Vol.2011-DPS-149 No.7
2011/11/25

SumiTag：あまり目立たないARマーカーとGPGPUを利用した読み取り方法

柴田直樹[†1]　山本眞也[†2]

本稿では，主に屋内位置推定に使用することを目的として，新しいARマーカとGPGPUを利用した読み取り手法を提案する．提案するマーカは，形状と配色を工夫し，室内に配置したときの違和感が少ないことを特徴とする．近年，スマートフォンなどの情報機器には高解像度のカメラやGPUが搭載されている．屋内での正確な位置推定のために，カメラとARマーカが利用できるが，既存のARマーカは形状や配色の制限から目立ちやすい欠点があった．また，遠くにあるマーカを識別するためには，高性能なカメラにより撮影した高解像度の画像を解析する必要があるが，CPUの処理能力と消費電力の制限があるため，なるべく効率よく解析する手法が求められる．本稿では，GPGPUを利用することで，高速に高解像度の画像からマーカを識別するための手法を提案する．既存のARマーカと比較して，より広い範囲で識別できることを評価実験を通して確認した．

SumiTag : Inconspicuous Fiducial Marker and GPGPU-Assisted Tracking Method

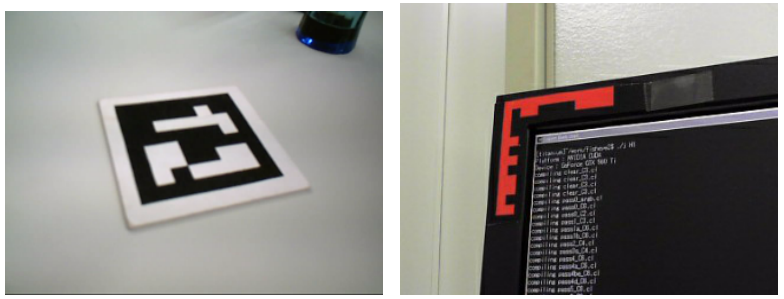
Naoki Shibata[†1] and Shinya Yamamoto[†2]

In this paper, we propose a new fiducial marker for indoor positioning. In recent years, smart phones have built-in high resolution cameras and GPUs which can be used for indoor positioning utilizing fiducial markers. However, existing fiducial markers are too conspicuous especially if we put many of them on the walls. In this paper, we propose a new fiducial marker that is not too conspicuous because of its shape and freedom of coloring. We also propose a GPGPU-assisted tracking method for the proposed marker. We need to recognize markers from high resolution images in order to make accurate measurement, but that is too burdensome for small CPUs on mobile devices. The proposed tracking method utilizes GPGPU so that the recognition can be performed quickly and efficiently. We confirmed that the proposed tracking method can recognize the proposed marker from wider angle and longer distance compared to existing methods.

1. はじめに

GPS受信機が安価になり，様々な情報機器に搭載されている．GPSにより屋外での位置推定が可能となるが，屋内での位置推定技術はまだ決定版といえるものがなく，研究が続けられている．一方，スマートフォンなどの情報機器は年々高性能になり，高解像度のカメラとGPUが搭載されるようになった．カメラとARマーカ[1)–3)]を利用することで，屋内での位置推定が可能であり，特にカメラが高解像度化したことにより，この方式は精度の点で有望と言える．本稿では，主に屋内位置推定に使用することを目的として，新しいARマーカとGPGPUを利用した読み取り手法を提案する．提案するマーカは，形状と配色を工夫し，室内に配置したときの違和感が少ないことを特徴とする．

遠くのマーカの位置の及び方向の推定精度は，カメラに写ったマーカのピクセル数に依存する．特に，マーカの方向を推定するには，縦横両方向にある程度のピクセル数で写っている必要がある．この条件を満たし，遠くの位置から正確に位置推定するためには，マーカ



(a) 縁のある既存のマーカ　(b) 提案マーカをディスプレイの隅に貼りつけた例

図1　マーカの例

†1 滋賀大学
Shiga University
†2 山口東京理科大学
Tokyo University of Science, Yamaguchi

の縦横の長さを長くすることと，高い解像度で撮影した画像からマーカを認識することの２点が重要になる．既存のマーカーのほとんどが正方形またはそれに近い形状をしており，マーカを印刷する色はほとんどの場合，白と黒が指定されている．また，クワイエットゾーンと呼ばれる，通常は白い縁取りが必要となる (図 1(a))．クワイエットゾーンを含めると，マーカの面積が大きくなってしまい，室内にマーカを配置すると目立ってしまう．提案するマーカは，形状をブーメラン型とし，またマーカの配色に関しても，コントラストがはっきりしている任意の二色をユーザが選ぶことができる．形状をブーメラン型とすることの長所として，ディスプレイの額の隅などに貼ることができ (図 1(b))，面積の割に縦横方向の長さを長くとれることが挙げられる．また，クワイエットゾーンの色を，貼り付ける場所のもとの色とすることができ，この場合，見た目のマーカの大きさは実質的にクワイエットゾーンを除いた部分となるため，マーカの見た目の大きさを小さくできる．また，位置推定など，各種アプリケーションに利用するためには，少なくとも 10 ビット程度のペイロードを埋め込むことができると便利である．提案するマーカには，16 ビットのペイロードと 3 ビットのパリティビットを埋め込むことができる．

最近のスマートフォンには高解像度のカメラが搭載されており，遠くにあるマーカの位置を高い精度で推定するためには，この高解像度の画像を解析できれば良いが，そのためには大きな処理能力が必要となる．GPGPU は画像認識のような用途に特に適しており高速・高効率でこのような処理を実行可能である．また，数年以内にスマートフォンでも OpenCL による GPGPU プログラムの実行がサポートされる見込みである．その一方で，GPGPU 上で実行されるプログラムは高度に並列化された形で作成する必要があり，従来 CPU 用に用いられてきたプログラムをそのままの形で用いることはできない．本稿では，提案するマーカの認識処理の大部分を OpenCL で書かれた GPGPU プログラムで効率よく行う手法を提案する．

提案する読み取り手法の性能を評価するため，識別が可能なカメラとマーカの距離，角度，マーカの色を変化させたときに識別が可能かどうかについて既存手法と比較を行い，提案する読み取り手法が既存の手法と同等かそれ以上の範囲からマーカを識別可能であることを確認した．

2. 関 連 研 究

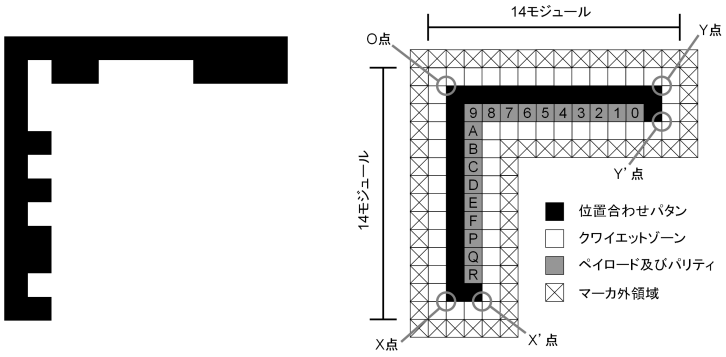
これまでに，いくつかの AR マーカシステムが提案されている[1)–3)].ARToolKit[1)] は，最も良く知られている AR マーカシステムである．異なるマーカを識別するためのペイロード部のデザインを自由に決められることに特徴がある．ペイロード部のデザインは 16 × 16 の画像を 4 方向分サンプリングし，テキストデータに変換したものを用いて指定する．ARToolKit のアルゴリズムは，周りの黒い縁を入力映像から検出し，その法線・外積から X 軸，Y 軸，Z 軸を算出する．これら既知の値から変換行列を生成し，回転・拡大縮小に関する値を算出した上で，登録されたペイロード部のデザインとのマッチングを行う．ARToolKit は，公式サイト[4)] にてライブラリを公開しており，位置推定に特化するよう拡張したもの[5)] をはじめとして各プラットフォームへ移植したものなどが第三者により提供されている．

ARTag[2)] は，ARToolKit を改良し，誤検知率，誤認識率，不均一なライティングへの耐性，ジッター率，検知スピードなどについて精度を向上させた AR マーカである．ペイロード部には誤り訂正符号を含む 36bit の情報を持ち，ノイズ耐性を持つモザイク模様の 2002 個のデザインを用意している．公式サイト[6)] でデモムービーが公開されている．

reacTIVision[3)] は，reacTable のコンポーネントの一部として開発された．特徴としては，形，フットプリントサイズ，中心点，回転角のセットから遺伝的アルゴリズムを用いて生成された独自のアメーバ模様の幾何学デザインを AR マーカとして用いる．デザインは 90 シンボルを標準パタンとして用いることができる．300 パタンの追加シンボルや classic, d-touch と呼ばれる既存パタンについてもサポートしている．さらにアメーバ模様の最小単位をフィンガートラッキング用途に用いることができる．各シンボルは木構造に変換されて管理される．マーカの識別はこの木構造を用いる．マーカの検出は，入力映像を白黒に変換し，木構造探索によりマーカ探索を行う．このため，マーカを構成するデザインを示す色とそれ以外の中抜き色のコントラスト比が高ければ，色に関係なく検出できる．また，木構造探索により，ある程度のノイズを許容する．また，公式サイト[7)] にてデモアプリケーションが公開されている．

AR マーカには様々な応用が考えられる．以降，AR マーカの応用の研究として文献 8)–10) について述べる．文献 8) は，ビデオストリームの管理をするためのユーザインターフェースに ARToolKit を利用している．この手法では，AR マーカを各面に張り付けた箱の操作でビデオストリーミングの各設定を操作する．AR マーカを張り付けた箱をつまみのように操作したり，移動させたりすることでビデオストリーミングにおけるプレイアウトやリサイズなどの各操作を実現している．これにより，帯域変動やパケットロスに即座に対応でき，ピア・ツー・ピア・テレカンファレンス，ビデオ・オン・デマンド，ビデオミキシング，ビデオ監視のような用途に用いることができる．

Vol.2011-DPS-149 No.7
2011/11/25



(a) 提案マーカの例　　(b) 提案マーカの構造

図 2　提案マーカ

文献 9) は，多数の収納箱に物が収納されている棚において，ARToolKit を位置推定用に改良した QPToolKit[5] と写真を利用し，収納箱の識別番号と写真と関連づけて保存することで，探し物を効率的に探すための物探し支援システムを提案している．

文献 10) は，DLNA を利用したネットワーク経由の家電操作において，複数の家電がある場合に，目視している家電をコントローラの家電一覧から検索することが煩雑である問題を解決するために，Web カメラで家電を撮影した際に対象となる家電のリモコンを AR インタフェースで表示する方法を採用している．この研究では，家電の識別に可視光マーカを採用している．可視光マーカを採用した理由として，家電に一般的な画像 AR マーカを張り付けることは見かけ上よくないと提言している．

3. 提案するマーカ

本章では，提案するマーカの仕様について述べる．図 2(a) に，提案するマーカの 1 例を示す．また，図 2(b) に，マーカの構造を示す．各マーカは，モジュールを図 2(b) 中で示されるように並べることで構成される．ここで，モジュールとはマーカを構成する正方形の単位セルのことで，1 モジュールで 1 ビットを表現することができる．また，モジュールの大きさは規定しないものとする．ユーザは任意の色 S と色 T を選択し，クワイエットゾーンを色 S で，位置合わせパタンを色 T で塗るとする．読み取り精度を向上させるためには，色 S と色 T はコントラストのはっきりした色の組合せである必要がある．ペイロード及びパリティ部の各モジュールは，色 S または色 T で塗られている必要があり，色 S で塗られたとき 1 を，色 T で塗られたときに 0 を表すとする．図中のモジュール P,Q,R がパリティビットを表し，モジュール P はモジュール 0,3,6,9,C,F の，モジュール Q はモジュール 1,4,7,A,D の，モジュール R はモジュール 2,5,8,B,E のそれぞれ奇数パリティである必要がある．ペイロードは，モジュール 0 から F までを 2 進数の各桁とみなし，最下位ビットから順に並べてできる数と 21845(16 進数で 5555) の排他的論理和で表される．

4. GPGPU を利用した読み取り方法

本章では，3 章で述べたマーカの，GPGPU を利用した読み取り方法について説明する．

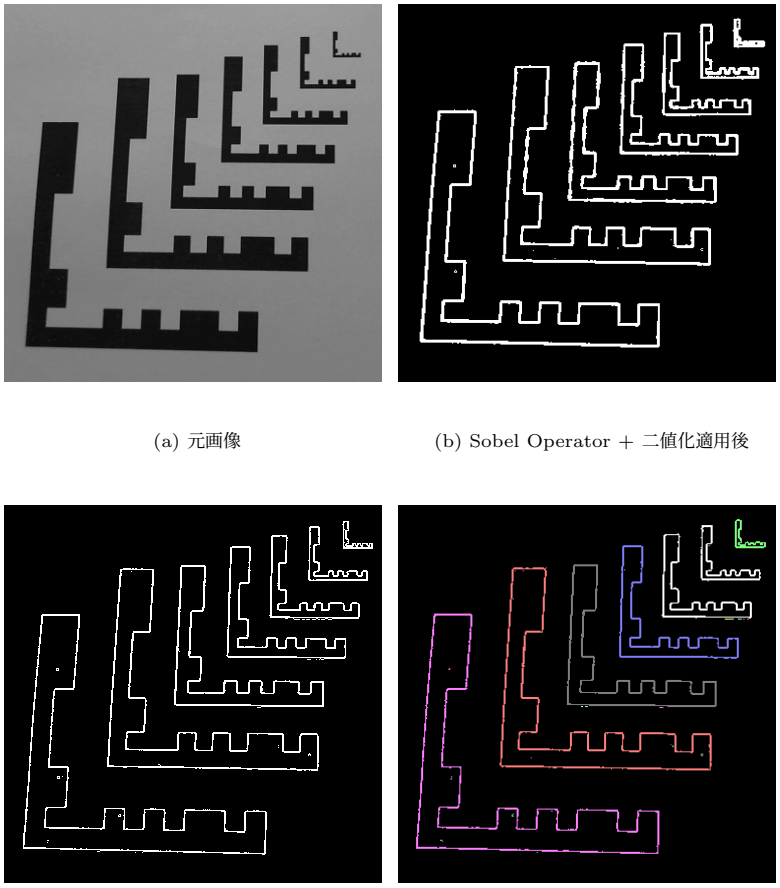
提案手法は，カメラで撮ったフレームを入力とし，フレーム内のマーカ全ての三次元座標とペイロードの値を出力する．本稿では，過去に認識したフレームの情報を利用することで認識精度等を向上させる処理は一切しておらず，各フレームを独立に認識する．提案手法は，前半の，GPU を利用して画像処理を行う GPU 処理部と，後半の，CPU により座標変換・読み取りを行う CPU 処理部に分かれる．

以下，GPGPU とそのプログラムに求められる性質について簡単に説明した後，提案手法の各部分について順番に説明する．また，最後に既存の読み取り手法との比較を行う．

4.1 GPGPU 用プログラムに求められる性質

GPU は従来から主に 3D 画像の描画に用いられてきたが，近年表現力の向上のためにプログラマブルシェーダが搭載されるようになった．これにより，従来の固定機能の描画パイプラインと異なり，描画手順をプログラムにより指定できるようになった．このプログラム実行のための新たな GPU の機能を一般的な処理のために利用する枠組みが GPGPU である．GPU は複数の ALU を含むスレッド実行ブロックを複数持ち，各ブロックは，異なるデータ入力に対し同じ命令列を一度に並列に実行する．命令列に条件分岐が含まれる場合は，両方のパスを実行して不要な結果を捨てるという処理が行われるため，一つのブロックで実行される一連の処理フローが全ての入力データに対してほぼ同一でないとパフォーマンスが悪化する．GPGPU を利用して高いパフォーマンスを得るためには，多数の独立した入力データに対し互いに依存しない形でほぼ同じフローの処理を実行して正しい処理結果が得られるようにアルゴリズムを設計する必要があり，CPU 向けに開発されたアルゴリズムをこのような形に変更するのは自明ではない．

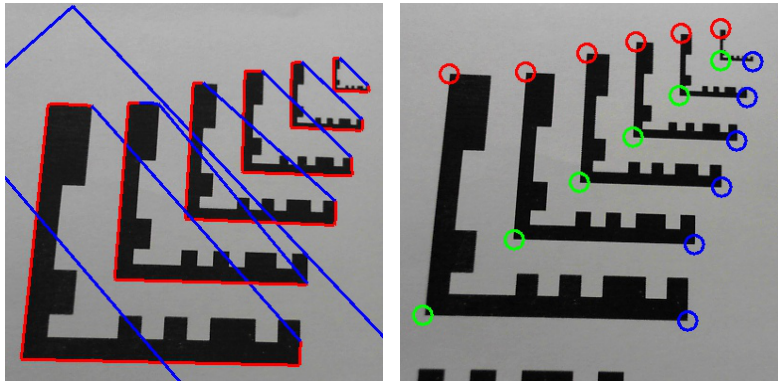
Vol.2011-DPS-149 No.7
2011/11/25



(a) 元画像　(b) Sobel Operator + 二値化適用後

(c) Sobel Operator + Edge Thinning + 二値化適用後　(d) 連結成分抽出後

図 3　各ステップ適用前後の画像



(a) 代表 8 点抽出後　(b) 頂点 O, X, Y 抽出後

図 4　各ステップ適用前後の画像 (続き)

4.2 GPU 処理部

GPU 処理部では，フレームから，マーカである可能性が高いピクセルの集合を抜き出し，それぞれについて代表 8 点の二次元座標を出力する．ここで代表 8 点の座標とは，該当するピクセルの集合について，それらのうちもっとも上，右上，右，右下，下，左下，左，左上にあるそれぞれのピクセルの座標である．ピクセルの集合がマーカであった場合，代表 8 点は，図 2(b) における O, X, X', Y, Y' 点となる．また，代表 8 点を頂点とする多角形の各辺に対して，X' と Y' を結ぶ辺がどれかを調べる．GPU 処理部は 5 つのステップからなる．それぞれのステップの内容を以下に示す．

ステップ G1　入力画像に対し，輪郭抽出と二値化を行う

ステップ G2　上の結果に対し連結成分抽出を行う．

ステップ G3　各連結成分の縦横サイズをカウントし，いずれかの長さが 15 ピクセル以下のものを除外する．

ステップ G4　残った各連結成分の代表 8 点の，二次元座標を計算する．

ステップ G5　各連結成分の代表 8 点を結んでできる八角形の各辺の中点付近に，連結成分に属するピクセルがあるか調べる．これにより，X' と Y' を結ぶ辺がどれかを調べる．

以下，それぞれのステップの詳細について述べる．

ステップ G1 元画像 (図 3(a)) に対し，輪郭抽出手法の一種である Sobel Operator を適用する (図 3(b))．このままだと，抽出された輪郭が太すぎるので，この結果に対し，Edge Thinning を適用する (図 3(c))．その後，二値化処理を適用する．

ARToolKit[1] では，最初に入力画像を二値化した後に認識処理を行っており，二値化する際の閾値の選択がうまくいかないと認識精度の低下につながる．提案手法では，最初に輪郭抽出を行い，その後二値化処理を行うことで，この問題を回避している．輪郭抽出の結果をそのまま二値化すると，抽出した輪郭が太くなりがちであり，遠くのマーカを識別しようとした際に周りの物体と連結してしまう．これを防ぐため，Edge Thinning を適用する．

ステップ G2 ステップ G1 の出力に対し，4 連結成分抽出処理[⋆1]を行う．本稿では，GPGPU の処理に適した連結成分抽出アルゴリズムを新たに開発した．このアルゴリズムについての詳細は付録 A.1 で述べる．図 3(d) では，連結成分に割り当てられた値の下位数ビットが表示された色に対応している．

ステップ G3 ステップ G2 で得られた各連結成分に対し，縦横いずれかの長さが 15 ピクセル以下のものを除外する．この処理は，2 つのフェーズからなる．

フェーズ 1 では，各連結成分に対し，その x 座標および y 座標の最小値，最大値を計算する．これは，ステップ G2 の結果より，各ピクセルの座標からそのピクセルの属する連結成分の ID を得ることができるので，グローバルメモリ上の ID に対応するアドレスに確保された変数と各ピクセルの x 座標および y 座標に対し， atomic_min および atomic_max[⋆2]を適用することで，実現できる．

フェーズ 2 では，次に，各連結成分に対し，上で計算した x 座標および y 座標の最大値から最小値を引き，15 ピクセル以下なら以降処理から外す．

ステップ G4 代表 8 点の，フレーム上での二次元座標を計算する．これは，ステップ G3 のフェーズ 1 と同様の方法で実現できる．

ステップ G5 各連結成分の代表 8 点を結んでできる八角形の各辺に対して，その中点付近に，連結成分に属するピクセルがあるか調べる (図 4(a))．代表 8 点を結んでできる八角形は，図 2(b) における O, X, X', Y, Y' 点を結んでできる五角形に近い形状である．この中で，X' と Y' を結ぶ辺がどれであるかこのステップで調べる．このために，八角形の各辺の中点から数ピクセル以内に同じ連結成分があるかどうか調べる．図 4(a) の青い辺は，その中点付近に連結成分に属するピクセルがないと判定された辺である．

なお，マーカが完全な平面上になく，わずかにたわんだ紙の上に印刷されている場合，辺 OX や辺 OY も $X'Y'$ と同様に中点付近に連結成分に属するピクセルがないと判定されるケースがある．これは，中点と連結成分に属するピクセル間の距離と，連結成分全体の縦横サイズの比を考慮して，対処を行っている．

4.3 CPU 処理部

CPU 処理部では，画像処理部の出力した代表 8 点の座標で表される連結成分について，後述のいくつかの基準を用いてマーカかどうか判別し，マーカであればその三次元位置を求め，ペイロードを読み取り，出力する．CPU 処理部の各ステップは，下記の通りである．

ステップ C1 マーカの頂点 O, X, Y の二次元座標を求める
ステップ C2 マーカの形状でフィルタリング
ステップ C3 マーカの頂点 O, X, Y の三次元座標を求める
ステップ C4 ペイロードを読み取る

以下，それぞれのステップの詳細について述べる．

ステップ C1 このステップでは，図 2(b) に示したマーカの頂点 O, X, Y の，二次元画像内の座標を求める．4.2 節の GPU 処理部のステップ G5 の結果より，図の頂点 X' と Y' を結ぶ辺がどれか既に判明している．ここで，マーカが遠方にあり，画像内でのマーカのサイズが小さいケースでは，画像の上で X と X'，Y と Y' の区別がつきにくい．このような場合でも，GPU 処理部のステップ G5 の結果は安定していることが，経験的に分かっている．提案手法では，この X' と Y' を結ぶ辺の位置と向きから代表 8 点より X と Y を選ぶ．

ステップ C2 この後のステップ C3 は処理に多少時間がかかるので，ステップ C3 に進む前に明らかにマーカでない連結成分を除外する．具体的には，まず O, X, X', Y', Y を結ぶ辺の長さの比を計算し，通常のマーカではありえない比になる場合は以降の処理から除外する．次に，各頂点付近のピクセルを読み出し，同様に通常のマーカではありえないパタンになるものを除外する．

ステップ C3 このステップでは，O, X, Y の二次元座標から，マーカの三次元座標を求める．図 5 に示すように，画像とカメラの画角から，仮想的に視点と画像の三次元座標と向きを定め，視点と，画像上の O, X, Y をそれぞれ結ぶ線上で，X と O の三次元座標間の距離と Y と O の三次元座標間の距離が 1 になり，三次元座標における各 XOY が直角になるという性質を満たす O, X, Y の座標を，ニュートン法と 2 分法を併用して求める．この条件で三点の三次元座標を求めると，複数の解が求まるが，ステップ C2 と同様に各頂点付近

⋆1 画像の連続した領域に対して各領域固有の整数値を割り当てる処理
⋆2 アトミック演算は，メモリから値を読み出し，その値に対して演算を行い，結果をメモリに書き戻す一連の操作を不可分の操作として行う

Vol.2011-DPS-149 No.7
2011/11/25

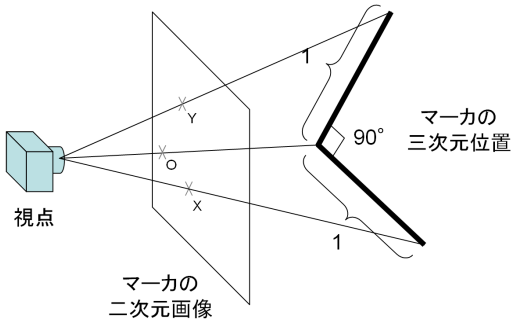


図 5 画像上の二次元座標からマーカの三次元座標を求める

のピクセルを読み出し，通常のマーカではありえないパタンになるものを除外する．

ステップ C4 ステップ C3 の結果から，二次元画像上での各モジュールの位置が分かるので，ペイロード・パリティを読み出す．

5. 評　　価

3 章で述べたマーカと 4 章で述べた読み取り法に関し，識別性能を評価するため，識別が可能なカメラとマーカの間の距離，識別可能なカメラの向きとマーカの法線のなす角度について調べた．また，マーカの色を変化させたときに識別が可能かどうかについて調査した．以下では，まず実験環境について述べ，次に評価結果について順に述べる．

5.1 評 価 環 境

4 章で述べた手法を，Java と OpenCL(JOCL) を用いて実装した．カメラによる動画のリアルタイム撮影のために，OpenCV を使用した．Java のメソッドから OpenCV の API を呼び出すために，JNI を利用した．画像などを高速に画面に表示するために，OpenGL(JOGL) を用いたクラスを作成し，使用した．実験には，Web カメラとして Microsoft LifeCamStudio Q2F-00008 を使用した．カメラはオートフォーカス機能を持ち，また，実験ではカメラを固定して使用した．カメラの解像度として，1280×720 ピクセルを使用した．特に記述がない場合，マーカはカメラの正面に設置した．原稿執筆時に実装が入手可能な ARToolKit と reacTIVision との比較を行った．また，各 AR マーカシステムは入力映像の解像度以外の設定はデフォルトのまま用いた．ARToolKit に関しては，サンプルの simpleTest.exe を使用して比較実験を行った．実験では，ARToolKit と reacTIVision については，白紙の A4

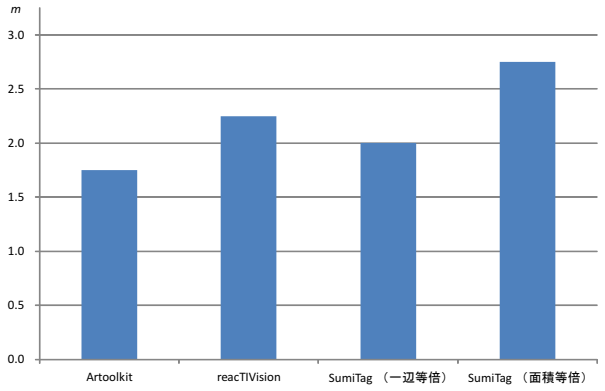


図 6 マーカを識別可能な距離

サイズ用紙の中央に一辺 5 cm のマーカを印刷したものを用いた．SumiTag は，他のマーカを大きく形状が異なるため，XO 及び YO の長さが 5 cm の場合と，クワイエットゾーンを除いた部分の面積が他のマーカと同じになる場合のそれぞれについて，実験を行った．

5.2 識別が可能な距離

カメラからマーカまでの距離を変化させ，どの距離まで認識が可能か評価を行った．

結果を図 6 に示す．縦軸はカメラからマーカまでの距離を示す．距離の単位は m である．図より，同面積で比較したとき，SumiTag は他の比較したマーカよりも遠くから識別できていることが分かる．また，SumiTag では，カメラとマーカの間が $3m$ 以上離れている場合においても，マーカを発見することはできており，ペイロードの抽出アルゴリズムを改良することができれば，さらなる性能の向上を望める．

5.3 識別が可能な角度

識別が可能な，カメラの向きとマーカの法線のなす角度について評価を行った．カメラの位置をマーカから正面から撮影したときを 90°，マーカ接地面と並行のとき 0° としたとき，30° ，45° ，60°，90° のそれぞれの角度について，また，マーカを時計回りに 45° ずつ回転させ，それぞれの組み合わせについてマーカを識別できるか調査した．なお，SumiTag についてはマーカが L 字に見える状態をマーカ回転角 0° とする．マーカ対カメラの距離は全て $1.5m$ とした．

結果を表 1 に示す．表における ◯ は完全に識別した場合，△ はマーカを発見できたが

表 1 カメラの角度と識別の可否

手法	カメラ位置（°）	マーカ回転角（°）							
		0	45	90	135	180	225	270	315
ARToolKit	30	△	△	△	△	△	△	△	△
	45	△	△	△	△	△	△	△	△
	60	△	△	△	△	△	△	△	△
	90	△	△	△	△	△	△	△	△
reacTIVision	30	×	×	×	×	×	×	×	×
	45	○	○	○	○	○	○	○	○
	60	○	○	○	○	○	○	○	○
	90	○	○	○	○	○	○	○	○
SumiTag（一辺等倍）	30	△	△	△	△	△	△	△	△
	45	△	△	△	△	△	△	△	△
	60	○	△	○	△	○	△	○	△
	90	○	○	○	○	○	○	○	○
SumiTag（面積等倍）	30	○	△	○	△	○	△	○	△
	45	○	△	○	△	○	△	○	△
	60	○	○	○	○	○	○	○	○
	90	○	○	○	○	○	○	○	○

正しく識別できなかった場合，× はマーカを発見できなかった場合を示す．ARToolKit はカメラがマーカより上にあるのか下にあるのかを正しく判定できていない．すなわち，正面以外の場所からマーカを識別しようとした場合，入力映像は二次元のため，正方形のマーカは台形として入力される．このとき，マーカの並行した辺がどのような角度を持つのかを正しく識別できていないため，正確に識別できているとは言えない．また，ARToolKit はペイロード部のデザインを自由に登録できるが，登録しているデザインの類似性および登録データが精度に大きな悪影響を及ぼす．今回は，配布されているサンプルプログラムを実行しており，デザイン数が 1 種類であるためこの影響はないが，実際に使用する際にはこの点について考慮する必要がある．reacTIVision は，極端に角度がついている場合を除き，識別できた．しかし，強い光をあてた場合に識別率が悪化する傾向があり，しばしば蛍光灯の光でマーカを見失うことがあった．SumiTag は，特にマーカが ∧ 字や ∨ 字に見える状態でペイロードを正しく識別できないことがあった．

5.4 マーカの配色と識別の可否

通常，AR マーカは白地に黒で印刷される．しかし，マーカの設置の仕方によっては，この配色では目立ってしまうので，様々な色でマーカを印刷できた方が便利である．ここでは，マーカと背景の色を変化させ，識別に及ぼす影響について調査した．結果を表 2 および 表 3，4，5 に示す．

表 2 色付きマーカにおける識別の可否

背景色	マーカ色	手法		
		ARToolKit	reacTIVision	SumiTag
白	黒	○	○	○
	赤	○	○	○
	青	○	○	○
	緑	○	○	○
	黄	×	×	×
黒	白	×	○	○
	赤	×	○	○
	青	×	○	○
	緑	×	○	○
	黄	×	○	○

表 2 は背景および中抜きを白・黒としたとき，マーカの色を黒・白・赤・青・緑・黄の 6 パタンについて調査したものである．ただし，背景とマーカが同色のものは除く．また，表 3，表 4，表 5 は，表 2 において認識できたパタンについて，様々な環境で使用した場合の結果である．

表 3 より，背景色と中抜き色が大きく違う場合，ARToolKit はマーカを識別できないことが分かる．窓ガラス（駐車場）で識別できた理由として，反射してガラスに映った色が偶然中抜き色と似ていたことが考えられる．これを考慮し，ARToolKit を実空間で使うには，中抜き色と同じ背景色をさらにマーカの一部として扱い外枠に用意する必要があり，マーカは一回大きくなる．また，物の影や本棚の隙間などをマーカと誤認識してしまうケースも多くみられた．表 4 より，reacTIVision は中抜き色とマーカ色の組み合わせを変更することで多くの環境に適応できる．しかし，5.3 節で述べたように，強い光の下で識別率が悪化する傾向があり，蛍光灯や太陽光でマーカを見失ってしまうことがある．また，中抜き色とマーカ色のコントラストでマーカを発見するため，逆光には対応できない．表 5 より，SumiTag は全般的に対応できることがわかる．特に中抜き色を必要とせず，形状だけでマーカを発見することができるために，逆光でもマーカを識別することができることは強みの一つである．これは，警報ランプやプロジェクタ光などの色光によってマーカが変色した際でも識別できることを示す．また，ディスプレイ端や戸棚のサッシ部分では，十分に識別できるサイズで枠部分に収めることができるため，比較的目立たないよう設置することが可能である．

表 3 実環境における色付きマーカの識別の可否（ARToolKit）

中抜き色	白				
マーカ色	黒	赤	青	緑	黄
ディスプレイ端（ピアノブラック）	×	×	×	×	×
机上（木目調）	○	×	×	×	×
ガラス戸棚 サッシ（クリーム）	○	○	○	○	×
ガラス戸棚 ガラス（本）	×	×	×	×	×
窓ガラス（逆光）	×	×	×	×	×
窓ガラス（駐車場）	○	○	○	○	×
コルクボード	○	×	×	×	×

表 4 実環境における色付きマーカの識別の可否（reacTIVision）

中抜き色	白					黒				
マーカ色	黒	赤	青	緑	黄	白	赤	青	緑	黄
ディスプレイ端（ピアノブラック）	○	○	○	○	×	○	×	×	○	○
机上（木目調）	○	○	○	×	×	○	×	○	○	○
ガラス戸棚 サッシ（クリーム）	○	○	○	○	×	○	×	×	○	○
ガラス戸棚 ガラス（本）	○	○	○	○	×	○	×	×	○	○
窓ガラス（逆光）	×	×	×	×	×	×	×	×	×	×
窓ガラス（駐車場）	○	○	○	○	×	○	×	×	×	○
コルクボード	○	○	○	○	×	○	○	○	○	○

表 5 実環境における色付きマーカの識別の可否（SumiTag）

中抜き色	なし				
マーカ色	黒	赤	青	緑	黄
ディスプレイ端（ピアノブラック）	×	○	○	○	○
机上（木目調）	○	×	×	○	○
ガラス戸棚 サッシ（クリーム）	○	○	○	○	×
ガラス戸棚 ガラス（本）	○	○	○	×	×
窓ガラス（逆光）	○	○	○	○	○
窓ガラス（駐車場）	○	○	○	○	×
コルクボード	○	×	×	○	○

6. おわりに

形状と配色を工夫し，室内に配置したときの違和感が少ないことを特徴とする新しい AR マーカと GPGPU を利用した読み取り手法を提案した．既存の AR マーカと比較して，より広い範囲で識別できることを評価実験を通して確認した．今後，評価項目を増やし，位置推定の誤差と，画像認識の処理時間について他の手法との詳細な比較を行うことを予定している．また，ペイロードの読み取りを工夫し，遠距離からの読み取り精度を改善したい．今回提案した手法では，途中で画像の二値化を行っているが，今後は二値化を行わないで，サブピクセル単位でマーカの座標を求めることで，さらなる精度の向上を実現する手法の開発に取り組む予定である．

参 考 文 献

1) Kato, H. and Billinghurst, M. : “Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System, ” Proc. of 2nd IEEE and ACM International Workshop on Augmented Reality(IWAR ’99), pp.85-94, 1999.
2) Fiala, M. : “ARTag, a fiducial marker system using digital techniques,” Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol.2, pp. 590-596 vol.2, 20-25, 2005.
3) Kaltenbrunner, M. and Bencina R. : “reacTIVision: a computer-vision framework for table-based tangible interaction,” Proc. of the 1st international conference on tangible and embedded interaction (TEI ’07), 2007.
4) ARToolKit : http://www.hitl.washington.edu/artoolkit/.
5) 工学ナビ : QPToolkit: middleware for detecting ARToolKit markers : http://kougaku-navi.net/QPToolkit/.
6) ARTag : http://www.artag.net/.
7) reacTIVision : http://reactivision.sourceforge.net/.
8) Ferretti, S., Roccetti, M. and Strozzi, F. : “On Developing Tangible Interfaces for Video Streaming Control: a Real Case Study,” Proc. of 18th Int’l workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV ’08), pp.51-57, 2008.
9) Komatsuzaki, M., Tsukada, K. and Siio, I.: “DrawerFinder: Finding Items in Storage Boxes using Pictures and Visual Markers,” Proc. of the 16th int.l conference on Intelligent user interfaces (IUI 2011), pp.363-366, 2011.
10) 島田 秀輝, 坂本 直弥, 岡田 昌和, 綾木 良太, 佐藤 健哉 : EVANS2:拡張現実感技術を利用した家電機器操作システム, マルチメディア，分散，協調とモバイル (DICOMO2011) シンポジウム, pp.1638-1645, 2011.
11) 鈴木 賢治, 堀場 勇夫, 杉江 昇 : 逐次局所処理をともなう正方向および逆方向ラスタ走査による高速ラベル付け手法, 情報処理学会論文誌 41(11), pp. 3070-3081, 2000.
12) Hawick, K., Leist, A. and Playne, D. : “Parallel graph component labelling with GPUs and CUDA, ” Parallel Comput. 36, 12, pp. 655-678, 2010.

Vol.2011-DPS-149 No.7
2011/11/25

付　　録

A.1 GPGPU での実行に適した連結成分抽出アルゴリズム

連結成分の抽出は，パタン認識の基本的な手法の一つであり，様々なアルゴリズムが発表されている[11),12)]．文献 12) 内の Kernel_D の手法は，文献 11) の手法をほぼそのまま GPU で実行するものであるが，文献 11) の手法は必ずしも GPGPU 向けの手法ではないため，入力データによっては GPGPU での処理に不利な場合が生ずる[*1]．本稿では，この問題を解決した，GPGPU 上での実行に適した連結成分抽出アルゴリズムを提案する．本手法では，フレームバッファの各ピクセルに対し，カーネルによる操作を繰り返し並列に適用する．この際，カーネルは定数ステップ以内に動作を終了し，また全ての配列の要素に対し並列にカーネルの操作を適用したとしても正しく動作するようにアルゴリズムが設計されている．

図 7 に，提案手法の OpenCL による実装を示す．入力として，フレームバッファのポインタ fb をとり，このフレームバッファの座標 (x, y) におけるピクセルは fb$[x + y * 2048]$ に格納されているとする．また，入力画像は二値化されており，各ピクセルは int 型の値をとり，背景色を 0，前景色をそれ以外の値とする．また，簡単のため，フレームバッファの最外周には，いずれも背景色が格納されているとする．図 7 のカーネル ccl_prepare は，フレームバッファの最外周を除く全ピクセルに対して最初に一度適用され，各ピクセルに対し，そのピクセルが背景色であれば 0 に，前景色であれば配列の添え字に初期化する．以降，フレームバッファの各ピクセルには，そのピクセルから探索した範囲内で，最も添え字の値が小さい，同じ連結成分に属するピクセルの添え字が格納される．最初の時点では，探索が全く行われていないので，探索した範囲内はそのピクセルのみであり，そのピクセルの添え字そのものが格納されている．この状態で，カーネル ccl_main をフレームバッファの最外周を除く全ピクセルに対して 10 回程度適用すると，4 連結成分抽出処理が完了し，各ピクセルが最初に背景色であれば 0，前景色であれば，そのピクセルが属する連結成分のうち最も添え字の小さなピクセルの添え字が代入される．カーネル ccl_main は 3 つのフェーズからなり，フェーズ 1 では，各ピクセルの上下左右のピクセルの値を読み出し，その値が背景色ではなく，もとのピクセルより小さい値なら，変数 g にその値を格納する．フェーズ 2 では，g =fb$[g]$ という代入を 2 回以上繰り返す．これにより，隣接するピクセルからの探索結果を g に反映することができる．フェーズ 3 では，g をフレームバッファに書き戻す．この際，フェーズ 1 が始まる時点でそのピクセルが指していたピクセル fb$[og]$ に対しても g を上書きする．これは，連結成分が例えば「W」字型をしているような場合に，添え字の大きいピクセル間の探索結果をより添え字の小さいピクセルに伝播させるために必要となる．fb$[og]$ を指しているピクセルは他にも多数ある場合が多く，それらのピクセルには次回カーネルが適用されたときに一度に新たな結果が伝播する．

```
kernel void ccl_prepare(global int *fb) {
  int width = 2048;
  int x = get_global_id(0), y = get_global_id(1), p0 = y * width + x, out = 0;
  if (fb[p0] != 0) out = p0;
  fb[p0] = out;
}

kernel void ccl_main(global int *fb) {
  int width = 2048;
  int x = get_global_id(0), y = get_global_id(1);
  int p0 = y * width + x, g = fb[p0], og = g, s;

  if (g != 0) {
    /* phase 1 */
    s = fb[(y - 1) * width + x];
    if (s != 0 && s < g) g = s;

    s = fb[(y + 1) * width + x];
    if (s != 0 && s < g) g = s;

    s = fb[y * width + x - 1];
    if (s != 0 && s < g) g = s;

    s = fb[y * width + x + 1];
    if (s != 0 && s < g) g = s;

    /* phase 2 */
    g = fb[fb[fb[fb[g]]]];

    /* phase 3 */
    if (g != og) {
      atomic_min(&fb[og], g);
      atomic_min(&fb[p0], g);
    }
  }
}
```

図 7　4 連結成分抽出処理の OpenCL カーネル

★1 具体的には，文献 12) 内の Algorithm 8 において，function Mesh_Kernel_D_Analysis_Phase 内の repeat〜until ループの繰り返し回数が入力画像の縦または横の長さに比例して大きくなる．