タイトル	手持ちスクリーンへのリアルタイムプロジェクションマッピングシステムの開発
著者	菊地, 慶仁; 上山, 凌; KIKUCHI, Yoshihito; Kamiyama, Ryo
引用	工学研究 : 北海学園大学大学院工学研究科紀要(15): 21-25
発行日	2015-10-30

研究論文

手持ちスクリーンへのリアルタイム プロジェクションマッピングシステムの開発

菊 地 慶 仁* ・ 上 山　　凌**

Development of Real Time Projection Mapping System for Handheld Screen

Yoshihito KIKUCHI* and Ryo Kamiyama**

要　旨

近年，駅舎や城郭などの大型建築物及び雪像を対象としたプロジェクションマッピングが行われ，多数の観客を呼び寄せている．一般にプロジェクションマッピングは，建築物などの固定された物体をスクリーンとし画像を投影している．大規模な建築物への投影は大型の投影機材などを必要とし結果的に高いコストが必要となる．また建築物への投影はスクリーンの配置が固定的になってしまう．本研究は一般的なプロジェクターを用い，手持ちの長方形のスクリーンに画像認識技術によって投影を行うシステムの開発を行った成果について報告する．

1．プロジェクションマッピングの基本的な方式

プロジェクションマッピングは，一般的には既存の建築物などにプロジェクターで紋様を投影することで意匠的な効果を与えるものである[1]．投影される画像は，キャラクターや建築物のテクスチャーなど自由に選択することができる．特に冬期間に雪像に対して投影する場合は，対象が白一色であるために効果的な投影となり好評を博している[2]．図１に札幌雪祭りで行われたプロジェクションマッピングの例を示す．２枚の図は，固定されたカメラで撮影した一つの動画から取り出した別々のシーンである．もともとは左側のように白色の馬の雪像であるが，右側では背景や馬の体内組織が見えるような画像を投影している．

プロジェクションマッピングでは一般的には，対象物の輪郭形状に合致させたモデリングを予め行い，そのモデル上での動画を作成して投影コンテンツとしている．特に凹凸形状などにまで合致

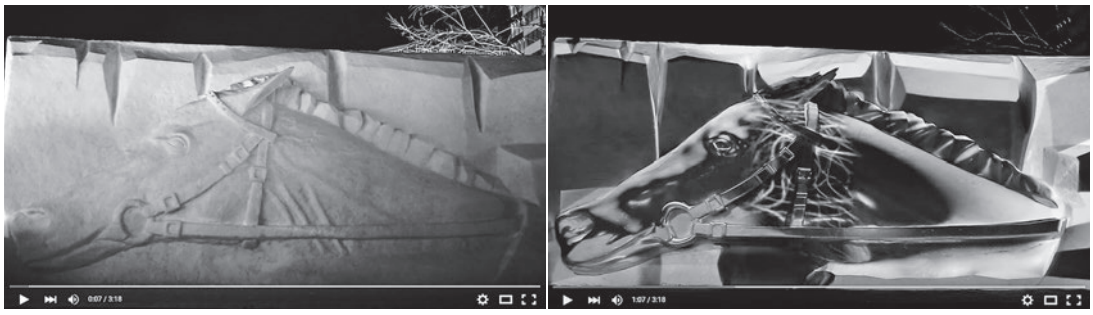

図１　札幌雪祭りで行われたプロジェクションマッピングの動画．左はオリジナルの雪像のみ．右は背景，馬の外形，体内の組織などの画像を投影している．

* 北海学園大学工学部電子情報工学科
Hokkai-Gakuen University Faculty of Engineering Department of Electronics and Information Engineering
** リコーIT ソリューションズ（北海学園大学工学部電子情報工学科卒）
RICOH IT Solutions (Graduated from Hokkai-Gakuen University)

させた効果が必要な場合や，角度のある投影による効果を期待している場合には，対象物体を 3 D でモデリングすることもある．

プロジェクションマッピング用のコンテンツ作成には動画作成用のソフトウェアが用いられることが多いが，投影時には一般的なプレゼンテーションソフトの他に，動的に動画を制御するためにスイッチャーと呼ばれる専用のソフトウェアが用いられることもある．

投影の対象は，前述のように屋外での建築物等を対象とすることが多かったが，近年では，室内や舞台などでより小型の対象にも行われるようになっている．このような場合には，投影対象の前後で人間が動くことが多いので，これらの障害物を避けての投影や，また投影対象自体も移動することが考えられる．

そこで本報告では，上述の小規模な会場における投影を想定し，投影させる対象の位置や姿勢が変わるような場合にでも対応できることを目的とする．投影対象の形状は予め決めておき，画像処理による投影対象の位置及び姿勢の検出と投影を同時に行うシステムの開発を行った内容について報告する．

2．画像認識方式による投影対象の認識における課題

本研究では，観測された画像から手持ちのスクリーンに関する情報を抽出して，そのスクリーンに画像を投影することを目的としている．

使用できる画像処理システムとして，一般的な USB（Web）カメラ，プロジェクター，及び画像処理用のライブラリとして Open-CV[4,5,6] を用いる．

認識の手順は，長方形のスクリーン（ホワイトボード）を人間が手に持ち，この状態をウェブカメラで撮影し，そのカメラ映像から Open CV を利用してスクリーンを認識させる．さらにスクリーンの大きさと傾きを検出することで画像を変形させ，スクリーンにぴったりと合うように画像処理をした後にプロジェクターを使ってスクリーンに投影する．実際には，プロジェクターからはその投影領域の境界を投影し，この境界の画像とスクリーンとの相対的な位置関係を求めることになる．より詳細には以下の各項目が課題となってくる．

1）スクリーンの境界（の包含領域）とプロジェクターの投影範囲を認識する
2）プロジェクターの投影範囲内でのスクリーン境界の位置・大きさ・傾きを決める
3）求めたスクリーンの位置・大きさ・傾きの値に基づいて画像を変形処理する
4）プロジェクターを用いてスクリーンに映像を投影させる
5）スクリーンが移動もしくは回転しても画像がスクリーンに投影され続ける
6）以上の処理をリアルタイムで行うこと．

取りだす情報を図 2 に示す．人間を包含する点線の領域がプロジェクターからの投射エリア境界である．またスクリーンは人間によって手持ちされる．一般的な表現とするため，スクリーンはプロジェクターの投影エリアの原点に対して一定のオフセットされた位置にあるとし，傾きも持っているとする．またスクリーンは，プロジェクターの投影エリアにその全体が含まれており，かつ短辺を上下の境界としていることとする．

3．画像の認識方法

3.1　スクリーンの位置及び姿勢の検出する方法について

スクリーンの位置及び姿勢を検出し，その位置に画像を投影する方法としては，AR-Toolkit[3] 用のマーカーをスクリーンに貼り付けて，その撮影画像に対してプロジェクターから投影する方式が考えられる．また Open-CV に付属する輪郭抽出

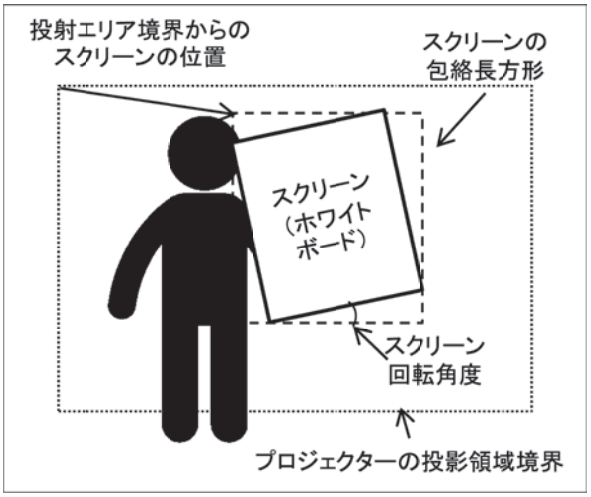

図 2　USB ウェブカメラ映像から取得する情報

表 1　本報告で用いた器材

器　材	名称，仕様等
プロジェクター	Ben-Q MX522P ワット数（光源）：190 W 輝度：3000 lm 画素数：XGA（1024×768） 解像度：QVGA（1280×960） コントラスト比：13000：1 使用接続入力端子：D-Sub
PC	CPU：Intel® Core™ i7-2600 ＠ 3.40 GHz Memory：8192MB Motherboard：MSI MS-7678 GPU：Intel® HD Graphics Family OS：Windows 7 Professional 64bit SP1
ウェブカメラ	ELECOM UCAM-DLA200HSV 画素数：200 万画素 フレームレート：25 FPS 解像度 CMOS 640×480（実験環境）
PC 周辺機器	ディスプレイ，マウス，キーボード
コンパイラ ライブラリ	Visual Studio Express 2013 C＋＋ Open CV Ver. 2.4.9

アルゴリズムを用いてスクリーンの輪郭を検出方式が考えられる．

AR-Toolkit はモニタ上のマーカー画像に対して仮想 3 D空間のオブジェクトを合成してモニタに出力するためにライブラリであり，マーカー上にオブジェクト画像を投影してさらに USB カメラで撮影した場合には安定した動作が保証できないことが考えられる．また映像を投影しない場合でもプロジェクターからの画像がスクリーンで反射されるとカメラでの撮影時にハレーションが発生する恐れもある．

このような点を考慮して本研究では AR マーカーをスクリーンに貼り付ける方式は取らずに，Open-CV に含まれる輪郭抽出アルゴリズムを用いることとした．

3.2　使用器材

本研究で使用した器材を表 1 にまとめる．CPU とメモリ量に余裕を持たせているが，グラフィック性能を特に要求されないので標準的なものを搭載している．

3.3　輪郭抽出

輪郭抽出は以下の手順で行った．説明中の「cv::」は，OpenCV のライブラリ中で CV クラスに属するメソッドであることの C＋＋に従った表記である．

1）入力画素の二値化
2）cv::findContours を用いてカメラの視界中にある輪郭点を求める．
3）輪郭点に対して cv::approxPolyDP を行い閉じた多角形近似を行う．
4）一定のサイズの範囲にある多角形を抽出することで cv::approxPolyDP の返り値としてスクリーンの頂点座標を抽出する．

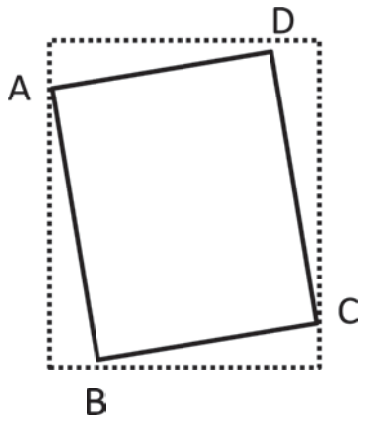

図 3　スクリーンの傾きの導出方法

3.4　座標，面積，傾きの抽出

スクリーンの左上の頂点座標を求めるために以下の手順に従って行った．

図 4　スクリーンを傾けて椅子の上に固定し位置及び傾き角を検出して投影した結果

1）cv::convexHull 関数を用い，また的確な変数指定を行うことで，Y 座標値が最少の（図では上側の）短辺の左側の頂点を始点として，反時計回りに頂点列を取り出す．
2）各頂点の座標からスクリーンのサイズ及び面積を求める．
3）スクリーンが右に傾いている場合と左に傾いている場合に分けて，スクリーンの短辺と水平線との角度を求める．図 3 での点 B と C に着目し，この 2 点の Y 座標値からスクリーンがどちらに傾いているかを判定できる．また BC 間の Y 座標値の差，及び X 座標値の差から次式を使って角度を求める．

$$\theta = \tan^{-1}\left(\frac{\text{高さ}}{\text{底辺}}\right) \qquad \text{（式 1）}$$

3.5　画像に対しての座標変換

座標変換は，OpenCV の関数を以下の手順で適用して座標を指定する．

1）cv::resize を用いて元の画像の縦横比を維持したまま画像のサイズを取得できたスクリーンのサイズに適合させる．
2）取得できたスクリーン座標から中心座標を求め，この点を中心に回転することでスクリーンの枠に画像を合わせる．具体的には，cv::getRotationMatrix2Dwo を用いて座標変換用のアフィン行列を作り出し，cv::warpAffine にこの行列を与えて画像の変換を行う．
3）変換を行う投影画像を，そのサイズのまま用いて変換すると回転によって ROI（Region of Interest）を超えた端点が切り取られてしまうので，予め余白部分を加えた画像を用意し，余白も含めて回転座標変換を適用し切り取りが起きないようにする．

4．画像投影実験

4.1　実験結果

図 4 に本研究で開発したシステムを用いてテスト画像を傾いたスクリーンに対して投影した結果を示す．

スクリーンに画像を投影することには成功し，スクリーンを上下左右に移動させても画像がしっかりとスクリーンに投影され，回転させても回転

に合わせて画像が回転したので動的なプロジェクションマッピングのプログラムとしての機能は達成しているといえる.しかし,いくつか問題は残った.

4.2　問題点

実験を行った際に判明した問題点としては以下の項目があった.

1）スクリーンの領域検出は，二値画像によって輪郭検出を行い直線近似によって検出しているが，投影画像がスクリーンの縁に当たった場合その反射光によってハレーションが起こり，直線として認識できない場合にスクリーンを見失ってしまう．同様に自然光などによりスクリーンの縁の明度が上がるとスクリーンを見失ってしまう．従って，画像による縁の認識ではなく，別研究で用いてきた赤外LEDなどを用いることで解消できる可能性がある.

2）スクリーンとプロジェクターの距離が変化した際にスクリーンからずれて投影される問題が発生した．この問題はスクリーンを測距センサーなどで距離を測り，それに応じて座標を修正する事も可能かもしれない.

3）プロジェクター用プレーンの範囲外へ描画しようとした際にOpenCVによる画像認識でエラーが発生し強制終了してしまう問題が発生した．この問題は，エラーが出ている関数であるROIの失敗した場合の戻り値を分析し，対応する必要がある.

4）PCのCPUのみを用いて座標及び角度検出を行い，さらに画像を投影するには処理速度に限界があった.現実の実装を行うためには，並列化された処理システムの開発が必要になると考えられる.

5．結論

本報告では以下の報告を行った.

1）手持ちスクリーンの座標及び回転角を計測することで，スクリーン内に任意の画像を投影するプロジェクションマッピング方式を提案した.

2）提案した方式を実装するために画像処理のみを入力とするシステムの検討を行った.

3）実際にOpenCVを用いた実装プログラムを開発し実験を行った．処理速度には限界があるが，当初提案した機能を実装できたことを確認できた.

今後の展開としては，より実際の利用を想定したシステムの高速化が課題となるので，画像処理のみによる検出以外のスクリーンの座標及び角度検出を行える技術との組み合わせを試みる必要がある.

【参考文献】

1）プロジェクション・マッピング入門（特別付録DVD-ROM・紙模型付録付き），MASARU OZAKI（尾崎マサル），玄光社

2）〝第66回さっぽろ雪まつり「サラブレッドの息吹」プロジェクションマッピング〞，https://www.youtube.com/watch?v＝b62ls_-jE9g

3）橋本　直，〝3Dキャラクターが現実世界に誕生！ARToolKit拡張現実感プログラミング入門〞，アスキー・メディアワークス，2008/9/17

4）Open CV公式サイト，http://opencv.jp/

5）Open CV 2.2 C＋＋リファレンス，http://opencv.jp/opencv-2svn/cpp/index.html

6）Stack Over Flow，http://stackoverflow.com/questions/220