シングルピクセルイメージング
専用計算機に関する研究

2022年8月

千葉大学大学院融合理工学府
基幹工学専攻電気電子工学コース

星　郁雄

(千葉大学審査学位論文)

シングルピクセルイメージング専用計算機に関する研究

2022年8月

千葉大学大学院融合理工学府
基幹工学専攻電気電子工学コース

星　郁雄

概要

一般的なイメージングでは一画素分にあたる光検出器を撮影したい画素数分だけ並べて作製したイメージセンサを用いて被写体の像を撮影する．イメージセンサとしてはCharge-Coupled Device (CCD) や Complementary Metal-Oxide-Semiconductor (CMOS) などがある．それに対してシングルピクセルイメージングは，一点の光の強さを検出する単一の光検出器を用いて，再構成計算により被写体の像を再構成する手法である．イメージセンサを必要としないため，コストの問題でイメージセンサの作製が難しい可視光以外での波長帯でのイメージングなどで応用が期待されている．しかし，シングルピクセルイメージングでは計算によって被写体の像を再構成するが，必要とされる計算量が多く組み込み向け Central Processing Unit (CPU) などの小型な計算機では処理が難しいため，高性能な計算機が要求される．本論文では，設計者が自由に書き換え可能な集積回路である Field-Programmable Gate Array (FPGA) を用いた小型かつ高性能なシングルピクセルイメージング専用計算機に関する研究を行った．

はじめに，小規模な FPGA 評価ボードを用いてシングルピクセルイメージングの再構成計算を処理する計算回路を実際に設計，試作し評価までを行った．結果としてデスクトップ PC よりも 15 倍高速に再構成計算を処理可能な計算回路を試作することができた．しかし課題として画質が低いほか，計算回路の機能のみを実装したため制御用の計算機が別途必要であった．そこで次に再構成画像の画質を改善するため，高画質なシングルピクセルイメージング画像再構成の研究を行った．

高画質なシングルピクセルイメージング画像再構成については，ディープラーニングを利用した画像再構成の研究を行った．ディープラーニングは，人の神経細胞を模した数理モデルであるニューラルネットワークをベースとした手法で機械学習の一種である．画像関連の分野はもちろんのことほかの様々な分野でも優れた成績を残していることで知られている．ディープラーニングを利用した画像再構成手法については二通りの研究を実施し，その両方で有効性が確認できた．

最後に，試作段階だった計算回路の課題を解決したシングルピクセルイメージング専用計算機の開発を行った．課題の一つであった画質の低さについては，計算回路規模などの観点からディープラーニングを利用した画像再構成のうち計算回路に適した手法を実装し画質の改善を達成した．もう一つの課題であった制御用の計算機が必要である課題については，Central Processing Unit と FPGA がワンチップに実装された System on a Chip (SoC) FPGA を利用し，SoC FPGA 上にある CPU から FPGA を制御することで課題解決を図った．結果として，ワンチップで小型，高性能かつスタンドアロンなシングルピクセルイメージング専用計算機の開発に成功した．本論文で開発した専用計算機を用いることでシングルピクセルイメージングの計算をワンチップで処理することが可能になる．これによって例えば，シングルピクセルイメージングを利用した小型かつ安価な赤外波長センサなどが実現できる．

Abstract

Conventional imaging techniques use an image sensor consisting of many light detectors, such as a charge-coupled device (CCD) and complementary metal-oxide-semiconductor (CMOS). By contrast, single-pixel imaging only uses a single-element photodetector without spatial resolution. A target object image is obtained by reconstruction calculation. It has advantages that enable broad wavelength imaging because it does not require an image sensor. This technique requires a high-performance computer for the reconstruction calculation; however, it is difficult to reconstruct for a compact computer such as an embedded central processing unit (CPU).

This dissertation aims to develop a compact and high-performance dedicated computer using a filed-programmable gate array (FPGA) for single-pixel imaging. FPGA is a large-scale integrated (LSI) that users freely rewrite a circuit. At first, I designed and implemented a calculation circuit for a reconstruction calculation with a small-scale FPGA evaluation board. As a result, the calculation circuit succeeded in 15 times faster processing than a desktop computer. However, the calculation circuit has problems with low image quality and requires a computer to control the calculation circuit.

To improve the image quality, I studied deep-leaning-based reconstruction methods with high image quality. Deep learning is one of the machine learning methods. It is based on a neural network, which is a mathematical model imitating neuron cells. Deep learning has achieved excellent performance in various fields such as image recognition and image processing. I developed two reconstruction methods using deep learning, and both could be confirmed effective.

Finally, I developed a dedicated computer that improved the problems of the calculation circuit. The image quality problem is resolved by implementing the reconstruction method using deep learning. The another problem requiring a computer is resolved by using a system on a chip (SoC) FPGA. A SoC FPGA is a CPU and FPGA implemented into one chip. As a result, I developed a compact, stand-alone, and high-performance dedicated computer. The dedicated computer enables to process of reconstruction calculation on a single chip. This allows, for example, a compact and inexpensive sensor in the infrared wavelength range to be realized with a single-pixel imaging.

目次

図目次

表目次

序論

現在一般的に広く利用されているイメージングでは，図 1 のように一画素分にあたる光検出器を撮影したい画素数分だけ並べて作製したイメージセンサを用いて被写体の像を撮影する．そのようなイメージセンサは 1960 年代に入って提案され [1, 2]，現在は Charge-Coupled Device (CCD) や Complementary Metal-Oxide-Semiconductor (CMOS) などの高性能なイメージセンサとしてイメージング技術を支えている．イメージセンサの基本性能としては，1) 画素数，解像度，2) イメージセンサのサイズ，3) 速度，フレームレート，4) 消費電力，5) 感度，6) ダイナミックレンジが挙げられ，現在それぞれの性能はイメージセンサとして達成できる物理的な限界に近付きつつあるとされている．しかし，イメージセンサの物理的な制約から，これらにはそれぞれ画素数や解像度とイメージセンサのサイズ，速度やフレームレートと消費電力，感度とダイナミックレンジがトレードオフとなっている [3]．そのため，これらの性能全てを両立したイメージセンサはいまだ実現していない．

そのようなイメージングを実現するための手段のひとつとしてシングルピクセルイメージングがある [4–6]．一般的なイメージングではイメージセンサを用いるのに対して，シングルピクセルイメージングは一点の光の強さを検出する単一の光検出器を用いて，再構成計算により被写体の像を再構成する手法である．単一の光検出器は空間的な分解能を持たないので，被写体の光強度を検出するだけではイメージングは不可能である．そこでシングルピクセルイメージングでは，符号化パターンを被写体に投影した際の光の強度を一点に集め単一光検出器で計測する．これを異なる符号化パターンで複数回繰り返し，取得した光強度の情報と既知の符号化パターン情報から再構成計算によって被写体の像を再構成することができる．再構成した像の解像度と画素数は投影した符号化パターンの解像度と再構成計算に依存する．この事から，イメージセンサのサイズなど物理的な制約にとらわれることなく高画素数，高解像度化が可能，単一の光検出器のみを用いるため省電力下で非常に高速な信号の計測が可能，光電子増倍管 (Photomultiplier Tube : PMT) などの光検出器を利用し高感度なイメージングも可能，などの特徴を持つ．そのため，イメージセンサのトレードオフにとらわれないイメージングの基本性能を全て満たすことができる．ほかにも，イメージセンサの作製技術が成熟していない波長帯でのイメージングにも利用することができる．これらの利点から期待されている応用は高速フローサイトメトリー [7]，リモートセンシング [8]，物体追跡 [9]，三次元計測 [10, 6], テラヘルツなど赤外波長帯でのイメージング [11–13] と多岐にわたるため，Internet of Things (IoT)[14–16] において重要な役割を担うことが期待されている．従来のインターネットはコンピュータ同士を接続するものであったが，IoT はこれまでインターネットに接続されていないようなものも含めて接続し，構成されるインターネットのことである．家

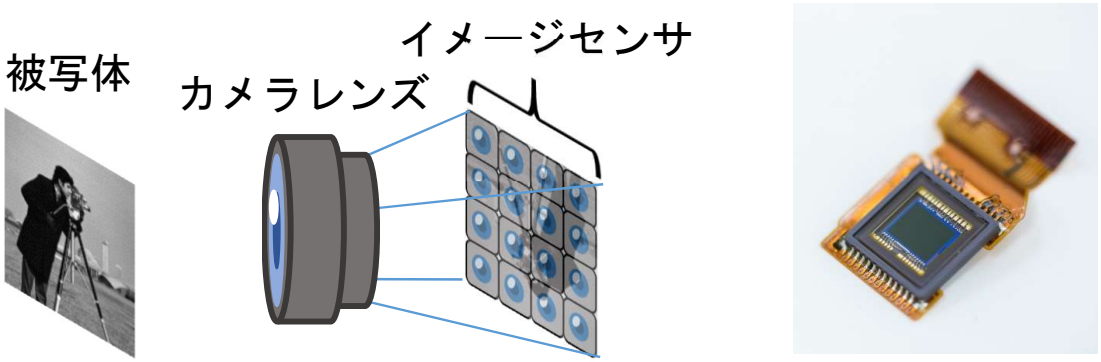


図 1: イメージセンサを利用した一般的なイメージングとイメージセンサの実物

電や自動車，工場設備などをインターネットで接続し，センサで計測した情報をそれぞれが取得，活用することで離れた機器の状態をモニターしたり制御することができる．シングルピクセルイメージングはイメージセンサを用いる場合よりも様々なセンシングが可能なため，IoT でのセンシングの幅を拡張することができる．

シングルピクセルイメージングでは計算処理によって画像を再構成するため，Personal Computer (PC) などの計算機が必要不可欠である．しかし，組み込み向けの Central Processing Unit (CPU) では計算性能が足りないので，結果としてセンシングシステム全体が大型化してしまう課題がある．これは前述した応用への大きな障害となる．そこで本研究では，シングルピクセルイメージングの実用化に向けて小型かつ高性能なシングルピクセルイメージング専用計算機開発に関する研究を行った．

本論文の研究は大きく四つに分かれる．一つ目は再構成計算を処理するための専用計算回路の試作，二つ目と三つ目は試作した専用計算回路の課題を受けた画像再構成手法についての研究であり，最後は残った課題を解決したシングルピクセルイメージング専用計算機の開発についての研究である．一つ目の研究では，専用計算機開発にあたってまず Filed-Programmable Gate Array (FPGA) を用いてシングルピクセルイメージングの計算を処理する専用計算回路を試作した．FPGA とはユーザーが自由に書き換え可能な Large-Scale Integration (LSI) であり，特定の計算に特化した回路を設計することで高速な計算機を実現できる [17–20]．高速計算で利用されるデバイスとしては，ほかに Graphic Processing Unit (GPU) があるが，FPGA は GPU よりも一般的に消費電力が少ない傾向があり小型化に向いている．他にも，FPGA ではゲート単位で計算回路の設計が可能なためリソースを効率的に利用可能なことや，センサなどから計測した入出力の信号を回路に直接入力できるため入力信号を高いスループットで処理可能などの利点があり本研究では FPGA を利用した．試作した計算回路は結果として，CPU で計算した場合と比較して 15 倍高速に計算が可能であることを示すことができた [21]．この結果から FPGA を利用したシングルピクセルイメージング専用計算機の有効性が確認できたが，試作した計算回路は再構成画像が低画質で，計算回路の機能のみを実装したため制御用の PC

が別途必要であった．また，シングルピクセルイメージングで主に扱われている画像サイズ (128 × 128pixel) よりも小さい (32 × 32pixel) 画像しか再構成できないなど，課題も多かった．

そこで次に画質を改善するため再構成手法についての研究を行った．シングルピクセルイメージング再構成手法の研究を二通り行い，どちらもディープラーニングを利用した．ディープラーニングは，人の神経細胞を模した数理モデルであるニューラルネットワークをベースとした手法で機械学習の一種である [22, 23]．ニューラルネットワークの層を多層化して高い性能を達成したためディープラーニングと呼ばれている．まず一つは，単一の光検出器から取得した光強度を直接ディープラーニングに入力として与え，出力として再構成画像を出力する手法の研究を行った．本研究では先行研究で良好な結果を残していたネットワーク構造に加え，時系列データの扱いに優れたネットワーク構造の一つであるリカレントニューラルネットワークを組み入れることで更なる画質の向上を図った．結果として，スパースな画像などに対して先行研究のネットワーク構造よりも画質の向上が確認できた他，より少ないパラメータ数で画像を再構成できることを確認した [24]．二つ目は，試作した計算回路に実装していた計算をディープラーニングの学習を通して最適化することで画質の向上を図った．こちらも結果として試作回路に実装していた再構成計算よりも画質の改善が確認できた [25]．

最後に，試作機の課題を改善したシングルピクセルイメージング専用計算機の開発を行った．画質の課題に関しては，開発した再構成手法を計算回路に実装することで改善した．二つの再構成手法のうち一つ目の手法は画質面で優れていたが回路規模が大きい．対して，二つ目の手法は回路規模を小さくでき大きいサイズの画像を再構成できる．よって本研究では課題であった画像サイズと画質の両方を向上できる二つ目の手法を計算回路に採用した．制御用の計算機が別途必要であった課題に関しては，System on a chip (SoC) FPGA を利用することで解決を図った．SoC FPGA は一つのチップ上にユーザーが設計可能な Programmable Logic (PL) 部分と組み込み向け CPU である Processing System (PS) 部分が実装された LSI であり，PL 部分に計算回路を実装し PS 部分から制御することでワンチップで完結するスタンドアロンな専用計算機の開発が可能である [26]．最終的な結果として，128 × 128pixel の再構成画像をワンチップで高速に再構成可能なシングルピクセルイメージング専用計算機の開発に成功した．本研究で開発したシングルピクセルイメージング専用計算機を用いることで，システム全体の小型化につながり，期待されている様々な応用の社会実装が期待できる．

本論文は全 5 章で構成されている．第 1 章ではシングルピクセルイメージングの概要と先行研究としていくつかの再構成手法について述べる．第 2 章では最初に試作した計算回路の構成について述べ，その後評価と考察について述べる．第 3 章ではリカレントニューラルネットワークを利用したシングルピクセルイメージング画像再構成手法について述べる．はじめにニューラルネットワークや畳み込みニューラルネットワーク，リカレントニューラルネットワークなどディープラーニングのネットワークについて述べた後，先行研究と提案手法について述べ，その後評価と考察について述べる．第 4 章ではディープラーニングで利用されている勾配降下法を利用したシングルピクセルイメージングの画質改善について述べる．先行研究と提案手法について述べた後，評価と考察について述べる．第 5 章では SoC FPGA を用いたシングルピクセルイメージング専用計算機の概要について述べる．専用計算機の構成

と計算回路について述べた後，再構成画像や速度，光学実験から評価し考察を述べる．最後に本論文を総括する．

第1章　シングルピクセルイメージング

1.1　シングルピクセルイメージングについて

シングルピクセルイメージングは計算処理によって被写体を再構成する．そのため，今日まで多くの再構成手法が考案されている．符号化パターンと単一の光検出器を用いてイメージングを行うものはそのほとんどがシングルピクセルイメージングとして呼称されているが，シングルピクセルイメージングとして扱われつつ別の呼称で呼ばれる手法なども存在する．そこで，まずシングルピクセルイメージングの歴史的な経緯を簡単に述べた後，これまで提案されてきた中でも代表的な再構成手法を紹介する．

シングルピクセルイメージングが最初に報告されたのは 1990 年代中頃で，現在では特にゴーストイメージングと呼ばれている [27]．2 光路に分割した光のうち片方を被写体に照射し，照射されていない光との相関から画像を再構成する．最初の報告では光子対の量子もつれに関する実験に利用されていたが，その後，図 1.1 に示すような散乱媒質などを通過しスペックルが生成された光源を利用するゴーストイメージングが提案された [28, 29]．これらは量子もつれに利用されていた方を量子論的ゴーストイメージング，散乱媒質を用いる方を古典論的ゴーストイメージングと区別して呼ぶことが多い．

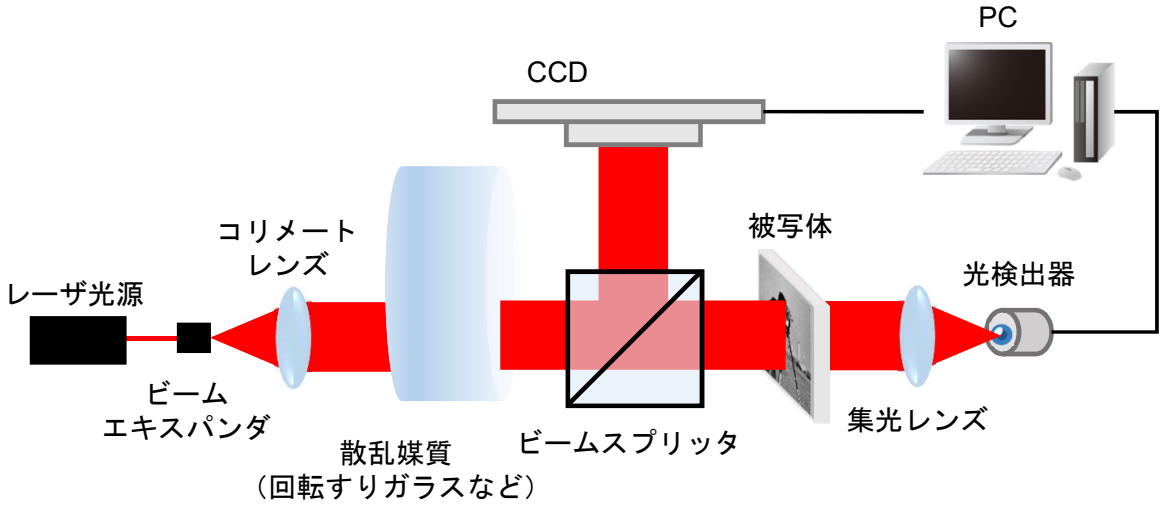


図 1.1: 古典的ゴーストイメージング

図 1.1 に示した光学系を用いる古典論的ゴーストイメージングでは，レーザなどのコヒー

レント光源をすりガラスに透過させることで散乱からスペックルパターンが生成される．更にすりガラスを回転させることで時間的に異なるスペックルパターンが生成される．これが現在のシングルピクセルイメージングで符号化パターンにあたる部分である．生成されたスペックルパターンを持つ光源はビームスプリッタで 2 光路に分割され，片方は被写体に照射後レンズで集光され光検出器で光強度を計測する．もう片方は生成されたスペックルパターンを観測するため CCD で波面を計測し，これをすりガラスを回転させながら繰り返し，異なるスペックルパターンで計測を繰り返す．光検出器で計測された光強度は以下のようにあらわされる．

$$S_i = \iint I_i(x,y)T(x,y)dxdy \tag{1.1}$$

$I_i(x,y)$ は CCD で i 回目に計測されたスペックルパターンを持つ光源の光強度分布で，$T(x,y)$ は被写体の強度係数である．この計測した光強度とスペックルパターンの光強度分布から被写体の像は

$$\begin{aligned} O(x,y) &= \langle \Delta S_i \Delta I_i(x,y) \rangle \\ &= \langle (S_i - \langle S_i \rangle)(I_i(x,y) - \langle I_i(x,y) \rangle) \rangle \\ &= \langle S_i I_i(x,y) \rangle - \langle S_i \rangle \langle I_i(x,y) \rangle \end{aligned} \tag{1.2}$$

の式で再構成される．本論文では $\langle S_i \rangle = \sum_{i=1}^{N} S_i$ として，N 個分のアンサンブル平均を表す．Δ はそれぞれの偏差を表しているため，式 (1.2) は数学的には光検出器により計測した光強度とスペックルパターンを持つ光源の各要素との共分散を計算していることになる．

古典論的ゴーストイメージングの後，新たに計算機ゴーストイメージングという手法が提案された [30]. 計算機ゴーストイメージングでは，古典論的ゴーストイメージングで回転すりガラスなどを利用して生成していた時間的に異なるスペックルパターンを持つ光源ではなく，空間光変調器 (Spatial Light Modulator : SLM) に符号化パターンを表示して変調した構造化照明を用いる．この手法の大きな特徴として，古典論的ゴーストイメージングでは図 1.1 からもわかるように，光学系は 2 光路かつ単一光検出器とは別に CCD などのイメージセンサなど，光源の波面を計測できるものが必要であった．それに対して計算機ゴーストイメージングでは空間光変調器に既知のランダムなパターンを表示することで時間変化する変調された構造化照明を生成するため，図 1.2 のように計算機シミュレーションで照射した光源の波面を取得でき光源を計測する機構が必要ない．なお，[30] で提案されたものは図 1.2(a) のように透過型の空間光変調器を用いているが，図 1.2(b) のように反射型の空間光変調器を用いて光学系を構築することもできる．

計算機ゴーストイメージングの欠点としては，光源の生成に空間光変調器を用いるため，撮影の速度が空間光変調器のフレームレートに律速されてしまい，光強度の計測を繰り返すことに時間がかかってしまう．そのため，現在では白と黒で構成されたバイナリのパターンに限り高速にパターンの切り替えが可能な反射型空間光変調器である Digital Micromirror Device (DMD) がよく用いられる．

計算機ゴーストイメージングの登場によって，これまでのゴーストイメージングよりも光学系の構築が容易になり実験が容易になった．それによって多くの関連研究が行われること

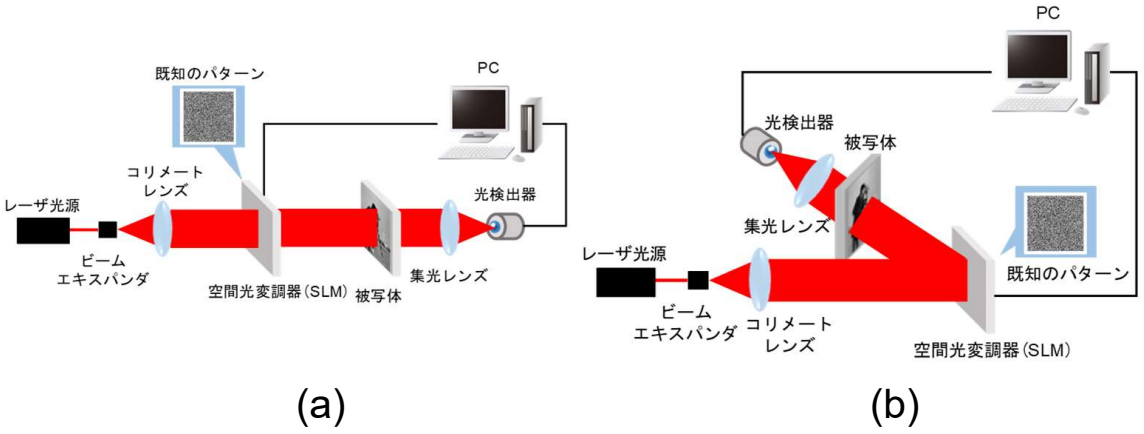


図 1.2: (a) 透過型空間光変調器を用いた計算機ゴーストイメージング，(b) 反射型空間光変調器を用いた計算機ゴーストイメージング

となり，今日まで様々な画像再構成手法が提案され，それらの多くはシングルピクセルイメージングと呼ばれている．ゴーストイメージングとシングルピクセルイメージングの定義は明確ではないが，傾向的に相関計算の原理を利用しているものはゴーストイメージングと呼ばれ，それ以外の再構成手法ではシングルピクセルイメージングと呼ばれている傾向がある．計算機ゴーストイメージングを契機に現在シングルピクセルイメージングと呼ばれる研究が増加したこと，現在シングルピクセルイメージングと呼ばれているものの多くは空間光変調器を用いており計算機ゴーストイメージングがベースにあることから，本論文中では図 1.3 のように計算機ゴーストイメージング以降の手法をシングルピクセルイメージングと位置づける．また，図 1.3 で重複している部分は，他の手法と統一するためこれ以降，相関計算を利用したシングルピクセルイメージングとして位置づけ，呼称することとする．

次節以降では，提案された様々な画像再構成手法について述べていく．多くの再構成手法が存在するシングルピクセルイメージングだが，それぞれの特徴を表 1.1 に示した．画質に関して，再構成画像の画素数を M として，測定回数を N としたとき，原理的には $M \leq N$ を満たせば画像は完全に復元されるはずであり，満たせない場合は不良設定問題となるので画質は劣化する．また，どの手法も測定回数の制限が無ければほぼ完全に画像を復元できる．そのため，測定回数が $M > N$ の場合での画質を 3 段階で評価した．撮影速度については，同程度の画質を得るのに要する測定回数の多さから評価している．

本論文中では表 1.1 のように，大きく 4 つの手法に分けて述べることとする．

1.2 相関計算を利用したシングルピクセルイメージング

まず相関計算を利用したシングルピクセルイメージングについて述べる．この手法では一般的に，乱数などで生成したランダムなパターンを空間光変調器に表示することで光源を変調する．平均や共分散など統計的な計算を利用しているので，ノイズに強い利点を持つ．一

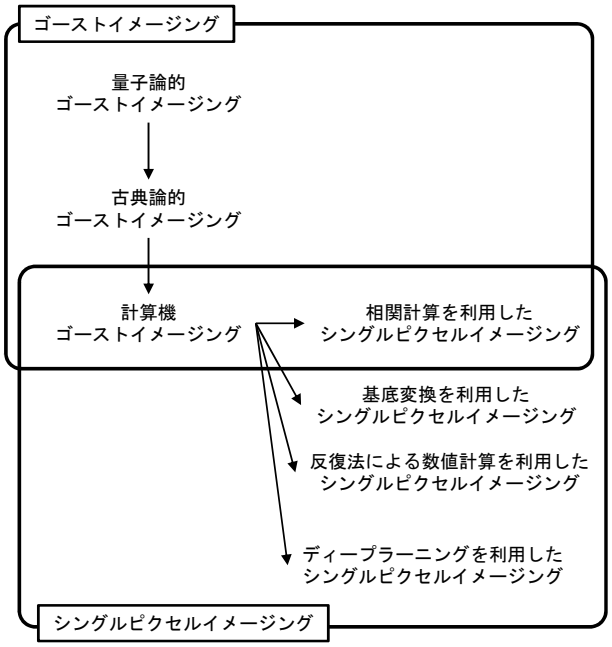


図 1.3: 本論文内におけるゴーストイメージングとシングルピクセルイメージングの分類

表 1.1: シングルピクセルイメージング再構成手法の種類と特徴

	画質	撮影速度	耐ノイズ性
相関計算	×	×	◯
基底変換	△	×	×
圧縮センシング	◯	△	△
ディープラーニング	◯	◯	△

方で，高画質な再構成画像を得るためには他の手法よりも多くの測定回数が必要となる．

相関計算を利用したシングルピクセルイメージングは，前節でも述べた計算機ゴーストイメージングの式 (1.2) から発展した再構成手法である．本節では，それらの手法や応用について述べることとする．

計算機ゴーストイメージングの発展として，再構成計算の式 (1.2) よりもコントラストがはっきり再構成できる Differential Ghost Imaging (DGI) という再構成手法が提案されている [31, 32]．再構成計算は式 (1.1) で示した光強度 S_i を用いて，

$$\begin{aligned} O(x,y) &= \langle (S_i - \frac{\langle S_i \rangle}{\langle R_i \rangle} R_i)(I_i(x,y) - \langle I_i(x,y) \rangle) \rangle \\ &= \langle S_i I_i(x,y) \rangle - \frac{\langle S_i \rangle}{\langle R_i \rangle} \langle R_i I_i(x,y) \rangle \end{aligned} \tag{1.3}$$

のようにあらわされる．R_i は被写体に投影した構造化照明の光強度であり，

$$R_i = \iint I_i(x,y) dx dy \tag{1.4}$$

とあらわすことができる．

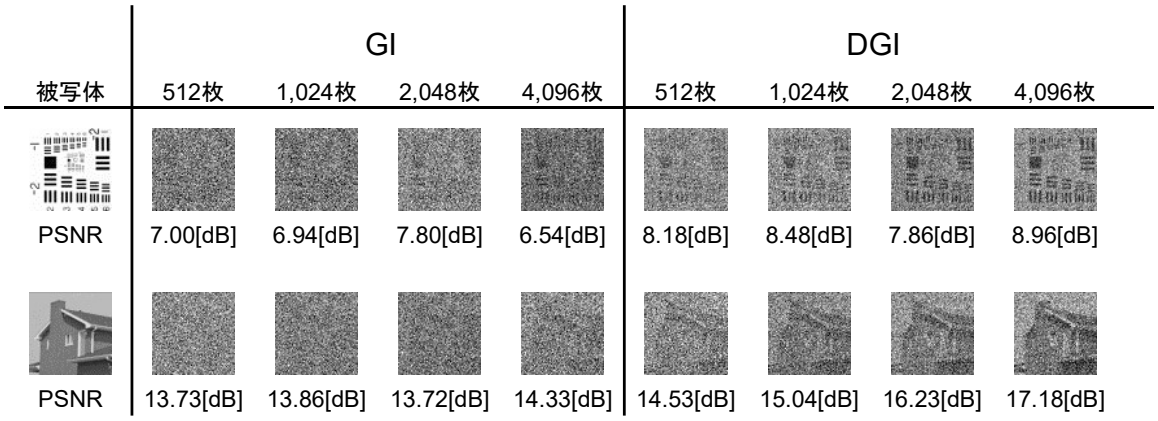


図 1.4: 各符号化パターン数での GI と DGI による再構成画像

図 1.4 に GI と DGI で画像再構成を行った結果を示す．符号化パターンは 512 枚，1,024 枚，2,048 枚，4,096 枚とした．また，再構成画像サイズは 64×64 [pixel] とした．画質の評価には Peak Signal-to-Noise Ratio (PSNR) と呼ばれる指標を用いた．これは画像の画質評価に用いられる指標で，以下のような式で計算される．

$$PSNR = 10 \log_{10} \frac{MAX_I^2}{MSE} \tag{1.5}$$

$$MSE = \frac{1}{XY} \sum_{j=1}^{Y} \sum_{i=1}^{X} (I(i,j) - K(i,j))^2 \tag{1.6}$$

MAX_I は画像が取る最大の画素値であり，256 階調化されている画像なら 255 である．$I(i,j)$ が基準画像，$K(i,j)$ が比較画像をあらわし，式 (1.6) は両者の平均二乗誤差を計算している．PSNR の単位は [dB] で，式 (1.5) より 2 枚の画像が類似しているほど PSNR は高い値を示すことがわかる．図 1.4 より，実際に GI より DGI の方が画質が向上している様子が確認できる．

相関計算を利用したシングルピクセルイメージングとしてほかに，再構成計算式 (1.2) で行う平均の計算を小領域に分割することで時間変化するノイズを軽減させた Time Division Ghost Imaging (TDGI) などが提案されている [33]．TDGI の再構成計算は以下に示すとおりである．

$$O(x,y) = \frac{1}{N}\sum_{i=1}^{N}(S_i - \langle S_i^{|1|}\rangle)I_i(x,y) + \cdots + \frac{1}{N}\sum_{i=(k-1)N+1}^{kN}(S_i - \langle S_i^{|k|}\rangle)I_i(x,y) \tag{1.7}$$

上の式では kN 回測定したものを k 個の領域に分割してそれぞれで画像を再構成し，それらを足し合わせて最終的な画像を再構成している．$\langle S_i^{|k|}\rangle$ は区間 k の範囲での平均を示す．小領域で分割した中で平均を求めるので誤差の蓄積を DGI より抑制でき，時間変化するノイズに強い再構成が行える．

相関計算を利用したシングルピクセルイメージングの応用としては，エッジ画像再構成を行うものがある [34]．被写体のエッジ情報は，

$$\nabla T(x,y) = T(x,y) - T(x+m, y+n) \tag{1.8}$$

のようにあらわされることを利用する．m,n は任意の整数である．m,n 分だけずらした符号化パターンを空間光変調器に表示する事で，

$$\begin{aligned} S_i &= \iint I_i(x,y)T(x,y)dxdy \\ S_i' &= \iint I_i(x-m,y-n)T(x,y)dxdy = \iint I_i(x,y)T(x+m,y+n)dxdy \end{aligned} \tag{1.9}$$

$$\nabla S_i = S_i - S_i' = \iint I_i(x,y)\nabla T(x,y)dxdy \tag{1.10}$$

のようにエッジ情報についての光強度が取得できる．その後，再構成計算式 (1.2)，(1.3) のように再構成を行うことでエッジ画像の再構成を行える．この手法に関しても，本手法の後に様々な手法が提案されている [35–37].

1.3 基底変換を利用したシングルピクセルイメージング

次に本節では，基底変換を利用したシングルピクセルイメージングについて述べる．基底変換を利用したシングルピクセルイメージングでは，空間光変調器に正規直交基底行列から生成した符号化パターンを表示する．符号化パターンと被写体を重ね合わせ，一点に集光し光強度として測定することで基底変換が光学的に行われる．計測した光強度に対して基底行

列に対応した逆変換を施すことで画像を再構成する．基底変換を利用したシングルピクセルイメージングの特徴は，他の手法とは異なり再構成したい画像の画素数分だけ測定を行うことで元の画像をほぼ完全に復元できる点である．測定回数が不足している場合は，その分だけ周波数成分が欠けていることになるので，再構成画像がぼやけてしまったり，ピッチがあらい画像になってしまう．また，正規直交基底は取得できる情報の重複がなく冗長性がないためノイズにも弱い．

利用される基底変換としては，主にアダマール変換 [38–40] やフーリエ変換 [40, 41] などがよく利用されているが，本節ではアダマール変換についてのみ述べることとする．アダマール変換はアダマール行列を用いる線形変換のことである．アダマール行列は各要素が ± 1 で構成される正方行列である．次数はかならず 2 の累乗数になり，2^k 次のアダマール行列は以下のように再帰的に生成される．

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.11}$$

$$\boldsymbol{H}_{2^k} = \begin{bmatrix} \boldsymbol{H}_{2^{k-1}} & \boldsymbol{H}_{2^{k-1}} \\ \boldsymbol{H}_{2^{k-1}} & -\boldsymbol{H}_{2^{k-1}} \end{bmatrix} \tag{1.12}$$

また，この行列の各行を波数の昇順に並び変えたものはウォルシュアダマール行列と呼ばれている．これ以降，$\boldsymbol{H}_N$ は N 次のウォルシュアダマール行列を示すものとする．二次元のウォルシュアダマール変換は，被変換行列を $N \times N$ の $\boldsymbol{T}$ として，変換後に得られる行列を $\boldsymbol{S}$ とすると，

$$\boldsymbol{S} = \boldsymbol{H}_N \boldsymbol{T} \boldsymbol{H}_N \tag{1.13}$$

のような行列積の形式であらわされ, それぞれの行列の要素の演算では

$$s_{i,j} = \sum_{n=0}^{N} \sum_{m=0}^{N} t_{m,n} h_{i,m} h_{n,j} \tag{1.14}$$

のようにあらわされる．よって，ウォルシュアダマール行列の i 行目の行ベクトル $\boldsymbol{h}_{row,i}$ と j 列目の列ベクトル $\boldsymbol{h}_{column,j}$ によって以下のように得られる $N \times N$ 個の N 次正方行列 $\boldsymbol{W}_{i,j}$ を，図 1.5 に示すようにそのまま符号化パターンとして単一の光検出器により光強度として測定することで，ウォルシュアダマール変換と同様の操作を光学的に行うことが可能になる．その際，-1 は光学的に表現することはできないので，白と黒を反転させたパターンを表示し，その時測定した光強度を元のパターンで測定した光強度から減算することで -1 を表現することができる．

$$\boldsymbol{W}_{i,j} = \boldsymbol{h}_{column,j} \times \boldsymbol{h}_{row,i} \tag{1.15}$$

アダマール行列は $H_N H_N = I_N$ になる性質を持つため，以下のようにあらわされるウォルシュアダマール逆変換を測定した光強度に施すことによって，画像を再構成することがで

$$\boldsymbol{H_2} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{-1} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}[\mathbf{1} \quad \mathbf{1}] \qquad \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}[\mathbf{1} \quad \mathbf{-1}] \qquad \begin{bmatrix} \mathbf{1} \\ \mathbf{-1} \end{bmatrix}[\mathbf{1} \quad \mathbf{1}] \qquad \begin{bmatrix} \mathbf{1} \\ \mathbf{-1} \end{bmatrix}[\mathbf{1} \quad \mathbf{-1}]$$

図 1.5: N=2 の場合でのウォルシュアダマール行列による符号化パターンの生成

きる．

$$\boldsymbol{T} = \boldsymbol{H}_N \boldsymbol{S} \boldsymbol{H}_N \tag{1.16}$$

図 1.6: アダマール変換での再構成画像

図 1.6 に，実際にアダマール変換によって再構成した画像と画質を示す．符号化パターン数は 512 枚，1,024 枚，2,048 枚，4,096 枚とした．また，再構成画像サイズは 64×64 [pixel] とした．ここまでで述べた通り，画像サイズと同数の符号化パターンを用いた場合はほぼ完全に被写体が復元できている事，符号化パターンを減らした分だけ解像度が粗くなっている事が確認できる．

以上に述べた方法で構造化パターンで被写体を基底変換した場合の再構成は行われるが，一方でこのようにして得られた光強度から，前節で述べた符号化パターンとの相関計算によっても画像再構成できることが報告されている [38]．その場合の画質はランダムパターンを用

いた相関計算と，基底逆変換で再構成した場合の中間の画質になる．

1.4 圧縮センシングを利用したシングルピクセルイメージング

次に本節では，圧縮センシングを利用したシングルピクセルイメージングについて述べる．圧縮センシングは，ある解を求めるのに本来必要な情報が不足している場合にほしい情報を再構成するための技術である [42]．解の大部分が零成分であるスパース性が期待できるのであれば，情報が不足していても比較的高い精度で情報を再構成することができる．そのような例として，連立方程式がある．連立方程式を構成する変数の数を N として，方程式の数が M である場合，一般に $M < N$ の場合にその連立方程式の解を一意に決めることはできない．このような系は劣決定系と呼ばれている．しかし，例えばこの解のうち K 個が零をとる仮定の下なら，$M > K$ なら解を一意に決めることが可能となり，このような解をスパース解と呼ぶ．圧縮センシングでは一般的に，そのようなスパース解を持つことが想定される問題を最小化問題として数値計算による反復法で解を得る．

シングルピクセルイメージングで圧縮センシングを利用する場合も同様に最小化問題として定式化する．符号化パターンを被写体に投影して一点に集光して測定する操作は線形変換なので $\boldsymbol{A}$ として表す．測定後の信号を $\boldsymbol{y}$，被写体の画像を $\boldsymbol{x}$ と置くと，シングルピクセルイメージングの観測過程は，

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n} \tag{1.17}$$

のようにあらわされる．$\boldsymbol{n}$ は観測誤差をあらわす．この観測過程を制約として，正則化項を追加するとシングルピクセルイメージングの最小化問題は以下のようにあらわされる．

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2\lambda}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \mathcal{R}(\boldsymbol{x}) \tag{1.18}$$

圧縮センシングでは，最小化される関数はコスト関数と呼ぶ．第 1 項が観測過程での制約，第 2 項が正則化項にあたる．また，コスト関数の第 1 項では次に示す L_2 ノルムを用いている．

$$\|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + x_1^2 + \cdots + x_N^2} \tag{1.19}$$

$\boldsymbol{x}$ が基底変換などの線形変換を施した場合にスパース性が期待できる場合，その線形変換を $\boldsymbol{B}$，逆変換を $\boldsymbol{B}^{-1}$ として $\boldsymbol{z} = \boldsymbol{B}\boldsymbol{x}$ と置くと，

$$\underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{1}{2\lambda}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{B}^{-1}\boldsymbol{z}\|_2^2 + \mathcal{R}(\boldsymbol{z}) \tag{1.20}$$

と置くことができ，実際に式 (1.18) の最小化問題と同様の方法で最小化を行える．正則化項目には，L_1 ノルムや，全変動 (Total variation) などがよく用いられる．L_1 ノルムは，以下のようにあらわされる．

$$\|\boldsymbol{x}\|_1 = |x_1| + |x_1| + \cdots + |x_N| \tag{1.21}$$

全変動は二次元行列で隣接要素同士の差分をとるもので，$x_{i,j}$ を要素に持つ要素数 $\mathbb{R}^{N_X \times N_Y}$ の二次元ベクトル $\boldsymbol{x}$ を用いて以下のようにあらわされる．

$$\mathrm{TV}(\boldsymbol{x}) = \sum_{j=1}^{N_y} \sum_{i=1}^{N_x} \sqrt{(x_{i,j} - x_{i,j+1})^2 + (x_{i,j} - x_{i+1,j})^2} \tag{1.22}$$

本論文では，式 (1.18) の正則化項に以下の L_1 ノルムを用いた場合について述べることとする [42].

$$\underset{\boldsymbol{x}}{\mathrm{argmin}} \frac{1}{2\lambda} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \|\boldsymbol{x}\|_1 \tag{1.23}$$

式 (1.23) の最小化問題は一般に Least Absolute Shrinkage and Selection Operators (LASSO) と呼ばれる．式 (1.23) の第 1 項は 2 次の凸関数であるため，そのような凸関数を $f(\boldsymbol{x})$ としたとき，以下のように微分して勾配を求めることで最小化が可能である．

$$\underset{\boldsymbol{x}}{\mathrm{argmin}}\{f(\boldsymbol{x})\} \tag{1.24}$$

$$\boldsymbol{x}[t+1] = \boldsymbol{x}[t] - \eta \nabla f(\boldsymbol{x}) \tag{1.25}$$

η は一度に更新するステップ幅をあらわす．しかし，正則化項に絶対値を含む場合は，絶対値関数を微分できないためそのような最小化は適用できない．そのため，絶対値を含んだ最小化問題は場合分けをする必要がある．まず，簡単のため以下のような最小化問題を考える．

$$\underset{x}{\mathrm{argmin}} \left\{ |x| + \frac{1}{2\lambda}(y - x)^2 \right\} \tag{1.26}$$

$x > 0$ の場合この最小化問題は

$$\underset{x>0}{\mathrm{argmin}} \left\{ x + \frac{1}{2\lambda}(y - x)^2 \right\} \tag{1.27}$$

となり，平方完成によってコスト関数は

$$\frac{1}{2\lambda}\{x - (y - \lambda)\}^2 + y - \frac{2}{\lambda} \tag{1.28}$$

のようにあらわすことができる．$x > 0$ であることと，式 (1.28) のコスト関数が平方完成された二次関数の形式になっていることを考慮すると，最小値を与える x は，$y - \lambda > 0$ の場合は頂点の $y - \lambda$ となり，$y - \lambda \leq 0$ の場合は $x = 0$ となる．$x \leq 0$ の場合も同様に解くことができ，その結果最小値を与える x は，

$$x^* = S_\lambda(y) \tag{1.29}$$

$$S_\lambda(y) = \begin{cases} y - \lambda & (y > \lambda) \\ 0 & (-\lambda \leq y \leq \lambda) \\ y + \lambda & (y < -\lambda) \end{cases} \tag{1.30}$$

となる．$S_\lambda(y)$ は軟判定しきい値関数 (Soft thresholding function) と呼ぶ．L_1 ノルムは式 (1.21) で示したように，ベクトルの場合各成分ごと独立に計算することができるため，以下のように考えることができる．

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ |\boldsymbol{x}| + \frac{1}{2\lambda}(\boldsymbol{y} - \boldsymbol{x})^2 \right\} \tag{1.31}$$

最小値は各成分ごとに軟判定しきい値関数を適用するので，

$$x_i^* = S_\lambda(y_i) \tag{1.32}$$

となる．ベクトル形式で標記すると，以下のように書ける．

$$\boldsymbol{x}^* = \boldsymbol{S}_\lambda(\boldsymbol{y}) \tag{1.33}$$

このようにして，絶対値を含むコスト関数でも最小化問題を解くことができる．しかし，式 (1.23) のコスト関数では線形変換 $\boldsymbol{A}$ を経ているため，以上の方法をそのまま適用することはできない．そこで，次は線形変換を含む L_2 ノルム部分に，メジャライザー最小化を適用する．メジャライザー最小化では，凸関数 $g(\boldsymbol{x})$ に対してメジャライザーと呼ばれる以下の関数を最小化していく．

$$q_L(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{y}) + (\nabla g(\boldsymbol{y}))^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{1.34}$$

$q_L(\boldsymbol{x}, \boldsymbol{y})$ は，$g(\boldsymbol{x}) \leq q_L(\boldsymbol{x}, \boldsymbol{y})$ が成立するため，$q_L(\boldsymbol{x}, \boldsymbol{y})$ を逐次的に最小化していくことで元の関数である $g(\boldsymbol{x})$ も最小化されていく．ここで，凸関数 $g(\boldsymbol{x})$ は微分がリプシッツ定数 L のリプシッツ連続である必要がある．リプシッツ連続とは，ある関数 $f(\boldsymbol{x})$ について以下の式を満たすことである．

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{1.35}$$

よって，凸関数 $g(\boldsymbol{x})$ の微分がリプシッツ連続であるということは

$$\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{1.36}$$

を満たす．メジャライザー $q_L(\boldsymbol{x}, \boldsymbol{y})$ は平方完成から，

$$q_L(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{y}) - \frac{1}{2L}\|\nabla g(\boldsymbol{y})\|_2^2 + \frac{L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{y} - \frac{1}{L}\nabla g(\boldsymbol{y})\right)\right\|_2^2 \tag{1.37}$$

と書き直すことができ，$q_L(\boldsymbol{x}, \boldsymbol{y}) \leq g(\boldsymbol{y})$ を満たすことがわかる．よって，$g(\boldsymbol{x}) \leq q_L(\boldsymbol{x}, \boldsymbol{y}) \leq g(\boldsymbol{y})$ となるので，あるステップ t に得られた解を $\boldsymbol{x}[t]$ として，メジャライザーを最小化する $\boldsymbol{x}$ で次のステップの解 $\boldsymbol{x}[t+1]$ を以下のように更新していく．

$$\boldsymbol{x}[t+1] = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, q_L(\boldsymbol{x}, \boldsymbol{x}[t]) \tag{1.38}$$

この時，先ほど示した不等式から以下の関係が得られるので，更新を繰り返すことで $g(\boldsymbol{x})$ は最小値へと到達する．

$$g(\boldsymbol{x}[t+1]) \leq q_L(\boldsymbol{x}[t+1], \boldsymbol{x}[t]) \leq g(\boldsymbol{x}[t]) \tag{1.39}$$

ここで，最小化問題の式 (1.23) にメジャライザー最小化を適用する場合は，$g(\boldsymbol{x}) = \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2/2\lambda$，$\nabla g(\boldsymbol{x}) = -A^{\mathrm{T}}(\boldsymbol{y} - A\boldsymbol{x})/\lambda$ とする．また，リプシッツ定数は式 (1.36) のリプシッツ連続から $L = \|A^{\mathrm{T}}A\|_2/\lambda$ となる．L_1 ノルムを加えても，$g(\boldsymbol{x}[t+1]) + \|\boldsymbol{x}[t+1]\|_1 \leq q_L(\boldsymbol{x}[t+1], \boldsymbol{x}[t]) + \|\boldsymbol{x}[t+1]\|_1 \leq g(\boldsymbol{x}[t]) + \|\boldsymbol{x}[t]\|_1$ となりメジャライザーの性質に変化はないので，逐次的に解く最小化問題を，

$$\boldsymbol{x}[t+1] = \underset{\boldsymbol{x}}{\mathrm{argmin}}\{q_L(\boldsymbol{x}, \boldsymbol{x}[t]) + \|\boldsymbol{x}\|_1\} \tag{1.40}$$

として値を更新していくことで，L_1 ノルムを含んだメジャライザー最小化により，式 (1.23) の最小解を目指す．更新式 (1.40) は，場合分けと平方完成，先に述べた軟判定しきい値関数の式 (1.30) を用いて以下のように解くことができる．

$$\begin{aligned}\boldsymbol{x}[t+1] &= \underset{\boldsymbol{x}}{\mathrm{argmin}} \left\{ \frac{L}{2} \left\{ \boldsymbol{x} - \left(\boldsymbol{x}[t] + \frac{1}{L\lambda} A^{\mathrm{T}}(\boldsymbol{y} - A\boldsymbol{x}[t]) \right) \right\}^2 + \|\boldsymbol{x}\|_1 \right\} \\ &= S_{1/L} \left(\boldsymbol{x}[t] + \frac{1}{L\lambda} A^{\mathrm{T}}(\boldsymbol{y} - A\boldsymbol{x}[t]) \right)\end{aligned} \tag{1.41}$$

最小化の手順としては, 最初に出発点となる $\boldsymbol{x}[0]$ を決定する．その後, 更新式 (1.41) によって, 終了基準を満たすまで更新を続けていく．この手法は Iterative Shrinkage Soft-thresholding Algorithm (ISTA) と呼ばれ, ほかにもより高速な手法である Fast Iterative Shrinkage Soft-thresholding Algorithm (FISTA)[43] や，ISTA の発展である Two-Step Iterative Shrinkage Soft-Thresholding Algorithm (TwIST)[44] などの手法がある．

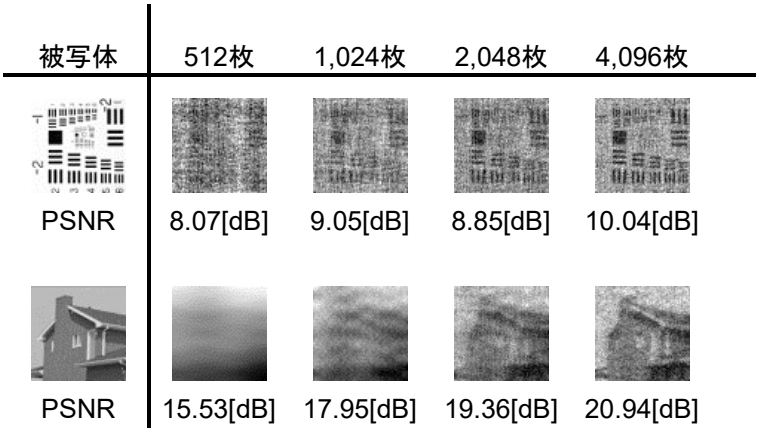


図 1.7: 圧縮センシングでの再構成画像

式 (1.41) に示した方法で再構成した画像を図 1.7 に示す．符号化パターン数は 512 枚，1,024 枚，2,048 枚，4,096 枚，再構成画像サイズは 64×64 [pixel]，$\lambda = 300$，反復回数は 1,000 回，式 (1.20) 中での基底変換には離散コサイン変換を利用した．また，式 (1.3) に示した DGI を $\boldsymbol{x}[0]$ とした．図 1.4 に示した DGI での再構成画像よりも画質が向上していることが確認できる．

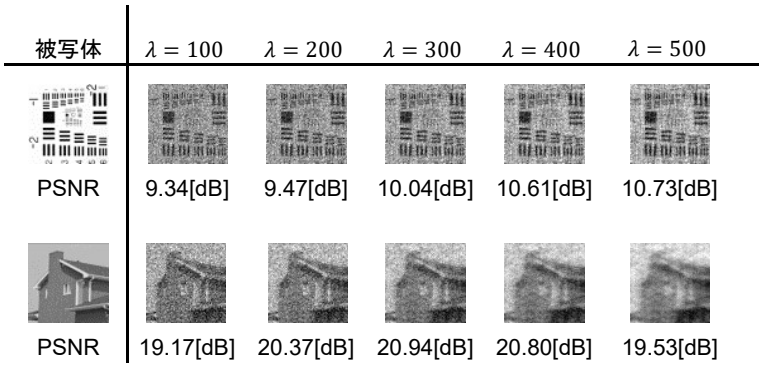


図 1.8: λ を変動させた場合の再構成画像

次に，図 1.8 に λ を変動させた場合での再構成画像を示す．符号化パターンは 4,096 枚とした．λ が増大するにつれて画像がぼやけていく代わりにノイズが減少している．これは λ は正則化項の影響力の強さを表しているため，λ が増大するほど線形変換後の L_1 ノルムの影響が強まった結果である．

圧縮センシングを利用したシングルピクセルイメージングでは，初期値は相関計算や基底に対する逆変換などで求めることが多い．圧縮センシングは，情報が不足している場合に，観測したい本質的な情報を抽出することを目的とした手法なので，これまで述べてきた再構成よりも少ない符号化パターンの投影回数で高画質な画像の再構成が可能である．そのため，正則化項について本論文で紹介した L_1 ノルムを用いた再構成や [45]，全変動を用いた再構成などが提案されてきた [7]．また，圧縮センシングでも被写体に投影する符号化パターンは画質に大きな影響を与えるため，提案された当初 [45] はランダムなパターンを用いていたが，より効率的に被写体の情報を得るための符号化パターンを利用した圧縮センシングの研究も数多く行われている [46, 47]．また，信号測定の過程が線形変換で記述できるものなら，本論文中で紹介した ISTA で基本的には画像再構成できるため，変則的な観測手法にも対応できる [7]．欠点としては，計算量の高い観測課程の線形変換を更新していく中で繰り返し計算する必要があり計算量が非常に多い．そのような計算量の課題があるにもかかわらず，再帰的に値を更新していくので Graphics Processing Unit (GPU) などの並列化で高速化がしづらいため，リアルタイムでの再構成には向かない．

1.5 ディープラーニングを利用したシングルピクセルイメージング

最後に，ディープラーニングを利用したシングルピクセルイメージングについて述べる．ディープラーニングは，人間の神経回路を模した数理モデルであるニューラルネットワークをベースにした手法である．ニューラルネットワークはいくつかの層によって構成されるため，それを深くしたものを一般にディープラーニングと呼称している．このディープラーニングは画像認識や自然言語処理など，様々な分野で優秀な成績を残している．そこでまず，ニュー

ラルネットワークとディープラーニングについて述べた後，ディープラーニングを利用したシングルピクセルイメージングについて述べる．

1.5.1 ニューラルネットワークとディープラーニング

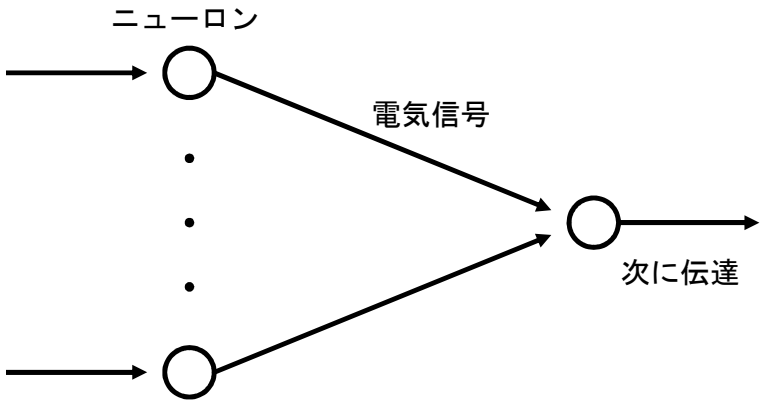


図 1.9: ニューロンの情報伝達

人間の脳は，図 1.9 に示すニューロンと呼ばれる神経細胞によって構成されており，それぞれは電気信号で情報の伝達を行う．あるニューロンが他のニューロンから電気信号を受け取ると，その電気信号がしきい値を超えていればニューロンが発火し，また別の結合しているニューロンに電気信号を渡す．この時，各ニューロンごとの結合度合いはそれぞれ異なるので，ニューロンごとの信号の受け取り方が変化すると最終的な出力も変化する．この変化によって人は様々なパターンを認識しているといわれる．ニューラルネットワークはこのようなニューロンのネットワーク構造を模した数理モデルであり，原型はパーセプトロンという 1950 年代に提案された数理モデルである [48]．パーセプトロンの数理モデルを以下に示す．簡単のため，2 入力 1 出力とした．

$$x_1 w_1 + x_2 w_2 + b = a \tag{1.42}$$

$$y = f(a) \tag{1.43}$$

$$f(a) = \begin{cases} 1 & (a \geq 0) \\ 0 & (a < 0) \end{cases} \tag{1.44}$$

x はニューロンへの入力，w はニューロン間の結合度合いを表すパラメータで重みと呼ばれている．b は発火のしやすさをあらわすパラメータでバイアスと呼ばれ，蓄積信号のしきい値を示している．$f(a)$ はニューロンが発火したかどうかを表す関数で活性化関数と呼ばれている．パーセプトロンが提案された当初は式 (1.44) のようなステップ関数が用いられていた．式 (1.42) から式 (1.44) に示したパーセプトロンの数理モデルの模式図を図 1.10 に示す．上述のような 2 入力 1 出力で解ける問題の例としては，AND ゲート，OR ゲートの演算などが挙

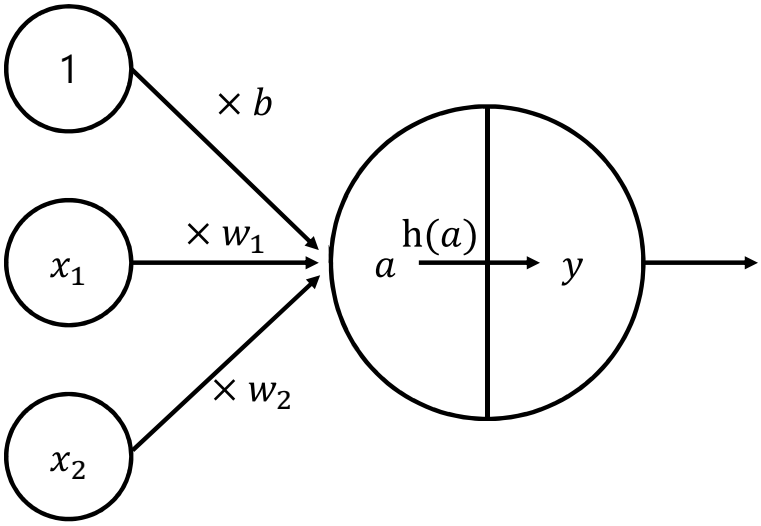


図 1.10: パーセプトロンの模式図

げられる．AND ゲートは入力がそれぞれ 0 と 1 で，両方の入力が 1 の時のみ 1 を出力し，それ以外は 0 を出力するので，例えば $(w_1, w_2, b) = (2, 2, -3)$ とすれば AND ゲートとなる．OR ゲートも入力は AND ゲートと同様で，両方の入力が 0 の時のみ 0 を出力し，それ以外は 1 を出力するので，こちらも例えば $(w_1, w_2, b) = (2, 2, -1)$ とすれば OR ゲートとなる．それぞれのゲートが入力を分離する様子は図 1.11 のようになっている．

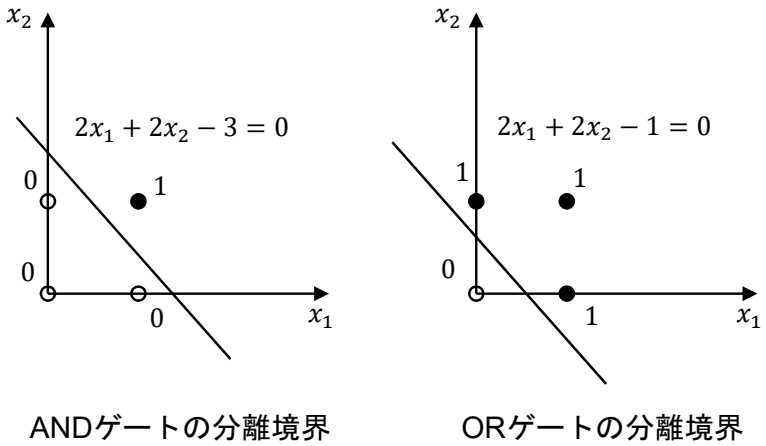


図 1.11: AND，OR ゲートの分離境界

式 (1.42) から式 (1.44) に示したパーセプトロンの数理モデルでは 2 入力 1 出力の場合を示したが，入力や出力が複数個あり，ニューロンの結合が複数になっても基本的には同様に考えることができ，ニューロンへの入力を n 個，出力の数を m 個とすると，パーセプトロンは

以下のようにあらわすことができる．

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{1.45}$$

$$\boldsymbol{y} = f\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right) \tag{1.46}$$

ここまでで示したような，入力が出力に直結しているようなパーセプトロンは特に，単純パーセプトロンと呼ばれている．単純パーセプトロンでは図 1.11 や式 (1.42)，式 (1.45) が線形であることからわかるように，入力の値を線形に分離する事しかできない線形分類器であった．そのため，図 1.12 に示す，異なる値が入力された時のみ 1 を出力する XOR ゲートのような簡単な分類でも，線形で分離できない問題には適用できなかった．これは，単純パーセプトロンを重ねた複数層のネットワークが可能であれば非線形の分類問題も解けるとされていたが，当時は複数層では重みの学習方法などが課題であった．

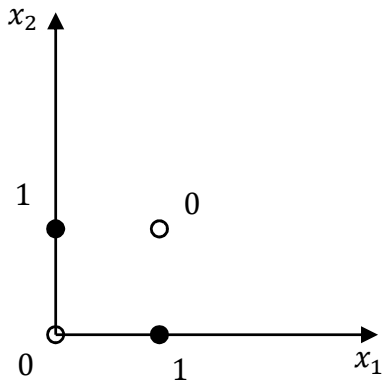


図 1.12: XOR ゲートの入力と出力．線形で分離するには最低でも線が 2 本必要

この課題は長い間解決されず，その期間，ニューラルネットワークの研究はほぼ進展がなかったが，誤差逆伝播法 [49] という微分の連鎖律を利用した，複数層のネットワークでも機械学習を可能にする手法が 1980 年代に考案された．微分の連鎖律と誤差逆伝播法についてここでは簡単に述べる．微分の連鎖律は合成関数を構成するそれぞれの関数の微分の積であらわされる性質のことである．例えば，$t = \sin(x^2 + y^2)$ のような関数があったとして，この関数は次のように二つの関数であらわすことができる．

$$\begin{aligned} t &= \sin(z) \\ z &= x^2 + y^2 \end{aligned} \tag{1.47}$$

この式の x についての微分 $\frac{\partial t}{\partial x}$ は先ほどの定義に従えば次のように求めることができる．

$$\begin{aligned}\frac{\partial t}{\partial x} &= \frac{\partial t}{\partial z}\frac{\partial z}{\partial x} \\ \frac{\partial t}{\partial z} &= \cos(t) \\ \frac{\partial z}{\partial x} &= 2x\end{aligned} \tag{1.48}$$

結果として，微分の積から以下のように合成関数の微分が得られる．

$$\frac{\partial t}{\partial x} = 2x\cos(t) \tag{1.49}$$

誤差逆伝播法ではこの連鎖律を利用して，所望の出力と実際の出力との誤差に対する各パラメータの微分値，つまり勾配を取得する．その勾配によってパラメータを更新していくことで，誤差が減少する方向にパラメータを学習していく．例として，先ほど式 (1.45) と式 (1.46) に示した多入力の単純パーセプトロンに誤差逆伝播法を利用した場合を示す．まず，式 (1.45) と式 (1.46) を以下のような合成関数として分解する．

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \tag{1.50}$$

$$\boldsymbol{a} = \boldsymbol{h} + \boldsymbol{b} \tag{1.51}$$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{a}) \tag{1.52}$$

$$E = E(\boldsymbol{y}, \boldsymbol{t}) \tag{1.53}$$

$E(\boldsymbol{y}, \boldsymbol{t})$ は損失関数と呼ばれる関数で，$\boldsymbol{t}$ は一般に教師データと呼ばれる所望の出力値である．微分の連鎖律によって，それぞれのパラメータ $\boldsymbol{W}$ と $\boldsymbol{b}$ を更新する勾配と学習は以下のようにあらわすことができる．

$$\frac{\partial E}{\partial \boldsymbol{W}} = \frac{\partial E}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{W}} \tag{1.54}$$

$$\boldsymbol{W}_{\text{new}} = \boldsymbol{W} - \eta\frac{\partial E}{\partial \boldsymbol{W}} \tag{1.55}$$

$$\frac{\partial E}{\partial \boldsymbol{b}} = \frac{\partial E}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} \tag{1.56}$$

$$\boldsymbol{b}_{\text{new}} = \boldsymbol{b} - \eta\frac{\partial E}{\partial \boldsymbol{b}} \tag{1.57}$$

η は更新式 (1.25) と同様の意味を持ち，ニューラルネットワークやディープラーニングでは特に学習率と呼ばれている．このように勾配を求める計算が，図 1.13 のように誤差が計算を逆方向に伝播していくようにみえるため誤差逆伝播法と呼ばれている．また，図 1.13 からわかるように，入力に伝播する誤差も同様に求められるので，入力 $\boldsymbol{x}$ が別の層からの入力であるような層が複数層ある場合でも同様に勾配が求められる．また，誤差逆伝播法では微分を利用して学習を行う都合上，式 (1.44) で示したようなステップ関数は利用できない．そのため，

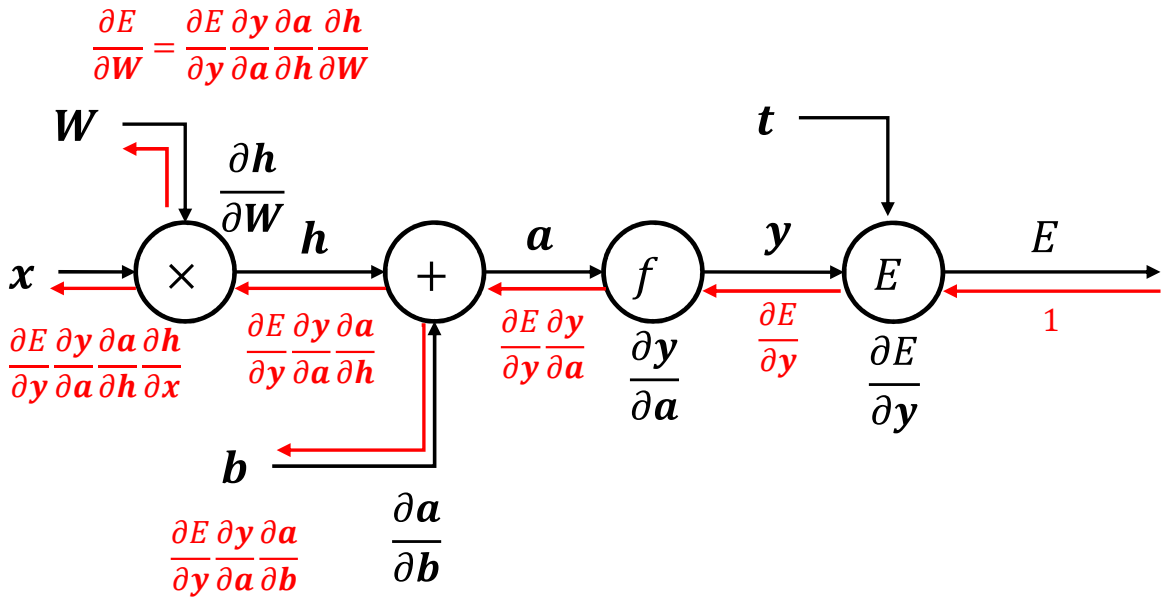


図 1.13: 式 (1.50) から式 (1.53) の計算に対する誤差が逆伝播する様子．表記には計算グラフを用いた [50]

ロジスティック関数，あるいはロジスティックシグモイド関数と呼ばれる関数が誤差逆伝播法では利用されていた．ロジスティックシグモイド関数の式を以下に示し，関数の外形を図 1.14 に示す．

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{1.58}$$

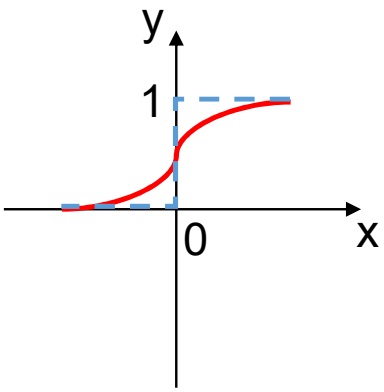


図 1.14: ロジスティックシグモイド関数の概形

誤差逆伝播法の利点として，局所的な微分計算とそれらの積から勾配を求められるので，数値微分で直接勾配を求めるよりも計算量を抑えることができる．この利点により，誤差逆伝播法は数値微分による直接計算では難しかった複数層パラメータの機械学習を可能にした．こ

れによって，単純パーセプトロンを複数層重ねた多層パーセプトロンと呼ばれるニューラルネットワークが可能になった．しかし，実現できたのは図 1.15 に示すような 2 層程度の浅いネットワークのみで，これ以上の深さを持つディープなネットワークは，出力から離れた層ほど誤差が伝播するにつれて局所的な微分の積を何度も乗算されることが原因で勾配が急速に小さくなってしまう勾配消失などの課題があり依然として学習は困難であった．

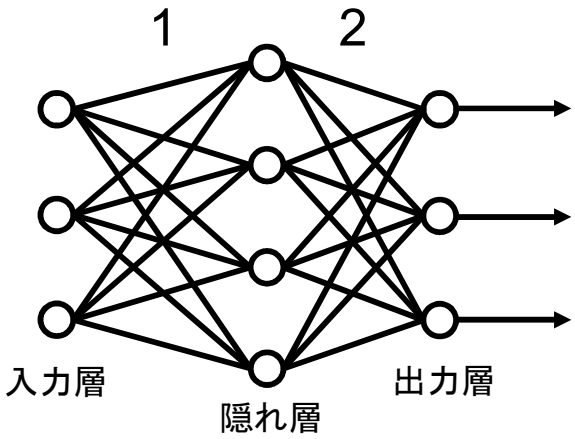


図 1.15: 多層パーセプトロン

しかし，2000 年代になり入力と出力が同じになるような自己符号化器などで，複数層を単層に分解し学習したパラメータを初期値とすることで深層のネットワークを学習可能であることが示された [51–53]．そして，2012 年に多層のディープニューラルネットワークが画像認識の分野で高い成績を残し [54]，そのようなディープニューラルネットワークがディープラーニングとして認知されるようになった．

ディープラーニングでは様々なネットワークが考案されており [54–58]，その多くは式 (1.46) で示したパーセプトロンからなる層と，畳み込み層から構成されている．パーセプトロンからなる層は，すべてのニューロン間で結合があるため，全結合層と呼ばれている．畳み込みニューラルネットワーク（CNN: Convolutional Neural Network）は，神経細胞の中でも特に人の視神経細胞における単純型細胞，複雑型細胞の働き [59] に着目したニューラルネットワークである．単純型細胞，複雑型細胞共に特定のパターンに反応する性質を持つが，単純型細胞はパターンの位置に敏感に反応するのに対して，複雑型細胞は入力パターンが多少ずれても反応する．図 1.16 にそれぞれの細胞モデルの例を示す．

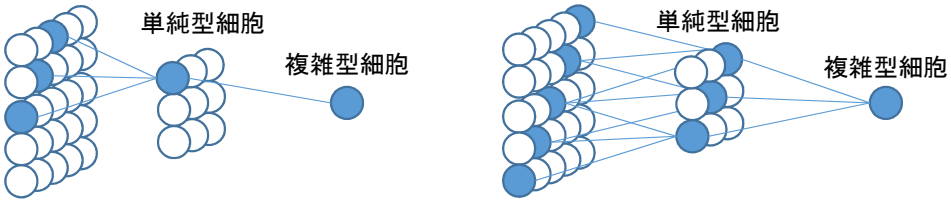


図 1.16: 単純型細胞と複雑型細胞の働きの違い

畳み込みニューラルネットワークは，主に畳み込み層とプーリング層と呼ばれる層から構成され，畳み込み層は単純型細胞の働きを模して，入力画像に対する明暗の境界や濃淡をフィルタを用いた畳み込み演算によって検出し，プーリング層は複雑型細胞の働きを模して，入力画像の位置ずれを吸収する働きを持つ層である．図 1.17(a) に畳み込みニューラルネットワークで用いる畳み込み演算の一例を示す．$*$ は畳み込み演算を表す記号である．畳み込み演算では，フィルタを入力画像上で，一定間隔でスライドさせ対応する要素同士を乗算後それらの和を求めて出力とする．畳み込み演算は，フィルタの種類によって様々な画像処理が行える．図 1.17(b) では，フィルタ畳み込みによってエッジ抽出を行っている例を示している．畳み込みニューラルネットワークではこのフィルタを学習することで入力画像の様々な濃淡のパターンなどを検出することができるようになる．また，畳み込みニューラルネットワークでは，出力画像は特徴マップなどと呼ばれている．

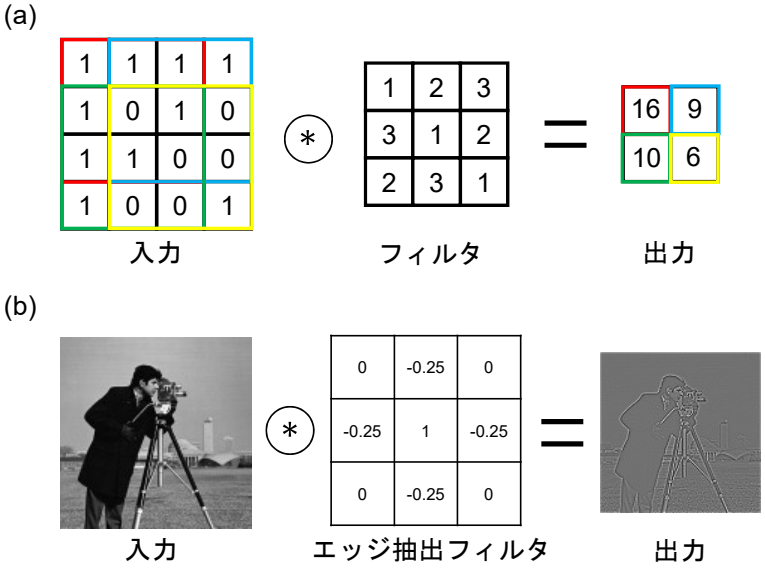


図 1.17: (a) 畳み込み演算，(b) フィルタ畳み込みによるエッジ抽出の例

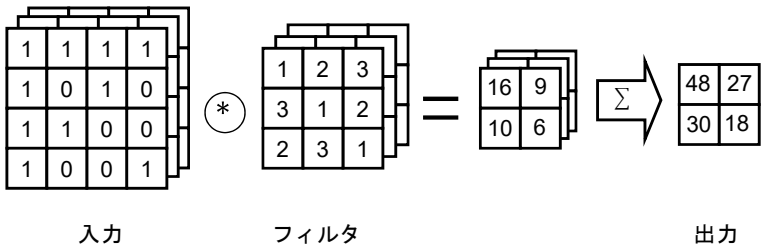


図 1.18: 3 次元データの畳み込み演算

図 1.17(a) では二次元データに対する畳み込みを示したが，画像の RGB ビットプレーンのような，複数の入力がひとまとまりでひとつの入力となる 3 次元データを畳み込む場合は図 1.18

のように入力と同じ数のフィルタを用意する．それぞれの入力とフィルタから得られた出力を対応している要素ごとにすべて足し合わせてひとつの出力を得る．この入力特徴マップの枚数はチャネル数と呼ばれている．畳み込みニューラルネットワークではこのように，チャネル数でひとまとめにしたフィルタをひとつのブロックとして，これを図 1.19 のように複数個用意し，用意したフィルタの数だけ特徴マップを出力する．

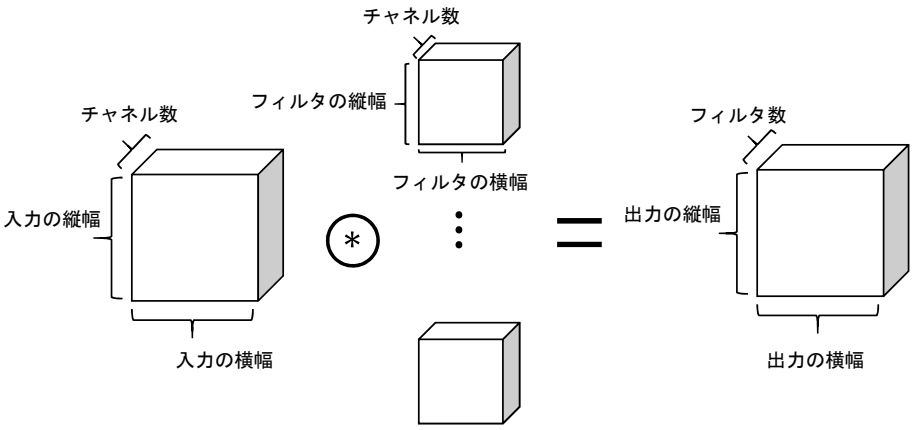


図 1.19: 複数フィルタでの畳み込み演算

プーリング層では，畳み込み層で得られた特徴マップに対して，サイズを縮小させる操作を行う．プーリング層の一例としてマックスプーリングと呼ばれている処理を図 1.20 に示す．マックスプーリングでは畳み込み演算のフィルタのようにウインドウを用意し，スライドさせていく．図 1.20 ではウインドウサイズは 2×2，スライド幅は 2 としている．その後，各ウインドウの中から最大値を抽出しそれを出力値とする．入力特徴マップのサイズが縮小されて出力特徴マップとなっている．

これらの層からなる畳み込みニューラルネットワークは，特に画像分野でよく利用されている．そのため，ディープラーニングによるシングルピクセルイメージング画像再構成にも非常によく利用されている．

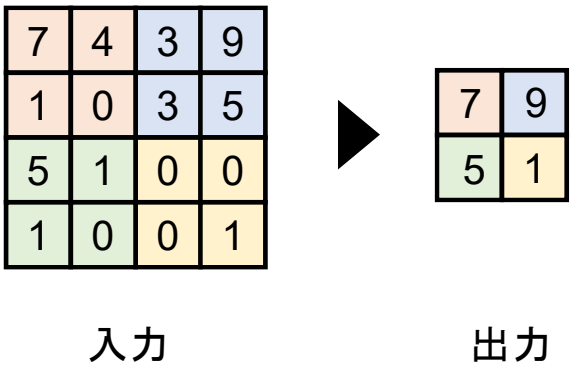


図 1.20: マックスプーリングの処理

1.5.2 ディープラーニングによるシングルピクセルイメージング画像再構成

シングルピクセルイメージングでディープラーニングを利用した研究としては，図 1.21 のように 1.2 節などで述べた相関計算によって再構成した画像を入力として，より高画質な画像を出力するような方法が提案された [60, 61].

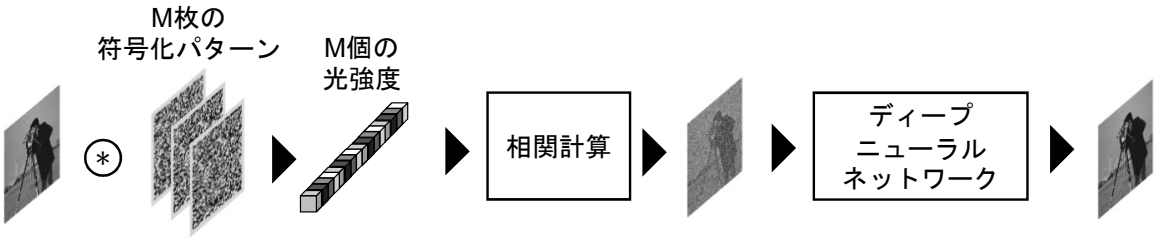


図 1.21: ディープラーニングを利用したシングルピクセルイメージングの一例

その後，図 1.22 のような相関計算など他の再構成手法を介さず，取得した光強度から直接画像を再構成する end-to-end な手法が提案された [62]．この手法では，学習段階では被写体を入力として，出力も被写体と同様のものを出力するオートエンコーダ構造の教師なし学習となっている．このように学習された後，推論を行う場合は，図 1.22(b) のように前半のエンコード部分を符号化パターンとして被写体に投影し取得した光強度を入力として画像を再構成する．その際，符号化パターンは高速投影のためにバイナリパターンにする必要があるが，パラメータを単純にバイナリに制限すると学習がうまく進まない．そのためこの手法では，最初はバイナリに制限せず普通に学習を進め，学習が終了したら符号化パターンに以下のような制約項を課し再学習することでバイナリ化を行っている．

$$\sum_{i=1}^{N}\sum_{y=1}^{Y}\sum_{x=1}^{X}\left(1+I_i(x,y)\right)^2\left(1-I_i(x,y)\right)^2\right) \tag{1.59}$$

N は符号化パターンの枚数，X，Y はそれぞれ符号化パターンのサイズである．二回目の学習ではこの制約を最小化するように符号化パターンが学習されるので，$+1$ と -1 の 2 値に収束しバイナリパターンが得られる．-1 は光学的にはあらわすことができないので，基底変換の時と同様反転したパターンで測定した光強度を反転前の符号化パターンで測定した光強度から減算することで -1 を表現する．このように学習を行うことで符号化パターンも含めて最適化するため高画質な画像再構成が期待できる．現在では，実際の光学系で取得した環境的な外乱を含んだデータで再学習させることで，より再構成画像の画質を向上させる手法なども提案されている [63].

図 1.23 に図 1.22 で示したネットワークで実際に再構成した画像を示す．符号化パターン数は 512 枚，1,024 枚，2,048 枚，4,096 枚，再構成画像サイズは 64×64 [pixel]，ネットワークを学習するためのデータセットには STL-10[64] を 64×64 [pixel] に縮小して用いた．STL-10 は 10 種類の動物や乗り物などの画像を集めたデータセットである．訓練データ数は $90,000$，評価データ数は $10,000$ で学習を行った．最適化アルゴリズムには Adaptive Moment Estimation

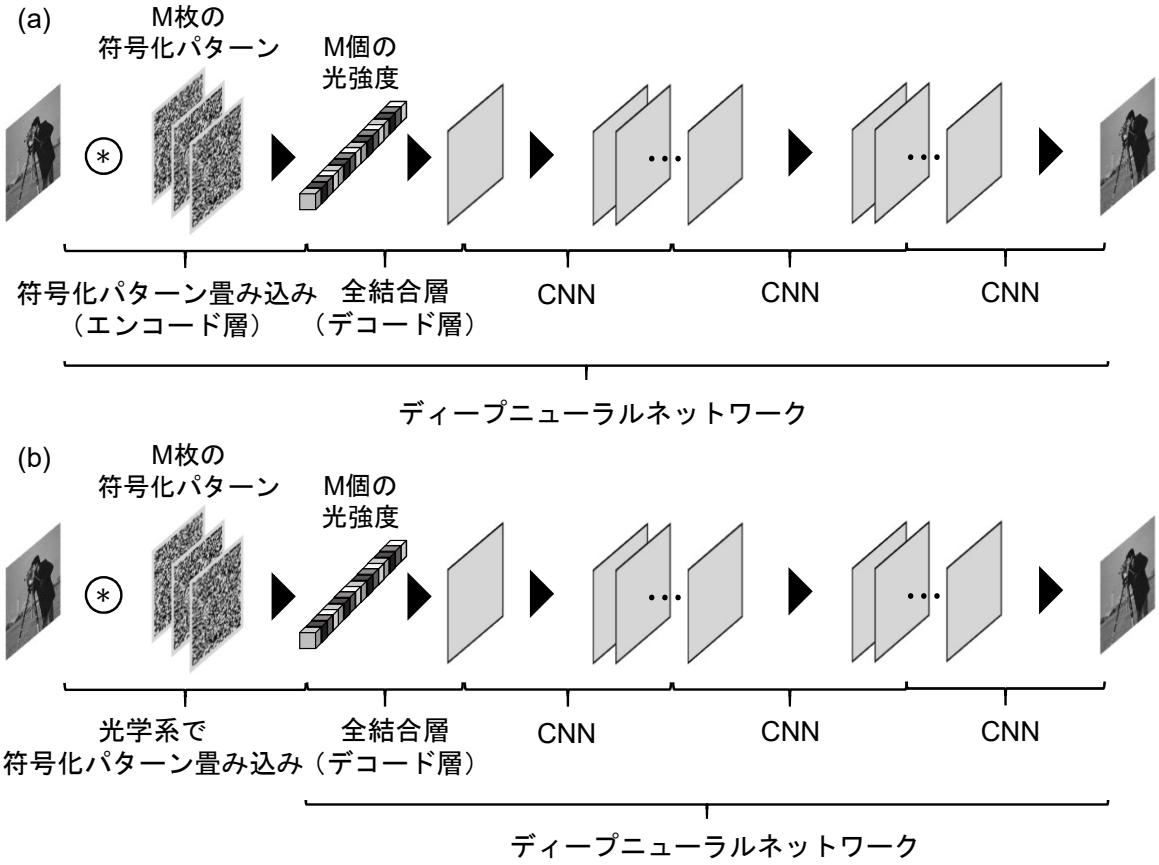


図 1.22: ディープラーニングを利用した end-to-end のシングルピクセルイメージング画像再構成．∗ は畳み込み演算を表す．

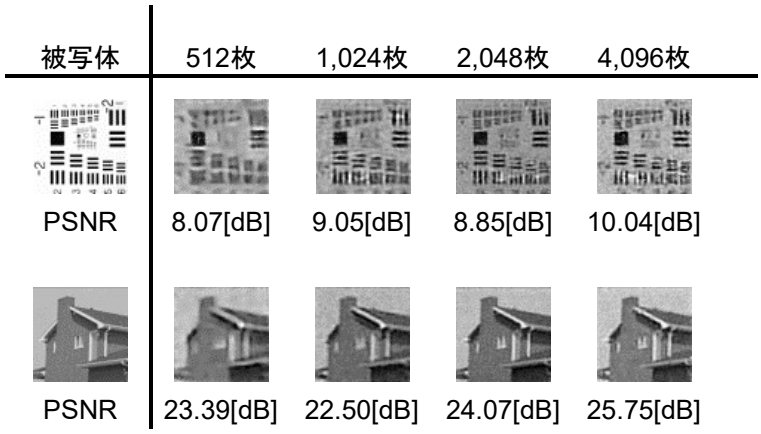


図 1.23: ディープラーニングでの再構成画像

(Adam)[65] を用いた．これまで図 1.4，1.6，1.7 に示してきた各手法と比較して，特に符号化パターン数が少ない場合での画質に優れていることが確認できる．

ディープラーニングを利用したシングルピクセルイメージング画像再構成の特徴として，圧縮センシングと同等かそれ以上の画質を実現できる点が挙げられる．また，圧縮センシングと同様に計算量は多いが，圧縮センシングとは異なり利用する演算は基本的に行列積の計算，もしくは畳み込み演算なので GPU や FPGA などの並列計算を得意とするデバイスを利用することで圧縮センシングよりも高速な計算が可能である．ほかには，google が開発するオープンソースのライブラリである Tensorflow[66] やそのラッパーである keras[67] などにより誰でも簡単に実装することができる．欠点は，学習にかかる計算量が推論による画像再構成にかかる計算量よりも多く，大きなサイズの画像を再構成できるモデルを得るのが難しい．また，パラメータ数が多くネットワーク構造によってはメモリを大量に消費する欠点がある．そのため，FPGA などリソースが限られている場合は画像サイズによっては実装が難しくなることがある．これらの欠点は，GPU などを用いる場合は大きな問題になることはないので，2022 年現在，シングルピクセルイメージング分野では GPU を用いた高速，高画質なディープラーニングでの研究が主流となっている．

第2章　小規模FPGAを用いたシングルピクセルイメージング専用計算回路

これまで述べてきたように，シングルピクセルイメージングは画像再構成にPCなどの計算機を要する．この際，利用する計算機は組み込み向けCPUなどを利用した小型なものが望ましいが，計算性能が足りないため大型なものを使わざるをえず，システム全体が大型化してしまう．そこで本研究では，小規模なFPGAボードを利用して小型かつ高性能なシングルピクセルイメージング画像再構成計算回路の設計，実装を行った．結果として，相関計算を実装した計算回路で，32×32 [pixel] の画像をCPUよりも高速な再構成に成功した．本章では，FPGAについて簡単に述べた後，実装したシングルピクセルイメージング画像再構成計算回路についてその構成を示し，再構成速度と再構成画像，光学実験の結果についてまとめる．

2.1　Field-Programmable Gate Array

FPGAは内部の論理回路構造を任意の構造に書き換え，再構成が可能なLSIである．クロック周波数はCPUよりも低いが，演算器を大量に並べて並列計算を行えるためハードウェアアクセラレータとしてよく用いられる [17–20]．FPGAが登場する以前は，ユーザーが設計したカスタムLSIの実装にはApplication Specific Integrated Circuit (ASIC) が使用されていた．ASICはセミカスタムの特定用途向け専用LSIで，省スペース，大量生産時の単価が安い，高性能であるといった利点があるが，一度回路の書き込みを行った後は書き換えができず，結果としてで莫大な開発費用と開発期間が必要である欠点があった．そのような欠点があり，ASICは個人レベルでの設計や少量生産に利用できるものではなかった．FPGAは登場してからしばらくは，その性能の低さからASIC実装前の試作や数十台程度の少量生産LSIとしてしか利用されていなかった．その後，FPGAの性能向上に伴って現在では低コストかつ手軽なLSIとして，前述の並列計算による高速計算はもちろん，データセンター [68] など様々な用途で利用されている．図2.1にFPGA内部構造の概略を示す．FPGAはいくつかの基本的な回路リソースをひとまとめにした論理ブロックにグループ分けされており，論理ブロック間は配線領域となっている．論理ブロックには，Look-Up Table，Multiplexer (MUX)，Flip Flop (FF)，論理ゲートなどが決まった数搭載されている．LUTはすべての入力の組み合わせに対応した出力の定義を記憶しそれを出力する．FFは入力の状態を記憶する素子で，1bit分の情報を記憶する．複数bitを記憶するレジスタなどはこのFFを複数個束ねることで実装される．MUXは二つ以上の入力を制御信号により選択し出力する素子である．論理ブロックに搭載されている回路リソースはFPGAごとに異なるため，図2.1ではXilinx Artix-7 XC7A100T-2CSG324C

の論理ブロックに搭載されている回路リソースを示している．Artix-7XC7A100T-2CSG324C は本研究で計算回路の実装に用いた，特殊電子回路社が提供している FPGA 評価ボードに搭載されている FPGA であり，図 2.2 にそのボードの概観を示す．

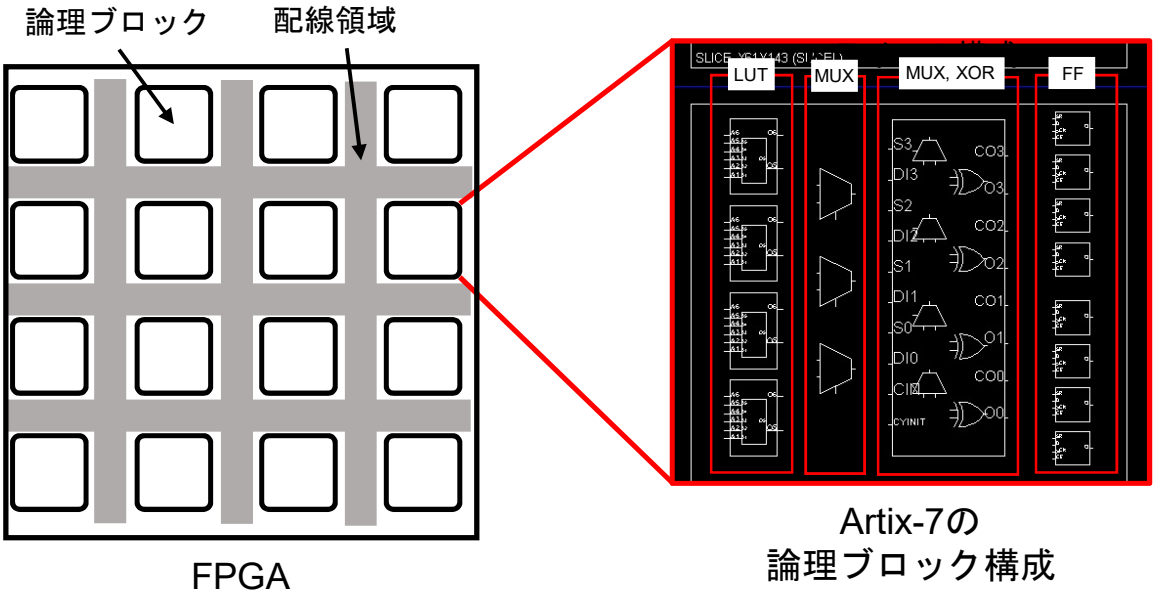


図 2.1: FPGA 内部構造の概略と論理ブロック

FPGA には，以上で述べたように論理ゲートや LUT，FF が多数搭載されており，それらを用いて所望の回路を設計することができる．このような基本回路リソースとは別に，FPGA には図 2.3 に示すような Digital Signal Processor (DSP) や Block Random Access Memory (Block RAM)，PCI Express など特定の機能のために最初から組み込まれている回路が存在する．これらはハードマクロと呼ばれていて，用途ごとに最適化されているため論理ゲートで同様のものを実装するよりも優れた性能や実装面積を誇る．例として，多くの FPGA でメモリ専用の用途として搭載されている Block RAM を利用することで，LUT や FF などでメモリを構成するよりも多くのメモリ領域を確保することができる．ハードマクロに対して，Intellectual Property (IP) コアと呼ばれるものは一般にソフトマクロと呼ばれ，LUT や FF などの回路リソースを消費し特定の機能を持ったモジュールを実装したり，ハードマクロのコンフィグレーション用途などで利用されている．

表 2.1 にハードマクロを含めた Artix-7 評価ボードに搭載されている回路リソースを示す．Block RAM は先に述べたようにメモリ機能に特化したハードマクロであり，制御信号とアドレス信号にアクセスすることでデータを書き込んだり読み込むことができる．36Kb を一つのブロックとして組み込まれているが，二つの 18Kb ブロックとして使用することもできる．DSP は乗算や加減算，積和演算などの信号処理を高速に行えるハードマクロであり，FPGA では主に乗算で使用されることが多い．

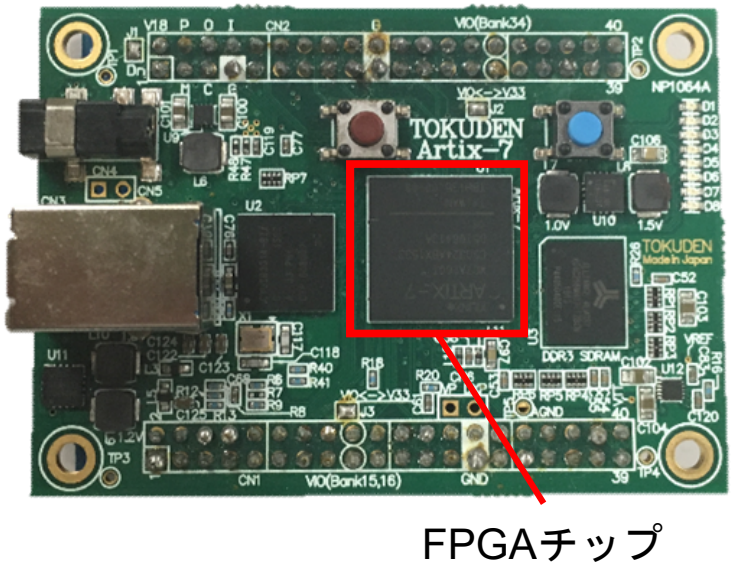


図 2.2: 計算回路実装に用いた Artix-7 評価ボード（特殊電子回路社）．サイズは 72mm × 50mm で名刺と同程度

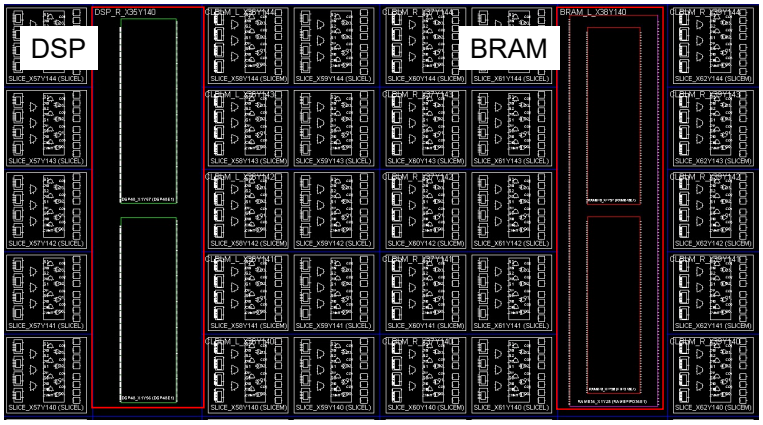


図 2.3: 論理ブロックの間に搭載されているハードマクロ

表 2.1: Artix-7 評価ボードの回路リソース

リソース	使用可能リソース数
LUT	63,400
FF	126,800
Block RAM(36Kb)	135
DSP	240

2.2 計算回路全体の構成

本節では開発した計算回路全体の構成について述べる．最初に回路全体の概略について述べた後，並列計算を行う計算部分について述べ，最後にその他の説明が必要な回路について述べる．

2.2.1 回路の概略

図 2.4 に開発したシングルピクセルイメージング画像再構成計算回路の概略を示す．使用した Artix-7 評価ボードは Universal Serial Bus (USB) 3.0 ポートで PC と通信を行うことができる．制御用の PC から光強度を送信し，計算回路で画像を再構成後 PC に値を返す構成となっている．USB3.0 ポートから値を受け取る通信回路，および PC 側の API は特殊電子回路社が評価ボードと共に提供しているものを利用した．

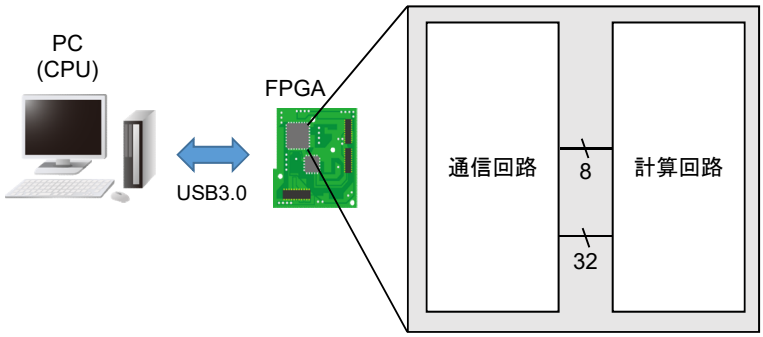


図 2.4: 回路全体の構成

2.2.2 計算回路

計算回路のブロック図を図 2.5 に示す．再構成画像のサイズは 32×32 [pixel] として，符号化パターンは 0 と 1 からなるバイナリパターンを用い，バイナリパターン数は $16,384$ 枚として設計を行った．また，小数の扱いは固定小数点数とした．再構成計算は，式 (1.3) で示したものを実装した．FPGA は四則演算の中でも，消費するリソースやレイテンシの問題から除算が最も苦手である．そこで計算回路として実装する際は，$\langle R_i \rangle$ が事前に用意した符号化パターンから決まる定数であることを考慮して，以下のように再構成計算式を変形した．

$$\langle R_i \rangle O(x,y) = \langle R_i \rangle \langle S_i I_i(x,y) \rangle - \langle S_i \rangle \langle R_i I_i(x,y) \rangle \tag{2.1}$$

このように式変形を行うことで，除算なしに再構成画像を計算できる．再構成計算の結果は画像化の際に 256 階調化するため，定数倍しても結果は変わらない．

計算回路では，式 (2.1) で示した計算のうち $\langle S_i I_i(x,y) \rangle$ および $\langle S_i \rangle$，$O(x,y)$ の計算を行う．$\langle R_i I_i(x,y) \rangle$ と $\langle R_i \rangle$ は符号化パターンが決定すれば事前に計算できるので，テーブルとして

実装した．計算量のオーダーは画像の縦幅，横幅を X，Y として，符号化パターンの枚数が N 枚の時，それぞれ $O(X \times Y \times N)$，$O(N)$，$O(X \times Y)$ となっており，最も計算量の多い $\langle S_i I_i(x,y) \rangle$ は計算モジュールを 64 個用意し並列に計算している．計算回路は，はじめに入力として光強度を受け取り，$\langle S_i I_i(x,y) \rangle$ と $\langle S_i \rangle$ を計算する．計算が終了したら，計算した値と事前計算したテーブルから式 (2.1) を計算する．その後，計算した値は通信回路に渡され，PC に送信された後に 256 階調化される．

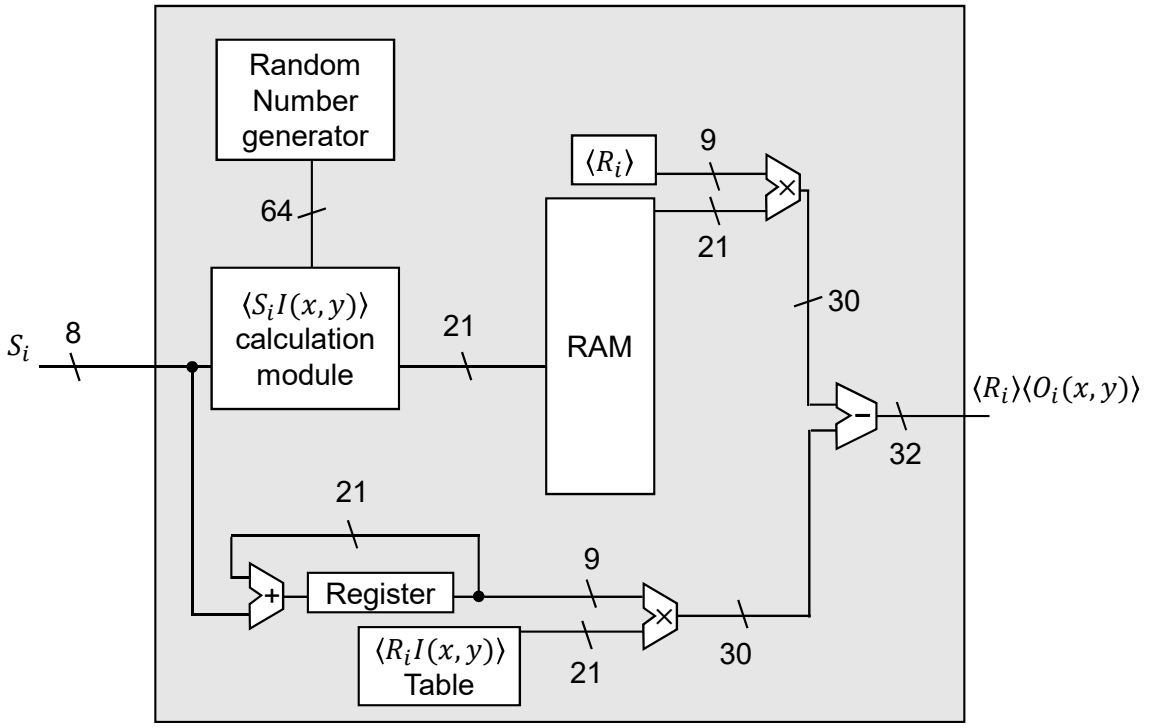


図 2.5: 計算回路の構成

図 2.5 中の $\langle S_i I_i(x,y) \rangle$ 計算モジュールについての詳細を図 2.6 に，並列計算に対する各画素の割り当てを図 2.7 に示す．リソースの制限から 32×32 個の画素全てに計算モジュールを用意することは困難なため，モジュールを 64 個用意し，図 2.7 のように計算モジュールを割り当てている．これ以降の列でも同様に割り当てているため，64 並列に並べた各計算モジュールはそれぞれ複数の画素を扱う必要があるため，内部に RAM を用意することで複数の値を記憶している．計算モジュールでは，取得した光強度と符号化パターンの 1 画素分が入力となっている．符号化パターンの分布はバイナリ値のため，乗算器を使わず AND ゲートで積をとった後，同じ画素の値を RAM から読み出し，足し合わせた値を RAM に格納することで平均を計算している．全ての画素の計算が終了したらマルチプレクサでパラレルシリアル変換を行うことで値を 1 つずつ読み出し次の回路に渡している．

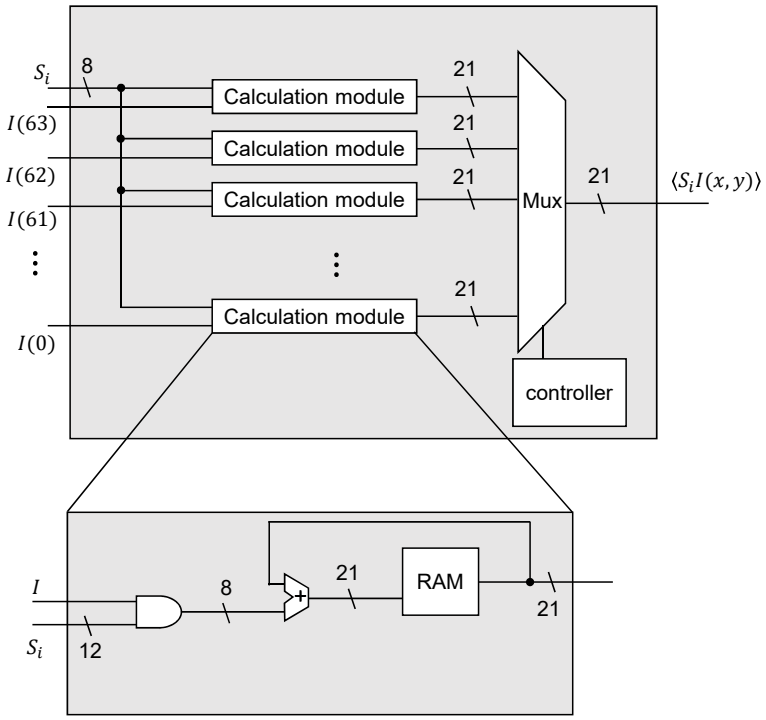


図 2.6: $\langle S_i I_i(x,y)\rangle$ 並列計算回路の構成

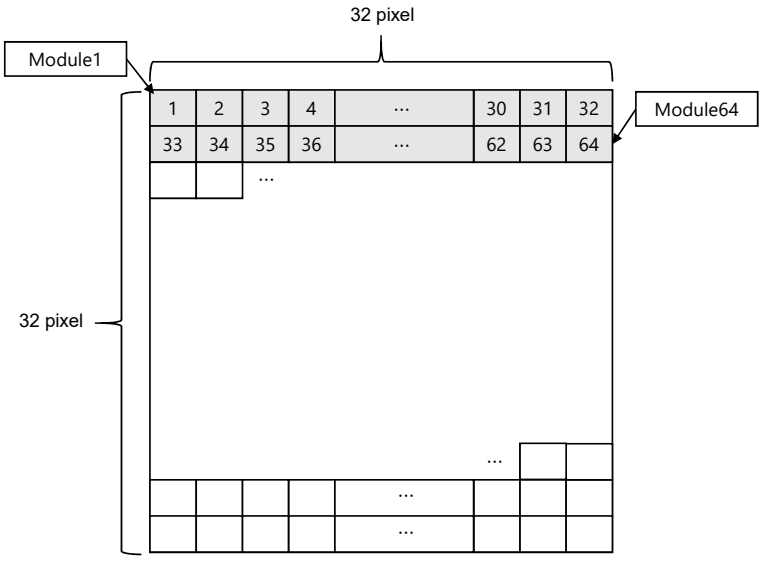


図 2.7: $\langle S_i I_i(x,y)\rangle$ 並列計算回路回路に対する画素の割り当て

2.2.3 その他の回路

計算回路では，$\langle S_i I_i(x,y) \rangle$ を計算するため，計算回路内に被写体に投影した符号化パターンの情報を保持しておく必要がある．本研究では乱数によって生成したランダムな符号化パターンを用いていたため，符号化パターンの情報をメモリに保持するのではなく，計算回路内で同様のパターンを生成している．符号化パターンを生成する回路を図 2.8 に示す．(a) は線形帰還シフトレジスタと呼ばれる回路で，シフトレジスタと XOR ゲートのみで乱数列を生成できるため回路実装に適している．通常のシフトレジスタの入力に XOR ゲートの出力を接続することで再帰的に乱数列を生成する．この時，XOR ゲートに入力するレジスタを特定のルールにそって決定することで，シフトレジスタの bit 数を n としたとき乱数の周期が 2^{n-1} になる．このような乱数列は M 系列と呼ばれている．出力した乱数は並列計算で使用するため，並列数分だけシフトできなければ実際に使用した符号化パターンとずれが生じてしまう．そのため，(b) のように並列数分だけシフトするよう設計している．

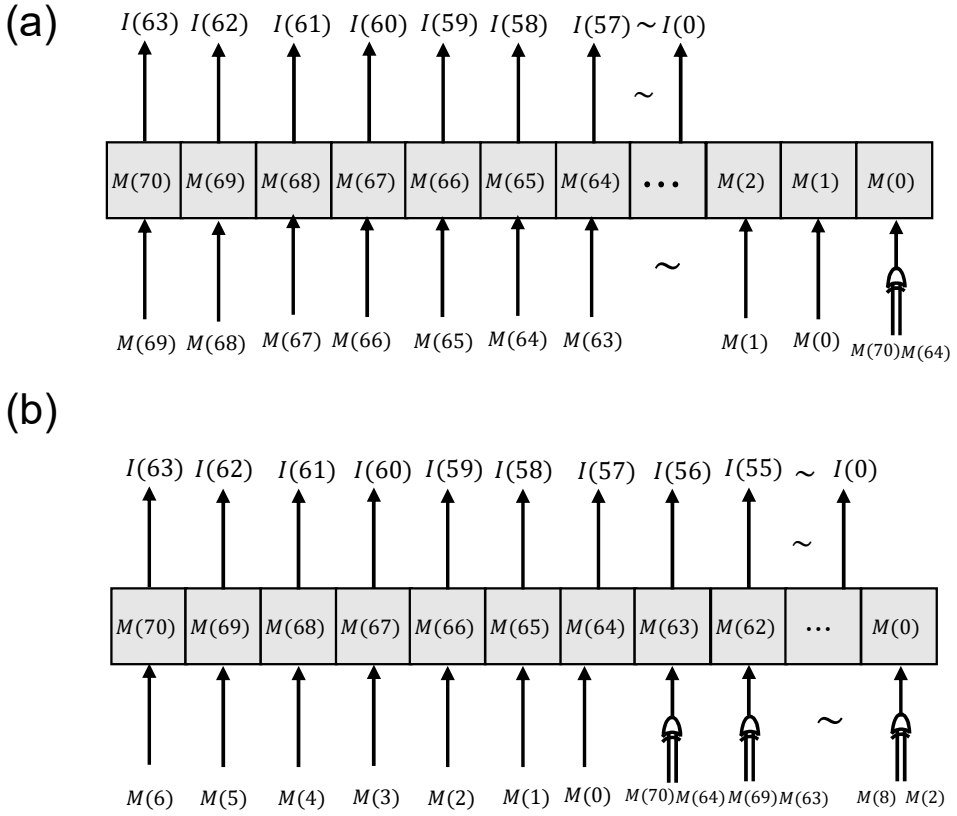


図 2.8: 並列出力乱数生成回路

2.2.4 リソース使用率

図 2.5 に示した回路を構成するのに使用した回路リソースを表 2.2 に示す．LUTRAM は LUT を利用して構成される RAM であり，分散 RAM とも呼ばれる．Block RAM より容量が少ないが，使用時の制約が Block RAM よりも少ない．LUT はメモリ以外にも使用されるため基本的には Block RAM の使用が推奨されている．パターンは保持せず乱数生成回路より出力しているため，DSP 以外の回路リソースは再構成画像の画素数に比例すると考えらえる．そのため，本研究で用いた小規模なボードでも 64×64 [pixel] 程度までの画像再構成回路なら実装できることがわかる．DSP ブロックは主に乗算で使用される回路リソースであり，図 2.5, 2.6 でも示したように，乗算器を用いた乗算を行っているのは最終段で $O(x, y)$ を計算する部分のみであるため使用リソースが非常に少なくなっている．

表 2.2: 計算回路全体のリソース使用率

リソース	使用リソース数	使用可能リソース数	リソース使用率 [%]
LUT	10,639	63,400	16.78
LUT RAM (64b)	1,444	19,000	7.60
FF	22,117	126,800	17.44
Block RAM(36Kb)	5	135	3.70
DSP	2	240	0.83

2.3 シミュレーションと光学実験

Artix-7 評価ボードを用いて開発した計算回路を用いて，画像シミュレーションによる再構成速度と再構成画像，実際の光学系による光学実験を行った．本節ではそれらの結果を示す．

2.3.1 再構成速度の評価

開発した計算回路の計算速度と CPU での計算速度を表 2.3 に示す．CPU 側の計算実行環境は，CPU は Intel® Core™ i5-4690 (3.5GHz)，メモリは 32.0GB，Operating System (OS) は Windows 10 Education，コンパイラは Visual Studio 2015 を使用した．最適化オプションや拡張命令セットは使用していない．高速化率は，CPU の計算時間を基準に計算回路が何倍高速に計算できるかを示している．表 2.3 から CPU の 15 倍程度高速な計算が可能であることが確認できた．

2.3.2 再構成画像の評価

計算回路では小数を固定小数点数で表現している．この固定小数点数は，信号全体の bit 幅はもちろん，小数部分の bit 幅などの取り方が結果に大きく影響する．そこで再構成が正常に

表 2.3: 計算回路の計算速度

プロセッサ	計算時間 [ms]	高速化率
CPU	45	1.0
計算回路	3	15.0

行えているかどうか，浮動小数点数で計算した場合よりも画質が低下していないかを確認するため，図 2.9 に CPU，計算回路それぞれで画像を再構成した結果を示す．結果として，開発した計算回路は CPU で浮動小数点数を用いて再構成した画像と比較しても画質の低下などはほぼ見られず，元画像を再構成可能なことが確認できた．

図 2.9: 再構成画像の比較

2.3.3 光学実験

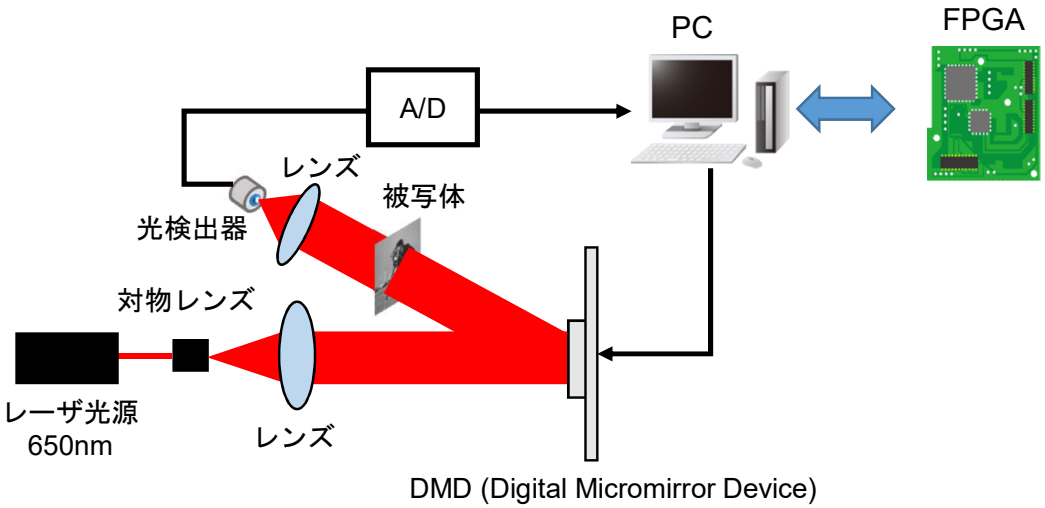
図 2.10(a) に構築した光学系の概略図を，(b) に実際の光学系を示す．DMD デバイスにはライトソースを外した DLP ®LightCrafter ™4500(Texas Instruments 社)，光検出器には，PDA10A (Thorlabs 社)，アナログデジタル変換器には FAD-16H シリーズ (ELMOS 社)，レーザ光源は 650nm の赤色レーザを用いた．レーザ光源から照射された光は DMD 上に表示した符号化パターンにより変調され，被写体を通過してレンズで 1 点に集光され光強度として測定する．測定された光はアナログデジタル変換器を通して PC に取り込まれる．用意したすべての符号化パターンで測定が終了次第，光強度を PC から FPGA の計算回路に送信し，画像再構成の結果を受け取る．

以上の光学系で光学実験を行った結果を図 2.11 に示す．被写体は 5mm 四方程度のアルファベットの C，矩形窓，スリットの 3 種類を撮影した．再構成の結果から，実際の光学系で測定した光強度からでも問題なく画像再構成できることが確認できた．

2.4 小括

本章では，開発したシングルピクセルイメージング画像再構成計算回路の回路構成と，再構成画像，計算速度などについて述べた．結果として，開発したシングルピクセルイメージン

(a)



(b)

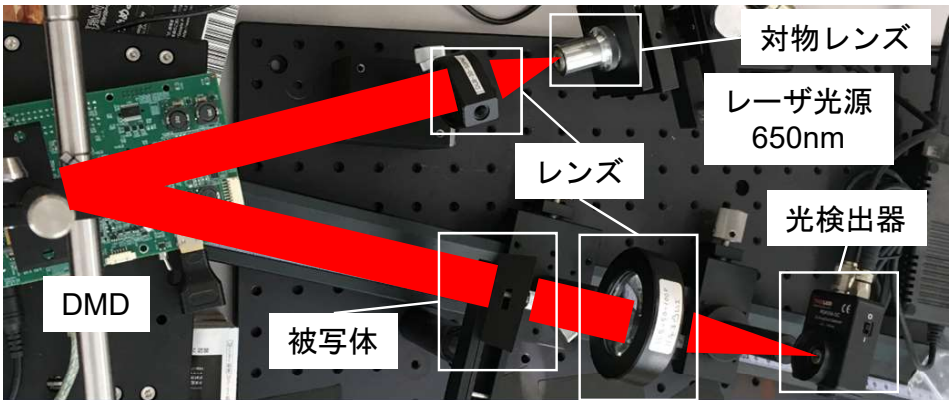


図 2.10: (a) 光学実験の光学系，(b) 実際に構築した光学系

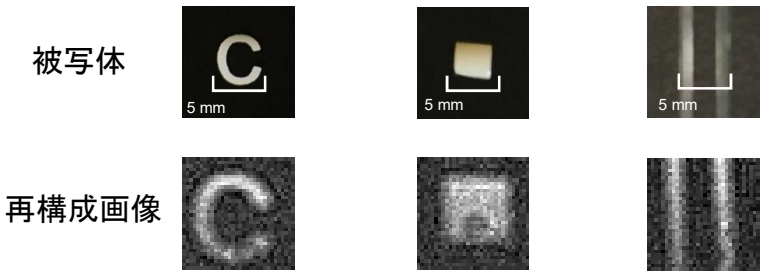


図 2.11: 光学実験の結果

グ画像再構成計算回路はCPUよりも高速に画像を再構成でき，シングルピクセルイメージング画像再構成がワンボードで可能であると示した．しかし開発した計算回路は，制御用PCを介さなければ光強度の入力や画像として表示するための出力ができず，システム大型化の課題を解決するには至っていない．他にも，再構成画像のサイズが32×32 [pixel]と小さい課題がある．しかし，再構成画像のサイズを改善するためには同時に再構成画像の画質についても改善しなければならない．計算回路では符号化パターン数を$16,384$枚としたが，本研究で使用したDMDの最大フレームレートが$4,425$Hzなので，$16,384$枚のパターン切り替えには約4秒程度かかる．画像サイズと比例して符号化パターン数も増加するため，画質の改善なく画像サイズのみを増加させた場合は画像サイズが64×64 [pixel]だと16秒，128×128 [pixel]だと64秒かかる計算になり現実的な撮影時間ではない．

これらの課題のうち，続く3，4章ではより高画質に画像を再構成することができる手法について行った研究を述べる．5章では，本章で開発した計算回路を発展させた，制御用PCの必要ないシングルピクセルイメージング専用計算機について述べる．

第3章　リカレントニューラルネットワークを用いたシングルピクセルイメージング

1 章でも述べたように，ディープラーニングを利用したシングルピクセルイメージング画像再構成は画質面で優れている．一方で FPGA は PC などに比べ内蔵メモリリソースが限られているため，計算回路の実装，特にサイズの大きい画像を再構成する場合，メモリ消費量が問題となる．本研究では，必要なパラメータ数を抑えるために，シングルピクセルイメージング画像再構成にリカレントニューラルネットワークを導入することで，パラメータ数を削減しつつ高画質な再構成を試みた．本章では，リカレントニューラルネットワークの概要について述べた後，提案手法の構成を示す．その後，シミュレーションや光学実験での結果をまとめる．

3.1　リカレントニューラルネットワーク

リカレントニューラルネットワークは，ニューラルネットワークのネットワーク構造のひとつであり，再帰型ニューラルネットワークとも呼ばれている [69]．このネットワークは主に時系列データを扱うために考案され，通常のニューラルネットワークで用いられる全結合層に内部状態を保持する機構を持たせることで，過去にネットワークに入力したデータの情報も判断材料として用いることができる．リカレントニューラルネットワークでは，ある時刻 t での入力を $\boldsymbol{x}[t]$ としたときの出力 $\boldsymbol{y}[t]$ は以下のようにあらわされる．

$$\boldsymbol{y}[t] = f\left(\boldsymbol{W}\boldsymbol{x}[t] + \boldsymbol{U}\boldsymbol{y}[t-1] + \boldsymbol{b}\right) \tag{3.1}$$

通常の全結合層の入力 $\boldsymbol{x}[t]$ とバイアス $\boldsymbol{b}$ に加えて，ひとつ前の入力 $\boldsymbol{x}[t-1]$ に対する出力 $\boldsymbol{y}[t-1]$ も入力として加えられていることがわかる．$\boldsymbol{W}$ と $\boldsymbol{U}$ はそれぞれある時刻 t での入力 $\boldsymbol{x}[t]$ に対する重みとひとつ前の時刻 $t-1$ での出力 $\boldsymbol{y}[t-1]$ を入力する際の重みである．$f(\boldsymbol{x}$ は活性化関数である．このような RNN の構造は図 3.1(a) のように再帰構造を持つことから，リカレントニューラルネットワークと呼ばれている．図 3.1(a) だけに注目すると単層のネットワークとあまり変化がないようにみえるが，実際は図 3.1(b) のように過去の層で多層のネットワークを形成しているので層の深いネットワークとなっている．また，通常のディープニューラルネットワークでは層を重ねていく毎にパラメータ数は増加していくが，RNN では過去の層の重みは時間をさかのぼっても変わらないため，表現力は劣るがパラメータ数が増加する事がない．RNN は理論上はこのように過去の入力を全て現在の出力に反映させることができ

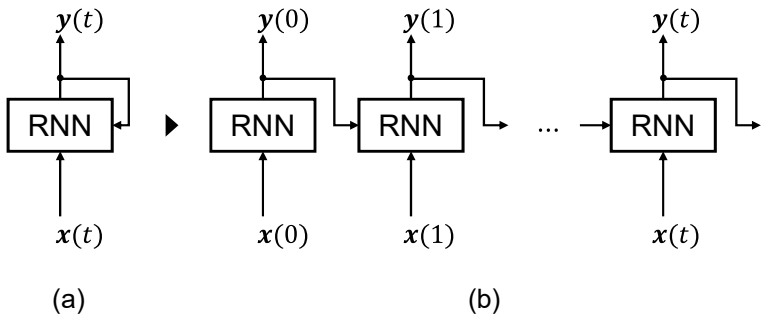


図 3.1: (a) リカレントニューラルネットワーク，(b) 時間方向に展開した場合

るが，実際には過去への層へ誤差を伝播させる時，ディープラーニングの場合と同様に勾配消失などの問題が起きてしまうため古いデータを考慮することは難しい．本研究では利用しないが，これらの課題を解決するために Long Short-Term Memory (LSTM) や Gated Recurrent Unit (GRU) などのネットワーク構造が考案されてきた [70, 71]．現在では，2017 年に発表された注意機構 (Attention) と呼ばれる構造を利用した transformer と呼ばれているネットワークが主流となっている．RNN では入出力の長さが可変なため，図 3.2 のように様々なモデルが考案されており，例えば one to many は画像からその画像の説明文を生成する文章生成タスク [72]，many to one は逆に文章から画像を生成する画像生成タスク [73]，many to many は翻訳タスク [74] などにそれぞれ利用されている．

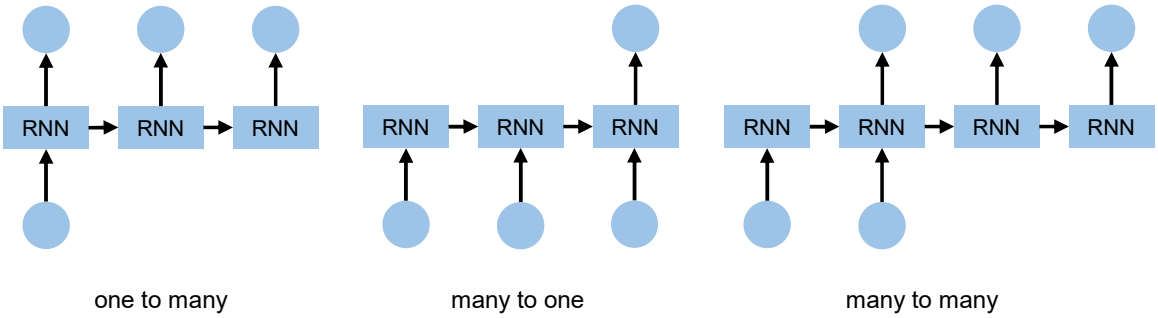


図 3.2: リカレントニューラルネットワークの入出力例

3.2 提案手法

このように，RNN は時系列データを扱うことに優れているネットワーク構造である．シングルピクセルイメージングでは符号化パターンを一枚ずつ投影し光強度を取得するため，取得した光強度は符号化パターンの枚数で離散化された時系列信号と考えることができる．そこで，光強度間の順番などの関連性を併せて RNN を用いて学習することでシングルピクセルイメージングの再構成画像の画質向上を試みた．図 3.3 に提案する画像再構成の概略図を示す．

N は符号化パターンの枚数を示している．被写体に符号化パターンを投影し測定した光強度を複数個のブロックに分け，このブロックを RNN の各時間ステップごとの入力とする．T はブロックの個数を示している．CNN ブロックは，1 章で述べた畳み込み層を利用したネットワークを示している．取得した光強度は，取得した光強度を行列 $\boldsymbol{S}$ として表現すると，

$$\boldsymbol{S} = \begin{bmatrix} s_1 & s_2 & \cdots & s_{\frac{N}{T}} \\ s_{\frac{N}{T}+1} & s_{\frac{N}{T}+2} & \cdots & s_{2\frac{N}{T}} \\ \vdots & \vdots & \ddots & \vdots \\ s_{(T-1)\frac{N}{T}+1} & s_{(T-1)\frac{N}{T}+2} & \cdots & s_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}(0) \\ \boldsymbol{s}(1) \\ \vdots \\ \boldsymbol{s}(T) \end{bmatrix} \tag{3.2}$$

のようにあらわすことができ，行ベクトルが一度に入力する光強度となる．この行ベクトルを 0 から T まで順に入力していき，全てのブロックを入力し終えた時の出力が再構成画像となる．

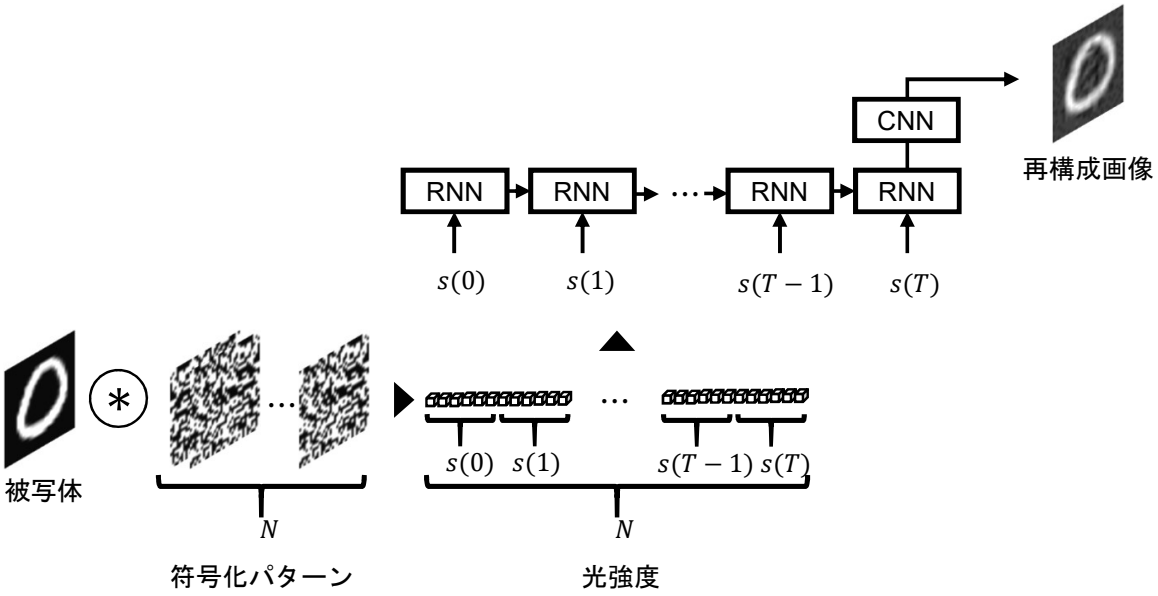


図 3.3: リカレントニューラルネットワークを利用したシングルピクセルイメージング画像再構成概略図

実際に構築したモデルを図 3.4 に示す．Fully connected は全結合層，Conv は畳み込み層を表している．畳み込み層ブロックの数字はフィルタサイズと出力チャネル数を表している．例えば最初の層の Conv$(9, 64)$ は 9×9 の畳み込みフィルタで畳み込みを行い，64 チャネルの特徴マップを出力する．畳み込み層間の数字は出力の形式を示し，$(X, Y, 64)$ なら $X \times Y \times 64$ の要素を出力することを示している．光強度をネットワークの入力にする関係上，構築したネットワークのモデル構造は，end-to-end 型のネットワーク構造 [62] を元にして，その初段に RNN 層を挿入した．初段に RNN を追加することで，光強度を時系列データとして入力することができる．また，符号化パターン数を N，再構成画像の横，縦のサイズをそれぞれ X

と Y とした場合，本来は $N \times X \times Y$ だった全結合層部分のパラメータ数が行列分解によって，$k \times X \times Y$ となり，パラメータ数の削減にも寄与している．k は RNN の出力数である．行列分解はディープラーニングでパラメータ数を削減するための手法 [75] で，図 3.5 のように $n \times m$ 行列を $n \times k$ と $k \times m$ の行列積として分解することで $n \times m$ 個の要素を $n \times k + k \times m$ 個の要素で表現でき，k が十分小さければパラメータ数を削減できる．行列分解は，一つの全結合層を二つの全結合層に置換することで簡単に実装できる．X，Y，k はそれぞれ本研究では $X = Y = 64$，$k = 200$ とした．T は分割したブロックの数で，M は単位ブロックに含まれる光強度の個数となる．本研究では符号化パターン数 $N = 333$ としたので，$T = 37$，$M = 9$ とした．各層間の活性化関数は，RNN で起きやすい勾配消失問題を防ぐため，以下に示す Recrified Linear Unit (ReLU) 関数と呼ばれるものを採用している．

$$f(x) = \begin{cases} x & (x \geq 0) \\ 0 & (x < 0) \end{cases} \tag{3.3}$$

この関数はシグモイド関数などに比べ勾配が消失しづらいことが知られている．

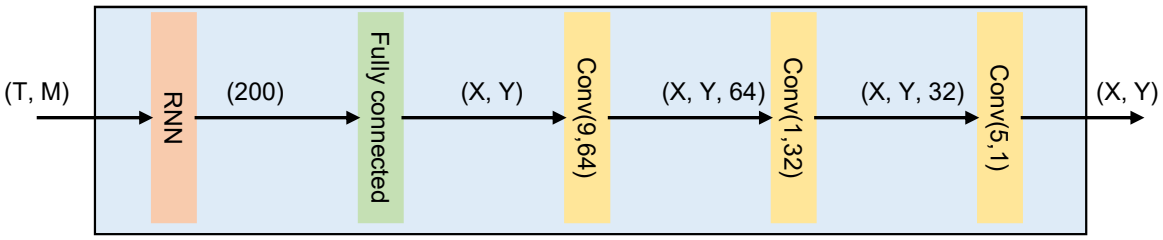


図 3.4: 提案する再構成ネットワークモデル

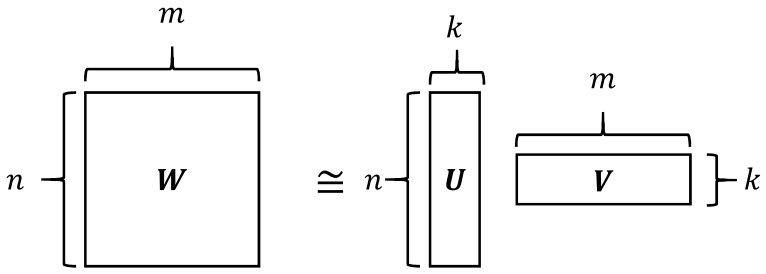


図 3.5: 行列分解

3.3 シミュレーションと光学実験

提案したモデルを評価するために，シミュレーションと光学実験による評価を行った．従来手法の比較用ネットワークを図 3.6 に示す．符号化パターンはランダムパターンを用いた場

合と，先行研究 [62] で提案された学習パターンを用いた場合の二通りで評価する．ランダムパターンを用いる場合では，RNN の有効性を確認するため，図 3.6(a) のように提案したネットワークモデルの RNN を全結合層に置換したものと比較した．学習パターンを用いる場合では，先行研究に対して有効性を確認するため，図 3.6(b) のように先行研究のネットワークモデルと比較した．ネットワークを学習するためのデータセットには STL-10[64] を 64×64 [pixel] に縮小して用いた．訓練データ数は $90,000$，評価データ数は $10,000$ で学習を行った．最適化アルゴリズムには Adaptive Moment Estimation (Adam)[65] を用いた．それぞれのネットワークモデルのパラメータは，図 3.4 の提案手法が $873,425$，図 3.6(a) の従来手法が $898,225$，図 3.6(b) の従来手法が $1,376,193$ と従来手法が最もパラメータが少なく，パラメータ数の削減に成功している．

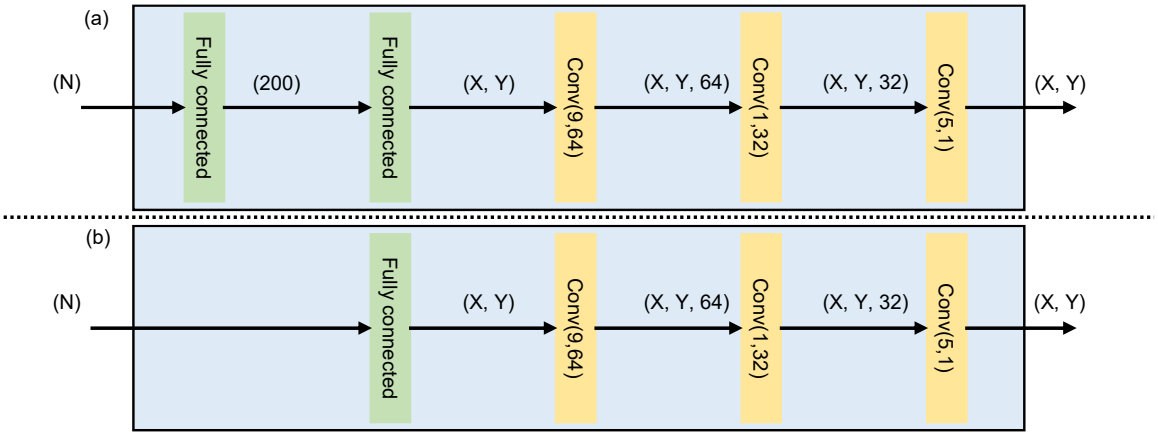


図 3.6: (a) ランダムパターンを利用する場合の比較用従来モデル，(b) 学習パターンを利用する場合の比較用従来モデル

3.3.1 再構成画像の評価

シミュレーションによる評価は MNIST[76]，Fashion-MNIST[77]，STL-10 の 3 種類のデータセットを 64×64 [pixel] に拡大縮小したものを $5,000$ 枚用いた．MNIST は 0 から 9 の手書き数字画像を集めたデータセットで，Fashion-MNIST は 10 種類のファッション商品画像を集めたデータセットである．表 3.1 にそれぞれのネットワークを用いた再構成手法での画質の平均をデータセット毎に比較したものを示す．ランダムパターンを用いた場合では，RNN を採用している提案手法の方がすべてのデータセットで従来手法よりも高い画質となっている．一方で学習パターンを用いた場合では MNIST 以外のデータセットでは従来手法の画質が高い結果となった．

次に，各手法で再構成した画像をデータセット毎に抽出したものを示す．最初に MNIST での再構成結果を図 3.7 に示す．表 3.1 で示した通り，どちらのパターンでも提案手法の画質の方が優れていることがわかる．ランダムパターンを用いた場合では，従来手法では再構

表 3.1: 各テストデータでの再構成画像画質の平均

	ランダムパターン		学習パターン	
データセット	従来手法	提案手法	従来手法	提案手法
MNIST	8.13[dB]	14.73[dB]	18.27[dB]	22.27[dB]
Fashion MNIST	9.88[dB]	15.83[dB]	21.68[dB]	20.5[dB]
STL-10	12.28[dB]	15.82[dB]	21.8[dB]	19.82[dB]

成がほとんどできていない一方，提案手法では画質は悪いが再構成できている．学習パターンを用いた場合では，どちらもほぼ正確に再構成されているが，従来手法ではノイズが目立つ．提案手法では従来手法で見られるようなノイズはほとんど見られない結果となった．次に Fashion-MNIST での再構成結果を図 3.8 に示す．ランダムパターンを用いた場合の傾向は MNIST の時と同様だが，学習パターンを用いた場合は従来手法の方が画質が優れている画像がいくつかみられる．特に，提案手法では画像内のロゴや模様などの高周波成分を多く含んでいる部分がぼやけてしまっている．また表 3.1 から，従来手法の方が平均値では上だったが提案手法の方が画質がよい場合もあることが確認できた．最後に STL-10 での再構成結果を図 3.9 に示す．ランダムパターンを用いた場合は他のデータセットと同様である．学習パターンを用いた場合は，従来手法の方が高画質な再構成画像がほとんどであり，Fashion-MNIST の場合と同様に，提案手法では画像のエッジ部分などがぼやけてしまっている．

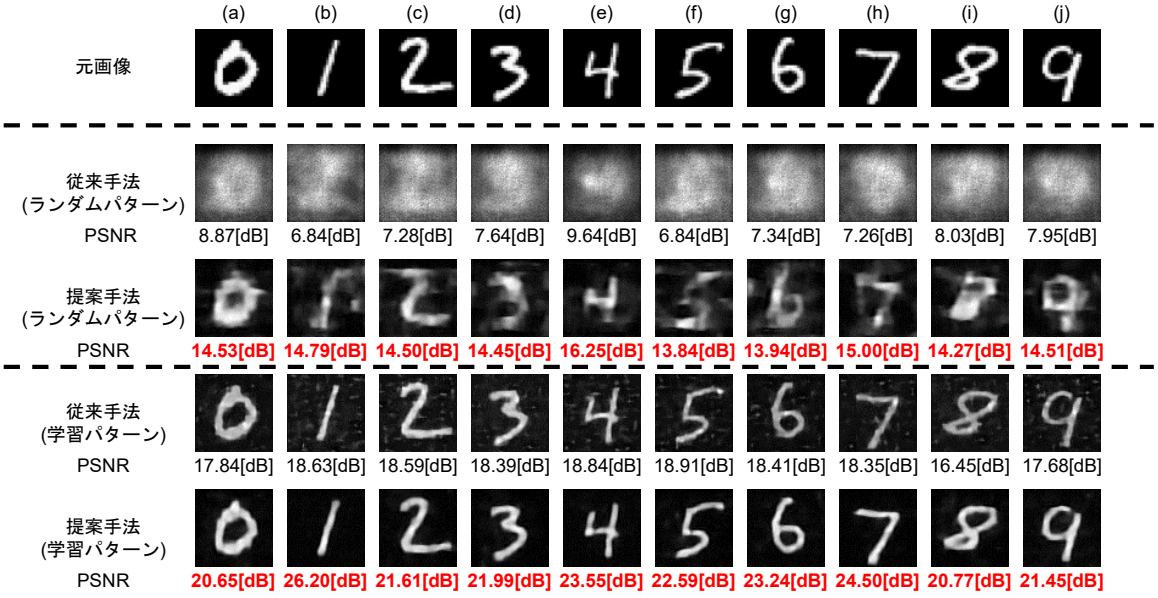


図 3.7: MNIST 画像での再構成結果

ここまでのシミュレーションから，提案手法は従来手法よりも高周波成分の再構成が不得意な代わりに，ノイズに強いであろうことが推測できる．そこでノイズ耐性を更に調べるた

図 3.8: FASHION MNIST 画像での再構成結果

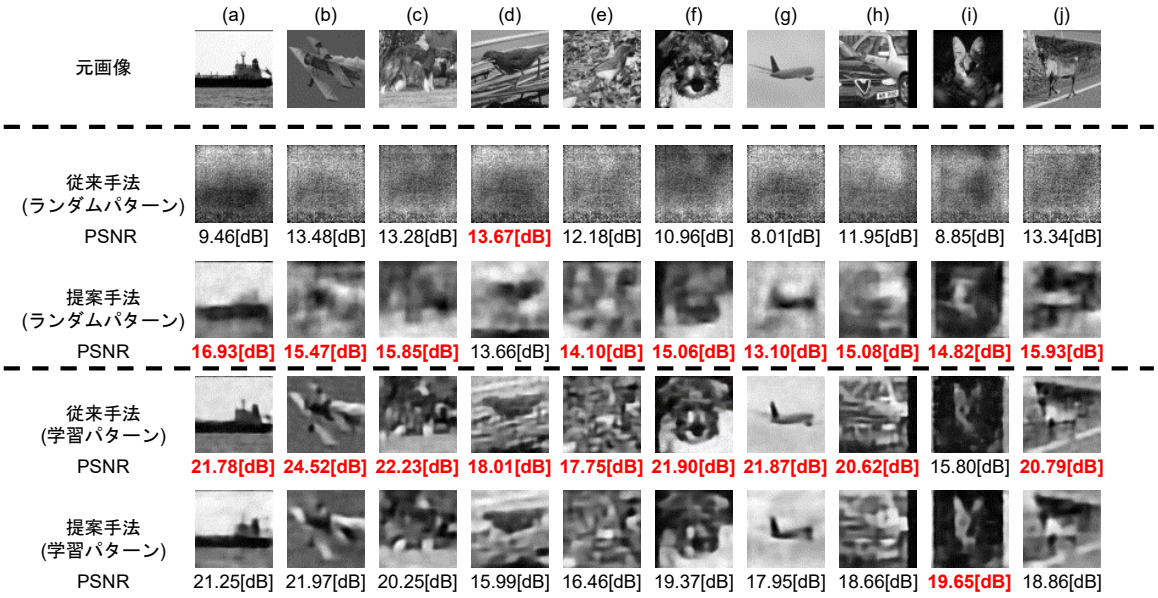


図 3.9: STL-10 画像での再構成結果

め，正規分布から生成したノイズを重畳した符号化パターンを用いて画像再構成を行った．図 3.10 にノイズを重畳したランダムパターンを用いた場合での各データセットの画質平均を示す．また図 3.11 に STL-10 データセットの再構成画像を抽出したものを示す．横軸のノイズの標準偏差は重畳したノイズの強さをあらわしている．図 3.10 ではどのデータセットでもノイズが強くなるにつれて従来手法と提案手法の差が無くなっていっている．従来手法ではノイズを増加させても画質が変わっていかないことと，図 3.11 の再構成画像からノイズの有無にかかわらず再構成できていないためこのような結果になっていることがわかる．提案手法では，ノイズの標準偏差が低いうちは再構成できているが，ノイズが増加するにつれて再構成は難しくなっている．次に，図 3.12 にノイズを重畳した学習パターンを用いた場合での各データセットの画質平均と，図 3.13 に図 3.11 と同様に再構成画像を抽出したものを示す．図 3.12 からは，ランダムパターンを用いた場合とは対照的に，ノイズが増加しても大幅な画質減少はみられない．また，どのデータセットでも従来手法より提案手法の方が画質の減少がゆるやかな様子が確認できる．そのため，MNIST では常に提案手法の画質が優れており，他のデータセットでも σ が 0.3 を超えたあたりから提案手法の画質の方が従来手法の画質よりも高くなる結果となった．図 3.13 に示した再構成画像からも，提案手法は従来手法よりもノイズの影響が少ないことが確認できる．

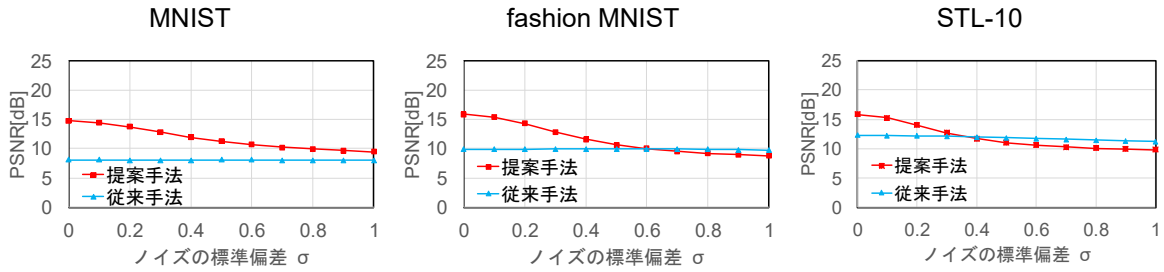


図 3.10: ランダムパターンを用いた場合でのノイズ耐性

3.3.2 光学実験

従来手法，提案手法共に学習は画像データと符号化パターンから測定される光強度を計算により求め，それらを入力として学習を行っていた．そのため，実際に光学系で得られる値とはスケールが異なることから，シミュレーションで画像再構成が可能であっても，光学系で同様に画像再構成できるとは限らない．そのため，本節では実際の光学系で測定した光強度から被写体の像を再構成する光学実験を実施した．図 3.14(a) に，光学実験で用いた光学系の概略図を示す．また，図 3.14(b) は実際に構築した光学系の写真である．DMD デバイスには最大フレームレート $9,523$[Hz] の DLP ®LightCrafter ™6500(Texas Instruments 社)，光検出器には，PDA10A (Thorlabs 社)，アナログデジタル変換器には FAD-16H シリーズ (ELMOS 社)，レーザ光源は 650nm の赤色レーザを用いた．

光学実験での画像再構成結果を図 3.15 に示す．ランダムパターンを用いた場合は従来手法

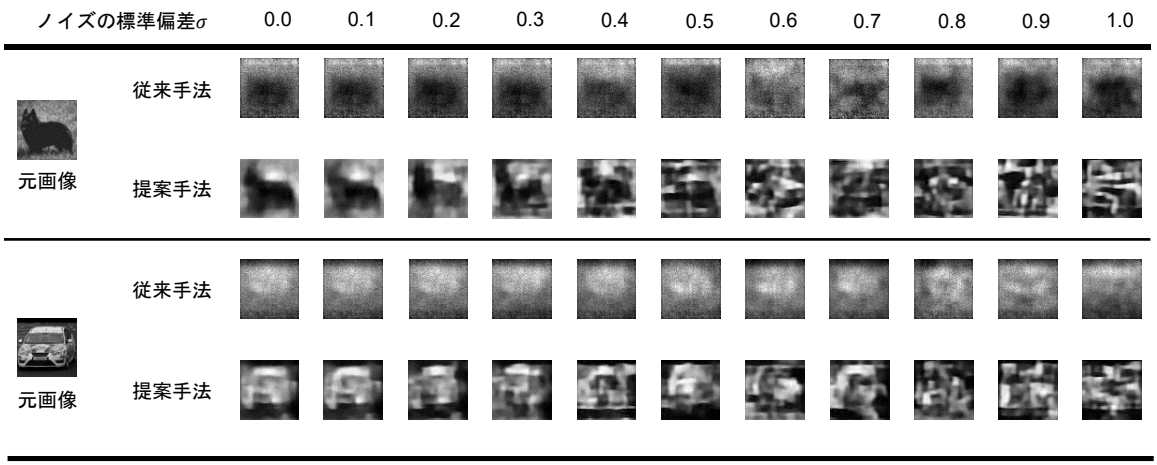


図 3.11: ランダムパターンを用いた，ノイズを付与した場合の再構成画像

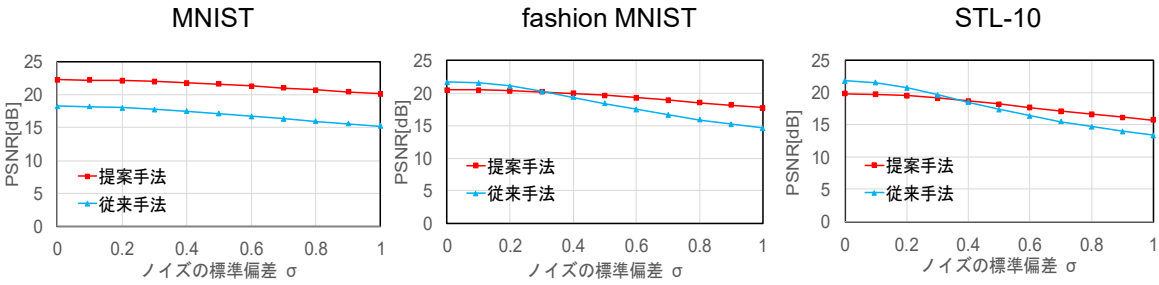


図 3.12: 学習パターンを用いた場合でのノイズ耐性

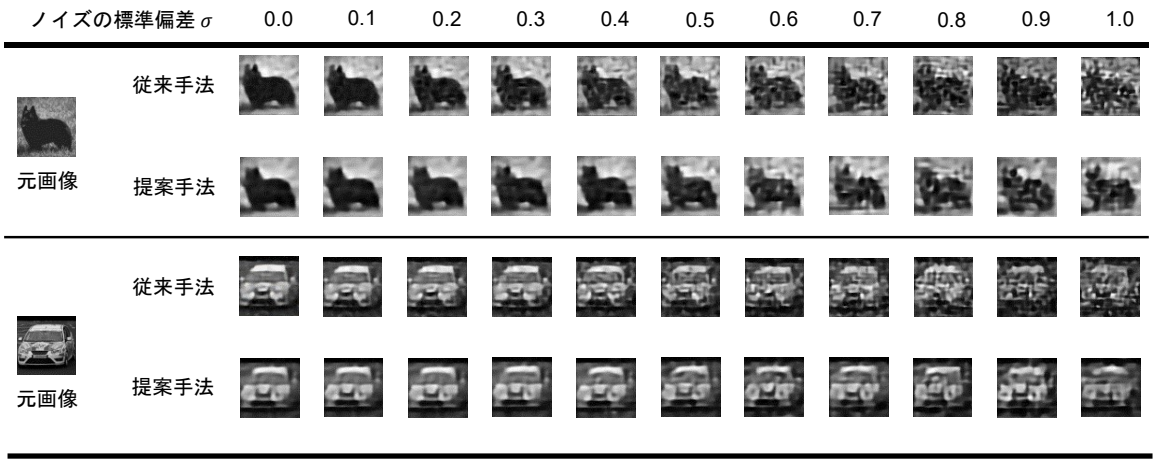


図 3.13: 学習パターンを用いた，ノイズを付与した場合の再構成画像

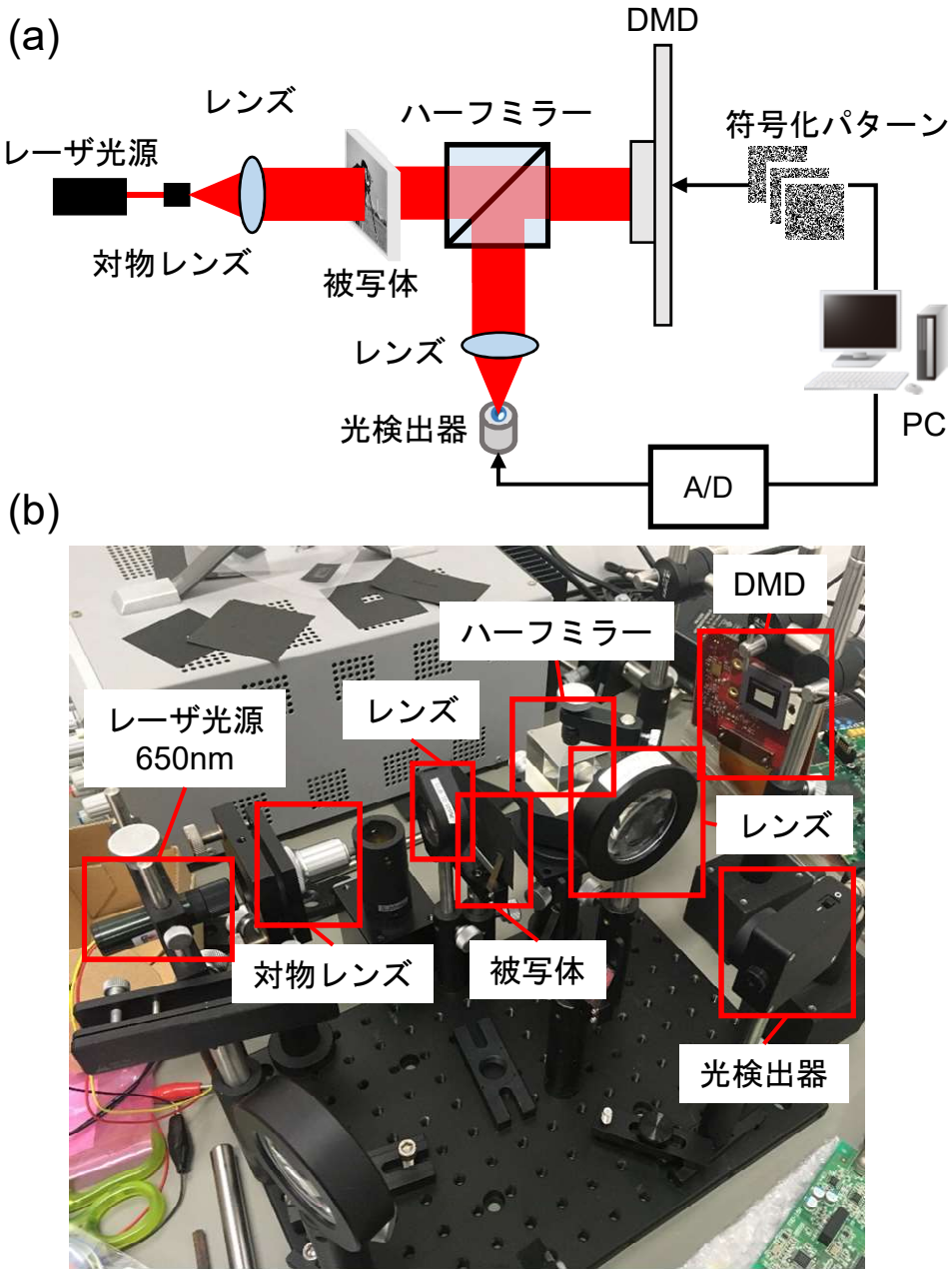


図 3.14: (a) 実験に用いた光学系，(b) 実際の写真

は画像再構成ができない一方，提案手法は低画質ながら被写体の像を再構成できている．学習パターンの場合も提案手法の方がより鮮明かつ低ノイズで画像再構成ができていることを確認できた．この結果から，実際の光学系で測定した光強度でも問題なく画像再構成が可能なことが確認できた．

3.3.3 考察

ここまで示したシミュレーションや実験で，RNN を利用した提案手法はランダムパターンを用いた場合は画像再構成を可能にし，学習パターンを用いた場合は従来手法に比べ高周波成分の再構成が難しくなる代わりにノイズに強くなることが分かった．RNN を利用した提案手法で高周波成分の再構成が難しくなった理由としては，光強度を一度符号化パターンの数よりも小さい次元に圧縮する時に情報が失われているのではないかと考えられる．より少ない情報量で画像を再構成しようとすれば，一般的に高周波成分が失われることが多く，その結果として高周波成分から欠落していく．また，ランダムパターンを用いた場合では提案手法だけが画像再構成に成功していることから RNN の方がより多くの情報を圧縮するのに成功していると考えることができる．学習パターンを用いた場合でも同様の事が発生した結果として再構成画像に発生するノイズの抑制につながったのではないかと考えられる．

本研究では，学習パターンを用いた場合では従来手法の方が高画質に画像再構成できている場合もあった．しかし学習パターンを用いた場合での従来手法のパラメータ数は提案手法の 1.6 倍程度であったことや以上の考察から，ランダムパターンを用いた場合ではもちろんのこと，学習パターンを用いる場合でも RNN をシングルピクセルイメージング画像再構成に利用することは十分有効であったといえる．

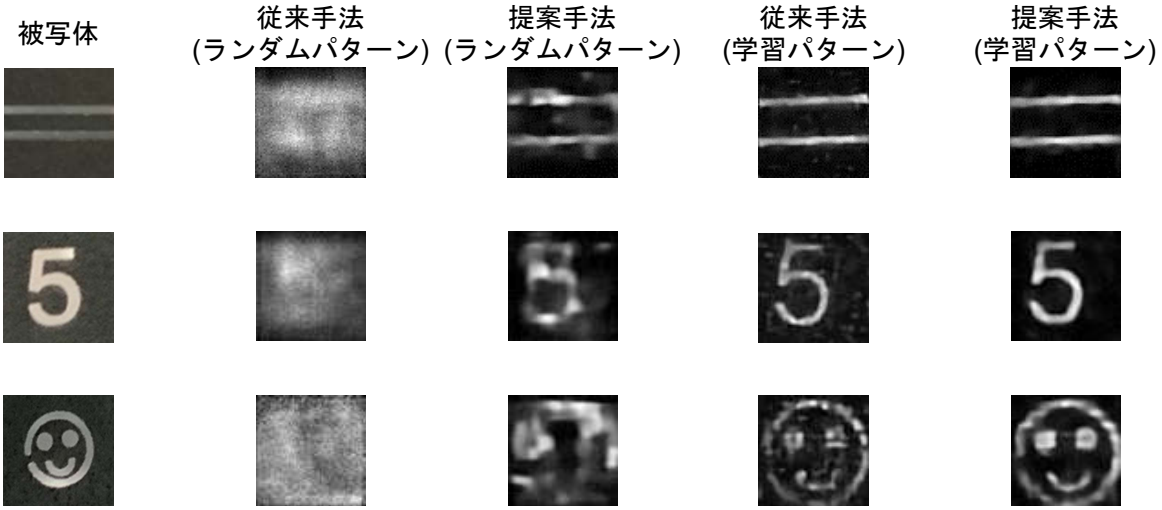


図 3.15: 光学実験での画像再構成結果

3.4 小括

本研究ではRNNを利用したシングルピクセルイメージング画像再構成を提案し，計算回路開発のためパラメータ数を従来手法と比較して40%程度削減した．そのうえで，提案手法の画質や特徴を評価するためシミュレーションと光学系を用いた光学実験を行った．その結果，ランダムパターンを用いた場合では従来手法では画像を再構成できず，RNNを利用した提案手法では画像再構成に成功し有効性が確認できた．また，従来手法で提案されていた学習パターンを用いた場合では，複雑な構造を持った被写体の場合は従来手法に比べ高周波成分が再構成できない結果となったが，単純な被写体の場合では提案手法の方が再構成画像に発生するノイズを抑制できた．符号化パターンにノイズを重畳したシミュレーションからも，提案手法は従来手法よりノイズに強いことが確認できた．これらの結果から，パラメータ数を削減しつつ高画質なシングルピクセルイメージング画像再構成手法として提案手法は有効であったことが確認できた．今後の展望として，学習した提案手法のネットワークモデルを計算回路に実装することが挙げられる．また，今回は計算回路実装が前提にあったため最もパラメータ数が少なく実装がシンプルである原始的なRNNを用いた．今後はtransformerなどより性能の高いネットワーク構造を用いて画像再構成を検討していく．

第4章　勾配降下法を利用したシングルピクセルイメージングの画質改善

前章では高画質なシングルピクセルイメージング画像再構成計算回路を実現するため，RNNを利用した画像再構成の研究について述べた．本章では，同様の目的で行ったもうひとつの研究として，計算回路実装を考慮した高画質なシングルピクセルイメージング画像再構成手法の研究について述べる．本章では，相関計算を利用したシングルピクセルイメージング画像再構成にディープラーニングでの画像再構成で提案された符号化パターンの最適化を応用することで画質の向上を試みた．また，ニューラルネットワークの学習をバイナリ値で行う手法を利用することで，最適化したバイナリの符号化パターンを得る手法を提案した．本章では，提案手法について述べた後，提案手法でシミュレーションと光学実験を行った結果についてまとめる．

4.1　提案手法

1 章でも述べた通り，相関計算を利用したシングルピクセルイメージング画像再構成は他の再構成手法よりも一般にノイズに強い代わりに低画質であるが，符号化パターンにランダムパターンではなくアダマールパターンなどの基底行列を用いることで画質が向上することが確認されている [38]．そのようなパターンは他にも様々なパターンが提案されているが [46, 47]，これらのパターンは空間周波数に基づいて情報を取得する．そのため，符号化パターン数で解像度が決まり，画素数を増やしても符号化パターン数を増やさなければ解像度は上がらない．他にも，基底パターンは冗長性がないため画質が向上する代わりにノイズに弱くなる．このような背景から本研究では，基底パターン，ランダムパターンを用いるのではなく，相関計算向けに最適化されたパターンを得ることで画質の向上を試みた．

1 章で示した相関計算を利用した再構成計算を以下に示す．S_i は被写体と符号化パターンを重ね合わせた際の光強度，R_i は符号化パターン全体の光強度である．$I_i(x,y)$ は符号化パターンの光強度分布で，$T(x,y)$ は被写体の強度係数，$O(x,y)$ は再構成した被写体の像である．

$$
\begin{aligned}
O(x,y) &= \langle S_i I_i(x,y)\rangle - \frac{\langle S_i\rangle}{\langle R_i\rangle}\langle R_i I_i(x,y)\rangle \\
S_i &= \iint T(x,y) I_i(x,y) dxdy \\
R_i &= \iint I_i(x,y) dxdy
\end{aligned}
\tag{4.1}
$$

相関計算を利用した式は以上のように単純な四則演算で計算されるため微分可能な計算となっている．よって，再構成画像と元の画像との誤差を表す損失関数 $E(T, O)$ を定義し符号化パターンに対する勾配を求めることが可能である．その後，取得した勾配でパターンを以下のように更新していくことで相関計算に最適化された符号化パターンを取得できる．

$$I_{new,i}(x, y) = I_i(x, y) - \frac{\partial E}{\partial I_i(x, y)} \tag{4.2}$$

シングルピクセルイメージングでは符号化パターンを高速に切り替えるためパターンをバイナリ化する必要があるが，最適化した符号化パターンをそのままバイナリ化すると最適な符号化パターンではなくなってしまうため画質が低下してしまう．そのため，最適化が終了した時点でバイナリパターンであることが望ましい．そのような最適化を行うため，ニューラルネットワークのパラメータをバイナリ値にしたバイナリニューラルネットワークで提案されている Xnor-Net[78] の手法を利用する．

Xnor-Net について簡単に述べる．Xnor-Net はバイナリのパラメータで推論を行い，学習は実数で行うことでバイナリのパラメータを学習する．実数のパラメータを $\boldsymbol{W}$ としたとき，まずはバイナリのパラメータ $\boldsymbol{W}_b$ を以下のように $\boldsymbol{W}$ から取得する．w は重みの要素であり，$\mathrm{sign}(w)$ は各要素をバイナリ化する関数である．

$$\boldsymbol{W}_b = \mathrm{sign}(\boldsymbol{W}) \tag{4.3}$$

$$\mathrm{sign}(w) = \begin{cases} 1 & (w \geq 0) \\ -1 & (w < 0) \end{cases} \tag{4.4}$$

このようにして得た $\boldsymbol{W}_b$ を用いて推論を行った結果から損失関数 E を計算し，その損失に対する $\boldsymbol{W}_b$ の勾配を求める．ここで，この勾配を用いて $\boldsymbol{W}_b$ を更新するのではなく，以下のように $\boldsymbol{W}$ を更新する．

$$\boldsymbol{W}_{new} = \boldsymbol{W} - \frac{\partial E}{\partial \boldsymbol{W}_b} \tag{4.5}$$

パラメータを更新後は式 (4.3) により $\boldsymbol{W}_b$ を更新することでバイナリパラメータの学習を進めていく．このように学習することでバイナリのパラメータを最適化することができる．Xnor-Net ではこのほかに，畳み込み演算に対してスケーリング係数と呼ばれるパラメータを導入し実数パラメータを近似することで更に精度を向上させている．スケーリング係数 α は以下の式で，実数パラメータ $\boldsymbol{W}$ とバイナリパラメータ $\boldsymbol{W}_b$ の差を最小化するように決定される．

$$\alpha^* = \underset{\alpha}{\mathrm{argmin}} \, \|\boldsymbol{W} - \alpha \boldsymbol{W}_b\|_2^2 \tag{4.6}$$

この式は，以下のように α について微分することで解析的に解を求めることができる．N は

パラメータの要素数である．

$$\frac{\partial}{\partial \alpha}\|\boldsymbol{W}-\alpha \boldsymbol{W}_b\|_2^2 = -2\boldsymbol{W}^{\mathsf{T}}\boldsymbol{W}_b + 2\alpha \boldsymbol{W}_b^2 = 0$$
$$\alpha = \frac{\boldsymbol{W}^{\mathsf{T}}\boldsymbol{W}_b}{\boldsymbol{W}_b^2} \tag{4.7}$$
$$= \frac{\|\boldsymbol{W}\|_1}{N}$$

図 4.1(a) に通常の畳み込み演算，(b) にフィルタをバイナリ化した場合の畳み込み演算，(c) にスケーリング係数を利用した畳み込み演算を示す．求めたスケーリング係数 α を図 4.1(c) のようにバイナリパラメータでの畳み込み演算後にかけることで実数パラメータで演算した場合の値に近づけることができバイナリ化時の誤差を低減できる．よって，Xnor-Net は学習方法の工夫とスケーリング係数による実数近似によってバイナリパラメータでの学習を行う．

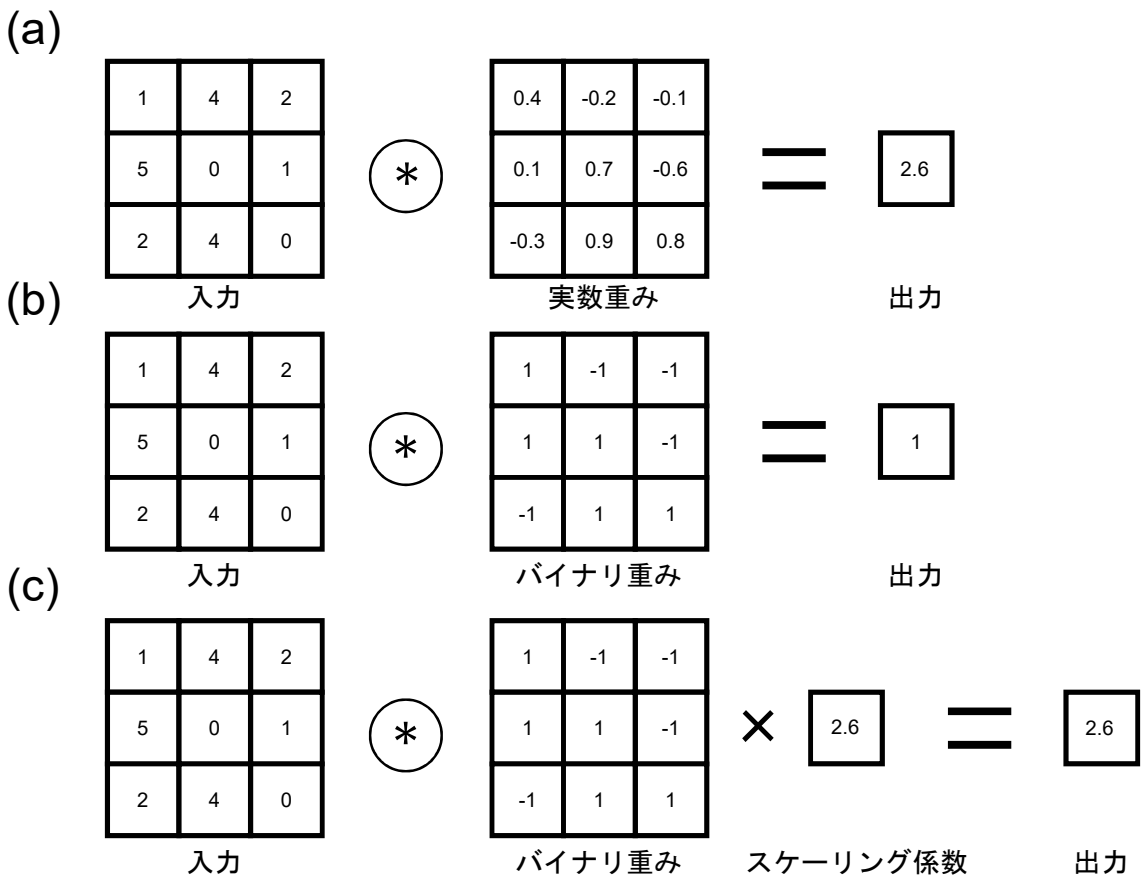


図 4.1: (a) 実数パラメータでの畳み込み演算，(b) バイナリパラメータでの畳み込み演算，(c) スケーリング係数を用いたバイナリパラメータでの畳み込み演算

図 4.2(a) に Xnor-Net での畳み込み演算，(b) にシングルピクセルイメージングで符号化パターンを投影した際光学的に行われる演算を示す．図 4.2 が示すように，畳み込み演算はシン

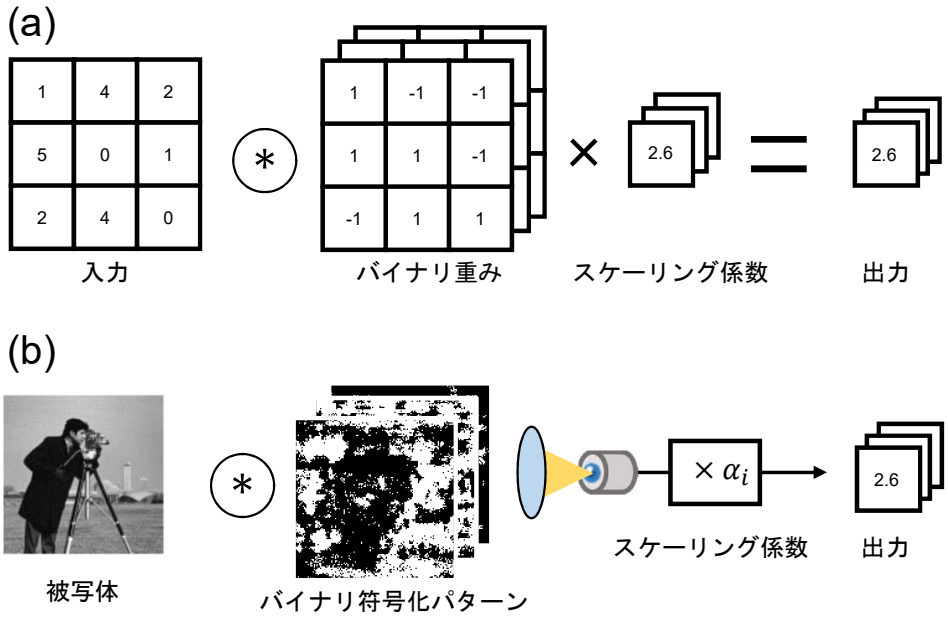


図 4.2: (a) スケーリング係数を利用した畳み込み演算，(b) スケーリング係数を利用したシングルピクセルイメージングでの符号化パターン投影

グルピクセルイメージングにおいて光強度の測定と同様の演算となっている．よって，符号化パターン毎にスケーリング係数を用意し測定後の光強度に乗算することで Xnor-Net と同様に実数符号化パターンをバイナリ符号化パターンで近似することができる．その際，バイナリ符号化パターン $B_i(x, y)$ は以下のようにあらわされる．

$$\begin{aligned} S_i &= \iint T(x,y) I_i(x,y) dxdy \\ &\approx \alpha_i \iint T(x,y) B_i(x,y) dxdy \\ R_i &= \iint I_i(x,y) dxdy \\ &\approx \alpha_i \iint B_i(x,y) dxdy \end{aligned} \tag{4.8}$$

シングルピクセルイメージングでは符号化パターンの畳み込み演算は光学的に行うため，1 と 0 にバイナリ化した方が都合がよいので以下のようにバイナリ化を行う．

$$\boldsymbol{B}_i = \mathrm{binarize}(\boldsymbol{I}_i) \tag{4.9}$$

$$\mathrm{binarize}(I_{i,k}) = \begin{cases} 1 & (I_{i,k} \geq 0.5) \\ 0 & (I_{i,k} < 0.5) \end{cases} \tag{4.10}$$

ここで，それぞれの符号化パターンは要素数 $\mathbb{R}^{X \times Y}$ の行列形式である．実数符号化パターン，バイナリ符号化パターンの要素はそれぞれ $I_{i,k}$，$B_{i,k}$ としてあらわした．X，Y はそれぞれ符

号化パターンの横，縦の画素数であり，添え字は i が符号化パターンのインデックス，k が要素のインデックスをあらわしている．バイナリ化の定義を変更してもスケーリング係数は同様に求められるが，結果が少し異なるため改めて符号化パターンに対するスケーリング係数の計算を以下の式に示す．

$$\begin{aligned}\alpha_i &= \arg\min_{\alpha_i} \|\boldsymbol{I}_i - \alpha_i \boldsymbol{B}_i\|^2 \\ &= \arg\min_{\alpha_i} \alpha_i^2 \|\boldsymbol{B}_i\|^2 - 2\alpha_i \boldsymbol{I}_i^{\mathsf{T}} \boldsymbol{B}_i + \|\boldsymbol{I}_i\|^2\end{aligned} \tag{4.11}$$

$$\alpha_i = \frac{\boldsymbol{I}_i^{\mathsf{T}} \boldsymbol{B}_i}{\|\boldsymbol{B}_i\|^2} \tag{4.12}$$

スケーリング係数導入後の再構成計算式は以下のようになる．

$$O(x,y) = \langle S_i \alpha_i B_i(x,y) \rangle - \frac{\langle S_i \rangle}{\langle R_i \rangle} \langle R_i \alpha_i B_i(x,y) \rangle \tag{4.13}$$

スケーリング係数は以上のようにしてシングルピクセルイメージング画像再構成計算に適用することができ，式 (4.3) から式 (4.5) で示した実数値でのバイナリ学習と併せて適用することで Xnor-Net を応用したバイナリ符号化パターンの学習を行った．

本研究では以上の方法に加えて，2 段階に分けた学習を行った．図 4.3 にその概略を示す．実数の符号化パターンを最適化する事前学習を Step 1，最適化した実数の符号化パターンを元にして，バイナリ符号化パターンを最適化する学習を Step 2 としている．詳細な処理はアルゴリズム 1 に示す．学習は実数の符号化パターンと画像データセットを入力として，最適化されたバイナリ符号化パターンを出力する．画像データセットは学習を行う際にいくつかのグループに分割される．このグループはミニバッチと呼ばれ，符号化パターンはミニバッチ毎に更新される．最初のステップでは実数の符号化パターンを用いて画像データから再構成画像を計算する．その後，以下の式 (4.14) に示す式で再構成画像を正規化し，定義した損失関数から元の被写体と再構成画像間の誤差を計算する．

$$O_{new}(x,y) = \frac{O(x,y) - O_{min}(x,y)}{O_{max}(x,y) - O_{min}(x,y)} \tag{4.14}$$

損失関数には式 (4.15) に示す平均二乗誤差を用いた．

$$E(\boldsymbol{O}, \boldsymbol{T}) = \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{O}_n - \boldsymbol{T}_n\|_2^2 \tag{4.15}$$

N はミニバッチの数である．その後，勾配を計算し最適化アルゴリズムによって符号化パターンを更新していく．最適化アルゴリズムには Adam[65] を用いた．これを規定回数だけ繰り返すことで最適化された実数の符号化パターンを取得する．次のステップでは，取得した実数の符号化パターンを初期値として学習を行う．まず，式 (4.10)，式 (4.11) から実数の符号化パターンからバイナリの符号化パターンとスケーリング係数を計算する．その後，式 (4.13) で

画像を再構成し，最初のステップと同様に勾配を計算後，実数の符号化パターンを学習する．その後，更新した実数の符号化パターンからバイナリの符号化パターンとスケーリング係数を更新する．このステップも規定回数だけ繰り返すことで最終的に最適化されたバイナリの符号化パターンを取得できる．

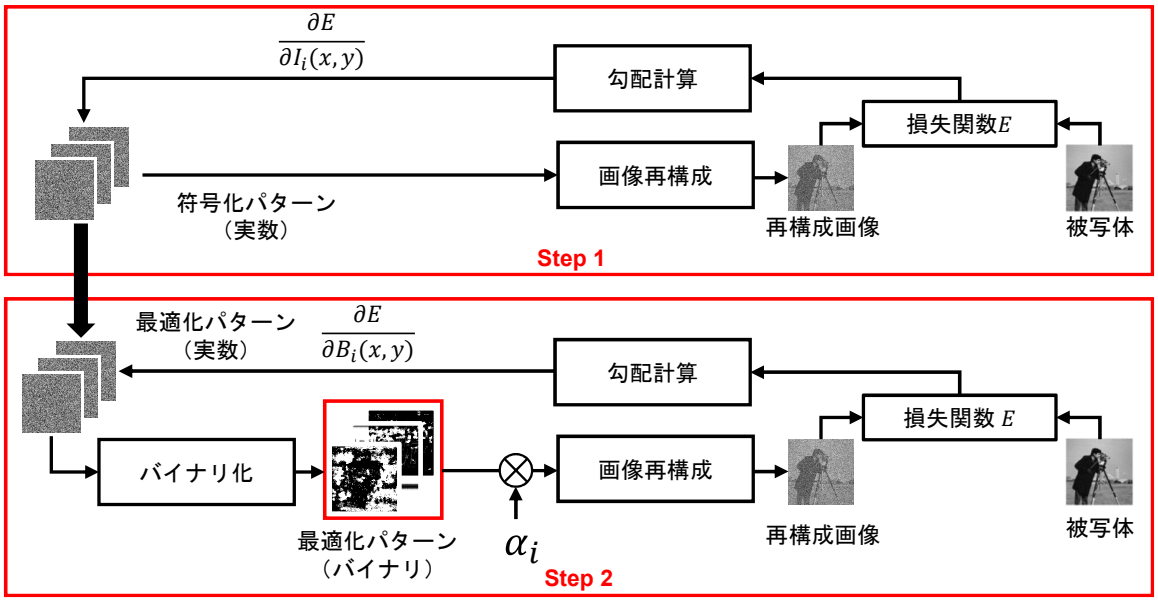


図 4.3: 提案手法の概略図

4.2 評価と考察

提案手法を評価するために，シミュレーションと光学実験を行った．符号化パターン最適化のためのデータセットには画像サイズを 128×128 [pixel] に拡大した STL-10[64] を用いた．訓練データ数は $100,000$ 枚，評価データとテストデータ数はそれぞれ $4,000$ 枚として学習を行った．

4.2.1 再構成画像の評価

図 4.4 に最適化された符号化パターンを示す．相関計算による画像再構成によく用いられているランダムパターンと，アダマール行列を元に考案された基底パターン [47] は比較対象として示した．比較対象のどちらのパターンとも強度分布が異なっていることが確認できる．

これらの符号化パターンでテスト画像 $4,000$ 枚を再構成した場合の画質平均を表 4.1 に示す．符号化パターンの枚数はそれぞれ 256，512，$1,024$ 枚で画像再構成を行った．符号化パターン数の横に示した数字は画像サイズに対する符号化パターン数の割合を示している．提案手法である最適化パターンは従来手法であるランダムパターンよりも高画質な再構成が行

Algorithm 1 Proposed two-step optimization. Blue texts denote comments.

Input: Random modulation patterns $\boldsymbol{I} = \boldsymbol{I}_{i(i=1,2,\ldots,N)}$, a minibatch of ground truth images $\boldsymbol{T}$

Output: Optimized binary patterns $\boldsymbol{B} = \boldsymbol{B}_{i(i=1,2,\ldots,N)}$, scaling factor $\boldsymbol{\alpha} = \alpha_{i(i=1,2,\ldots,N)}$

First step optimization:

for Number of epoch **do**
  for Number of minibatch **do**
    $\boldsymbol{O} \leftarrow \mathrm{DGI}(\boldsymbol{T}, \boldsymbol{I})$
    // Equation (4.1) is used
    $\boldsymbol{O} \leftarrow \mathrm{Normalize}(\boldsymbol{O})$
    // Equation (4.14) is used
    $E \leftarrow \mathrm{ErrorFunction}(\boldsymbol{O}, \boldsymbol{T})$
    // Mean squared error is used
    $\boldsymbol{I}_{new} \leftarrow \mathrm{Optimizer}(\boldsymbol{I}, \frac{\partial E}{\partial \boldsymbol{I}})$
    // Adam [65] is used as optimizer
    $\boldsymbol{I} \leftarrow \boldsymbol{I}_{new}$
  end for
end for

Second step optimization:

$\boldsymbol{B} \leftarrow \mathrm{Binarize}(\boldsymbol{I})$
// $B_{i,k} = 1$ if $I_{i,k} \geq 0.5$ else $B_{i,k} = 0$
$\alpha_i \leftarrow \frac{\boldsymbol{I}_i^{\mathsf{T}} \boldsymbol{B}_i}{\|\boldsymbol{B}_i\|^2}$
// Equation (4.11)
for Number of epoch **do**
  for Number of minibatch **do**
    $\boldsymbol{O} \leftarrow \mathrm{Reconstruction}(\boldsymbol{T}, \boldsymbol{B}, \boldsymbol{\alpha})$
    // Equation (4.13) is used
    $\boldsymbol{O} \leftarrow \mathrm{Normalize}(\boldsymbol{O})$
    $E \leftarrow \mathrm{ErrorFunction}(\boldsymbol{O}, \boldsymbol{T})$
    $\boldsymbol{B}_{new} \leftarrow \mathrm{Optimizer}(\boldsymbol{B}, \frac{\partial E}{\partial \boldsymbol{B}})$
    // Adam [65] is used as optimizer
    $\boldsymbol{I}_{new} \leftarrow \boldsymbol{I} + (\boldsymbol{B}_{new} - \boldsymbol{B})$
    $\boldsymbol{B} \leftarrow \mathrm{Binarize}(\boldsymbol{I})$
    $\alpha_i \leftarrow \frac{\boldsymbol{I}_i^{\mathsf{T}} \boldsymbol{B}_i}{\|\boldsymbol{B}_i\|^2}$
  end for
end for

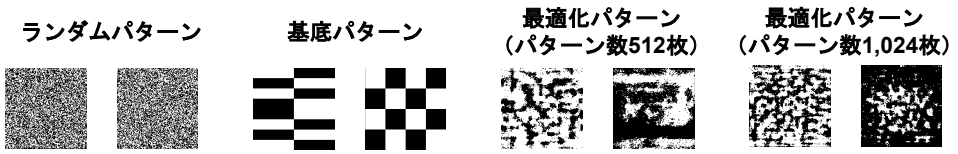


図 4.4: 提案手法で得られた符号化パターン．それぞれランダムに選択した 2 枚を掲載した．

えていることが表 4.1 より確認できる．一方で基底パターンと比較すると，画質が低くなっていることがわかる．また符号化パターン数が少ない場合よりも，多い場合の方が画質の差が顕著である．

表 4.1: 各符号化パターンでの再構成画像 PSNR 平均

符号化パターン数	ランダムパターン	基底パターン	最適化パターン
$256(1.6\%)$	11.18[dB]	17.70[dB]	16.70[dB]
$512(3.1\%)$	11.35[dB]	19.64[dB]	17.69[dB]
$1024(6.3\%)$	11.62[dB]	21.94[dB]	18.75[dB]

表 4.1 の再構成結果から符号化パターン数 $1,024$ 枚の場合での再構成結果をランダムに抽出したものを図 4.5 に示す．
画質は表 4.1 でも示した通り基底パターンが最も画質が高い場合が多い．画像の一部分を拡大し比較すると，基底パターンでは解像度が粗いような再構成画像になっているが，提案手法ではなめらかな再構成画像となっている．

次に，符号化パターンにノイズを重畳した場合での再構成画像の画質を評価した．印加したノイズは正規分布から生成した．図 4.6 にノイズの強さを変化させていった場合の画質を示す．示した画質はテスト画像 $4,000$ 枚での平均である．提案手法と基底パターンを用いた場合での画像再構成について，符号化パターンとノイズの Signal-to-Noise Ratio (SNR) を変化させた場合での画質について比較を行った．横軸の SNR は $10\log_{10}(S_{pattern}/N_{noise})$ で計算した．$S_{pattern}$，N_{noise} はそれぞれ符号化パターンと印加したノイズの平均二乗和とした．10 dB で符号化パターンの $1/10$，0 dB で符号化パターンと同じ強さのノイズとなる．図 4.6 より SNR が 8.64 [dB] より強いノイズを重畳している場合は提案手法の方が高画質となっている．

図 4.7 に，より詳細な比較として 10 dB から -10 dB まで 2 dB 刻みでノイズの強さを変化させていった場合で基底パターンと提案手法の再構成画像とその画質を示した．-10 dB 付近では基底パターンでは被写体の判別が難しくなってくるが提案手法では基底パターンでの再構成画像と比較して画質の劣化が少ないことが確認できる．

次にバイナリ化手法についての評価を行った．表 4.2 に提案手法で取得したバイナリの符号化パターンと，アルゴリズム 1 の First Step で取得した実数の符号化パターンを式 4.10 でバイナリ化した符号化パターンをそれぞれ用いた場合での画質を示す．画質はこれまでと同様テスト画像 $4,000$ 枚で再構成を行った場合の画質を平均したものである．符号化パターン数によらず，提案手法でバイナリ化した符号化パターンで再構成した画像の方が単純なしきい

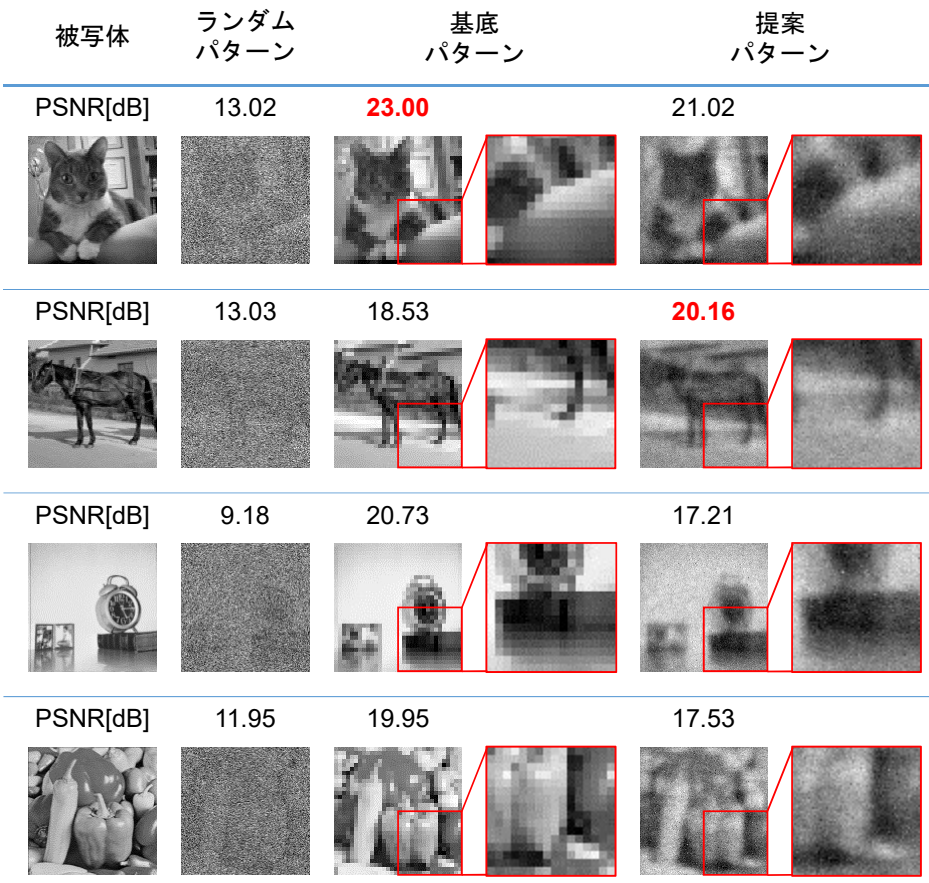


図 4.5: 再構成画像の比較

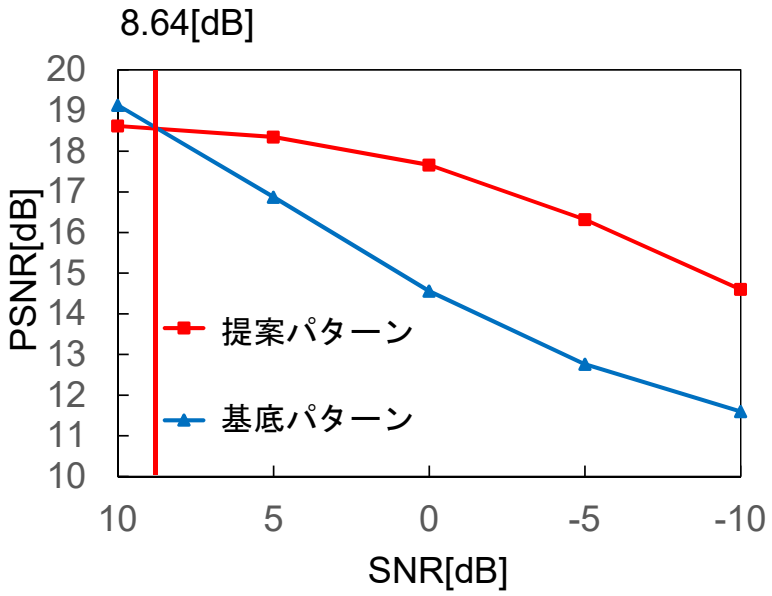


図 4.6: ノイズを重畳した場合での画質平均グラフ

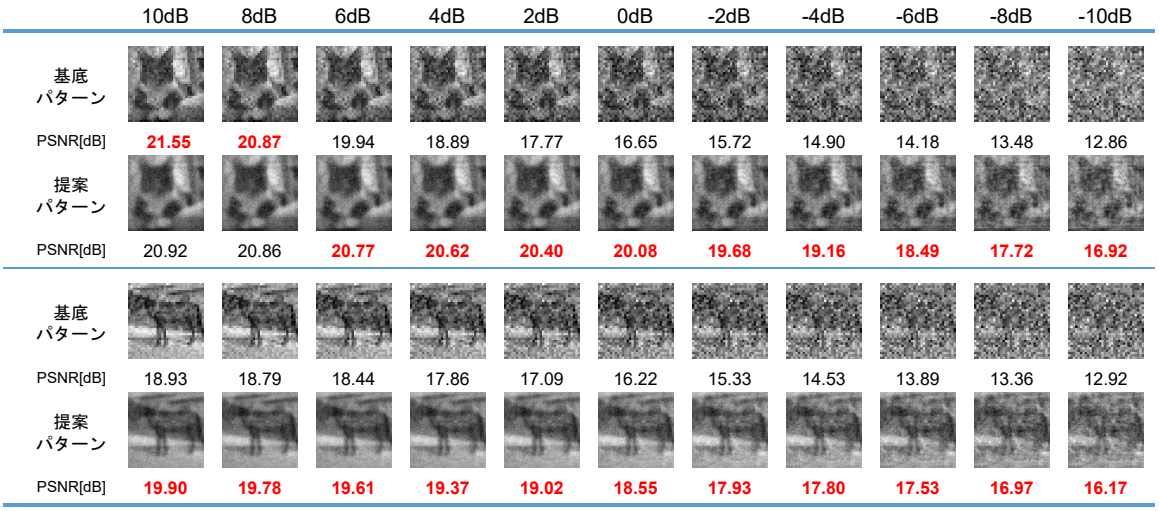


図 4.7: ノイズを重畳した場合での再構成画像の比較

値処理でバイナリ化するよりも画質が高いことがわかる．

表 4.2: バイナリ化手法の画質評価

符号化パターン数	しきい値処理	提案手法
256(1.6%)	13.47[dB]	16.7[dB]
512(3.1%)	13.62[dB]	17.69[dB]
1024(6.3%)	14.22[dB]	18.75[dB]

図 4.8 にそれぞれの符号化パターンで再構成した画像を示す．表 4.2 で示した画質と同様に，提案手法の方がより鮮明な画像の再構成が可能であることがわかる．

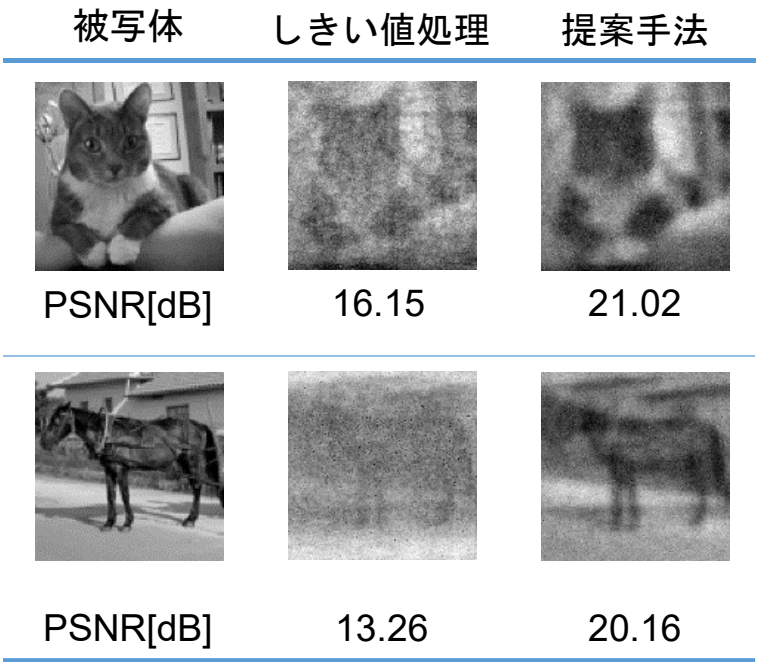


図 4.8: バイナリ化手法の比較

4.2.2 光学実験

次に，図 4.9 に示した光学系で光学実験を行った．使用した実験器具は 3 章の光学実験で使用したものと同様で，DMD デバイスは DLP ®LightCrafter ™6500(Texas Instruments 社)，光検出器には PDA10A (Thorlabs 社)，アナログデジタル変換器には FAD-16H シリーズ (ELMOS 社)，レーザ光源は 650nm の赤色レーザを用いた．

図 4.10 に光学系で測定した光強度から画像を再構成した結果を示す．符号化パターンはランダムパターン，基底パターン，提案手法のパターンを用いた．また，基底パターンと提案手法のパターンではノイズがある場合での画像再構成も行った．その際，光学実験でのノイズ

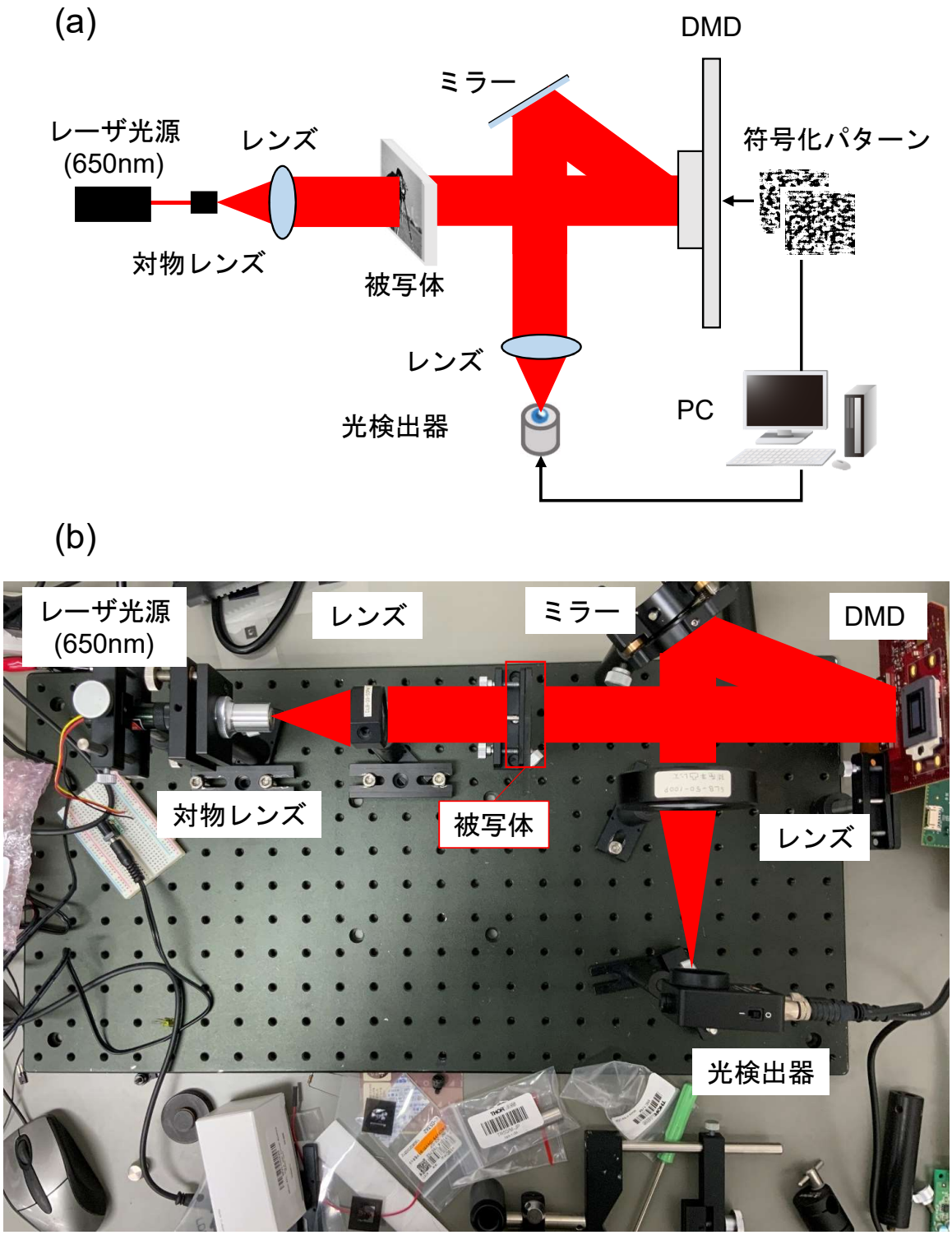


図 4.9: (a) 光学系の概略図，(b) 実際に構築した光学系

は線香の煙をファスナー付きの透明な密封容器に閉じ込め，それを図 4.9(b) の被写体と DMD の間に配置することで時間変化するノイズとした．PSNR は元画像がデータとして存在しないので，ノイズがない場合での再構成画像に対して PSNR を計算することでノイズでの画質劣化度合いとした．ノイズの強さは SNR を実際に測定したところ −5 dB から −10 dB の間であった．図 4.5 で示したシミュレーションでの再構成画像と同様，基底パターンでは解像度が粗い再構成画像だが，提案手法ではそのような粗さはみられない．また，ノイズがある場合での再構成結果は提案手法の符号化パターンを用いたほうが画質の劣化が抑えられていることが確認できた．

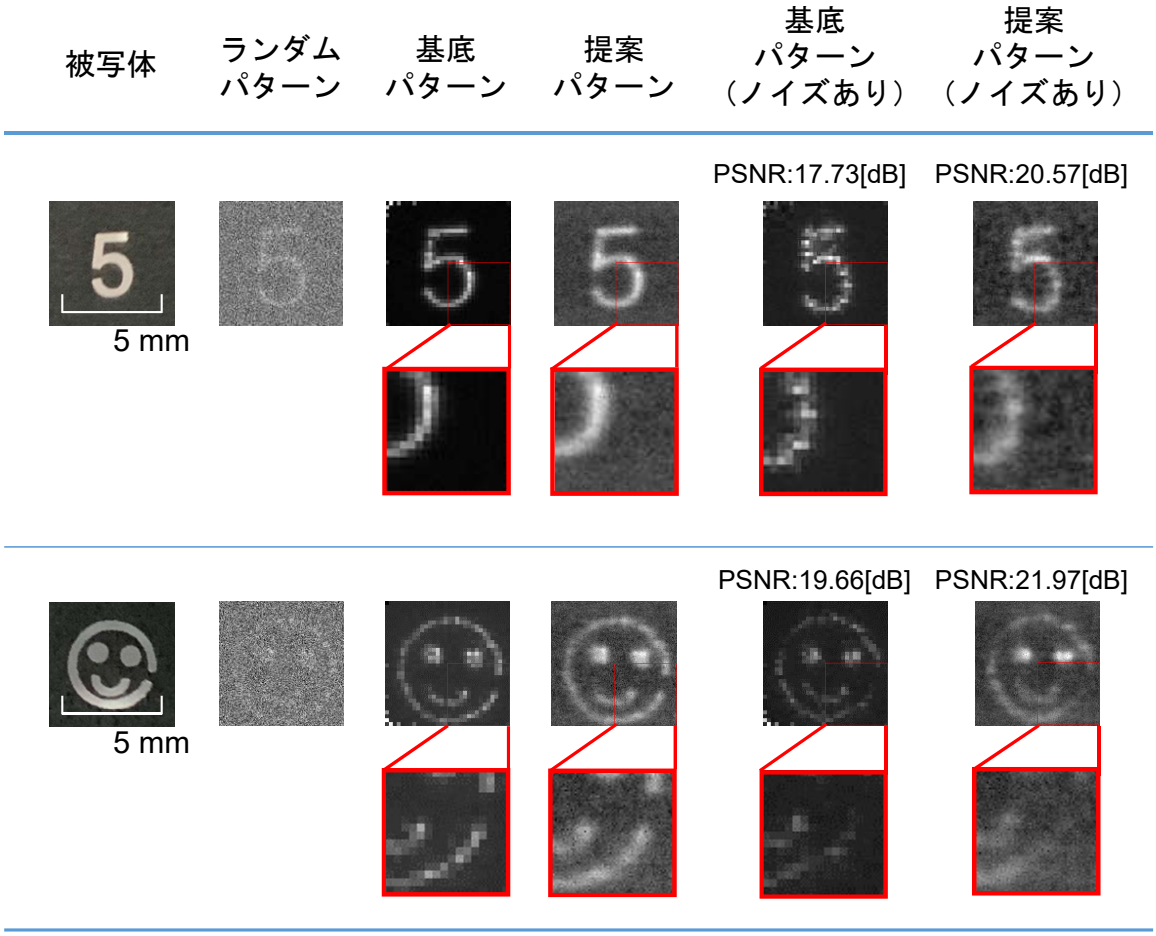


図 4.10: 光学実験での画像再構成結果

4.2.3 考察

これまでの結果から相関計算を利用したシングルピクセルイメージング画像再構成において，テストデータの画質平均は良い方から基底パターン，提案手法で最適化したパターン，ラ

ンダムパターンの順に高画質であった．一方で基底パターンにより再構成した画像は，解像度が粗い画像になっており明らかに分解能が低下している．これは基底パターンは空間周波数の順に被写体の情報を取得していくため，画素数に対して符号化パターン数が少ない場合，その分だけ高周波成分が欠落してしまうためである．図 4.5 で示した画像では符号化パターンは $1,024$ 枚投影している．基底パターンが $1,024$ 枚の場合に取得できる情報量は，32×32 [pixel] 相当の画像となる．一方で提案手法では基底パターンほどの明らかな欠落はみられず，なめらかな画像が再構成できている．また，基底パターンよりもノイズに強いことからも提案手法で得られた符号化パターンはランダムパターンと基底パターンの中間の性質を持つと推測できる．相関計算を利用したシングルピクセルイメージングは一般的に画質よりもノイズ耐性を優先したい場合に用いられることが多い．そのような場合，ノイズ耐性を持ったまま画質を改善できる本手法は有効であると考えられる．

本研究で提案した手法で利用した，損失関数を最小化する問題では定義する損失関数や追加された正則化項によって得られる結果が変わることがある．よって，本章では損失関数には式 (4.15) に示した平均二乗誤差を用いたが，正則化項の追加や損失関数の変更によっては全く別の性質を持つ符号化パターンが得られる可能性があると考えられる．

4.3 小括

本研究では，相関計算を利用したシングルピクセルイメージングに最適化された符号化パターンを提案し，従来良く用いられてきたランダムパターンを利用した場合よりも画質を向上させることができた．また，基底パターンに対しても再構成画像の空間分解能やノイズ耐性の点で有効性を示すことができた．今後の展望としては損失関数の変更や正則化項を追加することによる最適化への影響の調査などが考えられる．

第5章　SoC FPGAを用いたシングルピクセルイメージング専用計算機

本章では，2章で述べたFPGAを用いた専用計算回路を改良し課題を克服したシングルピクセルイメージング専用計算機について述べる．2章でも述べた通り，開発した専用計算回路は単体動作が不可能，得られる画質の割には符号化パターンの切り替えに時間がかかってしまう，などの課題があった．一つ目の課題については，System on a Chip (SoC) FPGAを用いることで，二つ目の課題については3と4章で開発した再構成計算を検討，実装することで解決を図った．本章の構成は，最初にSoC FPGAについて述べ，その後，専用計算機の画像再構成回路，周辺回路について述べる．最後に，シミュレーション，光学実験から開発した専用計算機の評価を行った．

5.1　System on a Chip Field-Programmable Gate Array

2章でも述べたように，FPGAのリソースには論理ゲートやFFやLUTなどの基本回路リソースの他に，特定の用途に最適化された回路リソースであるハードマクロが存在する．SoC FPGAは図5.1に示すような組み込み向けのCPUがハードマクロとして搭載されたFPGAである．CPU部分はPS，FPGA部分はPLと呼ばれる．

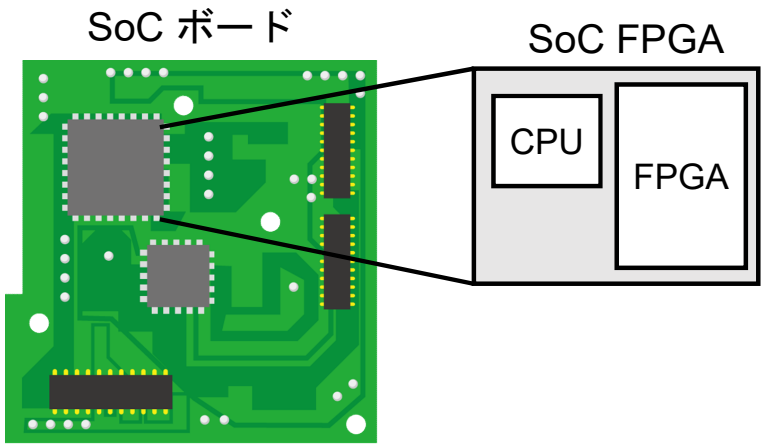


図5.1: SoC FPGAの概略図

FPGAでCPUを動作させるものとしては従来よりXilinx社からMicroBlazeと呼ばれてい

るソフトマクロがあったが，これは FPGA のリソースを消費して機能するため動作周波数は 100MHz 程度であり実用的とは言えなかった．一方でハードマクロである SoC FPGA は LSI の回路面積を占有する代わりに，1,200MHz 程度の動作周波数を達成できる．

SoC FPGA を用いることで，OS を組み込み CPU 上で動作させ FPGA 部分にユーザーが実装した回路を制御させることが可能になる．そのため，ワンチップであたかも計算アクセラレータを備えた PC のような動作を実現することができるほか，映像出力やネットワーク接続などの豊富なソフトウェア資源を CPU 部分で利用することができる．

本研究では，Xilinx 社が提供する SoC FPGA である Zynq UltraScale+ MPSoC ZCU104 評価キット（以下 ZCU104 と呼称する）を用いた．概観を図 5.2 に示す．ボード中央の黒いファンの下に FPGA チップが搭載されている．

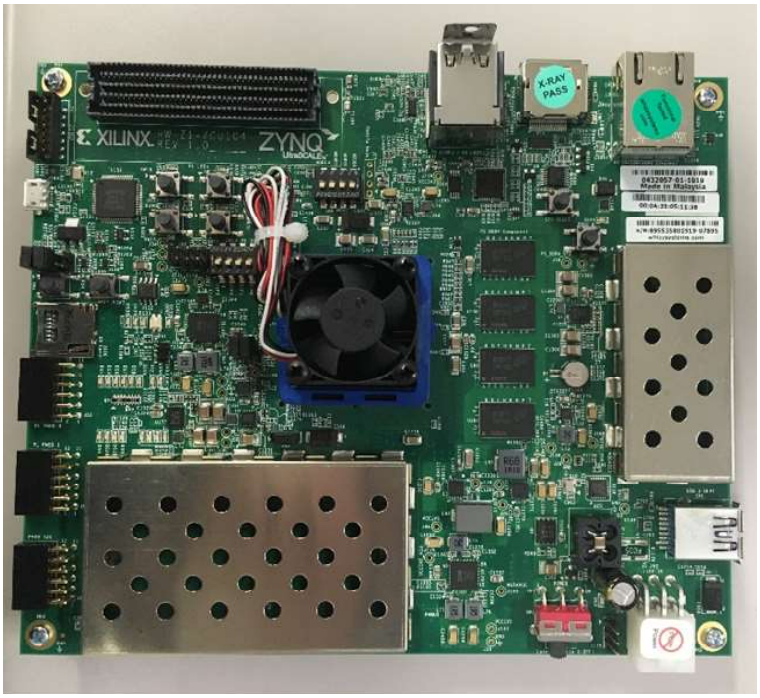


図 5.2: ZCU104 の概観

表 5.1 に ZCU104 の回路リソースを，表 5.2 に CPU 環境を示す．表 5.1 中の UltraRAM は従来まで FPGA でメモリブロックとして利用されていた分散 RAM や Block RAM に加えて，新しく追加されたハードマクロである．ブロックとしての数は少なく，他のメモリブロックとは異なり初期値を 0 以外に設定できないため Read Only Memory (ROM) としては利用できない代わりに，大容量なメモリブロックとして機能する．ZCU104 では分散 RAM は 6.2 Mb，Block RAM は 11 Mb，UltraRAM は 27 Mb まで利用できる．

表 5.2 に示した通り，組み込み CPU 上では Linux ディストリビューションの一つである Ubuntu18.04.1 が動作している．組み込み CPU で Linux を動作させるにはブートローダや Linux カーネル，root file system などのファイルが必要である．これらのファイル群は通常，linux

表 5.1: ZCU104 の回路リソース

リソース	使用可能リソース数
LUT	230,400
FF	460,800
Block RAM(36Kb)	312
UltraRAM(288Kb)	96
DSP	1,728

表 5.2: ZCU104 の CPU 環境

CPU	Quad Core Arm Cortex-A53 MPCore
Operating frequency	$1,200$ MHz
OS	Ubuntu 18.04.1 LTS

のパッケージを利用して生成することができる．本研究では，Xilinx 社が提供する Petalinux ツールを利用してこれらを生成した．しかし，Petalinux で生成したシステムはパッケージの追加など細かな調整が難しくそのまま使用するのは現実的ではないため，root file system のみ別でビルドされた ubuntu18.04.1 のファイルシステムを利用している．図 5.2 に示したように，利用したボードは評価ボードであるため，様々なインターフェースが付属している．評価ボードのサイズは $15\mathrm{cm} \times 15\mathrm{cm}$ 程度なため，現状でも PC を用いるより小型にシステムを構築できるが，チップサイズは $3.5\mathrm{cm} \times 3.5\mathrm{cm}$ 程度なので最終的にはチップのみを搭載した専用基板を開発することでより小型なシステムを構築できる．

5.2 専用計算機の構成

シングルピクセルイメージング専用計算機の構成と，システム全体の概要を図 5.3 に示す．再構成画像の画像サイズは 128×128 [pixel] となっている．本研究で構築したシステムでは，SoC FPGA を利用することで光学系で測定した信号から，スタンドアロンで画像再構成し映像として外部ディスプレイに表示することを可能とした．光学系では被写体の像をカメラレンズにより符号化パターンを表示させた DMD 上に結像させることで被写体と符号化パターンを重ね合わせる．その時の光強度を 1 点に集光し光検出器で測定する．DMD のパターン切り替え時に発する同期信号を利用してアナログデジタル変換制御回路でアナログデジタル変換器（図 5.3 中では A/D と表記）を制御し，SoC FPGA の PL 部に直接値を入力する．入力された光強度は計算回路に渡され，画像再構成を行う．その後，画像再構成結果は CPU に送信された後正規化処理を行いディスプレイに表示される．

研究の本質ではないため図 5.3 中に描画はされていないが，外部制御が必要な DMD を用いる場合は，PC に接続し動作させる．アナログデジタル変換器には FPGA から直接制御することのできる Pmod AD1 (Digilent Inc.) を用いた．SoC FPGA を用いることで計算回路を制御す

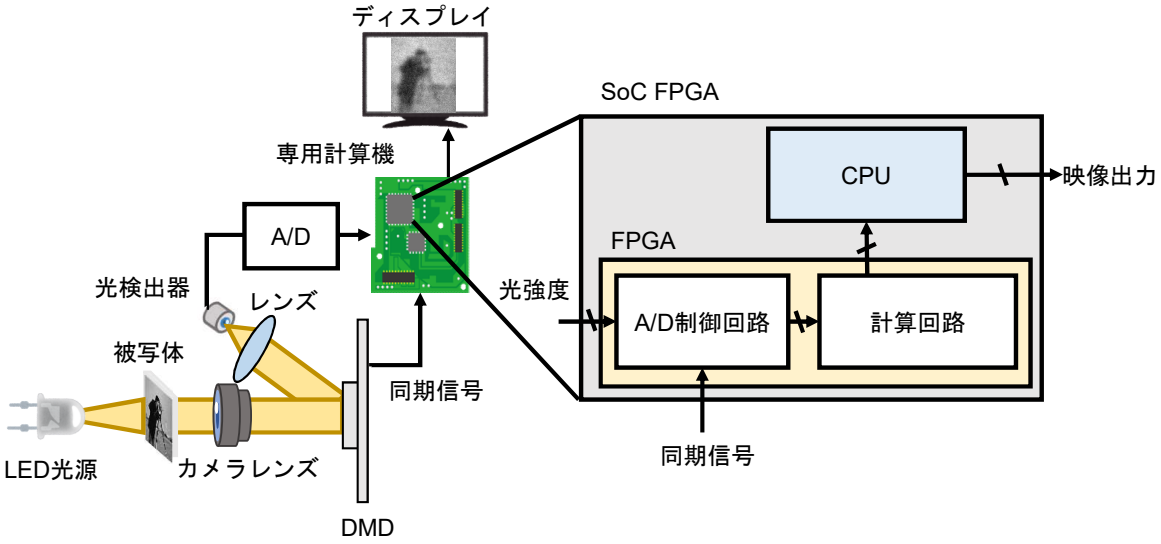


図 5.3: 専用計算機を用いたシステム全体の概略図

るための PC が必要なくなっているため，専用計算機単体で動作が可能となっていることが図 5.3 からわかる．

5.2.1 実装する再構成計算の検討

本論文では，2 章で試作した計算回路の課題であった再構成時の画質を改善するべく，3 と 4 章で再構成手法の開発を行った．結果として，本章で述べる専用計算機中の計算回路には 4 章で述べた再構成手法を採用している．理由としては，3 章で述べた RNN を利用した手法よりも 4 章で述べた最適化した符号化パターンを用いた相関計算の方が ZCU104 に実装する際より大きな画像を再構成することができるからである．表 5.3 にそれぞれの手法に必要なパラメータ数とメモリ量を示す．符号化パターン数は 512 枚として，再構成画像サイズは 64×64 [pixel] と 128×128 [pixel] の場合を示した．数値はパラメータ数を示し，かっこ内の数値は bit 単位でのメモリ量を示す．Mb は 2^{20}bit である．

表 5.3: 各再構成手法のパラメータ数と必要メモリ量

再構成手法	64×64 [pixel]	128×128 [pixel]
RNN を用いたディープラーニング	$2,970,377$ (23 Mb)	$11,731,721$ (90 Mb)
最適化パターンを用いた相関計算	$2,097,152$ (2 Mb)	$8,388,608$ (8 Mb)

パラメータ数はあまり差がないが，RNN を用いたディープラーニングのメモリ量が最適化パターンを用いた相関計算の 10 倍程度になっていることがわかる．これはディープラーニングの推論には一般的に 8bit 精度があれば十分とされていることから，1 パラメータにつき 8bit 使用する前提で計算したためである．対して，相関計算のパラメータはバイナリの符号化パター

ンなので 1bit で済む．ZCU104 は前述のように分散 RAM は 6.2 Mb，Block RAM は 11 Mb，UltraRAM は 27 Mb で合わせて 44 Mb の内部メモリを持つ．この事と表から，ディープラーニングで再構成する場合は 64×64 [pixel] が限界だが相関計算で再構成する場合は 128×128 [pixel] の画像まで再構成できることがわかる．

5.2.2 並列計算回路

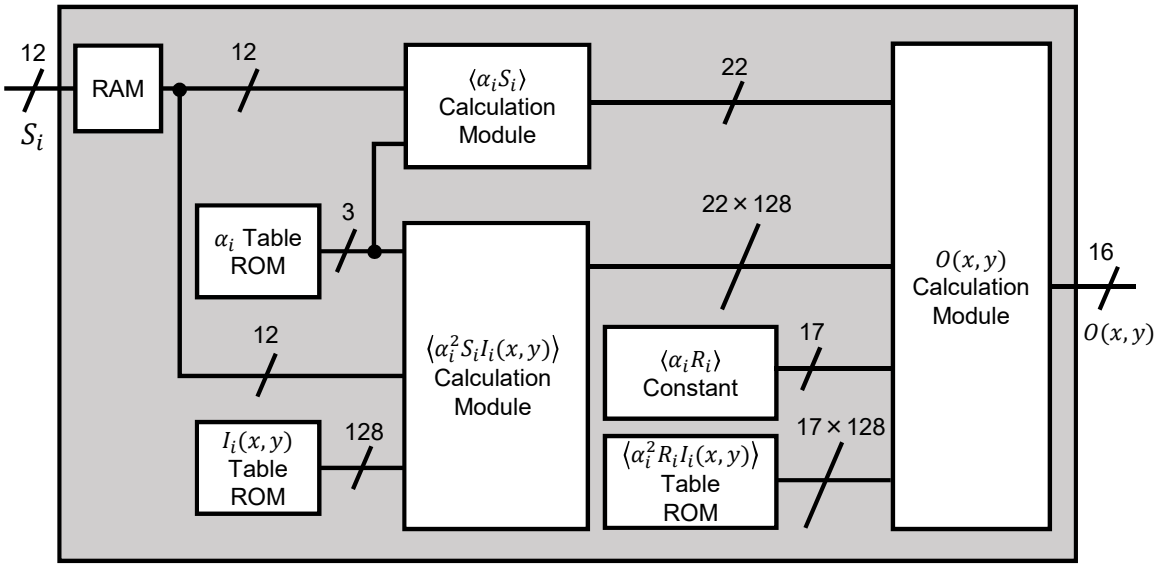


図 5.4: 計算回路のブロック図

設計した計算回路のブロック図を図 5.4 に示す．実装した再構成計算は 4 章で述べたものになるが，再構成計算回路の計算に沿って書き直したものを以下にもう一度示す．本章では計測した光強度を $S_i = \iint T(x,y) I_i(x,y) dxdy$，符号化パターンの強度を $R_i = \iint I_i(x,y) dxdy$ としている．

$$O(x,y) = \langle \alpha_i^2 S_i I_i(x,y) \rangle - \frac{\langle \alpha_i S_i \rangle}{\langle \alpha_i R_i \rangle} \langle \alpha_i^2 R_i I_i(x,y) \rangle \tag{5.1}$$

FPGA は除算が苦手なため 2 章で述べた計算回路と同様，実際の計算では以下のように，両辺に定数項である $\langle \alpha_i R_i \rangle$ を乗算したものを計算している．

$$\langle \alpha_i R_i \rangle O(x,y) = \langle \alpha_i R_i \rangle \langle \alpha_i^2 S_i I_i(x,y) \rangle - \langle \alpha_i S_i \rangle \langle \alpha_i^2 R_i I_i(x,y) \rangle \tag{5.2}$$

$\langle \alpha_i R_i \rangle$，$\langle \alpha_i^2 R_i I_i(x,y) \rangle$ は使用する符号化パターンによって決定される既知の値なため，事前に計算しテーブルとして実装している．また，符号化パターンとスケーリング係数 α_i も同様にテーブルとして実装している．実際に計算を行っているのは $\langle \alpha_i S_i \rangle$ の項と $\langle \alpha_i^2 S_i I_i(x,y) \rangle$ の項，それぞれの項を乗算，減算し $\langle \alpha_i R_i \rangle O(x,y)$ を求める部分である．

計算回路の構成として，まずは入力として光学系で測定した光強度が A/D 変換制御回路から送信される．光強度は RAM に格納された後，$\langle \alpha_i S_i \rangle$ Calculation Module と $\langle \alpha_i^2 S_i I_i(x,y) \rangle$ Calculation Module に渡され計算を行う．それらの計算が終了次第，結果が $O(x,y)$ Calculation Module に渡され，回路全体の計算結果として再構成画像が出力される．斜線の上の数字は信号の bit 数である．bit 数は，アナログデジタル変換器が 12bit なので入力信号である S_i を 12bit として，それを元に他の信号の bit 数を決定した．その後，自然画像を対象として，CPU で再構成したものとの PSNR が 30[dB] を下回らないよう bit 数を削減したものを最終的な信号幅とした．$\langle \alpha_i^2 S_i I_i(x,y) \rangle$ Calculation Module と $O(x,y)$ Calculation Module は内部で 128 個の計算モジュールを並列で動作させている．

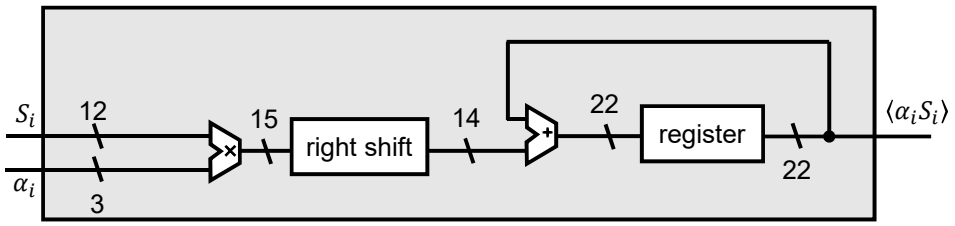


図 5.5: $\langle \alpha_i S_i \rangle$ Calculation Module のブロック図

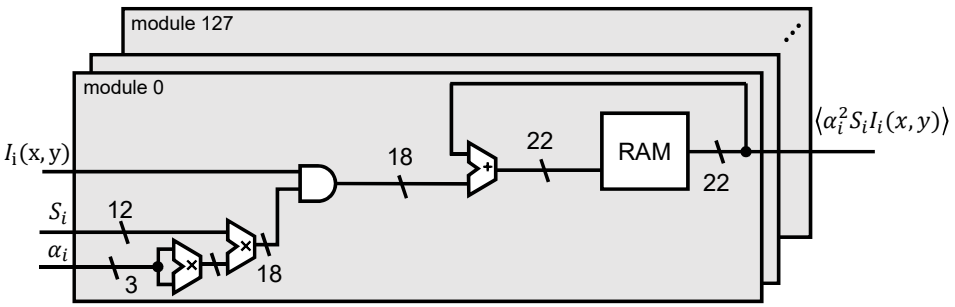


図 5.6: $\langle S_i \alpha_i^2 I_i(x,y) \rangle$ Calculation Module のブロック図

図 5.5 に $\langle \alpha_i S_i \rangle$ Calculation Module のブロック図を，図 5.6 に $\langle \alpha_i^2 S_i I_i(x,y) \rangle$ Calculation Module のブロック図を，図 5.7 に $O(x,y)$ Calculation Module のブロック図を示す．$\langle \alpha_i S_i \rangle$ Calculation Module では，光強度 S_i とスケーリング係数 α_i を入力として，両者を乗算後，最下位 bit を削り 14bit としたものを加算する積和演算を行う．全ての光強度を入力し終えたら，その時の出力を次の回路に渡す．

$\langle \alpha_i^2 S_i I_i(x,y) \rangle$ Calculation Module では，光強度 S_i とスケーリング係数 α_i，符号化パターン $I_i(x,y)$ を入力として，それぞれの積を計算する．bit に幅のある光強度 S_i とスケーリング係数の二乗 α_i^2 を計算する際はそれぞれ乗算器を用いているが，符号化パターンは 1bit なので各ビットの AND 演算により乗算を行っている．その後，同じ座標同士で値を足し合わせていき，全ての値の足し合わせが終了したら出力として次の回路に値を渡す．図 5.6 に示すモジュールは 128 個並列に用意しており再構成画像サイズが 128×128 [pixel] なのでひとつの

モジュールが 1 ライン分の計算を担当する．そのため，複数の値をモジュール内で保持しておくため RAM が用意されている．

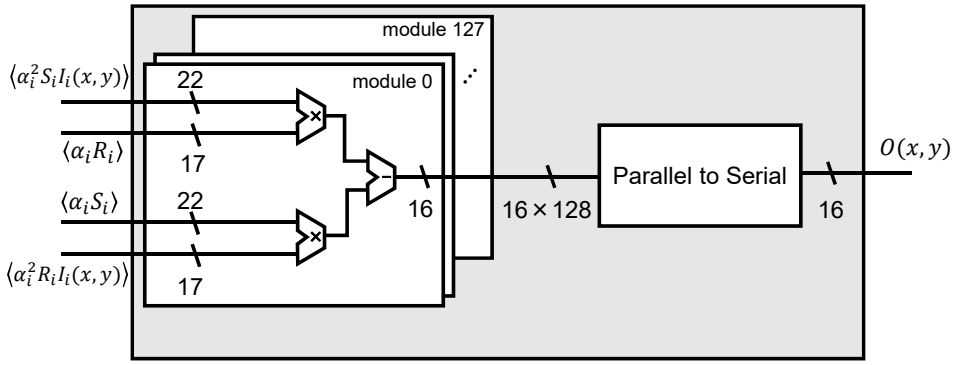


図 5.7: $O(x,y)$ Calculation Module のブロック図

図 5.7 に示す $O(x,y)$ Calculation Module では，前段の回路で計算された $\langle \alpha_i S_i \rangle$, $\langle \alpha_i^2 S_i I_i(x,y) \rangle$ とテーブルに格納された $\langle \alpha_i^2 R_i I_i(x,y) \rangle$，定数として実装されている $\langle \alpha_i R_i \rangle$ を受け取り，それぞれの乗算結果を減算する．乗算と減算部分を行う回路モジュールは $\langle \alpha_i^2 S_i I_i(x,y) \rangle$Calculation Module と同様に 128 個並列動作しており，最終的な出力はパラレルシリアル変換回路を通して 16bit の信号として出力される．

5.2.3 通信回路

本研究では PL，PS 間の通信インターフェースには Advanced extensible interface (AXI) と呼ばれる Arm 社が策定，公開しているマスター，スレーブ間のインターフェースを規定したものを用いている．Xilinx 社が提供する IP コアのインターフェースは AXI に共通化されており，ハードマクロである CPU もこのインターフェースも持っているため，自作の計算回路に AXI インターフェースを実装することで CPU と通信を行うことができる．AXI には通常の AXI のほかに，AXI-Lite，AXI-Stream と呼ばれる仕様がある．AXI は大容量，高速転送が可能であり，アドレスを与えた後，連続に転送するバースト転送を行う．AXI-Lite は AXI のサブセットで，バー-スト転送機能がなく AXI の一部の信号のみで構成されるインターフェースである．AXI-Stream は一方通行でデータを流し続ける高速大容量かつ軽量なインターフェースである．

図 5.8 に AXI の転送の仕組みを示す．AXI では送信側から受信側へデータを送信する際，VALID 信号と READY 信号でハンドシェークして通信を行う．送信側は有効なデータを送信している間は VALID 信号を 1 に，受信側は受信準備ができている間は READY 信号を 1 にし，両者が 1 の時に通信が成立する．この転送の単位を AXI ではチャネルと呼び，図 5.9 に示すように AXI では AR，R，AW，W，B の 5 つチャネルが存在する．各チャネルにはそれぞれ図 5.9 に示したようにデータ信号，VALID 信号，READY 信号があるため，各チャネルごとに独立して動作することができる．マスターがアドレスを送信した後，すべてのデータ転送

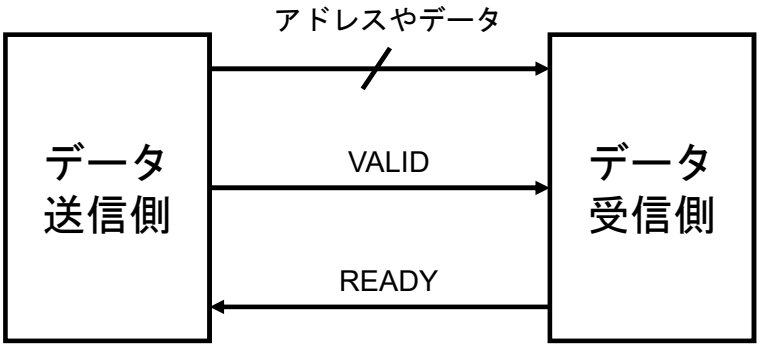


図 5.8: AXI の転送

が終了するまでの一連のやりとりをトランザクションと呼ぶ．読み出しトランザクションはマスターが AR チャネルで読み出しアドレスを与えた後，R チャネルでデータを読み出すまでを指し，書き込みトランザクションはマスターが AW チャネルで書き込みアドレスを与えた後，W チャネルでデータを書き込み，B チャネルで応答を受け取るまでのやりとりを指す．AXI-Lite も同様の仕組みでトランザクションを行うが，AXI-Stream では AR，AW チャネルが存在せず，データは常に一方向に送信される．

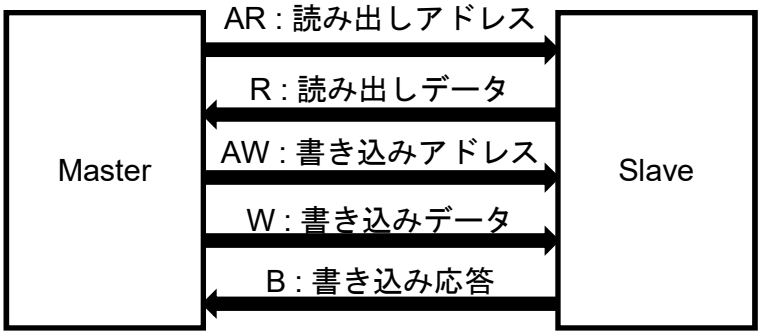


図 5.9: AXI のチャネル

図 5.10 に AXI を用いた PL，PS 間の通信を行う回路を示す．PS がマスター，PL がスレーブとなっている．図 5.7 から出力された再構成結果が Calculation circuit から渡されるので，それを First In First Out (FIFO) に格納している．FIFO はその名前の通り格納した順にデータを取り出すデータ構造である．読み出しトランザクションが発生すると FIFO からデータを順に読み出し CPU に送信する．PS は AXI 用にアドレス空間を持っているが，本研究ではそのうち最初のアドレスのみに送信データを連続で書き込んでいる．図 5.10 では CPU から回路に送信する信号があることも示している．CPU から回路に送信する書き込みトランザクションは，主に回路中の ROM を CPU から初期化するために用いている．

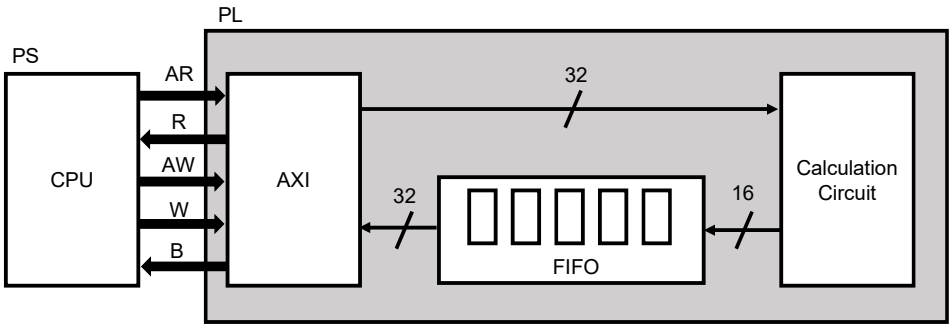


図 5.10: 通信回路のブロック図

5.2.4 その他の回路

図 5.11 に A/D 変換制御回路のブロック図を示す．アナログデジタル変換器に用いた Pmod AD1 は Serial Peripheral Interface (SPI) プロトコルで通信を行うため SPI Controller モジュールで Pmod AD1 を制御している．SPI プロトコルはマイクロコントローラや周辺 IC でよく用いられる通信プロトコルであり，図 5.12 に示すように Chip Select (CS)，CLK，Master In Slave Out，Master Out Slave Out 信号で構成される．Pmod AD1 は Master Out Slave In 信号は持たないが，図 5.12 では一般的な場合の SPI を示している．CS 信号はスレーブをアクティブにするための信号で，一般的に信号がローの場合にスレーブはアクティブとなる．CS 信号がハイの場合はデータのやり取りは行われない．CS がローの間はクロックに同期し Master In Slave Out からデータを受信したり，Master Out Slave In でデータを送信することが可能である．

図 5.11 の A/D 変換制御用受信回路では DMD からパターン切り替え時の同期信号を受け取り CS をアクティブにすることでパターンに同期したタイミングで Pmod AD1 と通信を行い光強度を測定する．アナログデジタル変換された光強度は 12bit の値が 1bit ずつシリアルで送信されてくるので，シリアルパラレル変換で 12bit の値にした後 FIFO に格納される．

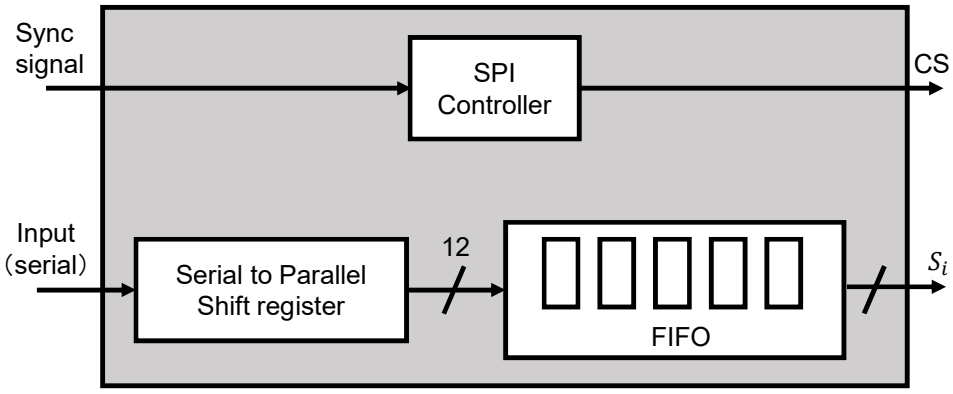


図 5.11: A/D 変換制御用受信回路のブロック図

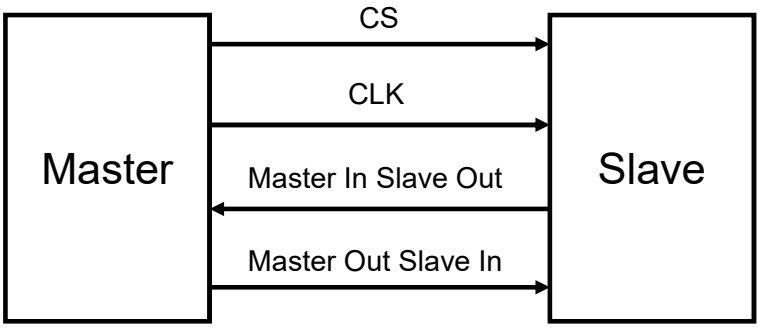


図 5.12: SPI プロトコル

5.2.5 回路リソース

表 5.4 に開発した専用計算機の回路リソース使用率を示す．本研究では計算回路について，符号化パターン数が 512 枚の場合と 1,024 枚の場合の 2 通りの回路を設計，実装した．どちらの場合もこれまで示してきた回路ブロック図から変化はない．符号化パターン数が 512 枚の場合では符号化パターンを保持しておくのに必要なメモリ量は $128 \times 128 \times 512 = 2^{23}$b で 8 Mb あれば良いため UltraRAM の使用率は 0 になっているが，符号化パターン数が 1,024 枚の場合はその倍の 16 Mb 必要で Block RAM だけでは足りないため UltraRAM を使用している．

表 5.4: 専用計算機のリソース使用率

		符号化パターン数 512		符号化パターン数 1,024	
リソース	リソース数	使用リソース数	使用率 [%]	使用リソース数	使用率 [%]
LUT	230,400	18,246	7.92	18,190	7.89
LUTRAM	101,760	10,971	10.78	11,013	10.82
FF	460,800	2,312	0.50	2,581	0.56
Block RAM	312	267.50	85.74	39.50	12.66
UltraRAM	96	0	0	64	66.67
DSP	1,728	384	22.22	384	22.22

5.3 評価と考察

開発した専用計算機を評価するため，シミュレーションによる画質，計算速度の評価と実際に専用計算機を使用した光学システムを構築し光学実験を行った．前述の通り再構成画像のサイズは 128×128 [pixel]，符号化パターン数は 512 枚，1,024 枚で評価を行った．

5.3.1 再構成画像の評価

専用計算機内に実装されている計算回路では 2 章で述べた試作段階の計算回路と同様に小数を固定小数点数で表現している．この固定小数点数は，信号全体の bit 幅はもちろん，小数部分の bit 幅などの取り方が結果に大きく影響する．そこで再構成が正常に行えているかどうか，浮動小数点数で計算した場合よりも画質が低下していないかを確認するため，図 5.13 に CPU，計算回路それぞれで画像を再構成した結果を示す．画質の評価には PSNR を用いた．結果としては，PSNR による定量的な評価では差がほぼ見られず，視覚的にも画質の劣化などは見られない．この事から，十分な bit 幅で再構成計算を行えていることが確認できた．

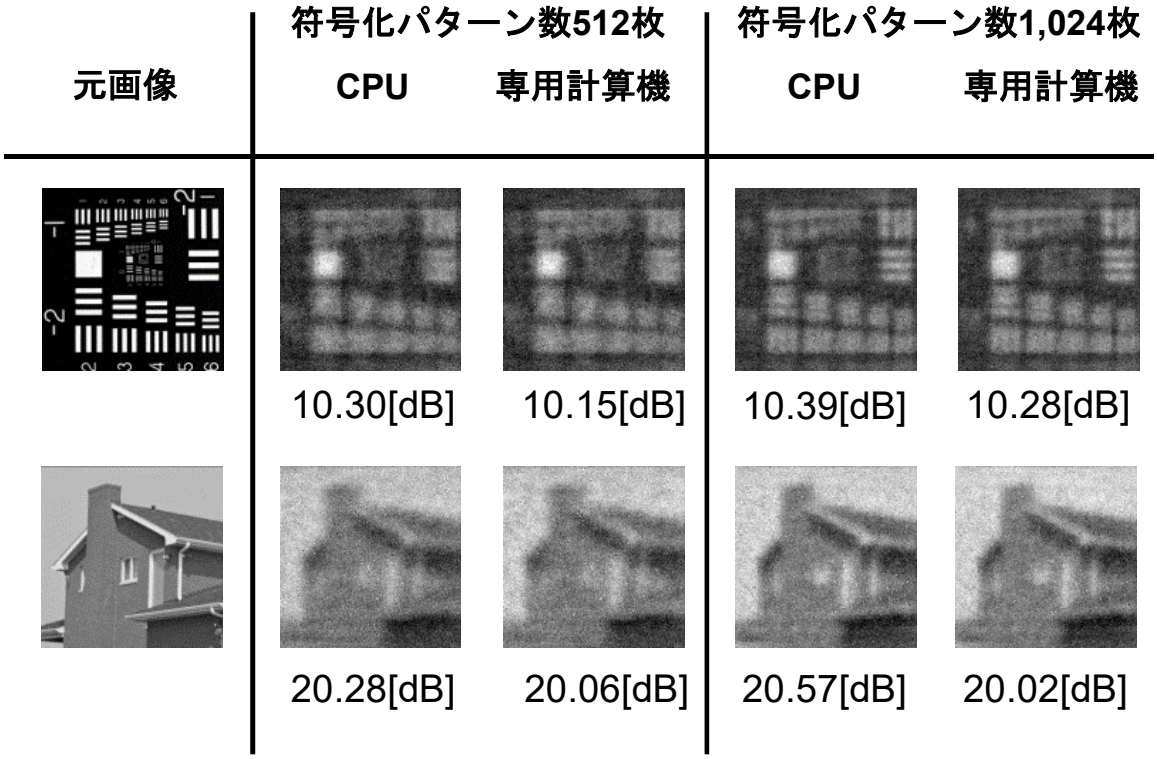


図 5.13: 再構成画像の比較と評価

5.3.2 再構成速度の評価

次に専用計算機で画像再構成した際の速度について，CPU で計算した場合との比較を行った．比較 CPU は組み込み CPU とデスクトップ CPU の 2 種類を用意した．組み込み CPU は，専用計算機の CPU 部分だけを利用し計算を行う．計算環境は，CPU が Quad Core Arm Cortex-A53 MPCore (1.2 GHz)，メモリが 2.0 GB，OS が ubuntu18.04.1LTS，コンパイラが gcc 7.5.0 (最適化オプション -O2）となっている．デスクトップ CPU の計算環境は，CPU が Intel® Core™ i5-4690 (3.5 GHz)，メモリが 32.0GB，OS が Windows 10 Enterprise，コンパイラが Visual Studio

2015 (最適化オプション -O2，SIMD 拡張命令セット有効）となっている．専用計算機内の PL の動作周波数は 200 MHz となっている．組み込み CPU，デスクトップ CPU は計算速度を直接計測した．専用計算機は，計算回路中の計算速度を正確に測定することは難しいため，回路シミュレーションから算出した計算時間に PS，PL 間の通信時間を足したものを計算時間とした．PS，PL 間の通信時間は送受信のみを行う回路を作成し，送受信中の時間を測定したものである．

表 5.5 に再構成計算の速度比較を示す．高速化率は組み込み CPU を基準として計算した．同程度の大きさの組み込み CPU と比較して，符号化パターン数 512 枚の場合は 50 倍，1,024 枚の場合は 100 倍程度高速化できている事が確認できた．また，デスクトップ CPU に対しても符号化パターン数 512 枚の場合 5 倍程度，1,024 枚の場合は 10 倍程度高速化できていることが確認できた．

表 5.5: 再構成計算の速度比較

	符号化パターン数 512		符号化パターン数 1,024	
プロセッサ	計算時間	高速化率	計算時間	高速化率
組み込み CPU	186.05[ms]	1.00	372.03[ms]	1.00
デスクトップ CPU	18.64[ms]	9.98	37.80[ms]	9.84
専用計算機	3.41[ms]	54.56	3.74[ms]	99.47

5.3.3 光学実験と評価

図 5.3 に示した専用計算機を用いた画像再構成システムを実際に構築し，光学実験を行った．図 5.14 に実際の光学系を示す．カメラレンズは MVL50M23 (Thorlabs 社) を用いた．DMD にはフレームレートが 22, 727 Hz の Hi-Speed V-Modules V7000 (Vialux 社) を用いた．そのため，符号化パターンの切り替えには，512 枚の場合は 22.53 ms，1,024 枚の場合は 45.06 ms の時間を要する．

まず，図 5.14 に示した光学系で専用計算機による画像再構成を行い，その結果を PSNR により評価した．再構成結果を図 5.15 に示す．

画素数と同数のパターンを用いた完全なアダマール変換により取得した被写体に対して PSNR を計算することで画質を評価した．比較は 2 章で述べた試作機に実装した再構成手法である DGI と比較した．専用計算機の方が試作機よりも高画質に再構成できることが確認できる．

次に，図 5.14 に示した光学系でのリアルタイム描画実験について述べる．表 5.6 に組み込み CPU，デスクトップ CPU，専用計算機それぞれで再構成画像のリアルタイム表示を行った際のフレームレートを示す．比較は frame per second (fps) で行った．表 5.5 と同様，高速化率は組み込み CPU を基準として計算した．フレームレートの計測は，あるフレームの再構成画像が表示されてから，次のフレームの再構成画像が表示されるまでを 1 フレームにかかる時間とした．デスクトップ CPU で再構成を行う場合には，アナログデジタル変換器には FAD-16H

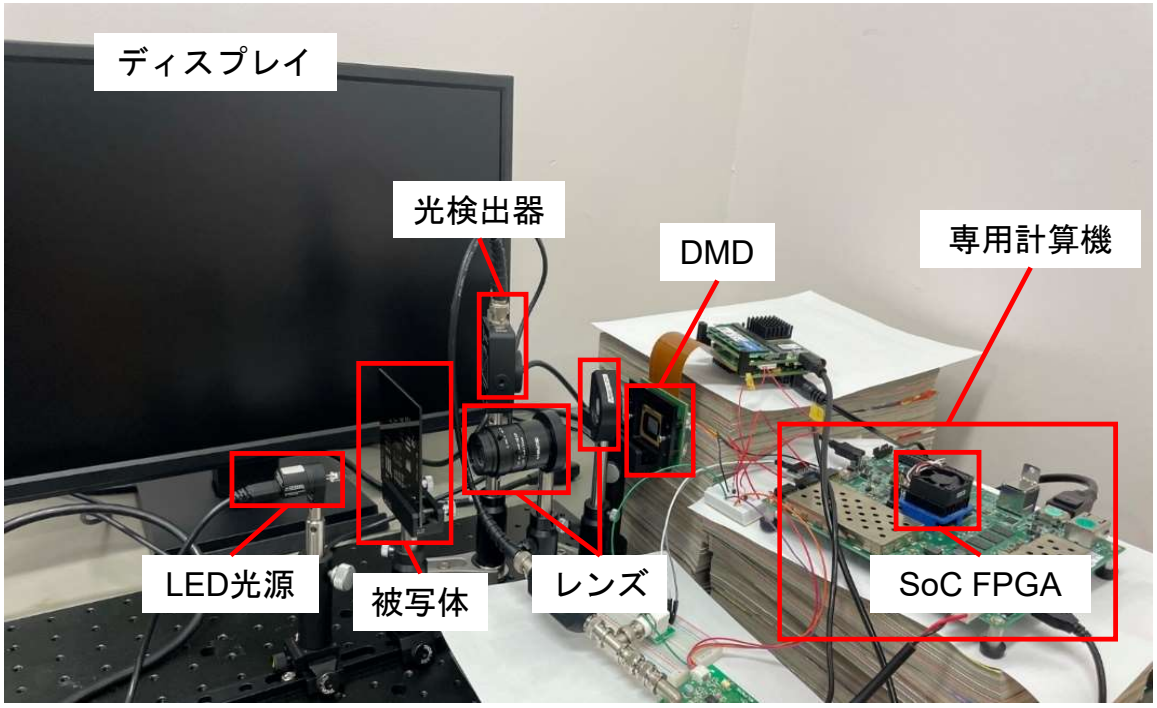


図 5.14: 実際に構築した光学系

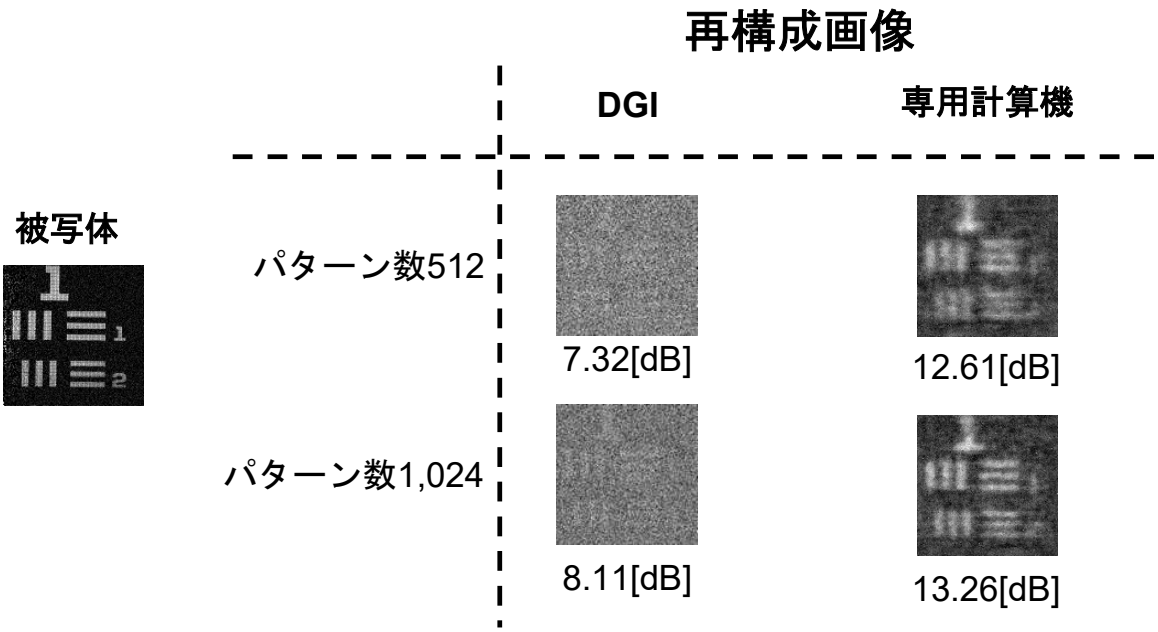


図 5.15: 光学系での再構成結果

シリーズ (ELMOS) を用いた．結果として，組み込み CPU に対して 9 倍程度のフレームレー

表 5.6: リアルタイム表示のフレームレート比較

	符号化パターン数 512		符号化パターン数 1,024	
プロセッサ	フレームレート	高速化率	フレームレート	高速化率
組み込み CPU	5.08[fps]	1.00	2.61[fps]	1.00
デスクトップ CPU	21.63[fps]	4.26	11.06[fps]	4.24
専用計算機	45.36[fps]	8.93	21.89[fps]	8.39

トでリアルタイム表示が可能であることが確認できた．また，デスクトップ CPU よりも高いフレームレートでリアルタイム表示が可能であることも確認できた．

図 5.16 に図 5.14 に示した光学系で行った画像再構成の結果を示す．動的なシーンとして，被写体を水平方向に動かした場合と，奥行き方向に動かした場合で撮影を行った．図 5.16(a) は撮影した被写体であり，被写体の中でも赤い枠で囲った範囲を撮影した．白い矢印は撮影中被写体を動かす方向であり，カメラに対して横向き方向となっている．図 5.16(b)，(c) はそれぞれ符号化パターン 512，1,024 枚で被写体を水平方向に移動させながらリアルタイム表示した各フレームでの再構成画像である．どちらも赤枠で示した被写体を左から右側に移動していく様子が撮影できている．図 5.16(d) はカメラに対して奥行き方向に被写体を移動させた際の被写体の様子である．図 5.16(a) と同様白い矢印は被写体を移動させる方向を示しており，赤枠で示した範囲内を撮影した．図 5.16(e)，(f) はそれぞれ符号化パターン 512，1,024 枚で被写体を水平方向に移動させながらリアルタイム表示した各フレームでの再構成画像である．最初はピントがずれていてぼやけていた像が，フレーム数が変化するにつれてピントが合い像が鮮明になっていく様子が確認できる．

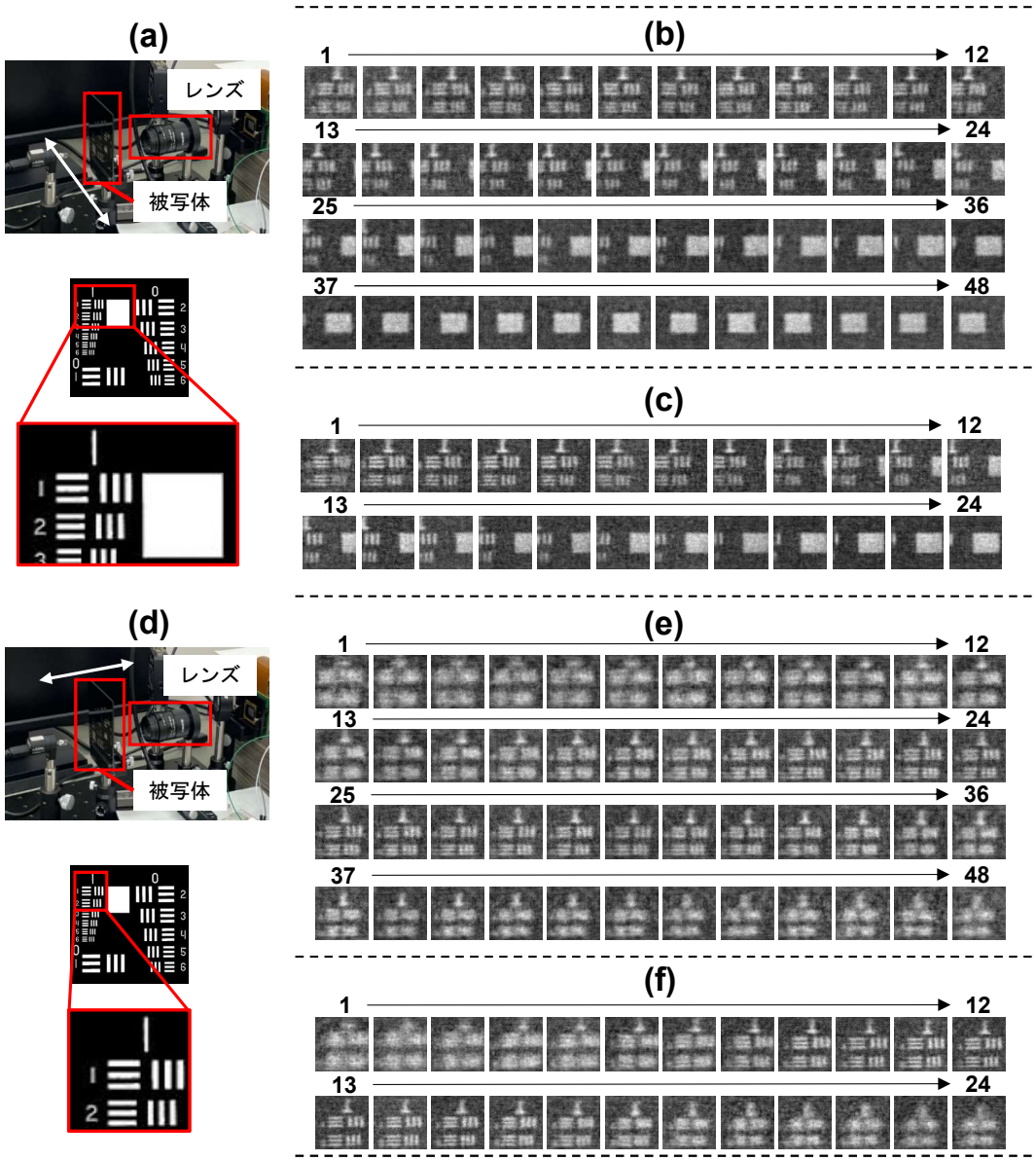


図 5.16: 光学系での動的シーン再構成結果．(a) カメラに対して水平方向に移動させる際の被写体．白い矢印は被写体を動かす向きを示す．被写体は赤枠で示した範囲を左から右方向に移動させる，(b) 符号化パターン数 512 枚で被写体を水平方向に動かし連続撮影した再構成画像像，(c) 符号化パターン数 1,024 枚で被写体を水平方向に動かし連続撮影した再構成画像像，(d) カメラに対して奥行き方向に移動させる際の被写体．白い矢印は被写体を動かす向きを示す．被写体は −1 cm から 1 cm まで 1 秒間で移動させた，(e) 符号化パターン数 512 枚で被写体を奥行き方向に動かし連続撮影した再構成画像像，(f) 符号化パターン数 1,024 枚で被写体を奥行き方向に動かし連続撮影した再構成画像像，(b)，(c)，(e)，(f) で再構成画像上部の数字はフレーム数を示している．

5.3.4 考察

表 5.6 で，専用計算機が 1 フレーム更新するのにかかる時間はフレームレートの逆数より，符号化パターン数 512 枚の場合は 22.05 ms，1,024 枚の場合は 45.68 ms と DMD によるパターン切り替えに要する時間と同じであることがわかる．符号化パターンを投影しながら前のフレームの画像を再構成することで計算時間をパターン切り替えに要する時間で隠蔽しているためこのような結果となっている．そのため理論上，専用計算機のフレームレートは計算機側で達成できる高速化の限界であるといえる．ここで，組み込み CPU は計算時間の方が符号化パターン切り替えに要する時間より長いため計算時間の隠蔽ができないため，フレームレートから算出した 1 フレーム当たりにかかる時間が再構成計算時間とほぼ一致する．しかしデスクトップ CPU は再構成計算時間がパターン切り替えに要する時間より短いのにもかかわらず，パターン切り替えに要する時間と 1 フレームあたりにかかる時間が一致しない．このような結果になった理由を図 5.17 に示す．デスクトップ CPU ではアナログデジタル変換は CPU の API から制御している．そのため，図 5.17(a) に示すように CPU に値を送信している間はアナログデジタル変換することができず，次の符号化パターン列では待機状態となる．結果として符号化パターン列を 2 回に 1 回サンプリングしているようなタイムチャートになる．一方で専用計算機ではアナログデジタル変換を図 5.11 に示した回路でハードウェアレベルの制御を行っている．よって Pmod AD1 から渡される値の受信と，計算回路への値の送信を同時に行うことができるため，連続してアナログデジタル変換が可能である．実際に，表 5.6 でデスクトップ CPU のフレームレートは専用計算機のフレームレートの半分になっている．

専用計算機は現在，実装の面ではデスクトップ CPU よりも優れている．一方で速度の面ではデスクトップ CPU でも計算時間の隠蔽が可能であり専用計算機とのフレームレートの差は少ない．しかし，例えばカラー化や可視光以外の波長など再構成画像の枚数を増加した場合ではデスクトップ CPU では計算時間の隠蔽は不可能になりフレームレートは遅くなっていく．対して，専用計算機ではパターン切り替えに要する時間に対する計算時間にまだ余裕があるので 1 フレーム当たりの再構成画像数を増加してもフレームレートは変わらないはずである．よって今後 1 フレームで複数の波長について画像を再構成する場合において専用計算機はデスクトップ CPU よりも高いフレームレートで再構成ができると考えられる．

5.4 小括

本研究では，小型かつ単体で動作することができるシングルピクセルイメージング専用計算機を開発した．開発した専用計算機は小型ながらデスクトップ CPU よりも高速に再構成計算を行うことができる．また，光学系を実際に構築し，同程度の規模である組み込み CPU はもちろん，デスクトップ CPU よりも高いフレームレートで再構成した画像をリアルタイム表示可能なシステムを実現した．もう一つの課題であった画像サイズや画質についても，4 章で開発した手法を実装することで 128×128 [pixel] と 2 章の試作機よりも大きく，画質の良い画像の再構成を可能にした．今後の展望としては，複数波長で再構成した画像を高速に表示するよう専用計算機と光学システムを改良することなどが挙げられる．

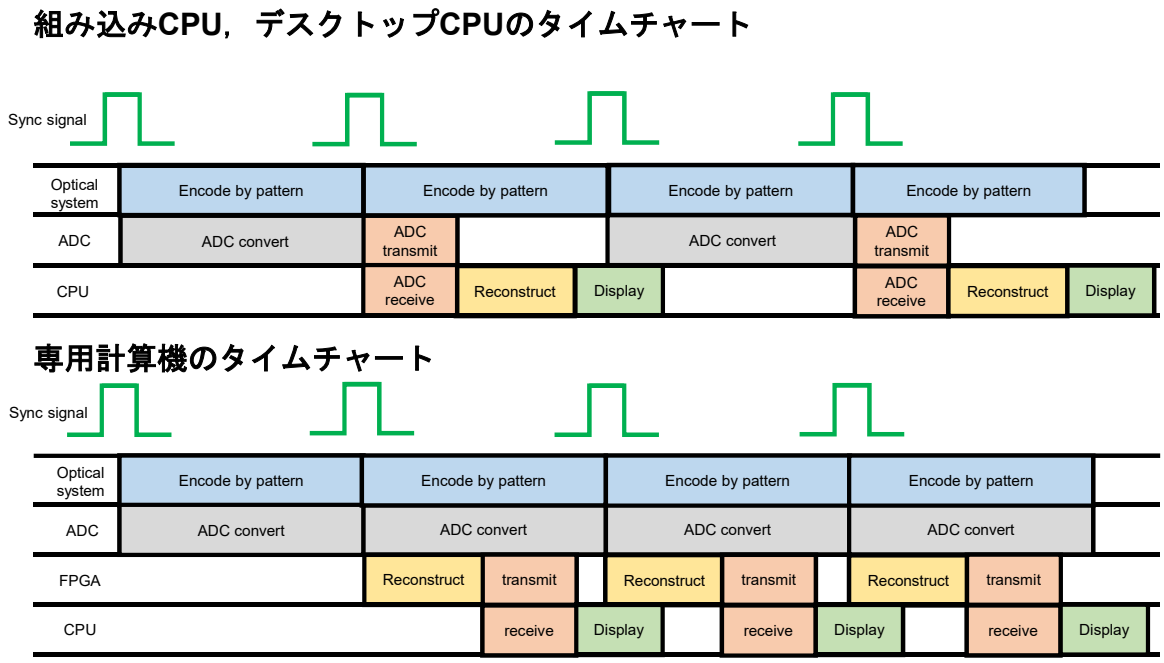


図 5.17: プロセッサごとのリアルタイム表示タイムチャート

結論

本研究では，シングルピクセルイメージングの応用を実用へ近づけるための，小型でシングルピクセルイメージングの画像再構成を計算できる性能を持った専用計算機を FPGA を用いて開発した．2 章で述べた，初めに試作したシングルピクセルイメージング計算回路では FPGA を利用した並列計算回路で小型ながらデスクトップ CPU よりも高速に画像再構成を行えることを示すことができた．試作回路は画質，画像サイズ，単体で動作が困難といった課題があったため，3，4 章では画質について改善する手法の開発を行った．5 章では開発した手法を実装した計算回路を開発し画質，画像サイズについて改善を行いつつ，SoC FPGA を用いることで，単体動作可能，小型かつ試作回路より画質，画素数を改善した専用計算機を開発した．

2 章で提案した FPGA を用いた専用計算回路は，シングルピクセルイメージング再構成計算を FPGA で小型かつ高速に処理できることを示した点に意義がある．3 章で提案した RNN を用いた画像再構成は，シングルピクセルイメージングで符号化パターンの順に測定した光強度を時系列データとして関連付けた点，またその結果としてノイズ除去作用など再構成画像の画質が変化することが確認できた点に意義がある．4 章で提案した最適化パターンを用いた相関計算による画像再構成は，これまでディープラーニングのみで行われ効果が認められていた符号化パターンの最適化をディープラーニングの一部ではなく，最適化手法としてとらえることでディープラーニング以外の再構成手法でも符号化パターンの最適化は画質改善に有効であることを示した点に意義がある．5 章で開発したシングルピクセルイメージング専用計算機は，実用化を志向し小型な計算機を用いてシングルピクセルイメージングでリアルタイムに画像を再構成，表示を可能にした点に意義がある．

本研究で開発したシングルピクセルイメージング専用計算機は小型な計算機でリアルタイム表示を行える点で既に有用であるが，1 フレームにつき複数波長の画像を一度に再構成することで更にその有効性は際立つ．よって，今後はカラー化や赤外波長など複数波長に画像を一度に再構成するように計算回路を改良することや，開発，改良した再生システムを組み込んだアプリケーションや IoT を開発することで専用計算機を活用し，シングルピクセルイメージングアプリケーションの社会実装を目指す．

参考文献

[1] Willard S Boyle and George E Smith, “Charge coupled semiconductor devices,” Bell System Technical Journal, Vol. **49**, No. 4 (1970), pp. 587–593.

[2] Peter JW Noble, “Self-scanned silicon image detector arrays,” IEEE Transactions on electron Devices, Vol. **15**, No. 4 (1968), pp. 202–209.

[3] 成利須川『イメージセンサ技術』電子情報通信学会論文誌 C，Vol. **100**，No. 10 (2017)，pp. 474–482.

[4] Matthew P Edgar, Graham M Gibson, and Miles J Padgett, “Principles and prospects for single-pixel imaging,” Nature photonics, Vol. **13**, No. 1 (2019), pp. 13–20.

[5] Graham M Gibson, Steven D Johnson, and Miles J Padgett, “Single-pixel imaging 12 years on: a review,” Optics Express, Vol. **28**, No. 19 (2020), pp. 28190–28208.

[6] Ming-Jie Sun and Jia-Min Zhang, “Single-pixel imaging and its application in three-dimensional reconstruction: a brief review,” Sensors, Vol. **19**, No. 3 (2019), p. 732.

[7] Sadao Ota, Ryoichi Horisaki, Yoko Kawamura, Masashi Ugawa, Issei Sato, Kazuki Hashimoto, Ryosuke Kamesawa, Kotaro Setoyama, Satoko Yamaguchi, Katsuhito Fujiu, et al., “Ghost cytometry,” Science, Vol. **360**, No. 6394 (2018), pp. 1246–1251.

[8] Baris I Erkmen, “Computational ghost imaging for remote sensing,” Journal of the Optical Society of America A, Vol. **29**, No. 5 (2012), pp. 782–789.

[9] Dongfeng Shi, Kaixin Yin, Jian Huang, Kee Yuan, Wenyue Zhu, Chenbo Xie, Dong Liu, and Yingjian Wang, “Fast tracking of moving objects using single-pixel imaging,” Optics Communications, Vol. **440** (2019), pp. 155–162.

[10] Baoqing Sun, Matthew P Edgar, Richard Bowman, Liberty E Vittert, Stuart Welsh, Adrian Bowman, and Miles J Padgett, “3D computational imaging with single-pixel detectors,” Science, Vol. **340**, No. 6134 (2013), pp. 844–847.

[11] Luana Olivieri, Juan S Totero Gongora, Luke Peters, Vittorio Cecconi, Antonio Cutrona, Jacob Tunesi, Robyn Tucker, Alessia Pasquazi, and Marco Peccianti, “Hyperspectral terahertz microscopy via nonlinear ghost imaging,” Optica, Vol. **7**, No. 2 (2020), pp. 186–191.

[12] Adam Vallés, Jiahuan He, Seigo Ohno, Takashige Omatsu, and Katsuhiko Miyamoto, “Broadband high-resolution terahertz single-pixel imaging,” Optics Express, Vol. **28**, No. 20 (2020), pp. 28868–28881.

[13] Rayko Ivanov Stantchev, Xiao Yu, Thierry Blu, and Emma Pickwell-MacPherson, "Real-time terahertz imaging with a single-pixel detector," Nature communications, Vol. **11**, No. 1 (2020), pp. 1–8.

[14] Li Da Xu, Wu He, and Shancang Li, "Internet of things in industries: A survey," IEEE Transactions on industrial informatics, Vol. **10**, No. 4 (2014), pp. 2233–2243.

[15] Kevin Ashton et al., "That 'internet of things' thing," RFID journal, Vol. **22**, No. 7 (2009), pp. 97–114.

[16] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake, et al., "Internet of Things (IoT): A literature review," Journal of Computer and Communications, Vol. **3**, No. 05 (2015), p. 164.

[17] Jerome LVM Stanislaus and Tinoosh Mohsenin, "Low-complexity FPGA implementation of compressive sensing reconstruction," 2013 International conference on computing, networking and communications (ICNC), 2013, pp. 671–675.

[18] Takashige Sugie, Takanori Akamatsu, Takashi Nishitsuji, Ryuji Hirayama, Nobuyuki Masuda, Hirotaka Nakayama, Yasuyuki Ichihashi, Atsushi Shiraki, Minoru Oikawa, Naoki Takada, et al., "High-performance parallel computing for next-generation holographic imaging," Nature Electronics, Vol. **1**, No. 4 (2018), pp. 254–259.

[19] Takashi Nishitsuji, Yota Yamamoto, Takashige Sugie, Takanori Akamatsu, Ryuji Hirayama, Hirotaka Nakayama, Takashi Kakue, Tomoyoshi Shimobaba, and Tomoyoshi Ito, "Special-purpose computer HORN-8 for phase-type electro-holography," Optics Express, Vol. **26**, No. 20 (2018), pp. 26722–26733.

[20] Sparsh Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," Neural computing and applications, Vol. **32**, No. 4 (2020), pp. 1109–1139.

[21] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, and Tomoyoshi Ito, "Computational ghost imaging using a field-programmable gate array," OSA Continuum, Vol. **2**, No. 4 (2019), pp. 1097–1105.

[22] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal processing magazine, Vol. **29**, No. 6 (2012), pp. 82–97.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, Vol. **25** (2012).

[24] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, and Tomoyoshi Ito, "Single-pixel imaging using a recurrent neural network combined with convolutional layers," Optics Express, Vol. **28**, No. 23 (2020), pp. 34069–34078.

[25] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, and Tomoyoshi Ito, "Optimized binary patterns by gradient descent for ghost imaging," IEEE Access, Vol. **9** (2021), pp. 97320–97326.

[26] Yota Yamamoto, Nobuyuki Masuda, Ryuji Hirayama, Hirotaka Nakayama, Takashi Kakue, Tomoyoshi Shimobaba, and Tomoyoshi Ito, "Special-purpose computer for electroholography in embedded systems," OSA Continuum, Vol. **2**, No. 4 (2019), pp. 1166–1173.

[27] Todd B Pittman, YH Shih, DV Strekalov, and Alexander V Sergienko, "Optical imaging by means of two-photon quantum entanglement," Physical Review A, Vol. **52**, No. 5 (1995), R3429.

[28] Alessandra Gatti, Enrico Brambilla, Morten Bache, and Luigi A Lugiato, "Correlated imaging, quantum and classical," Physical Review A, Vol. **70**, No. 1 (2004), p. 013802.

[29] Fabio Ferri, Davide Magatti, Alessandra Gatti, Morten Bache, Enrico Brambilla, and Luigi A Lugiato, "High-resolution ghost image and ghost diffraction experiments with thermal light," Physical review letters, Vol. **94**, No. 18 (2005), p. 183602.

[30] Jeffrey H Shapiro, "Computational ghost imaging," Physical Review A, Vol. **78**, No. 6 (2008), p. 061802.

[31] F Ferri, D Magatti, LA Lugiato, and A Gatti, "Differential ghost imaging," Physical review letters, Vol. **104**, No. 25 (2010), p. 253603.

[32] Baoqing Sun, Stephen S Welsh, Matthew P Edgar, Jeffrey H Shapiro, and Miles J Padgett, "Normalized ghost imaging," Optics Express, Vol. **20**, No. 15 (2012), pp. 16892–16901.

[33] Teruaki Torii, Yuta Haruse, Shintaro Sugimoto, and Yusuke Kasaba, "Time division ghost imaging," Optics Express, Vol. **29**, No. 8 (2021), pp. 12081–12092.

[34] Xue-Feng Liu, Xu-Ri Yao, Ruo-Ming Lan, Chao Wang, and Guang-Jie Zhai, "Edge detection based on gradient ghost imaging," Optics Express, Vol. **23**, No. 26 (2015), pp. 33802–33811.

[35] Sheng Yuan, Dong Xiang, Xuemei Liu, Xin Zhou, and Pibin Bing, "Edge detection based on computational ghost imaging with structured illuminations," Optics Communications, Vol. **410** (2018), pp. 350–355.

[36] Hong-Dou Ren, Le Wang, and Sheng-Mei Zhao, "Efficient edge detection based on ghost imaging," OSA Continuum, Vol. **2**, No. 1 (2019), pp. 64–73.

[37] Cheng Zhou, Gangcheng Wang, Heyan Huang, Lijun Song, and Kang Xue, "Edge detection based on joint iteration ghost imaging," Optics Express, Vol. **27**, No. 19 (2019), pp. 27295–27307.

[38] Kyuki Shibuya, Katsuhiro Nakae, Yasuhiro Mizutani, and Tetsuo Iwata, "Comparison of reconstructed images between ghost imaging and Hadamard transform imaging," Optical Review, Vol. **22**, No. 6 (2015), pp. 897–902.

[39] 澁谷九輝，水谷康弘，岩田哲郎『循環パターンによるゴーストイメージングの積算回数低減効果』精密工学会学術講演会講演論文集 2013 年度精密工学会秋季大会，2013，pp. 395–396.

[40] Zibang Zhang, Xueying Wang, Guoan Zheng, and Jingang Zhong, "Hadamard single-pixel imaging versus Fourier single-pixel imaging," Optics Express, Vol. **25**, No. 16 (2017), pp. 19619–19639.

[41] Zibang Zhang, Xiao Ma, and Jingang Zhong, "Single-pixel imaging by means of Fourier spectrum acquisition," Nature communications, Vol. **6**, No. 1 (2015), pp. 1–6.

[42] 大関真之『今日からできるスパースモデリング』，`http://www-adsys.sys.i.kyoto-u.ac.jp/mohzeki/Presentation/lecturenote20160822.pdf`, accessed 2022/6/2.

[43] Amir Beck and Marc Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM journal on imaging sciences, Vol. **2**, No. 1 (2009), pp. 183–202.

[44] José M Bioucas-Dias and Mário AT Figueiredo, "A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration," IEEE Transactions on Image processing, Vol. **16**, No. 12 (2007), pp. 2992–3004.

[45] Ori Katz, Yaron Bromberg, and Yaron Silberberg, "Compressive ghost imaging," Applied Physics Letters, Vol. **95**, No. 13 (2009), p. 131110.

[46] Ming-Jie Sun, Ling-Tong Meng, Matthew P Edgar, Miles J Padgett, and Neal Radwell, "A Russian Dolls ordering of the Hadamard basis for compressive single-pixel imaging," en. Scientific Reports, Vol. **7**, No. 1 (2017), p. 3464.

[47] Wen-Kai Yu and Yi-Ming Liu, "Single-pixel imaging with origami pattern construction," Sensors, Vol. **19**, No. 23 (2019), p. 5135.

[48] Frank Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review, Vol. **65**, No. 6 (1958), p. 386.

[49] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," nature, Vol. **323**, No. 6088 (1986), pp. 533–536.

[50] 斎藤康毅『ゼロから作る Deep Learning: Python で学ぶディープラーニングの理論と実装』，オライリー・ジャパン，2016.

[51] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," Neural computation, Vol. **18**, No. 7 (2006), pp. 1527–1554.

[52] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," science, Vol. **313**, No. 5786 (2006), pp. 504–507.

[53] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle, "Greedy layer-wise training of deep networks," Advances in neural information processing systems, Vol. **19** (2006).

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, Vol. **25** (2012).

[55] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, (2014).

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241.

[59] Pietro Berkes and Laurenz Wiskott, "Slow feature analysis yields a rich repertoire of complex cell properties," Journal of vision, Vol. **5**, No. 6 (2005), pp. 9–9.

[60] Meng Lyu, Wei Wang, Hao Wang, Haichao Wang, Guowei Li, Ni Chen, and Guohai Situ, "Deep-learning-based ghost imaging," Scientific reports, Vol. **7**, No. 1 (2017), pp. 1–6.

[61] Tomoyoshi Shimobaba, Yutaka Endo, Takashi Nishitsuji, Takayuki Takahashi, Yuki Nagahama, Satoki Hasegawa, Marie Sano, Ryuji Hirayama, Takashi Kakue, Atsushi Shiraki, et al., "Computational ghost imaging using deep learning," Optics Communications, Vol. **413** (2018), pp. 147–151.

[62] Catherine F Higham, Roderick Murray-Smith, Miles J Padgett, and Matthew P Edgar, "Deep learning for real-time single-pixel video," Scientific reports, Vol. **8**, No. 1 (2018), pp. 1–9.

[63] Fei Wang, Chenglong Wang, Chenjin Deng, Shensheng Han, and Guohai Situ, "Single-pixel imaging using physics enhanced deep learning," Photonics Research, Vol. **10**, No. 1 (2022), pp. 104–110.

[64] Adam Coates, Andrew Ng, and Honglak Lee, "An analysis of single-layer networks in unsupervised feature learning," Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 215–223.

[65] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, (2014).

[66] Martı´ n Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[67] François Chollet et al., "Keras," `https://keras.io`. 2015.

[68] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, et al., "A reconfigurable fabric for accelerating large-scale datacenter services," 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), 2014, pp. 13–24.

[69] Jeffrey L Elman, "Finding structure in time," Cognitive science, Vol. **14**, No. 2 (1990), pp. 179–211.

[70] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," Neural computation, Vol. **9**, No. 8 (1997), pp. 1735–1780.

[71] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, (2014).

[72] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, "Show and tell: A neural image caption generator," Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3156–3164.

[73] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov, "Generating images from captions with attention," arXiv preprint arXiv:1511.02793, (2015).

[74] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," Advances in neural information processing systems, Vol. **27** (2014).

[75] 本山義史，遠藤敏夫，松岡聡，横田理央，福田圭祐，佐藤育郎『低ランク近似行列による CNN における畳み込み演算の最適化』研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. **2017**，No. 25 (2017)，pp. 1–7.

[76] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, Vol. **86**, No. 11 (1998), pp. 2278–2324.

[77] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, (2017).

[78] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” European conference on computer vision, 2016, pp. 525–542.

謝辞

本研究を進めるにあたって，多くの貴重な機会や研究環境を与えてくださった千葉大学　伊藤智義教授に深謝いたします．千葉大学　下馬場朋禄教授には研究指導教員として学部時代から，多くの知見や終始適切な研究指導を賜りました．ここに深謝の意を表します．千葉大学　角江崇助教には光学系関連において多くのご指導を頂きました．厚く御礼申し上げます．千葉大学　小圷成一教授には修士論文および博士論文の主査として，論文審査にあたり多くのご助言を頂きました．感謝申し上げます．千葉大学　白木厚司准教授には副査として，論文審査において細やかな指導を頂き感謝申し上げます．東京理科大学の増田信之教授には，外部審査委員として論文を審査していただき，感謝を申し上げます．博士課程での研究，博士論文の執筆を様々な形で支えていただいた国立研究開発法人　情報通信研究機構電磁波先進研究センターデジタル光学基盤研究室の皆様に感謝申し上げます．研究を行うにあたり，多くの議論を交わした研究室の皆様にも大変お世話になりました．そして，博士課程進学に理解を示し後押ししてくれ，修了まで支えてくれた家族に心より感謝します．簡単ではありますが，以上をもって謝辞とさせていただきます．

査読付き論文（筆頭）

[1] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, “Real-time single-pixel imaging using a system on a chip field-programmable gate array,” *Sci. Rep.*, Vol. **12**, 14097, 2022.

[2] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, “Optimized Binary Patterns by Gradient Descent for Ghost Imaging,” *IEEE Access*, Vol. **9**, pp. 97320-97326, 2021.

[3] 星 郁雄, 天野 洋, 橋本 大志, 田中 大智, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『リカレントニューラルネットワークを用いた学習・推論器の FPGA 実装及び音声識別』電子情報通信学会論文誌 D, Vol. **J103-D**, No. **11**, pp. 808-816, 2020.

[4] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, “Single-pixel imaging using a recurrent neural network combined with convolutional layers,” *Optic Express*, Vol. **28**, No. 23, pp. 8029-8037, 2020.

[5] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, “Computational ghost imaging using a field programmable gate array,” *OSA Continuum*, Vol. **2**, No. 4, pp. 1097-1105, 2019.

査読付き論文（共著）

[1] Tomoyoshi Shimobaba, David Blinder, Tobias Birnbaum, Ikuo Hoshi, Harutaka Shiomi, Peter Schelkens, Tomoyoshi Ito, “Deep-learning computational holography: A review [invited],” *Frontiers in Photonics*, Vol. **3**, pp. 854391, 2022.

[2] Tomoyoshi Shimobaba, Tatsuki Tahara, Ikuo Hoshi, Harutaka Shiomi, Fan Wang, Takayuki Hara, Takashi Kakue, Tomoyoshi Ito, “Real-valued diffraction calculations for computational holography,” *Applied Optics*, Vol. **61**, No. 5, pp. B96-B102, 2022.

[3] Tomoyoshi Shimobaba, Ikuo Hoshi, Harutaka Shiomi, Fan Wang, Takayuki Hara, Takashi Kakue, Tomoyoshi Ito, “Mitigating ringing artifacts in diffraction calculations using average subtractions,” *Applied Optics*, Vol. **60**, No. 22, pp. 6393-6399, 2021.

[4] Tomoyoshi Shimobaba, Shuhei Katsuyama, Takashi Nishitsuji, Ikuo Hoshi, Harutaka Shiomi, Fan Wang, Takashi Kakue, Naoki Takada, Tomoyoshi Ito, “Motion Parallax Holograms Generated from an Existing Hologram,” *Applied Science*, Vol. **11**, No. 7, pp. 2933, 2021.

[5] Daiki Yasuki, David Blinder, Tomoyishi Shimobaba, Yota Yamamoto, Ikuo Hoshi, Peter Schelkens, Takashi Kakue, Tomoyoshi Ito, “Dedicated processor for hologram calculation using sparse Fourier bases,” *Applied Optics*, Vol. **59**, No. 29, pp. 8029-8037, 2020.

[6] Tomoyoshi Shimobaba, Takashi Kakue, Yota Yamamoto, Ikuo Hoshi, Harutaka Shiomi, Takashi Nishitsuji, Naoki Takada, Tomoyoshi Ito, "Hologram generation via hilbert transform," *OSA Continuum*, Vol. **3**, No. 6, pp. 1498-1503, 2020.

[7] Tomoyoshi Shimobaba, Michal Makowski, Takayuki Takahashi, Yota Yamamoto, Ikuo Hoshi, Takashi Kakue, Tomoyoshi Ito, "Reducing computational complexity and memory usage of iterative hologram optimization using scaled diffraction," *Applied Science*, Vol. **10**, No. 3, pp. 1132, 2020.

[8] Tomoyoshi Shimobaba, Takayuki Takahashi, Yota Yamamoto, Ikuo Hoshi, Atsushi Shiraki, Takashi Kakue, Tomoyoshi Ito, "Simple complex amplitude encoding of a phase-only hologram using binarized amplitude," *Journal of Optics*, Vol. **22**, No. 4. pp. 045703, 2020.

[9] Yoshiya Wagatsuma, Tomoyoshi Shimobaba, Yota Yamamoto, Ikuo Hoshi, Takashi Kakue, Tomoyoshi Ito, "Phase retrieval using axial diffracted patterns and ptychographic iterative engine," *Applied Optics*, Vol. **59**, No. 2, pp. 354-362, 2020.

[10] Tomoyoshi Shimobaba, David Blinder, Michal Makowski, Peter Schelkens, Yota Yamamoto, Ikuo Hoshi, Takashi Nishitsuji, Yutaka Endo, Takashi Kakue, Tomoyoshi Ito, "Dynamic range compression scheme for digital hologram using a deep neural network," *Optics Letters*, Vol. **44**, No. 12, pp. 3038-3041, 2019.

国際会議（筆頭）

[1] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, "Implementation of Dedicated Circuit for Scaled Binary Ghost Imaging," *OSA Imaging and Applied Optics Congress 2021 © OSA*, COSI JTh6A.21, Online, 19-23 July 2021.

[2] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, "Image Quality Enhancement of Ghost Imaging by Using Gradient Descent," *International Display Workshops (IDW' 20)*, DES2/AIS4-4L, Online, 9-11 Dec. 2020.

[3] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, "Investigation of Single-Pixel Imaging Using Recurrent Neural Network," *Three Dimensional Systems and Applications (3DSA2019)*, 3DSAp2/3Dp2-13, Sapporo, 27-29 Nov. 2019.

[4] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, "FPGA Implementation for Computational Ghost Imaging," *International Display Workshops (IDW' 18)*, DESp1-3, Nagoya, 12-14 Dec. 2018.

[5] Ikuo Hoshi, Tomoyoshi Shimobaba, Takashi Kakue, Tomoyoshi Ito, "FPGA Circuit Design for Computational Ghost Imaging," *Three Dimensional Systems and Applications (3DSA2018)*, 3DSA-p7, Taiwan, 29-30 Aug. 2018.

国際会議（共著）

[1] Tomoyoshi Shimobaba, Yota Yamamoto, Ikuo Hoshi, Takashi Kakue, Tomoyoshi Ito, "Dedicated processor for holography assisted by deep neural networks," *OSA Frontiers in Optics 2020 + Laser Science (FiO/LS),* FTh2F.1., Online, 13-17 Sep. 2020.

[2] Tomoyoshi Shimobaba, Yota Yamamoto, Ikuo Hoshi, Takashi Kakue, Tomoyoshi Ito, "Holographic classification and regression using binary neural network," *IEEE 18th International Conference on Industrial Informatics (INDIN2020)*, Online, 20-23 July 2020.

[3] Tomoyoshi Shimobaba, David Blinder, Peter Schelkens, Yota Yamamoto, Ikuo Hoshi, Atsushi Shiraki, Takashi Kakue, Tomoyoshi Ito, "Deep-learning-based dynamic range compression for 3D scene hologram," *International Conference on Optics & Electro-Optics XLIII Symposium of Optical Society of India (ICOL-2019)*, pp. 41-44, Uttarakhand India, 19-22 Oct. 2019.

[4] Tomoyoshi Shimobaba, David Blinder, Peter Schelkens, Yota Yamamoto, Ikuo Hoshi, Takashi Kakue, Tomoyoshi Ito, "Deep-learning-assisted hologram calculation via low-sampling holograms," *4th International Conference on Enterprise Architecture and Information Systems (EAIS2019)*, pp. 936-941, Toyama, 7-11 July 2019.

国内会議（筆頭）

[1] 星 郁雄, 涌波 光喜, 市橋 保之, 大井 隆太朗, 『光回折ニューラルネットワークを用いたDDHOE 設計の検討』 電子情報通信学会総合大会, D-11-3, オンライン開催, 15-18 Mar. 2022.

[2] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『リアルタイム表示可能なゴーストイメージングカメラの開発』 Optics & Photonics Japan 2021 (OPJ 2021), 29pC2, 東京, 26-29 Oct. 2021.

[3] 星 郁雄, 下馬場 朋禄, 角江 崇, 杉江 崇繁, 伊藤 智義, 『シングルピクセルイメージングのリアルタイム表示』 第 21 回情報科フォトニクス研究グループ研究会（オンライン合宿）, d1_ hoshi, オンライン, 15-17 Nov. 2021.

[4] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『SoC FPGA を用いたゴーストイメージング動画像システム』 第 20 回情報科学技術フォーラム（FIT2021）, C-007, オンライン, 25-27 Aug. 2021.

[5] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『最急降下法を利用したゴーストイメージング用投影パターンの学習』 Optics & Photonics Japan 2020 (OPJ 2020), 16aBS2, オンライン, 14-17 Nov. 2020.

[6] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『誤差逆伝播法を用いたゴーストイメージング投影パターンの最適化』 第 19 回情報科学技術フォーラム（FIT2020）, I-017, オンライン, 1-3 Sep. 2020.

[7] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『深層学習を用いたシングルピクセルイメージング 画像再構成手法の検討』3 次元画像コンファレンス 2020, 11-3, オンライン, 9-10, July, 2020.

[8] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『リカレントニューラル ネットワークを用いたシングルピクセルイメージングの画像再構成』 レーザー学会第 40 回年次大会, H03-21p-XII-07, 仙台, 20-22, Jan. 2020.

[9] 星 郁雄, 天野洋, 橋本大志, 田中大智, 下馬場朋禄, 角江崇, 伊藤智義, 『再帰型ニューラルネットワークを用いた学習・推論器の FPGA 実装および音声識別』 第 18 回情報科学技術フォーラム (FIT2019), C-016, 岡山, 3-5 Sep. 2019.

[10] 星 郁雄, 下馬場 朋禄, David Blinder, 角江 崇, 伊藤 智義, 『深層学習を用いたデジタルホログラム圧縮手法の研究』第 13 回新画像システム・情報フォトニクス研究討論会, 田町, 6 June 2019.

[11] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『FPGA を用いた大規模な計算機ゴーストイメージ ングの検討』 Optics & Photonics Japan 2018(OPJ2018), 31aP8, 茗荷谷, 30 Oct. - 2 Nov. 2018.

[12] 星 郁雄, 下馬場 朋禄, 角江 崇, 伊藤 智義, 『計算機ゴーストイメージング専用計算回路の設計および FPGA 実装』 第 12 回関東学生研究論文講演会, 日吉, 6 Mar. 2018.

解説記事

[1] 下馬場 朋禄, 星 郁雄, 山本 洋太, 塩見 日隆, 角江 崇, 伊藤 智義, 『小型化を志向したホログラフィックプロジェクション』 光アライアンス, 2020.

特許

[1] 伊藤 智義, 下馬場 朋禄, 星 郁雄, 『プライバシー保護をハードウェアで保証する単光受像素子 IoT デバイス』 特開 2021-100186, 1 July 2021 (特願 2019-230927, 20 Dec. 2019).

外部資金

[1] 公益財団法人双葉電子記念財団 博士後期課程奨学金 Apr. 2020.

受賞

[1] 千葉大学なのはなコンペなのはな賞 (正賞), Dec. 2020.

[2] 千葉大学なのはなコンペ双葉電子記念財団賞, Dec. 2020.

[3] コニカミノルタ光みらい学生奨励金, Nov. 2020.

[4] 千葉大学大学院融合理工学府長表彰, Mar. 2020.

[5] 令和元年度千葉市大学市長賞, Mar. 2020.

[6] 星 郁雄, 天野 洋, 田中 大智, 橋本 大志, 千葉市第 37 回教育・文化・スポーツ等功労者褒賞, Feb. 2020.

[7] レーザー学会第 40 回年次大会論文発表奨励賞, Jan. 2020.

[8] 3DSA2019 Best Poster Award, Dec. 2019.

[9] 星 郁雄, 天野 洋, 田中 大智, 橋本 大志, LSI デザインコンテスト 2019, 準優勝 Mar. 2019.