メニーコアプロセッサにおける画像対応付けアルゴリズムの実装とその応用

博士論文

メニーコアプロセッサにおける画像対応付けアルゴリズムの実装とその応用

情報基礎科学専攻

三浦 衛

Implementation of an image correspondence algorithm on many-core processors and its application

by

Mamoru MIURA

Graduate School of Information Sciences, Tohoku University, 2015

目次

第 1 章

緒言

計算機の発展に伴い，さまざまな機器や場面で高度な情報処理が可能になりつつある．黎明期の電子計算機は部屋を占有するほど巨大であった [1] が，小型化・低価格化によって個人でも所有可能な大きさと価格になり，ノートパソコンやタブレット，スマートフォンのように持ち運べるようになった．現在では，時計型や眼鏡型をはじめとするウェアラブルコンピュータの登場により，計算機は身に付けるものになりつつある [2]．また，家電製品や自動車といった身の回りの機器にも搭載されており，それらの機器にさまざまな機能を付加している．例えば自動車においては，車載コンピュータによって，エンジンを制御することで燃費や排気ガス中の有害物質を抑制したり，ブレーキやエアバッグを制御することで安全性を向上させたりしている [3].

このような計算機の発展の背景のひとつに，計算機の演算処理を司るプロセッサの性能向上が挙げられる．半導体技術の進歩によるプロセスの微細化や，アーキテクチャの改良により，プロセッサのクロック周波数は過去 40 年で 1,000 倍近く向上した．近年では，消費電力や発熱の観点から，単体のコアを複雑に作り込むことで性能を向上させるのではなく，複数のコアをひとつのチップに搭載することで性能向上を目指すマルチコア化が進んできた．さらに現在では，より単純な構造のコアを大量に並べて性能向上を目指すメニーコア化の動向がある．特に，GPU (Graphics Processing Unit) は，安価に広く普及し，かつ比較的容易にプログラミング可能なメニーコアプロセッサとして注目されている．

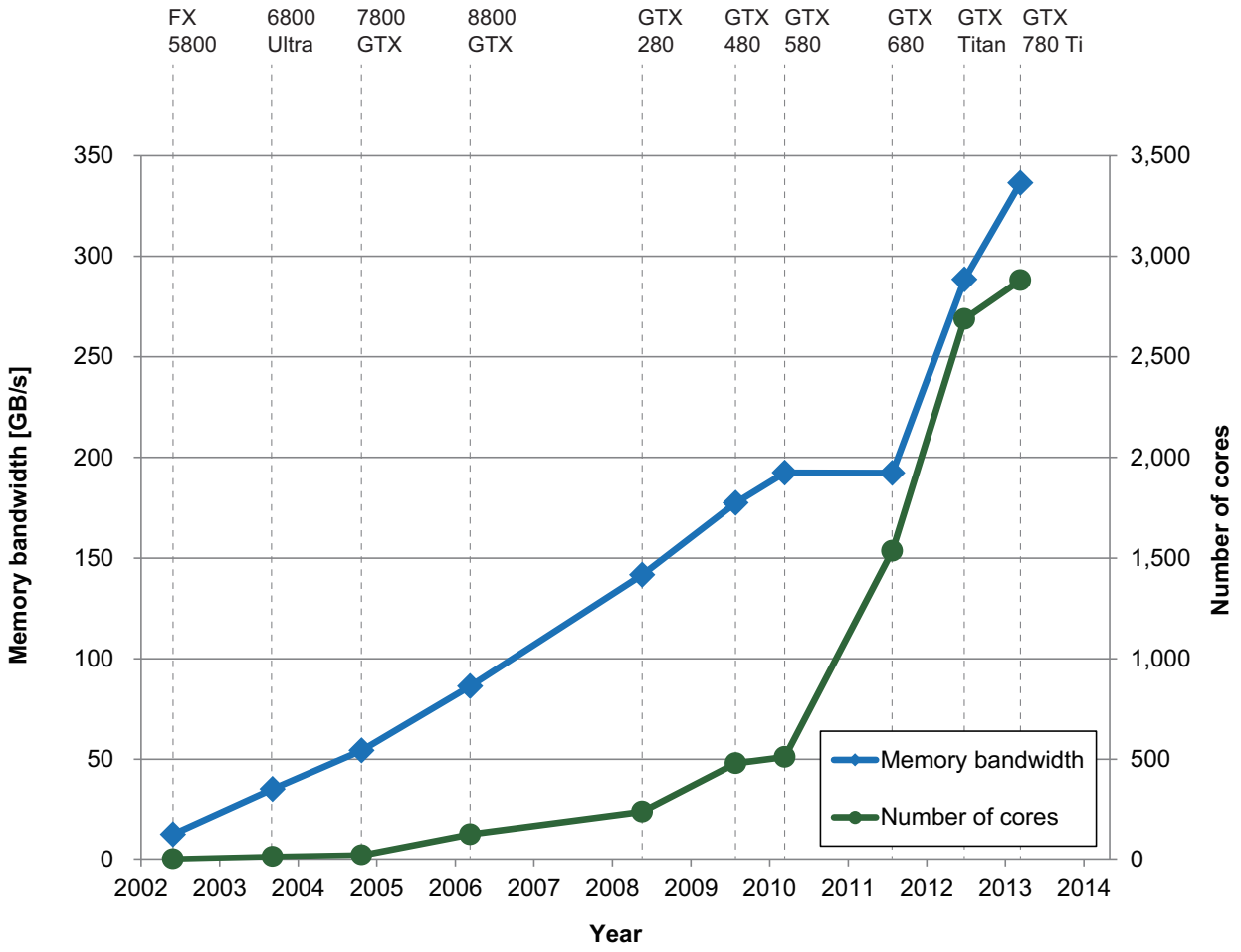


図 1.1　GPU におけるコア数の増加とメモリバンド幅の拡張 [4]

GPU は，計算機のグラフィックス処理を専門とする補助演算装置である．ディスプレイへの描画処理や 3 次元空間映像のレンダリングといった，並列性の高い処理を行うため，大量のコアを持つ SIMD (Single Instruction Multiple Data) 型の演算装置として設計されている．高品質な映像の高速なレンダリングに対する需要の高まりと，ディスプレイの高解像度化に伴い，図 1.1 に示すように，GPU に搭載されるコア数は年々増加し，メモリバンド幅も広くなりつつある．比較的単純な構造を持つためにプロセスの微細化による恩恵を受けやすいこともあり，図 1.2 に示すように，GPU では，CPU に比べて演算性能の向上率が高い． 近年では，この特長をグラフィックス処理だけでなく，汎用計算に利用する GPGPU (General-Purpose computation on Graphics Processing Units) と呼ばれる試みが注目を集めている [6], [7]．これまでに，電磁界解析 [8], [9] や流体シミュレーション [10], [11] をはじめとして，計算量が多

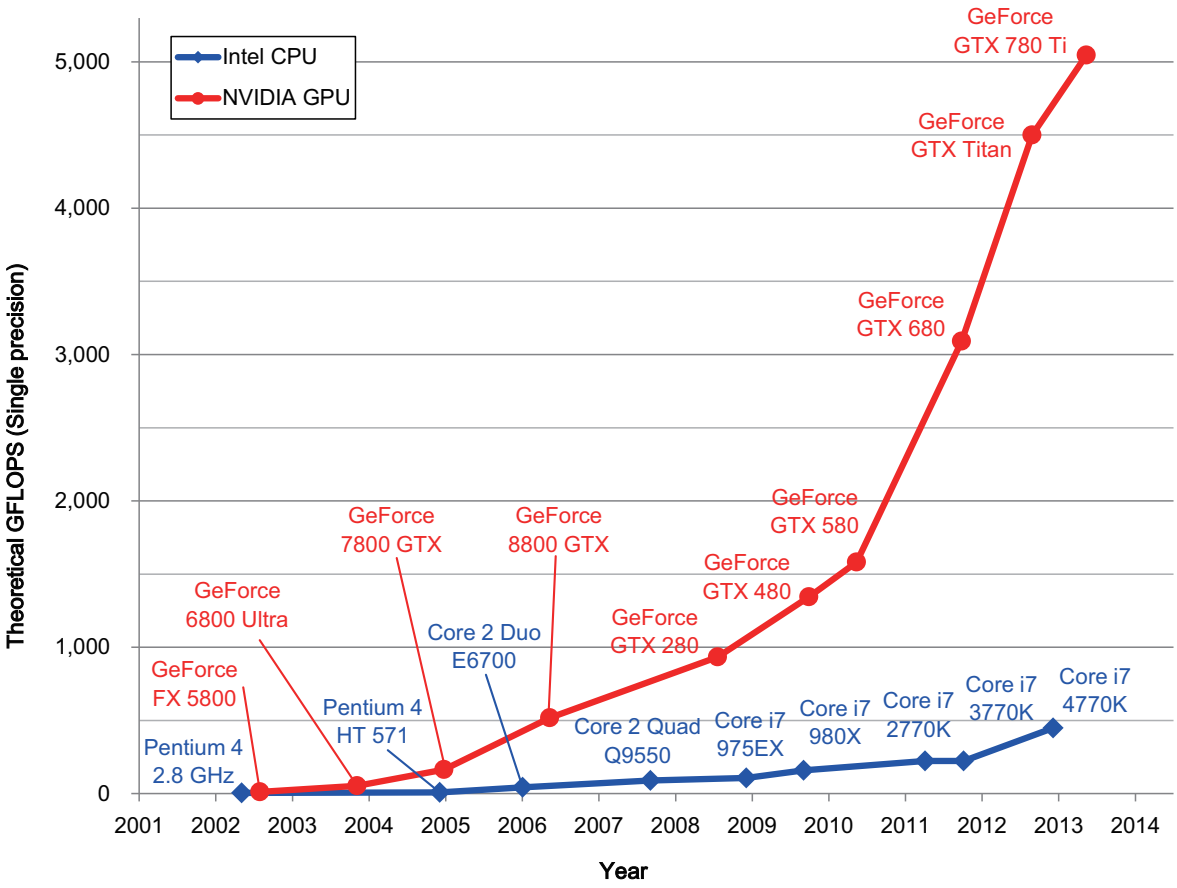


図 1.2　近年における CPU と GPU の演算性能の向上 [4], [5]

い科学計算を中心に，さまざまな実装が提案されている [12], [13]．アルゴリズムの実装に際しては，CUDA (Compute Unified Device Architecture) [13] や OpenCL (Open Computing Language) [14] をはじめとして，GPU を用いた並列コンピューティングのための開発環境が充実しており，従来のプログラミング言語の拡張によってアルゴリズムを GPU に実装できる環境が整っている．また，今日では，グラフィックス処理を補助する拡張カードであるビデオカードがコモディティとして広く普及しているため，ビデオカードに内蔵されている GPU は，容易かつ安価に入手し利用することができる．さらに，Intel や AMD といったプロセッサベンダでは，GPU を CPU と同じチップ上に統合することで，より GPU を利用しやすくしている [15], [16]．このように，GPU の性能や利便性は将来にわたって向上することが期待される．

GPU と CPU では，性能を向上させる戦略と，それを実現するためのアーキテクチャが全

表 1.1　CPU と GPU の比較

プロセッサ	CPU	GPU
モデル名	Intel Core i7-3770K	NVIDIA GeForce GTX 680
コア数	4	1,536
コアクロック [GHz]	3.5	1.0
キャッシュ [KB]	128 (L1)	16 (L1)
	1,024 (L2)	512 (L2)
	8,192 (L3)	
演算性能 [GFLOPS]	224	3,090
メモリバンド幅 [GB/s]	32.0	192.2
演算性能/バンド幅	7.00	16.08

く異なる．表 1.1 に示すように，コア数とそのクロック周波数を比較すると，CPU ではコア数が少なくクロック周波数が高い一方で，GPU はクロック周波数の低いコアを大量に搭載していることがわかる．CPU では，プロセッサのレイテンシ，すなわち，単一のスレッドを可能な限り高速に実行することを重視している．そのため，命令パイプラインの効果を維持するための分岐予測機構や，メモリアクセスのレイテンシを低減するための大容量のキャッシュを搭載している．一方で GPU では，プロセッサのスループットを重視し，数百のスレッドを交互に実行することで性能向上を図っている．広帯域のメモリバンド幅を備え，大量のデータにまとめてアクセスを行う．アクセスのレイテンシはマルチスレッディングで隠蔽するため，キャッシュの容量は CPU に比べて小さい．また，演算性能とメモリバンド幅の比，すなわち 1 バイトあたりに費やせる演算数は GPU の方が高い．つまり，同じデータサイズに対して，GPU は CPU よりも計算コストの高い処理が可能であるといえる．言い換えれば，GPU におけるアルゴリズムの実装は，演算強度（メモリアクセス回数に対する演算回数の比）の高いアルゴリズムの実装に適していると言える．

プロセッサの発展によって実用化が進んだ分野のひとつに，画像処理技術が挙げられる [17]．プロセッサの性能向上によって，画像が持つ膨大な情報量を扱えるようになり，より高度な処

理が現実的な時間内で可能になった．従来では，画像を劣化させている要因を取り除く画質改善や画像の歪みの補正といった，狭義の画像処理と呼ばれる処理が主であったのに対し，これに加えて，画像から文字や物体などの意味のある対象を選別・抽出する画像認識や，被写体の 3 次元位置や形状，寸法などを取得する画像計測といった技術が実用化されつつある．今日では，プロセッサと同様に，カメラもさまざまな電子機器に搭載されつつある．タブレットやスマートフォンといった携帯情報端末のほぼすべてにカメラが搭載されており，エアコンのような家電製品にもセンサとして搭載されつつある [18]．また，カメラの性能向上によって，カメラで撮影できる画像の解像度は大きくなり，撮影するフレームレートは高くなっている．このように，カメラの普及と高性能化に伴い，画像処理技術を応用する機会は増え，より多くの情報量に対して，より高度な処理を，より短い時間で行うための高い演算性能がプロセッサに求められることが予想される．

画像処理において重要となる基礎技術のひとつに，画像対応付けが挙げられる．画像対応付けは，複数の画像間において共通する画素を見つける処理であり，さまざまな画像処理の基本になる．例えば，生体認証では，人物の顔や手のひらの画像を撮影し，予め登録しておいた画像と対応付けることで，個人の認証を行うことができる [19]．また，医用画像処理では，複数のボリュームデータ（3 次元画像）間を対応付けることで，異なる装置を用いて撮影したボリュームデータを重ね合わせたり，病変部位を特定したりすることができる [20]．このような画像対応付けのさまざまな応用の中でも，ステレオビジョンに基づく 3 次元計測は，複数の視点から撮影された画像から被写体の位置や形状といった 3 次元情報を取得する技術であり，対応付けの密度，精度，速度のすべてにおいて高い性能が求められている．ステレオビジョンに基づく 3 次元計測では，異なる視点から撮影された画像間で対応付けを行うことで，三角測量の原理より，対応付けた点の 3 次元空間中の座標を計算する．対応付けた点の数だけ 3 次元点の座標を計算できるため，計測物体の詳細な立体構造を求める場合，画像間で密な対応付けが必要である．また，画像の対応付け精度は，3 次元計測の精度に大きく影響する．特に，3 次元計測精度の向上の観点からは，対応付けの精度にサブピクセル分解能が求められている．さらに，対応付けに要する時間は，3 次元計測における処理時間の大部分を占めるため，高速な計測を行うためには対応付けを高速に行う必要がある．近年では，リアルタイムでの計測が求

められる応用や，大量の入力画像を対応付けて計測を行う応用 [21]–[23] があり，対応付けの処理速度が重要とされている．例えば，FA (Factory Automation) における部品の位置合わせや製品の品質検査，監視カメラや車載カメラを用いた周辺監視への応用では，カメラで取得された画像をリアルタイムで処理することが必須である．

画像対応付けアルゴリズムは，表 1.2 に示すように，特徴ベースのアルゴリズムと領域ベースのアルゴリズムに分けられる．特徴ベースの対応付けでは，画像からエッジやコーナーといった特徴点を検出し，その周辺領域から特徴量と呼ばれるベクトルを計算することで，ベクトル間の距離に基づいて特徴点同士をマッチングする．Scale Invariant Feature Transform (SIFT) [24] や Speeded-Up Robust Features (SURF) [25] をはじめとした，回転やスケール変化，輝度変化などに頑健な特徴量を用いることで画像間をロバストに対応付けることができる一方で，画像中の特徴点のみでしか対応付けを行うことができない．したがって，特徴ベースの対応付けは，画像全域にわたる密な対応付けを行うことが困難であり，密な対応付けが求められるステレオビジョンには不向きである．一方で，領域ベースの対応付けでは，注目点の周辺における局所的な画像ブロックにおいて類似度または相違度をコスト関数として定義し，それらを最大化または最小化することで対応付けを行う．領域ベースの対応付けは，画像上の任意の点について対応付けを行うことが可能であるため，画像全体にわたる密な対応付けをすることで物体の詳細な形状を計測できる．画像ブロックにおける類似度や相違度の計算では，画素ごとの差分を用いる Sum of Absolute Differences (SAD) [17] のような単純なものから，相関関数を用いる Normalized Cross-Correlation (NCC) [17] のような複雑なものがある．一般に，精度と計算コストはトレードオフの関係にあり，精度を向上するためには相応の計算コストがかかる．

領域ベースの画像対応付けアルゴリズムのうち，精度を重視したアルゴリズムのひとつとして，位相限定相関法 (Phase-Only Correlation: POC) に基づくアルゴリズムが知られている [26]．POC に基づくアルゴリズムでは，画像の離散フーリエ変換 (Discrete Fourier Transform: DFT) から得られる位相成分を用いて相関関数を計算し，画像ブロック間の位置ずれ（平行移動量）を推定することで画像の対応付けを行う．同アルゴリズムは，対応付けの精度とノイズに対する高いロバスト性が特徴であり，ステレオビジョンに基づく 3 次元計測へ

表 1.2　代表的な画像対応付けアルゴリズムの比較（◎：高，○：中，△：低）

対応付けアルゴリズム		密度	精度	計算コスト
特徴ベース	SIFT [24]	△	◎	△
	SURF [25]	△	○	○
領域ベース	SAD [17]	◎	△	◎
	NCC [17]	◎	○	○
	POC [26]	◎	◎	△

の応用において，密度および精度の観点から有効であることが示されている [27], [28]．しかしながら，計算コストが大きいという欠点のため，ステレオビジョンへの応用においては，処理速度の観点から応用範囲が限られてしまっていた．POC に基づくアルゴリズムは，他の領域ベースのアルゴリズムに比べ，用いる画像ブロックの大きさが同程度であり，かつ画像ブロックにおける計算コストが高いため，演算強度が比較的高いといえる．この特徴より，メモリバンド幅に比べて高い演算性能を持ち，演算強度の高いアルゴリズムに適合する GPU を用いた実装が有効であると考えられる．また，対応付けのアルゴリズムは，基準点ごと，および画像ブロック内の画素ごとに独立である処理が多く，この点でも GPU の並列演算能力が有効であることが予想される．

以上述べた通り，本研究では，POC に基づく画像対応付けアルゴリズムが GPU の特長に適しているとの見通しから，同アルゴリズムの GPU を用いた実装を提案する．GPU における実装では，パフォーマンスを阻害する要因がいくつか考えられる．まず，対応付け処理では，2 組の局所的な画像ブロックに対する DFT をそれぞれ計算する必要があり，これが処理のボトルネックになると予想される．次に，対応付けにおいて用いる画像ブロックの大きさが，GPU 実装において経験則的に知られている適切な並列処理の単位よりも大きいという問題がある．そのため，単純に画素単位でスレッドに処理を割り当てると，スレッドあたりの使用可能なレジスタ数の制限により同時にスケジューリングされるスレッド数が減少してしまい，パフォーマンスが低下してしまうことが考えられる．また，グローバルメモリへのアクセスを低減させることも課題である．例えば，$1,280 \times 960$ ピクセルのステレオ画像を入力

とし，32 × 15 ピクセルからなる画像ブロックを用いて 10,000 点を対応付けるとすると，画像ブロック中の画素値として 96 万ワードのデータをグローバルメモリから読み出すことになる．これは，ステレオ画像の全画素数の約 4 倍である．これらの問題を解決するため，本論文では，POC に基づく画像対応付けアルゴリズムの GPU に適する実装手法を提案する．具体的には，まず，(i) 2 組の実数列の同時 DFT を用いた DFT の回数削減，および (ii) 回転因子のテーブル実装により，処理のボトルネックである DFT を効率良く実行する．次に，(iii) スレッドブロックのサイズ調整により，プロセッサで同時にスケジューリングされるスレッド数を増加させる．さらに，(iv) ローカルメモリを用いた中間値の保存により，グローバルメモリへのアクセスを低減させる．さまざまな GPU を用いた性能評価を通して，前述の実装を用いることで，素朴な実装に比べて 3.5 倍の高速化が達成できることを示す．また，CPU を用いた実装との性能評価を通して，提案する実装が CPU を用いた実装に比べ 40 倍の高速化が達成できるだけでなく，電力効率の観点からも優れていることを示す．加えて，提案手法を応用し，ステレオビジョンに基づく高速な 3 次元計測システムを構築することで，POC に基づく画像対応付けの GPU 実装の有効性を実証する．まず，リアルタイム 3 次元計測への応用として，人物の顔を 3 次元計測するシステムを構築し，汎用の計算機を用いても，高密度な 3 次元計測がリアルタイムで実現できることを示す [29]–[32]．次に，多視点画像からの 3 次元計測への応用として，汎用のデジタルカメラを用いて撮影された 2 枚の画像から 3 次元計測を行うシステムを構築する [33]–[39]．システムの評価を通して，2 視点からの高密度な 3 次元計測が高精度かつ高速に行えることを示す．

本論文は，以上の内容を取りまとめたものであり，以下に示す 5 章より構成されている．

第 1 章は緒言であり，研究の背景と目的，およびその概要をまとめたものである．

第 2 章では，画像対応付けに関する基礎的考察を行う．はじめに，画像対応付けアルゴリズムについて概説し，本論文で応用として着目するステレオビジョンに基づく 3 次元計測に適するアルゴリズムについて議論する．次に，本論文で用いる POC に基づく画像対応付けアルゴリズムについて述べる．最後に，性能評価を通して，POC に基づく画像対応付けアルゴリズムがステレオビジョンに基づく 3 次元計測において有効であることを示す．

第 3 章では，POC に基づく画像対応付けアルゴリズムの GPU 実装を提案する．まず，

GPU のアーキテクチャと，プログラミングモデルについて概説する．次に，POC に基づく画像対応付けアルゴリズムを GPU に実装する手法について述べる．また，GPU のアーキテクチャに合わせた実装の最適化についても議論する．最後に，性能評価を通して，GPU を用いた実装が，CPU を用いた実装に比べて高速に対応付けを行うことができることを示す．

第 4 章では，ステレオビジョンに基づく 3 次元計測へ提案手法を応用し，その有効性を実証する．本章では，(a) リアルタイムに 3 次元計測を行うシステムと，(b) 汎用デジタルカメラを用いた 2 回の撮影から物体の 3 次元計測を行うシステムを構築する．それぞれのシステム評価を通して，POC に基づく画像対応付けアルゴリズムの GPU 実装がステレオビジョンに基づく 3 次元計測において有効であることを示す．

第 5 章は，結言である．

以上，本論文を企図するところを概説した．

第 2 章

画像対応付けに関する基礎的考察

2.1　まえがき

画像の対応付けは，コンピュータによる画像の解析・計測・理解で重要となる基本的な処理である．画像対応付けに求められる要件は応用によって異なる．本論文では，画像対応付けの応用として，ステレオビジョンに基づく 3 次元計測を取り扱う．ステレオビジョンに基づく 3 次元計測では，異なる視点から撮影された画像の 2 次元座標から，三角測量の原理に基づき，実世界の 3 次元座標を推定する．物体の 3 次元形状を高精度かつ高密度に復元するためには，異なる視点から得られた画像間を高精度かつ高密度に対応付ける必要がある．

本章では，まず，画像対応付けアルゴリズムについて概説し，ステレオビジョンに基づく 3 次元計測への応用に適するアルゴリズムを議論する．次に，画像対応付けの応用として，ステレオビジョンに基づく 3 次元計測について説明し，高精度かつ高密度な画像対応付けが重要となることを述べる．さらに，高精度かつ高密度な画像対応付けアルゴリズムとして本論文で着目する，位相限定相関法に基づく画像対応付けアルゴリズムについて述べる．最後に，3 次元計測を通して画像対応付けアルゴリズムに関する性能評価を行い，位相限定相関法に基づく画像対応付けアルゴリズムを用いることでステレオ画像間を高精度に対応付けることができることを示す．

2.2　画像対応付けアルゴリズムの分類

2 枚の画像について，一方を基準画像，もう一方を参照画像と考えたとき，画像対応付けは，基準画像中の任意の点が参照画像中のどこに存在するかを推定する処理である．画像対応付けアルゴリズムは，特徴ベースのアルゴリズムと領域ベースのアルゴリズムに大別できる．

2.2.1　特徴ベースの画像対応付けアルゴリズム

特徴ベースの対応付けアルゴリズムでは，画像からエッジやコーナーなどの特徴を抽出し，画像間の特徴同士の対応を求める．特徴ベースの対応付けは，(i) 特徴点の検出，(ii) 特徴量の記述，(iii) 特徴量のマッチングの 3 つのステップからなる．まず，(i) 画像中からエッジやコーナーなどの特徴点を検出する．次に，(ii) 特徴点の周囲の画素情報を用いて，特徴量を抽出する．最後に，(iii) 基準画像–参照画像間で，抽出した特徴量同士をマッチングする．

特徴ベースの対応付けアルゴリズムとして, SIFT (Scale Invariant Feature Transform) [24] に基づく画像対応付けアルゴリズムがよく知られている．以下では，特徴ベースの対応付けの一例として，SIFT に基づく画像対応付けアルゴリズムの手順を述べる．まず，(i) 特徴点の検出では，Difference-of-Gaussian (DoG) 処理により，LoG (Scale-normalized Laplacian-of-Gaussian) オペレータ出力値が極大となる画素を特徴点として検出する．次に，(ii) 特徴点を中心とした領域における輝度の勾配情報を，128 次元のベクトルとして抽出する．特徴点の周囲の領域を $4 \times 4 = 16$ ブロックに分割し，それぞれのブロックについて 8 方向（45 度ずつ）の輝度勾配ヒストグラムを作成する．最後に，(iii) 基準画像–参照画像間で，抽出した特徴量同士をマッチングする．特徴量である 128 次元ベクトル間のユークリッド距離を算出し，最も距離が近い点の組を対応付ける．

SIFT は，画像の回転やスケール変化，照明変化などに頑健な特徴量であり，イメージモザイキング [40], [41] や特徴点の追跡 [42]，画像分類 [43] をはじめとして，さまざまな応用が提案されている．SIFT のアルゴリズムを基本とし，特徴点の検出や特徴量の記述を高速化する改良として，Speeded Up Robust Features (SURF) [25] や，バイナリコードを用いた特徴量

の記述を行う Binary Robust Invariant Scalable Keypoints (BRISK) [44] などが提案されている．また，より高いロバスト性の実現を目的として，アフィン変換をしながら対応付けを行う Affine-SIFT (ASIFT) [45] などが提案されている．

特徴ベースの対応付けでは，回転やスケール変化，照明変化などに頑健な特徴量を用いることで，画像間をロバストに対応付けることが可能である．しかし，このような特徴量は，画像中の特徴点のみでしか安定して抽出することができない．したがって，特徴ベースの対応付けでは，画像全域にわたって密な対応付けを正確に行うことは困難である．

2.2.2　領域ベースの画像対応付けアルゴリズム

領域ベースの対応付けでは，注目点の周辺領域について類似度または相違度をコスト関数として定義し，それらを最大化または最小化することで対応付けを行う．一般に，領域ベースの対応付けでは，画像間の局所的な変形を平行移動のみと仮定することが多く，画像の大きな幾何学的歪みに対応できない．しかし，注目点の周囲の輝度パターンを用いて対応付けを行うので，画像上の任意の点について対応付けを行うことが可能である．

いま，基準画像 $I(u,v)$ および参照画像 $J(u,v)$，基準画像 $I(u,v)$ 上の基準点 $\boldsymbol{p}=(p_u,p_v)$ が与えられたとき，基準点 $\boldsymbol{p}$ の対応点の座標 $\boldsymbol{q}=(q_u,q_v)$ を参照画像 $J(u,v)$ 上で求めることを考える．まず，基準画像 $I(u,v)$ における基準点 $\boldsymbol{p}$ の周辺領域を $\boldsymbol{W}$ とする．次に，$\boldsymbol{W}$ を $\boldsymbol{d}=(d_u,d_v)$ だけ動かし，参照画像 $J(u,v)$ と重なった部分について類似度または相違度を計算する．このような計算を，図 2.1 に示すように探索領域 $d_u \leq |s_u|, d_v \leq |s_v|$ において繰り返し，類似度が最大，または相違度が最小となる $\boldsymbol{W}$ の位置を検出する．このとき，$\boldsymbol{W}$ の中心 (p_u+d_u, p_v+d_v) を対応点の座標 $\boldsymbol{q}$ とする．一般に用いられる類似度や相違度として，以下に示す指標が知られている．ただし以下では視認性のため，画像中の座標を $\boldsymbol{n}=(u,v)$ と記す．

- SAD (Sum of Absolute Differences)

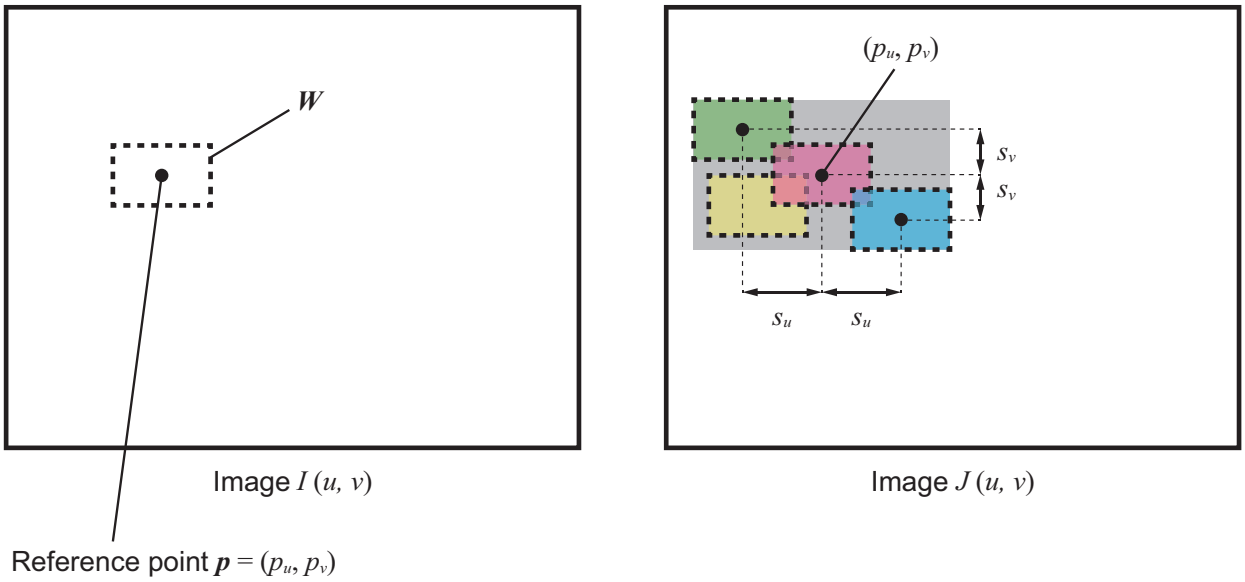


図 2.1　ブロックマッチングに基づく画像対応付けアルゴリズム

SAD は，次式で表される相違度である．

$$R_{\mathrm{SAD}}(\boldsymbol{d}) = \sum_{\boldsymbol{n}\in\boldsymbol{W}} \left| I(\boldsymbol{n}) - J(\boldsymbol{n}+\boldsymbol{d}) \right| \tag{2.1}$$

1 回の SAD 計算では，ある位置 (d_u, d_v) における相違度のみが得られる．対応付けを行う場合には，ある探索領域を設け，その領域内で探索ウィンドウを移動させながら複数回の SAD 計算を行う．そして，相違度 R_{SAD} が最小となる点を選ぶことで対応点を求める．

$$\begin{aligned} \boldsymbol{q} &= \boldsymbol{p} + \boldsymbol{d} \\ &= \boldsymbol{p} + \arg\min_{\boldsymbol{d}} R_{\mathrm{SAD}}(\boldsymbol{d}) \end{aligned} \tag{2.2}$$

- SSD (Sum of Squared Differences)

  SSD は，次式で表される相違度である．

$$R_{\mathrm{SSD}}(\boldsymbol{d}) = \sum_{\boldsymbol{n}\in\boldsymbol{W}} \left(I(\boldsymbol{n}) - I(\boldsymbol{n}+\boldsymbol{d}) \right)^2 \tag{2.3}$$

  SSD に基づくアルゴリズムでは，SAD に基づくアルゴリズムと同様に，複数の SSD 計

算を行い，相違度 R_{SSD} が最小となる点を選ぶことで対応点を求める．

$$\begin{aligned}\boldsymbol{q} &= \boldsymbol{p} + \boldsymbol{d} \\ &= \boldsymbol{p} + \arg\min_{\boldsymbol{d}} R_{\mathrm{SSD}}(\boldsymbol{d})\end{aligned} \tag{2.4}$$

- NCC (Normalized Cross-Correlation)

  NCC は，次式で表される類似度である．

$$R_{\mathrm{NCC}}(\boldsymbol{d}) = \frac{\sum\limits_{\boldsymbol{n}\in\boldsymbol{W}} \overline{I}(\boldsymbol{n})\overline{J}(\boldsymbol{n}+\boldsymbol{d})}{\sqrt{\sum\limits_{\boldsymbol{n}\in\boldsymbol{W}} \overline{I}(\boldsymbol{n})^2 \sum\limits_{\boldsymbol{n}\in\boldsymbol{W}} \overline{J}(\boldsymbol{n}+\boldsymbol{d})^2}} \tag{2.5}$$

  ただし，$\overline{I}(\boldsymbol{n}), \overline{J}(\boldsymbol{n})$ は画像領域 $\boldsymbol{W}$ 内の輝度値の平均値を引いた画像であり，次式で表される．

$$\overline{I}(\boldsymbol{n}) = I(\boldsymbol{n}) - \frac{1}{N_B} \sum_{\boldsymbol{n}\in\boldsymbol{W}} I(\boldsymbol{n}) \tag{2.6}$$

$$\overline{g}(\boldsymbol{n}) = J(\boldsymbol{n}) - \frac{1}{N_B} \sum_{\boldsymbol{n}\in\boldsymbol{W}} J(\boldsymbol{n}) \tag{2.7}$$

  ここで，N_B は画像領域 $\boldsymbol{W}$ 内のピクセル数である．$R_{\mathrm{NCC}}(\boldsymbol{d})$ は $-1 \sim 1$ までの値をとり，値が大きいほど，画像が類似していることを表す．したがって，NCC に基づく対応付けでは，R_{NCC} が最大となるような $\boldsymbol{d}$ を求める．

$$\begin{aligned}\boldsymbol{q} &= \boldsymbol{p} + \boldsymbol{d} \\ &= \boldsymbol{p} + \arg\max_{\boldsymbol{d}} R_{\mathrm{NCC}}(\boldsymbol{d})\end{aligned} \tag{2.8}$$

以上，代表的な領域ベースの画像対応付けアルゴリズムについて述べたが，これらのアルゴリズムで得られる対応点はピクセルレベルの対応である．画像対応付けの応用によっては，対応点の座標を整数値で求めるピクセルレベルの対応付けではなく，実数値で求めるサブピクセルレベルの対応付けが必要となる．サブピクセルレベルの対応付けには，関数フィッティングが用いられることが多い．関数フィッティングでは，相違度が最小となる点，およびその周辺の相違度に対して何らかの関数をフィッティングすることで，サブピクセルレベルの対応付けを行う．SAD に基づくアルゴリズムでは等角直線フィッティングを，SSD や NCC に基づくアルゴリズムではパラボラフィッティングを用いることでサブピクセルレベルの対応付けが高

精度に行えることが知られている [46], [47]．ここでは簡単のため，対応点の探索を水平方向のみに仮定し，相違度が最小の位置における相違度の値を $r(0)$，その隣接位置における相違度の値を $r(-1)$ および $r(1)$ とする．等角直線フィッティングによるサブピクセルの移動量 δ は，次のように推定することができる．

$$\delta = \begin{cases} \frac{1}{2}\frac{r(1)-r(-1)}{r(0)-r(-1)} & r(1) < r(-1) \text{ のとき} \\ \frac{1}{2}\frac{r(1)-r(-1)}{r(0)-r(1)} & \text{その他のとき} \end{cases} \tag{2.9}$$

推定された δ を対応点の座標に加算することで，サブピクセルレベルでの対応点の座標を得る．同様に，パラボラフィッティングによるサブピクセル推定は，次のように表すことができる．

$$\delta = \frac{r(-1) - r(1)}{2r(-1) - 4r(0) + 2r(1)} \tag{2.10}$$

以上述べたサブピクセルレベルの対応付けは，相違度ではなく類似度を用いる場合についても同様である．

2.3　画像対応付けの応用：ステレオビジョンに基づく 3 次元計測

画像対応付けの応用の 1 つに，ステレオビジョンに基づく 3 次元計測が挙げられる．ステレオビジョンに基づく 3 次元計測は，複数のカメラの視線が交わる点として物体の 3 次元位置 $\boldsymbol{X} = (X, Y, Z)$ を計測する．

2.3.1　画像とカメラモデル

まず，カメラで撮影される画像と，画像が撮影される過程のモデルについて述べる．今日の画像処理では，画像は連続した値を持つアナログ画像ではなく，離散的な値を持つディジタル画像として処理されることが一般的である．以下では簡単のため，特に断りのない限り，ディジタル画像を単に画像と記す．画像は 2 次元の画素（ピクセル）配列で構成されており，各画素は 8 ビットの情報量を持ち，$0 \sim 255$ の値を取りうる．画像における特定の画素の位置を記

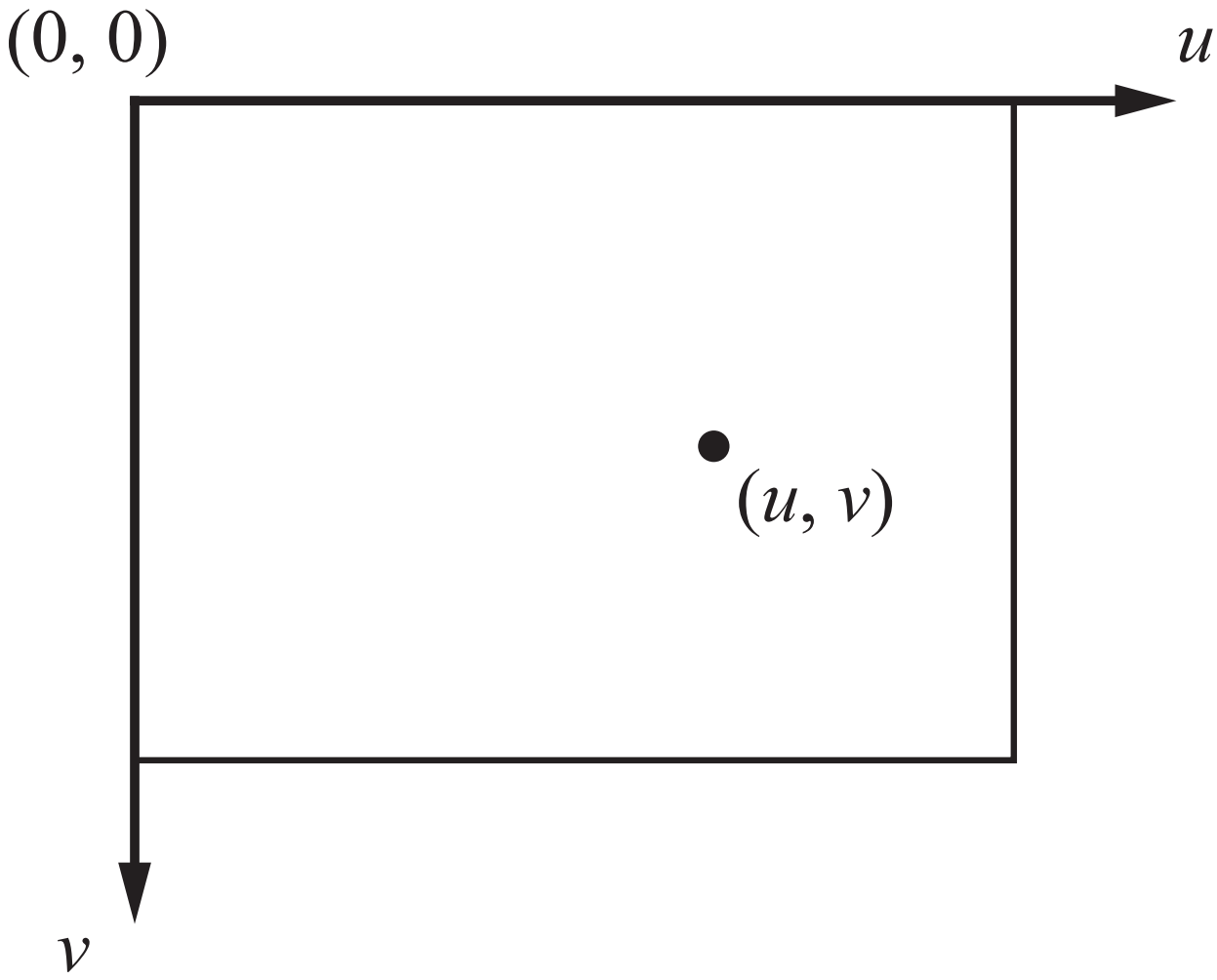


図 2.2　画像座標系．原点が画像の左上にあり，右方向を水平軸，下方向を垂直軸とする．座標系におけるインデックスは 0 オリジン（添字が 0 から始まるインデックス方式）である．

述するため，図 2.2 に示すように画像上の座標 (u, v) を定義する．以下では，この座標系を画像座標系と呼ぶ．

あるシーンをカメラで撮影した場合，3 次元空間中の物体が 2 次元空間に投影され，画像として記録される．このモデルとして，図 2.3 に示すピンホールカメラモデルが広く利用されている．図 2.3 に示すように，画像平面 I から距離 f_l のところに I に平行な面を置き，面上の点 C にピンホールを開ける．物体からくる光はピンホール，すなわち C を通り，画像平面に

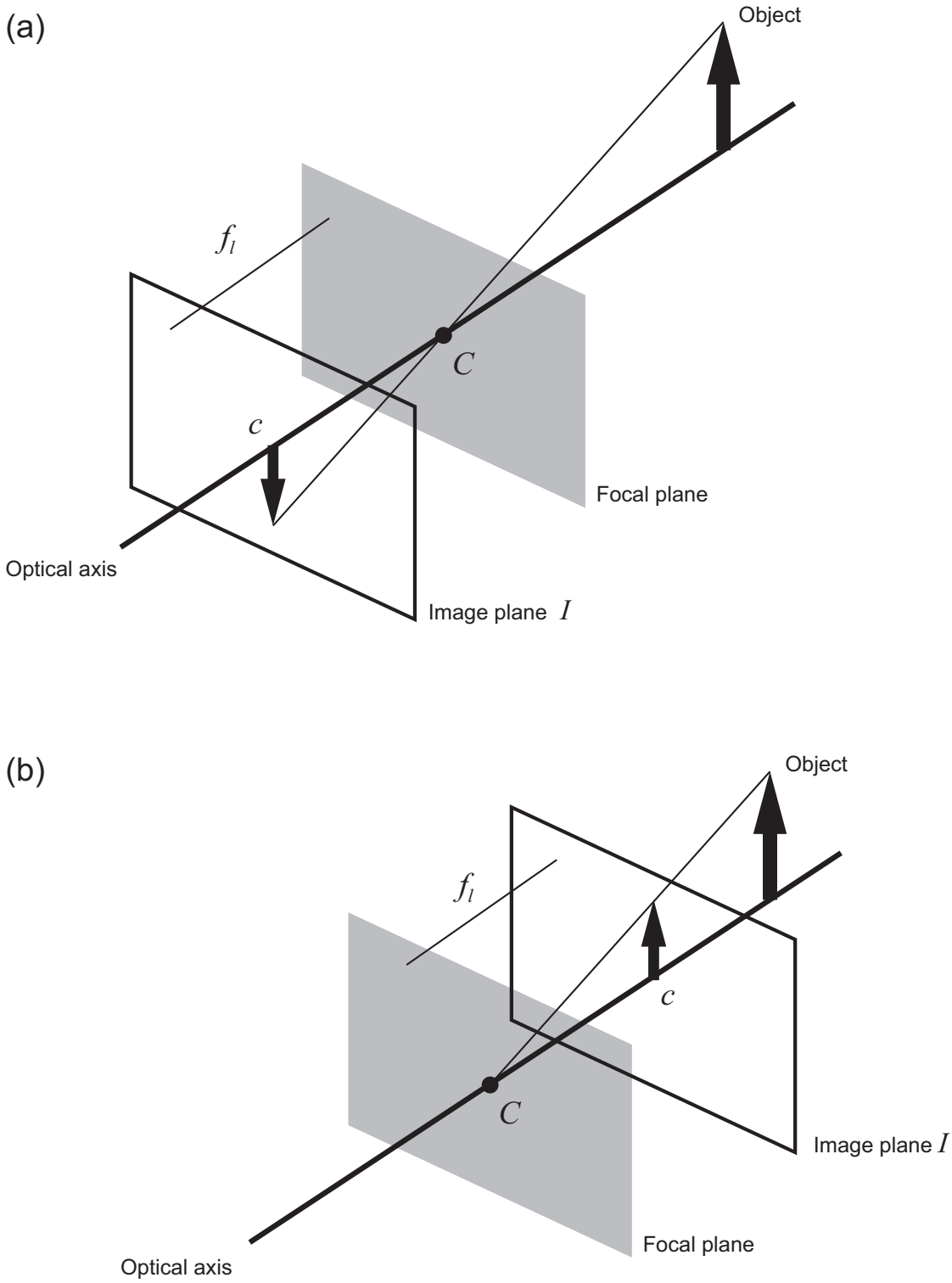


図 2.3　ピンホールカメラモデル．(a) に示すように，画像平面 I から距離 f_l のところに面を置き，点 C にピンホールを開ける．物体からくる光はピンホールである C を通り，画像平面に像を結ぶ．コンピュータビジョンでは，(b) のような仮想の画像平面をレンズ中心の前に置いたモデルがよく用いられる．

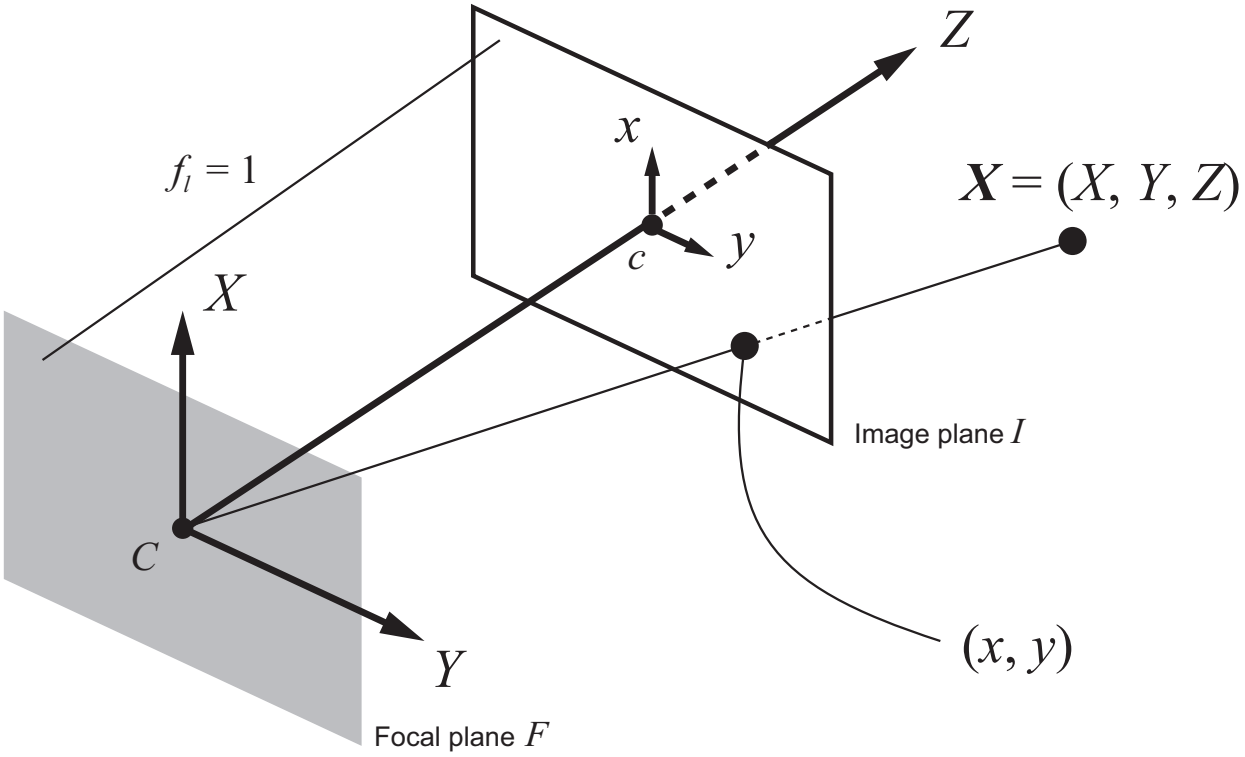


図 2.4　ピンホールカメラモデルにおける座標の定義

像を結ぶ．ここで，点 C を焦点またはレンズ中心，レンズ中心から画像平面までの距離 f_l を焦点距離という．また，点 C を通り画像平面に垂直な線を光軸といい，光軸と画像平面との交点 c を画像中心という．このようなモデルは CCD カメラを正確に記述している．

ピンホールカメラモデルをもとに，3 次元空間中の点 $\boldsymbol{X} = (X, Y, Z)$ が画像に投影されるまでの過程を記述する．まず，図 2.4 に示すように，カメラのレンズ中心 C を原点とする 3 次元空間の座標系を定義する．レンズ中心 C を原点に，光軸を Z 軸とし，右手系となるように X 軸と Y 軸を決める．このような $C - XYZ$ 座標系をカメラ座標系と呼ぶ．また，画像中心 c を原点に，x 軸と y 軸を決める．このとき，カメラ座標系と同様に，光軸と等しい Z 軸を第 3 の軸としたときに，右手系となるように順序を決める．ここで，正規化カメラの概念を導入する．正規化カメラの画像平面は，焦点 C から単位長のところにある．[[[すなわち，焦点距離 f_l は正規化されており，$f_l = 1$ である．このとき，(x, y) で表される座標系を正規化画像座標と呼ぶ．以上の座標系を用いると，3 次元空間内の点の 3 次元座標 $\boldsymbol{X} = (X, Y, Z)$ とそ

の 2 次元座標 (x, y) との間に，次の関係が成り立つ．

$$x = f_l \frac{X}{Z} \tag{2.11}$$

$$y = f_l \frac{Y}{Z} \tag{2.12}$$

この式は以下のように線形表現できる．

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_l & 0 & 0 & 0 \\ 0 & f_l & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.13}$$

$\sim$ は，その両辺が定数倍の違いを許して等しいことを意味する．

3 次元空間中の点 $\boldsymbol{X} = (X, Y, Z)$ が画像平面上の座標 (x, y) に投影される関係を式 (2.13) により記述することができる．しかしながら，$c - xy$ 座標系は，座標の原点が画像中心であり，前述した画像座標系と異なるため，3 次元空間中の点と画像上の画素との対応を扱いにくい．そこで，$c - xy$ 座標系と画像座標系の関係を記述するため，

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.14}$$

または

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.15}$$

なる行列 $\boldsymbol{A}$ を導入する．この行列は，カメラの内部変数（内部パラメータ）と呼ばれ，以下のように書ける．

$$\boldsymbol{A} = \begin{bmatrix} f_p & 0 & u_0 \\ 0 & f_p & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

ここで，f_p はピクセル単位の焦点距離であり，焦点距離 f_l をカメラの撮像素子の幅で割ったものである．また，u_0 および v_0 は，画像中心 c の画像座標系における水平および垂直座標を示す．

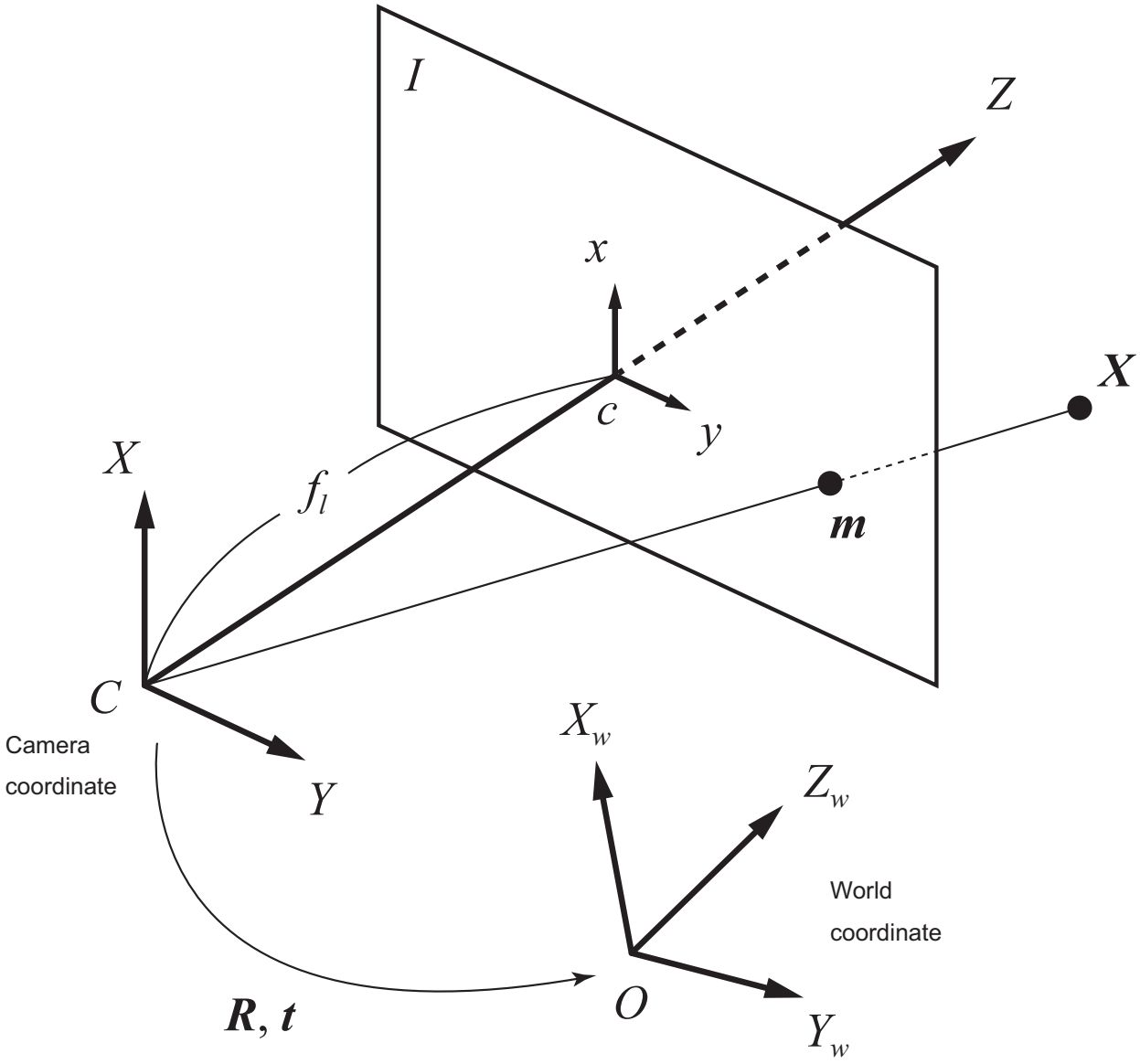


図 2.5　3 次元空間中の点 $\boldsymbol{X}$ が画像に投影される透視投影のモデル．カメラ座標系 $C-XYZ$ と，世界座標系 $O-X_wY_wZ_w$ との位置関係は，回転を表す行列 $\boldsymbol{R}$ と，並進を表すベクトル $\boldsymbol{t}$ を用いて表現される．

これまでは，3 次元空間中の点をカメラ座標系 $C-XYZ$ において記述してきたが，複数のカメラを用いた投影モデルを扱う場合には，図 2.5 に示すような世界座標系における記述を用いることも多い．世界座標系とカメラ座標系の位置関係は，3×3 の行列 $\boldsymbol{R}$ で表される回転と，3×1 のベクトル $\boldsymbol{t}$ で表される並進を用いて記述する．$\boldsymbol{R}$ と $\boldsymbol{t}$ をカメラの外部変数（外部

パラメータ）という．ある点のカメラ座標系における座標 $\boldsymbol{X}_c$ と，その世界座標系における座標 $\boldsymbol{X}_w$ との関係は，

$$\boldsymbol{X}_c = \boldsymbol{R}\boldsymbol{X}_w + \boldsymbol{t} \tag{2.17}$$

あるいは

$$\tilde{\boldsymbol{X}}_c = \boldsymbol{D}\tilde{\boldsymbol{X}}_w \tag{2.18}$$

と表せる．ここで，

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.19}$$

であり，

$$\mathbf{0}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.20}$$

である．また，$\tilde{\boldsymbol{X}}_c$ および $\tilde{\boldsymbol{X}}_w$ はそれぞれ $\boldsymbol{X}_c$ および $\boldsymbol{X}_w$ の同次座標表現である．内部パラメータ $\boldsymbol{A}$, 外部パラメータ $\boldsymbol{R}$, $\boldsymbol{t}$ を用いると，3 次元世界座標における点 $\boldsymbol{X} = (X, Y, Z)$ が画像座標 $\boldsymbol{m} = (u, v)$ に射影される関係は

$$\tilde{\boldsymbol{m}} \sim \boldsymbol{A}\,[\boldsymbol{R}, \boldsymbol{t}]\,\tilde{\boldsymbol{X}} \tag{2.21}$$

と書ける．

2.3.2　エピポーラ幾何

3 次元空間の対象物を複数のカメラに投影した場合には，複数の画像間に特有な幾何が現れる．このような幾何のことをエピポーラ幾何と呼ぶ [48], [49]．エピポーラ幾何は複数のカメラ間の関係を知る上で重要な役割を果たす．複数のカメラから対象物の 3 次元形状を復元するときに，カメラ間の相対的な位置関係や姿勢の情報が必要となる．

いま，3 次元空間中の点 $\boldsymbol{X}$ が視点 $\boldsymbol{C}_1$ の画像に投影されているとする．$\boldsymbol{X}$ と $\boldsymbol{C}_1$ を結ぶ直線が画像面 I_1 と交差してできる点 $\boldsymbol{x}_1$ が 3 次元空間中の点 $\boldsymbol{X}$ の画像平面への投影像である．同様に視点 $\boldsymbol{C}_2$ を考えると，$\boldsymbol{X}$ と $\boldsymbol{C}_2$ とを結ぶ直線が画像面 I_2 と交差してできる点 $\boldsymbol{x}_2$ が空間中の点 $\boldsymbol{X}$ の第 2 の画像面への投影像である．このとき，$\boldsymbol{C}_1$, $\boldsymbol{C}_2$, $\boldsymbol{X}$ の 3 点は空間中に一

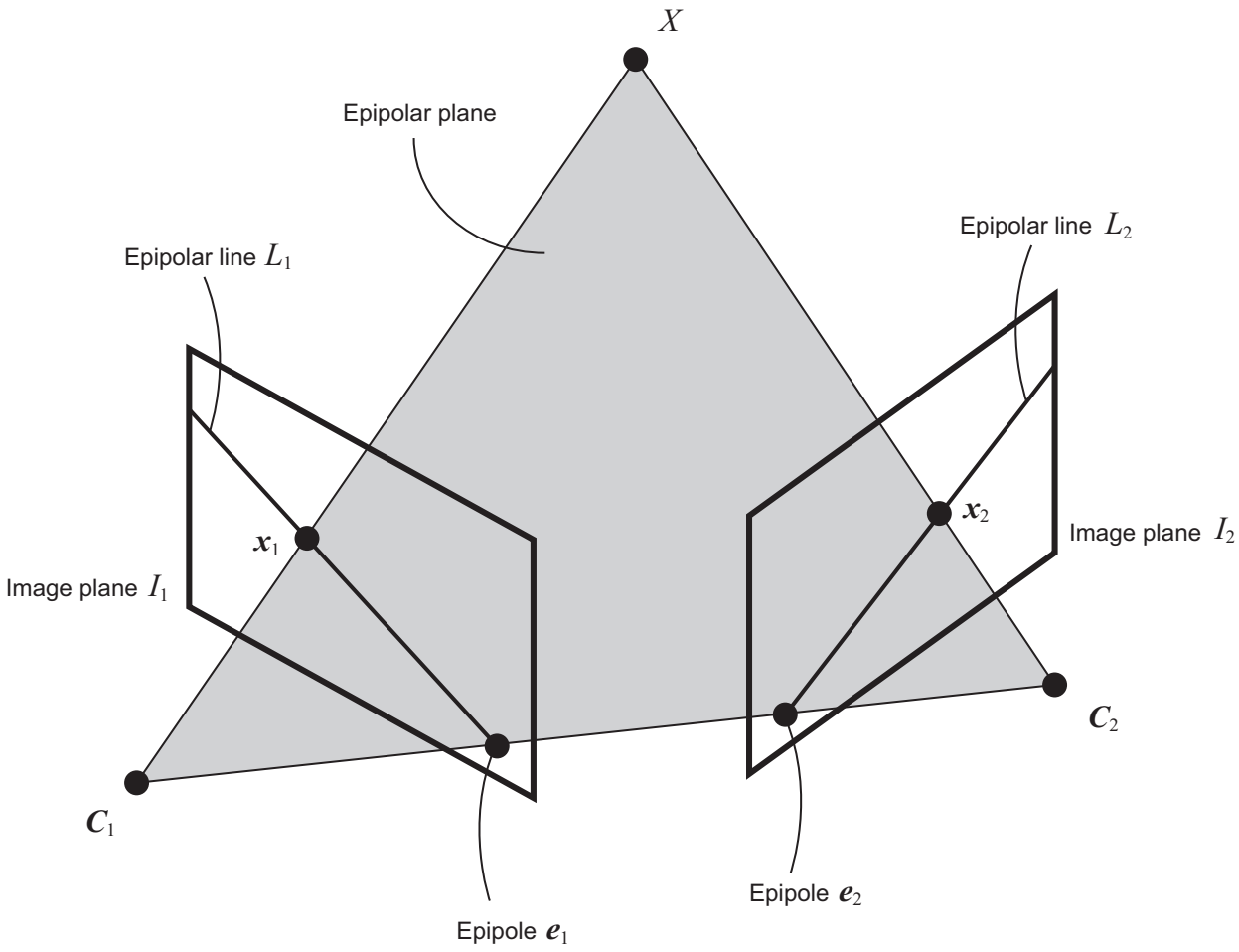


図 2.6　エピポーラ幾何

つの平面を定義する．この平面をエピポーラ平面 (epipolar plane) と呼ぶ．3 次元空間中の点 $\boldsymbol{X}$ の 2 つの画像面への投影像 $\boldsymbol{x}_1, \boldsymbol{x}_2$ もこのエピポーラ平面上にあることは明らかである．エピポーラ平面と画像面とが交差してできる直線 L_1 および L_2 はエピポーラ線 (epipolar line) と呼ばれる．

いま，視点 $\boldsymbol{C}_1$ で画像上の点 $\boldsymbol{x}_1$ が観測されたとき，これに対応する空間中の点は $\boldsymbol{C}_1$ と $\boldsymbol{X}$ を結ぶ直線のどこかに存在するため，この直線は $\boldsymbol{x}_1$ に対応する点の集合であると考えられる．この直線を画像平面 I_2 に投影したものがエピポーラ線 L_2 である．すなわち，第 2 の画像上のエピポーラ線 L_2 は，第 1 の画像上の点 $\boldsymbol{x}_1$ に対応する第 2 の画像上の対応点の集合である．同様に，第 1 の画像上のエピポーラ線 L_1 は，第 2 の画像上の点 $\boldsymbol{x}_2$ に対応する第 1 の画像上の対応点の集合である．このように，エピポーラ先は画像上の点同士の対応関係を表してい

る．また，視点 $\boldsymbol{C}_1$ と $\boldsymbol{C}_2$ が結ぶ直線が二つの画像面と交差してできる点 $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ をエピポールと呼ぶ．画像平面 I_1 上のエピポール $\boldsymbol{e}_1$ は視点 $\boldsymbol{C}_2$ を視点 $\boldsymbol{C}_1$ に投影した像と考えることができる．同様に，画像平面 I_2 上のエピポール $\boldsymbol{e}_2$ は視点 $\boldsymbol{C}_1$ を視点 $\boldsymbol{C}_2$ に投影した像とみなすことができる．

エピポーラ幾何には，カメラの並進および回転の情報が含まれており，画像情報からエピポーラ幾何が得られれば，カメラの 3 次元空間における運動に関する情報を得ることができる．透視投影モデルのカメラの場合，エピポーラ幾何は，基本行列と基礎行列によって表現できる．以下の議論では，視点 $\boldsymbol{C}_2$ におけるカメラ座標を世界座標とし，これを基準に二つのカメラ間の関係を考える．いま，二つのカメラ間の相対的な位置関係を表す回転行列と並進ベクトルを $\boldsymbol{R}_{1\to2}$, $\boldsymbol{t}_{1\to2}$ とし，3 次元空間中の点 $\boldsymbol{X}$ のそれぞれの画像への投影点を正規化画像座標系で $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ とすると，三つのベクトル $\tilde{\boldsymbol{x}}_2$, $\boldsymbol{R}_{1\to2}\tilde{\boldsymbol{x}}_1$, $\boldsymbol{t}_{1\to2}$ はすべてエピポーラ平面内に存在する．そこで，この三つのスカラー 3 重積は 0 となることから，次の式が成り立つ．

$$\tilde{\boldsymbol{x}}_2(\boldsymbol{t}_{1\to2} \times \boldsymbol{R}_{1\to2}\tilde{\boldsymbol{x}}_1) = 0 \tag{2.22}$$

上式を変形すると以下のようになる．

$$\tilde{\boldsymbol{x}}_2^T[\boldsymbol{t}_{1\to2}]_\times \boldsymbol{R}_{1\to2}\tilde{\boldsymbol{x}}_1 = 0 \tag{2.23}$$

ただし，$[\boldsymbol{t}_{1\to2}]_\times$ は次式に示すベクトル $\boldsymbol{t}_{1\to2}$ の各要素からなる 3×3 の行列を表す．

$$[\boldsymbol{t}_{1\to2}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{2.24}$$

さらに，以下のように置き換える．

$$\boldsymbol{E} = [\boldsymbol{t}_{1\to2}]_\times \boldsymbol{R}_{1\to2} \tag{2.25}$$

これより，二つの画像における投影点の正規化画像座標 $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ には，次の関係が成り立つ．

$$\tilde{\boldsymbol{x}}_2^T \boldsymbol{E}\tilde{\boldsymbol{x}}_1 = 0 \tag{2.26}$$

ここで，$\boldsymbol{E}$ は 3×3 の行列で，基本行列と呼ばれる．

上記の関係を画像座標系において表現する．それぞれの画像への投影点の画像座標を $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ とすれば，正規化画像座標 $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ との関係は次のようになる．

$$\tilde{\boldsymbol{m}}_1 \sim \boldsymbol{A}_1 \tilde{\boldsymbol{x}}_1 \tag{2.27}$$

$$\tilde{\boldsymbol{m}}_2 \sim \boldsymbol{A}_2 \tilde{\boldsymbol{x}}_2 \tag{2.28}$$

ここで，$\boldsymbol{A}_1$ および $\boldsymbol{A}_2$ は，視点 $\boldsymbol{C}_1$ および $\boldsymbol{C}_2$ の位置にあるカメラの内部パラメータである．式 (2.26), (2.27), (2.28) より，次の関係が得られる．

$$\tilde{\boldsymbol{m}}_2^T (\boldsymbol{A}_2^{-1})^T [\boldsymbol{t}_{1\to 2}]_\times \boldsymbol{R}_{1\to 2} \boldsymbol{A}_1^{-1} \tilde{\boldsymbol{m}}_1 = 0 \tag{2.29}$$

ここで,

$$\boldsymbol{F} = (\boldsymbol{A}_2^{-1})^T [\boldsymbol{t}_{1\to 2}]_\times \boldsymbol{R}_{1\to 2} \boldsymbol{A}_1^{-1} \tag{2.30}$$

と置き換えることにより，二つの画像における投影点の画像座標 $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ との間に次の関係式が成り立つ．

$$\tilde{\boldsymbol{m}}_2^T \boldsymbol{F} \tilde{\boldsymbol{m}}_1 = 0 \tag{2.31}$$

ここで，$\boldsymbol{F}$ は 3×3 の行列で，基礎行列と呼ばれる．式は，二つの画像上の投影点の間には，それぞれの式で表される拘束条件が存在することを示しており，エピポーラ拘束と呼ばれる．

エピポーラ拘束は，ステレオ画像間の対応付けにおいて重要な性質であり，一方の画像に対するもう一方の画像上の対応点は直線上に制限されることを意味する．したがって，ステレオ画像の対応付けは，図 2.7 に示すように，エピポーラ線上だけを探索すればよいことになり，画像対応付けにおける誤対応の軽減や，計算量の削減が可能になる．

2.3.3　ステレオ視

図 2.8 に示すように，2 つのカメラが平行である場合を平行ステレオと呼ぶ．平行ステレオにおいて，左カメラ画像上の基準点 $\boldsymbol{p} = (p_u, p_v)$ について，右カメラ画像上で対応点 $\boldsymbol{q} = (q_u, q_v)$ が見つかったとき，3 次元空間中の点の座標 $\boldsymbol{X} = (X, Y, Z)$ を次式で計算することができる．

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (u - u_0) B_l / d \\ (v - v_0) B_l / d \\ f_p B_l / d \end{bmatrix} \tag{2.32}$$

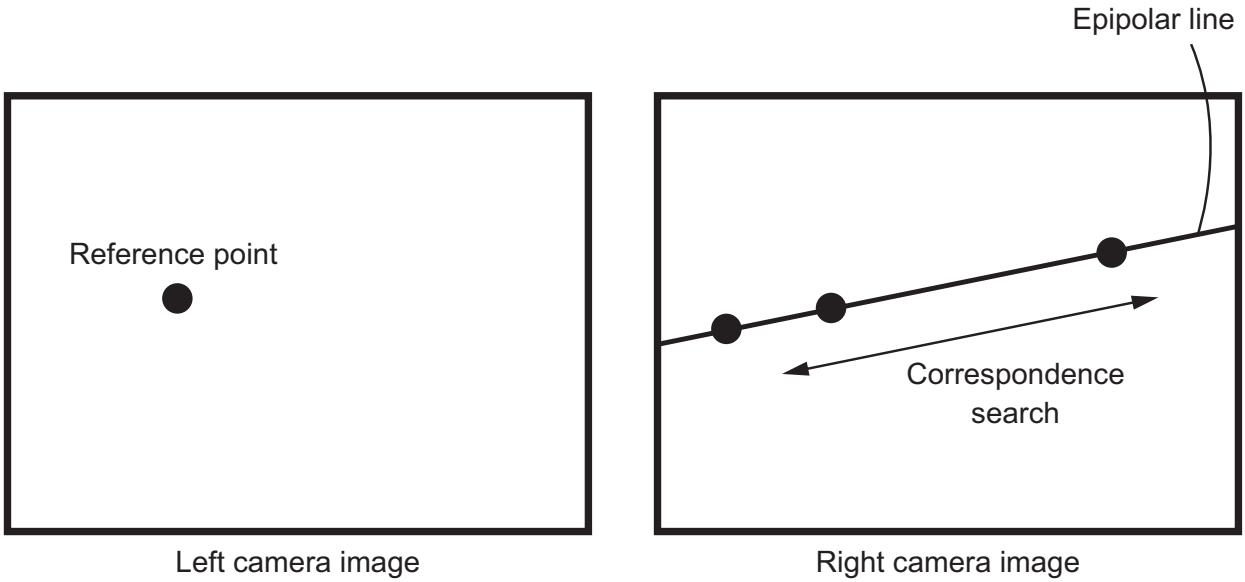


図 2.7　エピポーラ拘束を用いたステレオ画像の対応付け．片方の画像上に配置した任意の基準点 (reference point) について，もう片方の画像上における対応点は，エピポーラ線上に存在する．したがって，対応付け（対応点の探索）は，エピポーラ線上でのみ行えばよい．

ここで，(u_0, v_0) は左カメラの画像中心，(u'_0, v'_0) は右カメラの画像中心，B_l はカメラ間の距離，f_p は左カメラの焦点距離，$d = (u - u_0) - (u' - u'_0)$ は視差である．

一般的なステレオカメラによって得られたステレオ画像では，平行ステレオの仮定が成り立たないことが多い．これに対して，図 2.9 に示すように，左右カメラの画像を共通の仮想画像平面 (virtual image plane) に投影することで，どのようなステレオ画像でも，あたかも平行ステレオで撮影したかのようなステレオ画像に変換することが可能である [48], [50]．この変換はステレオ画像の平行化と呼ばれる．平行化されたステレオ画像では，基準点 $\boldsymbol{p}$ と対応点 $\boldsymbol{q}$ の垂直座標が一致するので，水平方向のみの探索で対応付けを行うことが可能である．これは，計算量の削減や，計算に伴うメモリアクセスの効率化に役立つ．

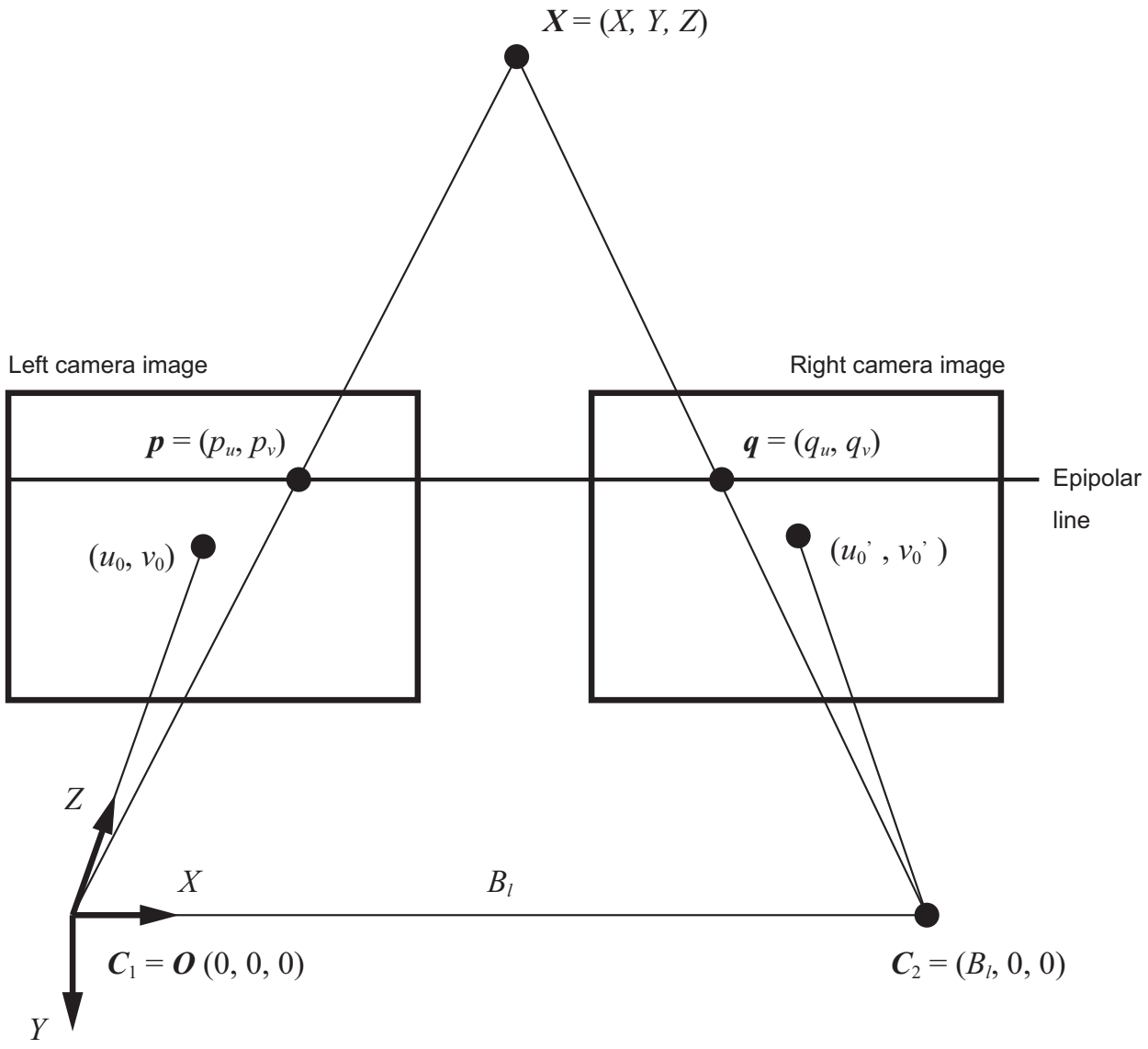


図 2.8　平行ステレオに基づく 3 次元計測

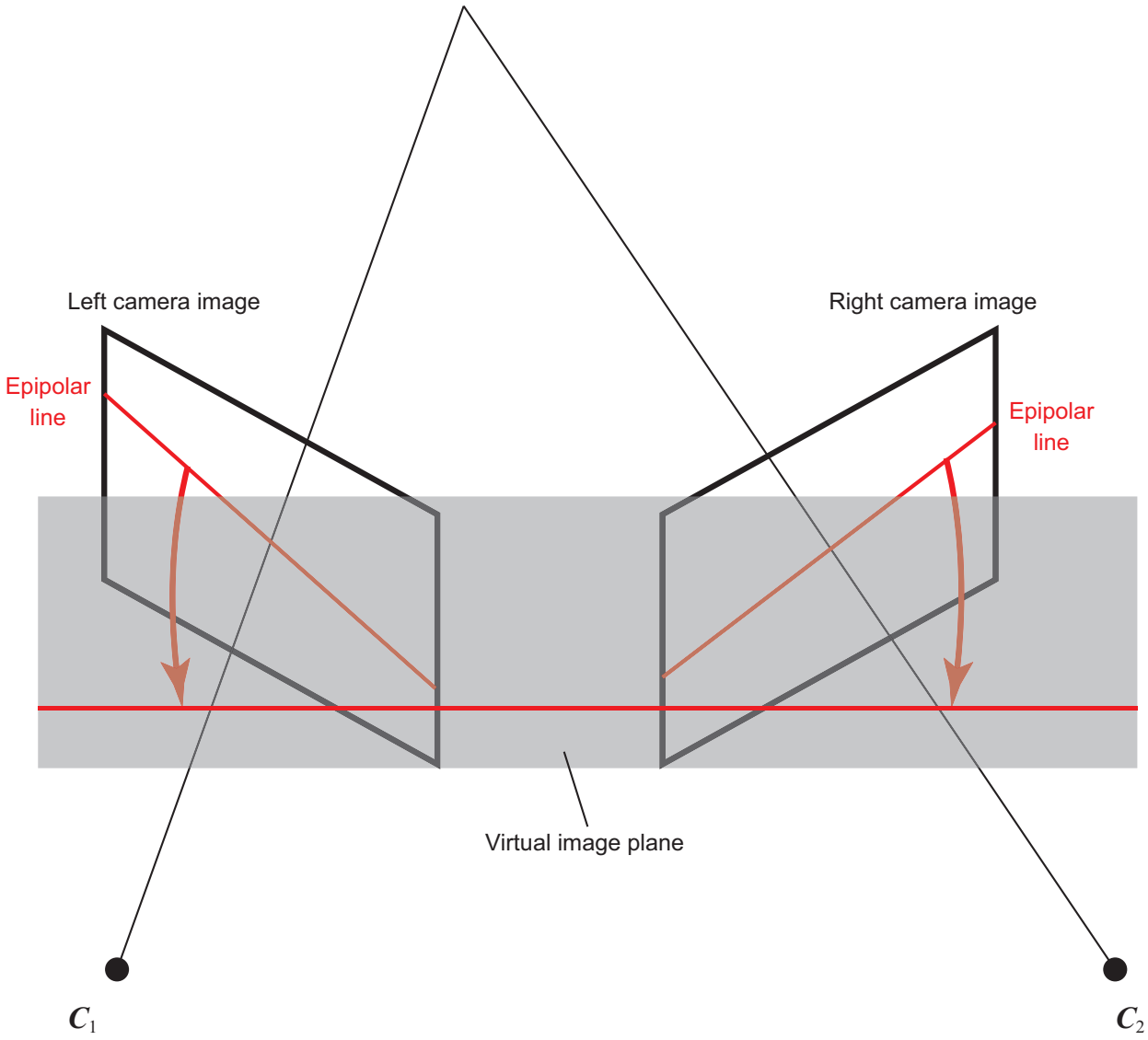


図 2.9　ステレオ画像の平行化

2.3.4　ステレオビジョンに基づく 3 次元計測における画像対応付け

ステレオビジョンに基づく 3 次元計測では，(a) 高密度かつ (b) 高精度で (c) ロバストなステレオ画像の対応付けが求められる．ステレオビジョンに基づく 3 次元計測では，ステレオ画像間で基準点–対応点の 1 組を対応付けるごとに 3 次元点を 1 点だけ計測できる．したがって，ステレオ画像間を高密度に対応付けることができれば，より多くの計測点を得ることができ，物体や空間の詳細な構造を把握することができる．また，式 (2.32) より，3 次元計測の精

度は，視差 d を推定する精度に依存していることがわかる．すなわち，ステレオ画像間を高精度に対応付けることができれば，3 次元計測を高精度に行うことができる．同様に，式 (2.32) より，3 次元計測の分解能は視差 d を推定する分解能に依存している．一般に，ステレオビジョンにおいて，対応付けにより得られる対応点の座標はピクセル精度では不十分であり，サブピクセル分解能での対応付けが要求される．さらに，ステレオカメラを用いて撮影されるステレオ画像には，ぼけ，雑音，ひずみ，画像間の輝度変化など様々な外乱が加わる．これらの外乱に対してロバストな対応付けができない場合，3 次元計測を行う計測対象や環境が著しく制限されてしまう．

これまでに述べたことを踏まえて，ステレオビジョンに基づく 3 次元計測に適する画像対応付けアルゴリズムについて考える．特徴ベースの対応付けでは，特徴点のみ対応付けを行うため，計測対象によっては，ステレオ画像間で対応付けできる点が少なかったり，ある領域に偏っていたりする．一方で，領域ベースの対応付けでは，画像中の任意の点について対応付けを行うことができるため，画像全域にわたって高密度に対応付けを行うことが可能である．したがって，本論文では，物体や空間の詳細な 3 次元計測を行うために，領域ベースの対応付けアルゴリズムに着目する．

2.4　位相限定相関法に基づく画像対応付けアルゴリズム

領域ベースの画像対応付けアルゴリズムの中でも，対応付けの精度とロバスト性に優れる手法として，位相限定相関法に基づく画像対応付けアルゴリズムが知られている [26]．本論文では，対応付けの精度とロバスト性に優れ，かつ演算強度の高いアルゴリズムの実装に向くという GPU の特徴に適するという理由から，位相限定相関法に基づく画像対応付けアルゴリズムに着目する．位相限定相関法では，2 つの信号の位相のみに着目し，位相限定相関関数 (POC 関数) を計算することで信号間の平行移動量を求める [51]．位相限定相関法を画像の局所的なブロックマッチングに適用することで，ステレオ画像間の対応付けが可能となる [52]．さらに，ステレオ画像が平行化されており，画像間の移動ベクトルが水平方向に制限されているとき，水平方向に沿って画像から 1 次元信号を切り出し，平行移動量の推定を行うことで，計算

量を削減できることが知られている [28]．本論文では，ステレオ画像が平行化されていることを前提とし，高精度なステレオ画像間の対応付け手法として 1 次元位相限定相関法に基づく手法を用いる．以下にそれらの概要を述べる．まず，二つの画像が与えられたとき，位相限定相関法を用いてこれら画像間の位置ずれ（平行移動量）を求める方法を述べる．次に，2 次元画像ではなく，1 次元信号に対する平行移動量の推定手法として，1 次元位相限定相関法ついて述べる．最後に，1 次元位相限定相関法を階層的探索と組み合わせた画像対応付けアルゴリズムについて述べる．

2.4.1　位相限定相関関数

いま，$N_1 \times N_2$ ピクセルの二つの画像 $f(n_1, n_2)$ および $g(n_1, n_2)$ が与えられたとき，これら二つの画像の平行移動量（位置ずれ）$\boldsymbol{\delta} = (\delta_1, \delta_2)$ を求めることを考える．ここで，2 次元画像信号の離散空間インデックス（整数）を $n_1 = -M_1, \cdots, M_1$ および $n_2 = -M_2, \cdots, M_2$ とする．ただし，M_1 および M_2 は正の整数であり，$N_1 = 2M_1 + 1$ および $N_2 = 2M_2 + 1$ である．なお，ここでは説明を簡単にするために離散空間のインデックスを正負対称にとり，かつ 2 次元画像信号の大きさ N_1 および N_2 を奇数にしているが，これは必須ではない．すなわち，通常よく用いられるように非負のインデックスを用い，N_1 および N_2 を任意の正の整数に設定するように一般化することが可能である．

画像 $f(n_1, n_2)$ および $g(n_1, n_2)$ の 2 次元離散フーリエ変換 (Discrete Fourier Transform: DFT) を次式で定義する．

$$
\begin{aligned}
F(k_1, k_2) &= \sum_{n_1, n_2} f(n_1, n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} \\
&= A_F(k_1, k_2) e^{j\theta_F(k_1, k_2)} \qquad (2.33) \\
G(k_1, k_2) &= \sum_{n_1, n_2} g(n_1, n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} \\
&= A_G(k_1, k_2) e^{j\theta_G(k_1, k_2)} \qquad (2.34)
\end{aligned}
$$

ここで，$k_1 = -M_1, \cdots, M_1$ および $k_2 = -M_2, \cdots, M_2$ は離散周波数インデックス（整数）であり，回転因子を $W_{N_1} = e^{-j\frac{2\pi}{N_1}}$ および $W_{N_2} = e^{-j\frac{2\pi}{N_2}}$ と定義する．$A_F(k_1, k_2)$ および $A_G(k_1, k_2)$ は振幅スペクトルであり，$\theta_F(k_1, k_2)$ および $\theta_G(k_1, k_2)$ は位相スペクトルを表す．

このとき，$F(k_1, k_2)$ と $G(k_1, k_2)$ の正規化相互パワースペクトル $R(k_1, k_2)$ を次式で与える．

$$\begin{aligned} R(k_1, k_2) &= \frac{F(k_1, k_2)\overline{G(k_1, k_2)}}{|F(k_1, k_2)\overline{G(k_1, k_2)}|} \\ &= e^{j\{\theta_F(k_1,k_2) - \theta_G(k_1,k_2)\}} \end{aligned} \tag{2.35}$$

ここで，$\overline{G(k_1, k_2)}$ は $G(k_1, k_2)$ の複素共役を表す．

位相限定相関関数（POC 関数）$r(n_1, n_2)$ を，正規化相互パワースペクトルの 2 次元逆離散フーリエ変換 (Inverse Discrete Fourier Transform: IDFT) として次のように定義する．

$$r(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1, k_2} R(k_1, k_2) W_{N_1}^{-k_1 n_1} W_{N_2}^{-k_2 n_2} \tag{2.36}$$

二つの画像が類似している場合，POC 関数は，デルタ関数に近い極めて鋭いピークを有する．この相関ピークの位置は画像間の平行移動量 $\boldsymbol{\delta}$ に対応する．一方，相関ピークの高さは画像の類似度の尺度として有用であり，画像照合の用途で用いられる [26].

2.4.2　1 次元位相限定相関関数

これまでの議論では，二つの 2 次元画像を入力とし，それらの間の平行移動量（位置ずれ）を 2 次元方向に求める問題を取り上げた．しかしながら，平行ステレオカメラで撮影された画像間を対応付ける場合は，基準点と対応点の垂直座標が一致しているので，水平方向のみの探索で対応付けを行うことが可能である．平行でないステレオカメラで撮影された画像に対しても，ステレオ平行化と呼ばれる処理を施すことで，あたかも平行ステレオカメラで撮影されたかのような画像に変形することができる [48], [50]．このとき，水平方向に沿ってステレオ画像からそれぞれ 1 次元信号を切り出し，それら 1 次元画像の間でマッチングを行う．すなわち，二つの 2 次元画像間の平行移動量を 2 次元方向で求める問題が，二つの 1 次元画像間の平行移動量を 1 次元方向で求める問題になる．この場合，2.4.1 節で述べた 2 次元の位相限定相関関数（2 次元 POC）ではなく，1 次元の位相限定相関関数（1 次元 POC）を用いることでマッチングが可能である．1 次元 POC を用いることで，2 次元の離散フーリエ変換が不要になるため，計算コストを大幅に削減することができる．また，対応付けにおいて画像から抽出する画像ブロックを任意の形状にとることができ，応用に応じて 2 次元 POC よりも高い精度を実

現できる可能性がある [28].

いま，N 点の 1 次元画像信号 $f(n)$ と $g(n)$ が与えられたとき，$f(n)$ と $g(n)$ の平行移動量 δ を求めることを考える．ただし，1 次元画像信号の離散空間インデックスを $n = -M, \cdots, M$ とする．M は正の整数であり，1 次元画像信号の長さは $N = 2M + 1$ である．なお，ここでは説明を簡単にするために離散空間のインデックスを正負対称にとり，1 次元画像信号の長さ N を奇数にしているが，これは必須ではない．すなわち，通常よく用いられるように非負のインデックスを用い，N を任意の正の整数に設定するように一般化することができる．

1 次元画像信号 $f(n)$ および $g(n)$ の 1 次元離散フーリエ変換を次式で定義する．

$$F(k) = \sum_{n=-M}^{M} f(n) W_N^{kn} = A_F(k) e^{j\theta_F(k)} \tag{2.37}$$

$$G(k) = \sum_{n=-M}^{M} g(n) W_N^{kn} = A_G(k) e^{j\theta_G(k)} \tag{2.38}$$

ただし，$W_N = e^{-j\frac{2\pi}{N}}$ である．ここで，$A_F(k)$ および $A_G(k)$ はそれぞれ $f(n)$ と $g(n)$ の振幅成分であり，$e^{j\theta_F(k)}$ および $e^{j\theta_G(k)}$ はそれぞれの位相成分である．また，離散周波数のインデックスを $k = -M, \cdots, M$ とする．このとき，正規化相互パワースペクトル $R(k)$ を次式で与える．

$$R(k) = \frac{F(k)\overline{G(k)}}{|F(k)\overline{G(k)}|} = e^{j\{\theta_F(k) - \theta_G(k)\}} \tag{2.39}$$

ただし，$\overline{G(k)}$ は $G(k)$ の複素共役を表す．

$f(n)$ と $g(n)$ の 1 次元 POC 関数 $r(n)$ を，正規化相互パワースペクトル $R(k)$ の 1 次元逆離散フーリエ変換として定義する．

$$r(n) = \frac{1}{N} \sum_{k=-M}^{M} R(k) W_N^{-kn} \tag{2.40}$$

二つの画像信号が類似している場合，POC 関数はデルタ関数に近い極めて鋭いピークを有する．この相関ピークの位置は信号間の平行移動量 $\boldsymbol{\delta} = (\delta_1, 0)$ に対応する．一方，相関ピークの高さは画像の類似度の尺度として有用であり，画像照合の用途で用いられる [26].

以上より，与えられた二つの 1 次元画像信号に対して，POC 関数 $r(n)$ の計算値から相関ピークの位置を検出することにより，それらの間に生じている平行移動量 δ を検出することが

できる．一般には，画像間の平行移動量は整数値ではなく，実数値であることが多い．これに対して，POC 関数 $r(n)$ の離散値に対してピークモデルのフィッティングを行うことで，平行移動量 δ を実数値で求めることができる．詳細については (III) 節で述べる．

2.4.3　1 次元位相限定相関関数に基づく平行移動量推定の高精度化

1 次元位相限定相関法による平行移動量推定の高精度化手法として，以下の手法が知られている [26], [28], [52], [53]．本論文では，これらの高精度化手法を用いる．

(I)　入力信号 $f(n)$ と $g(n)$ に対する窓関数の適用
(II)　正規化相互パワースペクトル $R(k)$ に対するスペクトル重み付け
(III)　1 次元 POC 関数 $r(n)$ のピーク値近傍における関数フィッティング
(IV)　複数の 1 次元信号の足し合わせ

それぞれの手法について，以下で説明する．

(I)　窓関数の適用

1 次元 DFT では，信号が循環することを仮定するため，端点での信号の不連続性が問題となる．この不連続性の影響を抑えるために，入力された 1 次元画像信号 $f(n)$ および $g(n)$ に対して窓関数を適用する．本論文では，次式で示される 1 次元ハニング窓を適用する．

$$w(n) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi n}{N}\right) \tag{2.41}$$

(II)　スペクトルの重み付け

一般に，自然画像のエネルギーは低周波成分に集中し，高周波成分のエネルギーは相対的に小さいことが知られている．このため，エイリアシング，雑音，歪みなどの影響により高周波成分に外乱が加わると，画像の高周波成分の SN 比が大幅に劣化する．そこで，信頼性の低い高周波成分の影響を抑制するために，正規化相互パワースペクトル $R(k)$ に対して，低域通過型のスペクトル重み付け関数 $H(k)$ を適用する．本論文では，次式で示される 1 次元ガウス関

数を用いる．

$$H(k) = \exp\left(-\frac{4\ln 2}{\sigma^2}\frac{k^2}{N^2}\right) \tag{2.42}$$

ここで，σ はガウス関数の幅を表す定数である．

(III)　相関ピークモデルのフィッティング

一般に平行移動量 δ は実数値をとるため，1 次元 POC 関数 $r(n)$ のピーク座標はサンプリング格子点の間に存在する．そこで，実際に計算された 1 次元 POC 関数 $r(n)$ の数値データに対して，ピーク形状のモデルをフィッティングすることによって，相関ピークの高さ α および位置 δ を推定する．

式 (2.42) によってスペクトル重み付けされた 1 次元 POC 関数のピークモデルは次式で表される [54].

$$r(n) \approx \frac{\alpha\sigma}{4\sqrt{\pi\ln 2}} \exp\left\{-\frac{\sigma^2 N^2}{16\ln 2}(n-\delta)^2\right\} \tag{2.43}$$

式 (2.43) の対数をとって得られる 2 次曲線を計算値にフィッティングすることで，δ と α を推定することができる．最大ピークの位置が $n = 0$ となるように，離散空間のインデックス n を振り直し，最大ピークを含む近傍 3 点 $r(0)$ および $r(-1)$, $r(1)$ を用いて平行移動量 δ を推定すると次式のようになる．

$$\delta \approx \frac{\ln[r(-1)] - \ln[r(1)]}{2\ln[r(-1)] - 4\ln[r(0)] + 2\ln[r(1)]} \tag{2.44}$$

また，α については，次式から求めることができる．

$$\alpha \approx \frac{4\sqrt{\pi\ln 2}}{\sigma N} \exp\left\{\frac{\sigma^2 N^2}{16\ln 2}\right\} r(0) \tag{2.45}$$

(IV)　複数の 1 次元画像信号の利用

これまで，1 次元画像信号 $f(n)$ と $g(n)$ について，平行移動量 δ を推定する手法を述べた．しかし，実際に撮影した画像において，1 組の 1 次元画像信号のみを用いた場合，高い信頼性で平行移動量を推定することは困難である．そこで，本論文では，複数の 1 次元画像信号の組

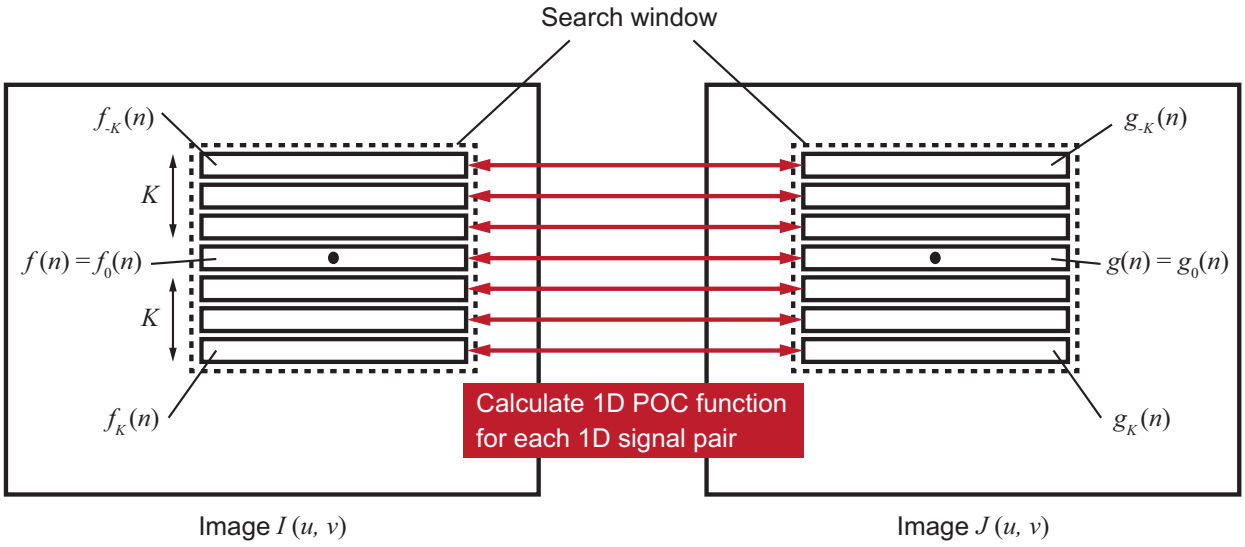


図 2.10　複数の 1 次元画像信号の利用

を近傍から抽出し，それらの POC 関数を統合することによって，信頼性の高い平行移動量推定を行う．

まず，図 2.10 に示すように，1 次元画像信号 $f(n)$ について，$f(n)$ を中心に画像の垂直方向に沿って $2K+1$ 本の 1 次元画像信号を抽出し，$f_{-K}(n)$, $f_{-(K-1)}(n)$, $\cdots$, $f_0(n)$, $\cdots$, $f_{K-1}(n)$, $f_K(n)$ とする．ただし，$f_0(n) = f(n)$ である．これらはすべて N 点の画像信号である．$g(n)$ についても同様に，$g_{-K}(n)$, $g_{-(K-1)}(n)$, $\cdots$, $g_0(n)$, $\cdots$, $g_{K-1}(n)$, $g_K(n)$ を抽出する．ここで，$g_0(n) = g(n)$ である．このように画像から取り出された 1 次元画像信号の集合を，本論文では探索ウィンドウと呼ぶ．

1 次元画像信号 $f_i(n)$ と $g_i(n)$ から計算できる 1 次元 POC 関数を $r_i(n)$ とする．ただし，$i = -K, \cdots, K$ である．これら $L = 2K+1$ 個の 1 次元 POC 関数の平均として，1 次元平均 POC 関数 $r_{\mathrm{ave}}(n)$ を次式で定義する．

$$r_{\mathrm{ave}}(n) = \frac{1}{L} \sum_{i=-K}^{K} r_i(n) \tag{2.46}$$

実際の実装においては，計算量を削減するため，$f_i(n)$ および $g_i(n)$ の離散フーリエ変換 $F_i(k)$ および $G_i(k)$ から計算される正規化相互パワースペクトル $R_i(k)$ を平均する．平均された正

規化相互パワースペクトル $R_{\mathrm{ave}}(k)$ を次式で定義する．

$$R_{\mathrm{ave}}(k) = \frac{1}{L}\sum_{i=-K}^{K} R_i(k) \tag{2.47}$$

このとき，$r_{\mathrm{ave}}(n)$ は $R_{\mathrm{ave}}(k)$ の 1 次元 IDFT であり，次式が成り立つ．

$$r_{\mathrm{ave}}(n) = \frac{1}{N}\sum_{k=-M}^{M} R_{\mathrm{ave}}(k) W_N^{-kn} \tag{2.48}$$

このように，あらかじめ $R_{\mathrm{ave}}(k)$ を計算してから，これを 1 次元 IDFT して $r_{\mathrm{ave}}(n)$ を計算することができる．この計算法は，1 次元 IDFT を行う回数を必ず 1 回にすることができるため，$R_i(k)$ の 1 次元 IDFT を個別に行う場合よりも計算コストを大幅に抑えることができる．

2.4.4　粗密戦略を用いた画像対応付けアルゴリズム

これまでの議論では，2 つの信号間の平行移動量が信号の長さに比べて十分に小さいということを想定していた．しかし，これはステレオカメラで撮影された画像に対しては一般的に成り立たないことが多い．この問題を解決しロバストな対応付けを実現するため，本論文では，1 次元 POC を用いた平行移動量推定を粗密探索と組み合わせる．ステレオ画像の解像度を段階的に下げることで階層画像を生成し，解像度を下げた上位層から平行移動量推定を開始して下位層へと移動していく．その様子を図 2.11 に示す．いま，画像 $I(u,v)$ における基準点の座標を $\boldsymbol{p} = (p_u, p_v)$ とし，画像 $J(u,v)$ における $\boldsymbol{p}$ の対応点 $\boldsymbol{q} = (q_u, q_v)$ を探索することを考える．ただし，ステレオ画像ペア $I(u,v)$ および $J(u,v)$ はステレオ平行化されているとする．すなわち，基準点 $\boldsymbol{p}$ と対応点 $\boldsymbol{q}$ の垂直座標は等しく，$p_v = q_v$ である．以下に，そのアルゴリズムの処理手順を説明する．

入力

- ステレオ画像ペア $I(u,v)$ および $J(u,v)$
- 画像 $I(u,v)$ 上の基準点 $\boldsymbol{p} = (p_u, p_v)$

出力

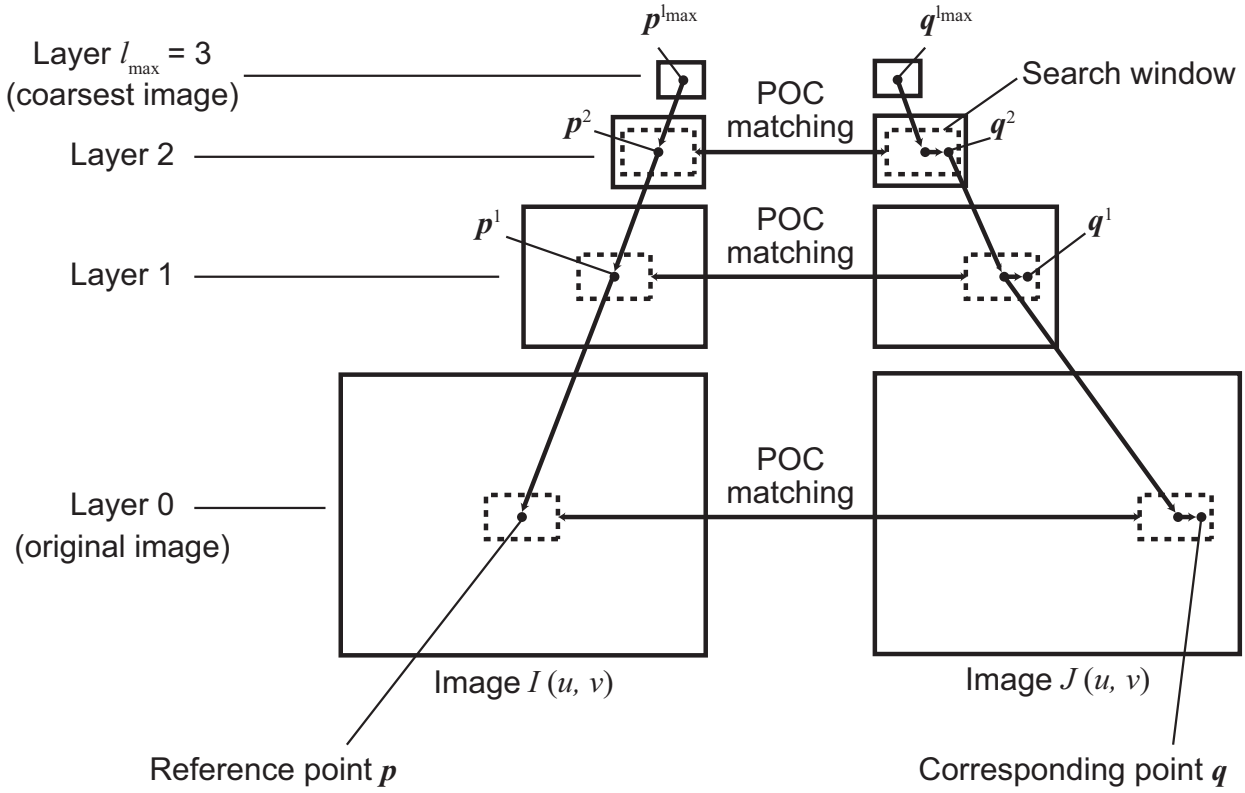


図 2.11　階層画像を用いた粗密探索

- 画像 $J(u,v)$ 上の対応点 $\boldsymbol{q} = (q_u, q_v)$

処理手順

[画像ピラミッドの生成]

Step 1

まず，入力であるステレオ画像 $I(u,v)$ および $J(u,v)$ を画像ピラミッドの最下位層（第 0 階層）とする．すなわち，$I^0(u,v) = I(u,v)$ および $J^0(u,v) = J(u,v)$ である．次に，$I^0(u,v)$ および $J^0(u,v)$ を $1/2$ 倍に縮小し，第 1 階層の画像 $I^1(u,v)$ および $J^1(u,v)$ を生成する．以下，同様に $l = 2, \cdots, l_{\max}$ のそれぞれについて，階層 l における画像 $I^l(u,v)$ および $J^l(u,v)$ を生成する．本論文では，$l_{\max}$ を階層数と呼ぶ.

階層 l の画像を生成する際には，階層 $(l-1)$ の画像において隣接した 4 ピクセルの平均値を階層 l の画像の 1 ピクセルとすることで画像の縮小を行う．言い換えれば，階層 l の画像に

おける画素値は以下のように表すことができる．

$$I^l(n_1, n_2) = \frac{1}{4}\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} I^{l-1}(2n_1 + i_1, 2n_2 + i_2) \tag{2.49}$$

$$J^l(n_1, n_2) = \frac{1}{4}\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} J^{l-1}(2n_1 + i_1, 2n_2 + i_2) \tag{2.50}$$

ここで，最下位層（原画像）における視差の最大値が d_m で与えられるとすると，最上位層 $l_{\max}$ における視差は $d_m/2^{l_{\max}}$ となる．つまり，画像から抽出する探索ウィンドウの長さに対して $d_m/2^{l_{\max}}$ が十分に小さくなるように階層数 $l_{\max}$ を決定する必要がある．

画像ピラミッドを生成したのち，$l = l_{\max}$ として階層的探索を開始する．ただし，最上位階層においては，基準点と対応点が同じ座標を持つと仮定し，$\boldsymbol{q}^{l_{\max}} = \boldsymbol{p}^{l_{\max}}$ とする．したがって，対応点 $\boldsymbol{q}^{l_{\max}}$ の座標は次式で表される．

$$\boldsymbol{q}^{l_{\max}} = (\lfloor 2^{-l_{\max}} p_u \rfloor, \lfloor 2^{-l_{\max}} p_v \rfloor) \tag{2.51}$$

ここで，$\lfloor z \rfloor$ は床関数であり，z を負の無限大方向の最も近い整数に丸めることを示す．

次に，$l \Leftarrow l_{\max} - 1$ として Step 2 に移る．

[ピクセルレベルの対応付け]

Step 2

第 l 階層上の基準点 $\boldsymbol{p}^l$ の座標を次式で与える．

$$\boldsymbol{p}^l = (\lfloor 2^{-l} p_u \rfloor, \lfloor 2^{-l} p_v \rfloor) \tag{2.52}$$

次に，第 l 階層での対応点 $\boldsymbol{q}^l$ の初期値を次式で与える．

$$\boldsymbol{q}^l = (2q_v^{l+1}, \lfloor 2^{-l} q_v \rfloor) \tag{2.53}$$

Step 3

画像 $I^l(u, v)$ において，中心が $\boldsymbol{p}^l$ となるように探索ウィンドウを抽出する．同様に，画像 $J^l(u, v)$ において，中心が $\boldsymbol{q}^l$ となるように探索ウィンドウを抽出する．その後，1 次元 POC に基づく平行移動量推定を用いて，ピクセル精度の平行移動量 δ を求める．第 l 階層における対応点 $\boldsymbol{q}^l$ の座標を次式で与える．

$$\boldsymbol{q}^l \Leftarrow \boldsymbol{q}^l - (\delta, 0) \tag{2.54}$$

Step 4

$l \Leftarrow l-1$ として，$l \geq 0$ である限り，Step 2 から Step 4 を繰り返す．

[サブピクセルレベルの対応付け]

Step 5

画像 $I(u,v)$ において，中心が $\boldsymbol{p}$ となるように探索ウィンドウを抽出する．同様に，画像 $J(u,v)$ において，中心が $\boldsymbol{q}^0$ となるように探索ウィンドウを抽出する．1 次元 POC に基づく平行移動量推定を用いて，サブピクセル精度の平行移動量 δ を求める．サブピクセル精度の対応点 $\boldsymbol{q}$ は次式で与えられる．

$$\boldsymbol{q} = \boldsymbol{q}^0 + (\delta, 0) \tag{2.55}$$

以上に示した処理 Step 1∼Step 5 によって，基準点 $\boldsymbol{p}$ の対応点 $\boldsymbol{q}$ を求めることができる．

2.5　画像対応付けアルゴリズムの性能評価

本節では，これまでに述べた領域ベースの対応付けアルゴリズムについて，ステレオビジョンに基づく 3 次元計測を通して性能評価を行う．まず，計測精度とロバスト性の観点からアルゴリズムを比較する．次に，計算量の観点からアルゴリズムを比較する．

2.2.2 節で述べた類似度や相違度に基づく対応付けアルゴリズムは，POC に基づく画像対応付けアルゴリズムと同様に，2.4.4 節で述べた粗密探索の枠組みに組み込むことが可能である．そこで，(a) SAD, (b) SSD, (c) NCC に基づく対応付けアルゴリズムをそれぞれ粗密探索の枠組みに組み込み，(d) POC に基づく対応付けアルゴリズムと比較する．すなわち，本節で評価する画像対応付けアルゴリズムは，以下 (a) ∼ (d) の 4 種類である．

(a)　SAD に基づくアルゴリズム

SAD を用いた平行移動量推定と階層探索を組み合わせた対応付けアルゴリズムである．サブピクセル推定には等角直線フィッティングを用いる [47].

(b)　SSD に基づくアルゴリズム

SSD を用いた平行移動量推定と階層探索を組み合わせた対応付けアルゴリズムである．

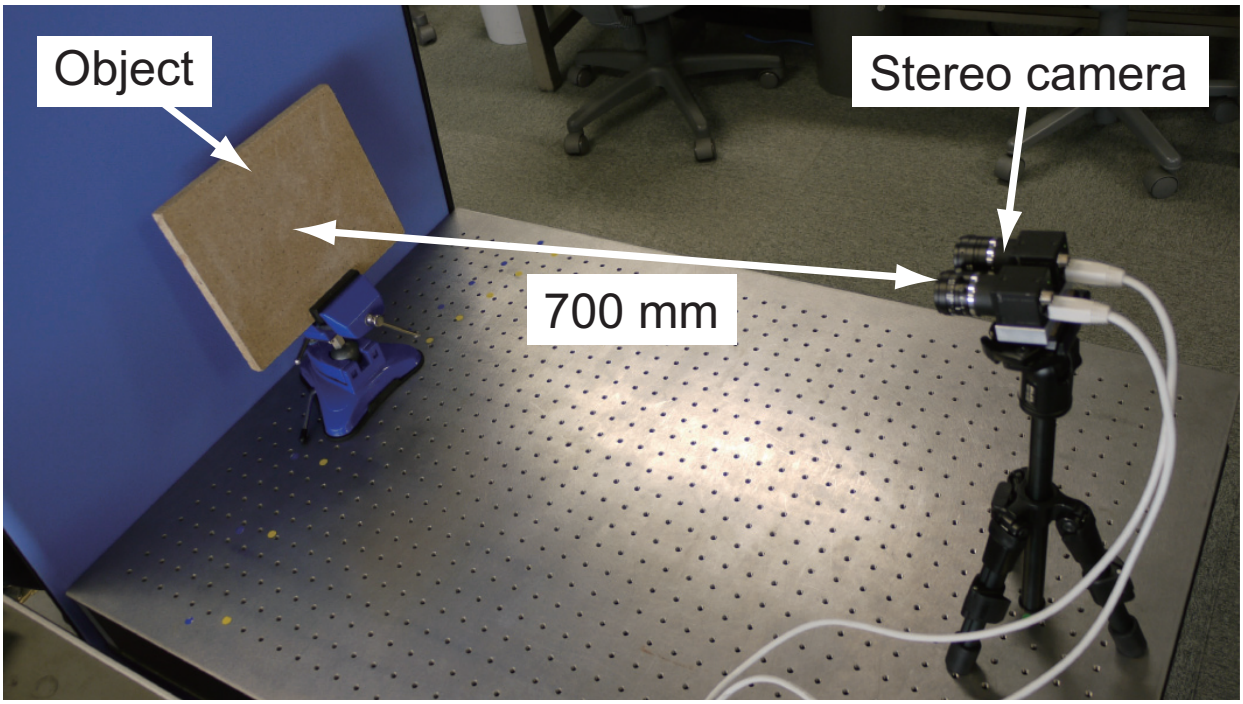


図 2.12　領域ベースの対応付けに関する性能評価実験のための環境

サブピクセル推定にはパラボラフィッティングを用いる [47].

(c)　**NCC に基づくアルゴリズム**

NCC を用いた平行移動量推定と階層探索を組み合わせた対応付けアルゴリズムである．サブピクセル推定にはパラボラフィッティングを用いる [47].

(d)　**POC に基づくアルゴリズム**

2.4.4 で述べたアルゴリズムである．POC を用いた平行移動量推定では，2.4.3 で述べた高精度化手法を用いる．

性能評価は，図 2.12 に示すステレオビジョンシステムを用いて行う．システムの諸元を表 2.1 に示す． 計測対象として，形状が既知である平板と球を用いる．まず，キャリブレーション済みのステレオカメラを用いて物体を撮影し，ステレオ画像を取得する．次に，ステレオ画像に手動で関心領域を設定し，関心領域において格子状に 5 ピクセル間隔で基準点を配置する．配置した基準点に対して，(a) ～ (d) のそれぞれの手法を用いてステレオ画像間を対応付け，3 次元計測を行う．最後に，3 次元計測によって得られた 3 次元点群について，既知

表 2.1　領域ベースの対応付けに関する性能評価実験で用いるステレオビジョンシステムの諸元

ステレオカメラ	Point Grey Research Inc., Flea2 (FL2G-13S2M-C), モノクローム ×2
基線長	40 mm
レンズ	μTRON FV1022, 焦点距離 10 mm
画像サイズ	$1,280 \times 960$ ピクセル

のモデル形状を当てはめる．モデル形状の当てはめでは，計測対象が平板の場合は平板の方程式，球の場合は球の方程式をそれぞれ当てはめる．当てはめ誤差がステレオ画像上で 1 ピクセル以上である点を誤対応点とし，計測を行った点のうち誤対応点が占める割合を誤対応率とする．また，誤対応点を除いた計測点について，誤差の RMS (Root Mean Square) 値を計算する．ここで，対応付けアルゴリズムのロバスト性を評価する尺度として誤対応率を，精度を評価する尺度として誤差の RMS 値を用いる．

それぞれのアルゴリズムにおける対応付けのパラメータを以下のように設定する．(a) SAD, (b) SSD, (c) NCC に基づくアルゴリズムでは，探索ウィンドウの大きさを 16×15 ピクセルとする．(d) POC に基づくアルゴリズムでは，探索ウィンドウの大きさを 32 ピクセル ×15 ラインとする．このとき，POC に基づくアルゴリズムでは，POC 関数の計算の際にハニング窓をかけるため，32×15 ピクセルのウィンドウサイズは，他の対応付けアルゴリズムで用いる 16×15 ピクセルに相当する．また，画像ピラミッドの階層数は 4 とする．

3 次元計測に用いた平板のステレオ画像および 3 次元計測結果を図 2.13 および図 2.14 に示す．球のステレオ画像および 3 次元計測結果を図 2.15 および図 2.16 に示す．3 次元計測における誤対応率を表 2.2 に示す．平板・球のいずれの場合においても，(a) SAD および (b) SSD, (c) NCC に基づくアルゴリズムに比べて，(d) POC に基づくアルゴリズムでは誤対応率が低

表 2.2　3 次元計測における誤対応率 [%]

対応付けアルゴリズム	SAD	SSD	NCC	POC
平板	11.65	2.39	0.00	0.00
球	0.72	0.30	0.42	0.02

表 2.3　3 次元計測における誤差の RMS 値 [mm]

対応付けアルゴリズム	SAD	SSD	NCC	POC
平板	4.273	2.707	0.817	0.784
球	3.155	2.461	2.089	0.591

いことが確認できる．これは，(a) SAD や (b) SSD に基づくアルゴリズムでは，左右のカメラ画像の明るさの違いや，物体表面の反射特性に影響を受けたためであると考えられる．また，誤差の RMS 値を表 2.3 に示す．平板・球のいずれの場合においても，(a) SAD および (b) SSD, (c) NCC に基づくアルゴリズムに比べて，(d) POC に基づくアルゴリズムでは誤差の RMS 値が低いことが確認できる．これは，サブピクセルレベルの平行移動量推定に関して，(a) SAD および (b) SSD, (c) NCC に基づくアルゴリズムではヒューリスティックな相関ピークモデルを用いているのに対し，(d) POC に基づくアルゴリズムでは解析的に導出された相関ピークモデルを用いているためであると考えられる．以上より，(a) SAD および (b) SSD, (c) NCC に基づくアルゴリズムに比べて，(d) POC に基づくアルゴリズムを用いることで，ステレオ画像間の対応付けをロバストかつ高精度に行うことができるといえる．

次に，それぞれの画像対応付けアルゴリズムについて，計算量を比較する．計算量を比較は，粗密探索の 1 階層において，1 点の基準点に対する演算回数を数えることで行う．その結果を表 2.4 に示す．表 2.4 では，加算 (ADD), 減算 (SUB), 乗算 (MUL), 除算 (DIV), 平方根 (SQRT), 絶対値 (ABS) のそれぞれについて演算回数，およびそれらの合計を示している．これら演算回数の計算においては，本実験で用いたパラメータで対応付けを行う場合を想定している．ただし，サブピクセル推定に要する計算量は，全体の計算量に対してごく僅かであるため，表 2.4 の演算回数には含まれていない．表 2.4 より，SAD や SSD に基づくアルゴリズ

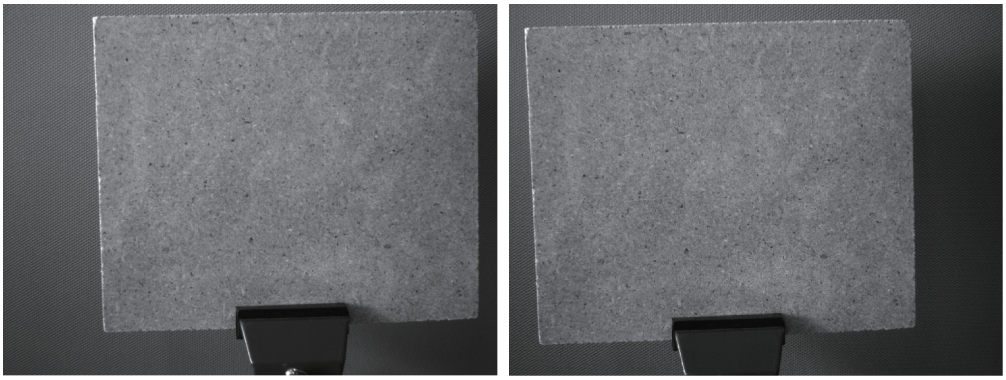
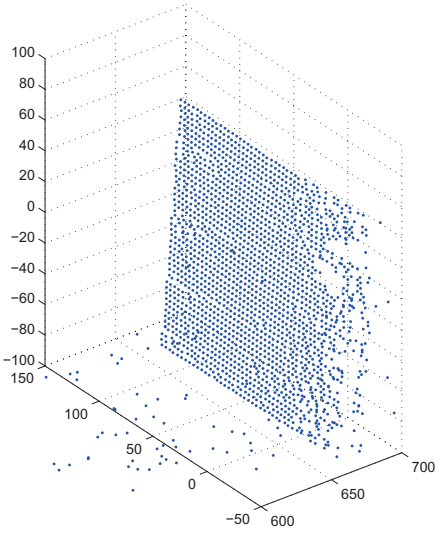
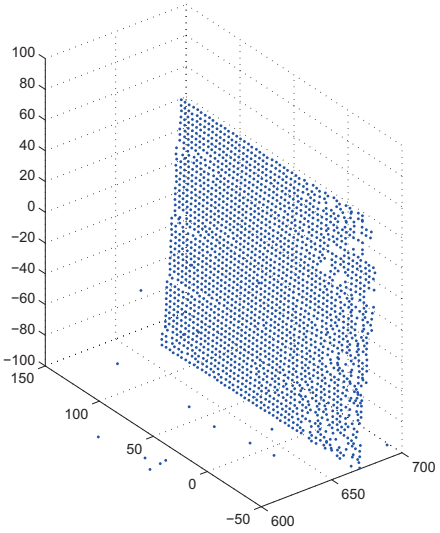


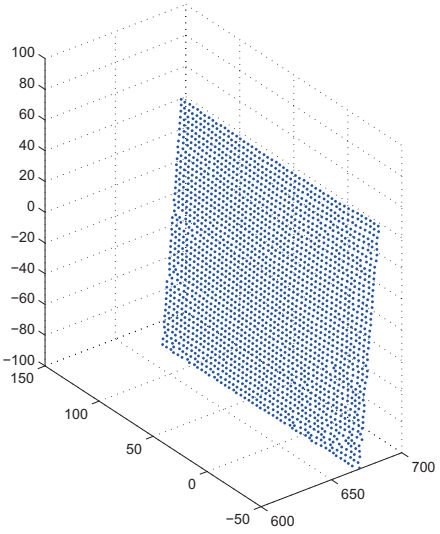
図 2.13　平板の 3 次元計測に用いるステレオ画像



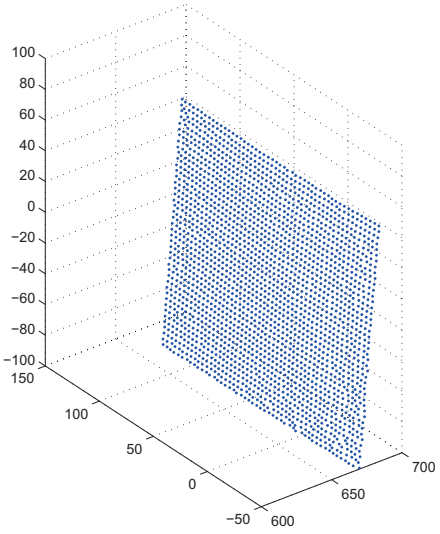
(a) SAD



(b) SSD



(c) NCC



(d) POC

図 2.14　平板の 3 次元計測結果

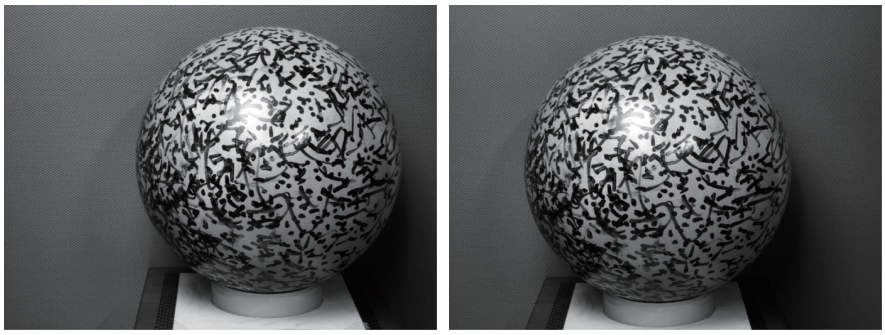
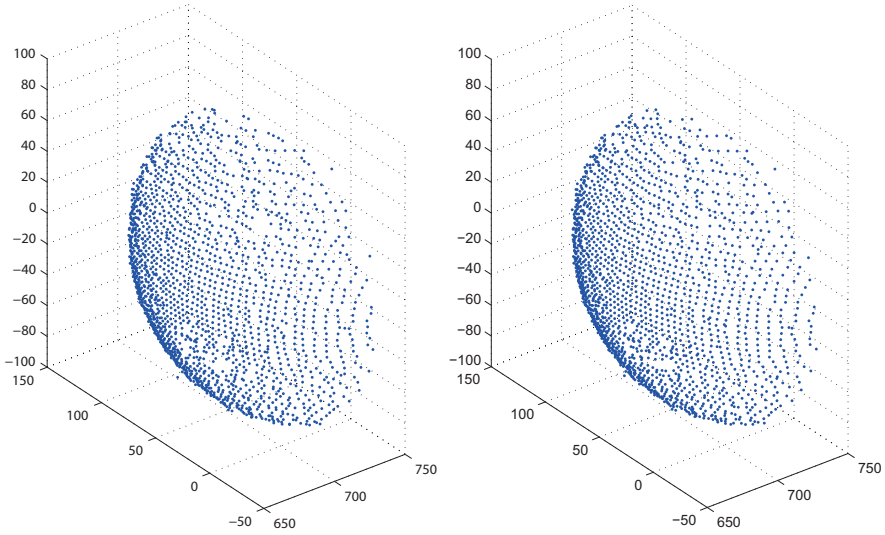


図 2.15　球の 3 次元計測に用いるステレオ画像



(a) SAD　(b) SSD

(c) NCC　(d) POC

図 2.16　球の 3 次元計測結果

表 2.4　領域ベースの画像対応付けアルゴリズムにおける演算回数の比較（1 階層，1 点に対する演算回数）

演算	SAD	SSD	NCC	POC
ADD	3,840	3,840	19,200	11,360
SUB	3,840	3,840	7,680	9,920
MUL	0	3,840	11,536	22,720
DIV	0	0	48	495
SQRT	0	0	16	480
ABS	3,840	0	0	0
合計	11,520	11,520	38,480	44,975

ムに比べ，NCC や POC に基づくアルゴリズムの演算回数が多く，約 4 倍であることがわかる．さらに，NCC に基づくアルゴリズムに比べ，POC に基づくアルゴリズムの演算回数が多いこともわかる．それぞれの対応付けアルゴリズムでは，ステレオ画像から抽出される探索ウィンドウ（局所的な画像ブロック）が含む画素数は等しい．したがって，POC に基づくアルゴリズムは，他の領域ベースのアルゴリズムに比べて演算強度（アルゴリズムにおける 1 バイトあたりの演算回数）が高いといえる．

2.6　むすび

本章では，画像の対応付けについて基礎的考察を行った．まず，画像対応付けアルゴリズムについて，特徴ベースと領域ベースのアルゴリズムに分けられることを述べた．次に，本論文で着目する応用であるステレオビジョンに基づく 3 次元計測において密な対応付けが必須であることから，領域ベースの対応付けアルゴリズムが適していることを述べた．さらに，本論文で用いる POC に基づく画像対応付けアルゴリズムについて述べた．最後に，領域ベースの画像対応付けアルゴリズムについて，3 次元計測を通して性能評価を行った．性能評価の結果，POC に基づく画像対応付けアルゴリズムを用いることでステレオ画像間をロバストかつ高精

度に対応付けることができることを示した.

第 3 章

画像対応付けアルゴリズムの GPU 実装

3.1 まえがき

GPU (Graphics Processing Unit) は，計算機のグラフィックス処理を専門とする補助演算装置である．GPU は，今日では，デスクトップ・コンピュータ，ラップトップ・コンピュータ，ワークステーションをはじめとして，ほとんどの計算機に搭載されている．計算機における GPU の一般的な役割は，2 次元や 3 次元のグラフィックスを生成することである．これらのグラフィックスは，オペレーティング・システムにおけるグラフィカル・ユーザー・インターフェース，ビデオ・ゲーム，ビジュアル・イメージング・アプリケーション，ビデオなど，広い分野で利用されている．

1990 年代前半まで，GPU は計算機におけるグラフィックスをレンダリングするための単なる演算装置であった．2 次元や 3 次元のグラフィックス生成は，処理が画素ごとに独立しているため，並列演算処理が可能である．そのため，GPU は，並列演算処理に優れる SIMD (Single Instruction Multiple Data) 型の演算装置として設計され, 生成する画像をグラフィックスパイプラインに基づいてレンダリングしていた．その後，2000 年代にかけて，高速・高品質なグラフィックス処理に対する需要から，より解像度の高い画面を出力したり，より詳細な 3 次元映像のレンダリングを行ったりするために，GPU では，演算器が増量され，フ

レームバッファ（ディスプレイに表示する画像データを保存する記憶領域）にアクセスする帯域幅が増加された．この結果，GPU は，数百のコアと広い帯域幅を持つ並列プロセッサとなった．さらに，GPU 内の演算器に関する命令セットが公開され，ユーザが GPU で行う演算をプログラムできるようになると，GPU の持つ特長を汎用計算に利用する GPGPU (General-Purpose computation on Graphics Processing Units) と呼ばれる試みが提案された [6], [7]．このような背景に後押しされ，GPU に関するプログラミング環境はさらに整備され，C や C++ といった一般的なプログラミング言語の拡張を用いて GPU のプログラムが作成できるようになった．このような技術革新によって，GPU は，汎用的にプログラミング可能で多数のコアを備えた並列プロセッサとして利用できるようになった．

本論文では，POC に基づく画像対応付けアルゴリズムにおいて，計測点ごと・画素ごとに処理が独立していることに着目し，GPU を用いた並列処理を提案する [55]–[61]．実装に際しては，NVIDIA 社と AMD 社とが提供するそれぞれの GPU に関して統一的な性能評価を行うために，OpenCL を用いる [14]．本章では，まず，一般的に使用されている GPU のアーキテクチャを概観する．次に，アルゴリズムを GPU に実装するためのプログラミングモデルについて概説する．これらを踏まえて，位相限定相関法に基づく画像対応付けアルゴリズムを GPU に実装する手法について述べる．最後に，性能評価を通して，(i) GPU のアーキテクチャに合わせた最適化によって高速化が達成できること，(ii) CPU による実装に比べて高速化が達成できることを示す．

3.2　GPU のアーキテクチャとプログラミングモデル

3.2.1　NVIDIA GPU のアーキテクチャ

図 3.1 に NVIDIA GPU のアーキテクチャを示す [62], [63]．NVIDIA GPU の基本的な構成は，複数のストリーミング・マルチプロセッサ (Streaming Multiprocessor: SM) と，オフ・チップのビデオメモリである．各ストリーミング・マルチプロセッサは，8 基のストリーミング・プロセッサ (Streaming Processor: SP), 2 基の SFU (Special Function Unit), 命令ユニット，命令キャッシュ，読み出し専用の定数キャッシュ，読み/書き用共有メモリで構成され

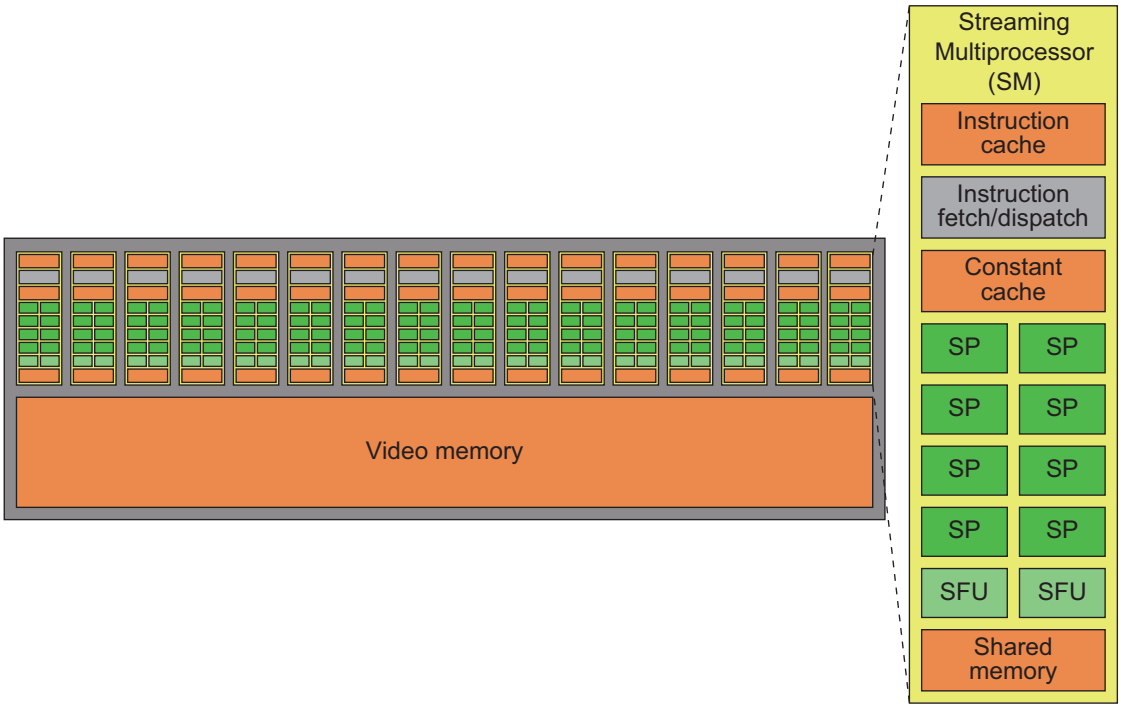


図 3.1　NVIDIA GPU のアーキテクチャ（G80 アーキテクチャ）

る．SP は，CUDA (Compute Unified Device Architecture) コアとも呼ばれ，浮動小数点および整数処理を行うスカラー型演算器である．SFU は，超越関数の演算に用いられる．命令ユニットは，命令のフェッチ・デコード・スケジューリングを行う．共有メモリは，SP 間で共有される記憶領域であり，データの受け渡しに利用される．ビデオメモリは，SM 間でのデータの受け渡しに利用される．SM が持つメモリ（共有メモリ）に比べ低速だが容量は大きい．

図 3.1 に示したのは，2006 年 11 月に発表された GeForce 8800 GTX に採用されている G80 アーキテクチャである．この後，2008 年 6 月には，G80 アーキテクチャの基本的な構成はそのままに SM を 16 個から 30 個に増やし，専用ユニットの追加によって倍精度浮動小数点演算に対応した GT200 アーキテクチャが発表された [64]．2010 年 3 月に発表された Fermi アーキテクチャでは，SM あたりの SP が 8 個から 32 個に増加し，共有メモリが増量しビデオメモリのキャッシュを兼ねるようになった [65]–[67]．また，複数のスレッドを同時に発行できるような構成になり，命令発行効率を格段に引き上げたのが特徴となっている [68]．GeForce GTX 480（GF100 アーキテクチャ）や GeForce GTX 580（GF110 アーキテクチャ）は，この Fermi アーキテクチャを採用している．Fermi に続き 2012 年に発表された Kepler アーキ

テクチャからは，純粋に性能を追求する方向ではなく，性能/消費電力比や，実行効率性能を重視するようになってきた [69]–[71]．表 3.1 に示すように，NVIDIA GPU では，度重なるアーキテクチャの改良が行われているが，複数の演算器 (SP) を持つコア (SM) の集合という構成には変化がない．

表 3.1　NVIDIA GPU のスペック一覧

Model	Launch	Process	Core		Memory			
		[nm]	amount	clock [MHz]	amount [MB]	clock [MHz]	bus width [bit]	bandwidth [GB/s]
8800 GTX [72]	2006/11/8	90	128 (8×16 SM)	575	768	1,800	384	86.4
GTX 280 [64]	2009/1/15	65	240 (8×30 SM)	602	1,024	2,214	512	141.7
GTX 480 [66]	2010/3/26	40	480 (32×15 SM)	607	1,536	3,696	384	177.4
GTX 580 [67]	2010/11/9	40	512 (32×16 SM)	772	1,536	4,008	384	192.4
GTX 680 [70]	2012/3/22	28	1,536 (192×8 SMX)	1,006	2,048	1,502	384	192.3
GTX 780 [71]	2013/5/23	28	2,304 (192×12 SMX)	863	3,072	1,502	384	288.4
GTX 780 Ti [73]	2013/11/7	28	2,880 (192×15 SMX)	875	3,072	1,750	384	336.4

3.2.2　AMD GPU のアーキテクチャ

図 3.2 に AMD GPU のアーキテクチャを示す [74]．AMD GPU は，複数の SIMD コア (SIMD Core) と，オフ・チップのビデオメモリから構成される．各 SIMD コアは，16 個の VLIW (Very Long Instruction Word) プロセッサ, ウルトラスレッド・ディスパッチ・プロセッサ (Ultra-Threaded Dispatch Processor), ローカルデータシェア・メモリ (Local Data Share Memory) を持つ．VLIW プロセッサは，ベクトル型の演算器であり，4 基の浮動小数点演算器と 1 基の浮動小数点演算および超越関数の演算を行う演算器によって構成されている．ウルトラスレッド・ディスパッチ・プロセッサは，命令のフェッチおよびスケジューリング，発行を行う．ローカルデータシェア・メモリは，SIMD コア内で共有される記憶領域であり，VLIW プロセッサ間のデータの受け渡しに利用される．ビデオメモリは，SIMD コア間でのデータの受け渡しに利用される．ローカルデータシェア・メモリに比べ低速だが容量は大きい．

図 3.2 に示したのは，2008 年 6 月に発表された Radeon HD 4780 に採用されている RV770 アーキテクチャである．この後，2009 年 9 月には，RV770 アーキテクチャの基本的な構成はそのままに，SIMD コアを 10 個から 20 個に増やした RV870 (Cypress) アーキテクチャが発表された [75], [76]．さらに，2010 年 12 月には，VLIW プロセッサあたりの演算器を 5 基（4 基の浮動小数点演算器と 1 基の超越関数演算ユニット）から 4 基（超越関数演算にも対応した 4 基の演算器）に減らした Barts アーキテクチャが発表された [77]．2011 年には，汎用コンピューティングにおける利便性を向上するため，VLIW プロセッサではなく，NVIDIA GPU と同様のスカラプロセッサを搭載した GCN (Graphics Core Next) アーキテクチャ [78] が発表された [79]．表 3.2 に示すように，AMD GPU は，NVIDIA GPU と同様にアーキテクチャに関する変更が何度か行われているが，複数の演算器（VLIW プロセッサ）を持つコア（SIMD コア）の集合という構成は変わっていない．

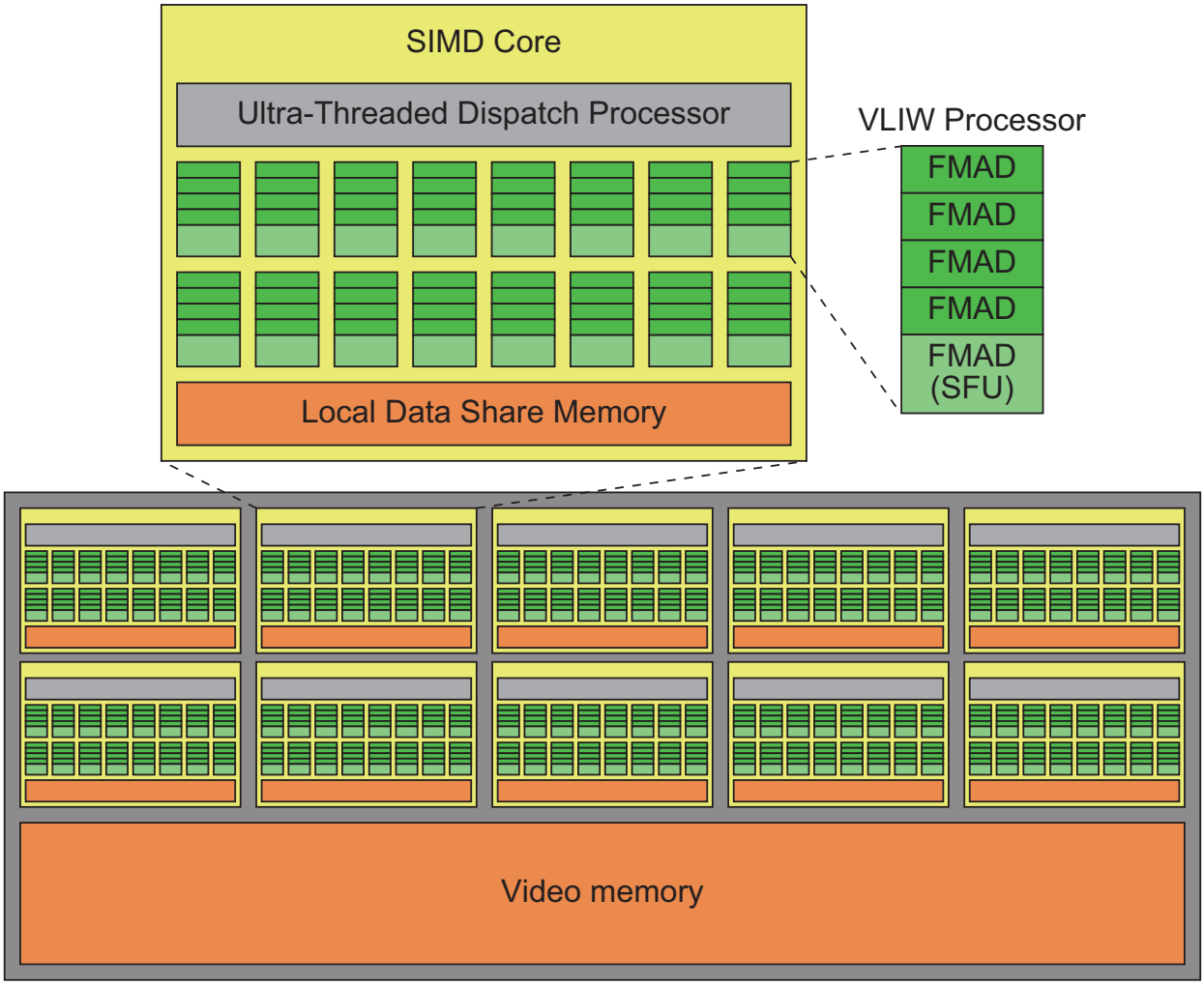


図 3.2　AMD GPU のアーキテクチャ（RV770 アーキテクチャ）

表 3.2　AMD GPU のスペック一覧

Model	Launch	Process	Core		Memory			
		[nm]	amount	clock [MHz]	amount [MB]	clock [MHz]	bus width [bit]	bandwidth [GB/s]
HD 4890 [74]	2009/4/2	55	800	850	1024	975	256	124.8
HD 5870 [75]	2009/9/23	40	1600	850	1024	1200	256	153.6
HD 6970 [77]	2010/12/15	40	1536	880	2048	1375	256	176
HD 7970 [80]	2012/1/9	28	2048	925	3072	1375	384	264
R9 290X [81]	2013/10/24	28	2816	947	4096	1250	512	320

3.2.3　OpenCL のプログラミングモデル

OpenCL は，マルチコア CPU や GPU，Cell プロセッサ，DSP などによる異種混在の計算資源を利用した並列コンピューティングのためのフレームワークである [14]．OpenCL では，これら計算ハードウェアを包括して取り扱うために，図 3.3 のようにプラットホームモデルを定義し，計算ハードウェアを抽象化している．OpenCL におけるプラットホームモデルは，1 台のホスト (CPU) とそれに接続する演算デバイス (Compute Device) で構成される．演算デバイスは 1 つ以上の演算ユニット (Compute Unit) を持つ．さらに，演算ユニットは複数の処理エレメント (Processing Element) から構成されている．3.2.1 節で述べた NVIDIA GPU アーキテクチャでは，演算ユニットがストリーミング・マルチプロセッサ (SM) に，処理エレメントがストリーミング・プロセッサ (SP) にそれぞれ相当する．また，3.2.2 節で述べた AMD GPU アーキテクチャでは，演算ユニットが SIMD コアに，処理エレメントが VLIW プロセッサにそれぞれ相当する．

演算デバイス上の並列処理は，多数のスレッドがカーネルと呼ばれる関数を同時に実行することで実現される．OpenCL の並列実行モデルでは，スレッドはワークアイテム (work-item) と呼ばれ，NDRange というインデックス空間において管理される．その様子を図 3.4 に示す．各ワークアイテムは組み込み変数を用いることで，NDRange 内で割り振られたインデックスの値を知ることができる．これは，各ワークアイテムに処理するデータを分配するのに役立つ．また，ワークアイテムは，ユーザが任意に指定するワークグループ (work-group) 単位でまとめることができる．同一ワークグループに属するワークアイテムは，処理の同期や，高速な記憶領域を用いたデータの共有を行うことが可能である．

ワークアイテムは，以下のように演算デバイスで処理される．(i) ユーザはワークグループの数とワークグループ内のワークアイテムの数を指定し，カーネルを呼び出す．(ii) ワークアイテムはワークグループ単位で演算ユニットに分配される．(iii) 演算ユニットは，分配されたワークアイテムの処理をスケジューリングし，それぞれの処理エレメントに割り当てて実行する．

OpenCL におけるメモリモデルを図 3.5 に示す．カーネルを実行するワークアイテムでは，

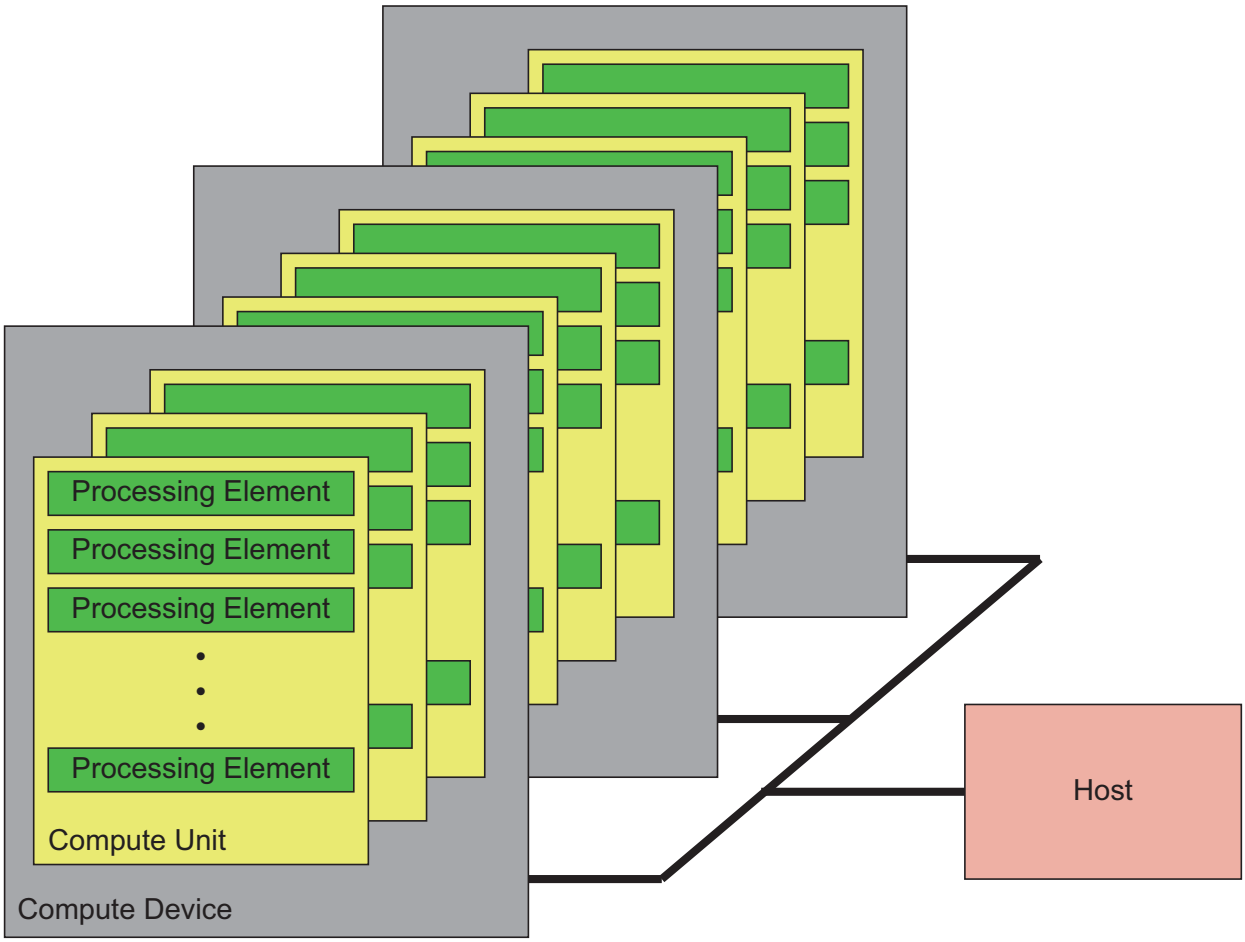


図 3.3　OpenCL のプラットホームモデル

以下の 4 種類のメモリ領域にアクセスできる．

- **グローバルメモリ**

  すべてのワークグループ・すべてのワークアイテムから読み書き両方のアクセスが許可されている領域である．GPU では，ビデオメモリがグローバルメモリに相当する．ビデオメモリは GPU のチップ外に存在するため，グローバルメモリへのアクセスは一般に低速である [82], [83]．GPU によっては，グローバルメモリに対する読み書きがキャッシュされる．NVIDIA GPU では Fermi アーキテクチャ以降，AMD GPU ではすべてのアーキテクチャでメモリ読み出しのキャッシュが行われる．

- **コンスタントメモリ**

  カーネル実行中に値が不変なグローバルメモリの領域である．GPU では，ビデオメモ

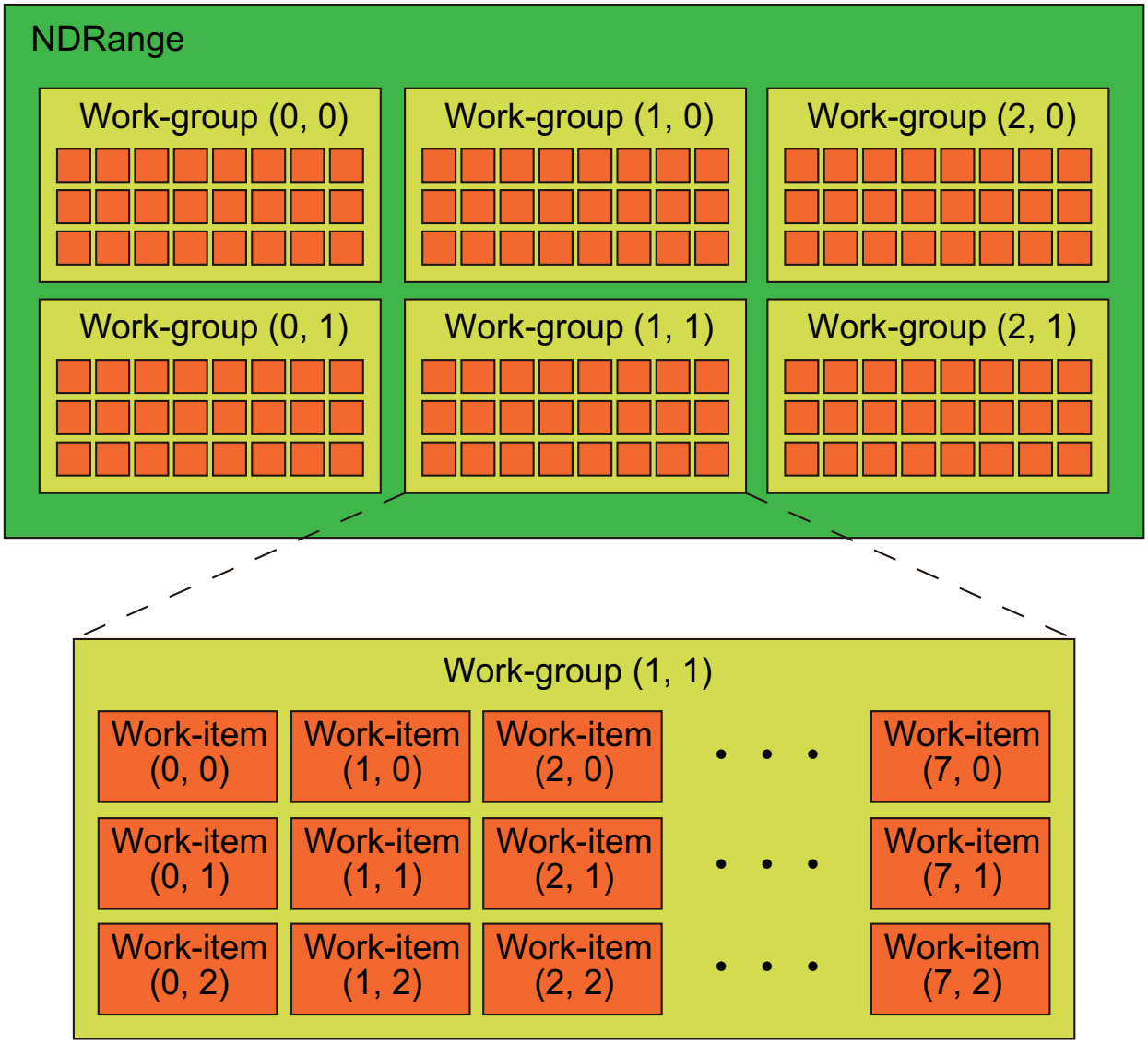


図 3.4　OpenCL における並列実行モデル

リ上の領域をコンスタントメモリとして割り当て，使用することができる．コンスタントメモリからの読み出しは，ワークグループごとにキャッシュされ，ワークグループ内のワークアイテムにブロードキャストされる [82], [83].

- **ローカルメモリ**

  ワークグループ内で共有されている領域である．ワークアイテムは，ローカルメモリを利用することで，ワークグループ内の他のワークアイテムと高速なデータの受け渡しが可能である [82], [83]．ローカルメモリは，NVIDIA GPU では共有メモリ，AMD

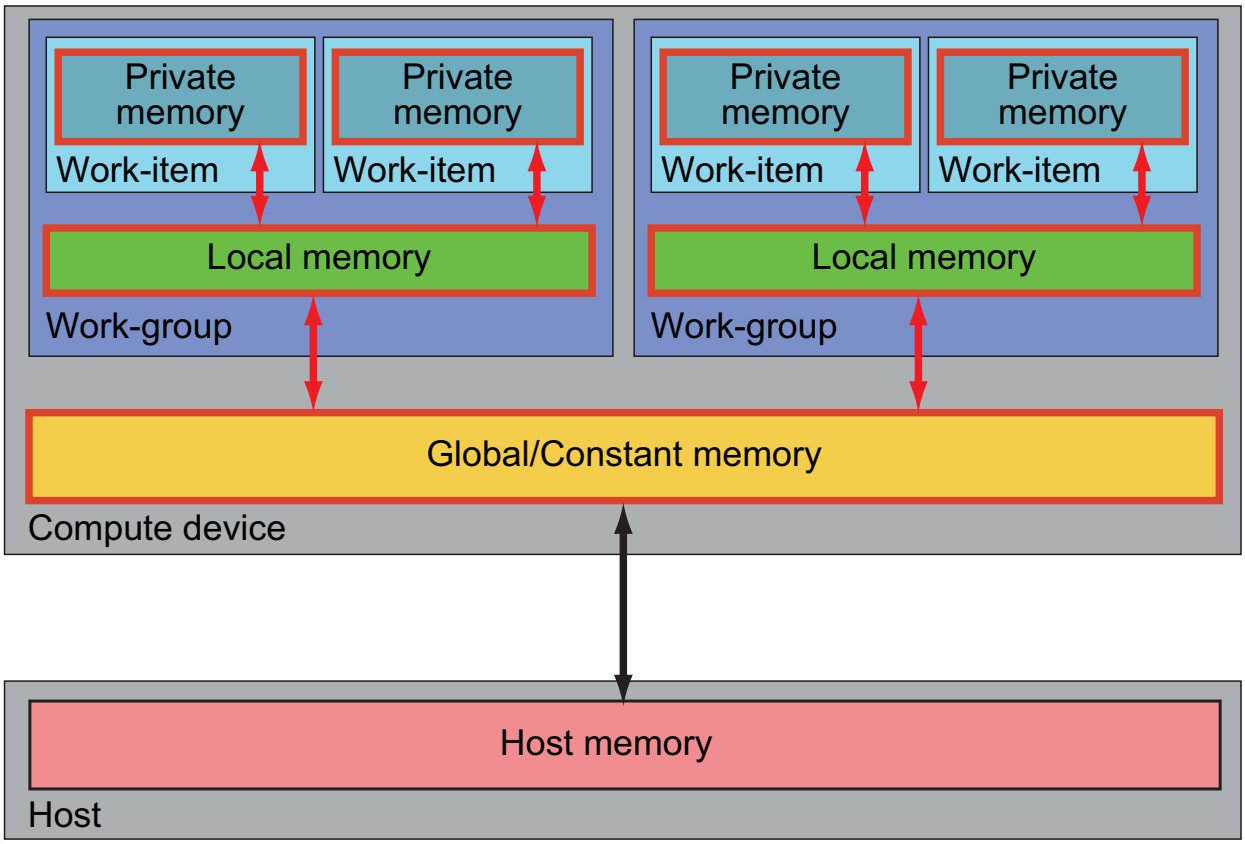


図 3.5　OpenCL におけるメモリモデル

GPU ではローカルデータシェア・メモリに相当する．

- **プライベートメモリ**

  ワークアイテムごとで独立に割り当てられた領域である．あるワークアイテムに割り当てられたプライベートメモリは，他のワークアイテムから参照できない．処理エレメントで行われた演算の結果は，プライベートメモリに記憶される．GPU において，プライベートメモリは，ローカルメモリと同様にオン・チップであるため，高速なアクセスが可能である [82], [83].

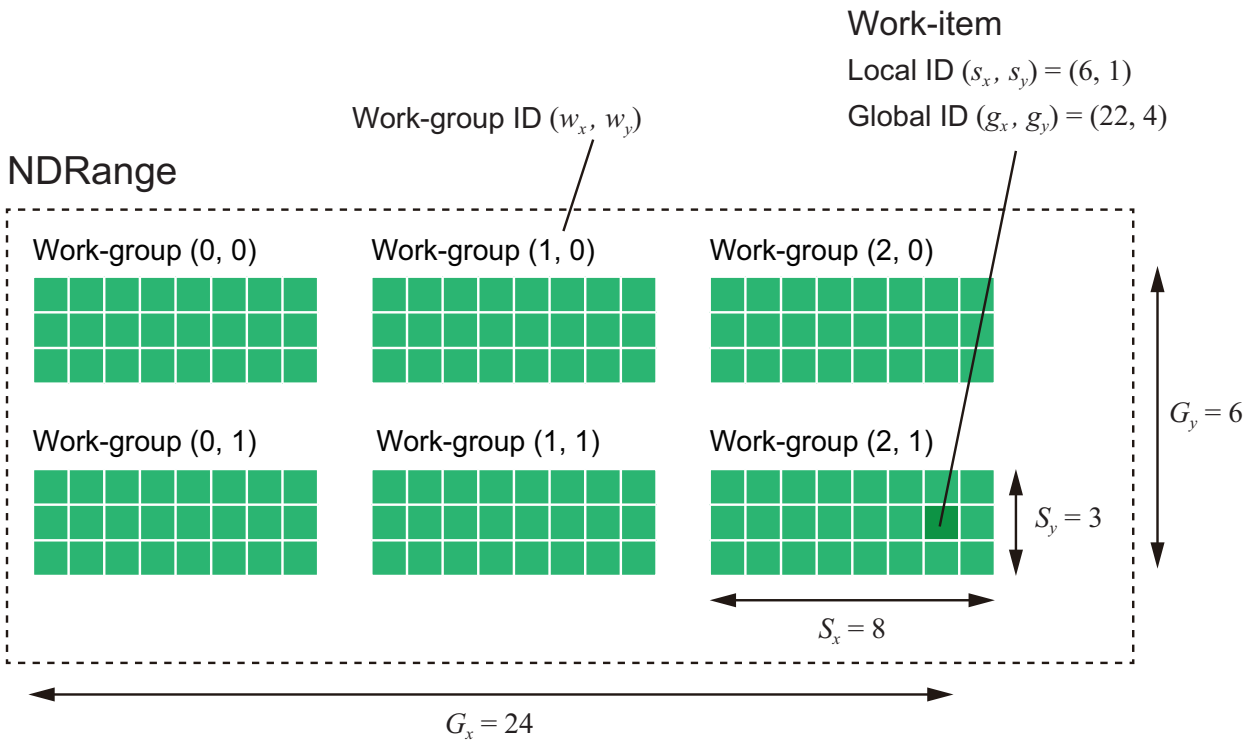


図 3.6　NDRange 空間におけるインデックスの定義．ワークグループとワークアイテムの数は説明のために設定されたものであり，実際の実装において用いるものとは異なることに注意されたい．

3.3　1 次元位相限定相関法に基づく画像対応付けアルゴリズムの GPU 実装

本節では，2.4.4 節で述べた対応付けアルゴリズムを，OpenCL を用いて GPU に実装する手法について述べる．

ここで，処理の分割粒度を記述するために，ワークグループやワークアイテムに関するインデックスの割り振りを図 3.6 に示すように定義する．

NDRange は，複数のワークグループから構成される．ワークグループは 2 次元的に配置されており，NDRange の中での位置を識別するために，それぞれのワークグループにはイン

デックス (w_x, w_y) を割り振る．

ワークグループは，複数のワークアイテムから構成される．ワークアイテムは 2 次元的に配置されており，水平方向の個数を S_x, 垂直方向の個数を S_y と表す．それぞれのワークアイテムには，ワークグループの中での位置を識別するために，ワークグループ内でのローカルなインデックス (s_x, s_y) を割り振る．

また，ワークグループという区分を考慮せずにワークアイテムの総数や位置を表現をするために，NDRange におけるワークアイテムについて，水平方向の総数を G_x, 垂直方向の総数を G_y とし，グローバルなインデックス (g_x, g_y) を割り振る．

3.3.1　実装における並列化手法

GPU 実装において広く知られているデザインパターンには次のものがある [62].

D1　GPU の持つ階層構造に合わせてアルゴリズムを並列化する．GPU は，演算器のクラスタ（演算ユニット）と演算器（処理エレメント）という階層構造を持つ．また，記憶領域についても，GPU 全体で共有されるグローバルメモリ，演算ユニット内で共有されるローカルメモリ，処理エレメントのそれぞれで使用するレジスタと，階層構造になっている．このような階層構造に合わせて，アルゴリズムを並列化し，実装することが重要である．

D2　演算回数やメモリアクセス量を削減する．これはあらゆるプロセッサにおける実装について一般的に言えることであるが，特に GPU の場合は，グローバルメモリへのアクセス量を削減することが重要である．グローバルメモリは容量が大きく，全てスレッドからアクセスできるが，アクセスには 400~600 サイクルのレイテンシが伴う．このレイテンシは，レジスタやローカルメモリへのアクセスに比べて数百倍低速である．このため，グローバルメモリへのアクセスは最小に抑える．

D3　グローバルメモリへのアクセスは，レイテンシの低いアクセスパターンを用いる．グローバルメモリへの複数スレッドからの同時アクセスは，アクセスのパターンによりレイテンシが異なる．Coalesced と呼ばれるメモリアクセスパターン [62] であれば，最

大のスループットとなる．このため，グローバルメモリへのアクセスは，可能な限り Coalesced アクセスとする．

D4　ハードウェアリソースの稼働率を高める．GPU では，ローカルメモリやレジスタの動的な確保ができないため，すべてのワークグループを同時にスケジューリングすることが困難である．そこで，カーネルの呼び出し時に，演算ユニットにおいて同時にスケジューリングおよび実行するワークグループを決め，それらを各演算ユニットへ割り付ける．割り付けたワークグループが終了した後，同様に残りのワークグループを割り付け，実行する．ここで，各演算ユニットで同時にスケジューリングされるワークグループは，(i) ワークアイテムあたりのレジスタの使用数 (ii) ワークグループあたりのローカルメモリの使用容量 (iii) 演算ユニットごとのスケジューリング可能なスレッドの最大数のそれぞれによって決まる数のうち，最小のものとなる．同時にスケジューリングされるワークグループが少ない場合，プロセッサ全体のスループットが向上せず，高速に処理できないことがある．これを回避するため，ワークグループあたりのワークアイテム数，およびレジスタやローカルメモリの使用量を調整する必要がある．

本節では，**D1** をもとに，1 次元 POC に基づく画像対応付けアルゴリズムを GPU に実装する手法を示す．後の節では，**D2**∼**D4** をもとに，GPU のアーキテクチャに適合した実装の工夫について述べる．

1 次元 POC に基づく画像対応付けアルゴリズムでは，ステレオ画像の片方の画像上で任意の基準点を配置し，それらの基準点それぞれに対して，探索ウィンドウを用いた平行移動量推定を行う．ここで，探索ウィンドウを用いた平行移動量推定は，基準点ごと（探索ウィンドウごと）に処理が独立である．また，探索ウィンドウ内の処理に着目すると，一部で同期が必要であるものの，ほとんどの処理が画素ごとに独立している．これらを踏まえて，基準点ごと（探索ウィンドウごと）の処理を演算ユニットへ割り当て，画素ごとの処理を処理エレメントに割り当てる．以下に，1 次元 POC に基づく画像対応付けアルゴリズムの GPU 実装について詳細を記す．ここで，対応付けを行う基準点の数を P とする．また，探索ウィンドウは，L ラインからなる長さ N ピクセルの 1 次元信号とする．

入力

- ステレオ画像ペア $I(u,v)$ および $J(u,v)$
  それぞれの画像の大きさは $w \times h$ ピクセルである．画素値はラスタスキャン順に $w \times h$ 個の 32 bit 浮動小数点数型 (float) の 1 次元配列として格納されている．
- 基準点の座標 $\boldsymbol{p}_i = (p_{iu}, p_{iv}) \quad (i = 0, 1, \cdots, P-1)$
  基準点 P 個分の水平・垂直座標が，それぞれ 32 bit 浮動小数点数型 (float) 配列として格納されている．

これらの入力データは，処理の開始に際して，ホストメモリ（計算機上のメモリ）からグローバルメモリ（グラフィックカード上のビデオメモリ）に転送されているものとする．

出力

- 対応点の水平座標 $q_{iu} \quad (i = 0, 1, \cdots, P-1)$
  対応点 P 個分の水平座標が 32 bit 浮動小数点数型 (float) 配列として得られる．2.4.4 で述べたように，本論文では，入力のステレオ画像ペアは平行化されていることを前提としている．したがって，対応点の垂直座標は基準点の垂直座標と等しく，対応点の探索は対応点の水平座標のみを求める問題になるため，対応点の水平座標のみを出力とする．
- 位相限定相関関数の相関ピーク値 $\alpha_i \quad (i = 0, 1, \cdots, P-1)$
  探索ウィンドウを用いたブロックマッチングのそれぞれにおいて計算された位相限定相関関数の相関ピーク値が 32 bit 浮動小数点数型 (float) 配列として得られる．

出力データは，GPU での演算処理後，グローバルメモリからホストメモリに転送される．

1 次元 POC に基づく画像対応付けアルゴリズムの GPU 実装は，**(I)** 画像ピラミッドの生成と，各階層ごとのマッチングに分かれる．各階層ごとのマッチングは，

(II)　探索ウィンドウの抽出

(III)　離散フーリエ変換

(IV)　正規化相互パワースペクトルの計算

(V)　離散フーリエ逆変換

(VI)　対応点座標の更新

の 5 つに分かれる．その様子を図 3.7 に示す．

2.4.4 節で述べたように，一回の対応付け処理では，画像ピラミッドの生成は一度行うだけで良いが，各階層ごとのマッチングは階層数 $l_{\max}$ と等しい回数だけ行うことに注意されたい．以下にそれぞれの処理の詳細な実装を述べる．

(I)　画像ピラミッドの生成

画像ピラミッド生成カーネルを用いて，画像ピラミッドを生成する．このカーネルは，大きさ $\frac{w}{2^{l-1}} \times \frac{h}{2^{l-1}}$ ピクセルの画像 $I^{l-1}(u,v)$ を入力とし，入力画像を 1/2 倍に縮小することで，大きさ $\frac{w}{2^l} \times \frac{h}{2^l}$ ピクセルの画像 $I^l(u,v)$ を出力する．画像 $I^l(u,v)$ の 1 画素をひとつのワークアイテムが計算するように，カーネルを実行するワークアイテムを次のように設定する．

- NDRange: $\left(\lfloor \frac{w}{2^l} \rfloor, \lfloor \frac{h}{2^l} \rfloor\right)$
- ワークグループ：$(8,8)$

カーネルが実行される様子を図 3.8 に示す．以下にワークアイテム (g_x, g_y) が行う処理を詳細に述べる．ただし，出力画像の幅または高さを超過するインデックスを持つワークアイテムは処理を行わない．言い換えれば，$\frac{w}{2^l} \times \frac{h}{2^l}$ ピクセルの画像 $I^l(u,v)$ を出力するカーネルにおいて，$(g_x+1) > \frac{w}{2^l}$ または $(g_y+1) > \frac{h}{2^l}$ を満たす[*1]ワークアイテム (g_x, g_y) は，処理を行わない．

Step1

グローバルメモリに格納されている画像 $I^{l-1}(u,v)$ より，4 つの画素の輝度値

[*1] g_x および g_y にそれぞれ 1 が加算されているのは，ワークアイテムのインデックスが 0 オリジン（添字が 0 から始まるインデックス方式）であるためであることに注意されたい．

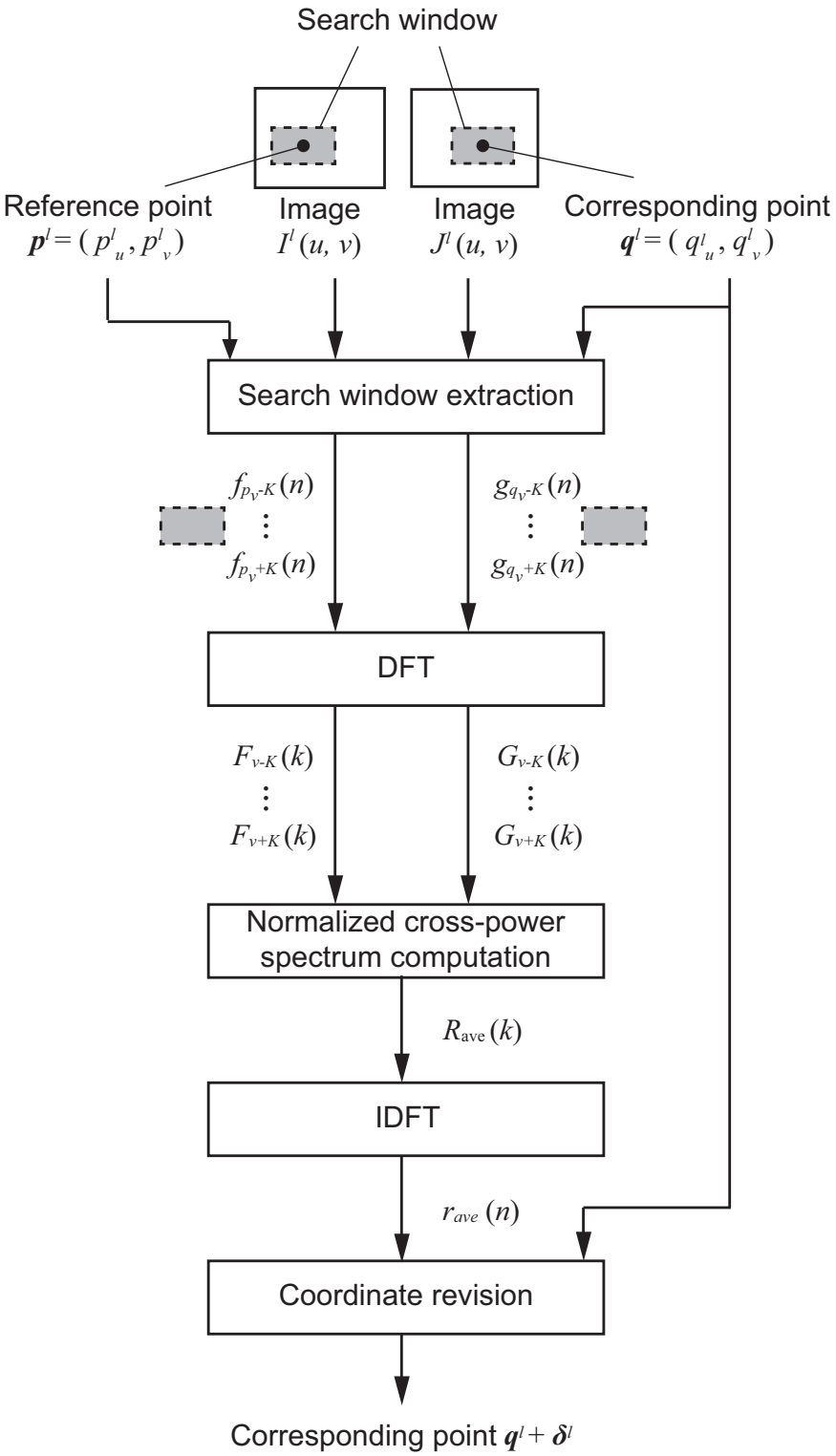


図 3.7　各階層ごとのマッチングを行うカーネル

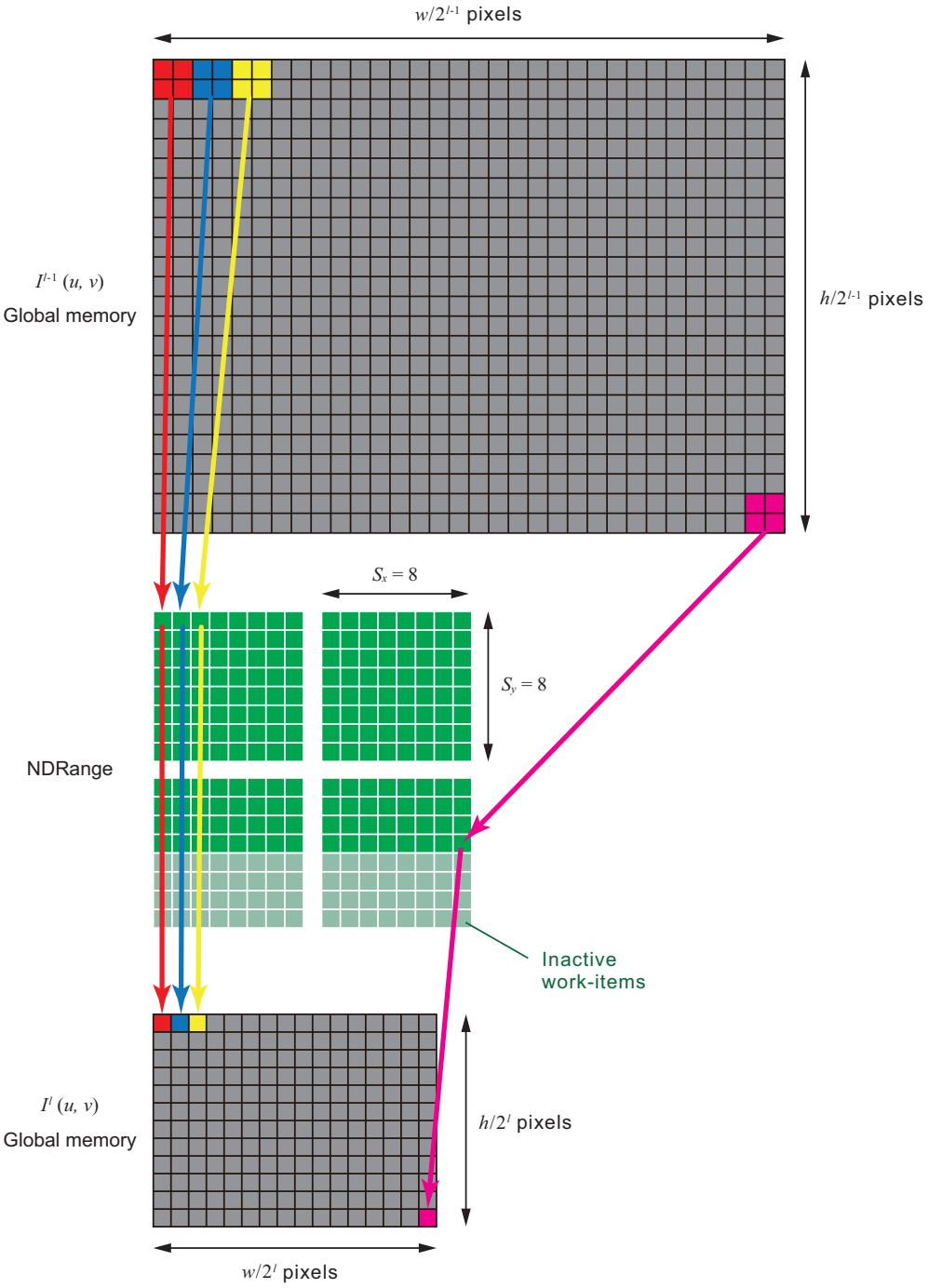


図 3.8　画像ピラミッド生成カーネル．生成する階層画像の画素数と等しいか，より多いワークアイテムを設定してカーネルを呼び出す．各ワークアイテムは，下位層の画像から 4 つの画素の輝度値を読み出し，それらを平均することで，上位層の画像におけるひとつの画素値を得る．

- $I^{l-1}(2g_x, 2g_y)$
- $I^{l-1}(2g_x + 1, 2g_y)$
- $I^{l-1}(2g_x, 2g_y + 1)$
- $I^{l-1}(2g_x + 1, 2g_y + 1)$

をそれぞれ読み出す.

Step2

読み出した 4 つの輝度値を平均する.

Step3

画像 $I^l(u, v)$ における画素 $I^l(g_x, g_y)$ の輝度値としてグローバルメモリに格納する.

画像ピラミッド生成カーネルは，1 度の呼び出しで 1 階層分の階層画像を生成する．$l = 1, 2, \cdots, l_{\max}$ と変化させながら，カーネルを $(l_{\max} - 1)$ 回だけ実行することで，一連の階層画像 $I^1(u, v), I^2(u, v), \cdots, I^{l_{max}}(u, v)$ を生成する．画像 $J(u, v)$ についても同様の処理を行う.

(II)　探索ウィンドウの抽出

ステレオ画像からブロックマッチングに用いる局所的な画像ブロック（探索ウィンドウ）を抽出する．並列化では，ワークグループと基準点，ワークアイテムと探索ウィンドウ内の画素がそれぞれ対応する．すなわち，ひとつのワークグループは基準点 1 点に関する探索ウィンドウを抽出する処理を行い，ひとつのワークアイテムは 1 画素を抽出する処理を行う．したがって，カーネルを実行するワークアイテムは次のようになる.

- NDRange: $(N \times P, L)$
- ワークグループ：(N, L)

カーネルの呼び出しでは，基準点数 P と同じ数だけのワークグループを水平方向に 1 次元的に配置する．各ワークグループは，割り振られたグループのインデックス w_x に基づき，w_x 番目の基準点および対応点に関する探索ウィンドウを抽出する．ただし，ハードウェアによっ

ては，水平方向に設定できるワークグループの最大値が基準点数 P よりも小さい場合がある．このような場合では，ワークグループを 2 次元的に配置することで，基準点数 P と等しい数のワークグループを設定する．このとき，NDRange において水平方向に配置されたワークグループの数 W_x を用いて計算できる線形インデックス $(W_x w_y + w_x)$ に基づいて処理を行う基準点を対応させる．以下では簡単のため，ワークグループが 1 次元的に配置できる場合を想定し，ワークグループを基準点と対応させるためのインデックスを w_x と記述するが，ワークグループを 2 次元的に配置しなければならない場合は，適宜 w_x を $(W_x w_y + w_x)$ に読み替えていただきたい．

以下に各ワークアイテムが行う処理の詳細を述べる．ただし，以下では画像ピラミッドの第 l 階層における画像 $I^l(u, v)$ から基準点の座標をもとに探索ウィンドウを抽出する場合について記述してある．画像 $J^l(u, v)$ についても同様の処理を行うことに注意されたい．

Step 1

所属するワークグループのインデックス w_x に基づき，画像ピラミッドの第 l 階層における基準点 $\boldsymbol{p}^l_{w_x}$ の座標を計算する．グローバルメモリから $\boldsymbol{p}_{w_x}$ を読み出し，水平・垂直座標をそれぞれ $1/2^l$ 倍することで，$\boldsymbol{p}^l_{w_x}$ を得る．なお，画像 $J^l(u, v)$ から探索ウィンドウを抽出する場合は，基準点の垂直座標 $p^l_{w_x u}$ を計算する代わりに，グローバルメモリから対応点の仮の水平座標 $q^l_{w_x u}$ を読み出し，その値を 2 倍する．

Step 2

図 3.9 に示すように，自身に割り振られたインデックス (s_x, s_y) に基づき，画像 $I^l(u, v)$ から探索ウィンドウ内の (s_x, s_y) にあたる輝度値を読み出す．ワークグループ (w_x, w_y) に所属し，インデックス (s_x, s_y) を持つワークアイテムが画像 $I^l(u, v)$ から読み出す画素の水平・垂直座標はそれぞれ以下のように書ける．

$$u|_{w_x, s_x} = p^l_{w_x u} - \left\lfloor \frac{W}{2} \right\rfloor + s_x \tag{3.1}$$

$$v|_{w_x, s_y} = p^l_{w_x v} - \left\lfloor \frac{L}{2} \right\rfloor + s_y \tag{3.2}$$

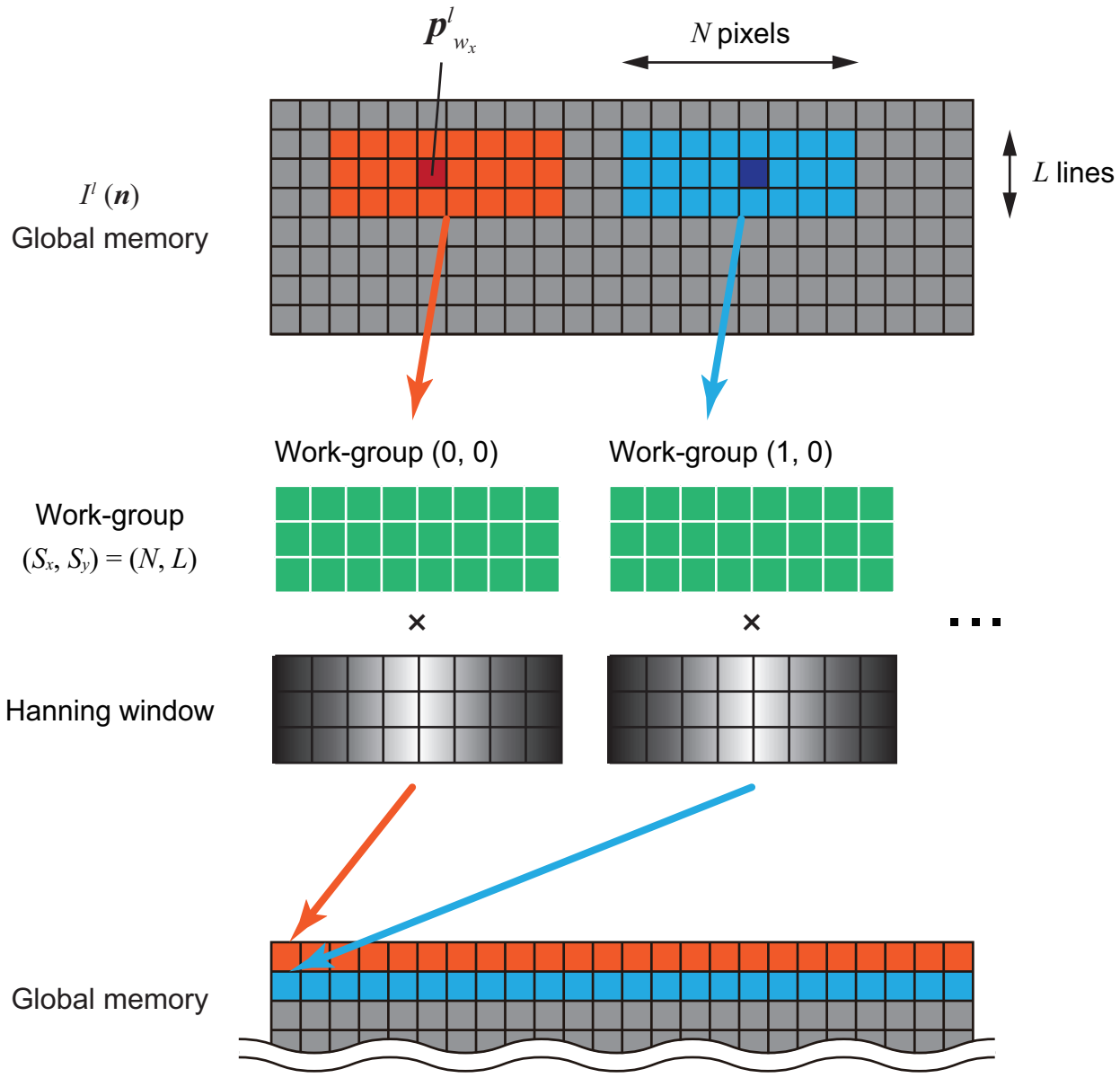


図 3.9　探索ウィンドウ抽出カーネル．ひとつのワークグループは，基準点 1 点に関する探索ウィンドウを抽出する．ワークグループは探索ウィンドウの画素と同じ数だけのワークアイテムから構成されており，ひとつのワークアイテムはひとつの画素に対応する．なお，視認性のため，それぞれのグローバルメモリは 2 次元的に表されているが，実際には 1 次元の配列である．

Step 3

ハニング窓を計算し，読み出した輝度値と掛け合わせる．ハニング窓は，ワークアイテムの水平方向のインデックス s_x に基づいて次式より計算する．

$$w(s_x) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi s_x}{N}\right) \tag{3.3}$$

なお，上式は式 (2.41) における n に s_x を代入したものである．

Step 4

抽出された探索ウィンドウ内の画素の輝度値をグローバルメモリ上のバッファ領域に格納する．バッファ領域では，探索ウィンドウ内の画素の輝度値がワークグループ順，ワークアイテム順に 1 次元的に格納される．すなわち，ワークグループ (w_x, w_y) に所属するワークアイテム (s_x, s_y) が輝度値を格納するアドレス（配列の先頭アドレスからのオフセット）は以下の通りである．

$$a|_{w_x,s_x,s_y} = NL \cdot w_x + N \cdot s_y + s_x \tag{3.4}$$

(III)　離散フーリエ変換

探索ウィンドウの離散フーリエ変換を行う．本論文では，離散フーリエ変換の実装として，Cooley-Tukey 型の高速フーリエ変換 [84]–[86] を用いる．離散フーリエ変換を高速に計算するアルゴリズムとしてさまざまな高速フーリエ変換が知られているが，Cooley-Tukey 型の高速フーリエ変換はそれらの中で最も一般的なアルゴリズムである．以下にその概略を述べる．

N 点の 1 次元信号 $f(n)$ の離散フーリエ変換を以下のように定義する．

$$F(k) = \sum_{n=-M}^{M} f(n) W_N^{kn} \tag{3.5}$$

ただし，$W_N = e^{-j\frac{2\pi}{N}}$ である．まず，$f(n)$ をインデックス n が偶数であるか奇数であるかに

よって二つのクラスに分ける．このとき，離散フーリエ変換 $F(k)$ は次のように記述される．

$$\begin{aligned} F(k) &= \sum_{m=0}^{N/2-1} f(2m) W_N^{k\cdot(2m)} + \sum_{m=0}^{N/2-1} f(2m+1) W_N^{k\cdot(2m+1)} \\ &= \sum_{m=0}^{N/2-1} f(2m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} f(2m+1) W_{N/2}^{km} \end{aligned} \tag{3.6}$$

ただし，

$$W_N^{k\cdot 2m} = W_{N/2}^{km} \tag{3.7}$$

$$W_N^{k\cdot(2m+1)} = W_N^k \cdot W_{N/2}^{km} \tag{3.8}$$

となることを利用している．ここで，

$$F^0(k) = \sum_{m=0}^{N/2-1} f(2m) W_{N/2}^{km} \tag{3.9}$$

$$F^1(k) = \sum_{m=0}^{N/2-1} f(2m+1) W_{N/2}^{km} \tag{3.10}$$

とおけば，$F^0(k)$ と $F^1(k)$ はどちらも $N/2$ 点の離散フーリエ変換であり，これらを用いて式 (3.6) は以下のように記述できる．

$$F(k) = F^0(k) + W_N^k F^1(k) \tag{3.11}$$

式 (3.11) は，N 点の離散フーリエ変換 $F(k)$ が，二つの $N/2$ 点離散フーリエ変換から計算できることを示している．$F(k)$ を $F^0(k)$ と $F^1(k)$ の和に分解したように，$F^0(k)$ と $F^1(k)$ のそれぞれも $N/4$ 点の離散フーリエ変換に分解することができる．このような分解は $(\log_2 N)$ 回だけ再帰的に行うことができる．上記で示したアルゴリズムは，$f(n)$ のインデックスを偶数と奇数に分けていることから，時間間引き型 (decimation in time) と呼ばれる．$f(n)$ のインデックスではなく，$F(k)$ のインデックスに着目して離散フーリエ変換を分解することも可能である．その場合は，周波数間引き型 (decimation in frequency) と呼ばれる．

Cooley-Tukey 型の高速フーリエ変換では，入力または出力の配列がビット逆順 (bit-reversed order) になるという特徴がある．具体的には，時間間引き型の高速フーリエ変換では，入力がビット逆順，出力がビット正順となる．反対に，周波数間引き型の高速フーリエ変

換では，入力がビット正順，出力がビット逆順になる．したがって，一般的には，入力データまたは出力データの配列に対して，ビット逆順に並べ替える作業が必要になる．しかしながら，位相限定相関関数の計算においては，図 3.10 に示すように，順変換において周波数間引き型，逆変換において時間間引き型のアルゴリズムを用いることで，ビット逆順に並べ替える作業が不要となる．

離散フーリエ変換の計算を行うカーネルでは，探索ウィンドウを抽出するカーネルと同様，ワークグループは探索ウィンドウ内の画素と同じ数だけのワークアイテムから構成される．各ワークアイテムは，1 画素および 1 つのフーリエ係数に関する処理を行う．カーネルを実行するワークアイテムは次のようになる．

- NDRange: $(N \times P, L)$
- ワークグループ：(N, L)

カーネルの詳細な処理を以下に述べる．

Step 1

各ワークアイテムは，まず，グローバルメモリ上のバッファ領域から探索ウィンドウ内の画素の輝度値を読み出す．読み出された画素値をローカルメモリに格納し，同一ワークグループ内のワークアイテムが参照できるようにする．

Step 2

次に，FFT のバタフライに従い，ローカルメモリから 2 つの値を読み出し，回転因子を掛け合わせてローカルメモリに書き戻す．各ワークアイテムがこの処理を同期しながら $\log_2 N$ 回だけ繰り返す．その様子を図 3.11 に示す．また，擬似コードを Algorithm 1 に示す．

Step 3

最後に，計算したフーリエ係数をグローバルメモリに書き戻す．

以上の処理を，画像 $I^l(u, v)$ から抽出した探索ウィンドウと，画像 $J^l(u, v)$ から抽出した探索ウィンドウについて行う．

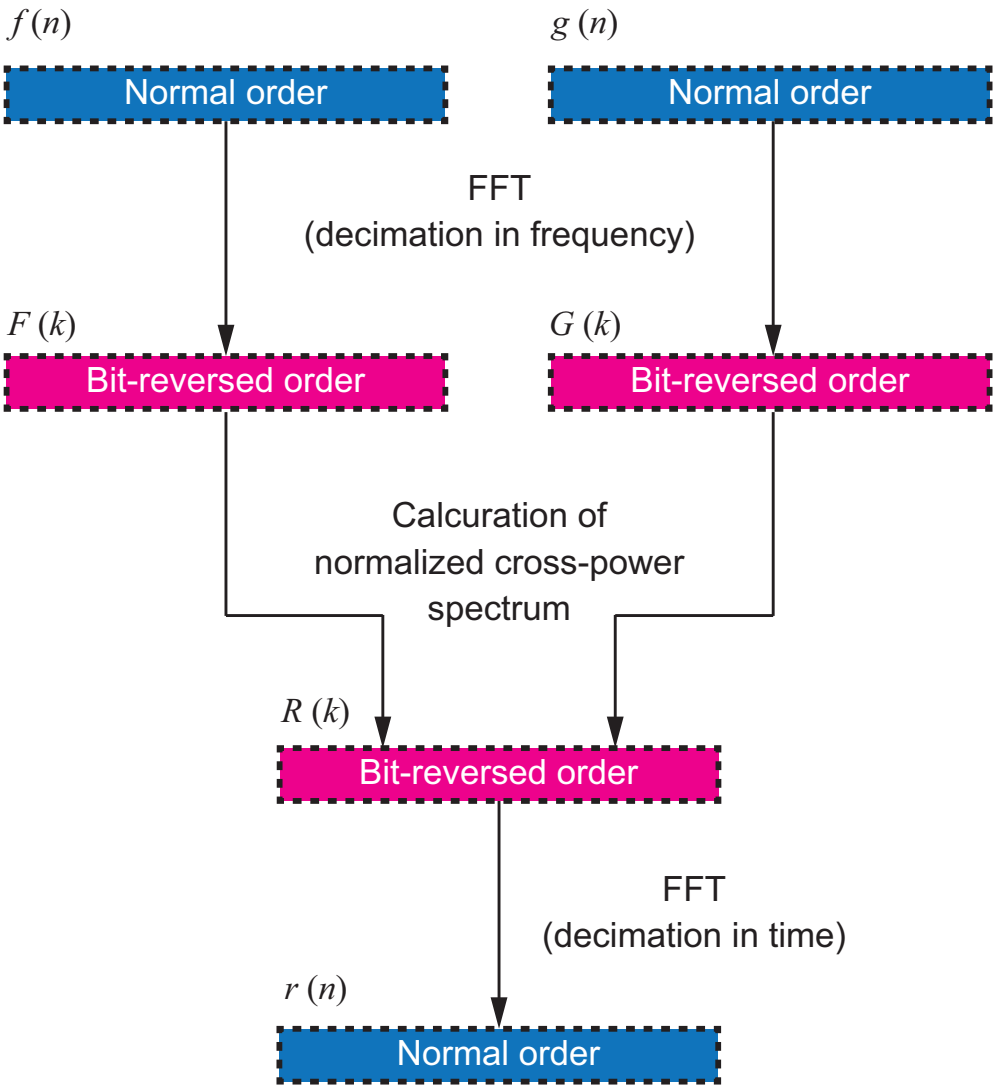


図 3.10　位相限定相関関数の計算におけるデータのインデックス配置．まず，入力信号 $f(n)$ および $g(n)$ をそれぞれ周波数間引き型のアルゴリズム（入力がビット正順，出力がビット逆順）を用いて高速フーリエ変換する．次に，得られた $F(k)$ および $G(k)$ を用いて，ビット逆順のインデックスを考慮せずに正規化相互クロスパワースペクトル $R(k)$ を計算する．最後に，時間間引き型のアルゴリズム（入力がビット逆順，出力がビット正順）を用いて高速フーリエ逆変換を計算する．最終的に得られた POC 関数 $r(n)$ はビット正順となる．

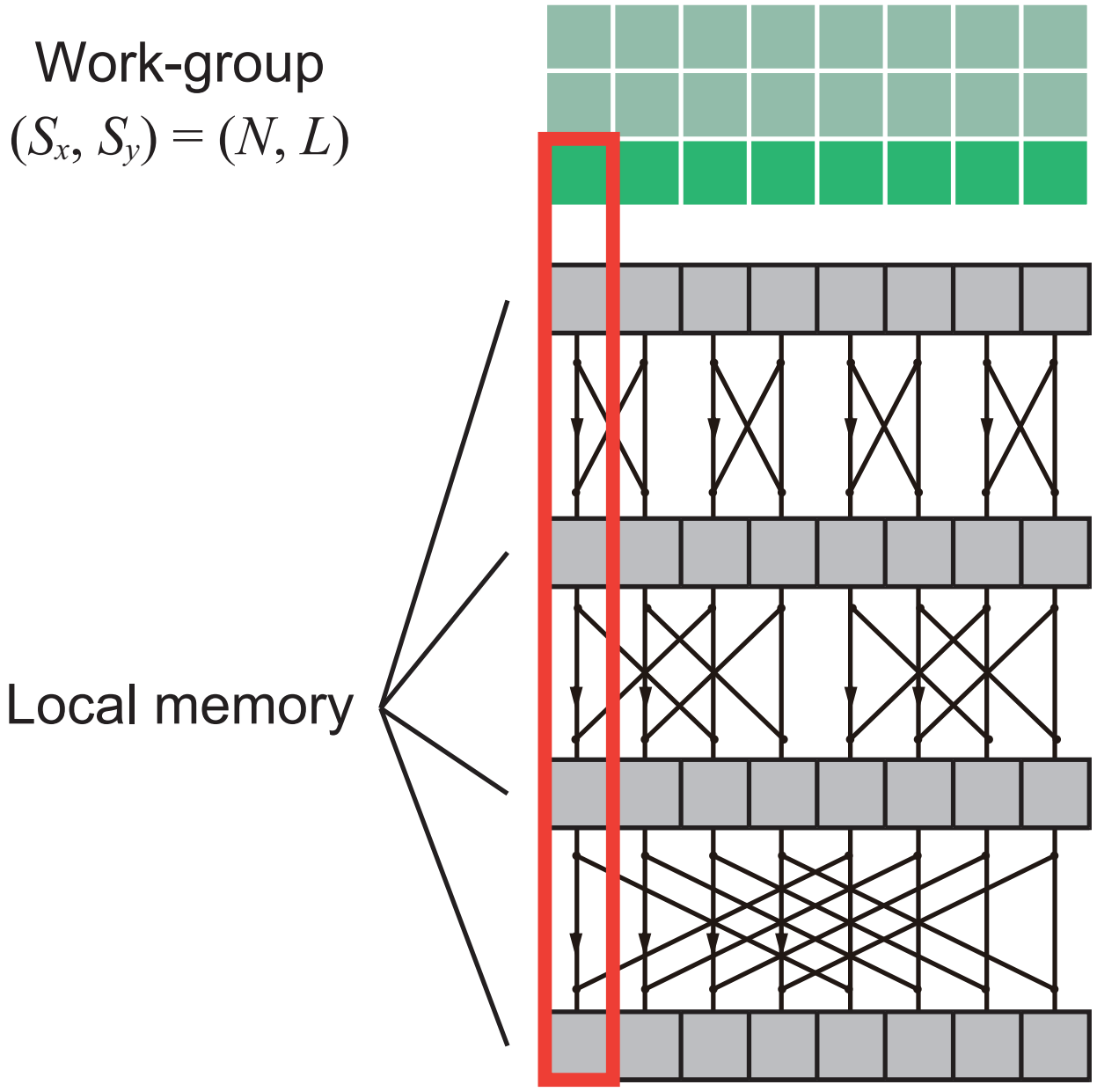


図 3.11　離散フーリエ変換カーネル．各ワークアイテムは，ローカルメモリを用いて他のワークアイテムと通信しながら高速フーリエ変換を行う．ひとつのワークアイテムはひとつのフーリエ係数およびその中間値を計算する．

Algorithm 1 N 点の 1 次元信号の高速フーリエ変換におけるワークアイテム (s_x, s_y) のバタフライ演算

$c := \log_2 N - 1$

$d := 2^c$

$s := 1$

for $i = 0$ to c **do**

$a := s_x \wedge (\neg d)$

$f := (s_x \wedge d)$ >>$(c - i)$

if $f = 0$ **then**

$m := 1$

else

$m := -1$

end if

$M_{\text{temp}} = M[s_y][a] + M[s_y][a + d] \times m$

wait the other work-items

$t \Leftarrow a \times s \times f$

$W \Leftarrow W_N^t = \exp\left(-j\frac{2\pi t}{N}\right) = \cos\left(\frac{2\pi t}{N}\right) - j\sin\left(\frac{2\pi t}{N}\right)$

$M[s_y][s_x] = M_{\text{temp}} \times W$

wait the other work-items

$d \Leftarrow d$ >>1

$s \Leftarrow s$ <<1

end for

(IV)　正規化相互パワースペクトルの計算

正規化相互パワースペクトルを計算する．正規化相互パワースペクトルの計算では，$F_i(k)$ および $G_i(k)$ から $R_i(k)$ を計算した後, 式 (2.47) に示すように $i = -K, -K+1, \cdots, K-1, K$ について $R_i(k)$ を平均し，R_{ave} を得る．ここで，正規化相互パワースペクトルの計算を k および i ごとに並列化する．すなわち，ひとつのワークアイテムは一組の i と k に対する $R_i(k)$ を計算する．カーネルを実行するワークアイテムは次のようになる．

- NDRange: $(N \times P, L)$
- ワークグループ：(N, L)

カーネルが実行される様子を図 3.12 に示す．以下に，ワークアイテムにおける詳細な処理を示す.

Step 1

グローバルメモリより，式 (3.4) で示されたアドレスからフーリエ係数 $F_i(k)$ および $G_i(k)$ をそれぞれ読み出す.

Step 2

読み出した値を用いて式 (2.39) の通り $R_i(k)$ を計算する．その後，計算した値をローカルメモリに格納する.

Step 3

ローカルメモリに格納された正規化相互パワースペクトルを平均する．この処理は，ワークグループにおける垂直方向のインデックス s_y が 0 であるワークアイテムのみ行う．すなわち，ワークアイテム $(s_x, 0)$ は，$i = -K, -K+1, \cdots, K-1, K$ について $R_i(s_x)$ を以下の通り計算する.

$$R_{\mathrm{ave}}(s_x) = \frac{1}{L} \sum_{i=-K}^{K} R_i(s_x) \tag{3.12}$$

上式は，式 (2.47) の k に s_x を代入したものである.

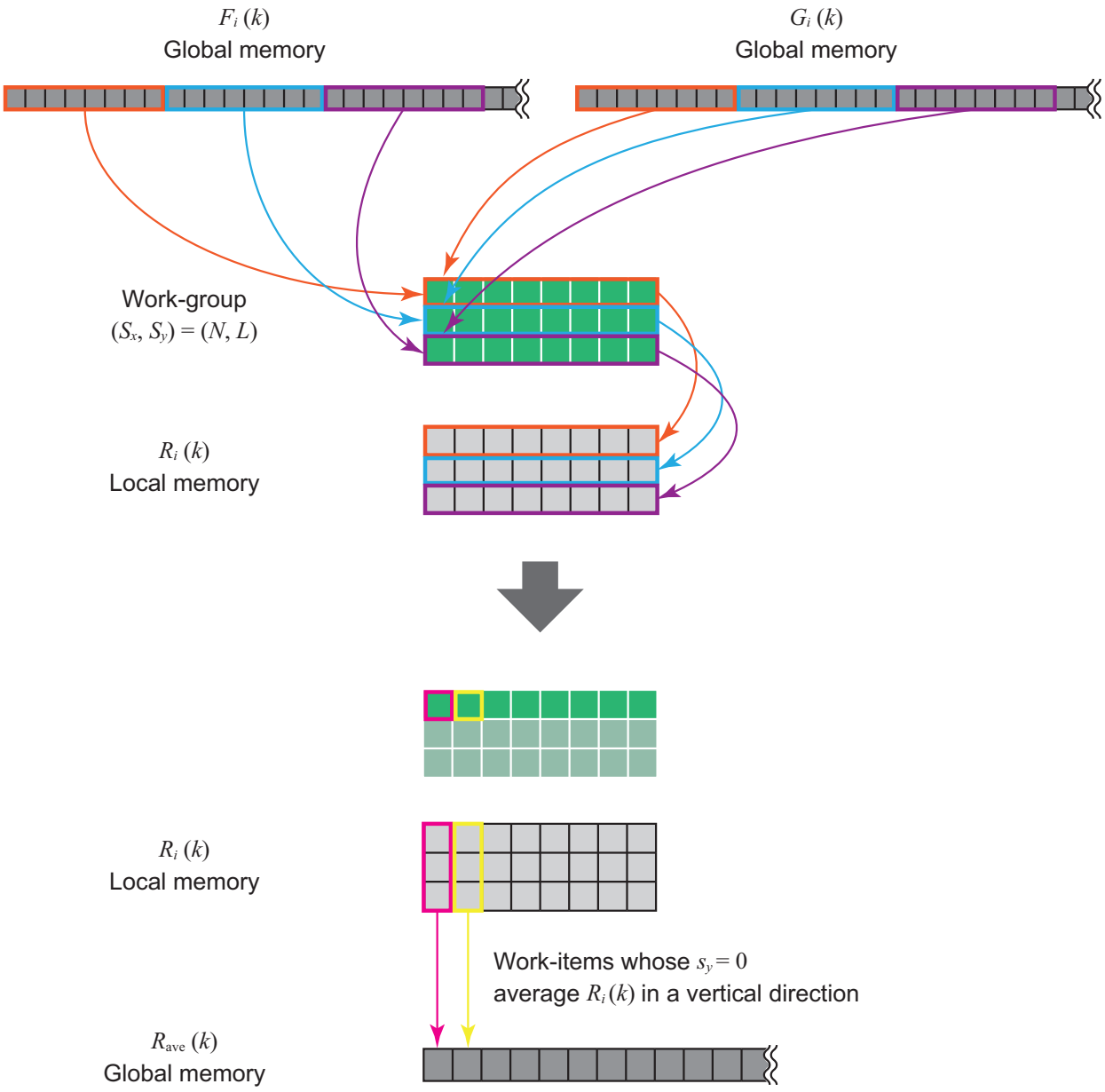


図 3.12　正規化相互パワースペクトル計算カーネル．各ワークアイテムは，あるラインにおける正規化相互パワースペクトルのうち，ひとつの要素だけを計算する．それぞれのワークアイテムは計算結果をローカルメモリに格納する．その後，1 行目のワークアイテム（垂直方向のインデックスが 0 であるワークアイテム）のみ，ローカルメモリを垂直方向に沿って読み込み，それらの値を平均する．平均された値はグローバルメモリに書き戻される．

Step 4

$R_{\mathrm{ave}}(k)$ をグローバルメモリに書き戻す．**Step 3** と同様に，この処理は，ワークグループにおける垂直方向のインデックス s_y が 0 であるワークアイテムのみ行う．格納先のアドレス（配列の先頭アドレスからのオフセット）は以下の通りである．

$$a_s|_{w_x,s_x} = N \cdot w_x + s_x \tag{3.13}$$

正規化相互パワースペクトルを探索ウィンドウのライン数だけ平均したため，アドレスが式 (3.4) と異なることに注意されたい．

(V)　離散フーリエ逆変換

正規化相互パワースペクトルを離散フーリエ逆変換し，位相限定相関関数（POC 関数）を得る．離散フーリエ変換と同様に，ひとつのワークアイテムはひとつのフーリエ係数に対する処理を行う．すなわち，カーネルを呼び出すワークアイテムは次のようになる．

- NDRange: $(N \times P, 1)$
- ワークグループ：$(N, 1)$

(IV) にて正規化相互パワースペクトルが平均されているため，ワークグループの垂直方向のサイズは探索ウィンドウのライン数 L ではなく，1 となることに注意されたい．

以下に，それぞれのワークアイテムにおける詳細な処理を述べる．

Step 1

ワークアイテム (s_x, s_y) は，平均された正規化相互パワースペクトル $R_{\mathrm{ave}}(k)$ をグローバルメモリから読み出す．読み出された値をローカルメモリに格納し，同一ワークグループ内のワークアイテムが参照できるようにする．

Step 2

次に，FFT のバタフライに従い，ローカルメモリから 2 つの値を読み出し，回転因子を掛け合わせてローカルメモリに書き戻す．各ワークアイテムがこの処理を同期しながら $\log_2 N$ 回だけ繰り返す．その擬似コードを Algorithm 2 に示す．

Algorithm 2 N 点の 1 次元信号の高速フーリエ逆変換（時間間引き）におけるバタフライ演算

$c := \log_2 N - 1$

$d := 1$

$s := 2^c$

for $i = 0$ to c **do**

$a := s_x \wedge (\neg d)$

$M_1 = M[a]$

$M_2 = M[a + d]$

wait the other work-items

$t \Leftarrow s_x \times s$

$W \Leftarrow W_N^t = \exp\left(-j\frac{2\pi t}{N}\right) = \cos\left(\frac{2\pi t}{N}\right) - j\sin\left(\frac{2\pi t}{N}\right)$

$M[s_x] = M_1 + M_2 \times W$

wait the other work-items

$d \Leftarrow d$ << 1

$s \Leftarrow s$ >> 1

end for

Step 3

最後に，計算結果（位相限定相関関数 $r_{\mathrm{ave}}(n)$）をグローバルメモリに書き戻す.

(VI)　対応点座標の更新

POC 関数における相関ピークの位置を探索し，対応点の座標を更新する．カーネルを呼び出すワークアイテムは次のように設定する.

- NDRange: $(N \times P, 1)$
- ワークグループ：$(N, 1)$

カーネルが実行される様子を図 3.13 に示す．以下に，ワークグループ (w_x, w_y) に属するワークアイテム (s_x, s_y) の詳細な処理を述べる．

Step 1

グローバルメモリから POC 関数を読み出し，ローカルメモリに格納する．

Step 2

ワークグループ内のインデックス $(s_x, s_y) = (0, 0)$ であるワークアイテムは，ローカルメモリを順に読み出し，POC 関数の最大値を探索する．

Step 3

Step 2 で探索した最大値の位置を用いてグローバルメモリ上の対応点の座標 $q^l_{w_x u}$ を更新する．また，最大値を α_{w_x} としてグローバルメモリに格納する．ただし，サブピクセル精度での対応付けを行う場合では，探索した最大値の位置および値に基づき，2.4.3 節に述べた方法を用いてサブピクセル分解能の平行移動量および相関ピーク値を計算し，$q^l_{w_x u}$ および α_{w_x} を更新する．

3.3.2　実装の最適化

3.3.1 節では，デザインパターン **D1** をもとに 1 次元 POC に基づく画像対応付けアルゴリズムを並列化し，GPU に実装する手法を示した．本節では，同実装に対して，デザインパターン **D2**∼**D4** をもとに 4 つの実装の工夫 **(T1)**∼**(T4)** を施す．まず，**D3** については，探索ウィンドウにおける画素ごとの処理に対してワークアイテムを 1 対 1 で対応させた結果，グローバルメモリのアクセスパターンが Coalesced になり，達成されている．次に，**D2** について，**(T1)** 2 組の実数列の同時 DFT を用いた DFT 回数の削減および **(T2)** 回転因子のテーブル実装によって演算回数とメモリアクセス量を削減する．さらに，**D3** について，**(T3)** 適切なワークグループサイズの選択により，ハードウェアの稼働率を向上させる．最後に，**(T4)** ローカルメモリを使用したカーネルの統合を行うことで，グローバルメモリへのアクセス量を削減する．これは **D2** に当てはまるが，同時に，同期やカーネル呼び出しのオー

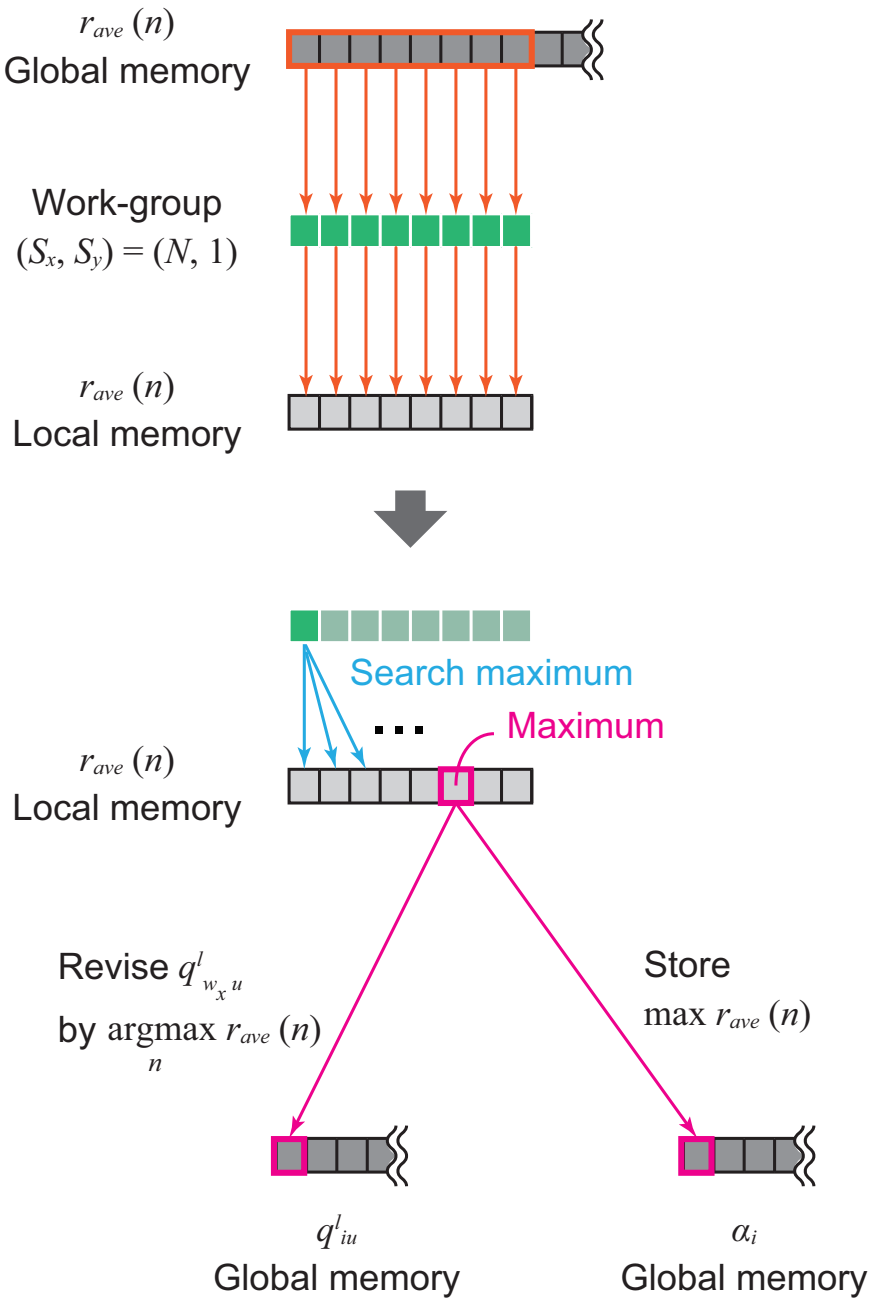


図 3.13　対応点座標の更新．各ワークアイテムはそれぞれ POC 関数のひとつの値をグローバルメモリから読み出し，ローカルメモリに保存する．その後，$(s_x, s_y) = (0, 0)$ なるワークアイテム，すなわち先頭のワークアイテムは，ローカルメモリを順に参照し，POC 関数の最大値を探索する．最大値の位置を用いてグローバルメモリ上に格納されている対応点の座標を更新する．同様に，最大値を相関ピークの値としてグローバルメモリに格納する．

バーヘッドの削減も達成できている．以下では，これらの実装の工夫 **(T1)**∼**(T4)** について詳細を述べる．

(T1) 2 組の実数列の同時 DFT を用いた DFT 回数の削減

対応付け処理では，抽出した探索ウィンドウのそれぞれにおいて，離散フーリエ変換を計算する．離散フーリエ変換を行うカーネルでは，他のカーネルと比べ，ローカルメモリへのメモリアクセス，ワークアイテム間の同期，および演算量が多いため，全体の処理のボトルネックになることが予想される．そこで，実数信号の離散フーリエ変換の性質を利用して，基準点を中心に抽出した探索ウィンドウと，対応点を中心に抽出した探索ウィンドウを同時に離散フーリエ変換することで，離散フーリエ変換の回数を 2 回から 1 回に削減する．

基準画像から抽出した 1 次元信号 $f(n)$ と，参照画像から抽出した 1 次元信号 $g(n)$ より，実数部に $f(n)$，虚数部に $g(n)$ を持つ信号 $m(n)$ を次式のように定義する．

$$m(n) = f(n) + jg(n) \tag{3.14}$$

$m(n)$ の離散フーリエ変換を $M(k)$ とすると，$M(k)$ より $F(k)$ および $G(k)$ は以下のように計算できる．

$$F(k) = \frac{\mathrm{Re}\{M(k) + M(-k)\}}{2} + j\frac{\mathrm{Im}\{M(k) - M(-k)\}}{2} \tag{3.15}$$

$$G(k) = \frac{\mathrm{Im}\{M(k) + M(-k)\}}{2} - j\frac{\mathrm{Re}\{M(k) - M(-k)\}}{2} \tag{3.16}$$

その導出を以下に示す．まず，$m(n)$ の離散フーリエ変換 $M(k)$ は次式のように計算できる．

$$\begin{aligned} M(k) &= F(k) + jG(k) \\ &= [\mathrm{Re}\{F(k)\} + j\mathrm{Im}\{F(k)\}] + j\,[\mathrm{Re}\{G(k)\} + j\mathrm{Im}\{G(k)\}] \\ &= [\mathrm{Re}\{F(k)\} - \mathrm{Im}\{G(k)\}] + j\,[\mathrm{Im}\{F(k)\} + \mathrm{Re}\{G(k)\}] \end{aligned} \tag{3.17}$$

ここで，$f(n)$ および $g(n)$ は実数の信号であるため，その離散フーリエ変換について，実部は偶関数，虚部は奇関数となる．すなわち，次式が成り立つ．

$$\mathrm{Re}\{F(-k)\} = \mathrm{Re}\{F(k)\} \tag{3.18}$$

$$\mathrm{Im}\{F(-k)\} = -\mathrm{Im}\{F(k)\} \tag{3.19}$$

これを用いて，$M(k)+M(-k)$ を計算すると，以下のようになる．

$$\begin{aligned}M(k)+M(-k)=&[\mathrm{Re}\{F(k)\}-\mathrm{Im}\{G(k)\}]+j\,[\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\&+[\mathrm{Re}\{F(-k)\}-\mathrm{Im}\{G(-k)\}]+j\,[\mathrm{Im}\{F(-k)\}+\mathrm{Re}\{G(-k)\}]\\=&[\mathrm{Re}\{F(k)\}-\mathrm{Im}\{G(k)\}]+j\,[\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\&+[\mathrm{Re}\{F(k)\}+\mathrm{Im}\{G(k)\}]+j\,[-\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\=&2\,[\mathrm{Re}\{F(k)\}+j\mathrm{Re}\{G(k)\}]\end{aligned}\tag{3.20}$$

同様に，$M(k)-M(-k)$ は，以下のように計算できる．

$$\begin{aligned}M(k)-M(-k)=&[\mathrm{Re}\{F(k)\}-\mathrm{Im}\{G(k)\}]+j\,[\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\&-[\mathrm{Re}\{F(-k)\}-\mathrm{Im}\{G(-k)\}]-j\,[\mathrm{Im}\{F(-k)\}+\mathrm{Re}\{G(-k)\}]\\=&[\mathrm{Re}\{F(k)\}-\mathrm{Im}\{G(k)\}]+j\,[\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\&-[\mathrm{Re}\{F(k)\}+\mathrm{Im}\{G(k)\}]-j\,[-\mathrm{Im}\{F(k)\}+\mathrm{Re}\{G(k)\}]\\=&2\,[-\mathrm{Im}\{G(k)\}+j\mathrm{Im}\{F(k)\}]\end{aligned}\tag{3.21}$$

$M(k)+M(-k)$ と $M(k)-M(k)$ の実部および虚部に着目すると，

$$\mathrm{Re}\{M(k)+M(-k)\}=2\mathrm{Re}\{F(k)\}\tag{3.22}$$

$$\mathrm{Im}\{M(k)+M(-k)\}=2\mathrm{Re}\{G(k)\}\tag{3.23}$$

$$\mathrm{Re}\{M(k)-M(-k)\}=-2\mathrm{Im}\{G(k)\}\tag{3.24}$$

$$\mathrm{Im}\{M(k)-M(-k)\}=2\mathrm{Im}\{F(k)\}\tag{3.25}$$

となっている．すなわち，

$$\mathrm{Re}\{F(k)\}=\frac{\mathrm{Re}\{M(k)+M(-k)\}}{2}\tag{3.26}$$

$$\mathrm{Re}\{G(k)\}=\frac{\mathrm{Im}\{M(k)+M(-k)\}}{2}\tag{3.27}$$

$$\mathrm{Im}\{G(k)\}=-\frac{\mathrm{Re}\{M(k)-M(-k)\}}{2}\tag{3.28}$$

$$\mathrm{Im}\{F(k)\}=\frac{\mathrm{Im}\{M(k)-M(-k)\}}{2}\tag{3.29}$$

が成り立つ．これら 4 つの式より，式 (3.15) および式 (3.16) が導出できる．

同時 DFT を用いることで削減できる計算量とメモリアクセス量について考察する．メモリアクセスの削減量については，グローバルメモリとローカルメモリへのそれぞれのアクセスについて取り扱う．

はじめに，計算量について述べる．高速フーリエ変換を用いて N 点の 1 次元信号の離散フーリエ変換の計算をするには，$N\log_2 N$ 回の複素加算と複素乗算が必要であることが知られている [86]．ここで，複素加算は 2 回の加算，複素乗算は 2 回の加算と 4 回の乗算によって実装されるものとする．すると，N 点の 1 次元信号の高速フーリエ変換における計算回数は，加算と乗算がそれぞれ $4N\log_2 N$ 回となる．したがって，それぞれ L ラインからなる探索ウィンドウについて離散フーリエ変換を計算するには，加算と乗算がそれぞれ $4NL\log_2 N$ 回行われることになる．また，式 (3.15) と式 (3.16) より，$M(k)$ から $F(k)$ と $G(k)$ を計算するために必要な演算は，加算と乗算がそれぞれ 4 回である．ただし，ここでは，2 での除算を 0.5 の乗算として実装するものとする．探索ウィンドウが N 点の 1 次元信号 L ラインからなるとすると，$4NL$ 回の加算と乗算が必要である．以上より，同時 DFT を用いることで削減できる演算回数の割合は，

$$\frac{4NL\log_2 N + 4NL}{(4NL\log_2 N)\times 2} = \frac{\log_2 N + 1}{2\log_2 N} \tag{3.30}$$

となる．$N = 32$ のとき，式 (3.30) は $\frac{3}{5}$ となり，すなわち演算回数が $\frac{3}{5}$ 倍に削減できることを示す．

次に，グローバルメモリへのアクセス量について述べる．1 つの探索ウィンドウに含まれる N 点の 1 次元信号を L ラインだけ離散フーリエ変換するとき，NL ワードをグローバルメモリから読み出す．それら 1 次元信号の離散フーリエ変換をグローバルメモリへ書き戻すとき，そのデータ量は $2NL$ ワードになる．ここで，入力は実数であり 1 ワード（単精度の浮動小数点数 4 バイト）であるのに対し，出力は複素数で 2 ワードであることに注意されたい．2 つの探索ウィンドウをそれぞれ離散フーリエ変換するのに比べると，同時 DFT では，入力となるデータ量は等しいが，出力となるデータ量は半分となる．したがって，削減できるグローバルメモリへのアクセス量は，

$$\frac{NL + NL + 2NL}{(NL + 2NL)\times 2} = \frac{2}{3} \tag{3.31}$$

となる．すなわち，探索ウィンドウの大きさに関わらず，グローバルメモリへのアクセス量は $\frac{2}{3}$ 倍に減少する．

最後に，ローカルメモリへのアクセス量について述べる．N 点の 1 次元信号の離散フーリ

エ変換では，$\log_2 N$ 段のバタフライ演算を行う．バタフライ演算では，複素数を用いた計算を行うため，1 段あたりローカルメモリへの読み込みと書き込みがそれぞれ $2N$ ワード発生する．また，同時 DFT では，後段のカーネル（正規化相互パワースペクトル）において $M(k)$ から $F(k)$ と $G(k)$ を計算する必要がある．この計算において発生するローカルメモリへのアクセスは，式 (3.15) と式 (3.16) に現れる $M(k)$ と $M(-k)$ の数より，$8N$ ワードである．探索ウィンドウには N 点の 1 次元信号が L ライン含まれることを考慮すると，削減できるローカルメモリへのアクセス量は，

$$\frac{2NL\log_2 N + 8NL}{(2NL\log_2 N)\times 2} = \frac{\log_2 N + 4}{2\log_2 N} \tag{3.32}$$

となる．$N = 32$ のとき，式 (3.32) は $\frac{9}{10}$ となり，すなわちローカルメモリへのメモリアクセス量が $\frac{9}{10}$ 倍に削減できることを示す．

以上の考察より，同時 DFT を用いることによって，演算回数，グローバルメモリへのアクセス量，ローカルメモリへのアクセス量のすべてが削減できることがわかる．

(T2) 回転因子のテーブル実装

GPU では，浮動小数点の加算や乗算に対して，sin や cos などの超越関数の計算コストが高い．また，超越関数の変数によっては，グローバルメモリをバッファとした計算が行われることもあり，グローバルメモリへのアクセスレイテンシによってパフォーマンスが低下する．離散フーリエ変換では，信号長が既知であるとき，計算中で用いる回転因子は一意に定まる．したがって，カーネルの呼び出しに際して一度だけ回転因子を計算し，ローカルメモリに格納することでテーブルとして参照できるようにする．

(T3) 適切なワークグループサイズの選択

3.3.1 節で述べた実装では，N ピクセル $\times$ L ラインの探索ウィンドウに対して，1 ワークグループあたり $N \times L$ 個のワークアイテムを用いて並列処理を行っている．このような並列化では，探索ウィンドウが大きい場合，各ワークグループが使用するリソースが多くなり，演算ユニットにおいてアクティブな（一度にスケジューリングや実行ができる）ワークグループの数が減少してしまうことがある．そこで，N ピクセル $\times$ L ラインの探索ウィンドウに対して，1 ワークグループあたり $N \times L'$ 個のワークアイテムを用意し，$\lceil \frac{L}{L'} \rceil$ 回の反復処理を行

う．ここで，最適な L' は GPU のアーキテクチャや探索ウィンドウの大きさによって変わるため，実測に基づいて最適な L' を選ぶ．

(T4) ローカルメモリを使用したカーネルの統合

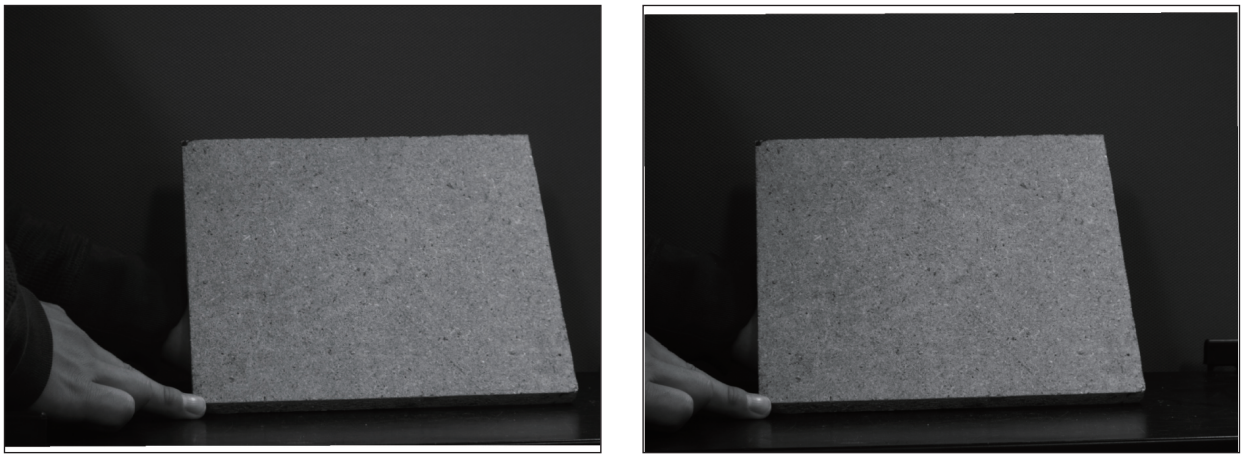
一般に，GPU を用いた実装においては，ワークグループおよびワークアイテムごとのリソース使用量（ローカルメモリやプライベートメモリの使用量）を抑え，パフォーマンスを向上するために，カーネルを可能な限り短くすることが高いパフォーマンスを得るために重要とされている．この知見に従ってカーネルを設計した場合，POC に基づく画像対応付けアルゴリズムは，3.3.1 節で述べたように，(i) 探索ウィンドウ抽出カーネル，(ii) 離散フーリエ変換カーネル (iii) 正規化相互パワースペクトル計算カーネル，(iv) 離散フーリエ逆変換カーネル，(v) 対応点の座標更新カーネルといった 5 つのカーネルに分割できる．しかし，これらのカーネルはそれぞれ，グローバルメモリからデータを読み出して処理し，結果をグローバルメモリに書き戻す．GPU におけるグローバルメモリは，オフ・チップの記憶領域（ビデオメモリ）であり，レイテンシが高い．したがって，このようにカーネルを分割した場合では，全体としてパフォーマンスが低下してしまう．そこで，(i)∼(v) のカーネルを統合し，グローバルメモリの代わりにローカルメモリを用いて処理結果を受け渡す．ローカルメモリは，GPU では各コア内で共有されるオン・チップの記憶領域であるので，高速なアクセスが可能である．

3.4　性能評価

本節では，POC に基づく画像対応付けアルゴリズムを 3.3 節で述べた手法によって GPU に実装し，その速度評価を行う．まず，GPU における処理速度が，提案する実装手法によって高速化されることを示す．次に，CPU を用いた実装と処理速度や電力効率を比較することで，GPU を用いた実装の優位性を示す．

3.4.1　GPU 実装の性能評価

POC に基づく画像対応付けアルゴリズムを 3.3 節で述べた手法を用いてそれぞれ実装し，処理時間を計測することで，実装の工夫による速度向上を確認する．実験環境は表 3.3 に



(a)　(b)

図 3.14　性能評価実験に用いるステレオ画像：(a) 左カメラ画像，(b) 右カメラ画像

示す通りである．GPU は，NVIDIA 社および AMD 社から販売されている合計 8 種類を用いる．3.3.1 節に述べた並列化手法のみを適用した素朴な実装を I1 とし, I1 に手法 (T1), (T1)∼(T2), (T1)∼(T3), (T1)∼(T4) を適用した実装をそれぞれ I2, I3, I4, I5 とする．ただし，AMD 製の GPU では，ワークグループあたりのワークアイテム数が 256 までという制限があるため，I1 においてワークグループあたり $(S_x, S_y) = (32, 8)$ 個のワークアイテムを用いて処理を行う．入力画像は図 3.14 に示すステレオ画像である．左右カメラの画像はともにモノクロ画像で，解像度は 1,280×960 ピクセルである．対応付けを行う基準点は 10,000 点であり，左カメラ画像において平板が写っている領域に 5 ピクセル間隔で格子状に 100×100 点を配置する．画像対応付けのパラメータは，探索ウィンドウの大きさを 32 ピクセル ×15 ライン, 階層数を 4 とする．処理時間は，入力データ（ステレオ画像と基準点の座標）がホストから GPU に転送される直前から，出力データ（対応点の座標と相関ピーク値）が GPU からホストに転送された直後までを計測する．時間の計測には，Windows の時間測定 API である `QueryPerformanceCounter` 関数と `QueryPerformanceFrequency` 関数とを組み合わせて用いる [87], [88]．それぞれの実装に対して 100 回の時間測定を行い，それらの平均を取る．

処理時間の計測結果を図 3.15 に示す．また，素朴な実装である I1 に対し，最適化を行った実装 I2∼I5 の速度向上比を表 3.4 に示す．表 3.4 より，各種の実装最適化がアーキテクチャの異なるどの GPU においても有効であり，GPU を用いた画像対応付けが 2.2 ∼ 3.5 倍高速化

表 3.3　GPU 実装および CPU 実装に関する評価環境

CPU	Intel Core i7-3770K (3.5 GHz)
メモリ	4 GB (DDR3-1333 SDRAM 4 GB×1)
チップセット	Intel Z77 Chipset
GPU	AMD Radeon HD 5870
	AMD Radeon HD 6970
	AMD Radeon HD 7970
	AMD Radeon R9 290X
	NVIDIA GeForce GTX 480
	NVIDIA GeForce GTX 580
	NVIDIA GeForce GTX 680
	NVIDIA GeForce GTX 780
Driver	GeForce Driver 344.11 (NVIDIA GPU)
	AMD Driver 14.301.1001 (AMD GPU)
OS	Windows 7 (32 bit)
コンパイラ	Visual C++ 9.0 Compiler (CPU code)

されていることが確認できる．

実装 I2 では，(T1) 2 組の実数列の同時 DFT を用いた DFT 回数の削減によって，すべての GPU において約 1.5 倍の高速化が達成できている．実装 I3 では，(T2) 回転因子のテーブル実装によって，すべての GPU において 1.7 ～ 2.1 倍の高速化が達成できている．実装 I4 では，(T3) 適切なワークグループサイズの選択によってすべての GPU において高速化が達成できている．各 GPU における，ワークグループのサイズを変化させたときのカーネルの処理時間を表 3.5 および表 3.6 に示す．ワークグループのサイズによって，各カーネルの処理時間が最大で 2 倍以上長くなることがわかる．表 3.4 において，実装 I3 から実装 I4 への速度向上に着目すると，NVIDIA GPU が 0.4 ～ 0.5 倍の向上が見られるのに対し，AMD GPU では 0.02 ～ 0.05 倍程度しか向上していない．これは，実験の条件でも述べたとおり，

表 3.4　実装の工夫に対する速度向上比（素朴な実装 I1 を基準とする）

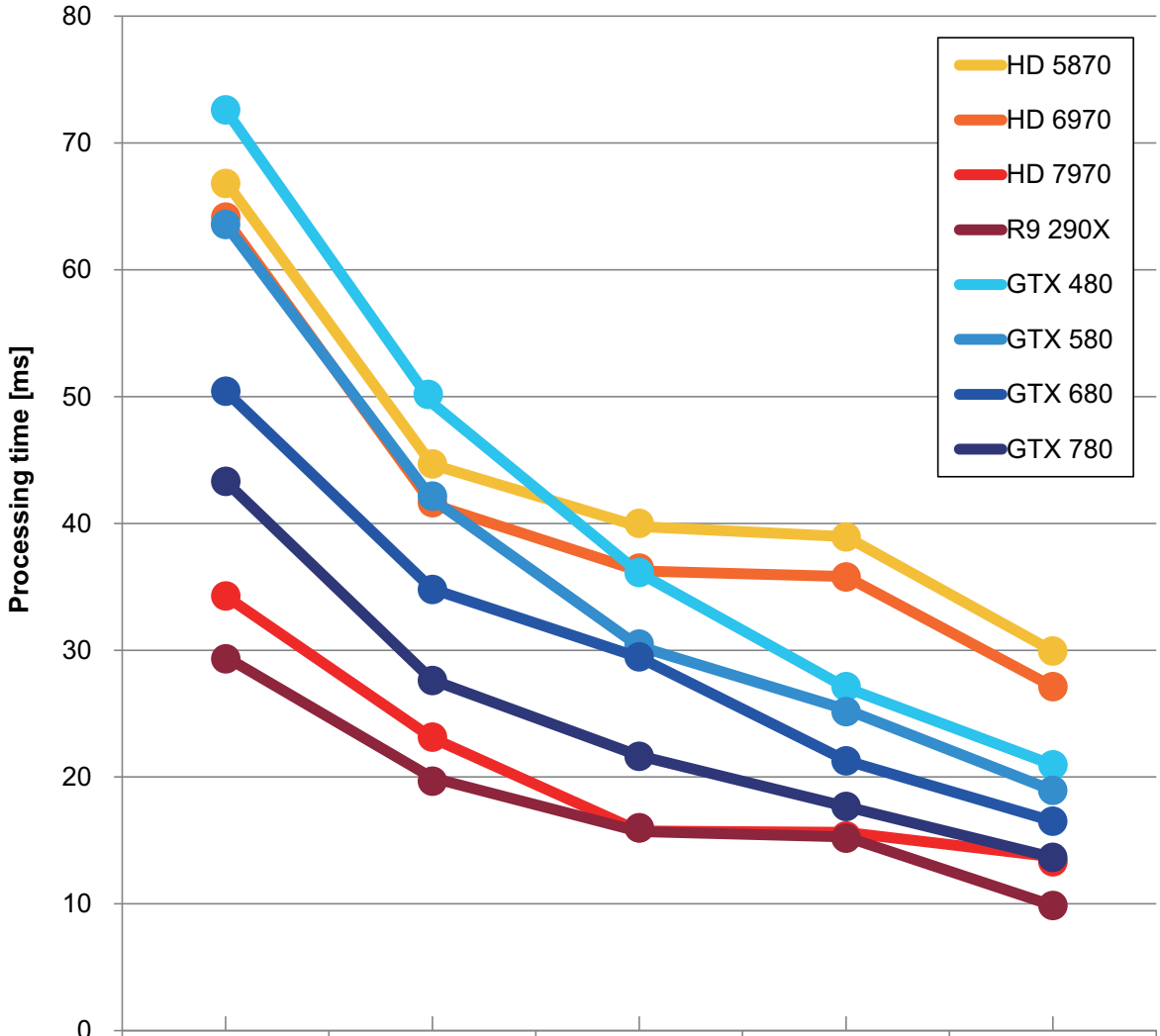
	I1	I2	I3	I4	I5
HD 5870	1.00	1.50	1.68	1.72	2.24
HD 6970	1.00	1.54	1.77	1.79	2.36
HD 7970	1.00	1.48	2.18	2.19	2.50
R9 290X	1.00	1.48	1.88	1.93	2.99
GTX 480	1.00	1.47	2.01	2.68	3.46
GTX 580	1.00	1.52	2.10	2.51	3.36
GTX 680	1.00	1.45	1.71	2.37	3.05
GTX 780	1.00	1.57	2.00	2.45	3.17

AMD GPU におけるワークアイテム数の制限から，実装 I1 の時点でワークグループあたり $(S_x, S_y) = (32, 8)$ のワークアイテムを用いて処理を行っていたため，ある程度リソースの稼働率が向上していたと考えられる．

実装 I5 では，(T4) ローカルメモリを使用したカーネルの統合によって，高速化が達成できている．これは，3.3.2 節で述べたとおり，処理全体でグローバルメモリへのアクセスが削減されたためである．また，実装 I5 に対して，ワークグループのサイズを変化させたときのカーネルの処理時間を表 3.7 に示す．

3.4.2　GPU 実装のボトルネック解析

3.4.1 節で処理時間を計測した実装 I1∼I5 に対して，それぞれの性能向上の要因と性能のボトルネックを解析する．解析では，NVIDIA が提供するプロファイラである Visual Profiler を用いる．同ソフトウェアでは，OpenCL で記述されたプログラムを詳細に解析することは困難であるため，3.4.1 節で評価した実装を CUDA に移植し，移植したプログラムを用いて解析を行う．使用した GPU は NVIDIA GeForce GTX 580 である．同 GPU の理論演算性能は 1,581.056 GFLOPS であり，理論メモリバンド幅は，グローバルメモリが 192.4 GB/s,



	I1	I2	I3	I4	I5
HD 5870	67.0	44.6	39.8	38.9	29.9
HD 6970	64.0	41.5	36.3	35.8	27.1
HD 7970	34.3	23.1	15.7	15.6	13.7
R9 290X	29.5	19.9	15.7	15.3	9.9
GTX 480	72.6	49.5	36.1	27.0	21.0
GTX 580	63.6	41.9	30.3	25.3	18.9
GTX 680	50.4	34.8	29.4	21.3	16.5
GTX 780	43.3	27.7	21.7	17.7	13.7

Implementation (applied in sequence)

図 3.15　各種の実装の工夫に対する処理時間

表 3.5　NVIDIA GPU における，ワークグループのサイズに対する各カーネルの処理時間 [ms]．各列はそれぞれ，（ワークグループの垂直方向の大きさ S_y）×（反復回数）として処理を行った場合の処理時間を示す．また，それぞれのカーネルについて最も短い処理時間を太字で示している．

GTX 480	1x15	2x8	3x5	4x4	5x3	8x2	15x1
探索ウィンドウ抽出	0.92	0.59	**0.50**	0.51	0.55	0.77	1.36
離散フーリエ変換	2.56	1.68	1.43	1.52	**1.41**	1.59	1.93
正規化相互パワースペクトルの計算	1.16	0.73	**0.60**	0.66	0.62	0.78	0.96

GTX 580	1x15	2x8	3x5	4x4	5x3	8x2	15x1
探索ウィンドウ抽出	0.79	0.51	**0.43**	0.44	0.48	0.67	1.11
離散フーリエ変換	2.30	1.50	1.27	1.35	**1.25**	1.40	1.58
正規化相互パワースペクトルの計算	1.03	0.65	**0.55**	0.62	0.58	0.71	0.77

GTX 680	1x15	2x8	3x5	4x4	5x3	8x2	15x1
探索ウィンドウ抽出	1.03	0.62	0.49	**0.45**	0.51	0.65	1.08
離散フーリエ変換	2.33	1.53	1.24	**1.20**	1.24	1.35	1.74
正規化相互パワースペクトルの計算	0.99	0.60	0.45	**0.41**	0.46	0.53	0.85

GTX 780	1x15	2x8	3x5	4x4	5x3	8x2	15x1
探索ウィンドウ抽出	0.64	0.43	0.37	0.37	**0.34**	0.44	0.73
離散フーリエ変換	1.44	1.01	0.86	0.85	**0.84**	0.91	1.09
正規化相互パワースペクトルの計算	0.62	0.38	0.30	**0.27**	0.29	0.33	0.46

表 3.6　AMD GPU における，ワークグループのサイズに対する各カーネルの処理時間 [ms]．各列はそれぞれ，（ワークグループの垂直方向の大きさ S_y）×（反復回数）として処理を行った場合の処理時間を示す．また，それぞれのカーネルについて最も短い処理時間を太字で示している．

HD 5870	1×15	2×8	3×5	4×4	5×3	8×2
探索ウィンドウ抽出	1.64	0.96	0.96	**0.78**	0.86	0.86
離散フーリエ変換	6.32	3.49	3.61	2.91	3.23	**2.84**
正規化相互パワースペクトルの計算	1.57	0.92	0.94	**0.77**	0.87	0.83

HD 6970	1×15	2×8	3×5	4×4	5×3	8×2
探索ウィンドウ抽出	1.39	0.77	0.92	**0.74**	0.83	**0.74**
離散フーリエ変換	5.36	2.95	3.43	**2.65**	2.99	2.79
正規化相互パワースペクトルの計算	1.30	0.76	0.87	**0.73**	0.84	0.81

HD 7970	1×15	2×8	3×5	4×4	5×3	8×2
探索ウィンドウ抽出	0.40	0.39	0.33	**0.32**	**0.32**	0.33
離散フーリエ変換	0.29	**0.21**	0.26	0.23	0.22	0.22
正規化相互パワースペクトルの計算	1.12	**0.65**	0.75	0.68	0.73	0.59

R9 290X	1×15	2×8	3×5	4×4	5×3	8×2
探索ウィンドウ抽出	0.25	0.23	**0.22**	**0.22**	**0.22**	**0.22**
離散フーリエ変換	0.55	0.38	0.41	**0.37**	0.40	0.41
正規化相互パワースペクトルの計算	0.21	**0.16**	0.18	0.17	0.17	0.17

表 3.7　ワークグループのサイズに対する統合カーネルの処理時間 [ms]．各列はそれぞれ，（ワークグループの垂直方向の大きさ S_y）×（反復回数）として処理を行った場合の処理時間を示す．また，それぞれの GPU において最も短い処理時間を太字で示す．

	1×15	2×8	3×5	4×4	5×3	8×2	15×1
HD 5870	9.14	5.06	5.30	**4.47**	5.18	5.04	N/A
HD 6970	6.90	4.34	4.95	**4.17**	4.74	5.02	N/A
HD 7970	1.53	**1.04**	1.34	1.17	1.32	1.34	N/A
R9 290X	0.94	**0.66**	0.83	0.73	0.85	0.85	N/A
GTX 480	3.88	3.19	**2.92**	3.51	3.07	3.90	4.70
GTX 580	3.82	2.95	**2.49**	2.97	2.59	3.33	4.07
GTX 680	3.80	2.64	2.18	**2.11**	2.24	3.39	4.95
GTX 780	2.41	1.79	1.53	**1.49**	1.55	2.21	2.76

ローカルメモリが 1,544 GB/s である [67]．また，STREAM ベンチマーク [89] で計測したメモリバンド幅は 153.702 GB/s である．

各カーネルにおけるカーネルのプロファイル結果を表 3.8 に示す．まず，実装 I1 におけるそれぞれのカーネルについて考察する．どのカーネルの実効性能も，理論演算性能や理論メモリバンド幅に比べて低いことがわかる．探索ウィンドウ抽出カーネルと離散フーリエ変換カーネルでは，計算律速となっている．探索ウィンドウ抽出カーネルにおける計算は，画像から抽出する画素の位置の計算と，ハニング窓の計算である．それぞれの計算は，それらが完了しない限り，グローバルメモリへのアクセスが開始できない．すなわち，抽出する画素の位置が判明しない限りグローバルメモリから画素値を読み出すことはできず，同様にハニング窓を作用させない限りはグローバルメモリへ画素値を書き出すことができない．このため，これらの計算によってカーネルが律速されていると考えられる．離散フーリエ変換カーネルでは，グローバルメモリから読みだした値をそれぞれの演算ユニット内にあるローカルメモリへ格納し，他

の演算ユニットとは通信を行うことなく処理を行う．このため，メモリバンド幅に比べて，計算が処理時間に対して支配的になると考えられる．正規化相互パワースペクトル計算，離散フーリエ逆変換，対応点座標の更新のそれぞれのカーネルでは，ワークグループのサイズが不適当であり，スループットが低下していることが予想される．

表 3.8　各カーネルのプロファイル結果

カーネル	処理時間 [ms]	律速	実効性能 [GFLOPS]	実効メモリバンド幅 [GB/s]
探索ウィンドウ抽出 (I1)	0.40	計算	226.51	50.74
探索ウィンドウ抽出 (I4)	0.22	メモリ	147.61	110.37
離散フーリエ変換 (I1)	2.89	計算	351.19	21.79
離散フーリエ変換 (I2)	2.95	計算	342.29	28.42
離散フーリエ変換 (I3)	1.28	計算	225.39	74.09
離散フーリエ変換 (I4)	0.97	計算・メモリ	278.90	93.50
正規化相互パワースペクトル計算 (I1)	0.97	命令・メモリレイテンシ	104.26	95.78
正規化相互パワースペクトル計算 (I2)	1.06	命令・メモリレイテンシ	115.12	32.63
正規化相互パワースペクトル計算 (I4)	0.59	計算	221.37	74.76
離散フーリエ逆変換 (I1)	0.17	命令・メモリレイテンシ	131.08	25.86
対応点座標の更新 (I1)	0.15	命令・メモリレイテンシ	0.07	11.05
統合カーネル (I5)	1.03	ローカルメモリ	382.71	3.40

次に，実装 I2 では，同時 DFT を用いることによって，離散フーリエ変換カーネルおよび正規化相互パワースペクトル計算カーネルが変更される．離散フーリエ変換カーネルでは，処理時間がわずかに増加している．I2 における離散フーリエ変換カーネルでは，単一のカーネルで読み出す探索ウィンドウが 1 つから 2 つに増えているが，カーネルが計算律速であったため処理時間には影響しなかったと考えられる．また，正規化相互パワースペクトル計算カーネルでは，2 本の 1 次元信号を同時に離散フーリエ変換した結果から，それぞれの信号の離散フーリエ変換を計算する処理が追加されたことより，処理時間がわずかに増加している．

実装 I3 では，回転因子のテーブル実装によって，離散フーリエ変換カーネルの処理時間が短縮されている．実効メモリバンド幅が増加した一方で，実効性能は減少している．これは，テーブル実装によって回転因子の計算が行われなくなったためであると考えられる．

実装 I4 では，探索ウィンドウ抽出，離散フーリエ変換，正規化相互パワースペクトル計算の 3 つのカーネルにおいて，スループットを向上するためにワークグループのサイズが調整される．ワークグループのサイズを調整したことにより，演算ユニットにおいて同時にスケジューリングされるワークグループの数が 3 から 8 に増加していることを確認している．その結果，探索ウィンドウ抽出カーネルでは，メモリ律速となり，実効メモリバンド幅が STREAM ベンチマークにおける計測値の約 72% となっている．すなわち，メモリバンド幅のほとんどを活用できているといえる．離散フーリエ変換カーネルでは，実装 I3 まで計算律速であったが，プロファイラによれば，計算とメモリの両方に律速され，使用率はどちらも約 70% 程度である．実効メモリバンド幅は STREAM ベンチマークの計測値の 70% に近い値であるが，実効性能は理論性能と大きく異なる．これは，プロファイラが示す使用率が，浮動小数点の演算だけでなく，整数の演算や，メモリの読み書き，分岐制御などの命令を含めて計算されているためであると考えられる．正規化相互パワースペクトル計算カーネルでは，ワークグループのサイズ調整によってスループットが向上し，実効性能と実効メモリバンド幅の両方が向上している．計算律速となっているのは，比較的計算コストの高い平方根の演算を行うためであると予想される．

実装 I5 では，5 つのカーネルを統合する．プロファイラによると，統合されたカーネルは，ローカルメモリの使用率が高く，ローカルメモリ律速である．これは，5 つのカーネルを統合

するにあたって，処理の中間値をローカルメモリを用いて受け渡すように実装したためである．ローカルメモリの実効メモリバンド幅は 1,130.50 GB/s であり，これは理論メモリバンド幅の約 73% に相当する．すなわち，ローカルメモリのメモリバンド幅のほとんどを活用できているといえる．

以上より，本論文で提案したアルゴリズムの並列化および実装の工夫により，それぞれのカーネルの性能が向上し，最終的にはローカルメモリ律速のカーネルとなる．このときの実効メモリバンド幅は理論値の約 73% に相当し，GPU のローカルメモリが持つメモリバンド幅をほとんど活用できているといえる．

3.4.3　CPU 実装に対する性能評価

POC に基づく画像対応付けアルゴリズムを GPU と CPU のそれぞれにおいて実装し，処理時間・電力遅延積を比較することで，CPU 実装に対する GPU 実装の有効性を確認する．

実験環境には表 3.3 に示す通りである．CPU を用いた実装では，シングルスレッドによる処理と 4 コア・8 スレッドによる処理をそれぞれ実装する．フーリエ変換については，高速フーリエ変換を実装したライブラリである FFTW [90] を用いる．GPU を用いた実装では，3.4.1 節において最も高速であった実装 I5 を用いる．処理時間の計測区間と計測手法については，3.4.1 節で述べたものと同様である．対応付けのパラメータおよび入力画像は 3.4.1 節で用いたものと同様である．ただし，基準点の数を 2,000 点, 5,000 点, 10,000 点，50,000 点と変化させて計測を行う．

電力遅延積は，プロセッサの処理中の消費電力（有効電力）と処理時間との積で表され，プロセッサが処理中に消費する総電力を示す．すなわち，電力遅延積が低いことは，消費電力の観点から効率が良いことを示している．消費電力については，図 3.16 に示す電力計（HIOKI AC/DC パワーハイテスタ 3334 [91] を用いてシステム全体の消費電力を計測する．同電力計が測定できる最短の時間は 200 ms であり，図 3.15 より GPU における対応付けの処理時間に比べて長い．そのため，単に対応付けの処理時におけるシステムの消費電力を測定するだけでは，GPU が処理を終えて CPU のみが動作している時間が長く，GPU が動作している間の消費電力を正確に測ることは困難であることが考えられる．そこで，なるべく計測時間と GPU

図 3.16　消費電力の計測に用いる電力計（HIOKI AC/DC パワーハイテスタ 3334）

の動作時間を重複させるため，以下のようにして測定を行う．まず，50,000 点の基準点に対する対応付け処理を繰り返し実行し，電力計を用いて 200 ms 経過するごとに消費電力を計測する．このようにして得られた消費電力の値のうち，最大値を消費電力とする．

GPU および CPU における画像対応付けの処理時間を表 3.9 に示す．表 3.9 より，基準点の数（2,000 点, 5,000 点, 10,000 点，50,000 点）に関わらず，いずれの GPU においても，CPU のシングルスレッドによる処理，およびマルチスレッドによる処理に比べ，対応付けの処理時間が短いことがわかる．

表 3.9　CPU 実装と GPU 実装の処理時間 [ms]

基準点数	2,000	5,000	10,000	50,000
Core i7-3770K (1 thr.)	81.8	195.9	393.7	1,919.0
Core i7-3770K (8 thr.)	19.6	43.0	195.9	389.8
Radeon HD 5870	11.7	18.1	29.9	119.7
Radeon HD 6970	9.0	16.4	27.1	111.9
Radeon HD 7970	7.3	11.3	13.7	32.5
Radeon R9 290X	7.1	8.0	9.9	23.5
GeForce GTX 480	9.3	13.4	21.0	80.3
GeForce GTX 580	8.8	12.5	18.9	69.8
GeForce GTX 680	8.1	11.1	16.5	55.5
GeForce GTX 780	7.4	9.6	13.7	44.2

以下では，30 fps のフレームレートで撮影可能なステレオカメラを用いて 3 次元計測を行うと仮定して，それぞれのプロセッサにおける対応付けの処理時間について議論する．30 fps はカメラの撮影における一般的なフレームレートのうちのひとつであり，画像の撮影間隔は 1/30 秒，すなわち 33.3 ms である．また，実際には，ステレオカメラを用いた 3 次元計測において，画像対応付け以外にもいくつかの処理を行う必要があるが，処理時間のほとんどは画像対応付けが占めるため，それ以外の処理に要する時間については以下では考慮しない．

表 3.9 より，基準点が 2,000 点の場合，CPU シングルスレッドにおける処理時間は 81.8 ms であり，画像の撮影間隔 (33.3 ms) を上回っている．つまり，CPU シングルスレッドを用いた対応付けでは，2,000 点の 3 次元計測をリアルタイム（ここでは，カメラのフレームレートである 30 fps を指す）で行うことができない．一方で，CPU マルチスレッド，および GPU における処理時間は 33.3 ms よりも短く，画像の撮影間隔以内に対応付けを完了することができ，計測をリアルタイムに行うことが可能である．

基準点が 5,000 点に増えた場合，CPU のマルチスレッドにおける処理時間は 43.0 ms となり，画像の撮影間隔を超えてしまう．すなわち，CPU を用いた処理では，5,000 点の 3 次元計

測をリアルタイムで行うことができない．一方で，表 3.9 に示すいずれの GPU においても，処理時間は 33.3 ms を下回っている．これは，基準点がさらに増え，10,000 点となった場合でも同様である．また，基準点が 50,000 点でも，一部の GPU では画像の撮影間隔を下回る時間での対応付けが可能であることもわかる．

表 3.9 と以上の議論より，GPU を用いた処理によって，CPU のシングルスレッド処理に比べ 13〜40 倍，CPU のマルチスレッド処理に比べても 7〜20 倍の高速化ができるといえる（基準点が 10,000 点の場合）．また，高速化により，CPU では困難であった密度（基準点数 5,000 点）でのリアルタイムな 3 次元計測が可能になるともいえる．これにより，3 次元計測の応用によっては，一定時間内に計測できる点数が増えたことにより，後段の処理の性能を向上できる可能性がある．また，計測時間が短縮されたことにより，後段の処理を新たに追加することも可能であろう．このような，3 次元計測システムが GPU 実装を用いた高速化によって受ける恩恵については，4 章で詳細に述べる．

また，CPU と GPU のそれぞれにおける処理に関する消費電力，および電力遅延積を表 3.10 に示す．なお，表 3.10 に示す電力遅延積の計算においては，表 3.9 における基準点数が 50,000 点の場合の処理時間を用いている．表 3.10 より，GPU における処理は，CPU のシングルスレッド処理およびマルチスレッド処理に比べ，消費電力が高い一方で電力遅延積は低いことがわかる．すなわち，GPU を用いた実装では，CPU を用いた実装に比べ，表 3.9 に示すように高速な処理が可能になるだけでなく，電力効率の観点からも優れているといえる．

3.5　むすび

本章では，POC に基づく画像対応付けアルゴリズムの GPU 実装について述べた．まず，今日の計算機における GPU の役割について述べ，一般的な GPU のアーキテクチャについて概説した．次に，アルゴリズムを GPU へ実装するに際して，さまざまな並列アーキテクチャを取り扱うためのフレームワークである OpenCL について概説した．さらに，POC に基づく画像対応付けアルゴリズムを GPU に実装するに際して，並列化の手法と最適化の手法について述べた．最後に，処理速度に関する評価実験を行い，POC に基づく画像対応付けアルゴリ

表 3.10　CPU 実装と GPU 実装の消費電力および電力遅延積

	消費電力 [W]	電力遅延積 [W×s]
Core i7-3770K (1 thr.)	56.2	107.9
Core i7-3770K (8 thr.)	98.1	38.2
Radeon HD 5870	152.1	18.2
Radeon HD 6970	171.1	19.1
Radeon HD 7970	160.9	5.2
Radeon R9 290X	175.7	4.1
GeForce GTX 480	258.3	20.7
GeForce GTX 580	207.0	14.5
GeForce GTX 680	185.2	10.3
GeForce GTX 780	174.6	7.7

ズムの GPU 実装について，最適化によって 2.2 ～ 3.5 倍の高速化を達成できることを示した．また，GPU を用いた実装では，CPU シングルコアを用いた処理の 13 ～ 40 倍，CPU4 コア・8 スレッドを用いた処理に対しても 7 ～ 20 倍高速であることを示した．

第 4 章

ステレオビジョンに基づく 3 次元計測への応用

4.1 まえがき

3 次元計測技術は，物体の形状や空間の構造を計測する技術である．近年，3 次元計測技術に対するニーズは，産業・医療・芸術・教育などのさまざまな分野において高まりつつある．産業の分野では，エレクトロニクス製品や自動車などの製品検査に用いられ，製品の性能および品質の向上を支えている [92]．医療の分野では，外科や歯科において，骨格や歯の形状を解析・診断する際に用いられる機会が増えつつある [93]．芸術・教育の分野では，文化遺産の保護を目的として，美術品や遺産などの 3 次元情報を記録する際に用いられている [94]．

一般に 3 次元計測技術は，能動型計測法と受動型計測法の 2 つに分類することができる．能動型計測法は，物体にレーザや構造光を投影し，その挙動を観察することで物体表面の 3 次元形状を取得する [93]．一般に，能動型計測法は精度に優れているが，レーザなどを物体に照射・投影するために計測システムが複雑になる．また，複数の計測システムによる相互干渉が起こる，計測対象および使用環境が制限されるという問題が存在する．一方，受動型計測法の 1 つとして，複数のカメラ画像を用いて 3 次元計測を行うステレオビジョンが知られている．ステレオビジョンは，異なる視点から同一の対象を観測し，それぞれの画像上への投影位置の違いから対象の 3 次元情報を計測する手法である．ステレオビジョンは，カメラと計算機の他

に特別な装置を必要としないため，安価で簡便なシステムで計測が可能である．また，計測の過程で取得した画像情報を用いることで広範な応用が期待できる．これらの利点から，近年ではステレオビジョンに基づく 3 次元計測に関する研究が盛んに行われている．

本章では，本論文で提案する 1 次元 POC に基づく画像対応付けアルゴリズムの GPU 実装を，ステレオビジョンに基づく 3 次元計測に応用し，その有効性を実証する．具体的には，(a) リアルタイム 3 次元計測システムと (b) 汎用デジタルカメラを用いた 2 視点からの 3 次元計測システムを構築する．これらのシステムの評価を通して，提案手法によって高密度・高精度・高速な受動型 3 次元計測が実現できることを示す．

4.2　リアルタイム 3 次元計測システム

2 章では，ステレオビジョンに基づく 3 次元計測において，POC に基づく画像対応付けアルゴリズムを用いることで高密度かつ高精度な計測を行うことが可能であることを示した．これまでに，POC に基づく画像対応付けアルゴリズムを用いたステレオビジョンシステムによって，金属部品に掘削されたスルーホールの検査 [95] や，車載ステレオカメラを用いた障害物の検出・測距 [96], [97] が提案されている．このような応用では，計測における画像対応付けアルゴリズムの精度だけでなく，対応付けの速度も重要である．特に，対応付けの速度に関しては，計測に必要な点数をカメラのフレームレート内で対応付けるリアルタイム性が求められる．POC に基づく画像対応付けをリアルタイムに行うことができれば，FA (Factory Automation) における部品の位置合わせや製品の品質検査のほか，監視カメラや車載カメラを用いた周辺監視など，幅広い応用展開が期待される．また，バイオメトリクスの分野においても，個人認証における 3 次元情報が注目されつつある．例えば，人物の顔の 3 次元計測によって，3 次元形状を用いた個人認証 [98], [99] や，2 次元の顔画像と組み合わせた性能向上 [100] が報告されている．人物の顔の 3 次元形状をリアルタイムで計測することができれば，個人認証の分野においても幅広い応用展開が可能である．

本節では，ステレオカメラと GPU を搭載した計算機を用いてリアルタイムに 3 次元計測を行うシステムを構築する [29]–[32]．同システムの計測対象として，人物の顔に着目する．人

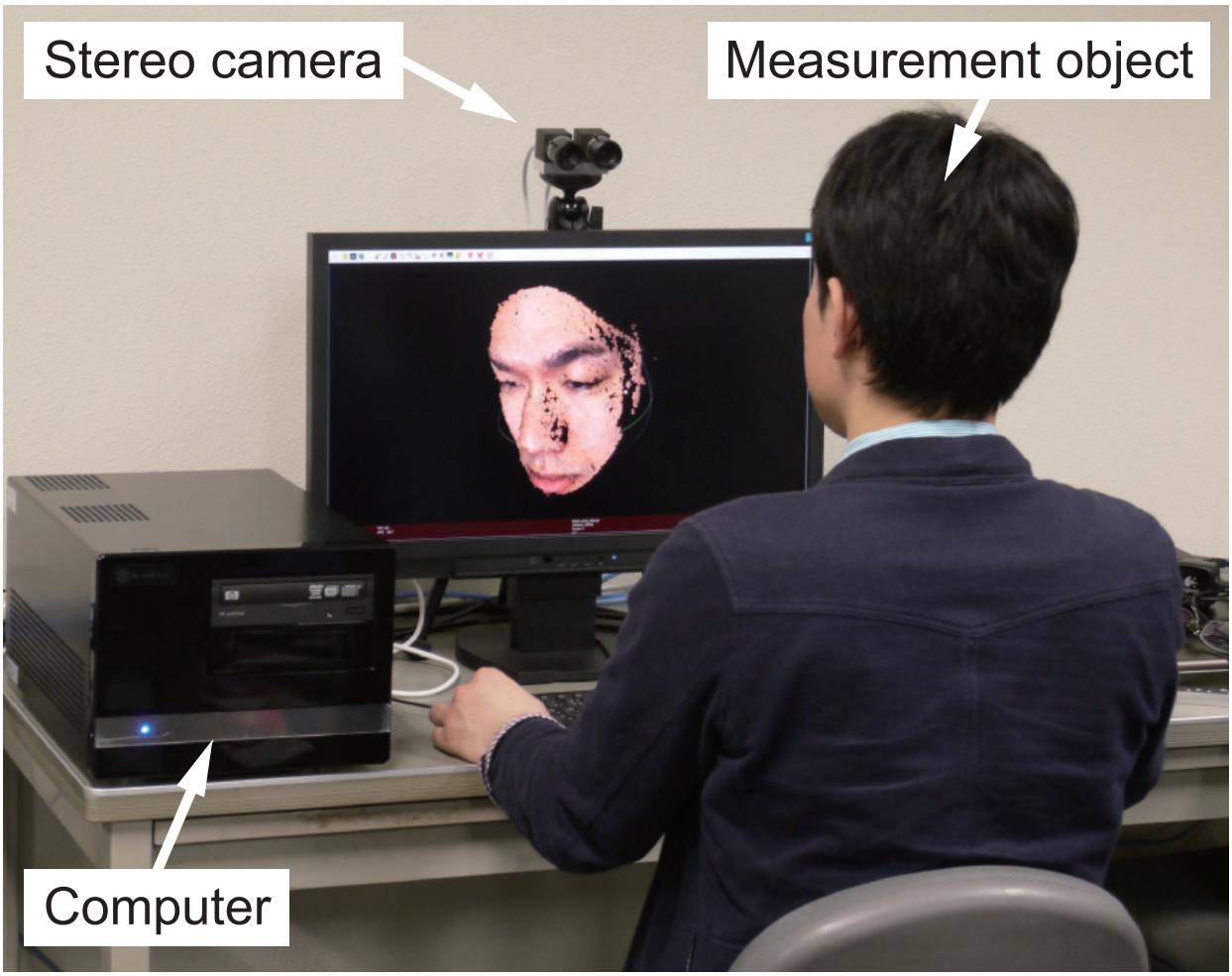


図 4.1　リアルタイム 3 次元計測システム

物の顔の形状は，テクスチャが豊富であり，表情の変化によって瞬間的に変化するため，高精度かつ高速な 3 次元計測に関する計測対象として適する．システムの評価を通して，提案手法を用いることで，高精度な 3 次元計測が高密度かつ高速に行えることを示す．

4.2.1　システムの概要

本システムは，図 4.1 に示すように，GPU を搭載した計算機とステレオカメラから構成される．その諸元を表 4.1 に示す．なお，ステレオカメラはあらかじめキャリブレーション [101] しており，カメラのパラメータは既知とする．

表 4.1　リアルタイム 3 次元計測システム諸元

ステレオカメラ	Point Grey Research Inc.,
（左カメラ）	Flea2 (FL2G-13S2C-C), color
（右カメラ）	Flea2 (FL2G-13S2M-C), monochrome
基線長	40 mm
レンズ	μTRON FV1022, 焦点距離 10 mm
撮影フレームレート	15 fps
撮影ソフトウェア	Point Grey FlyCapture SDK [102]
CPU	Intel Core i7-3990K (3.2 GHz)
メモリ	DDR3-1333 SDRAM 4GB×1
GPU	NVIDIA GeForce GTX 580
OS	Windows 7 (32 bit)
コンパイラ	Visual C++ 9.0 Compiler (CPU code)
	CUDA Toolkit 4.2 (GPU code)
画像サイズ	$1,280 \times 960$ ピクセル
探索ウィンドウ	32 ピクセル × 15 ライン
探索階層数	4 階層

4.2.2　3 次元計測フロー

本システムの処理フローを図 4.2 に示す．詳細な計測の手順を以下に述べる．

Step 1

ステレオカメラで画像を撮影する．使用するカメラ，レンズ，ステレオカメラの基線長，取得する画像サイズは表 4.1 に示すとおりである．画像の撮影では，PointGrey 社が提供するカメラのドライバおよび SDK を用いる [102]．撮影では，左カメラではカラー画像，右カメラではモノクロ画像を取得する．これは，計測対象の 3 次元形状だけでなく，色情報も取得する

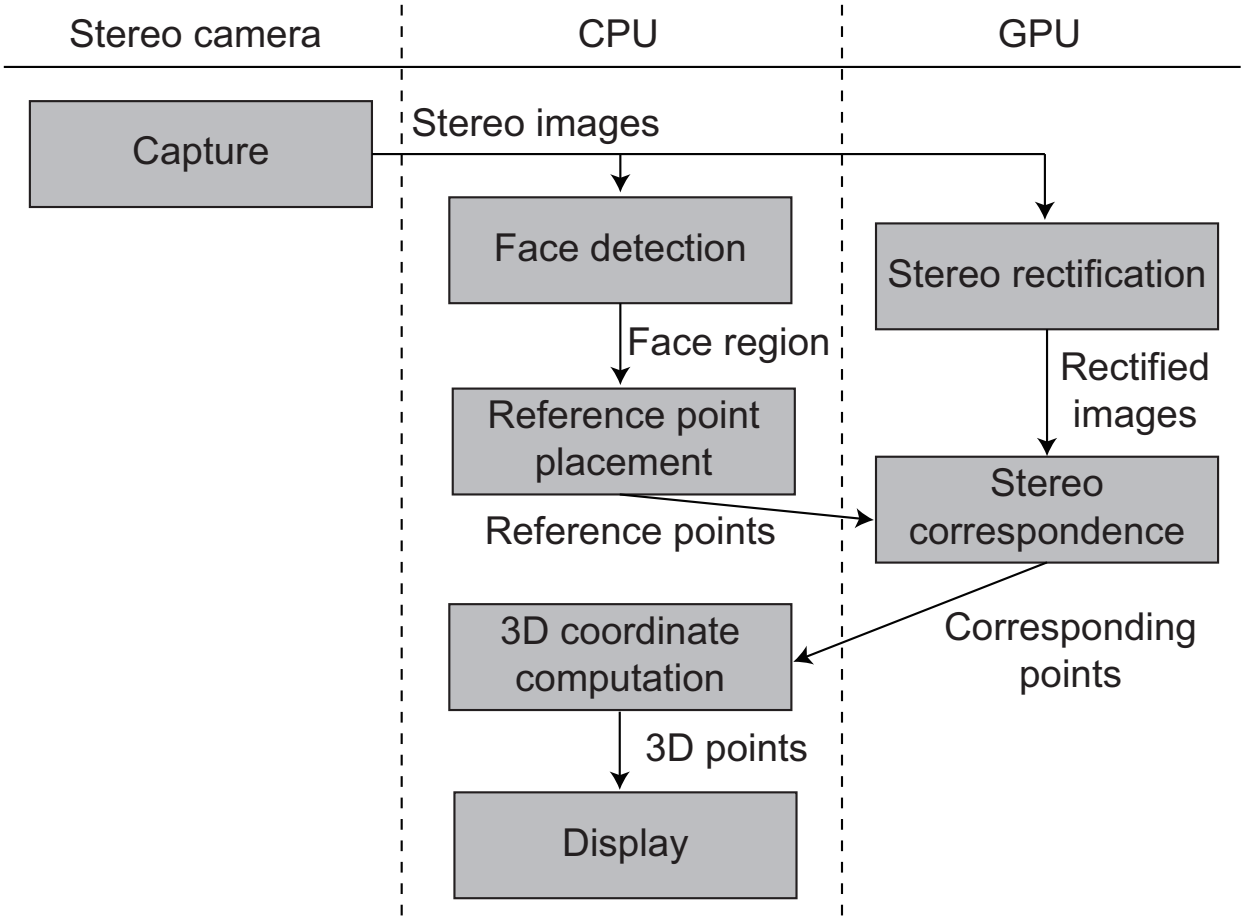


図 4.2　リアルタイム 3 次元計測システム 処理フロー

ためである．色情報が不要な場合は，左カメラを右カメラと同様にモノクロカメラとし，モノクロ画像を取得すればよい．

Step 2

左カメラで撮影した画像をグレースケールに変換し，変換後の画像から顔領域を検出する．顔領域の検出には，Haar-like 特徴量と AdaBoost に基づく物体認識フレームワークを用いる [103], [104]．この物体認識フレームワークの実装には，OpenCV で提供されているライブラリを用いる [105]．同ライブラリでは顔領域は矩形で表現されており，複数の顔領域が検出された場合は，最も面積の大きい領域のみを出力する．顔領域が検出された場合，次のステップに進む．顔領域が検出されなかった場合は，Step 1 に戻る．

Step 3

Step 2 で検出された顔領域に基準点を配置する．基準点は，格子状に 5 ピクセル間隔で配置する．

Step 4

撮影された画像を GPU を用いて平行化する．平行化では，ステレオ画像に射影変換行列を用いた画像変形を行う．平行化に用いる射影変換行列は，事前のカメラキャリブレーションによって取得した内部パラメータおよび外部パラメータから計算することができる．平行化では，平行化後の画像について，画素単位に処理が独立であるため，画素単位での並列化を行う．具体的には，$w \times h$ ピクセルの画像に対して，以下のようにワークアイテムを配列して処理を行う．

- NDRange: (w, h)
- ワークグループ：$(16, 16)$

Step 5

Step 3 で配置した基準点について，GPU を用いて対応付けを行い，対応点の座標を求める．対応付けのパラメータは表 4.1 に示すとおりである．

Step 6

Step 5 において求めた対応点の座標と基準点の座標より，3 次元空間中の座標を求める．

Step 7

Step 6 において座標を求めた 3 次元点群を表示する．点群の表示には PCL (Point Cloud Library) [106] を用いる．

以上 **Step 1∼6** の処理を，ステレオカメラで画像を取得するたびに繰り返す．

4.2.3　性能評価

本システムを用いて，人物の顔を計測した結果を図 4.3 に示す．撮影された左カメラ画像（図 4.3 (a)）および右カメラ画像（図 4.3 (b)）より，人物の顔の形状が計測できていることが確認できる（図 4.3 (c)）．図 4.3 (d) は計測された 3 次元点群をもとに生成された 3 次元メッシュであり，法線の方向に応じて彩色されている．同図より，しわや頬の隆起といった，人物

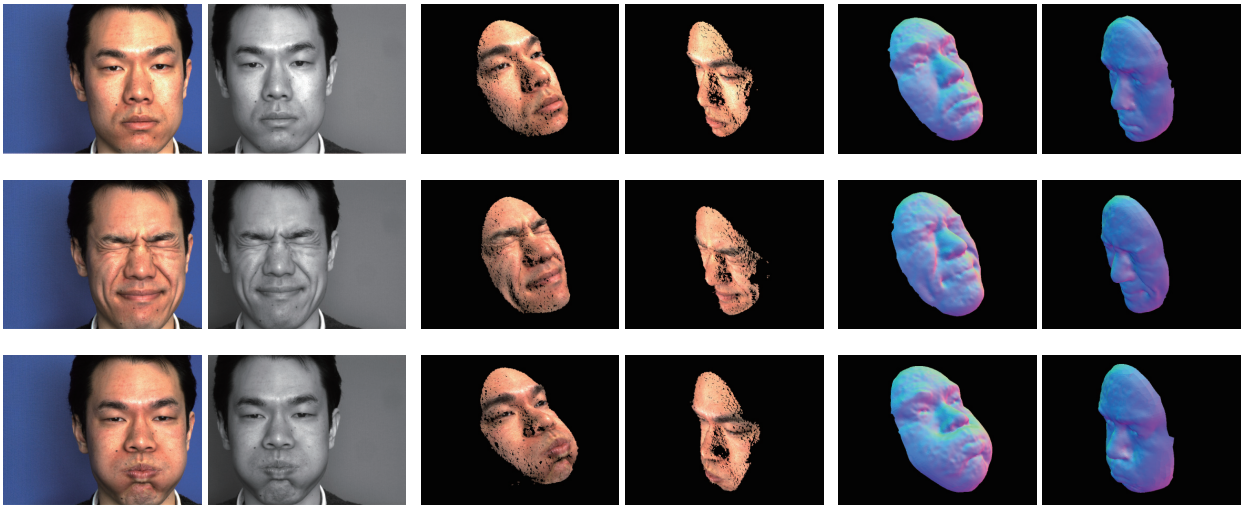
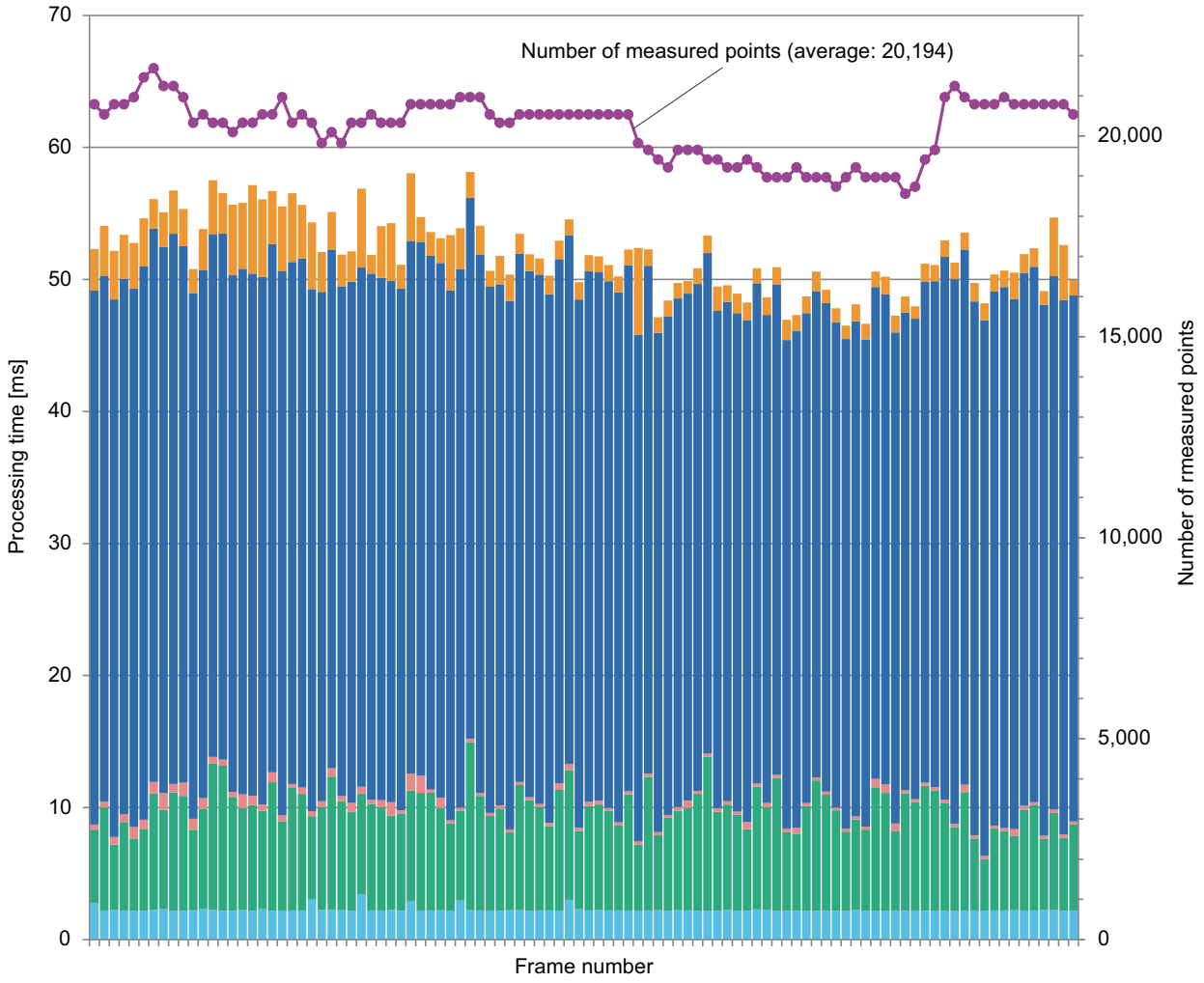


図 4.3　計測された人物の顔：(a) 左カメラで撮影したカラー画像，(b) 右カメラで撮影したモノクロ画像，(c) 計測された 3 次元点群（点群の色情報は左カメラ画像より取得），(d) 3 次元点群より生成した 3 次元メッシュ（生成には MeshLab [107] を使用）

の顔の詳細な形状が計測できていることがわかる．

システムの動作時に連続して撮影された 100 枚（フレーム）のステレオ画像に対し，計測された 3 次元点数，および計測に要した時間の内訳を図 4.4 に示す．使用したカメラのフレームレートは 15 fps であり，言い換えれば 66.6 ms ごとに画像が撮影される．図 4.4 より，どの時刻に撮影された画像に対しても，66.6 ms 以下の時間，すなわちカメラが次の画像を撮影するまでの間に 3 次元計測を完了していることがわかる．計測された 3 次元点の数は時刻ごとに異なるが，100 フレームの平均はおよそ 20,000 点である．すなわち，提案システムは，およそ 20,000 点の 3 次元点をカメラのフレームレート (15 fps) を上回る速度で計測できている．これを裏付けるものとして，システムの動作時に人物の表情が変化しても，追従して計測されることを確認している．

図 4.4 において処理時間の内訳を見ると，ステレオ画像対応付けの時間が最も多く，全体の約 75% を占めていることがわかる．第 3 章での実験結果（表 3.9）より，対応付けを CPU で



	Average [ms]
3D coordinate computation	2.31
Stereo correspondence	39.36
Reference point placement	0.42
Face detection	7.73
Stereo rectification	2.26
Total	52.08

図 4.4　人物の顔の 3 次元計測の処理時間内訳（100 フレーム）

行った場合，3 次元計測処理がカメラのフレームレート（15 fps または 66.6 ms）に間に合わないことが予想される．これを回避するためには，顔領域に配置する基準点の間隔を広くし，対応付ける基準点数を削減することが考えられる．その場合，カメラのフレームレートに間に合って計測できる点はおよそ 2,000 点程度であろう．

以上の結果より，ステレオ画像の対応付けを GPU に実装することで，3 次元計測において処理時間のボトルネックである対応付けを高速化し，高密度かつ高精度な 3 次元計測がリアルタイムで実現できることを示した．

以下では，提案システムに関する今後の展望について述べる．人物の顔の 3 次元計測に関する応用の一例として，文献 [99] では，人物の顔に対して約 4,000 点の 3 次元点を計測し，それらの点群を用いて個人認証を行なっている．提案システムでは，約 20,000 点の計測を 15 fps で行うことが可能であることを既に述べた．提案システムを文献 [99] の認証アルゴリズムと組み合わせることで発展が期待できる．例えば，より多くの点を用いることで認証の性能が向上したり，高速に認証が可能になることで画像中の複数の人物について認証が可能になったり，表情の変化を用いた認証が可能になるであろう．

4.3　デジタルカメラの移動撮影に基づく 3 次元計測システム

近年，文化財や都市のデジタルアーカイブを目的として，カメラの移動撮影によって取得された画像間を対応付け，3 次元計測を行う試みが盛んである [21]–[23]．このようなカメラの移動撮影に基づく計測は，(i) 計測システムがカメラと計算機のみから構成されるため，安価な計測が可能であること，(ii) カメラでの撮影という簡便な作業だけで，計測に必要な情報が取得できること，(iii) 大掛かりな装置が必要である能動型計測法に比べ，計測環境が限定されにくいことから，高い利便性を持つ 3 次元計測手法として注目されている．

カメラの移動撮影に基づく 3 次元計測では，これまでに様々な手法が提案されている [50] が，代表的な手法として Structure from Motion (SfM) [49] が知られている．SfM では，単眼カメラで移動撮影を行い，得られた多視点画像間の対応付けを行うことで，カメラパラメータの推定と物体の 3 次元計測を行う．近年では，Scale-Invariant Feature Transform

(SIFT) [24] をはじめとして，画像間の幾何学的変形や輝度変化に対してロバストな特徴量が提案され，画像間を安定かつ高精度に対応付けることが可能になった．さらに，SIFT に基づく画像対応付けアルゴリズムから得られたカメラパラメータに対してバンドルアジャストメントを適用することで，カメラパラメータを安定かつ高精度に推定できることが示された [108]．現在では，SfM による 3 次元計測アルゴリズムを容易に利用できる環境として，Bundler [109] や VisualSfM [110] といったソフトウェアや，Autodesk 123D Catch [111] や ARC 3D [112] といったクラウドサービスがあり，今後も SfM に関する研究や応用が発展することが予想される．

しかしながら，特徴ベースの対応付けに基づく SfM では，計測される 3 次元点が画像上の特徴点に限定されてしまうので，3 次元計測点が少ないという問題がある．そのため，物体の大まかな形状を計測することは可能だが，詳細な形状を計測することは困難である．入力画像の枚数を増やすことで，3 次元計測点を増やすことができるが，画像枚数の増加に伴って処理時間が増大してしまう．また，入力画像の枚数を増やしたとしても，テクスチャ情報が乏しい領域は，特徴点が検出されないために計測することができない．

これに対し，本節では，特徴ベースの対応付けに基づく SfM と領域ベースの密な対応付けを組合せることで，単眼カメラを用いた 2 回の撮影から高精度かつ密な 3 次元計測を行う手法を提案する．これまでに，単眼カメラを用いた移動撮影から密な 3 次元計測を行う手法として，SfM と多視点ステレオアルゴリズムを組合せた手法が提案されている [23], [50], [113]．これらのアルゴリズムでは，都市などを対象とした大規模な 3 次元計測を主眼に置いており，大量の画像と計算資源，および長時間の計算機処理を必要とする．これに対し，提案手法は，少ない撮影回数と一般的な計算機による実用的な時間内での計測を目的としており，部品や建築物の外観検査や，個人向けの 3 次元計測など，比較的小規模な応用において有用であると考えられる．また，これまでに，多視点画像間における特徴ベースの対応付け精度や，ステレオ画像間における領域ベースの対応付け精度については評価が行われている [114], [115] ものの，これらを組合せた 3 次元計測という観点からの精度評価は不充分であった．これに対して本節では，特徴ベースの対応付けと領域ベースの対応付けとの組合せについて総合的な評価を行うとともに，レーザを用いる 3 次元計測装置やキャリブレーション済みのステレオカメラに対し

て，提案手法の 3 次元計測精度を評価する．さらに，汎用のディジタルカメラと計算機を用いた簡便な 3 次元計測システムを構築することで，提案手法の有用性を示す [33]–[39]．さらに，POC に基づく画像対応付けアルゴリズムを GPU を用いて実装することで，カメラの移動撮影による 3 次元計測においても，画像対応付けアルゴリズムの GPU 実装が有用であることを示す．

4.3.1　システムの概要

提案システムは，図 4.5 (a) に示すように，汎用のディジタルカメラと計算機から構成される．カメラから計算機への画像の転送は，Eye-Fi カード [116] を用いる．Eye-Fi カードは Wi-Fi 通信機能を搭載したメモリーカードであり，カメラで撮影した画像を計算機内の任意のディレクトリに転送する．計算機では，転送先のディレクトリを監視し，画像が 2 枚転送された時点で 4.3.2 節に述べたアルゴリズムを用いて 3 次元計測を行う．計測結果は，Point Cloud Library (PCL) [106] を用いて表示され，ユーザは任意の視点から計測結果を確認したり，点群データとして保存したりすることができる．また，図 4.5 (b) および (c) に示すように，タブレット端末やスマートフォンといった，カメラと汎用プロセッサの両方を搭載した機器を用いることで，より小型で携行性の高いシステムの構築が可能である．

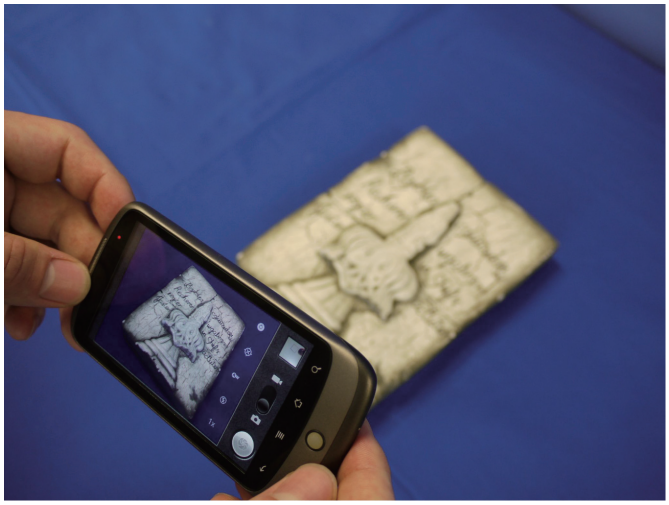
4.3.2　3 次元計測フロー

一般に，ステレオ画像からの 3 次元計測では，ステレオ画像間の対応関係とステレオカメラのカメラパラメータ（3 次元座標とカメラ画像の 2 次元座標の関係を表す内部パラメータ，およびステレオカメラの位置関係を表す外部パラメータ）から，三角測量の原理に基づいて 3 次元座標を計算する [48]．しかし，カメラの移動撮影によりステレオ画像を取得する場合は，ステレオカメラのカメラパラメータが未知であるため，3 次元座標の計算ができないという問題がある．

そこで，提案手法では，特徴ベースの対応付けに基づく SfM により，カメラパラメータを推定する．また，推定されたカメラパラメータを用いてステレオ画像を平行化することで，領域

(a)

(b)



(c)

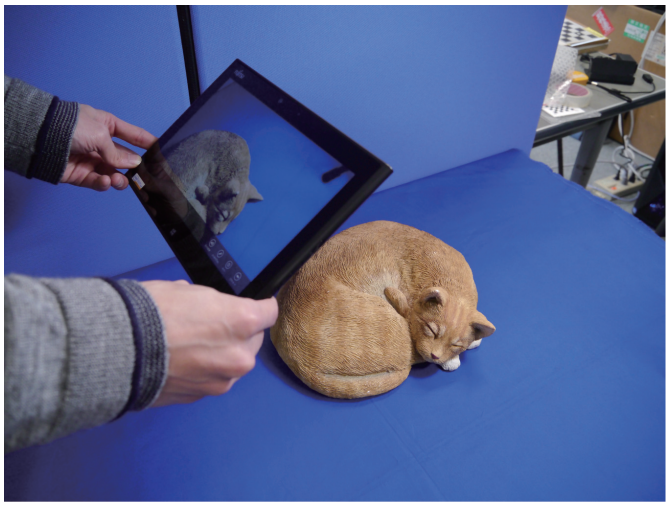


図 4.5　計測システムの構成例：(a) 汎用ディジタルカメラと計算機，(b) スマートフォン，(c) タブレット端末

ベースの密な対応付けを容易にし，物体全体の詳細な形状を計測する．具体的には，図 4.6 に示すように，カメラの移動撮影によって得られた 2 枚の画像をステレオ画像ペアとし，(I) カメラパラメータ推定，(II) ステレオ平行化，(III) 3 次元形状計測の三つの処理によって物体の 3 次元形状を計測する．以下では，(I)∼(III) の処理について詳細に述べる．

(I)　カメラパラメータ推定

カメラの移動撮影によって取得された 2 枚の画像（図 4.6 (a)）について，1 枚目の画像 I_1 を撮影したときのカメラの内部パラメータ $\boldsymbol{A}_1$，2 枚目の画像 I_2 を撮影したときのカメラの内部パラメータ $\boldsymbol{A}_2$，それらカメラの位置関係（回転行列 $\boldsymbol{R}_{1\to 2}$ および並進ベクトル $\boldsymbol{t}_{1\to 2}$）を推定する．

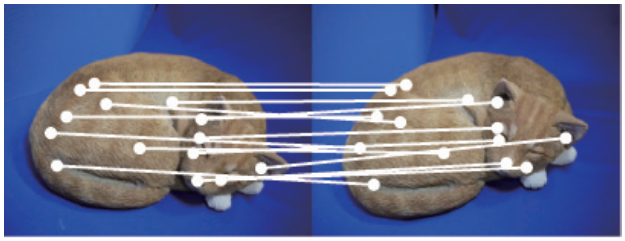
まず，カメラの内部パラメータ（$\boldsymbol{A}_1$, $\boldsymbol{A}_2$）を算出する．カメラの内部パラメータ $\boldsymbol{A}$ は，カメラの焦点距離 f_l [mm], 画像の解像度 $w \times h$ [ピクセル], カメラの撮像素子の幅 D [mm] より，次式で算出する．

$$\boldsymbol{A} = \begin{pmatrix} f_l \frac{w}{D} & 0 & \frac{w}{2} \\ 0 & f_l \frac{w}{D} & \frac{h}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

ここで，カメラの焦点距離 f_l と画像の解像度 (w, h) は，画像の Exif (Exchangeable image file format) 情報 [117] から取得する．また，カメラの撮像素子の幅 D は，使用するカメラに依存し，既知とする．

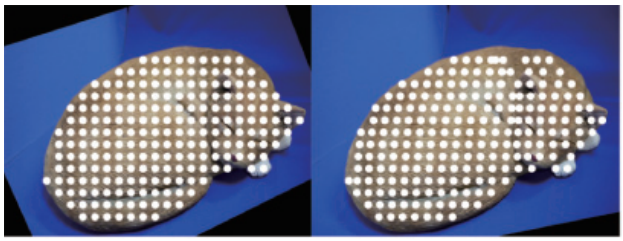
次に，カメラの外部パラメータ（回転行列 $\boldsymbol{R}_{1\to 2}$, 並進ベクトル $\boldsymbol{t}_{1\to 2}$）を算出するため，図 4.6 (b) に示すように，特徴ベースの対応付けを用いてステレオ画像間を対応付ける．対応付けによって得られた対応点ペアとカメラの内部パラメータ（$\boldsymbol{A}_1$ および $\boldsymbol{A}_2$）より, RANSAC (RANdom SAmple Consensus) [118] と 5 点アルゴリズム [119] を用いて $\boldsymbol{R}_{1\to 2}$ および $\boldsymbol{t}_{1\to 2}$ を算出する．カメラの移動撮影によって取得されるステレオ画像では，カメラの移動や焦点距離の変化により様々な画像変形が生じる．そのため，画像の幾何学的変形にロバストな特徴ベースの手法を用いてステレオ画像間を対応付ける．特徴ベースの対応付けでは，画像中のエッジやコーナなどの特徴点について特徴量を記述し，特徴量を比較することで特徴点同士の対応付けを行う．これまでにさまざまな特徴量が提案されているが，中でも SIFT は，画像

(a)

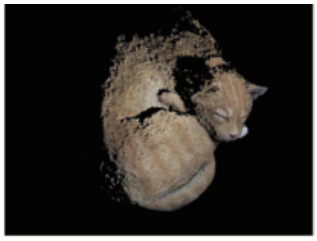


(b)

(c)



(d)



(e)

図 4.6　提案手法による 3 次元計測の処理過程：(a) ステレオ画像ペアの撮影，(b) 特徴ベースの対応付け，(c) ステレオ平行化，(d) 領域ベースの対応付け，(e) 3 次元座標の計算

間の幾何学的変形や輝度変化に対して，ロバストな特徴量としてコンピュータビジョンや画像処理の分野で広く用いられている [113]．また，SIFT を近似計算することで，高速化した Speeded Up Robust Features (SURF) [25] や，バイナリコードを用いた特徴量の記述を行う Binary Robust Invariant Scalable Keypoints (BRISK) [44]，アフィン変換をしながら対応付けを行う Affine-SIFT (ASIFT) [45] などが知られている．

最後に，バンドルアジャストメントを用いてカメラパラメータの最適化を行う [120], [121]．カメラの内部パラメータ $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ および外部パラメータ $\boldsymbol{R}_{1\to2}$, $\boldsymbol{t}_{1\to2}$ は，後段の処理である (ii) ステレオ平行化および (iii) 3 次元計測の処理に大きく影響する．そこで，これまでに算出されたパラメータ $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{R}_{1\to2}$, $\boldsymbol{t}_{1\to2}$ を初期値としてバンドルアジャストメントを適用し，これらのカメラパラメータを最適化する．

(II)　ステレオ平行化

図 4.6 (c) に示すように，画像 I_1 および I_2 を平行化し，画像 I_1' および I_2' を得る．ステレオ平行化とは，ステレオ画像をあたかも平行ステレオカメラで撮影された画像のように変形する処理である [48], [50]．後段の処理である (iii) 3 次元計測の処理では，物体全体の詳細な形状を計測するために，領域ベースの対応付けを用いてステレオ画像間の対応関係を求める．領域ベースの対応付けは，特徴点に限らず任意の画像座標について対応付けが可能である一方で，画像間に幾何的な変形が生じると正確な対応付けが困難になるという問題がある．

そこで，ステレオ平行化を行うことで画像間の画像変形を軽減する．画像 I_1, I_2 をそれぞれ I_1', I_2' に変換する射影変換行列は，(I) 節で推定したカメラパラメータを用いて算出することができる [101]．平行化後のステレオ画像間では，画像変形が水平座標方向に限定され，垂直座標方向の拡大縮小と回転が軽減される．これにより，カメラの移動撮影で取得されたステレオ画像に，ある程度の拡大縮小や回転が生じていた場合においても，密な 3 次元計測を行うことが可能になる．

(III)　3 次元形状計測

図 4.6 (d) に示すように，領域ベースの対応付けを用いてステレオ画像 I'_1–I'_2 間の対応関係を求める．図 4.6 (d) では，画像 I'_1 において HSV 色空間を用いた閾値処理によって背景と計測領域を分離し，計測領域に格子状に配置した基準点に対して対応関係を求めている．ただし，計測領域を判別する方法はこれに限らず，例えば，領域成長法のようなセグメンテーション手法を用いたり，ユーザに手動で計測領域を入力させたりするなど，計測の対象や環境によって最適なものを選択する必要がある．

さらに，求めた対応関係と (I) 節で推定されたカメラパラメータとを用いて，3 次元座標を計算する．計算された 3 次元点群を色付きで表示した様子を図 4.6 (e) に示す．3 次元形状計測では，物体の詳細な形状を計測するため，任意の点について対応付けが行える領域ベースの対応付け手法を用いる．領域ベースの対応付けアルゴリズムは，大域的な最適化に基づくアルゴリズムに比べて，容易にサブピクセル精度の対応付けが行えるため，対応付けの精度が計測精度に大きく影響する 3 次元計測に適している [50], [115]．領域ベースの対応付けアルゴリズムでは，ステレオ画像の片方の画像上において対応付けを行う点の周囲に局所的な画像ブロックを定義し，もう一方の画像上で類似した画像ブロックを探索することで対応付けを行う．ブロックマッチングに用いられるアルゴリズムとして，Sum of Absolute Differences (SAD) や Sum of Squared Differences (SSD) などの相違度に基づくアルゴリズム，および，Normalized Cross-Correlation (NCC) や Phase-Only Correlation (POC) などの類似度に基づくアルゴリズムが知られている [47], [52]．

4.3.3　性能評価

本節では，提案手法の 3 次元計測性能を評価する．まず，ステレオカメラで撮影された画像を用いて，カメラパラメータの推定精度および 3 次元形状計測の精度を評価する．次に，基線長を変化させたステレオ画像で 3 次元計測を行い，カメラの移動距離と計測精度の関係を調べる．さらに，ディジタルカメラや計測対象を変えて 3 次元計測を行い，ディジタル一眼レフカメラのような高価なカメラだけでなく，コンパクトディジタルカメラやスマートフォンに搭載

されているカメラを用いても，高精度な計測が可能であることを示す．最後に，提案手法の処理時間を計測することで，3 次元計測における処理時間のボトルネックが画像対応付けであること，および，画像対応付けアルゴリズムを GPU を用いて実装することでボトルネックが解消できることを示す．

(I)　ステレオカメラを用いた計測との比較

固定されたステレオカメラで撮影された画像を用いて，カメラパラメータの推定（特徴ベースの対応付けに基づく SfM）の精度，および 3 次元形状計測（領域ベースの密な対応付け）の精度をそれぞれ評価する．一般に，カメラパラメータの真値は不明であるため，推定されたカメラパラメータそのものを用いて，カメラパラメータの推定手法を評価することは難しい．これに対して，固定された 2 眼ステレオカメラでは，キャリブレーションによってカメラパラメータを求めることができる．そこで，固定された 2 眼ステレオカメラで撮影した画像を用いて，事前のキャリブレーションで求めたカメラパラメータと，SfM で推定したカメラパラメータとで 3 次元計測を行い，計測精度を比較する．

まず，図 4.7 に示すように，ステレオカメラを用いて猫の置物を撮影し，3 次元計測を行う．本実験で用いるステレオカメラの諸元は表 4.2 のとおりであり，撮影される画像の一例を図 4.8 (a) に示す．次に，コニカミノルタ社の 3 次元スキャナ VIVID910 [122] を用いて取得した 3 次元メッシュモデルを真値とし，計測した点群と 3 次元メッシュモデルとの間で ICP (Iterative Closest Point) を用いた位置合わせ [123] を行う．実験で用いる 3 次元メッシュモデルを図 4.8 (b) に示す．計測した点群にはスケールの不定性が存在するため，ICP を用いた位置合わせでは，回転，並進だけでなく，スケールも推定する．最後に，位置合わせした点群とメッシュモデルとの距離を誤差とし，誤対応率および誤差の二乗平均平方根 (Root Mean Square: RMS) を用いて評価を行う．ここで，3 次元計測した点のうち，3 次元メッシュモデルとの距離がステレオ画像上で 1 ピクセル以上である点を誤対応点とし，誤対応点が占める割合を誤対応率とする．なお，本実験の条件では，1 ピクセルの誤差が約 2.5 mm の誤差に相当する．誤差の RMS については，誤対応点を除いた計測点のみを用いて計算する．

カメラパラメータは，以下の 5 つの方法を用いてそれぞれ推定する．

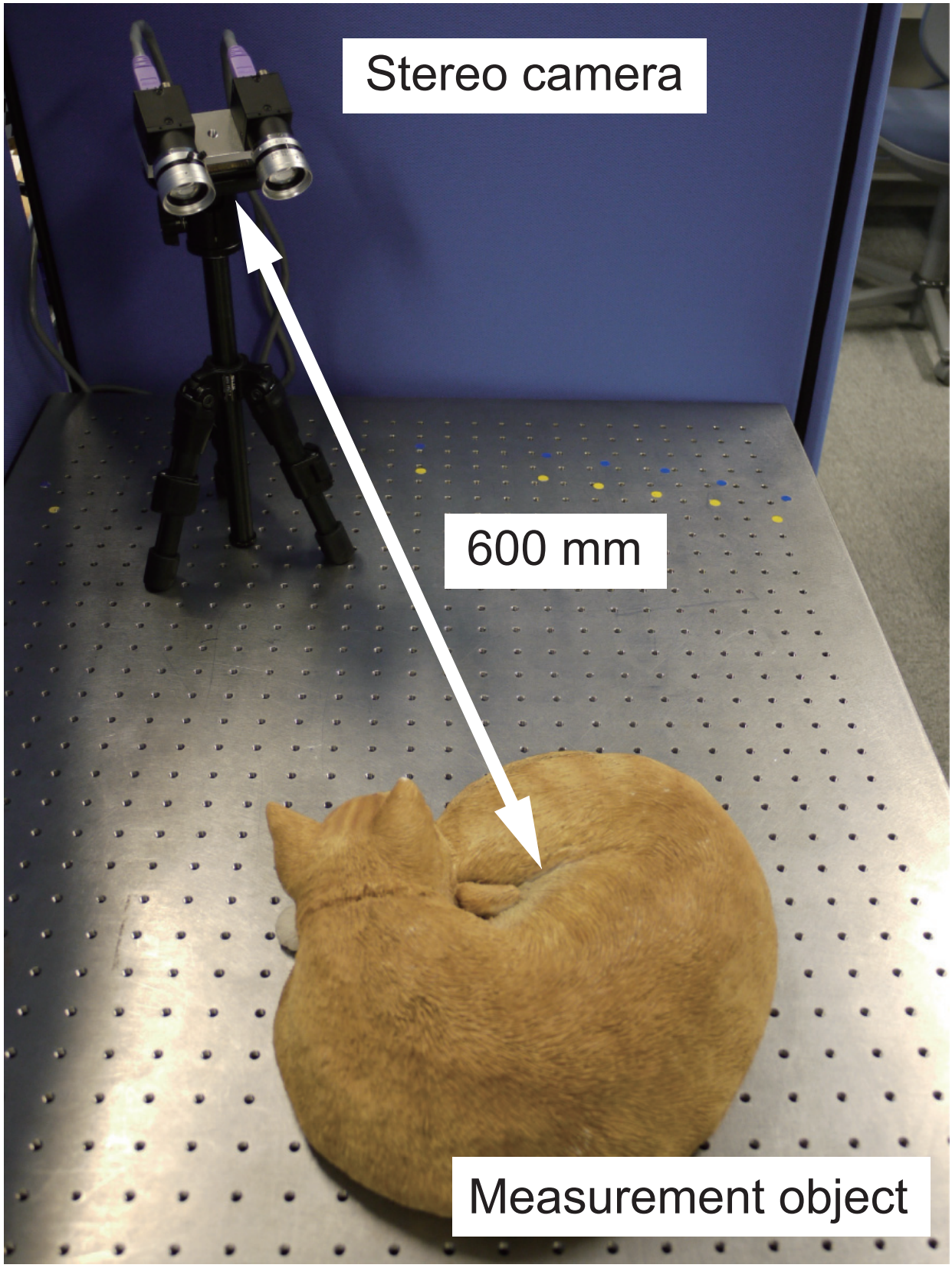


図 4.7　ステレオカメラを用いた画像の撮影

表 4.2　実験で用いるステレオカメラの諸元

ステレオカメラ	Point Grey Research Inc.,
	Flea2 (FL2G-13S2M-C), monochrome×2
基線長	60 mm
レンズ	μTRON FV0622, 焦点距離 6.5 mm
画像サイズ	$1,280 \times 960$ ピクセル

図 4.8　性能評価に用いるデータ：(a) ステレオ画像，(b) 真値のメッシュモデル

- **チェッカーパターン撮影に基づくキャリブレーション**

  あらかじめチェッカーパターンを撮影し, Zhang らのキャリブレーション手法 [101], [124] を用いて求める．以下では簡単のため，この手法を CALIB と記述する．

- **特徴ベースの対応付けに基づく SfM**

  特徴ベースの対応付けには，BRISK, SURF, SIFT, ASIFT をそれぞれ用いる．

  – **BRISK**

    Leutenegger らが公開しているソースコード [125] を用いて実装し，キーポイント検出の閾値を 60 とする．

  – **SURF に基づく対応付け**

    OpenCV [105] を用いて実装し，Fast-Hessian Detector の閾値を 500 とする．

  – **SIFT に基づく対応付け**

    Lowe が公開しているデモプログラム [126] を用いて実装する．

– **ASIFT**

Morel らが公開しているソースコード [127] を用いて実装する．

5 点アルゴリズムとバンドルアジャストメントについては，Bundler [109] を用いて実装する．

領域ベースの対応付けについては，以下の 4 つをそれぞれ用いる．

- SAD に基づくアルゴリズム
- SSD に基づくアルゴリズム
- NCC に基づくアルゴリズム
- POC に基づくアルゴリズム

それぞれのアルゴリズムでは，画像ピラミッドによる階層的探索 [28] を用いており，画像ピラミッドの階層数を 4 とする．探索ウィンドウのサイズは，SAD, SSD, NCC に基づくアルゴリズムでは 16 ピクセル ×15 ライン，POC に基づくアルゴリズムでは 32 ピクセル ×15 ラインとする．POC に基づく画像マッチングでは，ハニング窓を用いることでフーリエ変換の際に生じる信号端の不連続性の影響を抑えている．ハニング窓では信号長の半分が半値幅になることより，POC に基づくアルゴリズムにおいて 32×15 ピクセルのウィンドウに含まれる情報量は，他の対応付けアルゴリズムで用いる 16×15 ピクセルのウィンドウに含まれる情報量に相当する．サブピクセルレベルの平行移動量推定に関して，SAD に基づくアルゴリズムでは等角直線フィッティング [47] を，SSD, NCC に基づくアルゴリズムではパラボラフィッティング [47] を用いる．なお，撮影された画像において計測領域を手動で設定し，計測領域に 3 ピクセル間隔で格子状に配置した基準点に対して領域ベースの対応付けを行う．

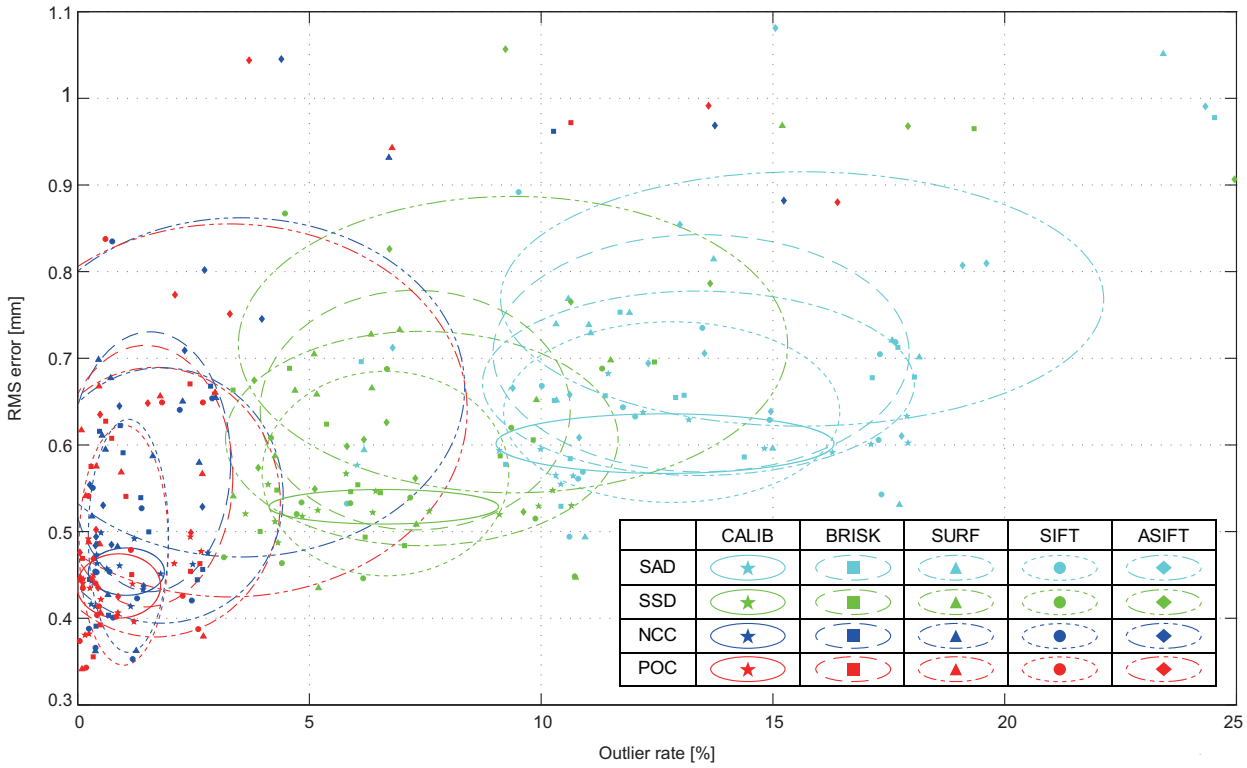


図 4.9　3 次元計測における誤対応率および誤差の RMS （カメラパラメータ推定手法の違いはマーカおよび線の種類，領域ベースの対応付けアルゴリズムの違いは色によって示す）

表 4.3　3 次元計測における誤対応率（14 回の試行における平均 ± 標準偏差）[%]

	CALIB	BRISK	SURF	SIFT	ASIFT
SAD	12.7 (±3.6)	13.4 (±4.7)	13.4 (±4.5)	12.8 (±3.6)	15.6 (±6.5)
SSD	6.6 (±2.5)	7.4 (±4.2)	7.3 (±3.4)	6.6 (±2.7)	9.4 (±5.9)
NCC	1.1 (±0.8)	1.8 (±2.6)	1.6 (±1.7)	1.1 (±0.9)	3.6 (±4.8)
POC	0.9 (±0.9)	1.7 (±2.7)	1.5 (±1.9)	1.0 (±1.0)	3.3 (±5.1)

表 4.4　3 次元計測における RMS 誤差（14 回の試行における平均 ± 標準偏差）[mm]

	CALIB	BRISK	SURF	SIFT	ASIFT
SAD	0.602 (±0.034)	0.671 (±0.107)	0.706 (±0.137)	0.638 (±0.103)	0.768 (±0.147)
SSD	0.529 (±0.020)	0.607 (±0.124)	0.642 (±0.137)	0.567 (±0.118)	0.716 (±0.171)
NCC	0.454 (±0.027)	0.542 (±0.148)	0.581 (±0.149)	0.494 (±0.136)	0.665 (±0.197)
POC	0.437 (±0.037)	0.533 (±0.156)	0.564 (±0.151)	0.484 (±0.139)	0.641 (±0.214)

計測における誤対応率および誤差の RMS を図 4.9 に示す．図 4.9 では，計測対象をさまざまな方向から 14 回撮影したそれぞれのステレオペアごとに，カメラパラメータの推定方法および領域ベースの対応付けのそれぞれについて，すべての手法の組合せを用いて計測および評価を行い，算出された誤対応率および誤差の RMS をプロットしている．さらに，手法の組合せごとに，誤対応率および誤差の RMS について，それぞれの平均を中心，それぞれの標準偏差を半径とした楕円を示している．これら平均および標準偏差を表 4.3 および表 4.4 に示す．以下では特に断りのない限り，平均を用いて議論する．

まず，領域ベースの対応付けの評価結果について考察する．カメラパラメータの推定方法に関わらず，SAD や SSD に基づくアルゴリズムと比べ，NCC や POC に基づくアルゴリズムは誤対応率が低い．図 4.10 は，それぞれのアルゴリズムを用いて計測を行った結果であるが，SAD や SSD に基づくアルゴリズムでは，誤対応が発生している領域が多いことがわかる．これは，SAD や SSD に基づくアルゴリズムでは，左右のカメラ画像の明るさの違いや物体表面の反射特性などの影響を受け易いためである．これに対して，NCC や POC に基づくアルゴリズムでは，これらの影響に対してロバストに対応付けができている．また，NCC に基づくアルゴリズムに比べ，POC に基づくアルゴリズムでは誤差の RMS が低い．NCC に基づくアルゴリズムでは，パラボラフィッティングを用いて離散点の間を補間することでサブピクセル精度の対応付けを行う．一方で，POC に基づくアルゴリズムでは，相関ピークの解析的なモデルを離散点にフィッティングすることでサブピクセル精度で対応付けを行うため，より正確に平行移動量を推定することが可能である．

次に，カメラパラメータの推定方法の評価結果について考察する．特徴ベースの対応付け (BRISK, SURF, SIFT, ASIFT) に基づく SfM の中では，SIFT に基づく SfM を用いてカメラパラメータを推定する場合に，誤対応率および誤差の RMS が最も低い．これは，領域ベースの対応付けについて，どのアルゴリズムを用いた場合も同様である．また，SIFT に基づく SfM では，キャリブレーションによってカメラパラメータを求める場合 (CALIB) と比べると，誤対応率は同程度である．しかしながら誤差の RMS に関しては，平均が低く，標準偏差が大きい．これは，キャリブレーションにおいて，さまざまな位置や向きを持ったチェッカーパターンを撮影することで画像全体にわたるコーナ点の対応関係を得ることができているのに

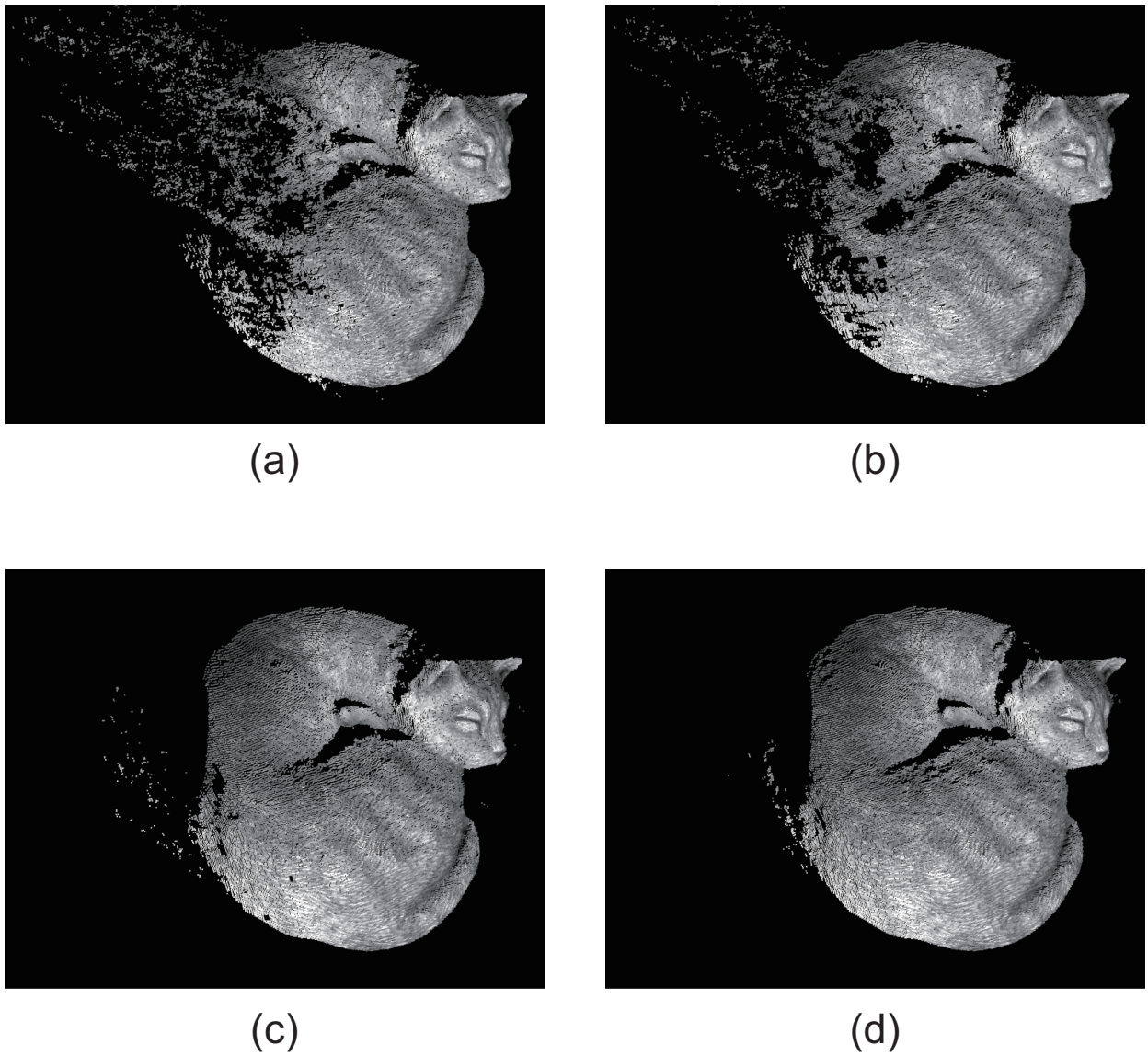


図 4.10　3 次元計測結果：(a) SAD, (b) SSD, (c) NCC, (d) POC（カメラパラメータは SIFT に基づく SfM を用いて推定）

対し，SIFT に基づく SfM では，対応関係を得ることができる特徴点の位置が画像中で偏ってしまうことがあるためだと考えられる．

以上より，提案手法では，SIFT に基づく SfM によるカメラパラメータの推定と，POC に基づく対応付けによる 3 次元形状計測の組合せが最も高精度に 3 次元計測を行うことができる．また，この組合せでは，キャリブレーションされたステレオカメラと比べ，サブミリレベルでの計測精度は低く，計測ごとにばらつきが大きいものの，大まかな形状を捉えるという観点からは同程度の 3 次元計測が可能である．

(II)　基線長を変化させた場合の評価

カメラの移動距離（基線長）を変化させた場合の 3 次元計測精度を評価し，提案手法が許容できる移動距離を調べる．

計測対象は，インテリア用のデコレーションタイルとする．これは，対象の形状に依存する画像変形の影響を抑え，基線長の変化による計測精度を評価するためには，平板に近い計測対象が適しているからである．評価に用いる画像は，図 4.11 に示すように撮影する．使用するディジタルカメラは Panasonic LUMIX DMC-GF3 であり, 撮影された画像を 1,280 × 960 ピクセルに縮小して計測処理を行う．まず，計測対象をカメラと正対する向きで回転ステージに載せ，画像を撮影する．このとき撮影された画像を基準画像 (0°) とする．次に，回転ステージを 5° から 40° まで 5° ずつ回転させながら画像を撮影し，撮影したそれぞれの画像と基準画像とをステレオペアとして 3 次元計測を行う．青地の背景下で撮影を行い，HSV 色空間における閾値処理によって計測領域を抽出する．計測領域に 3 ピクセル間隔で配置した格子状の基準点に対して 3 次元計測を行う．計測では，(I) 節で最も高精度であった SIFT に基づく SfM と POC に基づく対応付けの組合せを用いる．最後に，(I) 節と同様にして，計測した点群を真値のメッシュモデルとの位置合わせを行い，誤差のヒストグラムを用いて評価する．

計測結果を図 4.12，計測誤差のヒストグラムを図 4.13 に示す．図 4.13 のヒストグラムでは，誤差の小さい点の割合が多く，誤差の大きい点の割合が少ないほど，高精度に計測できていることを示す．すなわち，誤差 0 mm の頻度が高く，かつ，サイドローブの値が低くなるような急峻なピークを示していれば，計測誤差が小さいことを示している．回転ステージの角度

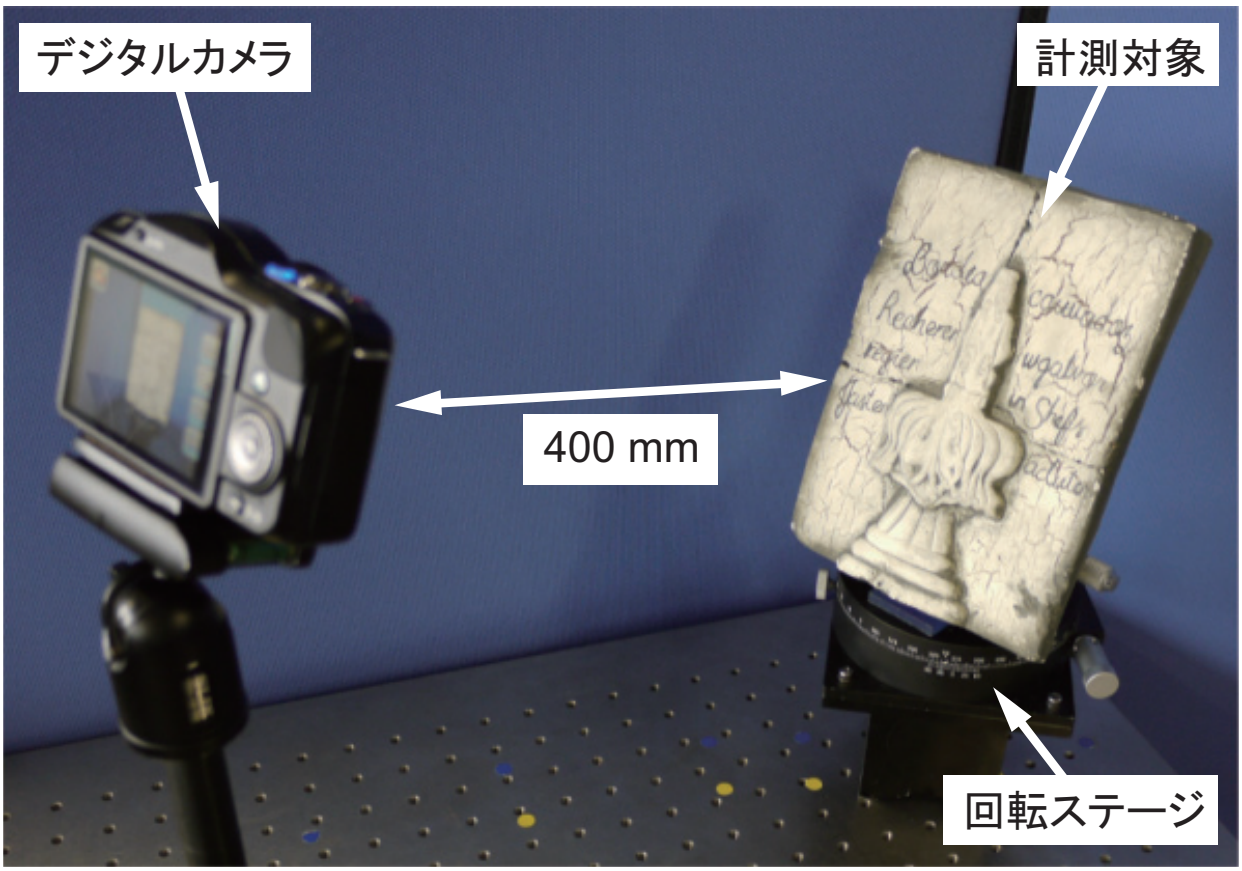


図 4.11　基線長を変化させた場合の評価

を 5° から 15° まで増加させると，誤差 0 mm の頻度が高くなるとともに，サイドローブの値が低くなる．これは，基線長が長くなることで，対応付け誤差が 3 次元計測結果に与える影響が小さくなっているためである．一方で，回転ステージの角度が 20° 以上になると，誤差 0 mm の頻度が低くなり，サイドローブの値が高くなる．これは，図 4.12 からもわかるように，ステレオ画像間の変形が大きくなり，計測された点の誤差が大きくなるからである．

以上の結果より，提案手法を用いた 3 次元計測では，カメラの輻輳角が 10° ～15° のときに，最も高精度な計測が可能である．カメラの輻輳角が 20° 以上になると，ステレオ画像間の変形が大きくなり，計測精度が低下する．このようにカメラの輻輳角が大きい場合，すなわち基線長の長いステレオ画像を用いて 3 次元計測を行う場合には，画像間の幾何変形にロバストな対応付けを用いる必要がある．これに対して，探索ウィンドウを拡縮することで，基線長の長いステレオ画像間においても POC に基づく画像対応付けアルゴリズムが可能であることが知ら

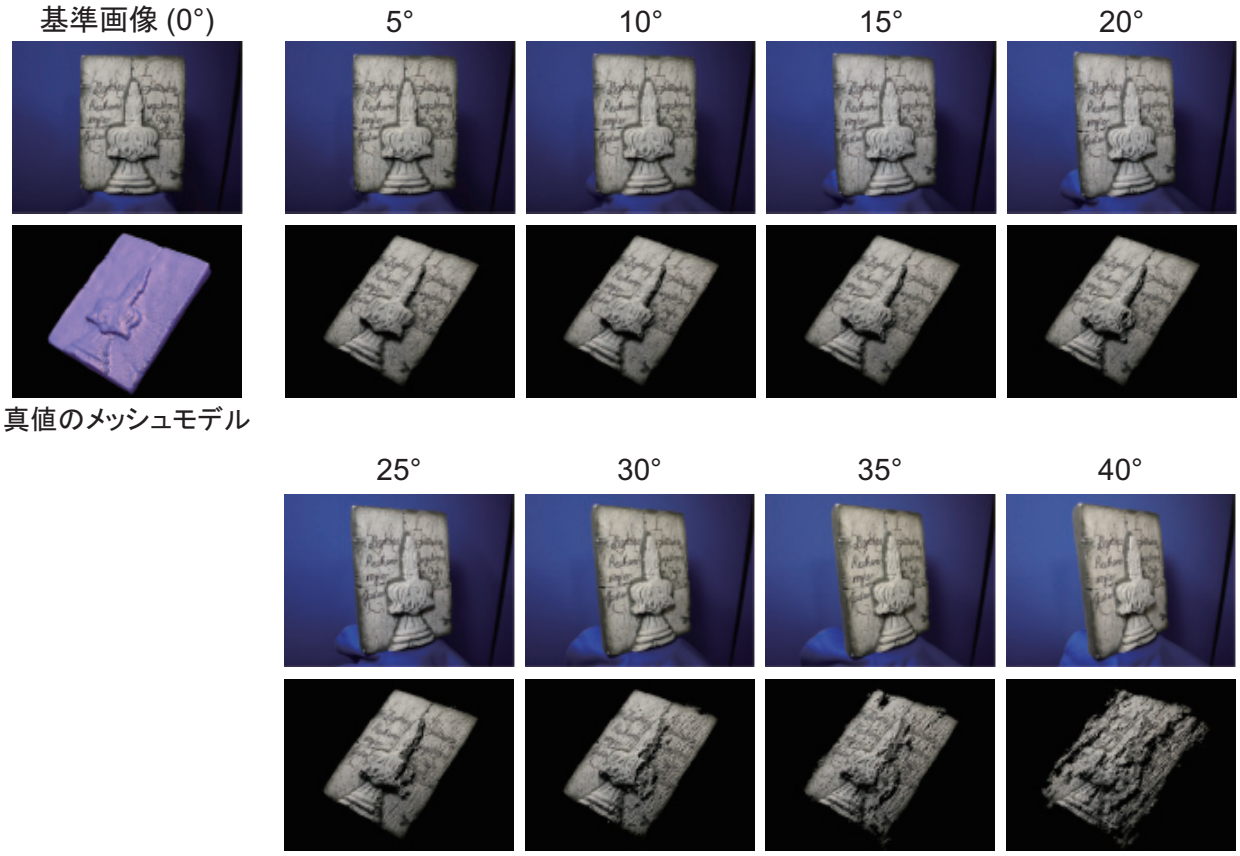


図 4.12　基線長を変化させた場合の 3 次元計測結果

れている [128]．なお，同手法は POC に基づく画像対応付けアルゴリズムを基礎としているため，本論文で述べた手法を用いて GPU による高速化が有効である．

(III)　汎用ディジタルカメラを用いた計測

撮影に用いるディジタルカメラ，および計測対象を変えて 3 次元計測を行い，提案手法がさまざまなカメラや計測対象に対して有効であることを確認する．

ディジタルカメラおよび計測対象を変えた場合の 3 次元計測結果を図 4.14 に示す．計測では，(I) 節で最も高精度であった SIFT に基づく SfM と POC に基づく対応付けの組合せを用いている．計測領域は，HSV 色空間における閾値処理によって自動的に抽出されている．計測処理におけるパラメータ，および誤対応率や誤差の RMS の評価方法は，(I) 節に述べたものと同様である．使用するディジタルカメラは，

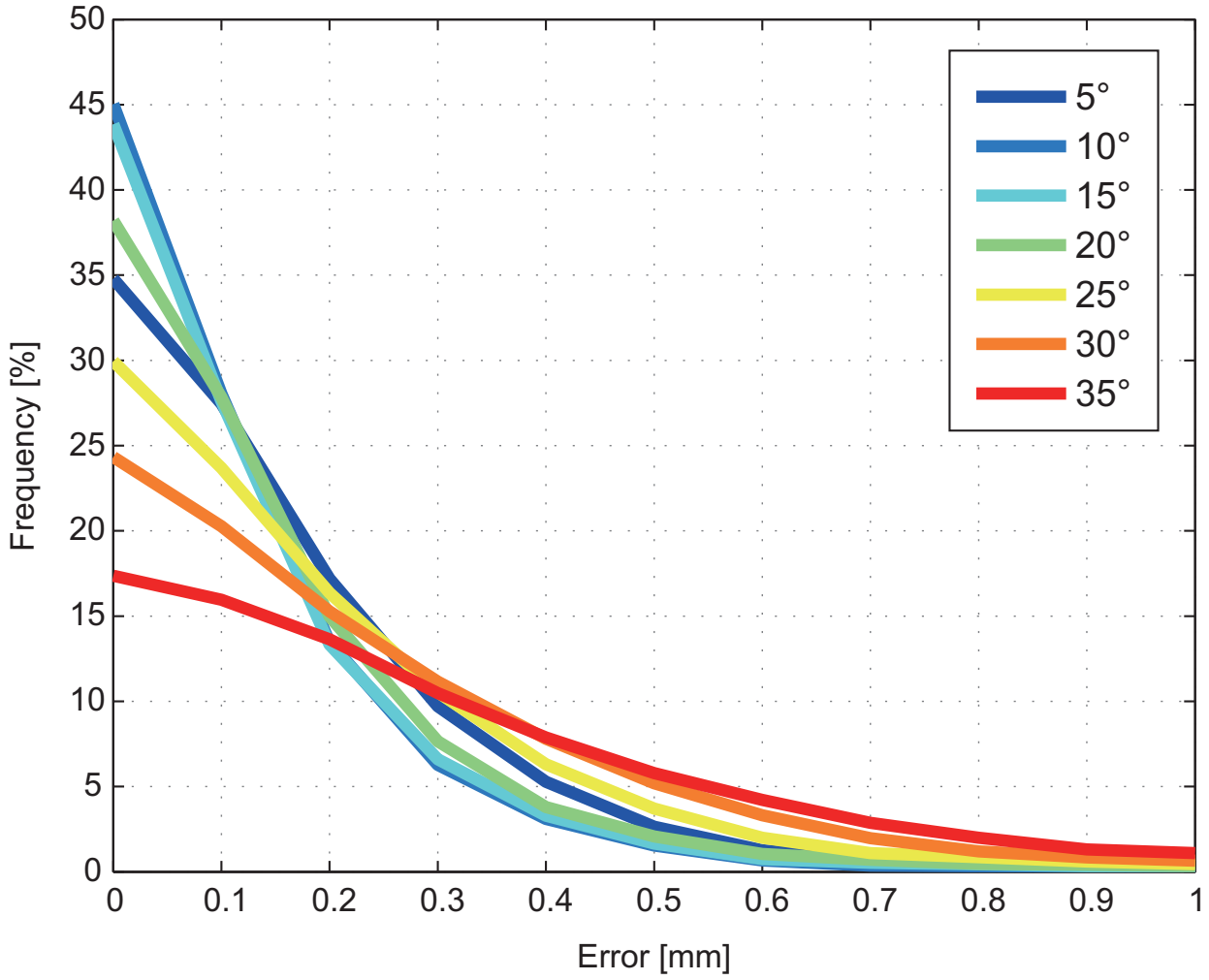


図 4.13　基線長を変化させた場合の計測誤差のヒストグラム

- Panasonic LUMIX DMC-GF3（ディジタル一眼レフカメラ）
- Canon IXY 30S（コンパクトディジタルカメラ）
- Google Nexus One（スマートフォン）

である．また，計測対象として，猫の置物，デコレーションタイル，犬の置物を撮影している．図 4.14 (a)~(c) より，ディジタル一眼レフカメラのように高性能なディジタルカメラだけでなく，スマートフォンに搭載されているカメラを用いても，3 次元スキャナに対して 1 mm 以下の誤差で 3 次元計測が可能である．さらに，図 4.14 (a), (d), (e) より，さまざまな計測対象に対して同様に 3 次元計測が可能である．

また，一般物体を対象に，システムを用いて 3 次元計測を行った結果を図 4.15 に示す．それぞれの物体について，特徴ベースの対応付けに基づく SfM のみでは計測が困難な，比較的テクスチャの乏しい領域も計測できていることが確認できる．

以上より，提案手法を用いることで，コンパクトデジタルカメラやスマートフォンに搭載されているカメラのような比較的安価な機材を用いても，さまざまな対象を 3 次元計測できることがわかる．

(IV)　処理時間

表 4.5 に示す環境において，システムが 3 次元計測に要する時間を計測する．計測は，SURF に基づく SfM と POC に基づく対応付けの組合せを用いて，すべての処理を CPU で行った場合と，POC に基づく対応付けをのみを GPU に実装した場合で行う．その結果を表 4.6 に示す．表 4.6 より，CPU 実装では処理時間のボトルネックとなっていた POC に基づく対応付けが，GPU を用いた実装によって解消されていることがわかる．また，特徴ベースに基づく対応付けとカメラパラメータの推定を高速化することで，さらに高速な 3 次元計測を行うことができると考えられる．例えば，SIFT に基づく対応付けについては，GPU を用いた並列処理によって 10 倍程度高速化できることが明らかにされている [129]．バンドルアジャストメントについても，CPU のマルチコアや GPU を用いた実装によって 10 ～ 30 倍の高速化が可能であることが報告されている [130]．

提案システムでは，2 視点のみから 3 次元計測を行っているが，3 視点以上からの 3 次元計測に拡張することが可能である [34], [36]．すなわち，(i) V 視点を用いて $V-1$ 組のステレオ画像ペアを作成し，(ii) それぞれのステレオ画像ペアにおいて 3 次元計測を行い，(iii) それぞれの 3 次元計測結果をカメラの外部パラメータを用いて同一の座標系に統合する．さらに，これまでに述べたアルゴリズムや実装を用いることで 4.3.1 節に述べたステレオ画像ペアごとの処理が高速化できれば，ビデオカメラなどを用いた動画像からのリアルタイムな 3 次元計測が可能であると考えられる．

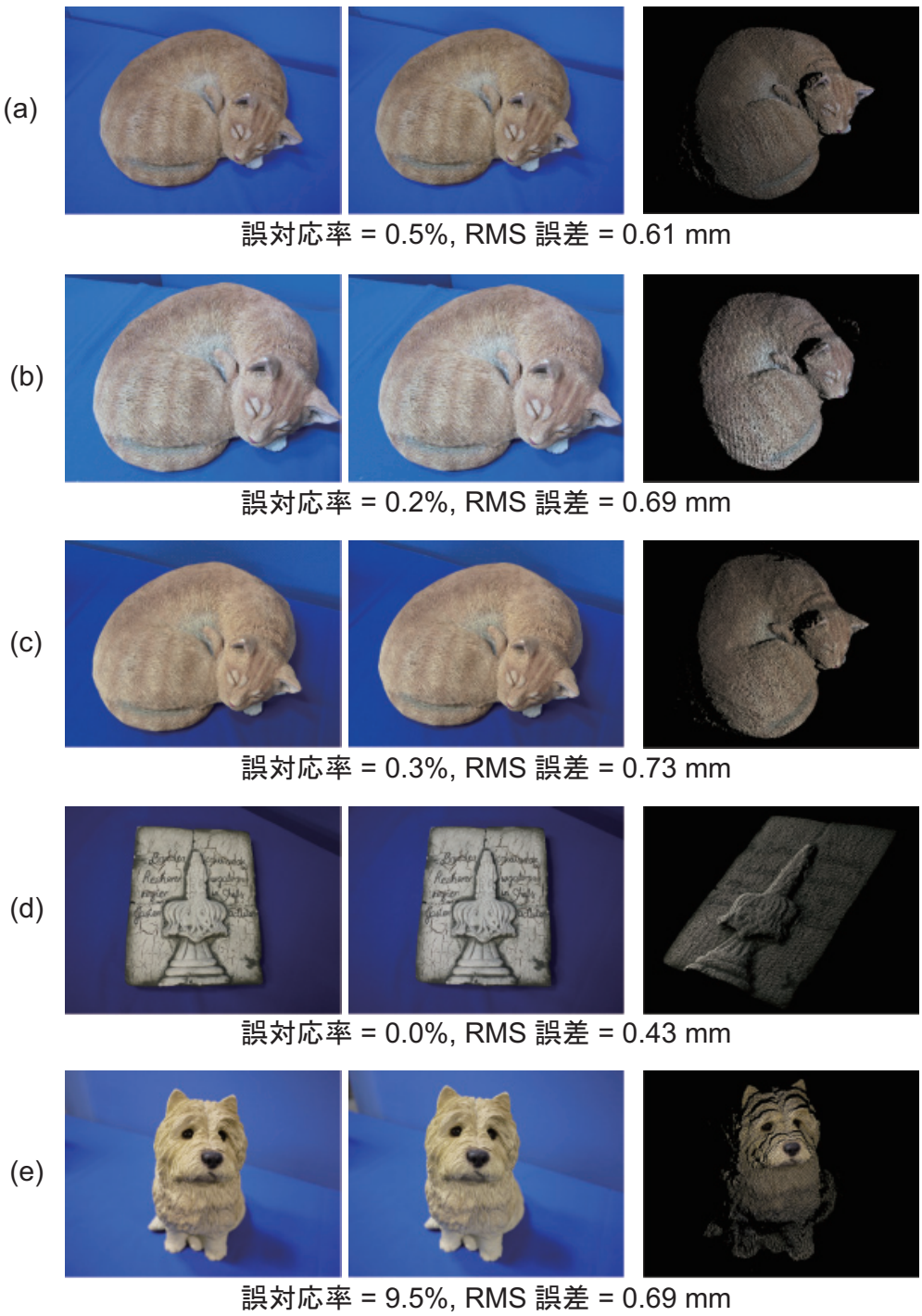


図 4.14　3 次元計測結果：(a) Panasonic LUMIX DMC-GF3, (b) Canon IXY 30S, (c) Google Nexus One, (d) LUMIX DMC-GF3, デコレーションタイル, (e) LUMIX DMC-GF3, 犬の置物

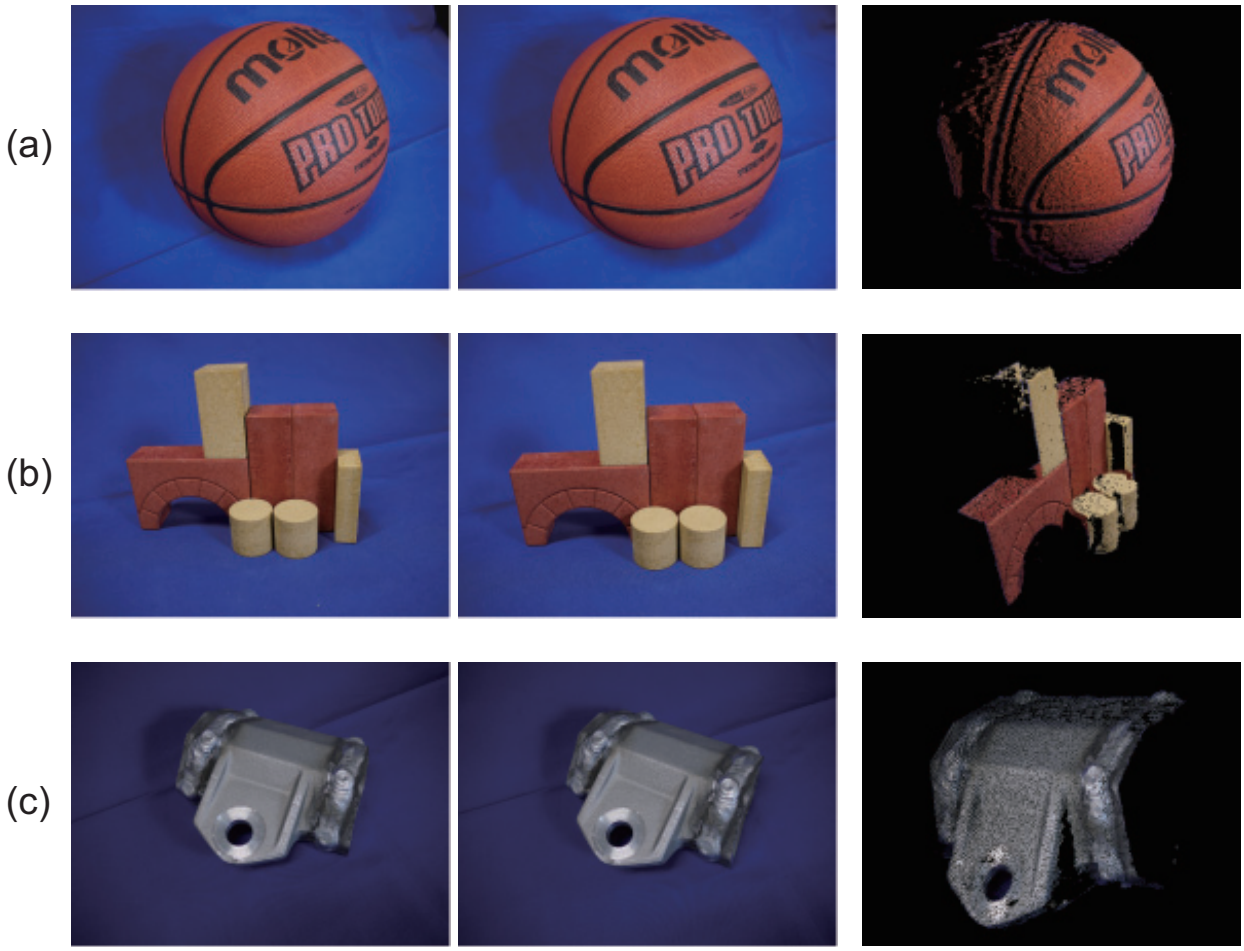


図 4.15　一般物体の 3 次元計測結果：(a) バスケットボール，(b) 積み木，(c) 金属部品

4.4　むすび

本章では，1 次元 POC に基づく画像対応付けアルゴリズムの GPU 実装をもとに，(a) リアルタイム 3 次元計測システムと (b) 汎用デジタルカメラを用いた 2 視点からの 3 次元計測システムを構築した．また，これらのシステムの評価を通して，1 次元 POC に基づく画像対応付けアルゴリズムを GPU に実装することが，ステレオビジョンに基づく受動型 3 次元計測において有効であることを実証した．

表 4.5　処理時間の測定環境

CPU	Intel Core i7-2760QM (2.4 GHz)
メモリ	DDR3-1333 SDRAM 4GB× 2
GPU	NVIDIA GeForce GTX 570M
OS	Windows 7 (64 bit)
コンパイラ	Visual C++ 9.0 Compiler (CPU code)
画像サイズ	$1,280 \times 960$ ピクセル
探索ウィンドウ	32 ピクセル × 15 ライン
探索階層数	4 階層

表 4.6　2 視点からの 3 次元計測における処理時間内訳 [ms]

処理	CPU 実装	CPU+GPU 実装
カメラパラメータの推定	2,077	2,077
ステレオ平行化	123	123
POC に基づく対応付け	3,023	229
3 次元座標の計算	19	19

第 5 章

結言

以上，第 2 章から第 4 章まで，メニーコアプロセッサにおける画像対応付けアルゴリズムの実装とその応用について述べた．

第 2 章では，画像対応付けアルゴリズムとその応用に関する基礎的考察を行った．はじめに，画像対応付けアルゴリズムを特徴ベースのアルゴリズムと領域ベースのアルゴリズムに分類し，それぞれについて概説した．次に，画像対応付けの応用のひとつとして，ステレオビジョンに基づく 3 次元計測について述べた．さらに，ステレオビジョンに基づく 3 次元計測においては，対応付けの密度，精度，速度のすべてにおいて高い性能が求められることに言及した．これを踏まえて，特徴ベースのアルゴリズムと領域ベースのアルゴリズムを比較し，ステレオビジョンに基づく 3 次元計測においては領域ベースのアルゴリズムが適していると結論した．続けて，領域ベースのアルゴリズムのうち，本論文で着目する位相限定相関法に基づく画像対応付けアルゴリズムについて述べた．特に，ステレオビジョンに基づく 3 次元計測では，画像対応付けを対応点の 2 次元方向の探索ではなく 1 次元方向の探索に帰着させることができ，1 次元位相限定相関法を用いることで計算量やメモリアクセス量の削減ができることにも触れた．最後に，領域ベースのアルゴリズムを用いた 3 次元計測の性能評価を通して，位相限定相関法に基づく画像対応付けアルゴリズムがステレオビジョンに基づく 3 次元計測において密度および精度，ロバスト性の観点から有効であることを示した．

第 3 章では，1 次元位相限定相関法に基づく画像対応付けアルゴリズムの GPU 実装を提案

した．まず，NVIDIA 製および AMD 製の GPU のアーキテクチャについてそれぞれ述べ，それら GPU においてアルゴリズムを実装するためのプログラミングモデルについて概説した．次に，1 次元位相限定相関法に基づく画像対応付けアルゴリズムを GPU に実装する手法について述べた．GPU における実装においては，チップ外のメモリ（ビデオメモリ，プログラミングモデルではグローバルメモリとも呼ぶ）へのアクセスを削減し，スループットの高いパターンでアクセスすること，および，演算ユニットにおいて同時にスケジューリング・実行されるスレッドの数を調整することが重要であることにも触れた．これを踏まえ，GPU のアーキテクチャに合わせた実装の最適化についても説明した．最後に，GPU を用いた実装の性能評価を通して，実装の最適化によって，素朴な実装に比べて約 3.5 倍の高速化が実現できることを示した．さらに，GPU を用いた実装と CPU を用いた実装の性能評価を通して，GPU を用いた実装が，CPU を用いた実装に比べて約 40 倍高速に対応付けが可能であること，および，電力効率の面からも GPU を用いた実装が優れていることを示した．

第 4 章では，ステレオビジョンに基づく 3 次元計測へ提案手法を応用し，その有効性を実証した．具体的には，(a) リアルタイムに高精度・高密度な 3 次元計測を行うシステムと，(b) デジタルカメラを用いた 2 回の撮影から物体の 3 次元復元を行うシステムを構築した．(a) リアルタイムに高精度・高密度な 3 次元計測を行うシステムでは，GPU を搭載した汎用の計算機とステレオカメラを組み合わせてシステムを構築し，人物の顔の形状を計測した．システムの性能評価を通して，GPU を用いた画像対応付けアルゴリズムの実装を用いることで，汎用の計算機を用いても高精度・高密度な 3 次元計測をリアルタイムで実現できることを示した．(b) デジタルカメラを用いた 2 回の撮影から物体の 3 次元復元を行うシステムでは，汎用のデジタルカメラを用いた移動撮影によって撮影された 2 枚の画像から，被写体の詳細な 3 次元形状を計測した．計測のボトルネックとなる密な対応付けの処理時間が，GPU を用いた実装によって低減されることを示した．

以上，本論文では，ステレオビジョンに基づく 3 次元計測において重要な基礎技術である画像対応付けに着目し，位相限定相関法に基づく高精度画像対応付けアルゴリズムの GPU を用いた高速な実装を提案することで，将来の普及が期待されるメニーコアプロセッサに対して画像処理アルゴリズムを適合させる一例を示した．メニーコアプロセッサのひとつとして GPU

に着目し，GPU のアーキテクチャに適したアルゴリズムの実装手法を提案した．提案する実装によって，素朴な実装に比べて高速な処理が可能であること，および，CPU を用いた実装に比べて処理時間と電力効率の面から優れていることを示した．また，提案手法を応用し，GPU が搭載された汎用の計算機を用いて 3 次元計測システムを構築することで，GPU を用いた画像対応付けアルゴリズムの実装によって，汎用の計算機を用いても高精度・高密度・高速な 3 次元計測が実現できることを示した．

今後の展望としては，まず，GPU を用いた画像対応付けアルゴリズムの実装に対するアーキテクチャレベルでのプロファイリング（性能解析）とモデル化が挙げられる．さまざまな画像対応付けアルゴリズムを実装し，その性能解析を行うことで，アルゴリズムがプロセッサで実行されるモデルを確立し，性能予測が可能になると考える．さらに，これを発展させ，プロセッサのアーキテクチャとアルゴリズムにおける演算やデータの依存関係をもとにスレッドやメモリの割り当てを決定し，プログラムや擬似コードを出力するような，メニーコアプロセッサにおける画像対応付けアルゴリズムの実装のためのフレームワークを確立することができると考える．また，本論文で構築した 3 次元計測システムを発展させ，実用化することも考えられる．例えば，Audi は，運転支援や自動運転に向けて，自動車にカメラなどのセンサを充実させ，センサから得られた情報を処理するために組込みプロセッサを搭載することを発表した [131]．この組込みプロセッサは ARM コアと NVIDIA 製の GPU から構成されており，本論文で提案する画像対応付けアルゴリズムの GPU 実装を応用することで，ステレオビジョンに基づく 3 次元計測による走行環境計測が実現できると考える．その他の応用としては，FA (Factory Automation) における部品の位置合わせや製品の品質検査，監視カメラを用いた周辺監視などが考えられる．いずれの応用にしても，今後は計算資源としてメニーコアプロセッサが普及し，プロセッサのアーキテクチャに適合した画像処理および画像対応付けの高速な実装に対する需要が高まることが予想される．

参考文献

[1] A. W. Burks and A. R. Burks, "First general-purpose electronic computer," *Annals of the History of Computing*, Vol. 3, No. 4, pp. 310–389, 1981.

[2] 総務省, "情報通信白書平成 26 年度版." http://www.soumu.go.jp/johotsusintokei/whitepaper/index.html.

[3] キーエンス, "走るコンピュータ！加速する自動車の電子化." http://www.keyence.co.jp/req/u/kaj3gri/show.jsp.

[4] NVIDIA Corporation, "GeForce Hardware." http://www.geforce.com/hardware.

[5] Intel Corporation, "ARK — Your Source for Intel®Product Information." http://ark.intel.com/.

[6] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, Vol. 26, No. 1, pp. 80–113, 2007.

[7] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, Vol. 96, No. 5, pp. 879–899, 2008.

[8] S. E Krakiwsky, L. E. Turner, and M. M. Okoniewski, "Acceleration of finite-difference time-domain (FDTD) using graphics processor units (GPU)," *Proc. Int'l Microwave Symposium*, Vol. 2, pp. 1033–1036, 2004.

[9] Samuel Adams, Jason Payne, and Rajendra Boppana, "Finite difference time domain (FDTD) simulations using graphics processors," *Proc. DoD High Performance Computing Modernization Program Users Group Conference*, pp. 334–338, 2007.

[10] M. Harris, “Fast fluid dynamics simulation on the GPU,” *GPU gems*, Vol. 1, pp. 637–665, 2004.

[11] Y. Liu, X. Liu, and E. Wu, “Real-time 3D fluid simulation on GPU with complex obstacles,” *Proc. Conf. Computer Graphics and Applications*, pp. 247–256, 2004.

[12] “GPGPU.org.” `http://gpgpu.org`.

[13] “CUDA ZONE.” `http://www.nvidia.com/object/cuda_home_new.html`.

[14] Khronos OpenCL Working Group, OpenCL 詳説. カットシステム, 2011.

[15] Intel Corporation, “Intel®64 and IA-32 architectures optimization reference manual.” `http://www.intel.com/Assets/PDF/manual/248966.pdf`.

[16] Advanced Micro Devices, Inc., “AMD fusion family of APUs: White paper.” `http://sites.amd.com/us/Documents/48423B_fusion_whitepaper_WEB.pdf`.

[17] 奥富正敏（編）, ディジタル画像処理. 画像情報教育振興協会, 2004.

[18] 日立アプライアンス, “エアコン S シリーズ 特長 [くらしカメラ].” `http://kadenfan.hitachi.co.jp/ra/lineup/sseries_c/feature/02/`.

[19] 青山章一郎, 伊藤康一, 青木孝文, 太田陽基, “携帯電話向け非接触掌紋認証アルゴリズム,” 電子情報通信学会論文誌 A, Vol. 96, No. 5, pp. 250–263, 2013.

[20] 田島裕一郎, 宮澤一之, 青木孝文, 勝亦敦, 小林孝次, “三次元位相限定相関法に基づく高精度ボリュームレジストレーション,” 電子情報通信学会論文誌 D, Vol. 94, No. 8, pp. 1398–1409, 2011.

[21] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, “Building rome in a day,” *Proc. Int’l Conf. Computer Vision*, pp. 72–79, 2009.

[22] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, “Multi-view stereo for community photo collections,” *Proc. Int’l Conf. Computer Vision*, pp. 1–8, 2007.

[23] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Towards internet-scale multi-view stereo,” *Proc. Int’l Conf. Computer Vision and Pattern Recognition*, pp. 1434–1441, 2010.

[24] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int’l J.*

Computer Vision, Vol. 60, No. 2, pp. 91–110, 2004.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346–359, 2008.

[26] 青木孝文, 伊藤康一, 柴原琢磨, 長嶋聖, "位相限定相関法に基づく高精度マシンビジョン―ピクセル分解能の壁を越える画像センシング技術を目指して―," *IEICE Fundamentals Review*, Vol. 1, No. 1, pp. 30–40, 2007.

[27] M. A. Muquit, T. Shibahara, and T. Aoki, "A high-accuracy passive 3D measurement system using phase-based image matching," *IEICE Trans. Fundamentals*, Vol. E89-A, No. 3, pp. 686–697, 2006.

[28] 柴原琢磨, 沼徳仁, 長嶋聖, 青木孝文, 中島寛, 小林孝次, "一次元位相限定相関法に基づくステレオ画像の高精度サブピクセル対応付け手法," 電子情報通信学会論文誌, Vol. J91-D, No. 9, pp. 2343–2356, 2008.

[29] 三浦衛, 高橋徹, 伊藤康一, 青木孝文, "位相限定相関法に基づくリアルタイム 3 次元計測システム," 第 12 回 DSPS 教育者会議, pp. 87–88, 2010.

[30] 三浦衛, 札野欽也, 伊藤康一, 青木孝文, "位相限定相関法を用いたステレオビジョンに基づくリアルタイム 3 次元計測," 第 17 回 画像センシングシンポジウム, No. IS4-30, 2011.

[31] 三浦衛, 札野欽也, 伊藤康一, 青木孝文, 滝沢寛之, 小林広明, "GPU 実装に基づくリアルタイムステレオビジョンシステム," 第 18 回画像センシングシンポジウム, No. DS1-03, 2012.

[32] 三浦衛, 札野欽也, 伊藤康一, 青木孝文, 滝沢寛之, 小林広明, "GPU 実装に基づくリアルタイム 3 次元計測システム," 第 15 回画像の理解・認識シンポジウム (MIRU 2012), No. DS-14, 2012.

[33] 三浦衛, 酒井修二, 石井純平, 山尾創輔, 伊藤康一, 青木孝文, "カメラの移動撮影に基づく 2 視点からの 3 次元形状計測とその性能評価," 映像情報メディア学会論文誌, Vol. 68, No. 4, pp. J135–J143, 2014.

[34] 山尾創輔, 三浦衛, 酒井修二, 伊藤康一, 青木孝文, “カメラの移動撮影に基づくオンライン 3 次元計測システム,” 第 20 回画像センシングシンポジウム, No. DS2-03, 2014.

[35] M. Miura, S. Sakai, J. Ishii, K. Ito, and T. Aoki, “An easy-to-use and accurate 3D shape measurement system using two snapshots,” *Proc. Int'l Workshop on Advanced Image Technology*, pp. 1103–1106, 2013.

[36] 山尾創輔, 三浦衛, 酒井修二, 伊藤康一, 青木孝文, “デジタルカメラの移動撮影に基づく 3 次元形状計測システム,” 第 16 回画像の理解・認識シンポジウム (MIRU 2013), No. DS-3, 2013.

[37] 三浦衛, 酒井修二, 石井純平, 伊藤康一, 青木孝文, “デジタルカメラを用いた簡便で高精度なステレオビジョンシステム,” 第 19 回画像センシングシンポジウム, No. DS1-08, 2013.

[38] 三浦衛, 酒井修二, 石井純平, 伊藤康一, 青木孝文, “カメラの移動撮影に基づく簡便な 3 次元形状計測システムとその性能評価,” 第 15 回画像の理解・認識シンポジウム (MIRU 2012), No. DS-13, 2012.

[39] 三浦衛, 酒井修二, 石井純平, 伊藤康一, 青木孝文, “汎用デジタルカメラを用いた 2 視点からの 3 次元復元システム,” 第 14 回画像の理解・認識シンポジウム (MIRU 2011), No. DS-6, pp. 1707–1708, 2011.

[40] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” *Int'l J. Computer Vision*, Vol. 74, No. 1, pp. 59–73, 2007.

[41] “AutoStitch.” `http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html`.

[42] 都築勇司, 藤吉弘亘, 金出武雄, “SIFT 特徴量に基づく Mean-Shift 探索による特徴点追跡,” 情報処理学会論文誌 CVIM, Vol. 20, pp. 35–45, 2007.

[43] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” *Proc. ECCV Workshop on statistical learning in computer vision*, Vol. 1, No. 1-22, pp. 1–2, 2004.

[44] S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary robust invariant

scalable keypoints," *Proc. Int'l Conf. Computer Vision*, pp. 2548–2555, 2011.

[45] J. M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," *SIAM J. Imaging Sciences*, Vol. 2, No. 2, pp. 438–469, 2009.

[46] M. Shimizu and M. Okutomi, "Precise sub-pixel estimation on area-based matching," *Proc. Int'l Conf. Computer Vision*, Vol. 1, pp. 90–97, 2001.

[47] 清水雅夫, 奥富正敏, "画像のマッチングにおけるサブピクセル推定の意味と性質," 電子情報通信学会論文誌 D, Vol. J85-DII, No. 12, pp. 1791–1800, 2002.

[48] 徐剛, 辻三郎, 3 次元ビジョン. 共立出版, 1998.

[49] R. Hartley, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2008.

[50] R. Szeliski, *Computer Vision: Algorithms and Applications.* Springer, 2010.

[51] K. Takita, T. Aoki, Y. Sasaki, T. Higuchi, and K. Kobayashi, "High-accuracy subpixel image registration based on phase-only correlation," *IEICE Trans. Fundamentals*, Vol. 86, No. 8, pp. 1925–1934, 2003.

[52] K. Takita, M. A. Muquit, T. Aoki, and T. Higuchi, "A sub-pixel correspondence search technique for computer vision applications," *IEICE Trans. Fundamentals*, Vol. E87-A, No. 8, pp. 1913–1923, 2004.

[53] T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi, "A sub-pixel stereo correspondence technique based on 1D phase-only correlation," *Proc. Int'l Conf. Image Processing*, Vol. 5, pp. V–221–V–224, 2007.

[54] S. Nagashima, T. Aoki, T. Higuchi, and K. Kobayashi, "A subpixel image matching technique using phase-only correlation," *Int'l Symp. Intelligent Signal Processing and Communications*, pp. 701–704, 2006.

[55] M. Miura, K. Fudano, K. Ito, and T. Aoki, "GPU implementation of phase-based image correspondence matching and its evaluation," *Proc. Int'l Symp. Low-Power and High-Speed Chips*, No. 19, 2011.

[56] M. Miura, K. Fudano, K. Ito, T. Aoki, H. Takizawa, and H. Kobayashi, "GPU

implementation of phase-based stereo correspondence and its application," *Proc. Int'l Conf. Image Processing*, pp. 1697–1700, 2012.

[57] M. Miura, K. Fudano, K. Ito, T. Aoki, H. Takizawa, and Kobayashi H., "Performance evaluation of phase-based correspondence matching on GPUs," *Proc. SPIE Optics + Photonics*, Vol. 8856, No. 885614, 2013.

[58] 三浦衛, 高橋徹, 伊藤康一, 青木孝文, "位相限定相関法に基づくリアルタイム 3 次元計測の検討," 平成 22 年度電気関係学会東北支部連合大会, No. 1D07, p. 110, 2010.

[59] 三浦衛, 高橋徹, 伊藤康一, 青木孝文, "位相限定相関法に基づく画像マッチングの GPU 実装とその応用," 精密工学会・映像情報メディア学会合同サマーセミナー, Vol. 34, No. 34, pp. 63–66, 2010.

[60] 三浦衛, 札野欽也, 伊藤康一, 青木孝文, 滝沢寛之, 小林広明, "位相情報に基づく画像対応付けの GPU 実装と評価," GTC Workshop Japan 2011 テクニカルセッション, 2011.

[61] M. Miura, K. Fudano, K. Ito, T. Aoki, H. Takizawa, and H. Kobayashi, "Performance evaluation of real-time stereo correspondence on GPU," 平成 23 年度 電気関係学会東北支部連合大会, No. 2A18, p. 31, 2011.

[62] D. B. Kirk and W.W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 2009.

[63] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: a unified graphics and computing architecture," *IEEE Micro*, Vol. 28, No. 2, pp. 39–55, 2008.

[64] NVIDIA Corporation, "NVIDIA GeForce GTX 280 specifications." `http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-280/specifications`.

[65] NVIDIA Corporation, "Fermi Architecture White Paper." `http://www.nvidia.com/content/pdf/fermi_white_papers/nvidia_fermi_compute_architecture_whitepaper.pdf`.

[66] NVIDIA Corporation, "NVIDIA GeForce GTX 480 specifications." `http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/specifications`.

[67] NVIDIA Corporation, "NVIDIA GeForce GTX 580 specifications." `http://www.`

geforce.com/hardware/desktop-gpus/geforce-gtx-580/specifications.

[68] 大原雄介, “骨まで理解する PC アーキテクチャ（GPU 編）第 2 回 ～GPGPU への最適化や電力効率向上へ進んだ NVIDIA GPU の歩み.” http://pc.watch.impress.co.jp/docs/column/1month-kouza/20140411_643848.html.

[69] NVIDIA Corporation, “Kepler Compute Architecture White Paper.” http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf.

[70] NVIDIA Corporation, “NVIDIA GeForce GTX 680 specifications.” http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-680/specifications.

[71] NVIDIA Corporation, “NVIDIA GeForce GTX 780 specifications.” http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-780/specifications.

[72] NVIDIA Corporation, “NVIDIA GeForce 8800 GTX specifications.” http://www.geforce.com/hardware/desktop-gpus/geforce-8800-gtx/specifications.

[73] NVIDIA Corporation, “NVIDIA GeForce GTX 780 Ti specifications.” http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-780-ti/specifications.

[74] GPUReview.com, “ATi Radeon HD 4890 Video Card - Reviews, Specifications, and Pictures.” http://www.gpureview.com/Radeon-HD-4890-card-608.html.

[75] Advanced Micro Devices, Inc., “AMD Radeon HD 5870 Graphics.” http://www.amd.com/en-us/products/graphics/desktop/5000/5870.

[76] Advanced Micro Devices, Inc., “AMD Evergreen-Family Instruction Set Archtecture.” http://developer.amd.com/wordpress/media/2012/10/AMD_Evergreen-Family_Instruction_Set_Architecture.pdf.

[77] Advanced Micro Devices, Inc., “AMD Radeon HD 6970 Graphics.” http://www.amd.com/en-us/products/graphics/desktop/6000/6970.

[78] Advanced Micro Devices, Inc., “AMD GCN microarchitecture whitepaper.” https://raw.githubusercontent.com/AMD-FirePro/SDK/master/documentation/GCN_Architecture_whitepaper.pdf.

[79] 大原雄介, “骨まで理解する PC アーキテクチャ（GPU 編）第 3 回 ～ ビッグ GPU から効率重視へ、そして GCN へと繋がる AMD GPU の歴史.” http://pc.watch.impress.co.jp/docs/column/1month-kouza/20140418_644945.html.

[80] Advanced Micro Devices, Inc., “AMD Radeon HD 7900 series graphics cards.” http://www.amd.com/en-us/products/graphics/desktop/7000/7900.

[81] Advanced Micro Devices, Inc., “AMD Radeon R9 series graphics.” http://www.amd.com/en-us/products/graphics/desktop/r9.

[82] NVIDIA Corporation, “OpenCL programming guide for the CUDA architecture.” http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/OpenCL_Programming_Guide.pdf.

[83] NVIDIA Corporation, “OpenCL best practices guide.” http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/OpenCL_Best_Practices_Guide.pdf.

[84] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Mathematics of computation*, Vol. 19, No. 90, pp. 297–301, 1965.

[85] 大浦拓哉, “FFT (高速フーリエ・コサイン・サイン変換) の概略と設計法.” http://www.kurims.kyoto-u.ac.jp/~ooura/fftman/index.html.

[86] 樋口龍雄, 川又政征, MATLAB 対応 ディジタル信号処理. 昭晃堂, 2000.

[87] “QueryPerformanceCounter function.” http://msdn.microsoft.com/en-us/library/ms644904(v=VS.85).aspx.

[88] “QueryPerformanceFrequency function.” http://msdn.microsoft.com/en-us/library/ms644905(v=VS.85).aspx.

[89] “CUDA STREAM benchmark.” https://github.com/nattoheaven/cuda_stream_benchmark.

[90] M. Frigo and S. G. Johnson, “The design and implementation of FFTW3,” *Proc. IEEE*, Vol. 93, No. 2, pp. 216–231, 2005.

[91] “HIOKI AC/DC パワーハイテスタ 3334.” https://www.hioki.co.jp/products/

product/power/402/.

[92] 吉澤徹（編）, 光三次元・産業への応用. アドコム・メディア株式会社, 2008.

[93] 吉澤徹（編）, 最新光三次元計測. 朝倉書店, 2006.

[94] K. Ikeuchi, T. Oishi, and J. Takamatsu, "Digital bayon temple -e-monumentalization of large-scale cultural-heritage objects-," *Proc. ASIAGRAPH*, Vol. 1, No. 2, pp. 99–106, 2007.

[95] 柴原琢磨, 青木孝文, 山華義博, 小野勝一, "位相限定相関法に基づく高精度ステレオビジョンシステムと金属部品検査への応用," 電子情報通信学会技術研究報告, Vol. 105, No. 147, pp. 73–78, 2005.

[96] 茄子川慈苑, 青木孝文, 新浩治, 南里卓也, "車載ステレオカメラを用いた障害物の検出と測距に関する検討," 情報科学技術フォーラム講演論文集, Vol. 8, No. 3, pp. 227–228, 2009.

[97] 伊藤康一, 三浦衛, 青木孝文, "車載ステレオカメラを用いた障害物検出とその性能評価," 第 13 回画像の理解・認識シンポジウム 論文集, pp. 498–503, 2010.

[98] A. Scheenstra, A. Ruifrok, and V. Remco, "A survey of 3D face recognition methods," *Audio- and Video-Based Biometric Person Authentication*, Vol. 3546, pp. 325–345, 2005.

[99] A. Hayasaka, K. Ito, T. Aoki, H. Nakajima, and K. Kobayashi, "A robust 3D face recognition algorithm using passive stereo vision," *IEICE Trans. Fundamentals*, Vol. 92, pp. 1047–1055, 2009.

[100] 伊藤康一, 早坂昭裕, 青木孝文, 中島寛, 小林孝次, "2 次元／ 3 次元顔認証アルゴリズムの組み合わせに関する検討," 暗号と情報セキュリティシンポジウム, No. 2F4-2, 2009.

[101] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab." http://www.vision.caltech.edu/bouguetj/calib_doc/.

[102] "FlyCapture SDK." http://www.ptgrey.com/products/pgrflycapture/index.asp.

[103] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of sim-

ple features," *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, Vol. 1, pp. 511–518, 2001.

[104] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," *Proc. Int'l Conf. Image Processing*, Vol. 1, pp. 900–903, 2002.

[105] "Open Computer Vision Library." `http://sourceforge.net/projects/opencvlibrary/`.

[106] "Point cloud library." `http://pointclouds.org/`.

[107] "Meshlab." `http://meshlab.sourceforge.net/`.

[108] M. Brown and D. G. Lowe, "Unsupervised 3D object recognition and reconstruction in unordered datasets," *Proc. Int'l Conf. 3-D Digital Imaging and Modeling*, pp. 56–63, 2005.

[109] N. Snavely, "Bundler: Structure from Motion (SfM) for Unordered Image Collections." `http://phototour.cs.washington.edu/bundler/`.

[110] C. Wu, "VisualSFM: A visual structure from motion system." `http://ccwu.me/vsfm/`.

[111] Autodesk, Inc., "Autodesk 123D Catch." `http://www.123dapp.com/catch`.

[112] "ARC 3D." `http://www.arc3d.be/`.

[113] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-views stereo reconstruction algorithms," *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pp. 519–528, 2006.

[114] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 27, No. 10, pp. 1615–1630, 2005.

[115] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int'l J. Computer Vision*, Vol. 47, No. 1, pp. 7–42, 2002.

[116] アイファイジャパン株式会社, "Eye-Fi カード." `http://www.eyefi.co.jp/`.

[117] 一般社団法人カメラ映像機器工業会, "Exif2.3 metadata for XMP." `http://www.`

cipa.jp/english/hyoujunka/kikaku/pdf/DC-010-2012_E.pdf.

[118] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, Vol. 24, No. 6, pp. 381–395, 1981.

[119] D. Nister, “An efficient solution to the five-point relative pose problem,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 26, No. 6, pp. 756–770, 2004.

[120] 岡谷貴之, “バンドルアジャストメント,” 研究報告コンピュータビジョンとイメージメディア, Vol. 167, No. 37, pp. 1–16, 2009.

[121] M. I. A. Lourakis and A. A. Argyros, “SBA: A software package for generic sparse bundle adjustment,” *ACM Trans. Mathematical Software*, Vol. 36, No. 1, pp. 1–30, 2009.

[122] KONICA MINOLTA, “3 次元スキャナ（3 次元デジタイザ）–製品情報.” http://www.konicaminolta.jp/instruments/products/3d/index.html.

[123] Z. Timo, J. Schmidt, and H. Niemann, “Point set registration with integrated scale estimation,” *Proc. Int’l Conf. Pattern Recognition and Image Processing*, pp. 116–119, 2005.

[124] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” *Proc. Int’l Conf. Computer Vision*, Vol. 1, pp. 666–673, 1999.

[125] S. Leutenegger, “BRISK: Binary Robust Invariant Scalable Keypoints.” http://www.asl.ethz.ch/people/lestefan/personal/BRISK.

[126] D. Lowe, “Demo Software: SIFT Keypoint Detector.” http://www.cs.ubc.ca/~lowe/keypoints/.

[127] J. M. Morel and G. Yu, “SIFT and ASIFT – online demo: try if your images match!.” http://www.cmap.polytechnique.fr/~yu/research/ASIFT/demo.html.

[128] S. Sakai, K. Ito, T. Aoki, and H. Unten, “Accurate and dense wide-baseline stereo matching using SW-POC,” *Proc. Asian Conf. Pattern Recognition*, pp. 335–339,

2011.

[129] S. N. Sinha, J. M. Frahm, M. Pollefeys, and Y. Genc, “Feature tracking and matching in video using programmable graphics hardware,” *Machine Vision and Applications*, Vol. 22, No. 1, pp. 207–217, 2011.

[130] W. Changchang, A. Sameer, C. Brian, and M. S. Steven, “Multicore bundle adjustment,” *Proc. Int’l Conf. Computer Vision and Pattern Recognition*, 2011.

[131] NVIDIA Corporation, “NVIDIA Tegra K1 によりスーパーコンピューティング・テクノロジを自動車に搭載.” http://www.nvidia.co.jp/object/tegra-k1-mobile-processor-power-automotive-car-20140105-jp.html.

謝辞

本論文は，筆者が東北大学 大学院情報科学研究科 情報基礎科学専攻 計算機構論分野（青木（孝）・本間（尚）研究室）において行った研究を取りまとめたものです．本研究を推し進めるにあたり，多くの方々からご助言とご協力をいただきました．

恩師 青木孝文教授には，研究に関する懇切なご指導と終始変わらぬ励ましをいただきました．特に，研究の途中成果に対する前向きな評価と，研究の遂行能力に関する高い信頼をいただいたことは，私が研究に取り組む上での大きな後押しとなりました．また，研究だけでなく，何事に対しても挑戦的かつ情熱的に取り組む先生の姿勢から，非常に多くのことを学ばせていただきました．学部 4 年次から博士後期課程修了まで，6 年間に渡り先生のもとで研究を行えたことは，私にとって大変な幸運であり，今後の人生の大きな糧になると確信しております．ここに改めて，深く感謝の意を表します．

本論文をまとめるにあたり，亀山充隆教授ならびに小林広明教授より，それぞれのご専門の立場から有意義なご意見を賜りました．深くお礼申し上げます．

本間尚文准教授には，学部 4 年次から研究全般に関する有意義なご意見をいただきました．深く感謝いたします．

滝沢寛之准教授には，GPU を用いた並列処理について，実装手法や性能評価に関するご討論とご支援を賜りました．本研究を推し進めるにあたり，その方向性を模索する際には，先生とのご議論が非常に参考となりました．心よりお礼申し上げます．

伊藤康一助教には，学部 4 年次より研究全般に関する直接的なご指導とご討論をいただきました．特に，研究に対する考え方や，文章作成や発表の技術など，研究の基礎について非常に多くの事柄を学ばせていただきました．ここに改めて深謝の意を示します．

NEC ソリューションイノベータの札野欽也氏には，画像対応付けアルゴリズムの GPU 実装に関するご討論をいただきました．本論文に記載した位相限定相関法に基づく画像対応付けアルゴリズムの GPU 実装は，同氏と共同で行ったものです．心より感謝申し上げます．

日本電信電話株式会社の土田勝博士には，画像対応付けアルゴリズムの GPU 実装の応用として，マルチバンドカメラを用いた色再現という非常に興味深いトピックについてご教授いただきました．深くお礼申し上げます．

青木（孝）・本間（尚）研究室の先輩である宮澤一之博士（現 三菱電機株式会社）には，位相限定相関法の基礎原理，および同手法に基づく 3 次元計測についてご指導を頂きました．また，高橋徹博士（現 日本電気株式会社）には，プロジェクタ・カメラシステムを用いた投影画像の幾何補正ついてご教授いただきました．加えて，お二人に研究に関するご指導をいただいたことは，私が博士後期課程に進学するきっかけとなりました．ここに改めて，感謝の意を示します．

酒井修二氏，石井純平氏（現 東北電力株式会社），山尾創輔氏には，多視点画像からの 3 次元計測について，有益なご議論とご支援をいただきました．本論文に記載したデジタルカメラの移動撮影に基づく 3 次元計測システムの開発は，諸氏と共同で行ったものです．心より感謝いたします．

和泉圭祐氏（現 トヨタ自動車株式会社）には，車載ステレオカメラを用いた障害物検出についてご議論をいただきました．また，同氏の研究に対する精力的な姿勢に感化され，研究に取り組む意欲を維持することができました．ここに感謝の意を示します．

青山章一郎氏，遠藤翔氏，佐藤拓杜氏，川島直揮氏，野呂和正氏，草薙大地氏，丸木大樹氏をはじめとする青木（孝）・本間（尚）研究室の諸氏には，本研究に関するさまざまなご意見やご協力をいただきました．心より感謝いたします．

このように，本研究を行うにあたり，多くの方々からご助言とご協力をいただきました．私一人の力では，このような形で研究を取りまとめ，論文として著すことは到底できませんでした．諸氏に重ねて厚くお礼申し上げます．

2015 年 1 月 20 日