Optical Engineering

OpticalEngineering.SPIEDigitalLibrary.org

# Developing concurrent coding: an unconventional encoding scheme applied to visible light communications

David M. Benton

SPIE.

David M. Benton, "Developing concurrent coding: an unconventional encoding scheme applied to visible light communications," *Opt. Eng.* **59**(4), 046107 (2020), doi: 10.1117/1.OE.59.4.046107

# Developing concurrent coding: an unconventional encoding scheme applied to visible light communications

**David M. Benton***
Aston University, Aston Institute of Photonic Technologies, Birmingham, United Kingdom

**Abstract.** An unconventional encoding scheme called concurrent coding has recently been demonstrated and shown to offer interesting features and benefits in comparison to conventional techniques, e.g., robustness against burst errors and improved efficiency of transmitted power. This concept has been demonstrated for the first time with optical communications, where standard light-emitting diodes have been used to transmit information encoded with concurrent coding. The technique successfully transmits and decodes data despite unpredictable interruptions to the transmission causing significant dropouts to the detected signal. The technique also shows how it is possible to send a single block of data in isolation with no presynchronization required between transmitter and receiver, and no specific synchronization sequence appended to the transmission. Our work also demonstrates the successful use of multithreaded (overlaid) concurrent codes for the first time. © *2020 Society of Photo-Optical Instrumentation Engineers (SPIE)*
[DOI: 10.1117/1.OE.59.4.046107]

**Keywords:** encoding; free space optics; visible light communication.

Paper 20200134 received Feb. 6, 2020; accepted for publication Apr. 8, 2020; published online Apr. 23, 2020.

## 1 Introduction

Concurrent coding is a unique method of encoding that differs significantly from conventional encoding methods[1–5] and has recently been investigated as a robust method of protecting data transmission.[6] Conventional approaches to protecting information transfer against the corrupting effects of noise see the characteristics of a block of information being encoded with and into the information, thus increasing the amount of data being sent and reducing the information rate. This characteristic information is almost always linked locally to the information itself, such as a parity bit next to the data bits it represents. This is true for block codes (see Refs. 7 and 8) including cyclic codes, Golay codes, and Reed–Solomon codes,[9] and is equally true for convolutional codes such as turbo coding[10] and Viterbi codes.[11] These schemes are designed to combat the effects of random noise affecting individual isolated bits and do not deal effectively with nonrandom errors.[12,13] To combat the effect of burst errors that affect a contiguous set of bits in a nonrandom way, interleaving is typically used. This deliberately converts the local connection between the related coded bits by distributing them in a regular fashion throughout a larger codeword. Thus, random errors and burst errors are treated separately. Alternative approaches, such as fire codes[14] and Reed–Solomon[9,15] encoding, treat data as a set of symbols and correct for symbol errors to help encompass nonrandom errors. Concurrent coding connects the characteristics of a block of data to the codeword in which it is transmitted. The data block is encoded globally into the codeword in a single step.

Concurrent coding is a binary asymmetric technique that encodes and decodes message vectors rather than bits but can only be implemented on binary modulation schemes. It has shown robustness against noise and in particular burst errors.[6] It was originally conceived as a method for providing protection against jamming, an alternative to spread spectrum techniques.[1–3] Concurrent coding can achieve jamming resistance without the need for a shared secret key as is required with code-division multiple access (CDMA) coding. Concurrent coding works by repeated use of a hashing function to distribute information throughout a codeword.

*Address all correspondence to David M. Benton, E-mail: d.benton@aston.ac.uk

Many hashing functions are appropriate,[4,6,16] but the emphasis is on distribution rather than security although attacks against the algorithm have been examined.[17] Recently, Hanifi et al.[18] have developed a new concurrent code based on the use of monotone Boolean functions.

Concurrent codes are appealing for their robust nature but also for other properties such as the efficient use of transmitted energy and the relative simplicity of the scheme in comparison to other comparable techniques such as Reed–Solomon encoding. Concurrent codes degrade more gracefully than interleaved codes where data loss increases dramatically once the error fraction increases beyond the code's capacity to correct it.[19] With concurrent codes, data are (ideally) not lost, only obscured. Thus, concurrent coding could be a suitable protocol to apply to free-space optical (FSO) connections where burst errors are a particular problem due to beam interruptions, misalignments, and atmospheric scintillation. On–off keying (OOK) is a commonly used intensity modulation scheme in optical communications[20–22] in which a binary representation is obtained from the presence or absence of light—hence optical communication is a natural ally for concurrent coding. Used with direct detection, OOK requires knowledge of the instantaneous fading coefficient of the channel for dynamic thresholding to be applied. In this sense, using concurrent coding with OOK, the encoding scheme is the modulation scheme. Other modulation schemes such as pulse position modulation, which are typically symbol transmission schemes, are not compatible with concurrent coding unless implemented in an asymmetric binary manner (i.e., large slot for binary 1), which would reduce their efficiency. The effect of atmospheric turbulence in FSO links can result in very large and deep signal fades. No matter which encoding scheme is used, the need for large-scale interleaving has been required for the successful operation of the encoding scheme. Concurrent coding may be the first genuine alternative to interleaving in FSO communication systems.

Therefore, it is appropriate to compare the behaviors of the encoding methods. However, it is important to first highlight the nature of concurrent codes to appreciate the difference with conventional codes and to get a proper appreciation of the comparative behaviors.

- Concurrent codes are an asymmetric binary encoding system that generates indelible marks to represent digital 1s into a codeword. Marks are substantive; a positive presence such as pulses of energy and cannot be removed to randomly convert a 1 back to a 0. A zero is then the absence of energy or substance, which can be converted to 1 by noise (or jamming).
- The result of using indelible marks is that encoded message vectors cannot be removed and will always be decoded.
- Original message vectors cannot be corrupted but can be obscured by spurious decodings called hallucinations.
- Providing protection for transmitted data against random errors, burst errors, and jamming might involve separate steps for each error and could be represented as

$$\text{Data} \rightarrow \text{Parity encoding} \rightarrow \text{Interleaving} \rightarrow \text{Spread spectrum coding} \rightarrow \text{Transmission}.$$

In contrast, concurrent coding follows:

$$\text{Data} \rightarrow \text{Concurrent coding} \rightarrow \text{Transmission}.$$

- Marks in the codeword are shared by many input vectors thus leading to improved efficiency in terms of transmitted energy.
- Concurrent codes contain an inherent method of synchronization
- Concurrent codes are significantly easier to comprehend than Reed–Solomon encoding or other block codes.

## 2 Description of Concurrent Coding

Descriptions of how concurrent coding works are given in previous references[1–6] and briefly given here for completeness. The concurrent coding principle encodes a digital word into a much larger codeword space by hashing incrementally increasing subsections (or prefixes) of the

digital word to produce addresses in the larger codeword into which indelible marks are placed to represent 1s. Thus, a 4-bit message vector $abcd$ would place marks in the codeword at positions given by $H(d)$, $H(cd)$, $H(bcd)$, and $H(abcd)$, where $H(x)$ represents the output of hash function on the digital subvector $x$. This is represented schematically in the decoding section of Fig. 1. The hash function is not explicitly defined and can be any suitable redistribution function. The process is repeated for additional message vectors again placing marks into the codeword. When all message vectors are installed, the codeword can be transmitted. Decoding the received codeword proceeds as follows: the receiver computes the mark positions for $H(0)$ and $H(1)$ and then checks the codeword to see if any of these marks are present—so $H(d)$ would correspond to one of these. These mark positions form the first branches of a decoding tree and where corresponding marks are found live branches are recorded. If specific marks are not found, all branches stemming from that point cannot exist in the codeword and these dead branches are not investigated further. Decoding then proceeds by calculating [assuming both $H(0)$ and $H(1)$ are live] marks positions for $H(00)$, $H(10)$, $H(01)$, and $H(11)$. Again, where corresponding marks are found, live branches are retained. This process repeats with the number of decoding rounds equal to the length of the message vectors. At the end of decoding, the remaining message vectors represent the original message vectors. This is shown schematically in the decoding portion of Fig. 1. If the codeword is subject to large amounts of noise, then it is possible for spurious messages to get through the decoding tree and these are termed hallucinations. It is noteworthy that the original message vectors will always be decoded—a result of using indelible marks—but the effect of hallucinations will be to obscure which messages are genuine. To kill hallucinations, additional checksum bits with fixed values are added to the message vectors so that for the earlier example 11 $abcd$ would be encoded. This increases the number of decoding rounds but is very effective at killing off hallucinations. Burst errors, such as those arising from obscurations or blockages of the signal, appear as blocks of contiguous zeros in the codeword. When these "gaps" in the codeword become large enough to be statistically unlikely to occur by chance, the decoding round can keep alive any hash calculations that result in marks that would appear in the gap. Allowing decoding branches to connect across the gap is an effective way of correcting for burst errors where even large missing fractions of the codeword (up to 40%) can still result in perfect decoding. All genuine message vectors are decoded again with

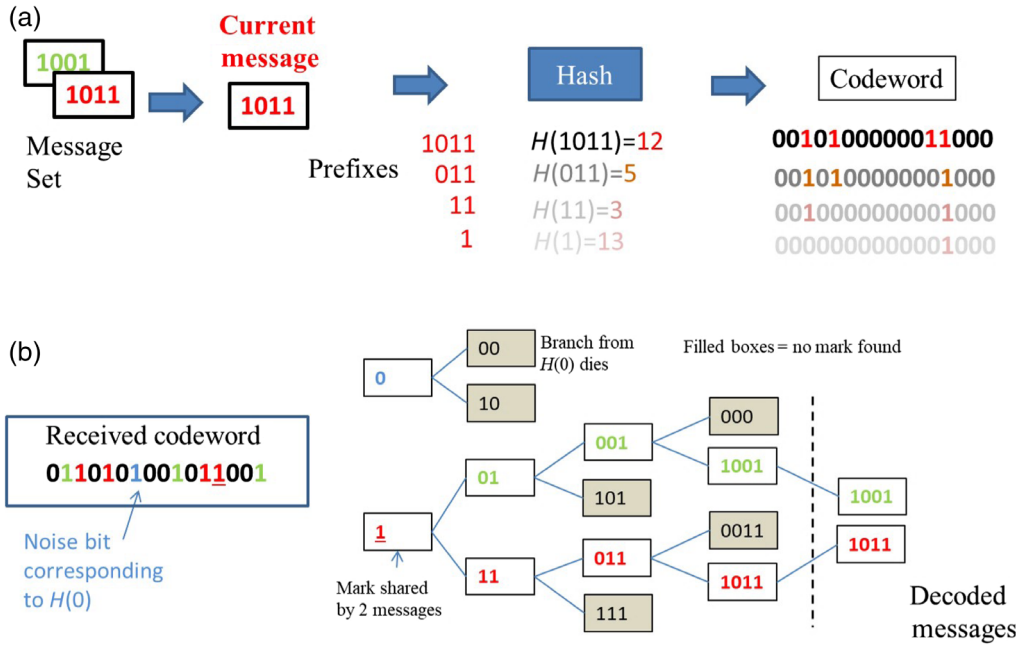


**Fig. 1** A schematic representation of (a) the encoding process and (b) the decoding tree. Prefixes of the input message are hashed to produce the addresses of marks in the codeword. The codeword is examined by comparing possible prefix hashes with the presence of marks in the codeword.

hallucinations appearing. A gap is made statistically significant by ensuring a minimum number of message vectors are encoded so that the randomizing effect of the hash function fills the codeword evenly with marks.

It is the robustness of this single encoding step that is of relevance to FSO links, where performance against burst errors is better than pure interleaving. Burst errors due to atmospheric effects or unstable alignment are a major issue in the implementation of FSO links. In this work, we discuss the topics that make the implementation of concurrent coding achievable and practical. This shows that multiple encoding streams can be overlaid into a single transmission, similar to the way multiple users can be overlaid with CDMA encoding. Also demonstrated is a property of concurrent coding that allows any single transmission to be decoded through the use of an inherent synchronization property. Finally, the first use of current coding with a visible light channel is demonstrated with multiple threads and burst errors present. This work was first presented in a conference proceeding.[26]

## 3 Modeling of Parallel Hashes

Previous work has highlighted the fact that concurrent coding produces fewer marks (1s) in its codeword than other techniques,[6] leading to a reduced transmission power. Compared to CDMA encoding, the transmitted power can be orders of magnitude less.[19] The efficiency of concurrent coding arises from common subsequences sharing marks in the codeword. This efficiency comes at the cost of being able to decode any particular message once, as multiple encoding will simply reproduce the same marks. There is also no specific order to the decoded messages, which are all decoded in parallel. This might not be an issue in situations such as a sensor network where individual sensors include a sensor ID with their data transmission. However, for more general communication, more information will be needed. In Ref. 6, multiple hash functions were used for the purpose of encoding the same information more than once or for providing decoding guidance information. In this work, the use of multiple hash functions within the same codeword is investigated and implemented.

The hash function used for concurrent coding can be any generic function that can redistribute data patterns throughout a large codeword in a suitably dispersed fashion. Various redistribution functions have been used as hash functions, including a pseudorandom bit sequence (PRBS), glow-worm hash,[16,17] and Fowler/Noll/Vo hash.[23] To ensure no collisions in the hash function, we use hash tables for a so-called closed concurrent code[19] where output addresses have been randomly and uniquely defined. Hash tables are not the ideal method for practical implementation, particularly with larger codewords—but here they meet the requirements.

Building on previous models, 8-bit messages with 2 checksum bits were encoded into a codeword space of $2^{11}$ bits. A set of 2048 element hash tables were generated and each table was assigned a thread number. A set of random messages were selected and then encoded within each thread into a single codeword—this will show that the same information can be encoded multiple times. Another method for overlaying data is to use a single hash function with a cyclic positional offset added, with a different offset representing each thread. In principle, the hash functions could be completely different, such as a mixture of PRBS, glowworm, and hash table.

The number of marks produced for a given number of messages $m$ is given by

$$Z(m) = (N + k)m - m\log_2 m + \frac{3m}{2}, \tag{1}$$

where $N$ is the number of bits in a message and $k$ is the number of checksum bits. This nonlinear increase in the number of marks is caused by the sharing of marks by multiple messages and is the source of the efficiency of concurrent coding. This is true for a single thread (i.e., data encoded with a particular hash function); however, multiple threads are independent. Because concurrent coding is an OR channel, independent threads can also share marks. For a small number of messages and a few threads, we would not expect this to occur often, but with increasing messages and threads this will become more prevalent. A simulation involving a concurrent code with multiple hash functions was created to investigate the properties and behavior. This can be seen in Fig. 2 where the actual number of marks produced by a concurrent code is

plotted for variation in the number of encoded messages and with different numbers of encoded threads with each containing the same messages. For a small number of threads, the number of marks increases linearly with threads, but as the number of threads is increased the relative increase in the number of marks is reduced. This is an indication that different threads are sharing marks in the codeword.

Because the threads are independent, any one thread sees marks from other threads as noise and this can influence the production of spurious false decodings (hallucinations) across all threads.

We can model the number of marks as follows: each thread is added serially into the codeword. Thus, each mark in the current thread has a probability of being shared with previous threads given by

$$P = \frac{M_{i-1}}{C}, \tag{2}$$

where $M_{i-1}$ is the total number of marks produced by all previous threads and $C$ is the codeword length. The number of shared marks produced is given as

$$S_i = \frac{Z_i M_{i-1}}{C}, \tag{3}$$

where $Z_i$ is the number of marks produced by the current independent thread according to Eq. (1). The total number of marks produced after the $i$'th thread is added is given as

$$M_i = M_{i-1} + Z_i - S_i. \tag{4}$$

This iterative relation was used to calculate the expected number of marks shown by the solid lines in Fig. 2 and in good agreement with the data points. Figure 3 shows the number of measured hallucinations produced in an actual concurrent code (using parameters given earlier), as the number of threads and the number of messages per thread is increased. More hallucinations are produced for the same number of marks when the number of threads is increased. Figure 4 is a bubble chart plot showing the production of hallucinations as the number of threads and the number of messages per thread is increased. This is instructive in showing us the boundaries at which multithreading challenges the encoding integrity. As a rough guide, when the total number of marks exceeds 1/4 of the codeword size, hallucinations will appear.

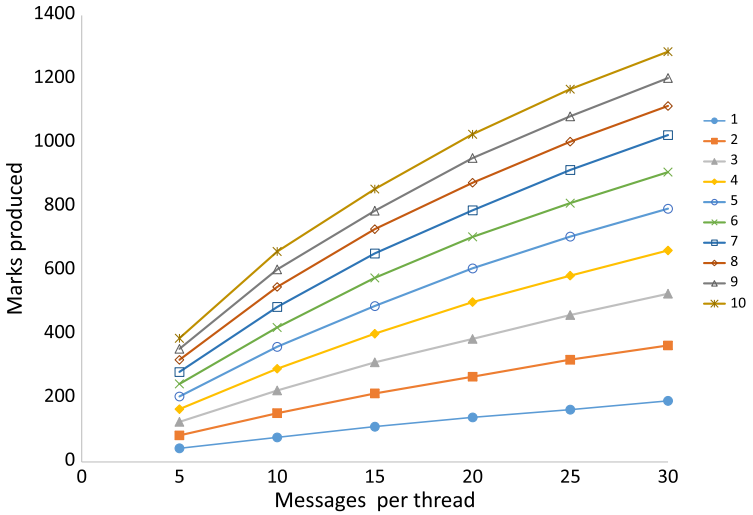


**Fig. 2** The total number of marks varying with number of encoded messages and number of threads. Symbols represent the measured number of marks produced in the code-decode process by a computer simulation running a concurrent code algorithm. Solid lines are the expected values from modeling.

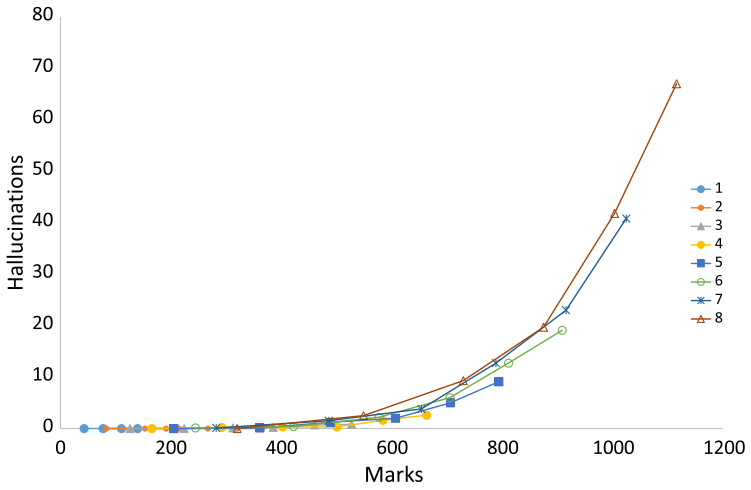


**Fig. 3** The number of hallucinations versus total marks generated for a limited number of threads, representing the data given in Fig. 2.

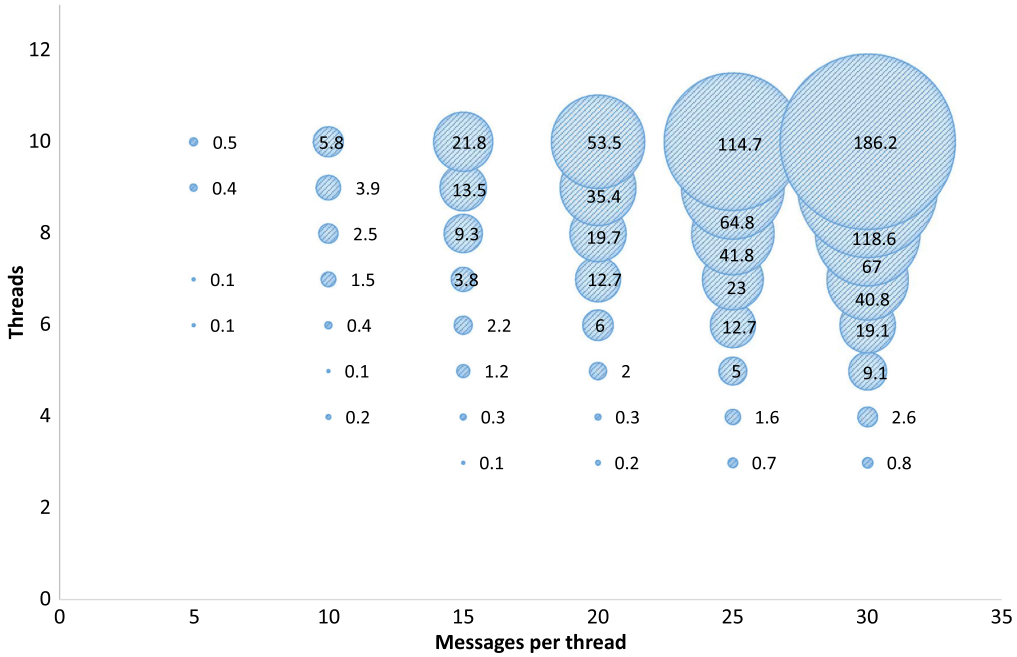


**Fig. 4** A bubble chart showing the hallucinations produced as the number of threads where the messages per thread are varied.

## 4 Implementation

To demonstrate multithreaded concurrent coding, an FSO system was used consisting of four light-emitting diodes (LEDs) and a single photodiode detector. Individual threads were generated from hash tables using a LabView program. Each individual thread was sent to an LED using a USB Ni-DAQ data-acquisition device and marks were represented by a light pulse in an OOK scheme. Each LED transmits its codeword thread in parallel. A single silicon photodiode was located 60 cm from the LEDs, which were about 1 cm apart and their natural divergence was utilized to overlap their light onto the photodiode. The overlaid LED signals (an OR process) are then indistinguishable and were amplified and sent to a second computer via another USB Ni-DAQ unit, which interpreted the pulses as digital signals rather than collecting analog voltages—this prevented the system from using relative intensity to identify the emitting source.

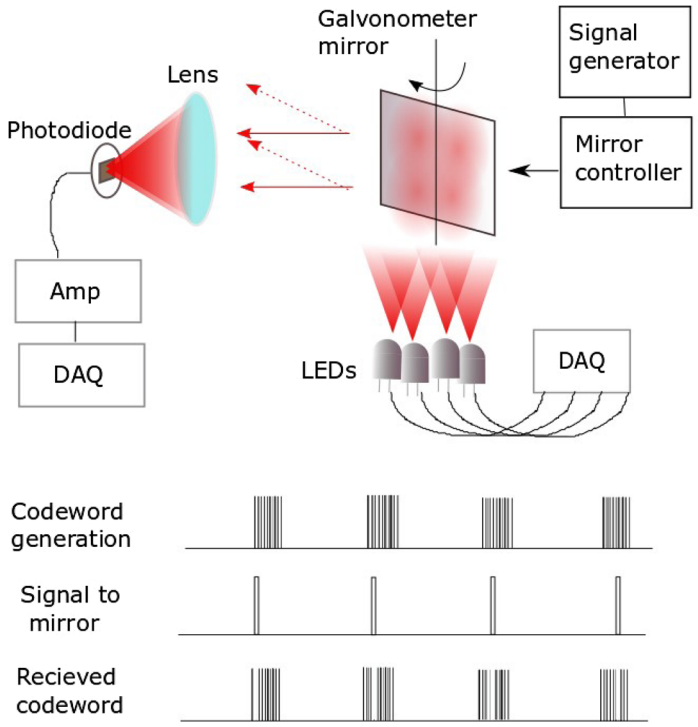


**Fig. 5** A schematic diagram of four LEDs illuminating a photodiode detector via a galvanometer mirror (top). The generation of gaps in the received codeword is represented in the lower diagram.

The requirement for each thread is that it uses a different hash function. This can be achieved in a number of ways such as using a different mathematical function in each case, or the same function with a different seed. In this work, a set of hash tables common to both receiver and transmitter were used to ensure minimal interactions between threads.

Initial developments on synchronization methods were performed with a single thread and LED. Multiple threads can be combined and transmitted by a single LED. Using multiple LEDs aligns with a multiuser system similar to that for CDMA techniques. To emulate the effects of burst errors that could be introduced by obstructions or beam wander, a galvanometer mirror was used to occasionally deflect the light away from the detector. This system is shown schematically in Fig. 5.

## 4.1 *Synchronization*

Hashed messages share marks that represent the first round of prefix encoding, thus the marks representing these prefixes will be occupied with high probability. The first two layers of prefixes—$H(0)$, $H(1)$, and $H(00)$, $H(01)$, $H(10)$, $H(11)$—are occupied with high probability when a few messages are encoded and these marks—referred to as the principle marks—can be used to synchronize the received codeword.[19]

The codewords were sent in isolation (that is, with no pre- or postinformation for synchronization) with the intention that only the codeword itself can be used for decoding. This means that no prior setup to establish a phase-locked loop was used. Before decoding starts, the boundaries of the codeword need to be established and the mark positions were set. Because zeros are represented by the absence of signal, the exact start of the codeword is not clear. This is established by setting up a synchronization vector, which is a codeword block containing only the principle marks (those corresponding to the two least significant bits in the decoding tree) from the first thread. This synchronization vector is correlated with the received vector and the maximum correlation value taken as the reference point within the codeword from where the first mark position can then be determined. Note that this method becomes problematic for the use of a single hash function with multiple cyclic offsets as it produces multiple genuine correlation peaks and requires an additional step of identifying which peak corresponds to which thread.

The transmitter and receiver were set to nominally the same sample frequency—typically 20 kHz. However, small differences of a few Hz lead to disparity in mark positions between the receiver and transmitter. Across the length of the codeword, a small drift causes marks to be spread across 2 mark positions or shifted by an entire mark. In this case, misplaced marks lead to a failure to correctly decode the full contents of transmission. This was understood and pointed out by Bahn[24,25] who quantified the precision with which oscillators should be matched and suggested the use of "bookend marks" to define the start and end of the codeword. As has been stated for the ideal modeling case, the nature of indelible marks means that encoded messages cannot be removed and will always be decoded. However, correct synchronization is the essential property required to make this true.

The steps to correct mark drift proceed as follows: A window around the position of each of the principle marks was established, typically 3 bins wide. Marks within the received codeword that fall within these windows are identified as being the principle marks and their positions recorded. The received positions are plotted against the expected positions to generate a linear relationship. Using the gradient and offset of this relationship, all marks in the codeword can then be adjusted to ensure there is a mark in the correct position within the codeword. Decoding can then proceed.

The transmitter would encode a fixed number of randomly selected messages. These messages were repeated in each thread to demonstrate how the same information can be encoded more than once. We can evaluate the decoding process by recording the number of decoded messages in each thread. The inherent synchronization approach was observed to work and allow the four encoded threads to be successfully decoded. However, it would suffer from a weakness arising from a reliance on randomly filled principle marks. A threshold level for correlation was set, typically corresponding to matching five of the possible six principle marks. Occasionally, this threshold is not passed and results in no decoding at all. To overcome this issue, two amendments were investigated. The first involves adding to the beginning of the codeword a 16-bit code that can be identified by correlation. The second involves distributing throughout the codeword a small number of static synchronization marks (similar to the bookend marks[25] but not at the ends of the codeword). The first approach is simple and adds to the codeword length a little. However, this represents an easily identifiable weakness to a would-be jammer and a small corruption of this sequence would result in no decoding results (as would be true for bookend marks). This is equally true for burst errors that occur over the sequence. The second approach should be more robust against both jamming and burst errors due to distributing marks throughout the codeword. Both approaches were investigated in relation to their robustness against burst errors by artificially introducing gaps of missing data into the codeword prior to transmission. The gaps were added to a fixed point centrally in the codeword (so as not to completely corrupt the decoding). The number of decoded messages for 300 transmitted and decoded streams, each containing 10 messages from a single thread with increasing gap sizes, was determined.

The plot in Fig. 6 shows the average number of decoded messages against gap size for codewords involving a sequence, one static synchronization point, and six static points. It is clear from these data that hallucinations are being produced more significantly than in modeling. This is most likely an effect caused by the synchronization, especially the drift correction, which would add marks into the codeword and appear as noise. As was discussed in Ref. 6, concurrent codes have resilience to a combined level of burst error and noise. Further development of efficient and effective synchronization schemes is required. It should be borne in mind that even though hallucinations are being produced, the original genuine messages are still being decoded with 30% of the codeword missing, which could not be done with interleaving.

In real world optical applications burst errors in the form of signal dropouts are unpredictable and can arise due to conditions such as misalignment or atmospheric scintillation. To better emulate this, a galvanometer-mounted mirror was used to steer the output light toward the detector. A square voltage pulse was applied to the mirror to cause it to twitch and briefly steer the light away from the detector. The pulse duration was chosen to ensure that the codeword gap was around 10% and the pulse was repeated at a rate such that the gap appeared in different places during the production of codewords. Examples of the decoding performance are shown in Fig. 7 where a single thread containing 10 random messages was transmitted and the process was repeated 300 times. Synchronization was examined using four methods; inherent

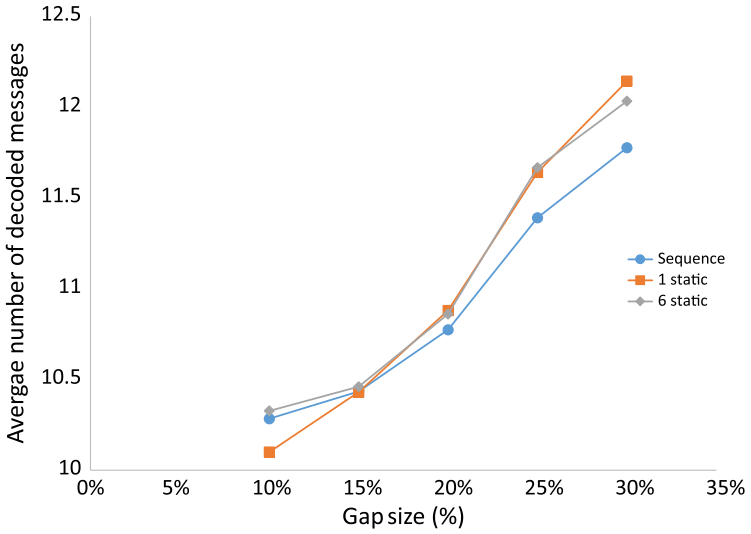


**Fig. 6** The number of decoded messages for different synchronization methods: an initial synch sequence, one additional static point (no sequence), and six additional static points.

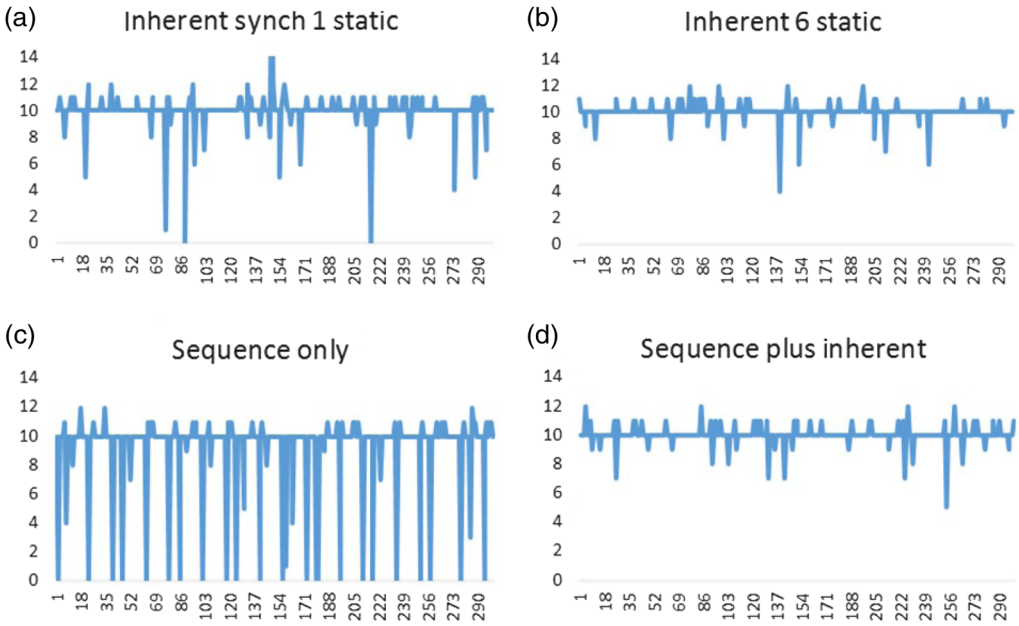


**Fig. 7** Plots showing the number of decoded messages for various methods of synchronization as a galvonometer mirror introduces burst errors. Each plot is the outcome of 300 decodings using a single thread. (a) Inherent synchronization with one static mark included. (b) Six static marks. (c) Only a 16-bit correlation sequence for synchronization. (d) Combines a sequence with inherent synchronization.

synchronization with one additional static mark, inherent synchronization with six additional static marks, synchronization from an appended sequence, and a combination of a sequence and inherent synchronization, which applies the inherent synchronization if no sequence correlation can be detected. Clearly, from this we can see that the sequence synchronization regularly fails to decode where the twitching mirror removes part of the sequence. The use of the inherent synchronization is more robust, and this situation is enhanced by adding additional static marks. The combination of sequence and inherent with static appears the most robust. All examples show that the system loses some messages on occasion, arising from a combination of the gap removing several principle marks and a small number of inherent marks. Clearly it would be

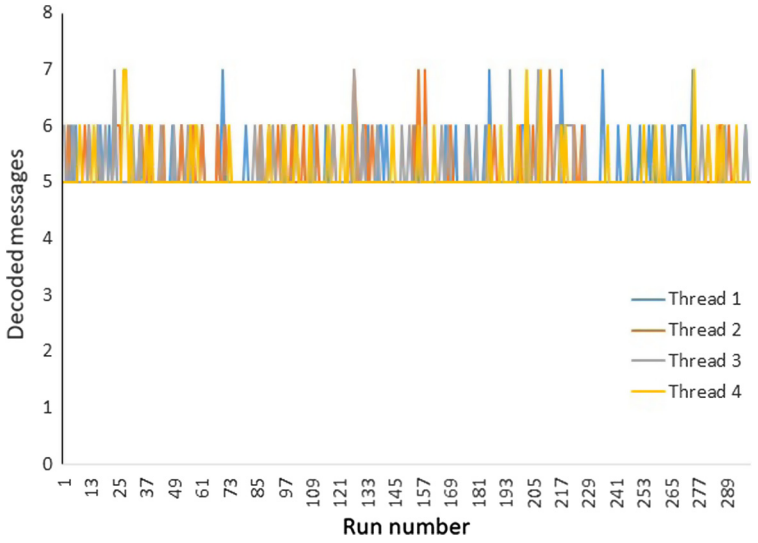


**Fig. 8** The number of decoded messages from 300 hundred decodings with four simultaneous threads per codeword. Five messages per thread were transmitted.

possible to add significantly more static marks to improve the robustness further. Nevertheless, this test clearly shows that the use of inherent synchronization significantly improves the robustness.

While the decoding performance is less than ideal, this is the first example of concurrent coding being used with an optical signal and the first example of inherent synchronization ever being implemented. Nonetheless, there is still room for improvement.

### 4.2 *Multiple Threads*

Having now established a method of synchronization, we can demonstrate the use of multiple threads. Codewords generated from different hash functions were sent to four spatially separated LEDs. With each LED transmitting a separate signal that is overlaid and indistinguishable from the other LEDs, this system behaves as if there are individual users or channels communicating simultaneously with a single receiver. In this case, all the threads and streams are synchronized to produce a single codeword, but this need not be the case in general. Decoding proceeds by correlating the principle marks (including multiple static marks) for the first hash/thread to establish synchronization and mark position correction. Then multiple stages of decoding are performed to decode each thread. In this case, five random messages per thread were encoded into four threads and the decoding results are seen in Fig. 8.

For each thread, the average number of decoded messages was 5.16, meaning an average of 0.16 hallucinations per decoding. With no burst errors introduced, there is no loss of data. This represents the first demonstration of the use of concurrent coding over an optical channel and the first demonstration of multiple overlaid threads or users with concurrent coding.

## 5 Conclusions and Discussion

Concurrent coding is an innovative approach to encoding information that can provide benefits of robustness of data recovery and simplicity of implementation. The use of indelible marks with concurrent coding aligns very well with the OOK modulation which is easily used with FSO techniques. Concurrent coding offers an alternative method of encoding to protect against both random noise and burst errors. Burst errors are a particular issue for free space communications, and concurrent codes have the potential to perform better than interleaving in the recovery of information when burst errors are present. Thus, concurrent coding is a potentially interesting tool to employ, particularly as the use of FSO communications is gaining significant interest. In this work, the characteristics of concurrent coding that will prove useful in future

implementations of FSO communications have been investigated. In particular, it has been demonstrated that concurrent coding possesses an inherent synchronization structure that allows codeword transmissions to be sent in isolation and reliably decoded with no inclusions or preamble transmissions. Inherent synchronization will help with security concerns (in this and other implementations, such as rf) because the first detected mark is not the start of the codeword transmission, and therefore, an eavesdropper must understand the hashing function being used to locate the start of the codeword.

This allows to overcome the burst errors present in the single codeword. In addition, it is shown that, by encoding information using different hashing functions, multiple overlaid codeword transmissions can be successfully and independently decoded. This can be considered as a multiuser access channel or a multilayer transmission channel. It has been shown that this approach works with an optical channel, where four LED sources transmit information encoded using different hashing functions and are all overlaid onto a single detector. All four channels were successfully decoded and this represents the first demonstration of multilayer concurrent code communication and the first use of concurrent coding over an optical channel.

It is the robustness of concurrent coding that is of interest to FSO communications, in particular where communication is required from mobile or unstable platforms. Mobile applications for FSO inevitably have unpredictable circumstances that require a level of flexibility and robustness of the system. Maintaining a constant link connection between the source and receiver can be a hardware problem tackled by accurate beam pointing systems, but it cannot overcome beam interruptions such as a moving object blocking the beam. This is where the encoding protocol helps. Future applications could see mobile sensors or systems needing to send a burst of data and know that the data will be correctly decoded. This could be in the form of compact FSO systems to perform financial transactions or authorization. Concurrent coding is a tool that will enable this without the need for presynchronization to delay transmission. In addition, the efficiency of concurrent coding can reduce the transmitted energy requirements, which is always a benefit to mobile systems and has potential defense benefits by having a low probability of intercept.

Concurrent coding is still at an early stage of development and much more needs to be done, in particular around synchronization and its effects on decoding quality. But concurrent coding offers an alternative way of thinking about encoding for robustness with benefits worth exploring in future applications.

## Acknowledgments

The author would like to thank Paul Brittan for his helpful support and discussions during this work.

## References

1. L. C. Baird, III, W. L. Bahn, and M. D. Collins, "Jam-resistant communication without shared secrets through the use of concurrent codes," Technical Report, USAFA-TR-2007-01, U. S. Air Force Academy (2007).
2. L. C. Baird, III et al., "Keyless jam resistance," in *Proc. 8th Annu. IEEE SMC Inf. Assur. Workshop*, Orlando, Florida, pp. 143–150 (2007).
3. W. L. Bahn, L. C. Baird, III, and M. D. Collins, "Jam resistant communications without shared secrets," in *Proc. 3rd Int. Conf. Inf. Warfare and Secur.*, Omaha, Nebraska (2008).
4. L. C. Baird, III, M. Carlisle, and W. L. Bahn, "Unkeyed jam resistance 300 times faster: the inchworm hash," in *MILCOM 2010 – Mil. Commun. Conf.*, San Jose, California (2010).
5. L. C. Baird, III et al., "A novel visual cryptography coding system for jam resistant communications," *J. Issues Inf. Sci. Inf. Technol.* **7**, 495–507 (2010).
6. D. M. Benton, "Concurrent codes: a holographic-type encoding robust against noise and loss," *PLoS One* **11**(3), e0150280 (2016)
7. M. Bossert, *Channel Coding for Telecommunications*, 1st ed., John Wiley & Sons, Inc., New York (1999).

8. I. Glover and P. M. Grant *Digital Communications*, Pearson Education, Harlow, England (2010).
9. B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed., Prentice Hall, New Jersey (2001).
10. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes. 1," in *IEEE Int. Conf. Commun., ICC 93*, IEEE, Vol. 2, pp. 1064–1070 (1993).
11. A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory* **13**(2), 260–269 (1967).
12. S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, New Jersey (1983).
13. G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, Springer Science & Business Media, New York (2013).
14. P. Fire, "A class of multiple-error-correcting binary codes for non-independent errors," Sylvania Rept. RSL-E-2, Sylvania Reconnaissance Systems Laboratory, New York (1959).
15. http://www.usna.edu/Users/math/wdj/files/documents/reed-sol.htm (accessed February 2016).
16. L. C. Baird et al., "The glowworm hash: increased speed and security for BBC unkeyed jam resistance," in *Mil. Commun. Conf., 2012-Milcom*, IEEE, pp. 1–6 (2012).
17. L. C. Baird and B. Parks, "Exhaustive attack analysis of BBC with glowworm for unkeyed jam resistance," in *Proc. IEEE Mil. Commun. Conf., MILCOM*, pp. 300–305 (2015).
18. H. Hanifi, L. Baird, and R. Thurimella, "A new algorithm for unkeyed jam resistance," in *Proc. 8th Int. Conf. Secur. of Inf. and Networks*, ACM, New York, pp. 210–216 (2015).
19. D. M. Benton, "Concurrent coding: a reason to think differently about encoding against noise, burst errors and jamming," arXiv:1901.09646 (2019).
20. M. A. Khalighi and M. Uysal, "Survey on free space optical communication: a communication theory perspective," *IEEE Commun. Surv. Tutor.* **16**(4), 2231–2258. (2014).
21. H. Henniger and O. Wilfert, "An introduction to free-space optical communications," *Radio Eng.* **19**(2), 203–212. (2010).
22. Z. Ghassemlooy and W. O. Popoola, "Terrestrial free-space optical communications," in *Mobile and Wireless Communications Network Layer and Circuit Level Design*, S. A. Fares and F. Adachi, Eds., InTech, https://www.intechopen.com/books/mobile-and-wireless-communications-network-layer-and-circuit-level-design/terrestrial-free-space-optical-communications (accessed 1 January 2010).
23. https://tools.ietf.org/html/draft-eastlake-fnv-03 (accessed February 2020).
24. W. Bahn, "Concurrent code spread spectrum: theory and performance analysis of Jam resistant communication without shared secrets," PhD Thesis, University of Colorado (2012).
25. W. L. Bahn, L. C. Baird, III, and M. D. Collins, "Oscillator mismatch and jitter compensation in concurrent codes," in *Proc. 2008 IEEE Mil. Commun. Conf.,* (2008).
26. D. M. Benton and P. St. John Brittan, "A new encoding scheme for visible light communications with applications to mobile connections," *Proc. SPIE* **10437**, 104370C (2017).

**David M. Benton** graduated in physics from the University of Birmingham in 1989. He completed his PhD in laser spectroscopy for nuclear physics in 1994 and then conducted postdoctoral research in positron emission tomography and laser spectroscopy for nuclear physics, all at the University of Birmingham. In 1998, he joined Defence Evaluation and Research Agency, which became QinetiQ, where he worked on a variety of optical projects. He was the leader of a group building quantum cryptography systems and was involved in a notable 140-km demonstration in the Canary Islands. He became chief scientist for L-3 TRL in 2010, working on photonic processing techniques for rf applications. He is now at Aston University with a variety of interests, including innovative encoding techniques, gas sensing, and laser detection techniques.