

卒業論文

LEDマトリクスパネルによる異なる点 滅周波数を用いた可視光通信の実験

An experiment on visible light communication using different blinking frequencies with an LED matrix panel.

method

園邊 翔大

立命館大学理工学部電子情報工学科

2025年12月

内 容 梗 概

目 次

内容梗概	i
第1章 序論	1
1.1 研究背景	1
1.2 本研究の目的	2
1.3 本論文の構成	3
第2章 LED マトリクスパネルを用いた画像表示システムの構築	5
2.1 使用デバイスと開発環境	5
2.1.1 HUB75 64 × 32 LED マトリクスの仕様	5
2.1.2 Raspberry Pi 4 と開発環境	6
2.2 画像アップロードシステムとネットワーク構成	8
2.2.1 Web アプリケーション構成 (Django)	8
2.2.2 Cloudflare Tunnel による外部アクセス	8
2.2.3 Web システムの構成	9
2.2.4 Web サイトからの画像アップロード～パネル表示までの処理フロー	9
2.3 256 階調表示システム	10
2.3.1 RGB 各色の点灯回数制御による 256 階調表現	10
2.3.2 点灯させるタイミングの計算方法	11
2.3.3 32 × 64 画像へのリサイズとマッピング処理	13
2.4 ガンマ補正による画質改善	14
2.4.1 ガンマ補正	15
第3章 2つの異なる点滅周波数を用いた可視光通信方式	19
3.1 提案方式の概要	19
3.2 可視光通信のロジック	19
3.2.1 点灯パターンの設計	19
3.2.2 バイナリ変換の方法	20
3.2.3 分割領域における点灯パターンの設計	21
3.2.4 4分割以上の検証	22
3.3 フレーム構成とシンボル境界推定	23
3.3.1 送信フレームの構成	23

3.3.2 FLAG 検出とシンボル境界推定	24
3.4 受信アルゴリズム	24
3.4.1 フレーム列からの時系列データ生成	24
3.4.2 bit パターンとの類似度評価と位相推定	25
3.4.3 復号フロー全体	25
第 4 章 評価および考察	27
4.1 人間の視覚における見え方の評価	27
4.1.1 旧方式と提案方式の色味の比較	27
4.1.2 周波数ごとの色差評価	27
4.2 考察	28
4.3 結論および今後の課題	28
参考文献	28
謝辞	31
発表論文リスト	33

図 目 次

2.1	HUB75 64 × 32 LED マトリクスの外観例	6
2.2	Cloudflare Tunnel の概要図 ¹	8
2.3	Web サイトの画面	9
2.4	システム全体の構成	10
2.5	RGB 各色の点灯回数制御による 256 階調表現	11
2.6	RGB 各色の点灯回数制御による 256 階調表現	12
2.7	元画像	14
2.8	リサイズ後画像	14
2.9	ガンマ補正の原理 ²	15
2.10	ガンマ 2.2 補正画像	16
2.11	ガンマ 2.2 補正画像（パネル表示）	16
3.1	提案方式におけるフリッカパターン	20
3.2	4 分割 LED マトリクスの走査制御	21
3.3	LED マトリクスパネルを 4 つの領域に分割した画像	22
3.4	LED マトリクスパネルを 8 つの領域に分割した画像	23
3.5	送信データの全体設計	23

表 目 次

2.1 Raspberry Pi 4 Model B の主要仕様	7
--	---

第1章 序論

1.1 研究背景

近年、IoT (Internet of Things) の普及やスマートシティの実現に向けた取り組みが活発化する中で、データ通信技術への需要はますます高まっている。特に、位置情報に基づいた情報配信や、公共空間における情報提供システムの重要性が増している。従来の無線通信技術（Wi-Fi、Bluetooth、携帯電話網など）は広く普及しているものの、混雑環境での通信品質の低下や、電磁波干渉の問題、さらには通信インフラの設置コストといった課題が存在する。

可視光通信（Visible Light Communication: VLC）は、人の目に見える可視光線帯域の電磁波を用いて無線通信を行う技術である。¹ LEDが発する光が届く範囲で通信が可能となるため、一般に使われているLED照明を通信手段として流用できる点が特徴である。また、可視光は生体への影響が小さく安全であり、電磁波による他機器への悪影響も少ないとされる。

一方、街中には大型のLEDマトリクスピネルを用いた広告看板や情報表示装置が数多く設置されている。これらのパネルは、高解像度で鮮明な画像や動画を表示できるだけでなく、高速な点滅制御が可能であることから、可視光通信の送信端末としての利用が期待されている。特に、スマートフォンのカメラ機能を利用して受信を行う方式は、ユーザーが専用の受信装置を用意する必要がなく、既存のデバイスで情報を受信できる点で実用性が高い。例えば、街中の大型広告パネルにスマートフォンのカメラを向けるだけで、その場所に関連する情報（店舗情報、イベント情報、クーポンなど）を自動的に取得できる可能性がある。

しかし、可視光通信を実用化する上では、いくつかの技術的課題がある。第一に、人間の視覚に違和感を与えない範囲で情報を埋め込む必要がある。高速な点滅は通信速度の向上に寄与するが、ちらつきとして知覚されると、表示装置としての機能を損なう可能性がある。第二に、スマートフォンカメラのフレームレートは通常30~60fps程度であり、それよりも高い周波数で点滅させる場合、受信側での工夫が必要となってくる。第三に、通信速度を向上させるためには、単一の領域で送信するよりも複数の領域を並列に利用する必要があるが、その際に人間の視覚に違和感を与えない範囲で情報を埋め込む必要がある。

本研究では、これらの課題を解決するため、LEDマトリクスピネルを用いた可

¹ 株式会社大塚商会、「LED通信（可視光通信）とは」<https://www.otsuka-shokai.co.jp/products/led/knowledge/vl> (最終閲覧日: 2026-01-18)

視光通信システムの実現を最終目標として、段階的なアプローチで研究を進めた。まず、LED マトリクスパネルの表示ロジックを理解するため、RGB 各色の点灯回数制御による 256 階調表示システムの実装から着手した。この実装を通じて、HUB75 インタフェースの制御方法やフレーム更新の仕組みを理解し、高品質な画像表示を実現するための技術を習得した。

次に、実用化に向けた機能として、Django フレームワークを用いた Web アプリケーションを構築し、Cloudflare Tunnel を利用して遠隔地から画像をアップロードして LED マトリクスパネルに表示するシステムを実現した。この実装により、実際の運用環境を想定したシステム構成を確立し、可視光通信システムの基盤となるインフラストラクチャを整備した。

さらに、表示品質の向上を目指し、ガンマ補正の理論を理解し実装した。通常のディスプレイは自動的にガンマ 2.2 変換を行うため、画像ファイルは予めガンマ 0.45 変換された状態で保存されている。しかし、LED マトリクスパネルではこの変換が行われないため、元の画像よりも淡く表示される問題が発生していた。ガンマ補正を実装することで、人間の視覚特性に合わせた適切な輝度表現を実現し、高品質な画像表示を可能にした。

これらの基礎的な実装を通じて、LED マトリクスパネルの制御技術を習得した後、最終的な目標である可視光通信システムの実現に取り組んだ。具体的には、1 枚の LED マトリクスパネルを 4 つの領域に分割し、各領域で同一データを並列に送信する方式を採用した。点灯パターンは 4 スロットで構成し、1010 と 1100 のバランス符号により 0/1 を表現する。両パターンともデューティ比が 50% となるため、4 スロット平均の輝度が一致し、人間の目には色味差やちらつきが生じにくい。一方で、受信側は平均輝度ではなくローリングシャッター由来の縞パターンを用いて点滅を検出する必要がある。この構成により、単一領域で送信する場合と比較して 4 倍の並列送信が可能となる。

本技術の実用化により、将来的には街中の大型広告パネルや情報表示装置が可視光通信の送信端末として機能し、通行人がスマートフォンのカメラを向けるだけで位置情報に基づいた情報を自動的に取得できる社会の実現が期待される。

1.2 本研究の目的

本研究の目的は、LED マトリクスパネルを用いた可視光通信の送信部システムを確立することである。本研究は、段階的なアプローチにより、基礎的な表示技術の習得から始まり、最終的な可視光通信システムの送信部の実現に向けて実用化を目指した。具体的には、以下の 4 つの段階的な目標を設定した。

第一に、RGB 各色の点灯回数制御により 256 階調で表示するシステムを構築する。これにより、ユーザーが指定した任意の画像を LED マトリクスパネルに表示することができ、実用的な画像表示システムの基盤を確立する。

第二に、Cloudflare Tunnel と Django という Web アプリケーションフレームワークを使用して、遠隔地から画像を指定し、その指定された画像を表示可能なサイズや形式にリサイズして表示するシステムを確立する。これにより、実際の運用環境を想定したシステム構成を実現し、可視光通信システムの基盤となるインフラストラクチャを整備する。

第三に、人間の視覚に違和感を与えない明度に調整するためにガンマ補正の理論を理解し、独自の 256 階調表示システムを改良して補正を適用する。これにより、高品質な画像表示を可能にし、可視光通信システムにおいても視覚的に違和感のない表示を実現する基盤を構築する。

第四に、LED マトリクスパネルを 4 つの領域に分割し、それぞれの領域で異なる点滅周波数を用いて同一のデータを並列に送信する方式を採用する。これにより、単一領域で送信する方式と比較して 4 倍のデータ送信速度を実現することを目指す。

1.3 本論文の構成

本論文は以下のように構成されている。

第 2 章では、LED マトリクスパネルを用いた表示システムの構築について述べる。まず、使用した HUB75 64 × 32 LED マトリクスのハードウェア仕様と、Raspberry Pi を中心とした開発環境について説明する。次に、RGB 各色の点灯回数制御による 256 階調表示システムの実装方法を述べ、画像のリサイズとマッピング処理、Web サイトからの画像アップロードからパネル表示までの処理フローを説明する。最後に、構築した表示システムの品質評価を行い、ガンマ補正などの画質改善手法について議論する。

第 3 章では、LED マトリクスパネルによる異なる点滅周波数を用いた可視光通信の実験について述べる。まず、実験の概要を説明し、送信側のハードウェア・ソフトウェア構成を述べる。次に、人間の視覚における色味差を抑えるためのバランス符号について説明し、シンボル設計と同期語の設計方針を述べる。さらに、スマートフォンカメラで撮影されたフレーム列から bit 列を復号する受信アルゴリズムについて説明する。最後に、実装コードと最適化の工夫について述べる。

第 4 章では、提案方式の評価と考察を行う。表示品質と通信性能の定量的評価を行い、実用化に向けた課題と今後の展望について議論する。

第 5 章では、本研究のまとめと今後の課題を述べる。

第2章 LEDマトリクスパネルを用いた画像表示システムの構築

本章では、LEDマトリクスパネルの基本的な仕様を理解した上で、独自の画像表示システムを構築する。256階調での画像表示システムの構築を始めとし、遠隔地から画像をアップロードして表示を行えるシステムや、ガンマ補正を適用した画像表示の品質向上を目指す。

2.1 使用デバイスと開発環境

本節では、使用した HUB75 64 × 32 LED マトリクスのハードウェア仕様と、Raspberry Pi を中心とした開発環境についてまとめる。

2.1.1 HUB75 64 × 32 LED マトリクスの仕様

本研究では、HUB75 インタフェースを採用した 64 × 32 ピクセルの LED マトリクスパネルを使用した。HUB75 は、RGB LED マトリクスパネルを制御するための標準的なインターフェース規格である。本システムでは、64 列 × 32 行の合計 2,048 個の RGB LED を制御し、各 LED は赤 (R)、緑 (G)、青 (B) の 3 色を独立に制御できる。

HUB75 インタフェースは、データ信号、クロック信号、ラッチ信号、出力イネーブル (OE) 信号、行アドレス信号などから構成され、高速な走査制御を実現する。本システムでは、この HUB75 インタフェースを用いて 120Hz 相当のフレーム更新を実現し、可視光通信の送信端末として機能させる。

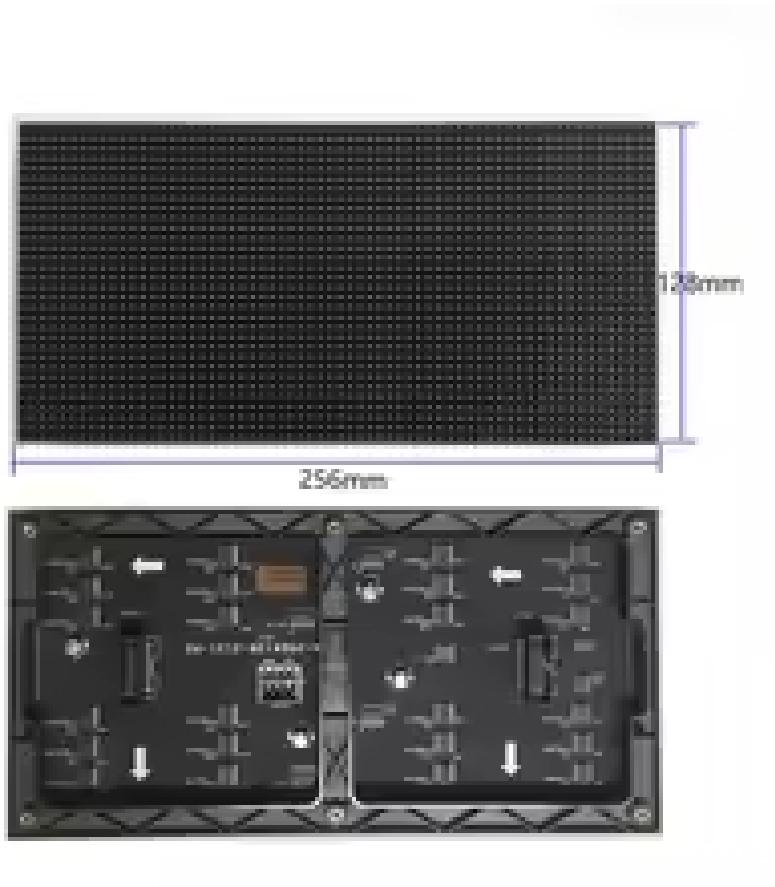


図 2.1: HUB75 64 × 32 LED マトリクスの外観例

2.1.2 Raspberry Pi 4 と開発環境

本研究では、Raspberry Pi 4 Model B を制御用コンピュータとして使用した。Raspberry Pi 4 は、ARM アーキテクチャを採用したシングルボードコンピュータであり、GPIO (General Purpose Input/Output) ピンを用いて HUB75 インタフェースの LED マトリクスパネルを直接制御することが可能である。本システムでは、Raspberry Pi 4 上で C 言語による LED パネル制御プログラムを実行し、高速な点滅制御を実現している。また、Python 環境を併用して、Web サーバの動作も Raspberry Pi 4 上で実現している。

Raspberry Pi 4 Model B の主要仕様を表 2.1¹に示す。

表 2.1: Raspberry Pi 4 Model B の主要仕様

基本仕様	
販売元	element14
製品型番	SC0195/0765756931199
リビジョン	1
SoC	Broadcom BCM2711
CPU	1.5GHz クアッドコア Cortex-A72 (ARMv8、64bit、L1=データ用 32KB 命令用 48KB/Core、L2=1MB)
GPU	デュアルコア VideoCore VI® 500MHz、OpenGL ES 3.0 対応、ハードウェア OpenVG 対応、H.265 (HEVC) 4Kp60 デコード、H.264 1080p60 デコード / 1080p30 エンコード
メモリー	8GB LPDDR4-3200 SDRAM
電源	USB type C ソケット 5V 3.0A / 2.54mm ピンヘッダー / PoE (要オプション PoE HAT)
消費電力 (本製品単体)	アイドル：約 3W、ストレス：約 6.25W
サイズ	85 × 56 × 18mm
生産国	英国
インターフェース	
イーサネット	10/100/1000 Base-T RJ45 ソケット (BCM54213PE)
無線 LAN (WiFi)	IEEE 802.11 b/g/n/ac 2.4/5GHz デュアルバンド (Cypress CYW43455)
Bluetooth	Bluetooth 5.0, Bluetooth Low Energy (Cypress CYW43455)
ビデオ出力	micro HDMI × 2、コンポジット 3.5mm 4 極ジャック (PAL、NTSC)、DSI 2-lane (15pin 1mm ピッチ)
オーディオ出力	3.5mm 4 極ジャック、micro HDMI (ビデオ出力と共有) × 2、I2S ピンヘッダー
カメラ入力	2-lane MIPI CSI (15pin 1mm ピッチ)
USB	USB 2.0 × 2、USB 3.0 × 2 (VIA VL805 PCIe)
GPIO コネクター	40 ピン 2.54mm ピンヘッダー (GPIO × 26 3.3V 16mA、UART、I2C、SPI、I2S、PWM、5V 出力 (使用電源に依存)、3.3V 出力 50mA (GPIO 信号との総和))
メモリー カード スロット	micro SD メモリーカード (SDIO)

⁰<https://raspberry-pi.ksyic.com/?pdp.id=552> より引用

2.2 画像アップロードシステムとネットワーク構成

本節では、遠隔地から画像をアップロードして LED マトリクスパネルに表示するための Web アプリケーションとネットワーク構成について説明する。

2.2.1 Web アプリケーション構成 (Django)

画像のアップロードと表示制御のための Web アプリケーションとして、Django フレームワークを用いた Web サーバを構築した。Django は、Python で記述された Web フレームワークであり、画像アップロード機能、ファイル管理、API エンドポイントの実装を容易に実現できる。本システムでは、Django を用いて画像アップロード用の Web インタフェースを提供し、アップロードされた画像を Raspberry Pi 4 上で処理して LED マトリクスパネルに表示する。

2.2.2 Cloudflare Tunnel による外部アクセス

本システムでは、Cloudflare Tunnel を用いて Raspberry Pi 4 上で動作する Django Web アプリケーションに外部からアクセスできるようにした。

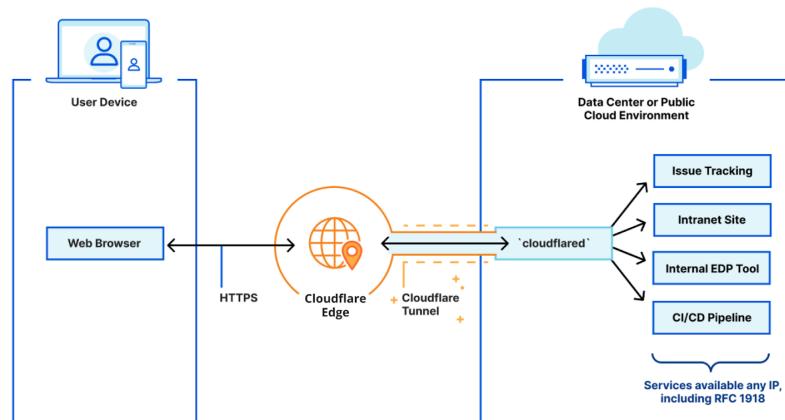


図 2.2: Cloudflare Tunnel の概要図¹

Cloudflare Tunnel は、Cloudflare が提供するトンネリングサービスであり、パブリック IP アドレスやポート開放なしに、ローカルネットワーク内のサービスをインターネット経由でアクセス可能にする。これにより、Raspberry Pi 4 が設置されているローカルネットワークの設定を変更することなく、外部の PC やスマート

¹<https://blog.cloudflare.com/getting-cloudflare-tunnels-to-connect-to-the-cloudflare-network-with-quic> より引用

トフォンからブラウザ経由で画像をアップロードし、LED マトリクスパネルに表示することが可能となった。

2.2.3 Web システムの構成

今回は、独自で Web サイトを構築した。なお、Web サイトのデプロイには Vercel というサービスを使用した。

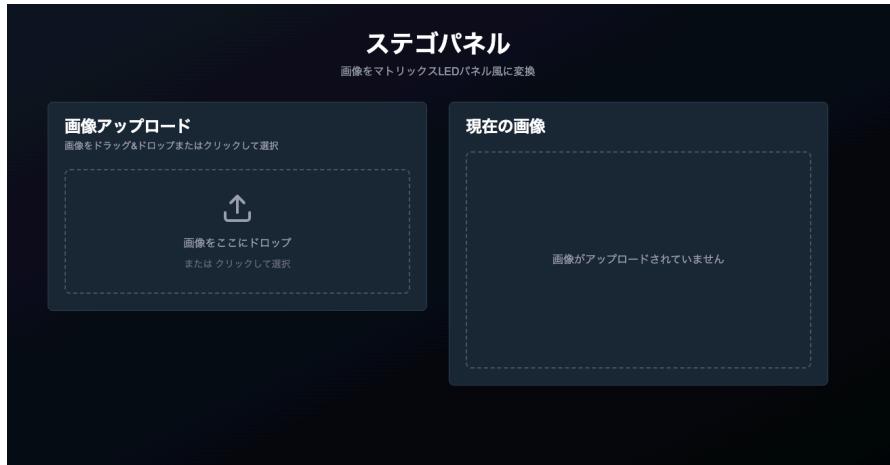


図 2.3: Web サイトの画面

このサイトでは、画像をアップロードして、パネルに表示することができる。サイト内で画像ファイルをアップロードするとその画像は Base64 形式にエンコードされ、Raspberry Pi で構築している Django Web アプリケーションに送信される。そして、Django Web アプリケーションでは、常に RestAPI でのリクエストを受け付けて、Web サイトから Base64 形式の画像を受け取り、デコードした後に、C 言語で作成したプログラムの実行を呼び出すようにした。C 言語で作成したプログラムでは、リサイズした画像をパネルに表示するようにした。これにより、Web サイトから画像をアップロードして、パネルに表示することができるようになった。

2.2.4 Web サイトからの画像アップロード～パネル表示までの処理フロー

本システムでは、Web サイトから画像をアップロードして、パネルに表示するシステムを構築した。これにより、遠隔地から画像をアップロードしてパネルに表示することが可能となった。本システムの全体構成を以下に示す。

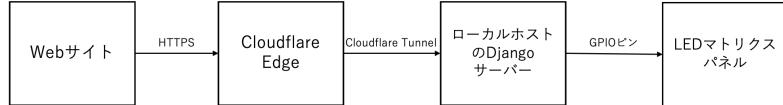


図 2.4: システム全体の構成

ユーザーは、PC やスマートフォンの Web ブラウザから Cloudflare Tunnel 経由で Django Web アプリケーションにアクセスし、画像をアップロードする。アップロードされた画像は、Raspberry Pi 4 上で処理され、リサイズやガンマ補正などの画像変換が行われる。変換された画像データは、C 言語で実装された LED パネル制御プログラムに渡され、HUB75 インタフェースを通じて LED マトリクスピネルに表示される。この一連の処理フローにより、遠隔地からでも LED マトリクスピネルの表示内容を制御することが可能となっている。

具体的には、以下のような流れで処理を行った。

1. Web サイトから画像をアップロードする。
2. アップロードされた画像をリサイズする。
3. リサイズされた画像をパネルに表示する。

2.3 256 階調表示システム

本節では、256 階調表示システムの構築について説明する。画像のリサイズから点灯制御まで、どのような方法で処理を行ったかを説明する。

2.3.1 RGB 各色の点灯回数制御による 256 階調表現

RGB 各色を点灯回数制御により 256 階調で表示するシステムを構築する。HUB75 では、一つの LED で R,G,B の 3 色をそれぞれ点灯させるかしないかを独立に制御が可能である。その仕様を利用して、R,G,B それぞれの点灯回数で 256 階調を表現する。

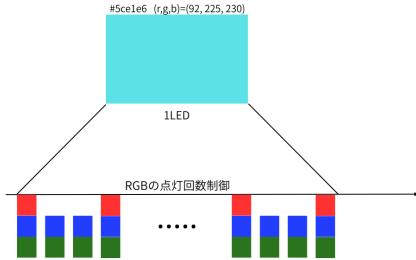


図 2.5: RGB 各色の点灯回数制御による 256 階調表現

例えば、図 2.4 の例では、 $(R,G,B)=(92,225,230)$ の場合は、256 階調のうち 92 回 R を点灯させ、225 回 G を点灯させ、230 回 B を点灯させることで表現が可能である。

2.3.2 点灯させるタイミングの計算方法

当初は、1 フレームを 256 スロットに分割し、各スロットにおける点灯 (ON/OFF) を整数演算で決定することで、8bit (0–255) の輝度値 R を時間方向の点灯回数として表現する方式を検討した。しかし、この「フレームを 256 分割する」という設計には主に 2 つの問題があることが分かった。

第一に、256 分割をそのままフレーム設計に直結させると、スロット更新周波数が過度に高くなる点である。例えば表示更新を 60 Hz のフレームとして固定した場合、内部のスロット更新は $60 \times 256 = 15,360$ Hz となり、輝度制御としては不要に高い周波数となる。さらに、HUB75 型 LED マトリクスは行走査（例：1/16 スキャン）を伴うため、実際の表示は「フレーム」と「走査」の二重の時間構造を持つ。このため、「256 分割 = 1 フレーム」という単純化は適切ではなく、以降は“フレーム”ではなく、行表示や OE 有効期間に対応する“スロット”を輝度制御の基本単位として扱う方針に改めた。

第二に、点灯タイミングを単純な整数除算や周期（例： $k = \lfloor 256/R \rfloor$ ）で決定する方式では、 R が 256 の約数でない場合に点灯を時間的に均等配置できず、階調の線形性や画質が損なわれる点である。例えば $R = 128$ であれば「2 回に 1 回点灯」のように等間隔配置が可能である一方、 $R = 200$ のように 256 の約数でない値では、点灯間隔が不均一になりやすく、周期の噛み合わせによって縞やちらつきの原因にもなる。

これらの問題を解決するため、本研究では累積誤差を用いた点灯タイミング制御方式を採用した。この方式は、各スロットごとに累積値（accumulator）を更新し、その累積値が 256 を超えた時点（オーバーフロー発生時）に点灯させることで、任意の輝度値 R に対して時間的に均等な点灯配置を実現する。この方法により、 R が 256 の約数でない場合でも、256 スロット中 R 回の点灯を均等に分散させることができ、階調の線形性と画質の向上が期待できる。

一般式は以下のようになる。

$$\text{acc}_{n+1} = (\text{acc}_n + R) \bmod 256 \quad (2.1)$$

ただし、 $\text{acc}_0 = 0$ とする。ここで、 acc_n は n 回目の点灯判定時の累積値、 R は目標とする輝度値（0 以上 255 以下の整数）、 acc_{n+1} は次回の点灯判定時の累積値を表す。この式により、 acc_{n+1} が acc_n より小さくなった場合（256 のオーバーフローが発生した場合）に点灯させることで、256 階調を均等に表現することができる。

具体例として $R = 200$ の場合を考える。初期値 $\text{acc}_0 = 0$ とすると、各スロットの更新は $\text{acc} \leftarrow \text{acc} + 200 \pmod{256}$ で進み、オーバーフローが発生したスロットで点灯する。例えば最初の 15 スロットでは、 acc は

$$\begin{aligned} 0 &\rightarrow 200 \rightarrow 144 \rightarrow 88 \rightarrow 32 \rightarrow 232 \rightarrow 176 \rightarrow 120 \\ &\rightarrow 64 \rightarrow 8 \rightarrow 208 \rightarrow 152 \rightarrow 96 \rightarrow 40 \rightarrow 240 \rightarrow 184 \end{aligned} \quad (2.2)$$

と推移し、このうち $200 \rightarrow 144$, $144 \rightarrow 88$, $88 \rightarrow 32$ のように値が減少する遷移がオーバーフローに対応する。したがって点灯列は「OFF, ON, ON, ON, OFF, ON, ON, ON, ON, OFF, ON, ON, ON, OFF, ON, ...」となる。これによって 256 階調を均等に点灯させ、256 階調を表現することができる。

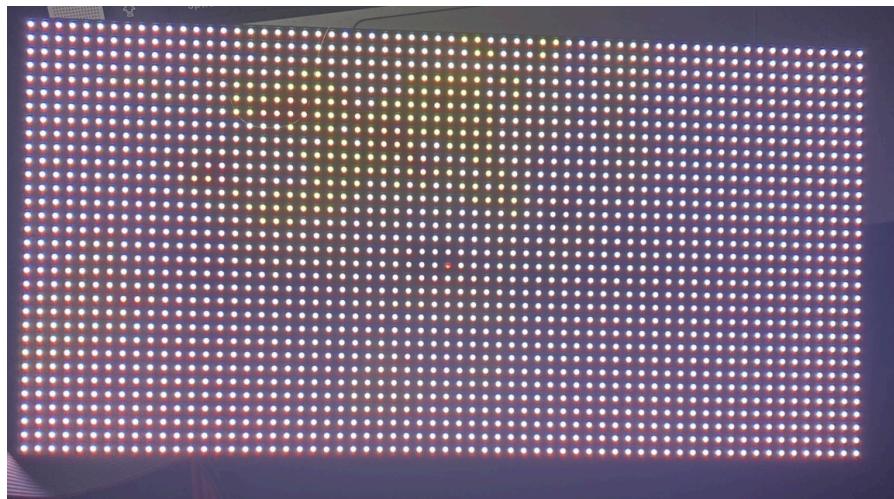


図 2.6: RGB 各色の点灯回数制御による 256 階調表現

この写真では、カメラで綺麗に撮影できるように、実際よりも高い 1000Hz の周

波数で実行している。実際に点灯すると色が淡くてよくわからないものの、画像をしっかりと表示できていることが確認できた。

2.3.3 32×64 画像へのリサイズとマッピング処理

今回は 32×64 の画像へリサイズするにあたり、各出力ピクセルが元画像上で占める領域を求め、その領域に含まれる画素の RGB 値を単純平均する方法（ボックス平均）を採用した。具体的には、出力画像の座標 (x, y) ($0 \leq x < 64$, $0 \leq y < 32$) に対して、元画像の幅を W 、高さを H とすると、対応する入力領域を

$$x_0 = \left\lfloor \frac{xW}{64} \right\rfloor, \quad x_1 = \left\lfloor \frac{(x+1)W}{64} \right\rfloor, \quad y_0 = \left\lfloor \frac{yH}{32} \right\rfloor, \quad y_1 = \left\lfloor \frac{(y+1)H}{32} \right\rfloor \quad (2.3)$$

で定義し、この矩形領域 $[x_0, x_1) \times [y_0, y_1)$ に含まれる全画素の RGB 値を加算して画素数で除算することで、出力画素の値を得る。RGB の各成分は独立に平均し、

$$R'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} R(i, j) \quad (2.4)$$

$$G'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} G(i, j) \quad (2.5)$$

$$B'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} B(i, j) \quad (2.6)$$

とした。ここで $\Omega(x, y)$ は上記入力領域に含まれる画素集合、 N はその画素数である。リサイズ後の RGB 値は、C 言語上で定義した RGB 構造体配列に格納し、LED マトリクスパネル表示処理に入力する。

実際に以下の図 2.6 のような画像をリサイズした結果を図 2.7 に示す。



図 2.7: 元画像



図 2.8: リサイズ後画像

2.4 ガンマ補正による画質改善

本節では、実際に画像を表示した際に色が淡くてよくわからない問題について原因を調査し、ガンマ補正を行って画質を改善する方法について説明する。

2.4.1 ガンマ補正

ガンマ補正とは、画像の輝度を補正するための手法である。通常のディスプレイではガンマ 2.2 の補正が行われているため、画像ファイルは内部的にガンマ 0.45 の補正が行われている。しかし、LED マトリクスピネルではこの補正が行われないため、元の画像よりも淡く表示される問題が発生していた。

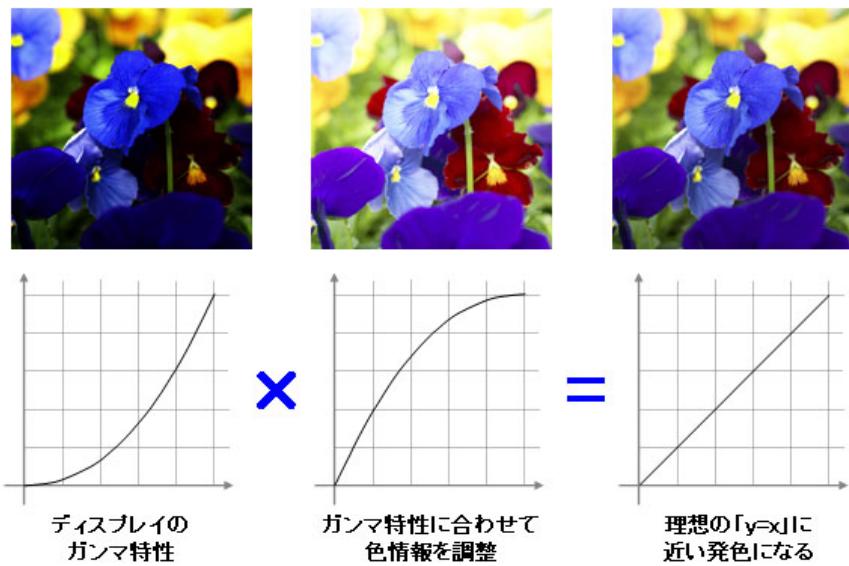


図 2.9: ガンマ補正の原理²

図 2.9を見てわかる通り、通常の画像ファイルはガンマ 0.45 の補正が行われている。この補正によって通常の見た目よりも淡くなる。この補正がかかっている状態でディスプレイに表示することで、ガンマ 2.2 の補正が行われ、明度は線形になる。しかし、LED マトリクスピネルではこの補正が行われないため、ガンマ 0.45 のまま表示していたことで、元の画像よりも淡く表示される問題が発生していた。

そこで本研究では、LED マトリクスピネル上で元画像と近い見た目を得るために、画像データをパネルへ出力する前段で逆ガンマ補正（ガンマデコード）を行った。具体的には、リサイズ後の各画素の RGB 値 (0–255) を 0–1 の範囲に正規化し、各成分 s に対して

$$L = s^{2.2} \quad (2.7)$$

を適用したうえで、再び 0–255 にスケーリングして整数化し、LED 表示用の RGB 構造体配列に格納した。実装上は、画素値 R , G , B をそれぞれ

$$s_R = \frac{R}{255}, \quad s_G = \frac{G}{255}, \quad s_B = \frac{B}{255} \quad (2.8)$$

²https://www.eizo.co.jp/eizolibrary/other/itmedia02_07/ より引用

とし、

$$R' = \text{round}(255 \cdot s_R^{2,2}) \quad (2.9)$$

$$G' = \text{round}(255 \cdot s_G^{2,2}) \quad (2.10)$$

$$B' = \text{round}(255 \cdot s_B^{2,2}) \quad (2.11)$$

として求めている。これにより、通常はディスプレイ側で行われているガンマ 2.2 の補正をソフトウェア側で補い、LED マトリクスピネル上でも元画像に近い明度分布を再現できるようにした。実際にこの方法でガンマ補正を行った結果は以下の図 2.10 の通りである。



図 2.10: ガンマ 2.2 補正画像

また、これを実際にパネルに表示した結果は以下の図 2.11 の通りである。



図 2.11: ガンマ 2.2 補正画像 (パネル表示)

写真越しにみると、あまり綺麗に見えないが、実際にパネルに表示した結果を

第 2 章

見ると、ガンマ 2.2 補正を行ったことで、補正を行っていない画像よりもより鮮明に綺麗に表示されていることが確認できた。

第3章 2つの異なる点滅周波数を用いた可視光通信方式

3.1 提案方式の概要

本節では、LED マトリクスパネルを用いた可視光通信方式について説明する。自分が提案する方式は、2つの異なる点滅周波数を用いてバイナリを表現してデータを送信する方式である。また、一枚の LED マトリクスパネルを 4 つの領域に分割し、一回で送信できるデータ量の増加を目指した。これらのロジック等について説明する。

3.2 可視光通信のロジック

本節では、具体的な可視光通信のロジックについて説明する。具体的には点灯パターンの設計と分割領域における点灯パターンの設計について説明する。

3.2.1 点灯パターンの設計

今回のシステムでは、2つの点滅周波数を用いて、バイナリを表現し、それを受信部で復号する方式を採用した。

当初は、領域を 4 つに分割することを中心に検討していたため、全体を高い方の周波数で駆動させ、低い方の周波数領域は $1/2$ になるように、2 回に 1 回点灯させるようにする方式を検討していた。しかし、そのやり方では、duty 比が高い方の周波数の時は 100%、低い方の周波数の時は 50% となる。これは、単位時間あたりで人間の目に入る光量が高周波領域の方が多くなってしまうため、人間の視覚的には激しいちらつきが発生してしまう。そのため、この方式は採用しなかった。(以下、この方式を「旧法」と呼ぶ) それを考慮して新たに点灯パターンを設計した。具体的には、両方とも duty 比が 50% となるようにする方式を採用した。

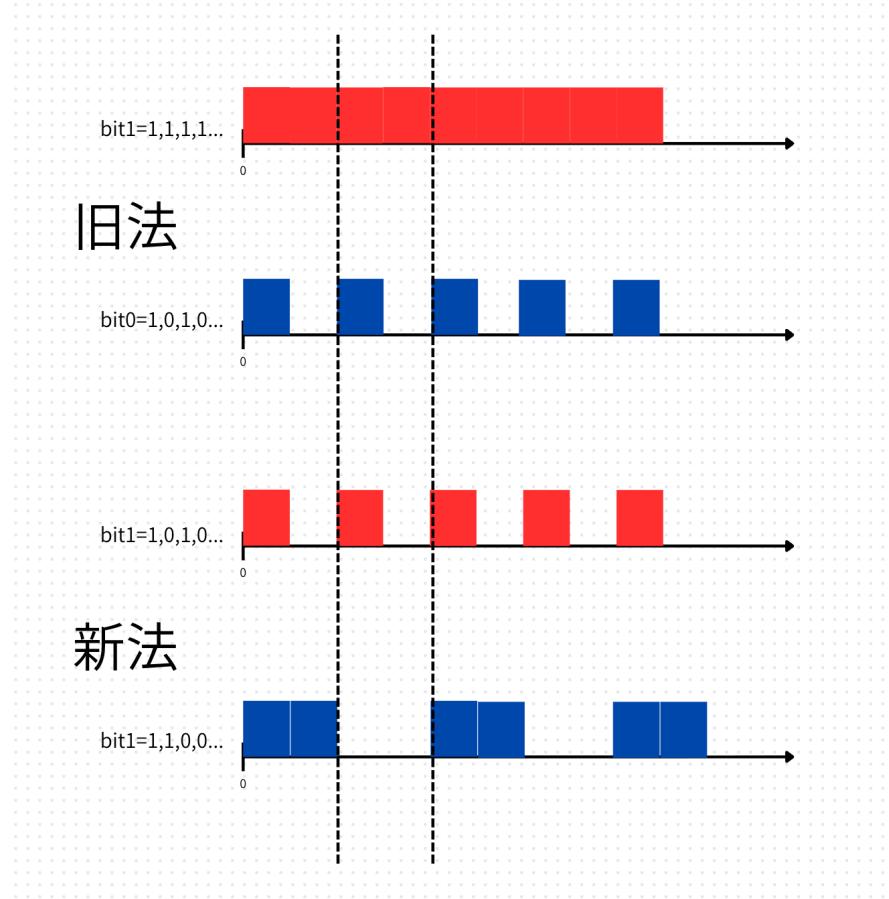


図 3.1: 提案方式におけるフリッカパターン

このパターンを用いることで、高い周波数の時は 50%、低い周波数の時も 50% となり、4 スロットで見たときに輝度値が揃うため人間の視覚的に違和感が生まれにくくなる。

3.2.2 バイナリ変換の方法

実際に点灯パターンから受信側でバイナリとして変換する方法について説明する。全体を 4 スロットで見て、点灯パターンが 1010 の場合は 1、1100 の場合は 0 としてバイナリを表現する。これを繰り返してバイナリを表現する。この方法を採用した理由は、バイナリのパターンによって見え方が変わらないようにするためである。例として、1 スロットごとに見て、点灯していたら 1、消灯していたら 0 としてバイナリを表現する方法を考える。この方法では、100.... というバイナリパターンの場合、人間の目では一瞬点灯した後にその後ずっと消灯するため、基本的に真っ暗になってしまう。しかし、自分が考案した方法の場合、10000... というバイナリパターンだとしても 4 スロットで見たときに輝度値が揃うため人間の

視覚的に違和感が生まれにくくなる。

当初は全体を 120Hz(1 スロット 1/120s)で駆動させ、60Hz と 30Hz で実験を行ったが、人間の目から見て若干のちらつきが生れたため、この方式は採用しなかった。

人間の視覚は、臨界フリッカ融合周波数 (critical flicker fusion) という周波数があり、その周波数以上はフリッカを感じなくなり、連続点灯に見えるという性質がある。条件によるが、およそ 60Hz から 100Hz と言われているため、120Hz は連続点灯に見える可能性が高い。そのため、高い周波数が 120Hz になるように全体を 480Hz(1 スロット 1/480s) で駆動させるようにした。

3.2.3 分割領域における点灯パターンの設計

ここでは、一枚の LED マトリクスパネルを 4 つの領域に分割し、それぞれの領域で異なる点滅周波数を用いて同一のデータを並列に送信する方式を採用した。

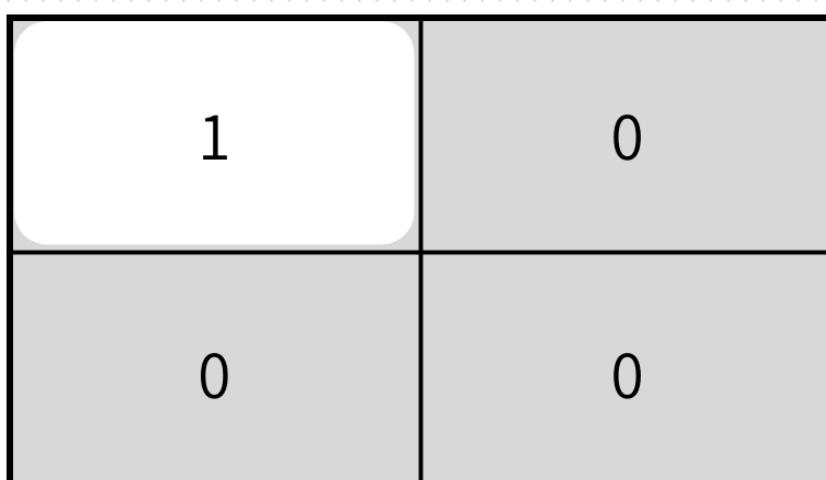


図 3.2: 4 分割 LED マトリクスの走査制御

図 3.2 のような場合、受信側では 1000 というデータに変換することができる。このように、分割領域における点灯パターンを設計することで、一回で送信できるデータ量を増加させることができる。この方法で実際に点灯した画像を図 3.3 に示す。

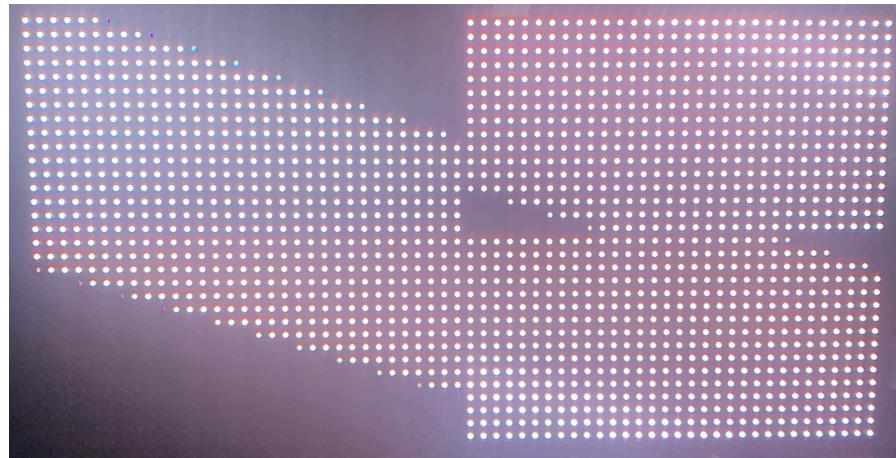


図 3.3: LED マトリクスパネルを 4 つの領域に分割した画像

図 3.3 では縞が見えるが、この画像だけではビットは推定ができない。理由は、露光時間が 1 スロットである $1/480\text{s}$ よりも長いため、正確にビットを推定することができないためである。しかし、画像からは縞模様ができていることが確認できる。スマホのカメラはローリングシャッター方式を採用しており、一回の撮影で上から下に光を取り込んでいるため、一回の露光でも上下で光を取り込む時刻が違う。そのため、今回は受信側のロジックまではしっかりと設計できていないため、細かな検討はできていないが、この縞模様のパターンからビットを推定することで、カメラのフレームレート以上のデータも受信できると考えられる。

3.2.4 4 分割以上の検証

受信側の実装が間に合わず、分割数の増加によって受信側の実装負担がどの程度増えるかを見積もれなかった。そこで、4 分割を前提に設計しつつ、8 分割についても試験的に検証した。

8 分割で点灯した場合、図 3.5 のようになる。

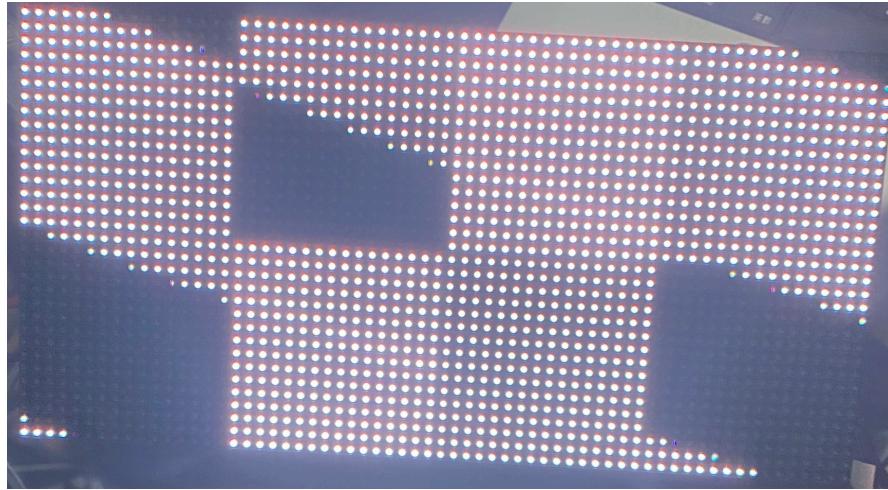


図 3.4: LED マトリクスパネルを 8 つの領域に分割した画像

このように、送信側の技術的には分割数を増やす余地がある。受信側で安定して読み取れるのであれば、分割数をさらに増やすことで、より高速な通信が実現できると考えられる。

3.3 フレーム構成とシンボル境界推定

本研究では同期語は使用しておらず、フレーム境界の推定は FLAG の検出と輝度パターンの整合性に基づいて行う。

3.3.1 送信フレームの構成

送信するデータの全体設計は以下の図 4.1 の通りである。

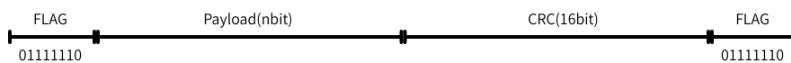


図 3.5: 送信データの全体設計

図 3.4において、FLAG はデータの開始位置と終了位置を示すものであり、Payload は実際に送信するデータの内容であり、CRC はビット誤り検出を行うものである。この設計は HDLC のフレーム同期方式を参考にしている。

3.3.2 FLAG 検出とシンボル境界推定

今回的方式では、FLAG を用いてデータの開始位置と終了位置を示す。今回、FLAG は 01111110 として設計したが、ペイロード内に 01111110 が含まれていた場合、そのビット列を FLAG として誤検出してしまう問題がある。そこで本研究ではビットスタッフィングを適用し、ペイロード中に 1 が 5 回連續して現れた場合、その直後に 0 を挿入する。受信側では、1 が 5 回連續した後に現れる 0 を取り除く（デスタッフィング）ことで元のペイロードを復元する。この処理により、ペイロード中に FLAG と同一のビット列が出現しないことが保証され、FLAG の検出が可能となる。

シンボル境界推定について、本研究では 1100 を 0、1010 を 1 に割り当て、1 シンボルを 4 スロットで構成する。受信側でシンボル境界（位相）がずれた場合、4 スロット窓で観測されるパターンは 1100 が 1001 や 0011 となるなど、送信時の並びと一致しなくなる。そのため、受信信号を 1 スロットずつシフトさせながら候補パターンとの整合を評価し、最も整合する位相をシンボル境界として採用する。特に 1010 は位相をずらしても 1010 または 0101 の交互パターンとなり、連続する 1 や 0 が生じにくい。この性質は位相推定における識別性を高める点で、本方式のシンボル設計の利点である。

ただし、カメラ撮像ノイズ等により、位相ずれではなく誤りによって 1001 のようなパターンが観測される可能性もある。そこで本研究では、ペイロード末尾に CRC を付与し、受信側で復号結果の整合性を検証する。具体的には、位相候補（0 ~3 スロットのシフト）ごとに復号を行い、CRC 検証に合格する候補を正しい復号結果として採用する。これにより、シンボル境界推定の誤りや伝送中のビット誤りを高確率で検出でき、誤ったデータ列を上位層へ渡すことを防ぐ。

3.4 受信アルゴリズム

本節では、スマートフォンカメラで撮影されたフレーム列から bit 列を復号するアルゴリズムについて述べる。

3.4.1 フレーム列からの時系列データ生成

受信側では、60fps 程度で取得した動画をフレームに分解し、各フレームに対して 4 分割領域内の縞パターンを抽出する。本方式は 4 スロットで平均輝度が一致するように設計しているため、領域内の平均輝度からは bit 差が得られない。そのため、ローリングシャッターによって生じる縞のパターンからビットの推定を行う。得られた縞プロファイルを時間方向に並べることで、領域ごとの時系列データを生成する。

3.4.2 bit パターンとの類似度評価と位相推定

送信側では 1 シンボルを 4 スロット (1 スロット = 1/480s) で構成しているため、受信側の 60fps では 1 シンボルあたりの観測点が不足する可能性がある。そこで、縞パターンから得られた時系列データに対して、理論パターン 1010 および 1100 に対応する縞の並びと一致するかを評価する。

また、ローリングシャッターやフレームレートの揺らぎによる位相ずれに対応するため、開始位置を 1 フレームずつずらして複数の位相候補を評価する。各位相で得られたスコアが最大となるパターンを選択し、1010 なら bit=1、1100 なら bit=0 として復号する。これにより、同一シンボル長でも撮影条件に依存しない復号が可能になる。

3.4.3 復号フロー全体

復号フローは、大きく「前処理」「FLAG 検出」「payload 復号」の 3 段階で構成される。まず前処理としてフレーム列から縞パターン時系列を生成し、正規化を行う。次に、FLAG パターンと一致する区間を探索してフレーム境界を決定する。このとき、位相をずらした複数候補を評価し、最も一致度の高い位置を採用する。

境界が確定した後は、payload 領域に対してシンボルごとの類似度評価を行い、bit 列を復号する。誤検出が発生した場合は、FLAG 再検出に戻ることで再同期を行い、長い payload でも復号を継続できる構成とした。

第4章 評価および考察

本章では、第2章から第3章まで構築してきたLEDマトリクス表示システムおよびフリッカ周波数差を用いた可視光通信方式について、表示品質と通信性能の観点から評価を行い、その結果を踏まえて考察を行う。まず実験条件を整理し、人間の視覚における見え方等を評価し、設計改善の流れと今後の課題についてまとめる。

4.1 人間の視覚における見え方の評価

本節では、旧方式（デューティ比に差がある単純なフリッカ方式）と、提案方式（平均デューティ50%のバランス符号）を比較し、人間の目で見たときの色味と違和感を評価する。

4.1.1 旧方式と提案方式の色味の比較

2章で提案した旧法と新法を比較する。旧法では、4スロットの範囲での光量が違うため、高い周波数の方が光量が多くなり、低い周波数では光量が少なくなる。その影響で明暗が激しく切り替わりチカチカしていた。一方、新法では、両周波数ともデューティ比が50%となるように設計しているため、人間の目には白色と黄色が混在したような色味差が生じない。また、新法では、4スロットの範囲での光量が揃うように設計しているため、人間の目には白色と黄色が混在したような色味差が生じない。

4.1.2 周波数ごとの色差評価

周波数ごとの色差についてだが、自分が検証したのは120Hz駆動と240Hz駆動と480Hz駆動の3つである。この3つの周波数について、色差を評価する。第一に、120Hz駆動だが、120Hz駆動の場合は60Hzと30Hzの点滅周波数となるが、30Hz成分が際立つやすい。そのため、人間の目からはかなりちらつきが目立った。第二に、240Hz駆動だが、240Hz駆動の場合は120Hzと60Hzの点滅周波数となるため、先ほどよりもちらつきが少なくなったがまだ若干ちらつきを感じた。第三に、480Hz駆動だが、480Hz駆動の場合は240Hzと120Hzの点滅周波数となり、

第一、第二と比較してほとんどちらつきを感じなくなった。

4.2 考察

今回LEDマトリクスパネルを用いて256階調での画像の表示、外部から画像をアップロードして表示を行えるシステムの構築、LEDマトリクスパネルを用いた可視光通信の実験を行った。本章の評価結果から、表示品質と通信性能の間には明確なトレードオフがあることが分かる。旧方式ではデューティ比の差により周波数ごとの明るさが不均一となり、ちらつきと色味差が目立った。一方、提案方式では4スロット平均で輝度が一致するように設計したため、人間の目には白色の揺らぎが少なくなり、表示装置としての違和感が大きく改善された。

周波数条件については、120Hz/240Hzでは低周波成分が残り、ちらつきが視認できたのに対し、480Hzではほぼ連続点灯に見えた。ただし、高周波化は縞パターンのピッチを細かくし、撮影条件によっては縞のコントラストが低下する可能性がある。すなわち、視覚的な快適さを優先すると通信の検出が難しくなりやすく、逆に検出性を優先するとちらつきが増えるという関係がある。

また、提案方式は平均輝度が一致するため、領域内の平均輝度からはbit差が得られず、受信側はローリングシャッターによって生じる縞パターンを用いて復号する必要がある。この設計により視覚的品質は高いが、受信アルゴリズム側の前処理や位相推定に依存する部分が大きく、撮影距離や角度、露光条件の変化に対しても安定して復号ができるようになるという点が今後の課題である。

さらに、4領域並列化は通信速度を向上させる一方、1領域あたりの画素数が減少するため、縞パターンの判別が難しくなる場合がある。実運用を想定する場合は、距離や表示サイズに応じて分割数やシンボル長を調整する必要があると考えられる。

4.3 結論および今後の課題

本研究では、LEDマトリクスパネルを用いた表示システムの構築と画質改善を行い、その上でフリッカ周波数差を用いた可視光通信方式を提案した。表示面では、256階調表示とガンマ補正により視覚的な違和感を抑え、通信面では4分割領域と自分が提案した点灯方式を用いることで、ちらつきを抑えつつ情報を埋め込める事を示した。一方で、受信側の実装と実測による通信性能評価は十分に行えておらず、縞パターン抽出や位相推定の頑健性を含めた検証が今後の課題である。

今後は、表示・通信・受信アルゴリズムを統合したシステムとして実装し、撮影距離や角度、露光条件に対する復号性能の評価を進める必要がある。また、誤り訂正符号やシンボル長の最適化、分割数の設計指針の整備を通じて、更なる実用性や安定性の向上を目指したい。

参考文献

謝辞

2025年12月 園邊 翔大

研究業績リスト

【公表論文】 【国際会議】