LightAnchors: Appropriating Point Lights for Spatially-Anchored Augmented Reality Interfaces

Karan Ahuja **Sujeath Pareddy** **Robert Xiao** **Mayank Goel** **Chris Harrison**
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213
{kahuja, spareddy, brx, mayank, chris.harrison}@cs.cmu.edu

ABSTRACT
Augmented reality requires precise and instant overlay of digital information onto everyday objects. We present our work on LightAnchors, a new method for displaying spatially-anchored data. We take advantage of pervasive point lights – such as LEDs and light bulbs – for both in-view anchoring and data transmission. These lights are blinked at high speed to encode data. We built a proof-of-concept application that runs on iOS without any hardware or software modifications. We also ran a study to characterize the performance of LightAnchors and built eleven example demos to highlight the potential of our approach.

Author Keywords
Augmented Reality; Smartphones, Tags, Markers, Visible Light Communication; Mobile Interaction.

CCS Concepts
Human-centered computing → Human computer interaction (HCI) → Interaction paradigms → Mixed / augmented reality

INTRODUCTION
Augmented reality (AR) allows for the overlay of digital information and interactive content onto scenes and objects. In order to provide tight registration of data onto objects in a scene, it is most common for markers to be employed. Various visual tagging strategies have been investigated in both academia and industry (e.g., retroreflective stickers, barcodes, ARToolKit markers [15], ARTags [7], AprilTag [31], QR Codes [14], and ArUco markers [29]).

In this paper, we present LightAnchors, a new method to display spatially-anchored data in augmented reality applications. Unlike most prior tracking methods, which instrument objects with markers (often large and/or obtrusive), we take advantage of point lights already found in many objects and environments. For example, most electrical appliances now feature small (LED) status lights, and light bulbs are common in indoor and outdoor settings. In addition to leveraging these point lights for in-view anchoring (i.e., attaching information and interfaces to specific objects), we also co-opt these lights for data transmission, blinking them rapidly to encode binary data.

UIST '19, October 20-23, 2019, New Orleans, LA, USA

ISBN 978-1-4503-6816-2/19/10 $15.00
https://doi.org/10.1145/3332165.3347884

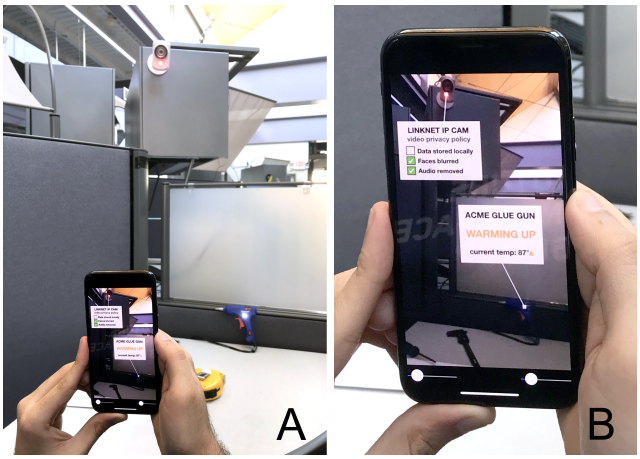

Figure 1. With LightAnchors, a security camera's LED can be used to share its privacy policy, while a hot glue gun transmits its live temperature (close-up of screen in B).

Another difference from conventional markers is that LightAnchors can transmit dynamic payloads, without the need for Wi-Fi, Bluetooth or indeed, any connectivity. Devices need only an inexpensive microcontroller (e.g., [22], which costs less than $0.50 USD) with the ability to blink a LED. This could allow "dumb" devices to become smarter through AR with minimal extra cost (much less than e.g., adding a screen to a device). For example, we created a glue gun that transmits its live temperature (Figure 1). For devices that already contain a microprocessor, LightAnchors opens a new information outlet in AR, for example, the LED found in many security cameras could be used to share the device's privacy policy (Figure 1).

As smartphones are the most pervasive AR platform at present, we created a proof-of-concept LightAnchors implementation for iOS. This requires no special hardware or operating system modifications, and simply takes advantage of high-speed cameras that have shipped on recent smartphone models (up to 240 frames per second on the iPhone 7 and later models). In addition to describing our algorithm, we also report the findings of a transmission performance study, which tested accuracy at different distances, with two light sizes, and while held still and in motion. We conclude by describing 11 example applications we built to illustrate the potential of LightAnchors.

RELATED WORK

LightAnchors overlaps with several disparate literatures, including marker-based tracking approaches, marker-less computer vision techniques, and visible light communication. We now briefly review these research areas.

Fiducial Markers

There are a wide variety of successful fiducial marking schemes. For example, ARTags [15] use black-and-white 2D patterns that allow conventional cameras to read a data payload and also estimate 3D position/orientation of the tag. Other popular schemes include QR Codes [14], April Tags [31] and ArUco markers [29]. These printed tags are highly visible, and thus often obtrusive to the visual design of objects. In consumer devices, tags are often placed out of sight (bottom or rear of devices), which precludes immediate use in AR applications. To make tags less obtrusive, researchers have explored embedding subtle patterns into existing surfaces, such as floors and walls [30].

Light-Based Markers

Many light-based schemes have been previously considered. For example, IRCube [11] studded a device with infrared LEDs and demonstrated position and orientation tracking. Bokodes [23] used tiny lenslet-covered illuminated tags that can be resolved when imaged by an out-of-focus camera. Markers or patterns can also be projected onto the environment, such as dot patterns [32] and m-sequences [35]. All of these approaches require special hardware and do not demonstrate dynamic data payloads.

Rarer are systems that support dynamic payloads. For example, SideBySide [33] digitally projected infrared ARTags onto the environment, which could e.g., identify users in multiplayer projected AR games. In CapCam [36], data is presented as colored sequences that were read by phones' rear-facing cameras when placed on a screen. Grundhöfer et al. [8] used a 120Hz digital screen with a synchronized camera to capture fiducial tags that appeared on specific frames. Prakash [28] inverted this approach, projecting a structured light pattern onto photosensors, which track themselves within the projected volume.

LightAnchors is closer in spirit to approaches that use active point lights as markers. ID CAM [21] proposed using LEDs as beacons, blinking at 4 kHz and sensed with a specially-designed high-speed camera. Similar approaches have been used to track drones [2] and capture human motion [12]. To support tracking of many tags in a scene, [12] synchronized LEDs with RF communication. Lastly, [4] describes a smartphone-based system that tracks LEDs using the camera and demonstrates transmission speeds of ~1 bit/sec for enhanced interactions with toys.

Marker-Less Strategies

Augmented reality systems can also track objects using innate features. Today, there are many object-recognition computer vision libraries that can track objects without special tags or markings (e.g., [9]). Snap-To-It [6] uses such capability to recognize objects using a smartphone camera, after which interactivity can be offered (see also the commercial Vuforia augmented reality software toolkit [24]). However, these systems require some form of preregistration of the to-be-recognized object or scene, and the object itself cannot transfer information beyond its identity.

Visible Light Communication with Commodity Devices

With specialized equipment, it is possible to transmit data at high speeds with visible light communication (VLC). For instance, the IEEE 802.15.7 standard [26] allows LEDs to transmit data at up to 96 Mbit/s using specialized photosensors to form line-of-sight wireless networks. More relevant to LightAnchors are VLC-style techniques that use commodity cameras, for e.g., room-level localization [25]. These VLC systems most often use diffuse or ambient modulated light and the rolling shutter operation of cameras to receive data faster than the full-frame rate of the camera [5, 13]. However, this approach does not work as well for point light sources (which only cover a small portion of the image at typical distances), nor when many lights are active in a scene.

IMPLEMENTATION

At a high level, for every incoming frame of video, our algorithm creates an image pyramid, such that lights – big or small, close or far – are guaranteed to be contained within a single pixel at least one level. Our algorithm then searches for candidate light anchors using a max-pooling template that finds bright pixels surrounded by darker pixels. We then track candidate anchors over frames, decoding a blinked binary pattern using an adaptive threshold. To drop false positive detections, only candidates with the correct preamble are accepted, after which their data payloads can be decoded. This process allows us to robustly track and decode multiple LightAnchors simultaneously.

Encoding & Point Lights

We encode all data as a binary sequence, prefixed with a known pattern. Since we repeatedly transmit the same message, the prefix appears at both the beginning and end of every transmission, which makes payload segmentation straightforward. We modulate lights with this pattern between high and low intensities at 120 FPS using a microcontroller (Teensy 3.6 or Arduino Mega) and its digital-to-analog converter (DAC). This frequency is at the human flicker fusion threshold, and the flashing is generally imperceptible, but depends on the particular payload.

Unlike prior approaches that synchronize light modulation with e.g., RF triggers [12], our lights and smartphones are unsynchronized. This means it is possible for the camera shutter to align with transitions in our blinked pattern, which at best reduces SNR, and at worse, means the pattern is unresolvable. To recover from this type of failure, we phase shift our transmitted signal by 36° after each transmission. We used basic binary transmission as a proof of concept, but LightAnchors could also be extended to use multiple illumination levels and colors.

Frame Capture

In our proof-of-concept iOS app, we use the AVCaptureSession API to grab video frames and OpenCV for image processing. We enqueue all video frames, which are consumed asynchronously by our detection-tracking-decoding thread (described in subsequent sections). Our software runs on the iPhone 7 or later, which can capture video frames at 240 FPS/720p. In general, this is too much data for our current implementation to process in real time at high resolutions, and so we drop frames and down sample to an operating resolution. Only at 320x180 can we process a 240 FPS image stream. When frames are scaled, we use iOS's optimized CoreGraphics API.

Detection

Our LightAnchor detection algorithm is designed to have high recall. Given a raw camera image, we first convert to grayscale and build an image pyramid (five layers, scaling by half). We model LightAnchors as bright spots surrounded by darker regions. Specifically, for each pixel, we compute the difference between the center pixel value and the maximum value of all pixels in a 4×4 diamond perimeter. We then threshold this result at every pixel and at every pyramid level, which produces an array of candidate anchors for each incoming frame of video. Finally, we flatten results from all pyramid layers so that candidate anchors are in the coordinate space of the highest resolution pyramid.

Tracking

Our detection process passes all candidate anchors to our tracker on every frame, which must be computationally inexpensive in order to maintain a high frame rate. First, we merge proximate candidate anchors – ones too close to be separate LightAnchors (this often happens when a LightAnchor is detected at multiple pyramid levels). We then attempt to pair all current candidates with candidates from the previous frame using a greedy Euclidean distance matcher with a threshold to discard unlikely pairings. If a match is found, the current point is linked to the previous candidate anchor, forming a historical linked list. Our tracker also uses a time-to-live of five frames to compensate for momentary losses in tracking (e.g., image noise, momentary occlusion, loss of focus). Although basic, this approach is computationally inexpensive and works well in practice due to our high frame rate.

Decoding

After each frame is tracked, we attempt to decode all candidate anchors. As noted above, our tracker keeps a history of candidate anchors over time, which provides a sequence of intensity values. Rather than use only the center pixel value, we average over a small region, which we found to be less sensitive to camera noise and sub-pixel aliasing during motion. To convert the analog light intensity signal into a binary sequence, we use a dynamic threshold. We employ preambles that contain both 1's and 0's (i.e., high and low brightness), which allows us to find the midpoint of the min and max intensity values at both the beginning and end of a transmission. We linearly interpolate between these two midpoints (Figure 2) to produce a binary string. This

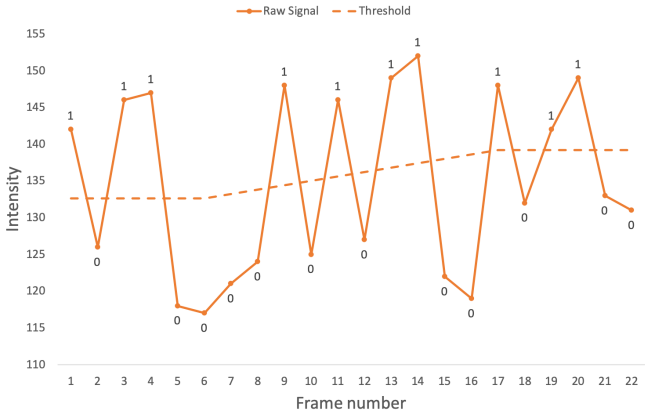

Figure 2. Light intensity over time for an example LightAnchor with 6-bit preamble (101100), 10-bit payload (0010101100) and 6-bit "postamble" (i.e., preamble of next transmission). Dashed line is interpolated binary threshold.

compensates for low-frequency changes in illumination (e.g., moving cloud cover, user motion, camera auto-exposure adjustment). We then test for the presence of our known preamble and postamble. If this is missing, the candidate is not decoded (i.e., we might be too early or late, or the tracked point is a static light and not a modulated light anchor). However, if the pre/postamble is correct, the data payload is saved to the anchor.

An interesting edge case that must be handled are reflections from LightAnchors (e.g., glints off specular objects, which also appear as point lights). Like true LightAnchors, these blink valid sequences and are decoded "correctly" by our pipeline. However, they almost always have a lower range of intensities (as they are secondary sources of light) and we use this fact to filter them. More specifically, if two candidate anchors are found to have valid, but identical sequences in the same frame, we only accept the one with higher signal variance.

Performance Analysis

We profiled our implementation using XCode on both an iPhone 7 and iPhone X. We tested different base resolutions (i.e., largest pyramid size), and for each, ran three trials of 500 frames each. Data was processed at 120 FPS, except for 320x180, which was processed at 240 FPS. The combined results are shown in Figure 3. Although reducing the image resolution greatly improves processing time, we found that scaling the image becomes a major bottleneck (often making up 50% of the processing). Even though we used the highly optimized iOS CoreGraphics API, we suspect that this bottleneck could be greatly mitigated with better GPU acceleration, which could allow LightAnchor detection to run at 240 FPS or more.

EVALUATION

To evaluate the robustness of our approach, we tested point lights of different size, across varying rooms, lighting conditions, and sensing distances. We also tested accuracy while the device was still and held by a user while walking. We now describe this procedure and results in detail.

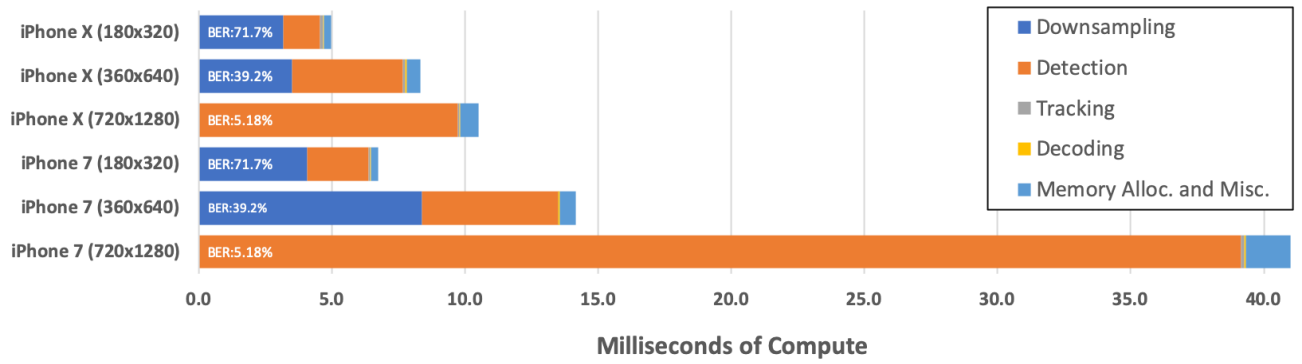

Figure 3. Compute time per frame on iPhone X and iPhone 7 at different input resolutions. Bit error rate (BER) results are computed from data collected in our main evaluation.

Apparatus & Procedure

We captured study data using an iPhone 7 (720p at 120 FPS) in three environments: workshop, classroom and office. In each of these settings, we varied the lighting condition: artificial light only (mean 321 lux), artificial light + diffuse sunlight (mean 428 lux) and lights off (mean 98 lux). We captured data using a tripod (i.e., smartphone held still) and also while walking slowly (~1 m/s, to induce some motion blur). We recorded approximately one second of video data at 2, 4, 6, 8, 10 and 12 meters. For our still condition, we used a surveyor's rope to mark distances, and for our walking condition, we used a 50 cm printed ArUco tag for continuous distance tracking (accepting frames within ±0.5m). Within each setting, we used two exemplar point lights: a standard 3mm LED and a larger 100x100mm LED matrix. These were placed 1.5m from the ground on a tripod and separated by 120cm. These two lights simultaneously emitted different (but known) 16-bit LightAnchors, driven by a single Arduino Mega. For all conditions, the LightAnchors pipeline ran with a base pyramid size of 1280x720, with 6-bit pre/postambles and 10-bit payloads.

LightAnchor Detection

Our detection rate did not change substantially across study conditions, and so we combine detection results for brevity. On average, our system found 50.8 candidate anchors (9.0 SD) in each frame. Of course, only two of these were actual LightAnchors, and our system detected these in all cases (i.e., a true positive rate of 100%).

After the pre/postamble filtering process, the true positive rate was still 100%, but our system found 3.1% false positives. The likelihood of any random pixel in the environment matching our pre/postamble is fairly low. Upon closer analysis of the video data, it appears most of these were actually small reflections of our actual LightAnchors, and thus transmitting correct patterns (an effect discussed in our Decoding section above). If we apply our variance filter, accepting only the best signal, it reduces false positives to 0.4% and true positives to 99.6%.

Bit Error Rate

Across all conditions and distances, we found a mean bit error rate (BER) of 5.2%, or roughly 1 error in every 20 transmitted bits. Note that this figure includes the 0.4% of false positives that made it through our various filtering stages. Overall, this level of corruption is tolerable and can be mitigated with standard error correction techniques, such as Hamming codes [10]. With respect to light size, the small LED had 6.5% BER, while the larger LED had 3.8% (Figure 4, left). Unsurprisingly, BER was higher while walking (mean 7.7%) than when the camera was still (2.7%), as seen in Figure 4 middle. Likewise, errors increased as ambient brightness increased (Figure 4, right). We also computed BER across the different base resolutions used in our performance analysis (Figure 3), and it is clear that high resolution (at least 720p) is needed to detect, track and decode LightAnchors accurately.

Recognition Latency

There are several effects that can cause incorrect rejection of LightAnchors, including poor tracking, motion blur, suboptimal camera-light synchronization, camera sensor noise, and variations in ambient lighting. As discussed above, it is rare

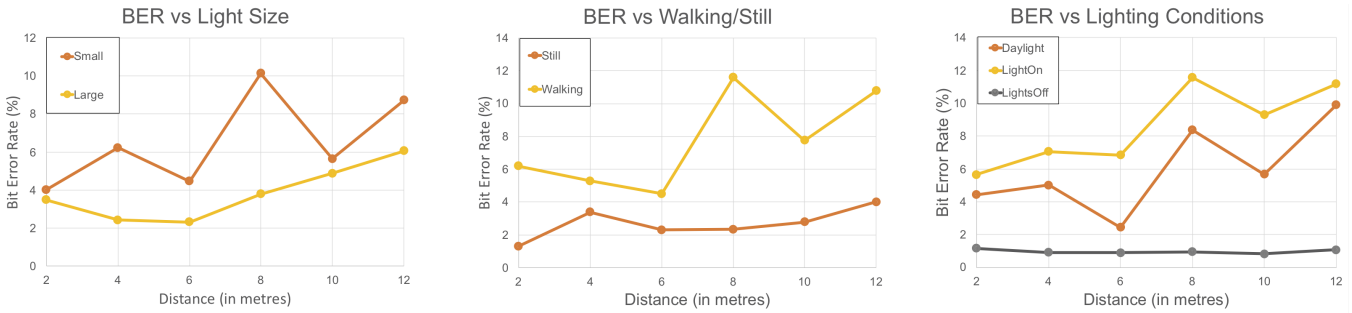

Figure 4. Bit error rate (BER) across different study conditions and distances.

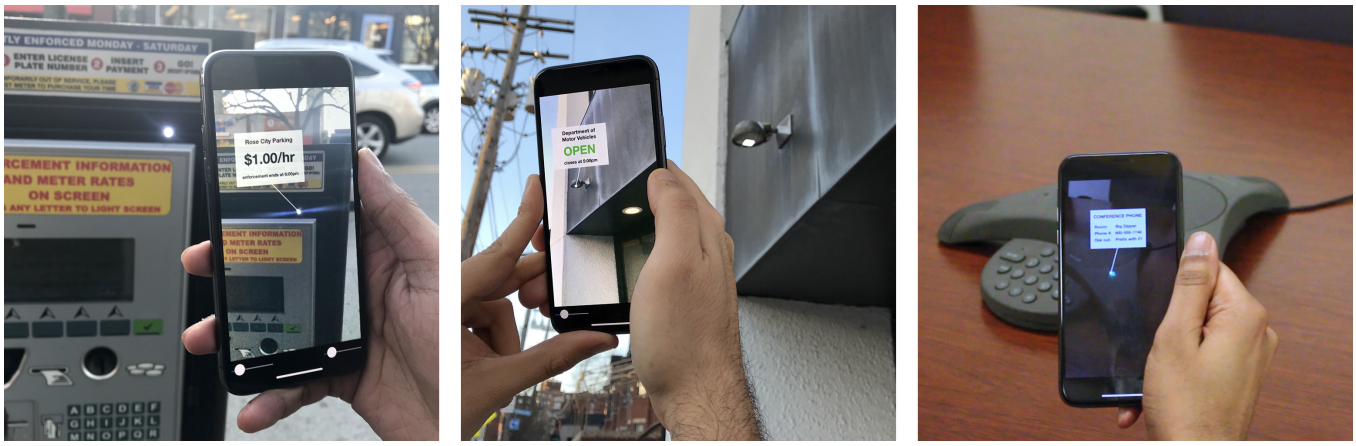

Figure 5. Example LightAnchor applications with fixed data payloads. Left: Parking meter displaying current rate. Center: Exterior light fixture denoting building operating hours. Right: Conference speaker phone displaying call-in number.

for our system to completely miss a visible LightAnchor, but it is common for a LightAnchors to have to transmit several times before being recognized. To quantify this, we used our collected data to compute the average time required to detect, track, and decode a LightAnchor. To do this, we started our detection pipeline at random offsets in our video data and recorded how long it took until LightAnchors were successfully decoded.

Across all conditions, we found a mean recognition time of 464 ms. As our test LightAnchors were 22 bits long (6-bit preamble, 10-bit payload, 6-bit postamble), they take a minimum of 183ms to transmit at 120 FPS. Because there is no synchronization, detection of a LightAnchor is almost certainly going to start somewhere in the middle of a transmission, meaning the system will have to wait on average 92 ms for the start of a sequence. The remaining 373 ms means that, on average, LightAnchors had to transmit twice before being recognized. We note that this latency varies across conditions, for example, mean recognition latency is 312 ms when the camera was held still (i.e., the first full transmission is often successful) vs. 615 ms when the user was walking (~3 transmissions before recognition).

PAYLOAD TYPES & EXAMPLE USES

The data payload of LightAnchors can be used in three distinct ways: fixed payloads, dynamic payloads, and connection payloads. To illustrate these different options, as well as to highlight the potential utility of LightAnchors, we created eleven demo applications. We note that these examples would require no *a priori* setup of devices and smartphones, and would allow anyone with the LightAnchors App on their phone to begin instantly interacting with objects in AR.

Fixed Payloads

The simplest use of LightAnchors is a fixed payload (similar to fiducial markers). Although this could contain plain text, the limited bitrate of LightAnchors makes this impractical. Instead, we envision transmission of an ID, which permits lookup through a cloud service, after which larger payloads (e.g., restaurant name, opening hours, menu, coupons, etc.) could be fetched over a faster connection (e.g., cellular, Wi-Fi). By utilizing geolocation in the lookup, it may be possible to use fairly small IDs (e.g., 16 bits).

As a demonstration of a fixed payload, we instrumented a street parking meter (Figure 5, left) with a light that transmits its enforcement zone ID (from which the rate schedule could

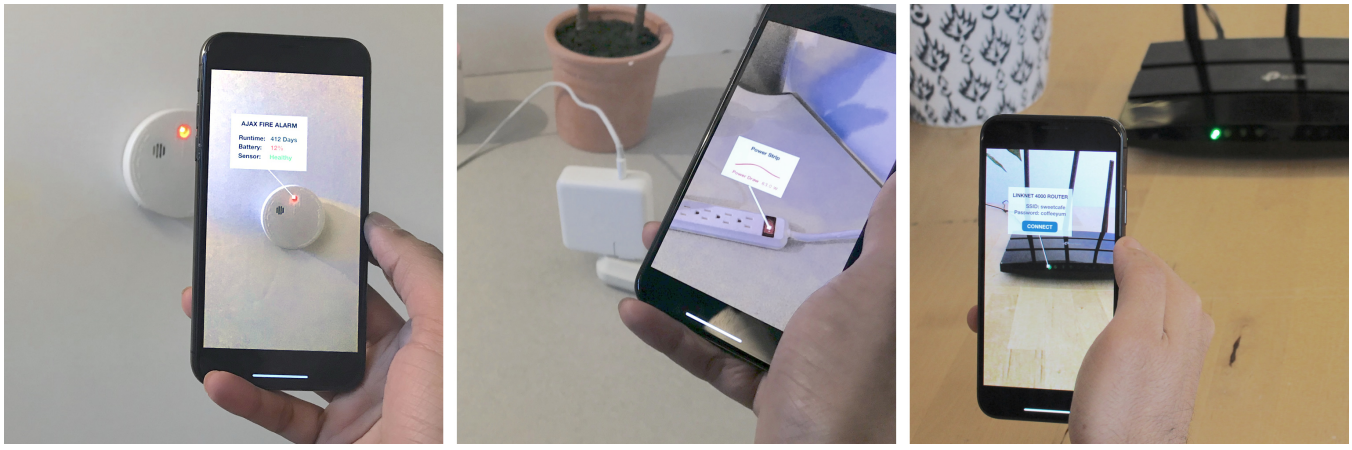

Figure 6. Example uses of LightAnchors with dynamic data payloads. Left: Smoke alarm that displays its real-time battery and alarm status. Center: Power strip that transmits its power usage. Right: WiFi Router displaying its SSID and guest password.

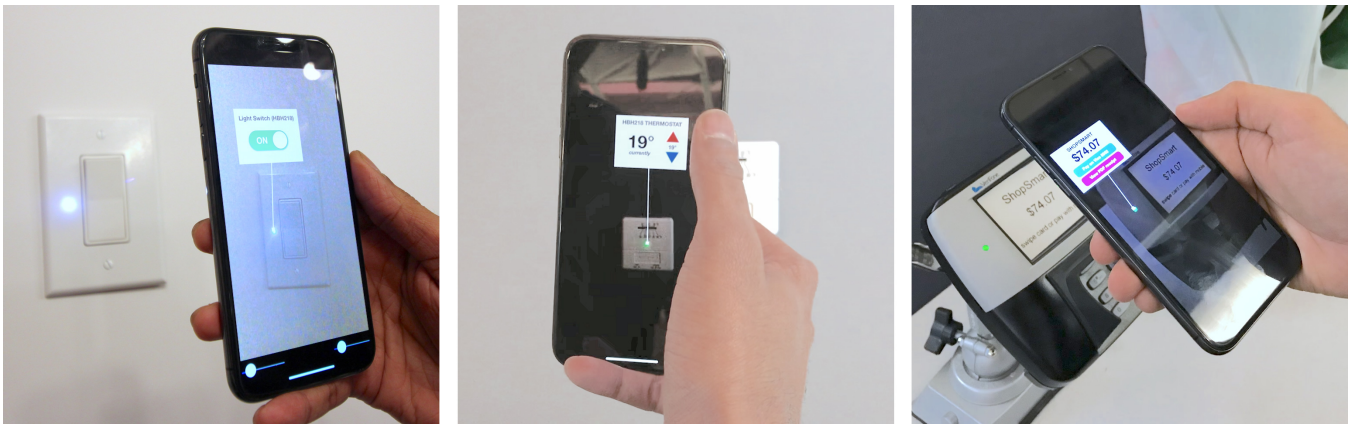

Figure 7. Example LightAnchors that use data payloads for connection information. Left: Light switch that offers remote control. Center: Thermostat that allows users to configure settings. Right: Terminal that permits payment via smartphone.

be retrieved). Similarly, we modified an outdoor entrance light to output a fixed UID (Figure 5, center); the summoned LightAnchor displays the building name (Department of Motor Vehicles), its current status (open), and closing time (5pm). Lastly, we modified a conference room phone demo that conveniently displays its call-in number (Figure 5, right).

Dynamic Payloads

More interesting are dynamic payloads, which contain a fixed UID that denotes the object, along with a dynamic value. For example, a glue gun (object identifier) and its live temperature (dynamic value), seen in Figure 1. A typical glue gun contains no digital components, and thus in practice, would require the addition of a microcontroller and thermistor, costing as little as $1 USD [22]. We also created two other demo devices that generally lack compute: a fire alarm that reports its operating status and an electrical strip that reports its power draw (Figure 6, left and center).

Of course, many devices already contain microprocessors that can control status lights and could be LightAnchor-enabled with a firmware update. For example, a networked security camera could be updated to use its recording light to share its privacy policy (Figure 1), and a router could share its SSID and a randomly generated guest password via its status lights (Figure 6, right).

Connection Payloads

Finally, LightAnchor payloads could be used to provide connection information. For example, a smart light switch could provide an IP address, allowing smartphones to open a socket and take control (Figure 7, left). To mitigate malicious behavior, a token with a short time-to-live could also be transmitted to ensure that users have at least line-of-sight. An internet connection could also allow devices to transmit a custom control interface, for example, a small HTML/CSS app. For instance, upon connecting to a smart thermostat, a simple temperature control widget could be downloaded and displayed in AR (Figure 7, center). As a final demo, we created a mock payment terminal (Figure 7, right) that could allow a smartphone to connect (securely) over the internet and enable payment.

LIMITATIONS

The biggest drawback of our method is limited bitrate, which is chiefly set by smartphone processors and camera FPS. This limits our practical payload size and makes our system prone to security issues found in schemes such as QR codes [16] . Fortunately, high-speed cameras are becoming increasingly commonplace in the market, and some Samsung devices can now capture video at 960 FPS. As camera FPS increases, LightAnchors can blink at higher rates, making the data imperceptible irrespective of the data payload and allow for much larger payloads. Smartphone processors also continue to improve, especially GPU performance, which should allow us to work with higher video resolutions, which would allow for LightAnchor detection at longer ranges.

There are also challenges in controlling the exposure and focus of the camera to enable robust tracking. We found the automatic camera settings were not ideal for LightAnchors (i.e., clipping in dark scenes), and so we had to lock settings such as exposure. However, as our user interface is a passthrough AR experience, settings that are ideal for LightAnchors are not always ideal for a human user.

Finally, at present, our LightAnchor widgets are flat with respect to the smartphone screen, as a single LightAnchor cannot provide 3D orientation. However, a known geometry of at least three non-planar LightAnchors (e.g., status lights on a microwave or Wi-Fi router) could allow for recovery of 6DOF position in the future. A similar effect might also be achieved using techniques such as structure from motion [20] and SLAM [17]. Either way, this would produce a more immersive AR effect.

CONCLUSION

We have presented our work on LightAnchors, a new approach for overlaying information and interfaces onto everyday objects in mobile AR. We take advantage of point lights (e.g., LEDs) that already exist in a wide range of products (or could be added for a few dollars). We described our implementation and results from an evaluation, which shows that our approach can be rapid and accurate. We plan to release our LightAnchors app on the Apple App Store, as it can run on recent iOS devices.

ACKNOWLEDGMENTS

This research was generously supported with funds from the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. We are also grateful to Anthony Rowe and his lab for early brainstorming on this effort.

REFERENCES

[1] Paramvir Bahl, and Venkata N. Padmanabhan. "RADAR: An in-building RF-based user location and tracking system." IEEE *Infocom*. Vol. 2. No. 2000. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2000. DOI: https://doi.org/10.1109/INFCOM.2000.832252

[2] Andrea Censi, Jonas Strubel, Christian Brandli, Tobi Delbruck, and Davide Scaramuzza. "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor." In 2013 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 891-898. IEEE, 2013. DOI: https://doi.org/10.1109/IROS.2013.6696456

[3] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. 2010. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking* (MobiCom '10). ACM, New York, NY, USA, 173-184. DOI: https://doi.org/10.1145/1859995.1860016

[4] Giorgio Corbellini, Kaan Aksit, Stefan Schmid, Stefan Mangold, and Thomas R. Gross. "Connecting networks of toys and smartphones with visible light communication." *IEEE Communications Magazine* 52, no. 7 (2014): 72-78. DOI: https://doi.org/10.1109/MCOM.2014.6852086

[5] Christos Danakis, Mostafa Afgani, Gordon Povey, Ian Underwood, and Harald Haas. "Using a CMOS camera sensor for visible light communication." In *2012 IEEE Globecom Workshops*, pp. 1244-1248. IEEE, 2012.

[6] Adrian A. de Freitas, Michael Nebeling, Xiang 'Anthony' Chen, Junrui Yang, Akshaye Shreenithi Kirupa Karthikeyan Ranithangam, and Anind K. Dey. 2016. Snap-To-It: A User-Inspired Platform for Opportunistic Device Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 5909-5920. DOI: https://doi.org/10.1145/2858036.2858177

[7] Mark Fiala. "ARTag, a fiducial marker system using digital techniques." In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR'05), vol. 2, pp. 590-596. IEEE, 2005.

[8] Anselm Grundhöfer, Manja Seeger, Ferry Hantsch, and Oliver Bimber. 2007. Dynamic Adaptation of Projected Imperceptible Codes. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (ISMAR '07). IEEE Computer Society, Washington, DC, USA, 1-10. DOI: https://doi.org/10.1109/ISMAR.2007.4538845

[9] Daniela Hall, Vincent Colin de Verdière, and James L. Crowley. "Object recognition using coloured receptive fields." In *European conference on computer vision*, pp. 164-177. Springer, Berlin, Heidelberg, 2000.

[10] Richard Hamming. "Error detecting and error correcting codes." *The Bell system technical journal*, 29, no. 2 (1950): 147-160.

[11] Seongkook Heo, Jaehyun Han, Sangwon Choi, Seunghwan Lee, Geehyuk Lee, Hyong-Euk Lee, SangHyun Kim, Won-Chul Bang, DoKyoon Kim, and ChangYeong Kim. 2011. IrCube tracker: an optical 6-DOF tracker based on LED directivity. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 577-586. DOI: https://doi.org/10.1145/2047196.2047272

[12] Impulse X2E Motion Capture – PhaseSpace Motion Capture. Retrieved April 4, 2019 from http://phasespace.com/x2e-motion-capture/

[13] Kensei Jo, Mohit Gupta, and Shree K. Nayar. 2016. DisCo: Display-Camera Communication Using Rolling Shutter Sensors. *ACM Trans. Graph.* 35, 5, Article 150 (July 2016), 13 pages. DOI: https://doi.org/10.1145/2896818

[14] Tai-Wei Kan, Chin-Hung Teng, and Wen-Shou Chou. 2009. Applying QR code in augmented reality applications. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry* (VRCAI '09). ACM, New York, NY, USA, 253-257. DOI: https://doi.org/10.1145/1670252.1670305

[15] Hirokazu Kato, and Mark Billinghurst. "Marker tracking and hmd calibration for a video-based augmented reality conferencing system." In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality* (IWAR'99), pp. 85-94. IEEE, 1999. DOI: https://doi.org/10.1109/IWAR.1999.803809

[16] Amin Kharraz, Engin Kirda, William Robertson, Davide Balzarotti, and Aurélien Francillon. "Optical delusions: A study of malicious QR codes in the wild." In 2014 *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 192-203. IEEE, 2014. DOI: https://doi.org/10.1109/DSN.2014.103

[17] Georg Klein, and David Murray. "Parallel tracking and mapping for small AR workspaces." In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1-10. IEEE Computer Society, 2007.

[18] Mani Kotaru, Manikanta, and Sachin Katti. "Position tracking for virtual reality using commodity wifi." *Proceedings of the IEEE Conference on Computer Vision*

and Pattern Recognition. 2017. DOI: https://doi.org/10.1109/CVPR.2017.286

[19] Katharina Krombholz, Peter Fruhwirt, Peter Kiesberg, Ioannis Kapsalis, Markus Huber, and Edgar Weippl. "QR code security: A survey of attacks and challenges for usable security." *International Conference on Human Aspects of Information Security, Privacy, and Trust*. Springer, Cham, 2014.

[20] Johannes L. Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104-4113. 2016.

[21] Nobuyuki Matsushita, Daisuke Hihara, Teruyuki Ushiro, Shinichi Yoshimura, Jun Rekimoto, and Yoshikazu Yamamoto. 2003. ID CAM: A Smart Camera for Scene Capturing and ID Recognition. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality* (ISMAR '03). IEEE Computer Society, Washington, DC, USA, 227-.

[22] Microchip Technologies Inc. ATtiny10. Retrieved April 4, 2019 from https://microchip.com/wwwproducts/en/ATtiny10

[23] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. 2009. Bokode: imperceptible visual tags for camera-based interaction from a distance. In *ACM SIGGRAPH 2009* papers (SIGGRAPH '09), Hugues Hoppe (Ed.). ACM, New York, NY, USA, Article 98, 8 pages. DOI: https://doi.org/10.1145/1576246.1531404

[24] PTC. Vuforia. Retrieved April 4, 2019 from https://www.ptc.com/en/products/augmented-reality

[25] Niranjini Rajagopal, Patrick Lazik, and Anthony Rowe. "Visual light landmarks for mobile devices." In *IPSN-14 IEEE proceedings of the 13th international symposium on information processing in sensor networks*. DOI: https://doi.org/10.1109/IPSN.2014.6846757

[26] Sridhar Rajagopal, Richard D. Roberts, and Sang-Kyu Lim. "IEEE 802.15. 7 visible light communication: modulation schemes and dimming support." *IEEE Communications Magazine* 50, no. 3 (2012): 72-82. DOI: https://doi.org/10.1109/MCOM.2012.6163585

[27] Short range active marker. Retrieved April 4, 2019. https://www.qualisys.com/hardware/accessories/active-markers/short-range-active-marker/

[28] Ramesh Raskar, Hideaki Nii, Bert deDecker, Yuki Hashimoto, Jay Summet, Dylan Moore, Yong Zhao, Jonathan Westhues, Paul Dietz, John Barnwell, Shree Nayar, Masahiko Inami, Philippe Bekaert, Michael Noland, Vlad Branzoi, and Erich Bruns. 2007. Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators. In *ACM SIGGRAPH* 2007. ACM, New York, NY, USA, Article 36. DOI: https://doi.org/10.1145/1275808.1276422

[29] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. "Speeded up detection of squared fiducial markers." *Image and vision Computing* 76 (2018): 38-47.

[30] Shigeru Saito, Atsushi Hiyama, Tomohiro Tanikawa, and Michitaka Hirose. "Indoor marker-based localization using coded seamless pattern for interior decoration." In 2007 *IEEE Virtual Reality Conference*, pp. 67-74. DOI: https://doi.org/10.1109/VR.2007.352465

[31] John Wang, and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection." In 2016 *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), pp. 4193-4198. IEEE, 2016. DOI: https://doi.org/10.1109/IROS.2016.7759617

[32] Christian Wienss, Igor Nikitin, Gernot Goebbels, Klaus Troche, Martin Göbel, Lialia Nikitina, and Stefan Müller. 2006. Sceptre: an infrared laser tracking system for virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology* (VRST '06). ACM, New York, NY, USA, 45-50. DOI=http://dx.doi.org/10.1145/1180495.1180506

[33] Karl D.D. Willis, Ivan Poupyrev, Scott E. Hudson, and Moshe Mahler. 2011. SideBySide: ad-hoc multi-user interaction with handheld projectors. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 431-440. DOI: https://doi.org/10.1145/2047196.2047254

[34] H.J. Woltring. New possibilities for human motion studies by real-time light spot position measurement. *Biotelemetry,* 1(3), 1974, 132-146.

[35] Robert Xiao, Chris Harrison, Karl D.D. Willis, Ivan Poupyrev, and Scott E. Hudson. 2013. Lumitrack: low cost, high precision, high speed tracking with projected m-sequences. *In Proceedings of the 26th annual ACM symposium on User interface software and technology* (UIST '13). ACM, New York, NY, USA, 3-12. DOI: https://doi.org/10.1145/2501988.2502022

[36] Robert Xiao, Scott Hudson, and Chris Harrison. 2016. CapCam: Enabling Rapid, Ad-Hoc, Position-Tracked Interactions Between Devices. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces* (ISS '16). ACM, New York, NY, USA. DOI: https://doi.org/10.1145/2992154.2992182