AirCode: Hidden Screen-Camera Communication on an Invisible and Inaudible Dual Channel

Kun Qian, *Tsinghua University and University of California San Diego;* Yumeng Lu, Zheng Yang, Kai Zhang, Kehong Huang, and Xinjun Cai, *Tsinghua University;* Chenshu Wu, *University of Maryland College Park;* Yunhao Liu, *Tsinghua University and Michigan State University*

https://www.usenix.org/conference/nsdi21/presentation/qian

This paper is included in the Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation.
April 12–14, 2021

978-1-939133-21-2

AIRCODE: Hidden Screen-Camera Communication on an Invisible and Inaudible Dual Channel

Kun Qian[*#], Yumeng Lu[*], Zheng Yang[*1], Kai Zhang[*]
Kehong Huang[*], Xinjun Cai[*], Chenshu Wu[†], Yunhao Liu[*‡]
[*]*Tsinghua University,* [#]*University of California San Diego*
[†]*University of Maryland College Park,* [‡]*Michigan State University*
{qiank10, lym.mongo, hmilyyz, nezharen1995, huangkh13, caixj08, wucs32, yunhaoliu}@gmail.com

Abstract

Hidden screen-camera communication emerges as a key enabler for the next generation videos that allow side information, such as TV commercials, augmented contents, and even the video itself, to be delivered to machines during normal watching. To guarantee imperceptibility to human eyes, existing solutions have to sacrifice data rate and reliability enormously. This paper presents AIRCODE, a hidden screen-camera communication system built upon invisible visual and inaudible audio dual channel. While ensuring great unobtrusiveness, AIRCODE achieves robust communication at a remarkably high rate of $>$1Mbps, for the first time, enabling imperceptible transmission of not only texts but also videos. AIRCODE makes two key technical contributions. First, AIRCODE takes the complementary advantages of video and audio channels by exploiting the reliable yet low-rate inaudible audio link as the control channel while the unreliable but high-rate visual link as the data channel. Second, AIRCODE incorporates visual odometry to accurately identify and track the captured screen, regardless of dynamic video contents and surrounding interference. Experiments on commercial monitors and smartphones demonstrate that AIRCODE significantly outperforms the state-of-the-art system, yielding a remarkable data rate of 1069 Kbps while with BER of 5%.

1 Introduction

Over the past few decades, video has risen into popularity across the globe. Billions of video are produced, captured, shared, and viewed every day and everywhere. Rather than merely watching them, audience often desires to acquire extra information related to the video content, especially with their carry-on devices, e.g., smart phone and smart glasses. For example, an advertisement can deliver a second video introducing detailed usage and function of the advertised product to its potential users. Instead of letting the viewers record a blurry video with unintended surrounding contents and color distortion, a video can provide a low quality version of itself for direct sharing. An AR video can compute and send the augmented contents to a viewer's device for direct rendering without draining its limited computing and power resources. To make these applications a reality, convenient and friendly communication approach, better with high data rates and reliability, is needed.

One intuitive solution is to integrate screens with Wi-Fi or Bluetooth, like nowadays smart TV, and convey side information through the wireless channel. However, such combination has several drawbacks. First, wireless connection requires explicit setup. A viewer wearing smart glasses may see multiple screens during his daily life and be bothered to search their Wi-Fi from many available networks. Second, wireless device and screen are not well synchronized. For example, a screen playing an AR video should deliver the augmented contents synchronized at the frame level, e.g., 16.7 ms with a frame rate of 60 Hz, which cannot be always achieved by Wi-Fi due to occasionally large latency and jitter [29]. Last but not least, the broadcasting behavior of wireless devices causes information spamming or leakage to unwanted people. For example, in TV commercial, only the viewers of the advertisement need the side information about the product. In video sharing, an eavesdropper must not access the low quality video due to copyright protection.

Alternatively, as cameras are indispensable to smart devices, it is more convenient, impromptu and secure to embed extra information into video and deliver them through the screen-camera channel without impacting viewers' watching experience. Thanks to the high refresh rates ($\geq$ 120 Hz) of modern screens, pioneer works [24, 36, 41] hide data in high frame rate videos to cheat human eyes that have much lower perception rates (40-50 Hz) [31]. However, they emphasize on human perception but sacrifice data rates (e.g., 551 Kbps with a bit error rate (BER) of 8% [41]), hardly reliable for real data communication of potential applications. Improving reliable data rates on hidden screen-camera channel, however, is non-trivial. Unlike conventional communication channels [8, 17, 22, 27], enhancing the signal does not help for the screen-camera channel because it immediately sets up

[1] Zheng Yang is the corresponding author.

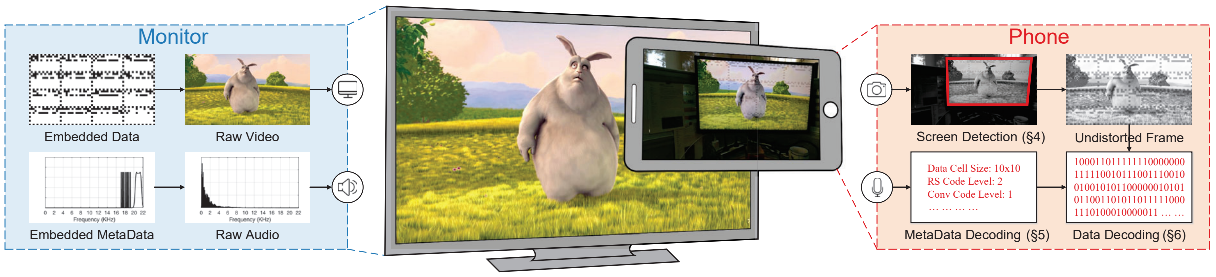


Figure 1: System overview of AIRCODE. The monitor displays a video with embedded data in raw video frames, and metadata (control information) in raw audio signals. The phone records the video, then detects screen in camera images to equalize distorted video frames, decodes metadata from audio signals, and finally decodes data embedded in video.

a conflict with unobtrusiveness to human eyes. We refer the reader to Section 2 for the more detailed analysis of the unique characteristics of the screen-camera channel. Essentially, we are facing the dilemma of foregoing three contradicting goals.

In this study, we first investigate the state-of-the-art works from distinct perspectives and discuss the root causes of unreliability. On this basis, we present AIRCODE, an imperceptible screen-camera communication system that supports high-throughput and reliable data transmission on top of regular video viewing, as demonstrated in Figure 1. The working scenario of AIRCODE is non-sophisticated: When a watcher intends to acquire information on the side channel, she simply shoots a video of the screen playing the encoded video using her smartphone, which automatically receives and decodes the embedded bits, at a rate of as high as 1069 Kbps, almost 2× higher than the state-of-the-art ChromaCode [41], underpinning various applications like video-in-video sharing. AIRCODE boosts data throughput while reducing BER by an invisible visual and inaudible audio dual-channel in three distinct ways:

(1) Precise screen tracking: Precise screen detection and tracking is critical to equalization of video frames, the key process for successful decoding of visual codes. According to our measurements, errors of a few pixels may significantly affect packet reception (§2). Precise screen detection, however, is non-trivial due to various factors, such as video contents, surrounding background, hand motions, etc. In AIRCODE, we exploit the idea of visual odometry [21] for this purpose. Specifically, AIRCODE constructs a 3-D map of a screen of interest and tracks the screen by estimating the phone pose with projections of map points and then projecting the screen with the pose estimated.

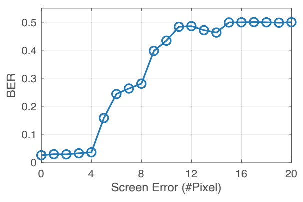
(2) Reliable audio channel: While the video channel shares the same frequency band with the ambient illuminance [41], the near-ultrasound audio channel is resistant to ambient low-frequency noises [38] and more reliable. Thus, AIRCODE allocates the inaudible speaker-microphone link as the control channel for the critical metadata of visual codes (e.g., code layout, coding scheme, etc.). To seamlessly comply with the video channel, the audio channel should fulfill two conditions, i.e., short packet duration that matches the high frame rate of the video channel, and low packet error rate (PER) that is required by the overall communication. To achieve this, AIRCODE carefully designs acoustic packets to overcome problems of reverberation and frequency spectral leakage caused by short packet duration and achieves high reliability with nearly zero PER.

(3) High-rate visual channel: AIRCODE leverages the screen-camera link as the high-rate data channel. Data bits are transmitted via imagery codes imperceptibly embedded in the primary carrier video. To minimize flickers, we adaptively choose the required lightness changes according to the spatial texture of the primary video content and perform lightness alteration as introduced in [36, 41]. The high data rate is then achieved by 1) embedding full-frame visual codes with a carefully designed frame structure; 2) effective error correction with an adaptive concatenated coding scheme.

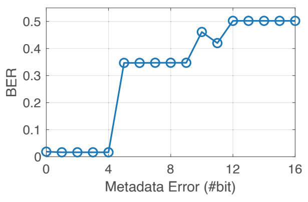
We implement AIRCODE using commodity computer monitors and smartphones. We conduct real-world experiments and extensive evaluations on key metrics including BER, throughput, goodput and screen-tracking accuracy. Besides, various system parameters, such as distance, angle, signal strength, background interference, and frame size, are tested. Videos with various texture, luminance, quality and audios with different types of sound are used for evaluation. The imperceptibility is tested with a user study. Experimental results demonstrate that AIRCODE achieves a remarkable data rate of 1069 Kbps with an average BER of 5%, which significantly outperforms existing approaches.

The core contributions of this paper are:

- We present AIRCODE, a hidden screen-camera communication system that achieves >1Mbps throughput for the first time, allowing not only text but also image and video transmission.
- To the best of our knowledge, we are the first to exploit an invisible visual and inaudible audio channel, which jointly enables fully imperceptible, high-rate, and reliable communication by their complementary advantages.
- We propose an algorithm using visual odometry that precisely tracks screen locations even with dynamic video content, complex ambient contexts, and camera motions caused by unconsciously hand motion and shake.



(a)



(b)

Figure 2: Key factors for screen-camera link quality. (a) Screen localization error, and (b) Metadata error.

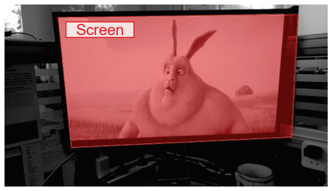
2 Motivation

Hidden screen-camera communication is possible with the natural difference between the human vision system and camera system: Human eyes cannot resolve flickers, or luminance fluctuations, at frequencies higher than 100Hz, but instead only perceive the average lightness, due to the low-pass flicker-fusion property of human vision system [31]. Thus, if we inversely modify the lightness of a pair of subsequent video frames, which are termed as complementary frames, and play them at high frame rate, e.g., 120 frames per second (fps) that is supported by modern commercial monitors and TVs, human eyes will not perceive the change of lightness. Instead, they will only observe the original video content, which is the average of complementary frames. In contrast, commodity cameras with a high capturing rate can acquire the differences and decode the data if certain information is modulated on top of the change of lightness. Pioneer work has exploited this phenomenon for hidden screen-camera communication by embedding data bits unobtrusively in primary carrier videos. The visual channel of AIRCODE is also built upon this idea. The hidden screen-camera channel is reported error-prone due to some unique intrinsic errors [24, 36, 41], such as projection distortions, blurring, Moiré patterns, rolling shutter effects, and frequent changes of camera pose caused by hand motion and shake, etc.

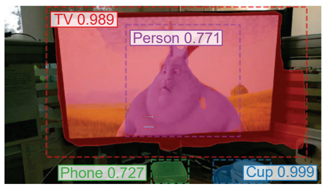
2.1 Challenges and Measurements

Previous proposals mainly aim at minimizing flickers for good unobtrusiveness, yet sacrifice data rates and robustness. The best result to date reported is from ChromaCode, with the data rate of up to 551 Kbps and BER of 8%, which is far from sufficiency for many applications such as the above-mentioned video-in-video sharing. What's even worse, such performance is obtained with limitations including visible border markers [36] and stable cameras [24, 41].

To reveal the root causes of these limitations, we consider the state-of-the-art works, ChromaCode, and evaluates how its reliability suffers when (1) the detected screen deviates from the ground-truth, and (2) the received metadata (e.g. visual code layout, coding scheme, etc.) has bit errors. Experiments are conducted in a cubicle space and with different types of videos, as listed in Table 1. During experiments, the phone is placed static with several fixed poses, so that the ground-truth



(a)



(b)

Figure 3: Examples of screen detection failure. (a) rule-based approach [41], and (b) learning-based approach [10].

of the screen can be manually marked. Besides, the metadata is known prior as well. To evaluate the impact of screen detection, we gradually shift the input of the screen location by a small number of pixels and calculate BER accordingly. To evaluate the impact of metadata decoding, we manually add random error bits in the channel code of metadata.

(1) Impact of screen detection. Figure 2a shows that the BER of the screen-camera link remains around 2% when the location error of the screen is less than 4 pixels (in a 720p video), which is about the half-length of the smallest data cell in ChromaCode, but explodes when the location error exceeds 4 pixels. It fails when the location error reaches 15 pixels. It means that highly accurate screen detection is necessary for the screen-camera link. Recent phones incorporates image stabilization [20, 32] to reduce blurring caused by camera motion. However, these techniques can only remove small handshakes and thus require the user to uncomfortably hold the phone at a fixed position during the recording. Figure 3 further illustrates failure cases of screen detection with two representative approaches, the rule-based approach used in ChromaCode, and Mask R-CNN [10], a learning-based approach. The rule-based approach tries to find a quadrilateral whose outer bound consists of edges in the image and has consistent similar lightness, which may be misled by video contents and the surrounding environment. While Mask R-CNN can accurately recognize objects in images after sufficient training, it fails to meet the stringent requirement on accurate segmentation with errors of only several pixels for screen-camera communication. Besides, both approaches lack pertinent mechanisms to deal with continuous tracking of the screen in a video.

(2) Impact of metadata decoding. Figure 2b shows that the screen-camera link becomes lost when the number of error bits in metadata exceeds 4, which is consistent with that ChromaCode encodes 5-bit metadata with 15-bit code and corrects at most 3-bit errors. Through all experiments, 24.6% frames have BERs over 8%, among which 29.3% are due to failure in decoding metadata. The drawback of existing sole screen-camera communication schemes is that while metadata is more important than data and requires robust communication with lower BER, it is conveyed in the same erroneous visual channel as data. To achieve reliable communication with the screen-camera link, a more robust channel is needed for transmission of metadata.

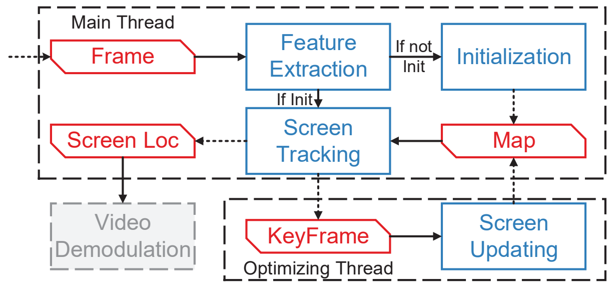


Figure 4: Logic flow of screen detection.

3 System Overview

Motivated by the measurements above, we design AIRCODE to address the two significant drawbacks, namely erroneous screen tracking and vulnerable transmission of metadata and deliver an imperceptible, high-rate, and reliable screen-camera communication system.

Figure 1 shows the system overview of AIRCODE. At the sender side, AIRCODE embeds data (visual codes) into raw video frames and embeds metadata of visual codes into raw audio signals. During playing the video, users run AIRCODE on their phones and shoot the monitor. At the receiver side, upon receiving camera images and acoustic signals, AIRCODE simultaneously tracks the screen in images with visual odometry to equalize distorted frames (§ 4) and decodes metadata embedded in acoustic signals (§ 5). Afterward, both undistorted frames and metadata are used to decode data conveyed in video frames (§ 6).

4 Screen Detection

Detecting screen is the key process for successful decoding of visual code, due to unknown perspective distortion of the screen in images. However, various video content and background environments may interfere in screen detection, leading to failure of decoding, as discussed in § 2. Existing works embed specific position codes at the edges and corners of video for screen detection, which wastes communication resources and affects the watching experience of audiences. In contrast to existing solutions that try hard to reduce the interference caused by complex background environment, AIRCODE exploits it as visual clues for odometry [21] to track the screen in frames.

Visual odometry alternately tracks the poses of image frames and maintains a global map of 3-D points seen by multiple image frames. Generally, visual odometry consists of three main steps. (1) Initialization. Visual odometry matches feature pixels of two image frames and estimates their relaive pose. The global map is then initialized with the 3-D map points of the matched feature pixels. (2) Tracking. Upon receiving a new frame, visual odometry matches its feature pixels with 3-D points in the global map, estimates the pose of the frame, and updates 3-D map points of the matched feature pixels in the global map. (3) Optimization. Visual odometry periodically buffers some frames, termed as keyframes, to refine the global map and avoid drifting error. When a new keyframe arrives, visual odometry jointly optimizes poses of all buffered keyframes that share common feature pixels with the new keyframe and all 3-D points seen by these keyframes in the global map.

To enable screen detection, AIRCODE modifies the main steps of visual odometry to further keep 4 screen points in the global map. Each screen point corresponds to one corner point of the rectangular screen. Thus, the screen can be uniquely identified and represented by its four corner points. Figure 4 shows the logic flow of screen detection of AIRCODE. In addition to visual odometry, during initialization, AIRCODE estimates the initial 3-D locations of the screen points. During the tracking process, AIRCODE projects the screen points in the frame with the frame pose for decoding. In the optimization process, AIRCODE updates the screen points with the refined poses of keyframes and the projections of the screen points on these keyframes to avoid drifting error.

4.1 Feature Extraction

Feature pixels are representative pixels invariant to translation, scaling and rotation of an image and can be used for robustly tracking successive image frames and mapping 3-D points corresponding to these feature pixels. AIRCODE uses ORB [28] which is fast to compute and match, and remains invariant across different viewpoints, yielding sufficient efficiency and accuracy for visual odometry.

In practice, the video content may change with time, and feature pixels within the screen are not consistent across successive frames. To avoid the negative impact of mismatching these feature pixels, AIRCODE first obtains a coarse estimation of the screen frame, and filter out all feature pixels within it. The initial screen frame is calculated by the rule-based algorithm during initialization (as in § 2) or inherited from the last frame during the tracking process. Figure 5a shows examples of feature extraction, where the screen frame is highlighted and feature pixels within it are removed.

4.2 Initialization

The initialization process computes the relative pose between two frames and triangulates an initial set of 3-D map points for tracking and 4 screen corner points for communication. Lacking the knowledge of screen points, AIRCODE uses the rule-based algorithm to detect the screen and decode frames during initialization. As a successful decoding indicates accurate screen detection, AIRCODE selects two decoded frames with their detected projections of the screen for initialization.

Figure 5a illustrates an example of initialization with two matched frames. Denote the two selected frames as F_0 and F_1, any matched feature pixels as $\mathbf{p}_0$ and $\mathbf{p}_1$, and any matched projection pixels of screen corner points as $\mathbf{s}_0$ and $\mathbf{s}_1$, AIRCODE uses epipolar geometry and computes the fundamental matrix $\mathbf{F}_{10}$ that connects any pair matched pixels [9]:

$$\mathbf{p}_0^T \mathbf{F}_{10} \mathbf{p}_1 = 0, \quad \mathbf{s}_0^T \mathbf{F}_{10} \mathbf{s}_1 = 0. \tag{1}$$

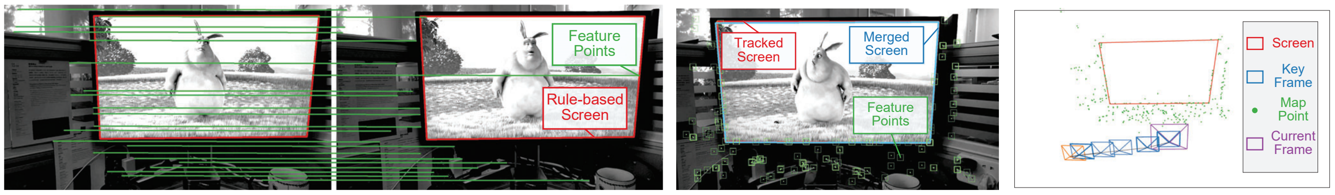


Figure 5: Key steps of screen detection: (a) Out-of-screen feature extraction and matching for initialization, (b) frame tracking with matched feature pixels and screen tracking with frame pose, and (c) global map and key frames for screen updating.

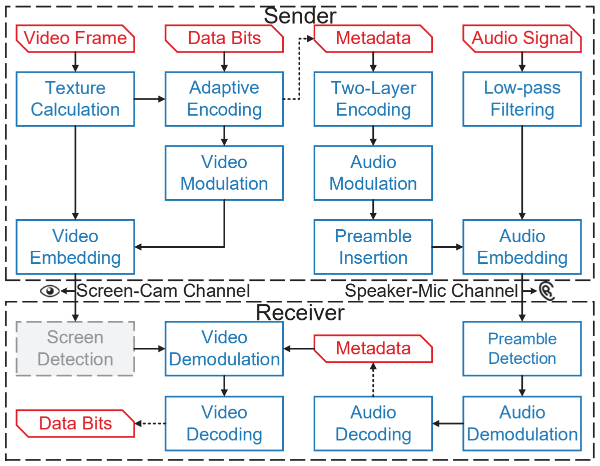


Figure 6: Logic flow of dual-channel communication.

F_{10} can be solved by integrating eight pairs of matched pixels [18]. Since the successful decoding strongly indicates that screen points are accurately located in the two frames, AIRCODE selects the projection pixels of four screen points (i.e., $\mathbf{s}$) as four pairs of the matched pixels, and four pairs of matched feature pixels (i.e., $\mathbf{p}$) as the rest.

To recover the camera pose, the fundamental matrix $\mathbf{F}_{10}$ is converted to the essential matrix $\mathbf{E}_{10}$ using the intrinsic matrix $\mathbf{K}$ of the camera:

$$\mathbf{E}_{10} \equiv \mathbf{t}_{10}^{\wedge}\mathbf{R}_{10} = \mathbf{K}^T \mathbf{F}_{10}\mathbf{K}, \tag{2}$$

where $\mathbf{t}_{10}$ and $\mathbf{R}_{10}$ the are relative translation vector and rotation matrix respectively, and the operator $(\cdot)^{\wedge}$ is to calculate the skew-symmetric matrix from a vector. The four possible poses (i.e., $\mathbf{t}_{10}$ and $\mathbf{R}_{10}$) are derived from the essential matrix $\mathbf{E}_{10}$ via singular value decomposition [9]. With each ambiguous candidate, AIRCODE triangulates 3-D points corresponding to all matched feature pixels and projection pixels of four screen points. A valid 3-D point with high confidence should be in front of the camera and have a significant parallax between the two frames. Thus, AIRCODE selects the candidate with most such points as the true pose. Finally, AIRCODE initializes the 3-D global map and the screen with all valid 3-D points of the true pose. The two frames used in initialization are set as initial keyframes.

4.3 Screen Tracking

After initialization, AIRCODE continuously tracks poses of new frames and projects the screen accordingly for decoding. As visual odometry, AIRCODE optimizes the pose of the current frame by minimizing the location error between the projections of map points and their matched feature pixels in the frame. Specifically, suppose that the pose matrix of the i-th frame is $\mathbf{T}_i = [\mathbf{R}_i \mid \mathbf{t}_i]$, where $\mathbf{R}_i$ and $\mathbf{t}_i$ are the corresponding rotation matrix and translation vector, and N matches $\langle \mathbf{P}_j, \mathbf{p}_{i,j} \rangle$ are detected, where $\mathbf{P}_j$ is the j-th map point and $\mathbf{p}_{i,j}$ is the matched feature pixel in the i-th frame, the pose $\mathbf{T}_i$ can be optimized via bundle adjustment [35]:

$$\mathbf{T}_{i,opt} = \arg\min_{\mathbf{T}_i} \sum_{j=1}^{N} \|\mathbf{p}_{i,j} - \pi(\mathbf{T}_i, \mathbf{P}_j)\|^2, \tag{3}$$

where $\pi(\cdot)$ projects the 3-D map points onto the image frame given its pose [9]:

$$\pi(\mathbf{T}_i, \mathbf{P}_j) = \left[\frac{(\mathbf{K}(\mathbf{R}_i\mathbf{P}_j + \mathbf{t}_i))_0}{(\mathbf{K}(\mathbf{R}_i\mathbf{P}_j + \mathbf{t}_i))_2}, \frac{(\mathbf{K}(\mathbf{R}_i\mathbf{P}_j + \mathbf{t}_i))_1}{(\mathbf{K}(\mathbf{R}_i\mathbf{P}_j + \mathbf{t}_i))_2}\right]^T. \tag{4}$$

With the estimation of the pose, AIRCODE projects screen points onto the current frame:

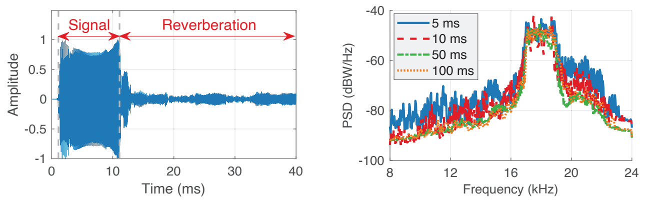
$$\mathbf{s}_{i,j} = \pi(\mathbf{T_i}, \mathbf{S}_j), \tag{5}$$

where $\mathbf{S}_j$ is the j-th screen point ($j = 1,2,3,4$). To further minimize projection error, AIRCODE searches outstanding Shi-Tomasi corners [34] within neighborhoods of these projections as the final estimation.

AIRCODE may fail to decode due to small but intolerant deviations of screen tracking. In contrast, despite frequent failures, the rule-based algorithm can accurately detect edges and corners in the frame and yield more accurate estimation if the screen is successfully detected. Thus, when decoding fails, AIRCODE further executes the rule-based algorithm to obtain a second estimation of the screen, denoted as $\mathbf{s}'_{i,j}$. Then, it merges the corner points of the two screens as:

$$\mathbf{s}''_{i,j} = \begin{cases} \mathbf{s}'_{i,j} & \|\mathbf{s}_{i,j} - \mathbf{s}'_{i,j}\| \geq \delta_h \\ \mathbf{s}_{i,j} & \|\mathbf{s}_{i,j} - \mathbf{s}'_{i,j}\| \leq \delta_l \end{cases}, \tag{6}$$

As screen tracking yields consistently small errors, an upper threshold, δ_h, is used to reject totally wrong estimation from the rule-based algorithm. Meanwhile, a lower threshold, δ_l, is used to accept more accurate estimation from the rule-based algorithm. When $\delta_l < \|\mathbf{s}_{i,j} - \mathbf{s}'_{i,j}\| < \delta_h$, AIRCODE has no extra information to determine which estimation is better. Thus, it forks two screen candidates, whose j-th points are assigned as $\mathbf{s}_{i,j}$ and $\mathbf{s}'_{i,j}$ respectively. Finally, AIRCODE tries all screen candidates until the frame is successfully decoded. Note that in the worst case where the screen tracking fails, e.g., due to lack of visual features around the screen, AIRCODE



(a) Reverberation (b) Spectral Leakage

Figure 7: Main problems of short packet duration. (a) enlarged impact of reverberation, and (b) aggravated spectral leakage.

degenerates to the rule-based algorithm and still maintains a moderate decoding rate. Figure 5b illustrates the process of screen tracking, where all matched feature pixels in the frame and both the tracked screen and the merged screen are highlighted.

To avoid drifting error during tracking, AIRCODE periodically selects frames and passes them to the optimizing thread. Specifically, it creates new keyframes only when one of the following conditions are satisfied:

1. More than 0.5 fps frames have passed after the creation of the last keyframe, and the current frame tracks less than 90% points than the last keyframe.
2. The current frame fails to be decoded with the tracking screen but can be decoded with the merged screen.

Condition (1) avoids computing cost with redundant frames, while condition (2) ensures timely updating of the screen.

4.4 Screen Updating

The optimizing thread periodically refines map points and screen points to avoid drifting error. Figure 5c shows the structure of global map and keyframes. Upon receiving a new keyframe, AIRCODE first deletes obsolete map points that are seen by fewer than 3 keyframes and add new points from matches between the current keyframe and previous keyframes, as that in visual odometry. Next, AIRCODE collects all map points seen by the current keyframe and all keyframes that see any of these map points for optimization. Specifically, suppose M key frames together with the current keyframe are selected, the pose of the i-th keyframe is $\mathbf{T}_i$, and N_i matches $\langle \mathbf{P}_{i_j}, \mathbf{p}_{i,j} \rangle$ are detected in the i-th keyframe. The pose $\mathbf{T}_i$ and map points $\mathbf{P}_k$ are optimized via bundle adjustment:

$$\{\mathbf{T}_{i,opt}\}_{i=0}^{M}, \{\mathbf{P}_{\mathbf{k},\mathbf{opt}}\}_{k=1}^{K} = \mathop{\arg\min}_{\{\mathbf{T}_i\}_{i=0}^{M}, \{\mathbf{P}_\mathbf{k}\}_{k=1}^{K}} \sum_{i=1}^{M} \sum_{j=1}^{N_i} \|\mathbf{p}_{i,j} - \pi(\mathbf{T}_i, \mathbf{P}_{i_j})\|^2. \quad (7)$$

Then, AIRCODE optimizes screen points via multi-view triangulation with the refined poses of keyframes. Specifically, denote the projection of the j-th screen point $\mathbf{S}_j$ on the i-th key frame as $\mathbf{s}_{i,j}$, the 3-D location of the j-th screen point is calculated by minimizing the following projection error:

$$\mathbf{S}_{j,opt} = \mathop{\arg\min}_{\mathbf{S}_j} \sum_{i=1}^{M} \|\mathbf{T}_i \bar{\mathbf{S}}_j - (\hat{\mathbf{s}}_{i,j}^T \mathbf{T}_i \bar{\mathbf{S}}_j) \hat{\mathbf{s}}_{i,j}\|^2, \quad (8)$$

where $\hat{\mathbf{s}}_{i,j} = \frac{\mathbf{K}^{-1}\bar{\mathbf{s}}_{i,j}}{\|\mathbf{K}^{-1}\bar{\mathbf{s}}_{i,j}\|}$ is the direction vector of the projection $\mathbf{s}_{i,j}$, and $\bar{\mathbf{S}}_j$ and $\bar{\mathbf{s}}_{i,j}$ are the homogeneous representation of $\mathbf{S}_j$ and $\mathbf{s}_{i,j}$, respectively. The term $(\hat{\mathbf{s}}_{i,j}^T \mathbf{T}_i \bar{\mathbf{S}}_j)\hat{\mathbf{s}}_{i,j}$ calculates the projection coordinate of $\mathbf{S}_j$ along the direction $\hat{\mathbf{s}}_{i,j}$. Intuitively, when $\mathbf{S}_j$ is accurately localized, $\mathbf{T}_i \bar{\mathbf{S}}_j$ has the same direction as $\hat{\mathbf{s}}_{i,j}$ and the error term becomes 0. The optimal solution $\mathbf{S}_{j,opt}$ is the eigenvector corresponding to the minimal eigenvalue of the matrix $\sum_{i=1}^{M} (\mathbf{T}_i - \hat{\mathbf{s}}_{i,j}\hat{\mathbf{s}}_{i,j}^T \mathbf{T}_i)^T (\mathbf{T}_i - \hat{\mathbf{s}}_{i,j}\hat{\mathbf{s}}_{i,j}^T \mathbf{T}_i)$, and can be calculated directly in closed form.

Finally, after optimization, AIRCODE discards redundant keyframes whose map points have been seen in at least other three keyframes, as in [33], to maintain a reasonable number of keyframes for fast optimization, and avoid large estimation uncertainty caused by co-located key frames [19].

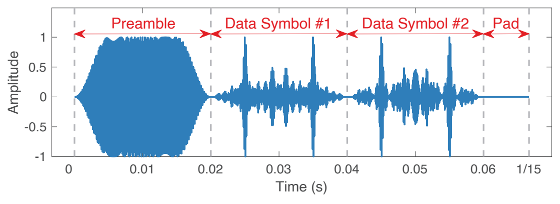
5 Audio Control Channel

Despite orders of magnitude smaller throughput, the speaker-microphone link is more reliable than the screen-camera link due to little interference in the near ultrasound band. Thus, AIRCODE exploits the speaker-microphone link to send critical metadata of visual codes, e.g., coding layout, coding scheme, etc. By doing so, not only is metadata robustly delivered, but also more video coding area is saved to convey more data. Figure 6 shows the logic flow of the audio communication part. At the sender side, metadata is encoded with a two-layer encoding scheme, which includes Golay coding and Manchester coding, and then modulated on frequency subcarriers. A chirp signal is prepended as a preamble for timing alignment. The raw audio signal is low-pass filtered and then embedded with the control signal. At the receiver side, the signal is first high-pass filtered to remove raw audio signal and background noises, then aligned with a chirp template, and finally demodulated and decoded to yield metadata.

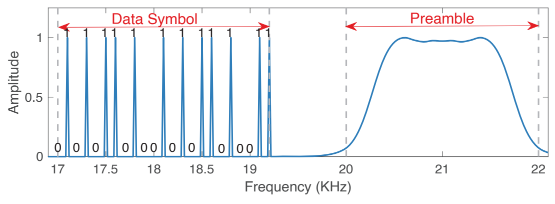
5.1 Design Challenges

Frame-level Packet Duration. To convey metadata for the video channel, whose data frame (a pair of complementary video frames) rate is 60 fps, the audio packets should be sent within tens of milliseconds. However, most existing solutions send packets at the second level, which is sufficiently reliable but cannot be agilely adjusted according to the video channel. For example, Dolphin [38] adopts long OFDM packets, which is 3.56 s long. Chirp packets, whose duration is 1.463 s, are used in [12]. To match the need for the high packet rate, a shorter packet should be designed.

However, reducing packet duration will cause two problems, reverberation in the time domain and spectral leakage in the frequency domain. First, the impact of reverberation is enlarged. Due to the multi-path effect, the microphone not only hears the direct signal but also delayed reverberating signals reflected by surrounding environments. Figure 7a shows an example of a 10 ms chirp signal received and its reverberations.



(a)



(b)

Figure 8: Audio packet design. (a) The 66.7 ms packet consists of one 20 ms preamble, two 20 ms data symbols and one silence pad. (b) The preamble and data symbol are separately located within 2 ranges of 20 – 22 kHz and 17 – 19.2 kHz, the data symbol consists of 23 subcarriers spaced at 100 Hz apart.

The dominant echo, spans the first 3 ms of the reverberation period, making it impossible to modulating bits at the sub-millisecond level in the time domain. Second, the spectral leakage aggravates as the length of bit duration decreases. Figure 7b depicts the spectrum of chirp signals modulated with on-off keying with different bit duration. Though with a duration of 5 ms, the leakage significantly increases, leading to the perception of the audience.

Low Packet Error Rate. AIRCODE assigns the audio link as the control channel. Thus, it must be reliable with a low packet error rate (PER). As audio channel suffers from multi-path effect and frequency selectivity of speakers and microphones, pilot symbols are usually used to account for different channel responses of subcarriers and decide demodulation thresholds accordingly [38]. However, the threshold-based method requests that all bits '1' across all symbols are modulated with the same amplitude, leading to waste of energy and low SNR, when some symbols contain only a few bits '1'.

5.2 Audio Packet Design

Figure 8a shows the audio packet format in both time and frequency domain. To cope with the video channel, we set the packet duration as $\frac{1}{15}$ s, corresponding to 4 data frames, during which video configuration is unlikely to change.

To solve the challenge of short packet duration, AIRCODE modulates bits in the frequency domain and sets data symbol length as 20 ms, which is about 4 times longer than the major echo and has low spectral leakage. To further avoid the impact of reverberation, AIRCODE separates the preamble and the data into different frequency bands. Specifically, as shown in Figure 8b, the chirp preamble is within 20 – 22 kHz, and the data symbol is within 17 – 19.2 kHz. Both of which are inaudible to most human ears [42]. To further reduce spectral leakage, AIRCODE applies a tapered-cosine window on the preamble and each data symbol.

To solve the challenge of low packet error rate, AIRCODE adopts the two-layer coding scheme. Specifically, each packet contains 12 control bits. The 12 control bits are first encoded into 23 bits codeword via Golay code [23], which can correct any 3-bit error. Then, to fully exploit signal power allocated to the control signal, AIRCODE further adopts Manchester coding, which encodes bits '1' as '10' and bits '0' as '01'. During modulation, AIRCODE assigns 22 Manchester bits of the first 11 bits and the first Manchester bit of the 23rd bit of the Golay codeword to the 23 subcarriers of the first data symbol, and the rest 23 Manchester bits to the second data symbol. Figure 8b shows an example of the spectrum of one data symbol, where the subcarrier spacing is 100 Hz. Since each pair of Manchester bits are assigned to either adjacent subcarriers or data symbols, they experience similar channel responses caused by multipath effect and frequency selectivity of audio devices. Thus, the receiver can demodulate the data symbol by directly comparing amplitudes of pairs of Manchester bits without thresholds, and the sender can distribute all allocated signal power evenly to bits '1' in each data symbol, to increase average SNR and reduce overall PER.

6 Video Data Channel

The screen-camera link supports high throughput communication with imagery codes invisible to human eyes. The basic principle is to exploit the low-pass flicker fusion property of human eyes. Specifically, screen-camera communication generates high-rate complementary frames by inversely modifying the lightness of a pair of adjacent raw video frames. When videos are played at high frame rate, e.g., 120 FPS, human eyes can only observe raw video frames, which are the average of complementary frames, and cannot perceive the embedded data frames. In contrast, cameras with a high capturing rate can still acquire and decode data frames. AIRCODE takes the idea of hidden screen-camera communication, but focus on lower BER as well as higher data rate, by incorporating robust speaker-microphone link as the control channel, and accurate screen detection based on visual odometry.

Figure 6 shows the video communication part of AIRCODE. At the sender side, data bits are firstly encoded with a concatenated coding scheme, including Reed-Solomon (RS) coding as source code and Convolutional coding as channel code. The coding scheme is adaptively selected according to the texture complexity of the raw video, to achieve high throughput in plain frames, as well as low BER in textured frames. The encoded bits are modulated as data frames and embedded into pairs of complementary raw video frames. At the receiver side, AIRCODE inputs camera images into the screen detection process to obtain successive locations of the screen in frames. Meanwhile, the associated audio signal is pro-

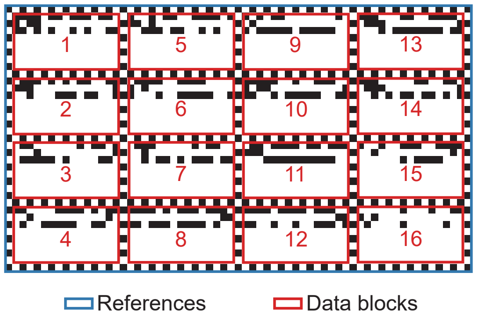


Figure 9: Data frame design. The frame consists of 16 data blocks, surrounded by black-and-white lines as references.

cessed to get corresponding metadata. Then, camera images are equalized with screen location information, and data is finally demodulated and decoded with metadata.

6.1 Data Frame Design

Figure 9 shows AIRCODE's data frame. AIRCODE follows the basic design of ChromaCode, but removes the code preamble blocks in ChromaCode that redundantly encode metadata of data frames, thanks to the use of the audio channel. AIRCODE divides the *whole* image frame into 16 data blocks with equal size. Similar to ChromaCode, each data block contains several data cells and is surrounded with alternate black and white cells as references. The encoded bits are interleaved and filled into data blocks. Specifically, the *i*-th bit is assigned to the $\lfloor \frac{i}{16} \rfloor$-th cell in the $(i \mod 16)$-th block.

The metadata of a data frame, including the levels of RS coding and Convolutional coding for encoding, and the data cell size for modulation, are transmitted as control packets over the reliable acoustic channel. Each control packet of metadata contains 12 data bits, with 2 bits for the RS coding level, 1 bit for the Convolutional coding level, 6 bits for the data cell size, and the rest 3 bits for parity bits for these three parameters. To save audio channel capacity, four successive data frames (embedded in eight video frames) share the same set of parameters in one control packet, as the video content is unlikely to drastically change during such a short period.

To avoid floating time offsets between the audio control channel and the video data channel, AIRCODE periodically (e.g., 5 s) sends video frames with the highest coding level and audio packets. These video frames and audio packets contain the synchronization information. At the receiver, each control audio packet will be aligned with 4 video data frames according to the latest synchronizing frame.

6.2 Adaptive Error Correction

In practice, textures of carrier video can considerably impact data transmission. In particular, the BER of the screen-camera link is lower in plain frames consisting of pixels in similar colors but significantly higher in textured frames. Therefore, to ensure robust data transmission, an error correction scheme with high correction ability should be used for textured frames. In contrast, a scheme with high efficiency yet potentially low correction ability will be preferred in plain frames to achieve high throughput.

This observation leads to the adaptive error correction scheme of AIRCODE, which adjusts the screen-camera channel parameters in metadata, i.e., RS coding level, Convolutional coding level, and data cell size, according to the dynamic texture complexity of the carrier video. In detail, AIRCODE supports 40 levels of data cell size, 2 levels of Convolutional coding and 4 levels of RS coding, which result in 320 levels of error correction. Given a video, a subset of error correction levels are selected and linearized by setting the data cell size as the most significant factor and the RS coding as the least significant factor. The video is segmented where a segment can be as short as 8 frames (i.e., $\frac{1}{15}$ s), corresponding to 1 audio packet. For each video segment, AIRCODE quantizes the texture complexity by calculating the average contrast of pixels against their neighboring 9×9 pixels, and maps the texture complexity to the error correction level.

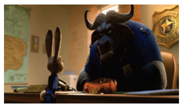
7 Evaluation

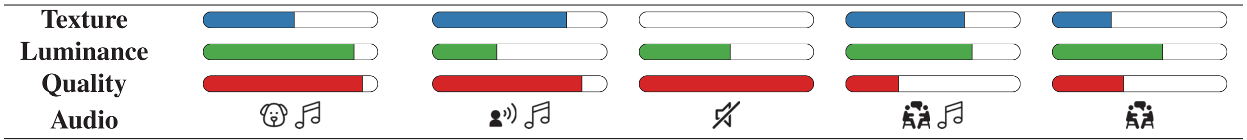
7.1 Experimental Methodology

Experiment Setting and implementation. We use an AOC AGON AG271QX monitor as the sender of encoded videos and audios. It supports 120Hz refresh rate along with 1920×1080 resolution and has two audio speakers on its back. One Nexus 6P smartphone without hardware modification is used as the receiver. videos and audios are played by a DELL XPS 8900-R17N8 desktop, which equips a GeForce RTX 2070 GPU, and MPV player with VDPAU acceleration. AIRCODE uses FFmpeg to extract and compress video frames and audio tracks. OpenCV is used for the implementation of the rule-based screen detection algorithm and the screen tracking algorithm. AIRCODE exploits MediaCodeC interface for efficient hardware video coding and Boost library for implement of audio coding.

Video & Audio Selection. Table 1 shows the selected videos and their audios. For the video with pure gray images, an empty audio track is used as its origin audio. The bars indicate the levels of texture complexity, luminance and quality of the selected video contents, which cover a variety of combination. The more portion of a bar is filled with colors, the higher level of the metric is achieved by the corresponding video. E.g., the video Dynasties (D) has complex texture, low luminance and high quality. The corresponding audio contents include different types, e.g., human speaking, animal sound, an object moving and dropping, and background music, indicated by the icons. In practice, the quality of a streaming video frequently switches. However, frames between two adjacent switches still have the same quality. As AIRCODE encodes data at the frame level, frequent switching of streaming videos has a minor impact on AIRCODE.
Evaluation Metrics. For screen-camera communication, we compare AIRCODE with ChromaCode and InFrame++ [36]

Table 1: Origin primary video & audio clips used in the experiments and their characteristics

	Big Buck Bunny (B)	Dynasties (D)	Gray (G)	Journey (J)	Zootopia (Z)
					



in terms of throughput, goodput and data BER. Throughput is the amount of all received bits. Goodput is the effective throughput of correctly decoded data bits. And BER is the number of error bits divided by total received data amount. ChromaCode uses the rule-based screen detection algorithm, and InFrame++ uses visible markers and alignment patterns at screen corners and borders. Since the speaker-microphone link serves as control channel and does not convey much data, we only evaluate its reliability in terms of BER and PER.

User Study. We invite 12 participants, including students and staffs on campus, to score the watching experience of encoded videos and audios. The ages of these participants range from 20 to 30. Nine participants are male, while the rest 3 are female. In total 40 videos together with their audios are prepared and tested. Videos and audios with or without data are randomly shuffled and played. Participants, not knowing their orders, are asked to give scores on the quality of them. Denote the scores for encoded videos and audios as S_enc and that for raw videos and audios as S_raw, the final scores for encoded videos and audios are normalized as $\frac{S\_enc}{S\_raw}$. In some cases, the normalized score is larger than 1, meaning that participants cannot statistically differentiate encoded videos and audios from their original counterparts. Moreover, to check whether young children can perceive the embedded code, we intentionally invite one 5-year-old child to watch the testing videos and audios and give feedback. Considering that children at such a young age may not have enough comprehension ability as adults to understand the meaning of the experiment, we neutrally ask him whether he can see or hear anything peculiar in videos or audios. All experiments are approved by our IRB, and do not raise any ethical issues.

7.2 Overall Performance

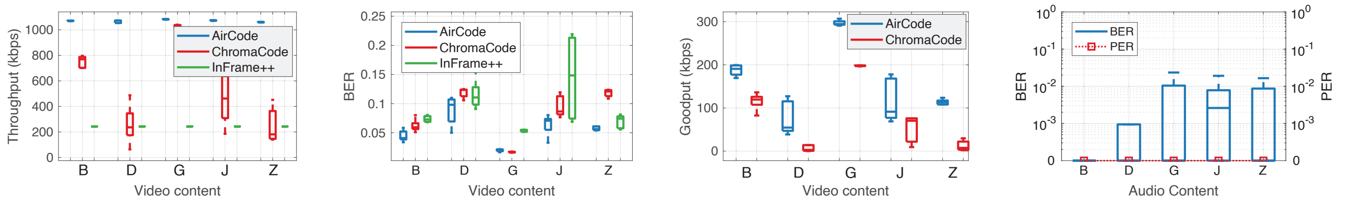
Performance of the communication processes. We first report the overall performance of AIRCODE in terms of throughput, goodput, and BER of the video channel, and BER and PER of the audio channel.

Figure 10a-10c show the performance of screen-camera communication in AIRCODE with different video contents. Two state-of-the-art approaches, ChromaCode and InFrame++ are evaluated for comparison. As shown in Figure 10a, AIRCODE achieves 1 Mbps throughput for all 5 videos, attributed to robust transmission of metadata through speaker-microphone channel. In contrast, though encoded with the whole screen, ChromaCode has lower throughput, especially with the videos 'D' and 'Z'. InFrame++ has the lowest throughput, due to its inefficiently CDMA-like coding scheme. Figure 10b shows that except the video 'D', the average BER achieved by AIRCODE is about 5%, which is statistically lower than its counterparts, owing to the accurate screen detection process. Figure 10c further compares the goodput of AIRCODE and ChromaCode. Due to the additive effect of accurate screen detection and robust metadata transmission, AIRCODE outperforms ChromaCode throughout all testing videos. Another key observation from Figure 10b and 10c is that the system performance is highly impacted by video content, where the video with more plain frames (i.e., 'G') tends to have lower BER and higher goodput. AIRCODE takes adaptive error correction, which further increases the reliability of AIRCODE with different video contents.

Figure 10d shows the BER and PER of the speaker-microphone link, whose reliable communication is of great importance for the overall system. Across all types of audio contents, the BER of the audio control channel consistently remains below 1%. With the help of Golay code, over 99.9% audio control packets can be successfully decoded, providing robust metadata for the screen-camera link.

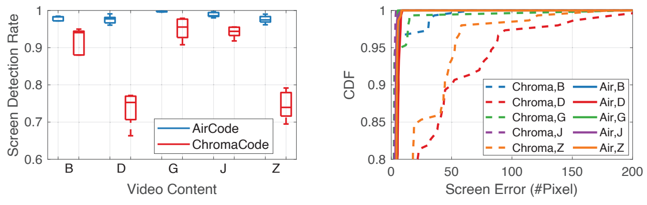
Performance of the screen tracking process. The screen detection rate and tracking error of AIRCODE are evaluated and compared with the rule-based algorithm of ChromaCode. Since labeling ground-truth of all video frames is labor-intensive, we indirectly calculate screen detection rate as the ratio $\frac{r_{method}}{r_{base}}$, where r_{method} is the frame decoding rate of the method, and r_{base} is the frame decoding rate when the camera is fixed at some sampling locations. To calculate screen tracking error, we randomly sample frames with 0.5s intervals from all videos and mark the ground-truth screen in them for comparison.

On average, AIRCODE takes 0.79 s to initialize. Figure 11a shows that AIRCODE consistently detects the screen in over 97% frames across different types of videos. In contrast, the rule-based algorithm suffers from confusing video contents, and ChromaCode can detect the screen in over 90% frames in videos 'B', 'G' and 'J', but only about 70% frames in videos 'D' and 'Z'. Figure 11b further shows the tracking error of the two methods in terms of image pixels. AIRCODE achieves



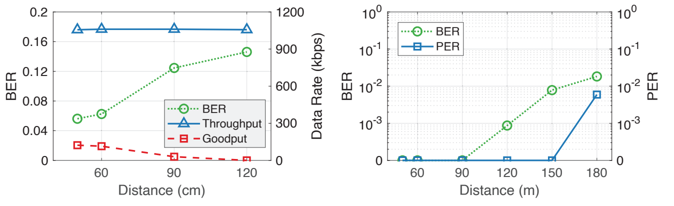
(a) Video throughput (b) Video BER (c) Video goodput (d) Audio BER & PER

Figure 10: Performance of the screen-camera communication and the speaker-microphone communication.



(a) Screen detection rate (b) Screen tracking error

Figure 11: Performance of screen tracking.



(a) Video (b) Audio

Figure 12: Impact of monitor-phone distance.

Table 2: Benefits of individual module.

		Throughput (kbps)	BER (%)	Goodput (kbps)
Screen Detection	**Tracking**	1086.4	6%	139.5
	Rule	1073.8	8.3%	77.3
Control Channel	**Audio**	1084.3	4.4%	149.1
	Video	893.7	4.6%	144.3
Error Correction	**Adaptive**	1086.3	5%	159.4
	Fixed	1086.4	4.4%	149.4

consistently small tracking error for all videos, while the performance of ChromaCode is highly related to video content. Specifically, all tracking errors of AIRCODE are within 9 pixels. In contrast, 12.8% tracking errors of ChromaCode exceed 9 pixels, and the maximal error is even beyond 60 pixels. This validates the effectiveness of screen tracking.

Benefits of individual module. We further evaluate the individual performance improvement from the screen detection, control channel and error correction of AIRCODE. When one module is evaluated, the other modules are disabled (error correction) or assumed to be perfect (screen detection and control channel). Table 2 shows the average system performance. First, the screen tracking method of AIRCODE has lower BER and significantly higher goodput than the rule-based method of ChromaCode, thanks to its consistently accurate screen tracking. Second, comparing with sending metadata via video, using the audio channel improves throughput and goodput. On one hand, it saves the area where metadata has to be duplicated for reliable detection in video frames. On the other hand, the audio channel is more robust and overcomes the occasion when even the duplicated metadata in video frames cannot be correctly decoded. Third, by considering the texture complexity of videos, although the adaptive error correction of AIRCODE has a slightly higher BER than error correction with fixed coding level, it achieves overall 10 kbps more goodput, as more data bits are adaptively encoded.

7.3 Parameter Study

Distance. Figure 12 shows the impact of distance between the monitor and the phone. For the video part, the data cell size is fixed to 10×10. As shown in Figure 12a, the goodput of AIRCODE abruptly decreases when the distance becomes longer than 90 cm. One main reason is that small data cells become hard to recognize when the distance increases. It suggests that larger data cells should be used at longer distances, however, with the sacrifice of the throughput. The other reason for performance deterioration is that the focal length of the receiving camera will automatically change more frequently with larger distances, leading to more blurry video records.

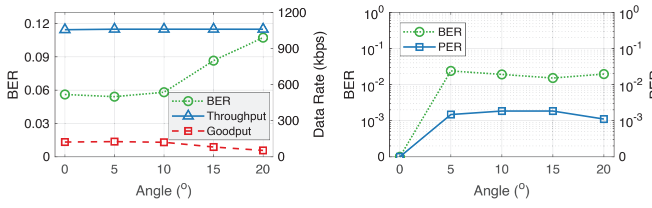
For audio part, as shown in Figure 12b, while the BER gradually increases with the distance, due to decay of receiving signal strength, all errors can be corrected and the PER remains 0 when the phone is within 150 cm from the monitor, which fulfills the requirement of the video data channel.

Angle. We evaluate the impact of relative angles between the phone and the monitor, as shown in Figure 13. For video, Figure 13a shows that the goodput decreases while the BER increases with a large angle. The main reason is that the projection distortion becomes severer as the camera deviates from the front of the monitor, and in this case, video frames cannot be easily decoded even with accurate screen estimation.

The same trend also appears in the speaker-microphone link, where the BER increases with the angle between the monitor and the phone, as shown in Figure 13b. It is mainly due to the high directivity of the monitor loudspeaker. Even though, AIRCODE still achieves ultra-low PER of about 1‰.

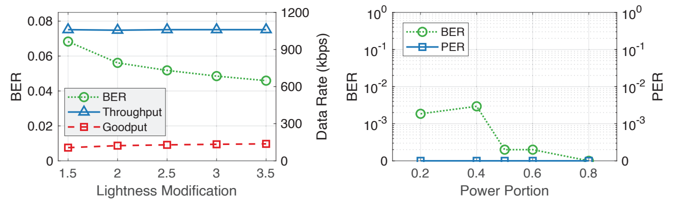
Signal strength. For the video part, the signal strength is represented as the quantity of lightness modification of data frames on raw video frames. As shown in Figure 14a, the BER gradually decreases as the amplitude of lightness modification increases, since data cells can be more easily recognized from the raw video content, and thus correctly decoded.

For the audio part, the signal strength is represented as the



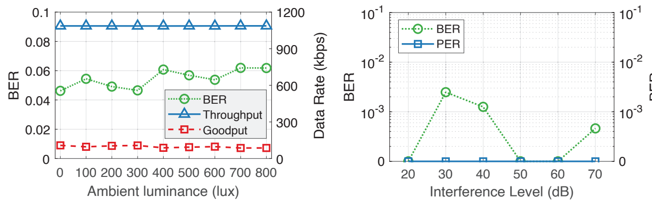
(a) Video (b) Audio

Figure 13: Impact of monitor-phone angle.



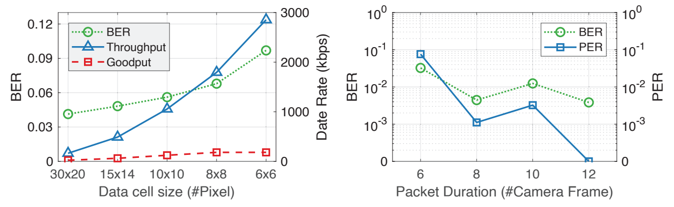
(a) Video (b) Audio

Figure 14: Impact of signal strength.



(a) Video (b) Audio

Figure 15: Impact of surrounding interference.



(a) Video (b) Audio

Figure 16: Impact of frame size.

portion of power allocated to the embedded signal. As shown in Figure 14b, the BER declines as the signal strength of the embedded signal increases, while the PER remains 0 even when only 20% power is allocated.

Background interference. We further evaluate the robustness under interferences from the environment. For the video part, the major source of interference is ambient luminance. Typical indoor ambient luminance ranging from 0 to 800 lux is evaluated. Specifically, the ambient luminance less than 600 lux is generated by an unshielded lamp suspended over the screen-camera link, while that between 600 and 800 lux is generated by natural sunlight during the daytime. A photometer is placed adjacent to the camera to measure the ambient luminance. As shown in Figure 15a, while strong ambient luminance reduces the contrast between complementary frames, the system performance only slightly degrades. It is concluded that ambient luminance has little impact on data transmission.

For the audio part, the major source of interference is environmental noises. For evaluation, we set AIRCODE to work normally at 1 m away from the monitor, and let a loudspeaker be 3 m away from the phone and play various types of audios as background interferences. The volume of the loudspeaker is adjusted to create interferences varying from 20 dB (0% volume) to 70 dB (100% volume) at the speaker. Figure 15b shows that the variance of BER is small and the PER remains 0 no matter how loud the loudspeaker plays interferences, demonstrating the robustness of the audio channel.

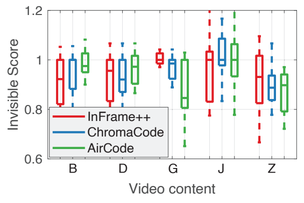
Frame size. For the video part, the frame size refers to data cell size. Figure 16a shows the data rate and BER with different data cell sizes. By gradually reducing data cell size, the throughput boosts from 163 Kbps to 2855 Kbps. AIRCODE achieves a high goodput of 182 Kbps with 8×8 data cell but saturates when data cells are further scaled down to 6×6, due to a higher chance of recognition failure and more bit errors. In practice, given the requirement of the BER and communication distance of an application, AIRCODE can trade-off the throughput by selecting appropriate data cell sizes.

For the audio part, the frame size refers to packet duration. As shown in Figure 16b, both BER and PER decrease with longer packets, as longer data symbols are less affected by reverberation and have finer frequency resolution for demodulation. By observing that the decline of BER is not obvious when the packet duration exceeds the length of 8 video frames (i.e., $\frac{1}{15}$ s), AIRCODE adopts $\frac{1}{15}$ s as the default packet duration for the audio control channel.

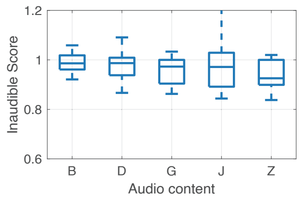
7.4 Perception Test

Video & audio content. Figure 17 shows the variation of code insensibility with different video and audio content. It is observed that all normalized scores for videos are higher than 0.6 and those for audios are higher than 0.8 for AIRCODE, indicating relatively high insensibility of the system. Among the results, some normalized scores are equal or even higher than 1, meaning that the quality of the encoded videos and audios is sufficiently high that participants cannot distinguish encoded videos and audios from original ones. For different video contents, higher scores are given to videos 'B', 'D' and 'J', which have more textured frames. It means that larger lightness modification can be applied to textured frames to ensure low BER without much impacting raw videos.

Biases of audiences. Scores given by 12 participants demonstrate different biases of their perception, as shown in Figure 18. For example, the participant 2 gives the statistically highest scores for encoded videos. From the interview after the experiment, he admits that it is difficult for him to recognize embedded data hidden in videos, meaning that the encoded videos have almost the same quality as the raw videos for him. In contrast, the participant 10 gives the lowest scores for encoded videos of all three approaches, meaning that he is relatively sensitive to unnatural alternation in videos. Among all 12 participants, only 2 (i.e., 6 and 7) of them give the highest average score to InFrame++. Yet, considering the scoring fluctuations of participants, there are no significant

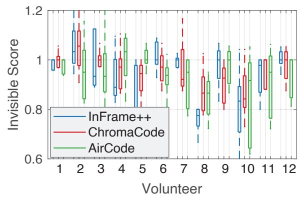


(a) Video

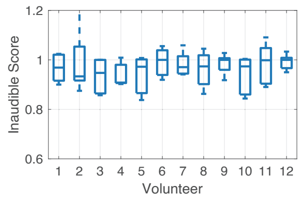


(b) Audio

Figure 17: Content diversity.



(a) Video



(b) Audio

Figure 18: Audience Diversity.

quality differences among the three approaches.

Besides 12 adults, one 5-year-old child also participates in the test. We first show him training videos with perceptible visual codes and acoustic noises. Then, the testing videos and audios are played intermittently and the child answers whether there are such codes or noises. To avoid biased hints to the child, we only give the instruction at the beginning and remain silence during playing of all testing videos. The result is that both encoded videos and audios do not discomfort the child, and he cannot observe or hear any interferences that do not belong to the origin videos and audios, demonstrating high insensibility of AIRCODE even for children.

8 Related Work

Hidden screen-camera communication. Hidden communication with the screen-camera link conveys information without interfering watching experience of audience [11, 15, 24, 30, 36, 37]. Along this direction, pioneer works, InFrame [37] and InFrame++ [36], propose and implement the basic idea, which is to switch barcodes with complementary lightness and leverage the flicker fusion property of human eyes. However, InFrame++ uses visible markers at corners of video frames for block localization, which explicitly interferes audience and consumes precious screen spaces. Extensive efforts are further made to improve the audience's experience. TextureCode [24] adaptively embeds data only in textured regions, which however is limited in throughput. PixNet [26] designs 2D OFDM symbols that can correct perspective distortions. However, they are not guaranteed to be invisible after embedded into video frames and the frequency property of OFDM symbols is destructed by changing video contents. ChromaCode [41] improves code invisibility by modifying lightness in uniform color space and achieves full imperception. However, pursuing code invisibility increases the difficulty of detecting code frames. In comparison, ensuring code invisibility, AIRCODE adopts visual odometry for accurate screen detection and achieves lower BER and higher goodput.

Speaker-microphone communication. Communication with off-the-shelf speaker-microphone links has been studied in [12, 14, 16, 22, 25, 38, 40]. Dhwani [22] adopts OFDM with PSK modulation and realizes acoustic near field communication. However, it works in the audible 6 – 7 kHz band, which interferes hearing experience of the audience and cannot be embedded in normal audio. Chirp signal is widely used in hidden acoustic communication in [12, 14, 16], for its fine correlation property. However, data rates achieved with chirp-based communication are less than 100 bps. Dolphin [38] proposes dual-mode unobtrusive communication, which still only achieves average data rates of up to 500 bps, even with the usage of a much wider frequency band from 8 kHz to 20 kHz. Noting the limitation of throughput and the advantage of the reliability of speaker-microphone communication with off-the-shelf devices, AIRCODE designs reliable and agile acoustic communication as the control channel of screen-camera link with high frame rate, boosting the overall throughput of the whole communication system. The acoustic channel will congest when multiple screens supporting AIRCODE coexist. In such a case, the screens should sense the audio channel before accessing it. Besides, audio packets should be modified to identify their belongings to the screens. We leave the support of multiple screens as future work.

Visual Odometry. Monocular visual odometry has the goal of estimating camera trajectory and reconstructing the environment with monocular commercial cameras [1–7, 13, 21, 39]. Pioneer works, e.g., MonoSLAM [2], applies filtering-based approach. Later, optimization-based methods [13] gradually substitutes the filtering-based ones for their tracking accuracy and efficiency. ORB-SLAM [21] is the representative work that uses ORB features in tracking and mapping. AIRCODE exploits the idea of visual odometry to accurately detect screens in images, enabling robust and practical screen-camera communication.

9 Conclusion

In this paper, we present the design and implementation of AIRCODE, the first hidden screen-camera communication system that achieves considerably high data rates of >1Mbps. AIRCODE is also the first system of its kind that exploits an invisible video and inaudible audio dual channel. With the high data rates being supported, AIRCODE opens up new opportunities and underpins various applications yet to be imagined for hidden screen-camera communication.

Acknowledgement

We thank the anonymous reviewers of NSDI, and our shepherd Rajalakshmi Nandakumar. This work is supported in part by the NSFC under grant 61832010, 61632008, 61672319, and Microsoft Research Asia.

References

[1] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of IEEE ICCV*, 2003.

[2] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.

[3] Erqun Dong, Jingao Xu, Chenshu Wu, Yunhao Liu, and Zheng Yang. Pair-navi: Peer-to-peer indoor navigation with mobile visual slam. In *Proceedings of IEEE INFOCOM*, 2019.

[4] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Proceedings of IEEE CVPR*, 2006.

[5] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.

[6] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Proceedings of Springer ECCV*, 2014.

[7] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Proeedings of IEEE ICRA*, 2014.

[8] Ezzeldin Hamed, Hariharan Rahul, and Bahar Partov. Chorus: truly distributed distributed-mimo. In *Proceedings of ACM SIGCOMM*, 2018.

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of IEEE ICCV*, 2017.

[11] M. Izz, Z. Li, H. Liu, Y. Chen, and F. Li. Uber-in-light: Unobtrusive visible light communication leveraging complementary color channel. In *Proceedings of IEEE INFOCOM*, 2016.

[12] Soonwon Ka, Tae Hyun Kim, Jae Yeol Ha, Sun Hong Lim, Su Cheol Shin, Jun Won Choi, Chulyoung Kwak, and Sunghyun Choi. Near-ultrasound communication for tv's 2nd screen services. In *Proceedings of ACM MobiCom*, 2016.

[13] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of IEEE ISMAR*, 2007.

[14] Hyewon Lee, Tae Hyun Kim, Jun Won Choi, and Sunghyun Choi. Chirp signal-based aerial acoustic communication for smart devices. In *Proceedings of IEEE INFOCOM*, 2015.

[15] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T. Campbell, and Xia Zhou. Real-time screen-camera communication behind any scene. In *Proceedings of ACM MobiSys*, 2015.

[16] Qiongzheng Lin, Lei Yang, and Yunhao Liu. Tagscreen: Synchronizing social televisions through hidden sound markers. In *Proceedings of IEEE INFOCOM*, 2017.

[17] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R Smith. Ambient backscatter: wireless communication out of thin air. In *Proceedings of ACM SIGCOMM*, 2013.

[18] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133, 1981.

[19] JM Martínez Montiel, Javier Civera, and Andrew J Davison. Unified inverse depth parametrization for monocular slam. Robotics: Science and Systems, 2006.

[20] Carlos Morimoto and Ramalingam Chellappa. Fast electronic digital image stabilization. In *Proceedings of IEEE ICPR*, 1996.

[21] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[22] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, Venkat Padmanabhan, and Ramarathnam Venkatesan. Dhwani: secure peer-to-peer acoustic nfc. In *Proceedings of ACM SIGCOMM*, 2013.

[23] Andre Neubauer, Jurgen Freudenberger, and Volker Kuhn. *Coding theory: algorithms, architectures and applications*. John Wiley & Sons, 2007.

[24] Viet Nguyen, Yaqin Tang, Ashwin Ashok, Marco Gruteser, Kristin Dana, Wenjun Hu, Eric Wengrowski, and Narayan Mandayam. High-rate flicker-free screen-camera communication with spatially adaptive embedding. In *Proceedings of IEEE INFOCOM*, 2016.

[25] Aditya Shekhar Nittala, Xing-Dong Yang, Scott Bateman, Ehud Sharlin, and Saul Greenberg. Phoneear: interactions for mobile devices that hear high-frequency sound-encoded data. In *Proceedings of ACM EICS*, 2015.

[26] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. Pixnet: Interference-free wireless links using lcd-camera pairs. In *Proceedings of ACM MobiCom*, 2010.

[27] Nirupam Roy and Romit Roy Choudhury. Ripple ii: Faster communication through physical vibration. In *Proceedings of USENIX NSDI*, 2016.

[28] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of IEEE ICCV*, 2011.

[29] Daniël Schoonwinkel. *Practical measurements of Wi-Fi Direct in content sharing, social gaming android applications*. PhD thesis, Stellenbosch: Stellenbosch University, 2016.

[30] Shuyu Shi, Lin Chen, Wenjun Hu, and Marco Gruteser. Reading between lines: High-rate, non-intrusive visual codes within regular videos via implicitcode. In *Proceedings of ACM UbiComp*, 2015.

[31] Ernst Simonson and Josef Brozek. Flicker fusion frequency: background and applications. *Physiological reviews*, 32(3):349–378, 1952.

[32] Myeong-Gyu Song, Hyun-Woo Baek, No-Cheol Park, Kyoung-Su Park, Taeyong Yoon, Young-Pil Park, and Soo-Cheol Lim. Development of small sized actuator with compliant mechanism for optical image stabilization. *IEEE Transactions on Magnetics*, 46(6):2369–2372, 2010.

[33] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. In *Proceedings of IEEE ISMAR*, 2013.

[34] Tiziano Tommasini, Andrea Fusiello, Emanuele Trucco, and Vito Roberto. Making good features track better. In *Proceedings of IEEE CVPR*, 1998.

[35] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*. Springer, 1999.

[36] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing Zeng. Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of ACM MobiSys*, 2015.

[37] Anran Wang, Chunyi Peng, Ouyang Zhang, Guobin Shen, and Bing Zeng. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In *Proceedings of ACM HotNet*, 2014.

[38] Qian Wang, Kui Ren, Man Zhou, Tao Lei, Dimitrios Koutsonikolas, and Lu Su. Messages behind the sound: real-time hidden acoustic signal capture with smartphones. In *Proceedings of ACM MobiCom*, 2016.

[39] Jingao Xu, Hengjie Chen, Kun Qian, Erqun Dong, Min Sun, Chenshu Wu, Li Zhang, and Zheng Yang. ivr: Integrated vision and radio localization with zero human effort. *Proceedings of the ACM IMWUT*, 3(3):1–22, 2019.

[40] Hwan Sik Yun, Kiho Cho, and Nam Soo Kim. Acoustic data transmission based on modulated complex lapped transform. *IEEE Signal Processing Letters*, 17(1):67–70, 2010.

[41] Kai Zhang, Chenshu Wu, Chaofan Yang, Yi Zhao, Kehong Huang, Chunyi Peng, Yunhao Liu, and Zheng Yang. Chromacode: A fully imperceptible screen-camera communication system. In *Proceedings of ACM MobiCom*, 2018.

[42] Eberhard Zwicker and Hugo Fastl. *Psychoacoustics: Facts and models*, volume 22. Springer Science & Business Media, 2013.