*. Springer, 356–371. https://doi.org/10.1007/978-3-319-16178-5_25

[57] Root Motion. 2020. FINAL IK - VRIK Solver Locomotion. Retrieved 2020 from http://www.root-motion.com/finalikdox/html/page16.html

[58] Root Motion. 2020. Root Motion. Retrieved 2020 from http://root-motion.com/

[59] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I Am a Smartwatch and I Can Track My User's Arm. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services* (Singapore, Singapore) *(MobiSys '16)*. Association for Computing Machinery, New York, NY, USA, 85–96. https://doi.org/10.1145/2906388.2906407

[60] Takaaki Shiratori, Hyun Soo Park, Leonid Sigal, Yaser Sheikh, and Jessica K. Hodgins. 2011. Motion Capture from Body-Mounted Cameras. In *ACM SIGGRAPH 2011 Papers*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1964921.1964926

[61] Snap Inc. 2020. Snapchat Lenses. Retrieved 2020 from https://lensstudio.snapchat.com/lenses/

[62] Ivan E. Sutherland. 1968. A Head-Mounted Three Dimensional Display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I* (San Francisco, California) *(AFIPS '68 (Fall, part I))*. Association for Computing Machinery, New York, NY, USA, 757–764. https://doi.org/10.1145/1476589.1476686

[63] Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. 2011. Motion Reconstruction Using Sparse Accelerometer Data. *ACM Trans. Graph.* 30, 3, Article 18 (May 2011), 12 pages. https://doi.org/10.1145/1966394.1966397

[64] Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. 2019. xr-egopose: Egocentric 3d human pose from an hmd camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV '19)*. IEEE, 7728–7738. https://doi.org/10.1109/ICCV.2019.00782

[65] Unity Technologies. 2020. Unity. Retrieved 2020 from https://unity.com/

[66] Verhaert. 2020. Verhaert. Retrieved 2020 from https://verhaert.com/

[67] Vicon Motion Systems Ltd. 2020. Vicon. Retrieved 2020 from https://vicon.com/

[68] Vive. 2020. HTC VIVE. Retrieved 2020 from https://www.vive.com/

[69] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. 2007. Practical Motion Capture in Everyday Surroundings. *ACM Trans. Graph.* 26, 3 (July 2007), 35–es. https://doi.org/10.1145/1276377.1276421

[70] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (Madeira, Portugal) *(ITS '15)*. Association for Computing Machinery, New York, NY, USA, 47–50. https://doi.org/10.1145/2817721.2817737

[71] Xsens. 2020. Motion Capture. Retrieved 2020 from https://www.xsens.com/motion-capture

[72] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. 2019. Mo2Cap2: Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera. *IEEE Transactions on Visualization and Computer Graphics* 25, 5 (2019), 2093–2101. https://doi.org/10.1109/TVCG.2019.2898650

[73] Yasuyoshi Yokokohji, Yuki Kitaoka, and Tsuneo Yoshikawa. 2005. Motion capture from demonstrator's viewpoint and its application to robot teaching. *Journal of Robotic Systems* 22, 2 (2005), 87–97. https://doi.org/10.1002/rob.20050

[74] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. 2018. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '18)*. IEEE, 7356–7365. https://doi.org/10.1109/CVPR.2018.00768