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

園邉　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

図 目 次

表 目 次

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、これらの課題を解決する技術として注目されている．可視光通信は、人間の目に見える光（可視光）を通信媒体として利用する技術であり、既存の照明設備を活用できることから、追加の通信インフラを必要としない点が大きな利点である．また、可視光は電磁波干渉を受けにくく、セキュリティ面でも優れている．さらに、LED（Light Emitting Diode）の普及により、高速な点滅制御が可能となり、実用的な通信速度を実現できるようになった．

一方、街中には大型の LED マトリクスパ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、送信側の点滅周波数と同期を取ることが難しい．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 マトリクスパネルを用いた可

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 つの 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．各領域では、単位時間ごとに点灯周波数を変えることでデータを送信し、高周波領域は 50 これにより、単一領域で送信する場合と比較して 4 倍の通信速度を実現することを目指した．また、デューティ比の制御により、高周波領域（120Hz）では各フレームの輝度が 0.5, 0.5, ... となり、低周波領域（60Hz）では 1, 0, 1, ... となる．この輝度の変化パターンの違いにより、カメラ側では点滅を検出して復号できる一方で、人間の目には連続した白い点灯として認識される．

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 2.1[1]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．

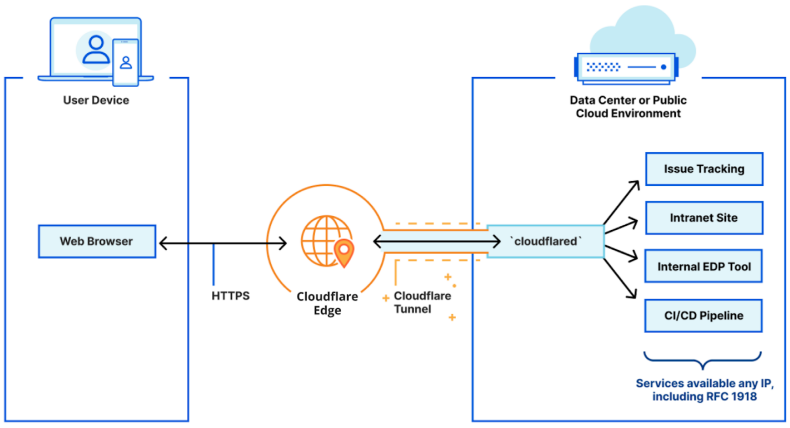

図 2.2: Cloudflare Tunnel の概要図[1]

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

[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。このサイトでは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.

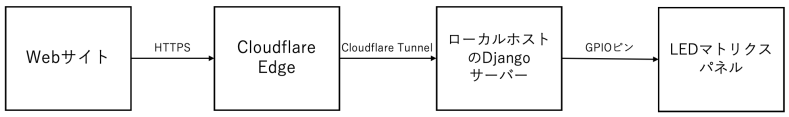

図 2.4: システム全体の構成

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 マトリクスパネルに表示される．この一連の処理フローにより、遠隔地からでも LED マトリクスパ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．

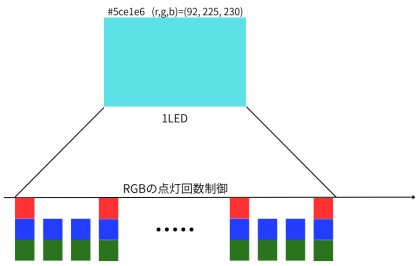

図 2.5: RGB 各色の点灯回数制御による 256 階調表現

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必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。

一般式は以下のようになる．

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

ただし，$\mathrm{acc}_0 = 0$ 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 $\mathrm{acc}_0 = 0$ とすると，各スロットの更新は $\mathrm{acc} \leftarrow \mathrm{acc} + 200 \pmod{256}$ 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

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

と推移し，このうち $200 \to 144$，$144 \to 88$，$88 \to 32$ 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．

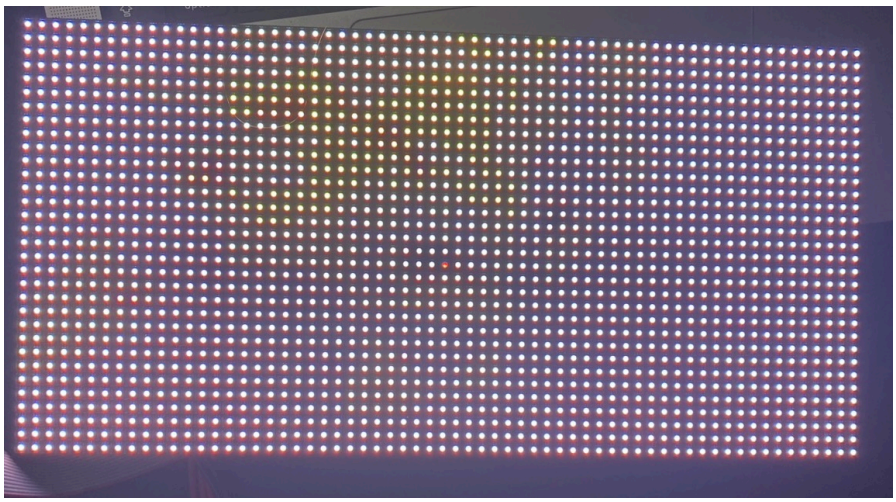

図 2.6: RGB 各色の点灯回数制御による 256 階調表現

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

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

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、

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

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

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

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 マトリクスパ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。

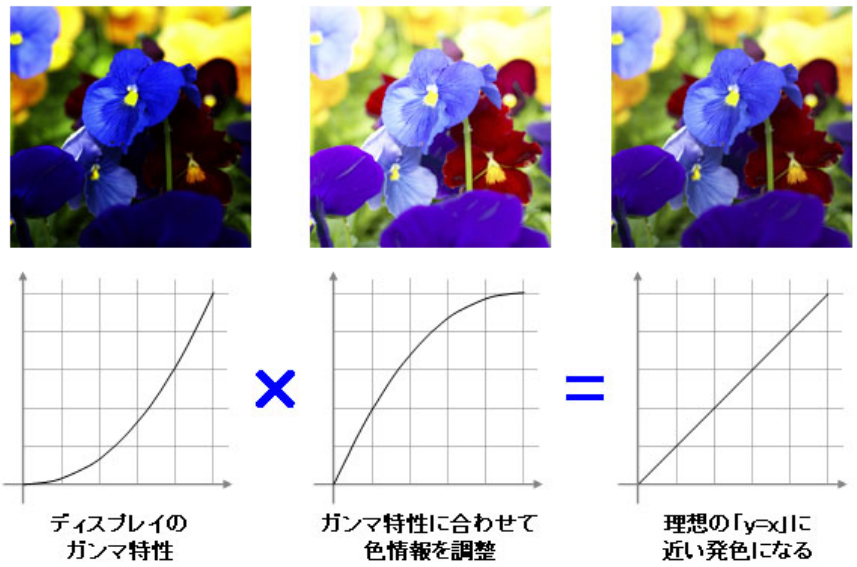

図 2.9: ガンマ補正の原理[2]

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 マトリクスパ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 マトリクスパ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

$$L = s^{2.2} \tag{2.7}$$

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

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

とし、

$$R' = \mathrm{round}(255 \cdot s_R^{2.2}) \tag{2.9}$$
$$G' = \mathrm{round}(255 \cdot s_G^{2.2}) \tag{2.10}$$
$$B' = \mathrm{round}(255 \cdot s_B^{2.2}) \tag{2.11}$$

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

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 マトリクスパ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。

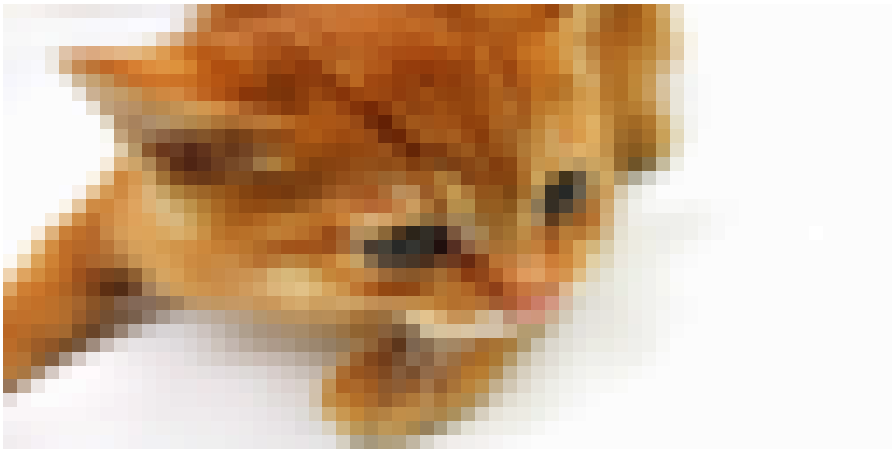

図 2.10: ガンマ 2.2 補正画像

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　LEDマトリクスのフリッカ周波数差を用いたカメラベース可視光通信方式

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同一のデータを並列に送信する方式である。今回のシステムでは、2つの異なる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。また、今回は4つの領域に分割して、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。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した。

今回の実験では、送信システムのみの実験を行った。

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%となってしまい、人間の視覚に違和感を与えるため、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、高い方の周波数の時は50%、低い方の周波数の時は100%となるようにする方式を採用した。

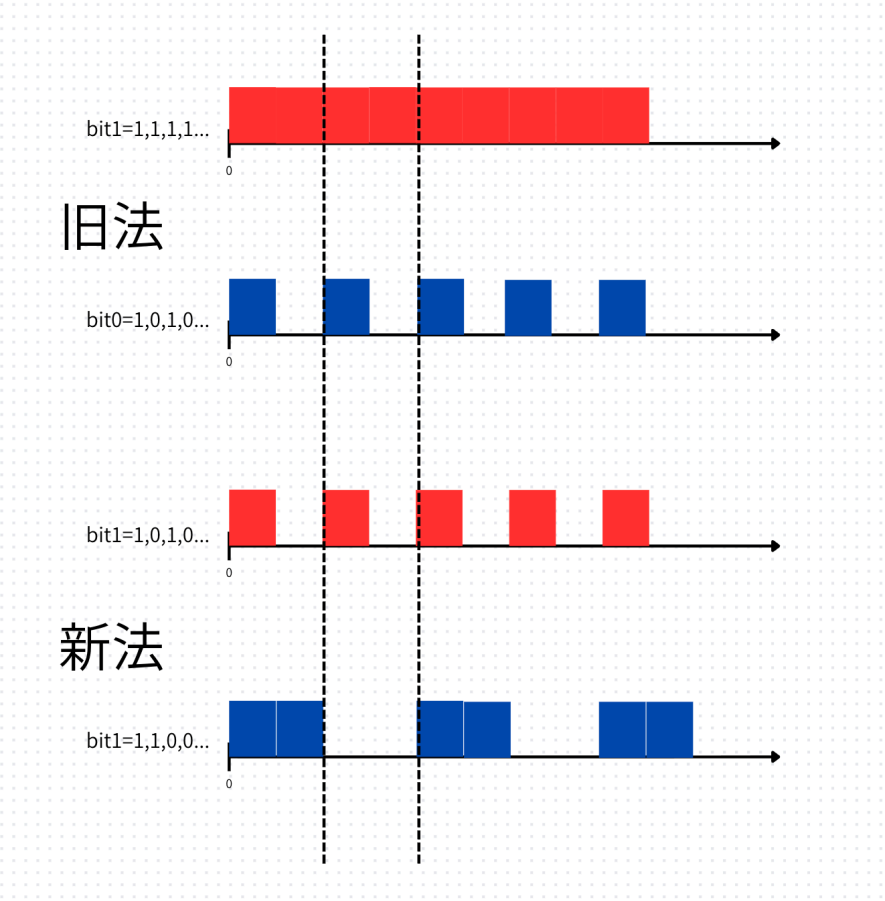

図 3.1: 提案方式におけるフリッカパターン

このパターンを用いることで、高い周波数の時は 50%、低い周波数の時は 100% となり、人間の視覚に違和感を与えないようにすることができる。

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。

また、当初は 60Hz と 120Hz 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まれた。

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 60 100Hz と言われているため、60Hz は連続点灯に見える可能性が高い。

3.2.3　分割領域における点灯パターンの設計

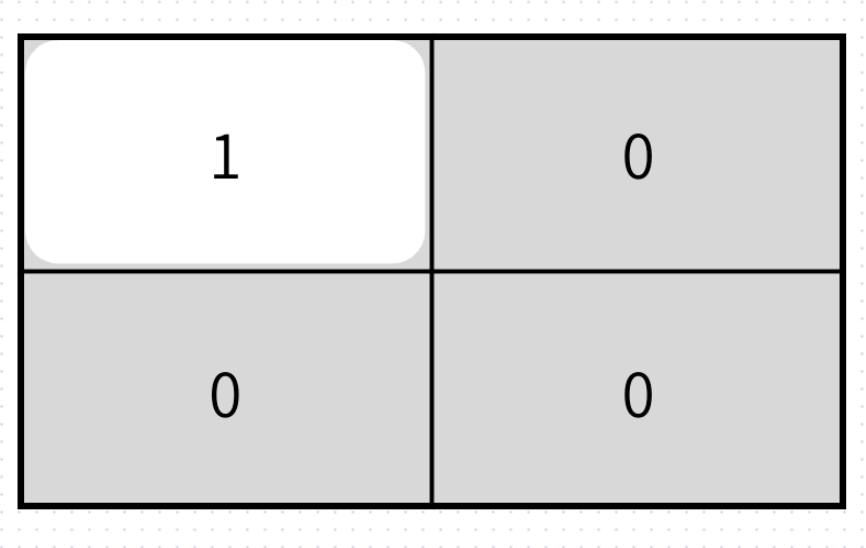

図 3.2: 4 分割 LED マトリクスの走査制御

3.2.4　C コード実装とシンボル列の構成

3.3　バランス符号による色味差低減

本節では，人間の視覚における色味差を抑えるために導入したバランス符号について述べる．

3.3.1　naive 方式で生じる色味差の問題

本小節では，論理 1 を常時点灯（100%デューティ），論理 0 を半分点灯（50%デューティ）とした単純なフリッカ表現では，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る。

3.3.2　平均デューティ 50%のバランス符号設計

本小節では，bit=1 を 1010…，bit=0 を 1100…とするような時間パターンを用いて，どちらの bit でも周期内の点灯回数を揃える「バランス符号」の設計方針を述べる．120Hz フレーム列に対して 8 フレームを 1 周期としたときの具体的な波形イメージを言葉で説明し，人間の視覚では平均輝度が揃って見える一方で，カメラ側では周期パターンの違いから bit を識別できる。

3.4 シンボル設計と同期語

本節では，シンボル系列の構成と同期取得のための同期語設計について述べる．

3.4.1 preamble シンボル列による同期取得

3.4.2 同期語とシンボル境界推定

3.4.3 理論スループットの算出

3.5 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．

3.5.1 フレーム列からの時系列データ生成

3.5.2 bit パターンとの類似度評価と位相推定

3.5.3 復号フロー全体

3.6 実装コードと最適化

3.7 小括

第4章　評価および考察

4.1　実験条件

4.2　人間の視覚における見え方の評価

4.2.1　旧方式と提案方式の色味の比較

4.2.2　白色との色差評価

4.3　通信性能評価

4.3.1　ビット誤り率と同期成功率の評価

4.3.2　送信可能な情報量と必要時間の試算

4.4　考察

4.5　結論および今後の課題

参考文献

謝辞

2025 年 12 月　園邉　翔大

研究業績リスト

【公表論文】【国際会議】