Vol.2011-EC-19 No.19
2011/3/27

モバイルプロジェクタカメラシステムを用いた任意非平面への実時間適応的投影手法

関　栄二[†1]　ダオ ヴィン ニン[†1]　杉本 雅則[†1]

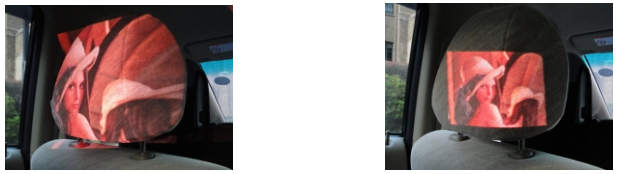
本稿では，モバイルプロジェクタカメラシステムを用いることにより，任意非平面へのリアルタイムかつ適応的な投影を行う手法を提案する．提案手法では，imperceptibleなチェッカーボード構造光を埋め込んだ画像を投影することにより，投影面の3次元形状取得と幾何学的に補正された画像の提示を高速に行う．さらに，システムの移動や回転に対し，与えられたユーザ視点において幾何学的に補正された画像を常に提示する．

A Realtime and Adaptive Technique for Projection onto Non-Flat Surfaces Using a Mobile Projector Camera System

Eiji Seki,[†1] Dao Vinh Ninh[†1]
and Masanori Sugimoto[†1]

In this paper, we describe a realtime and adaptive technique for projection onto non-flat surfaces using a mobile projector camera system. The proposed technique allows a user to rapidly capture the 3D geometry of an object by using an imperceptible checkerboard pattern and project a geometrically calibrated image onto surfaces of the captured object. The system can always show calibrated images to a given user viewpoint while it is moved or rotated.

†1 東京大学
The University of Tokyo



(a)　(b)

図 1　車内での投影の様子: (a) 補正をしていない画像; (b) 補正を行った画像
Fig. 1　Projection in a car: (a)Non-calibrated image; (b)Calibrated image

1. はじめに

近年，プロジェクタの小型化に伴い，様々なモバイル機器への搭載が進んでいる．時と場所を選ばずに手軽に投影が可能になることにより新しいアプリケーションの可能性が広がる[6)2)11)5)] 反面，いくつかの問題が生じる．

本論文では，特に投影面が非平面である場合の投影について考える．図 1(a) のように，投影面の形状によって，ユーザからは投影した画像が歪んで見える．そのため，図 1(b) のように，補正した画像を投影することが必要となる．幾何学補正についてはこれまでにいくつかの方法が提案されている[7)9)8)]．本稿では，投影面の 3 次元的な形状を取得し．得られた形状とユーザ視点の位置の情報を用いて，幾何学補正を行う手法を提案する．

モバイルプロジェクタカメラシステムでは，ユーザまたはシステムの移動や，動画投影の要求に対処する，などの理由から幾何学補正をリアルタイムで行う必要がある．しかし一方で，上記の方法には，多大な計算時間がかかるという問題がある．というのは，3 次元形状取得，投影画像の歪み補正共に多くの画像処理が求められるためである．特に，サイズや電力の問題から，高性能な CPU に頼ることが難しいモバイル機器においては，大きな問題となる．

こうした速度の問題に対して，GPGPU（General-Purpose computing on Graphics Processing Unit）が解決策となるのではないかと考えられる．モバイル用の GPGPU も現実

Vol.2011-EC-19 No.19
2011/3/27

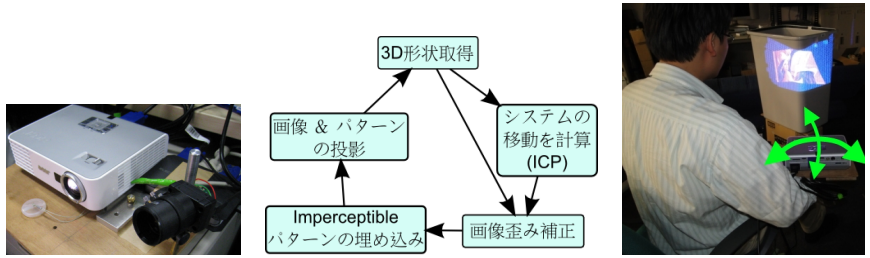


図 2 (a) システム構成; (b) システムの処理の流れ; (c) システム利用の様子
Fig. 2 (a)Configuration of the proposed system; (b)processing flow of the system; (c)the system in use

になりつつあり[*1]，将来的にはシステムのモバイル化にも対応できると考える．

以上の問題意識をもとに，任意非平面と視点が与えられた場合に，視点から見た歪みを補正する方法を考案した．また，将来的なモバイル化を視野に入れつつ，提案手法を用いた適応的投影システムを構築した．さらに，このシステムに対して GPGPU を適用し，高速化を行った．

2. 適応的投影システムについて

2.1 概　　要

構築したシステムを図 2(a) に示す．提案システムは，DLP プロジェクタ（ACER P3251）とカメラ（PointGrey Firefly MV），そして，互いの同期をとるためのモニタ分配器からなる．

本システムの処理の流れは図 2(b) の通りである．まず，imperceptible なチェッカーボードパターンを物体へ投影し，それをカメラで取得する．そして，パターンの特徴を利用して 3 次元的な形状を取得する．次に，この形状を利用し，ICP アルゴリズムによってシステムの移動を検出し，2(c) のように，システムを持ったユーザの手をトラッキングする．以上の情報から投影画像の補正を行う．1 つのプロジェクタから，画像とチェッカーボードパターンを同時に投影するため，パターンの埋め込みを行う．

[*1] GPGPU が可能と主張する PowerVR SGX（Imagination Technologies 社）[12] や，将来的な CUDA（NVIDIA 社提供の GPGPU のための C 言語環境）対応を目指す NVIDIA Tegra（NVIDIA 社）[10] など

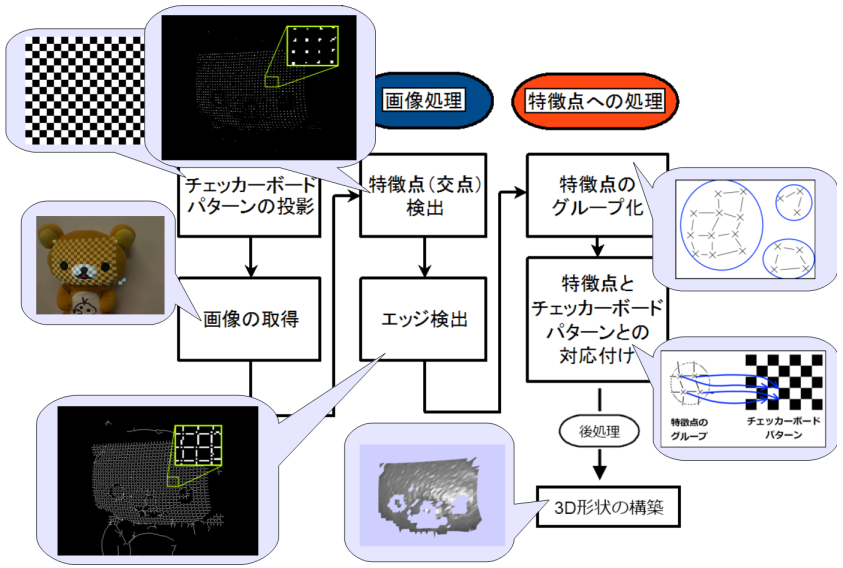


図 3 3 次元形状取得の流れ
Fig. 3 Flow of 3D acquisition

2.2 3 次元形状取得手法の概要

本システムでは，3 次元形状取得手法として，チェッカーボードパターンを用いた構造光による三角測量法[4] を利用する．本手法の特徴として，システムや物体が移動していても形状を取得できるという点や，物体の色や周囲の環境光に対して，ロバストに形状を取得できるという点が挙げられる．これらは，モバイル化を進める上での要求を満足する．

本手法において，3 次元形状取得の流れは図 3 のようになっている．まず，プロジェクタから物体にチェッカーボードパターンを投影し，これをシステムのカメラで取得する．取得された画像から，特徴点であるチェッカーボードパターンの交点を抽出する．次に，これらの特徴点をつなぐチェッカーボードパターンのエッジを抽出する．さらに，エッジでつながれた特徴点をグループ化し，特徴点のチェッカーボードパターン上での位置を，エピポーラ

Vol.2011-EC-19 No.19
2011/3/27

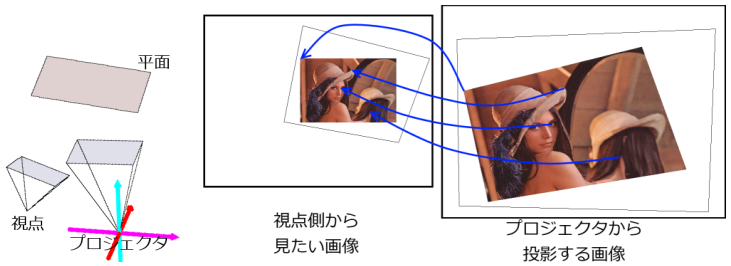


図 4 1 平面の場合の歪み補正: (a) 平面とプロジェクタ・視点の位置関係; (b) 投影する画像のピクセルごとに，対応するピクセルを参照する

Fig. 4 Image calibration for projection onto flat surfaces: (a)A plane, a projector and a view point;(b)Correspondence between a pixel on a projector and that from a view point

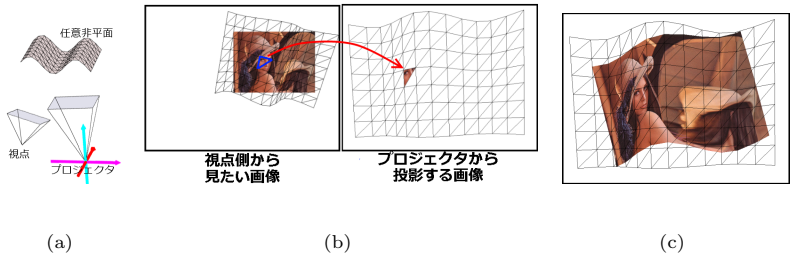


図 5 任意非平面に対する歪み補正: (a) 任意非平面とプロジェクタ・視点の位置関係; (b) 三角形ごとに対応をとる; (c) 任意非平面に対して補正を行った画像

Fig. 5 Image calibration for projection onto non-flat surfaces : (a)A non-flat surface, a projector, and a view point; (b)Correspondence between triangle on a projector and that from a view point; (c)A calibrated image to be projected onto a non-flat surface

幾何を用いて，グループごとに計算して対応付ける．

2.3 ICP アルゴリズムによる移動検出

歪み補正を行う上では，プロジェクタと視点との位置関係 —回転と平行移動— の情報が必要である．現時点では，初期位置関係を適宜与える一方で，プロジェクタカメラシステムの移動を検出可能にしている．これにより，図 2(c) のように，システムを持ったユーザの手の動きを検出することが可能になる．

本システムでは，ICP（Iterative Closest Point）アルゴリズム[1] により検出を行う．ICP の利点として，追加の機器が必要ないという点が挙げられる．その一方で，欠点としては，計算時間がかかることや，大きな回転・平行移動は検出が難しいという点が挙げられる．

2.4 3 次元歪み補正

2.4.1 対象が 1 平面の場合

任意非平面の場合について述べる前に，より単純な 1 平面の場合の歪み補正について述べる．この場合には、以下の式 1 で表される Homography と呼ばれる計算方法を用いることができる．

$$p_a = K_a \cdot H_{ba} \cdot K_b \cdot p_b \tag{1}$$

式 1 は，一方のカメラ平面上の点 p_b から，もう一方のカメラ平面上の対応する点 p_a を求める式である．K_a，K_b は各カメラのカメラ行列を表している．また，H_{ba} は以下の式 2 で表される．

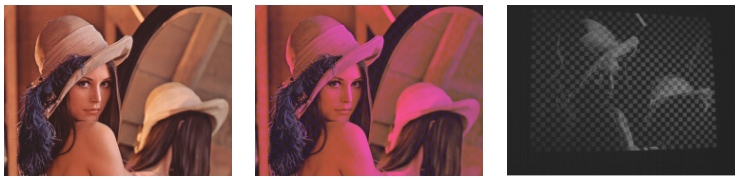
$$H_{ba} = R - \frac{t \cdot \mathbf{n}^T}{d} \tag{2}$$

R，t は共に 2 つのカメラの位置関係を表すパラメータであり，それぞれ回転移動，平行移動を表している．また，$\mathbf{n}$，d はともに平面のパラメータであり，それぞれ平面の法線ベクトル，カメラからの距離である．

最初の手順として，視点側から見て，画像がどのような矩形範囲 S に収まるべきかを決める．この S は，視点とプロジェクタの投影範囲の双方に収まり，かつ投影する画像と縦横比が同じとなる，最大の矩形である．次に，プロジェクタから投影すべき画像を作成する．まず，点で近似したプロジェクタ側の各ピクセルが視点側の平面上のどの点に当たるかを，Homography によって計算する．そして，対応する元画像の色でピクセルを塗りつぶすことで，図 4 のような結果が得られる．

2.5 対象が任意非平面の場合

1 平面の場合の画像補正法を応用し，対象が任意非平面の場合についての画像補正法を示す．なお，ここでの任意非平面は，3 次元取得手法により取得した三角形ポリゴンの集合で



(a) (b) (c)

図 6 画像への構造光パターンの埋め込み: (a) 元画像; (b) パターンを埋め込んだ画像; (c) カメラから見た画像
Fig. 6 Embedding structured light pattern in image: (a)An original image; (b)A pattern embeded image; (b)Captured image from camera

表されている．

任意非平面の場合も，1 平面の場合と同様の手順で補正を行うことができる．具体的には，①視点側から見た画像の収まるべき矩形範囲を決定し，②投影画像のピクセルと元画像のピクセルとの対応をとって投影画像を作成する，となる．ここでは，②の処理について示す．

任意非平面を構成する三角形の一つに注目すると，この三角形はある平面の一部であると見ることができる．したがって，この三角形をプロジェクタ平面側へ射影したときに，射影された三角形内部のピクセルに限って言えば，1 平面の場合と同様に処理が可能である．これを表したのが，図 5(b) である．この処理をすべての三角形について繰り返すことで，図 5(c) のような投影すべき画像が作成できる．

2.6 構造光パターンの投影画像への埋め込み

本システムでは，3 次元形状取得のための構造光パターンを投影するプロジェクタと，補正した画像を投影するプロジェクタは同一である．したがって，何らかの方法でパターンを画像へ埋め込み，同時に投影する必要がある．

そこで，本システムでは，パターンの埋め込みに Cotting 他[3] の方法を用いる．この手法では，DLP プロジェクタの特性を用いて，画像の各ピクセルの階調を調整する．これを，図 6(a) に対して行う．すると，図 6(b) のように，色の変化が見られるものの，人間の目からは，ほとんどチェッカーボードパターンが見えないことが分かる．一方，図 6(c) に示すように，同期をとったカメラの画像では，埋め込まれたパターンを確認できる．

3. GPGPU によるシステムの高速化

3.1 GPGPU の概要

GPGPU は「大量のデータへ」「同じ処理を」「並列に」行うことが得意である．逆に言えば GPGPU は，たとえば複雑な条件分岐を含むような処理には向かない．こうした特性を考えると，GPGPU を適用するには，最低限以下の条件が必要となる．

(1) 全ての要素への処理が同じである：多少の条件分岐は許されるが，分岐が増えるほど，処理が劇的に遅くなる
(2) 各要素の処理が独立している：要素間の処理に依存関係がある場合，並列に処理することは不可能である
(3) 要素数が十分多い：要素数が少ない場合，十分に処理速度が向上しない．そのため，メモリ転送といった GPGPU 特有のオーバーヘッドによって，むしろ遅くなりうる

以上から，GPGPU のための処理の最適化手順は①ボトルネックとなる処理を発見し，吟味すること，②要素ごとの処理を同じかつ独立にすること，③メモリ利用の最適化を行うこと，となる．なお，③については実装上は重要であるが，本稿では割愛する．

3.2 システムのボトルネック

GPGPU の適用箇所を考えるため，システムの速度を測定した．測定条件は，CPU : Intel Core i7 860（2.80 GHz），取得する三角形の数 : 約 1800 個，カメラで取得した画像の解像度 : 640 × 480，投影する画像の解像度 : 800 × 600，であった．この結果，ICP アルゴリズムが全体の 41%, 3 次元形状取得における画像処理が全体の 25%，歪み補正が全体の 11%を占め，ボトルネックとなっていることが分かった．

以下では，これらのボトルネックに対して GPGPU を適用する．ただし，ICP アルゴリズムに関しては，本稿では適用を見送り今後の課題とした．

3.3 GPGPU の適用

3.3.1 3 次元形状取得手法

3 次元形状取得においてボトルネックになっているのは，画像処理，すなわち特徴点抽出とエッジ抽出である．さらに詳しく見ると，特徴点抽出が画像処理全体の 43%をしめ，エッジ抽出に先立つ 4 種類のマスク処理が全体の 14%を占めている．これらが，本手法において最も重い 2 つの処理となっている．そこで，これらの処理に GPGPU を適用する．これらの処理は，元々 GPGPU との親和性の高い処理である．というのは，いずれの画像処理も，ピクセルごとに完全に独立な処理を行うためである．さらに，処理する要素が 640 × 480 ピ

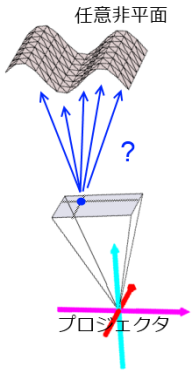


図 7 ピクセルごとの処理では，三角形の探索が必要
Fig. 7 Pixel-by-pixel processing is needed to find a corresponding triangle

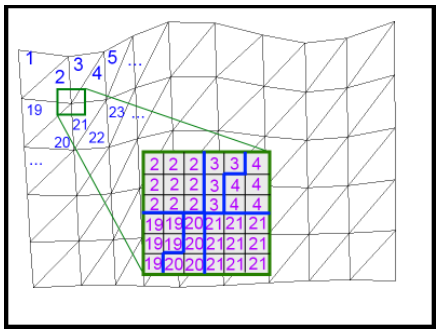


図 8 各ピクセルがどの三角形に対応するかを CPU によってマッピングしていく
Fig. 8 Finding a mapping between each pixel and triangle is conducted by a CPU

クセルと多いことも，GPGPU を適用する上での利点として挙げられる．

3.3.2 3 次元歪み補正

3 次元歪み補正処理は，必ずしも GPGPU への親和性が高くない．なぜなら，全てのピクセルを完全に独立して処理を行うことは，むしろ効率の低下を招くからである．これは，あるピクセルを選んだ時に，それが任意非平面を構成する三角形のいずれに当たるかを，図 7 のように 1 回 1 回探索しなければならないという事情による．したがって，まずは処理をする三角形を任意に選択し，それに対応するピクセル群を処理する，という手順を取ることになる．このとき，並列化の方針としては，（1）複数の三角形を並列に処理する場合と，（2）三角形内のピクセルを並列に処理する場合の 2 つが考えられる．

しかし，こうした並列化はいずれも GPGPU には適さない．（1）の方法では，要素（三角形）ごとに処理範囲が異なる，すなわち，処理が「同じ」でない．また，（2）の方法では，三角形一つごとに CPU 側から GPU 側へ処理範囲を転送するコストや，矩形状にしかスレッドを立てられないために，ピクセルの重複処理といったコストが発生する．

そこで，GPGPU に適したピクセルごとの処理を行うために，①図 8 のように，あるピクセルがどの三角形に対応するかを，三角形ごとに計算し，②その情報をもとにピクセルごとの処理を行う[⋆1]，という方法を提案する．ここまで述べたように，GPGPU は①のような三角形ごとの処理を苦手とするため，この部分は CPU で処理を行う．②でピクセルごとに行う処理は Homography であり，条件分岐のない行列演算である．したがって，GPGPU による高速化が期待できる．

以上，①のように条件判定が複雑だが軽い処理を CPU に任せ，②のように条件判定が単純だが重い処理を GPU に任せる，という分業を行うことで，処理の高速化が望めることを考察した．

表 1 特徴点抽出，マスク処理への GPGPU 適用結果
Table 1 Results of applying a GPGPU to feature point detection and mask processes

	CPU（ms）	GPU（ms）
処理対象の画像の GPU への転送	–	0.4
特徴点抽出	10.5	1.6
マスク処理 4 種類 合計	3.4	1.0
合計	13.9	3.0

表 2 3 次元歪み補正への GPGPU 適用結果
Table 2 Results of applying a GPGPU to image calibration

	CPU（ms）	GPU（ms）
インデックスの確定	–	1.58
歪み補正	11.22	3.09
合計	11.22	4.67

4. 評価結果

4.1 GPGPU の適用結果

4.1.1 速度比較の概要

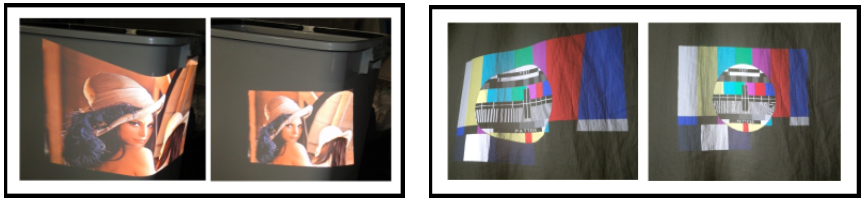
3 次元形状取得手法における画像処理の一部（特徴点検出とエッジ検出に先立つマスク処理）と，3 次元歪み補正への GPGPU の適用を行った．提案する GPGPU での処理速度を評価するため，CPU での処理と比較した．使用した CPU は 3.2 と同様であり，GPU には NVIDIA GeForce GTX 295，SP 数 240[⋆2]，1.24GHz を用いた．

4.1.2 3 次元形状取得への GPGPU 適用結果

実験結果を表 1 に示す．なお，両処理で共通に使う画像データを，1 度に GPU へ送っているため，この部分の処理時間は分けて記した．

表 1 から，転送時間を含めた両処理の合計時間は 4.6 倍，処理時間にして 10.9 ミリ秒の

⋆1 対応する三角形の存在しないピクセルに対しては，何もしない
⋆2 基板上に性能の同じ 2 枚のチップを搭載している．しかし，現状では 1 枚のみ使用している．そのため SP 数としては 1 枚分のものを記した



(a) ゴミ箱への投影　　(b) カーテンへの投影

図 9 任意非平面への投影の様子; (a)(b) ともに左が未補正の画像，右が補正した画像
Fig. 9 Projecting images onto non-flat surfaces; non-calibrated images (left) and calibrated images (right) in (a) and (b), respectively

高速化が行われていることが分かる．したがって，GPGPU の適用により十分な高速化が行なえたと考える．

4.1.3 3 次元歪み補正への GPGPU の適用結果

3.3.2 で述べたように，歪み補正に対しては一部の処理を CPU に担わせ，GPGPU の適用を行っている．CPU のみの場合と速度を比較した結果，表 2 のようになった．これより，CPU によるインデックス処理を含めても，GPGPU の適用により 2.4 倍程度の高速化ができたことが分かる．

4.2 任意非平面への適応的投影

任意非平面の適応的投影の実験結果として，二つの例を図 9 に示した．この結果から，ユーザ視点から見て，歪みが十分気にならない程度に，歪み補正ができたと考える．

5. 結論と課題

本稿では，モバイルプロジェクタカメラシステムを用いることによる非平面への適応的投影システムの提案と実装を行った．また，少なくとも PC 上においては GPGPU の適用によって，システムの高速化が行えることを示した．今後の研究では，解決すべき以下の課題に取り組む予定である．

- ICP アルゴリズムへの GPGPU 適用
- モバイルプロジェクタカメラシステムへの展開
- 視点のトラッキング
- 光学補正への対応

参 考 文 献

1) Besl, P. and McKay, N.: A Method for Registration of 3-D Shapes, *IEEE Trans. PAMI*, Vol.14, No.2, pp.239–256 (1992).
2) Cao, X., Forlines, C. and Balakrishnan, R.: Multiuser Interaction Using Handheld Projectors, *Proceedings of ACM UIST 2007*, pp.43–52 (2007).
3) Cotting, D., Naef, M., Gross, M. and Fuchs, H.: Embedding Imperceptible Patterns into Projected Images for Simultaneous Acquisition and Display, *Proceedings of IEEE/ACM ISMAR 2004*, pp.100–109 (2004).
4) Dao, V. and Sugimoto, M.: A Dynamic Geometry Reconstruction Technique for Mobile Devices Using Adaptive Checkerboard Recognition and Epipolar Geometry, *IEICE Trans. Information and Systems*, Vol.E94-D, No.2, pp.336–348 (2011).
5) Löchtefeld, M., Gehring, S., Schöning, J. and Krüger, A.: ShelfTorchlight: Augmenting a Shelf using a Camera Projector Unit, *Proceedings of Workshop on Ubiprojection 2010* (2010). retrieved at http://eis.comp.lancs.ac.uk/workshops/ubiproject 2010/pdf/loechterfeld_ubiprojection2010.pdf.
6) Mistry, P., Maes, P. and Chang, L.: WUW - Wear Ur World: A Wearable Gestural Interface, *Proceedings of ACM CHI 2009*, pp.4111–4116 (2009).
7) Raskar, R., Baar, R., Beardsley, P., Willwacher, T., Rao, S. and Forlines, C.: iLamps: Geometrically Aware and Self-Configuring Projectors, *Proceedings of ACM SIGGRAPH 2003*, pp.809–818 (2003).
8) Zollmann, S. and Bimber, O.: Imperceptible Calibration for Radiometric Compensation, *Proceedings of EUROGRAPHICS 2007*, pp.61–64 (2007).
9) Zollmann, S., Langlotz, T. and Bimber, O.: Passive-Active Geometric Calibration for View-Dependent Projections onto Arbitrary Surfaces, *Journal of Virtual Reality and Broadcasting*, Vol.4, No.6 (2007).
10) 笠原一輝：CUDA の世界を広げ “CUDA Everywhere” 構想〜モビリティデバイスでは x86 ではなく ARM に賭ける NVIDIA，Impress（オンライン），入手先⟨http://pc.watch.impress.co.jp/docs/column/ubiq/20100924_395960.html⟩（参照 2011-02-14）.
11) 細井，ダオ，森，杉本：CoGAME:ハンドヘルドプロジェクタを用いたロボットナビゲーションゲームの試作，日本バーチャルリアリティ学会論文誌，Vol.12, No.3, pp.285–294 (2007).
12) 北郷達郎：【CES】携帯機器でも GPGPU を実現, Imagination Technologies 社，，入手先⟨http://techon.nikkeibp.co.jp/article/NEWS/20090111/163928/⟩（参照 2011-02-14）.