

おっぱいマウスの画像処理

1. はじめに

本文書は動画「おっぱいマウス作ってみた」で用いられている画像処理に関する技術解説である。装置は市販のおっぱいマウスパッドを用いて作られており、3mm 黒色ビーズ（マーカ）がおおよそ均等に分布している(図 1)。



図 1 マーカ撮影画像（処理対象入力画像）

娯楽用 UI であるので、そのセンシング精度の定量性は重要ではなく、強いタッチ・弱いタッチといった感覚指標と操作者の意図に大きな乖離がなければ良いものとしている。その緩和条件のおかげで、特開 2009-28803 の光学式マーカのように事前にマーカの 3 次元位置を知る必要はない。ゲル下面から約 1cm のところに、おおよそ均等に分布しているという仮定のみを用いる(図 2)。

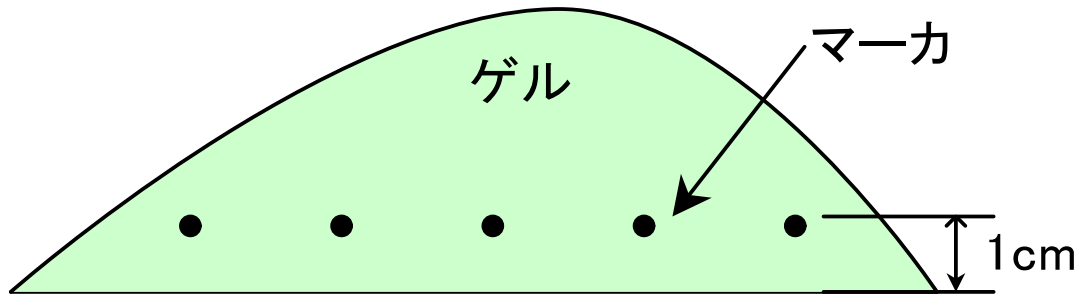


図 2 ゲル部分断面図

2. ファイル説明

本装置で用いられているプログラムⁱⁱ⁾は mouse.cc, marker.cc(同.h)の 2 つである。

2.1. mouse.cc

メインの処理を行う。検出されたマーカに基づいて、マウス機能(MODE_MOUSE)、圧力マップ機能(MODE_PRESS)、さわさわセンサ機能(MODE_SENS)を実行する。

2.2. marker.cc

マーカ検出処理を行う。各フレームで検出されたマーカは時系列に追跡される。これによりマーカの現在位置(now_pts)に加え、それに対応付いた基準位置(base_pts)、短時間平均位置(ave_pts)を得る。

3. 処理概要

3.1. 全体の流れ

全体の流れは図 3 のフローチャートの通り。フレームごとにマーカ検出を行い、各フレームのマーカ位置を各機能処理関数(mouseMode, press, sens)に渡す。

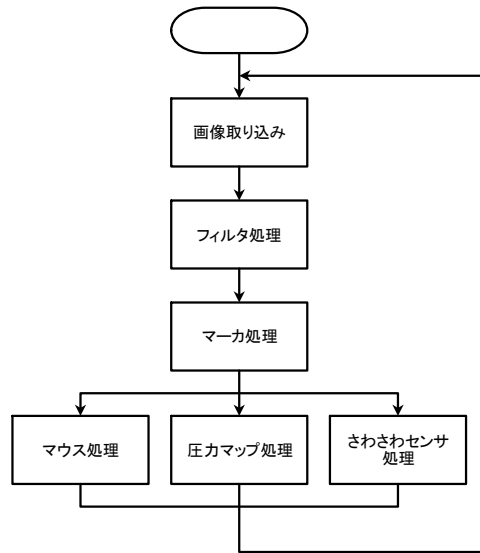


図 3 処理概要

4. フィルタ処理

以下では marker.cc の filter 関数で行われるマーカ検出のためのフィルタ済み画像を生成する工程を説明する。以下の説明で現れる mix,mini,mask はそれぞれ画像を格納した変数である。（marker.cc 参照）

4.1. グレイスケール変換

入力された VGA カラー画像をグレイスケールに変換する(mix)。ただし、青色チャンネルは散乱光成分が多く、マーカと背景のコントラストが低下する要因となるので加算せずに、赤と緑の加算を用いている。赤と緑を加算した結果から青を減算するとさらにコントラストは向上するが、気泡部分で過剰に暗いアーティファクトが発生しマーカと誤認するので本装置では用いていない。

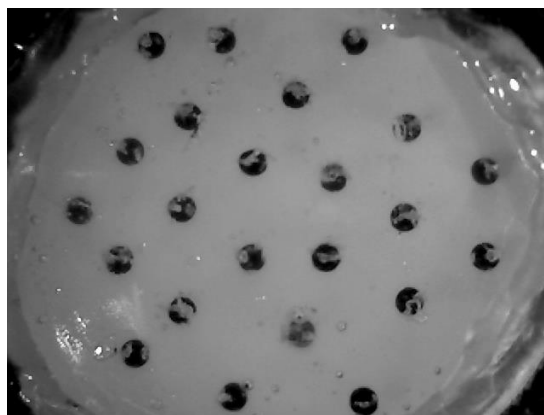


図 4 グレイスケール化

4.2. ノイズ除去

本装置で発生する主なノイズ原は気泡である。特にマーカの上が発生する気泡(図 5)は、マーカ上の輝度をあげる上に、気泡位置が動くので、輝度基準でマーカ中心を探る事を難しくしている。そのため近傍の画素値の最小値を取る erode 関数を用いて気泡を消している。その後平滑化移動平均フィルタ (bulr 関数) を 3 回かけている。bulr 関数を 3 回に分けてかける事でガウスクーネルを 2 次関数近似したものと同様に滑らかに平滑化できる。

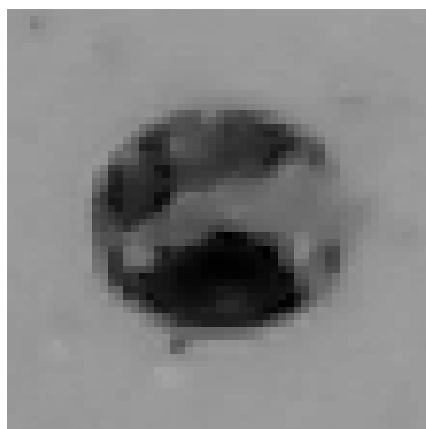


図 5 マーカの上に乗った気泡

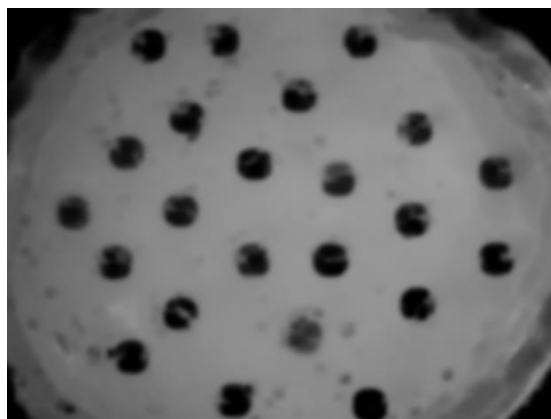


図 6 ノイズ除去

4.3. ダウンサンプル

処理速度を稼ぐためにマーカ検出用画像を 1/8 ダウンサンプルする。この比率は当然ながら画像中のマーカサイズに依存する。以降はダウンサンプルされた画像(mini)についてフィルタ処理を行う。

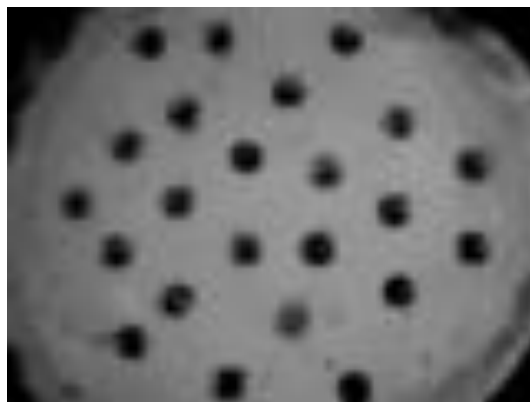


図 7 ダウンサンプル

4.4. 暗ノイズ除去

マーカ上以外に現れた気泡により、小さな暗点が発生するので、近傍の最大値を取る dilate 関数を用いてその暗ノイズを除去する。

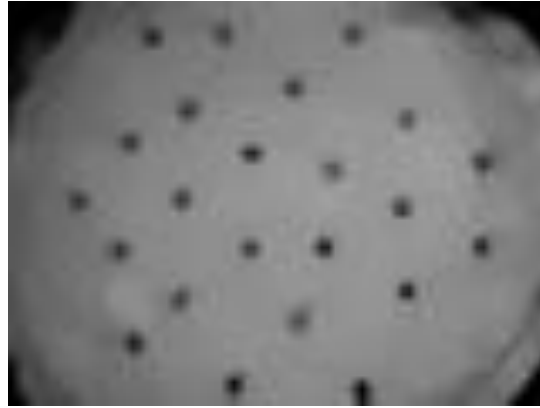


図 8 暗ノイズ除去

4.5. 背景除去

背景(mask)を推定し、除去する。ゲルの周りは黒く塗装してあるので、マーカよりも大きなサイズの黒い塊を除去すれば良い。そのため erode を用いてマーカを消した後に、dilate を実行して余裕をもって大きな黒い塊を背景として抽出する。その跡処理対象画像(mini)から背景(mask)を減算することでマーカのみが暗く表示される。さらにアンシャープマスクを適用することで光量変動をキャンセルし、マーカを際立たせる。

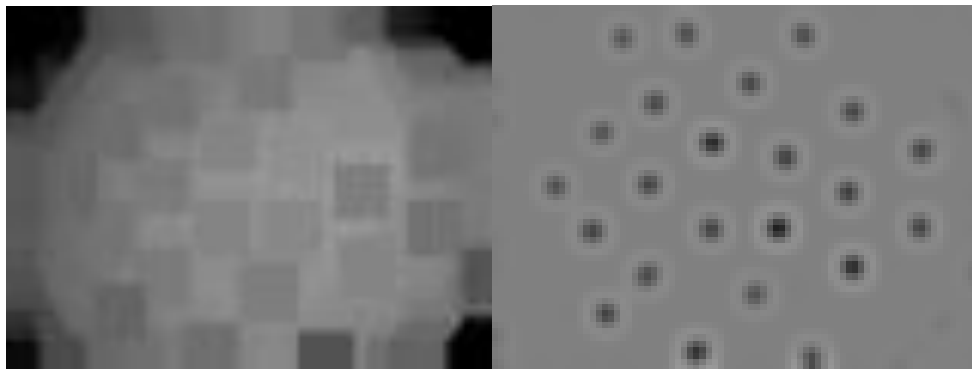


図 9 推定背景（左）背景除去済み（右）

4.6. アップサンプリング画像生成

以上の処理で mini の暗転（極小点）を探せばマーカが見つかるが、分解能を高めるため、アップサンプリング（resize 関数）を行い mix に代入している。当然ながらアップサンプリングしなくても、mini を用いてサブサンプリング単位で極小点を求めても良いが、簡便な処理でマーカ処理を実装するためにアップサンプリングを用いている。

5. マーカ処理

5.1. マーカ検出処理

マーカのうち 1 個を検出するには単純にフィルタ済み画像(mini)の最小輝度点を取れば良い(変数 minLoc)。2 個目については、1 個目の画像近傍を明るい色でぬりつぶした後に最小輝度点を探せばよい。このようにして n 個のマーカを探し出す図 10。実際にはノイズを誤ってマーカとして識別した場合の余裕をもって、マーカ数より 2, 3 個多い数のマーカを探索する。

この mini のマーカでは解像度が低く、やさしく撫でる等の動作が検出できない。そこで mini で検出したマーカ点近傍の中で最小輝度値をアップサンプリング画像から探す(変数 minLoc2)。

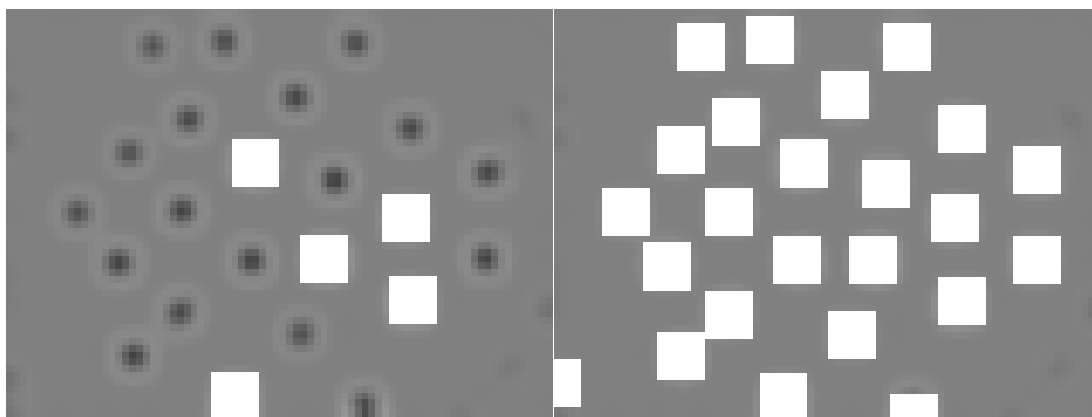


図 10 検出済み部分塗りつぶし (左:4 点 右:全点)

5.2. 過去マーカ対応付け処理

過去マーカとの対応付けは updateMarker 関数によって行われる。本関数の中では過去から引き継がれた基準位置(base_pts)と時間平均位置(ave_pts)が存在しそれぞれのインデックスは 1 対 1 で対応している。一方マーカ検出によって得られた now_pts はそのフレームでの暗い順に並んでいるので、base_pts とは一対一には対応付いていない。そこで base_pts の配列インデックスを now_pts での配列インデックスに変換する対応表(tarid)を作成するのがここでの主な目的である。updateMarker 関数では、この対応表を算出すると同時に、時間平均の算出や、base_pts の修正を行う。

対応付け処理

ave_pts に登録された各点からみて最近傍の now_pts 内の点がそれぞれ対応付いているとする。ただしこの手法で行くと、一つの now_pts に複数の ave_pts が対応尽く場合があるので、複数に対応付いている now_pts については距離が近い方のみを残して他の対応を削除する。

この処理によって得られる配列 tarid は、base_pts(or ave_pts)の配列引数を受け取り、now_pts に用いる配列引数を算出する変換表とみなせる。変換後の値が-1 になるときは、対応するマーカが検出されなかった事を示す。

base_pts 修正

base_pts の修正はカメラ or マウスパッドがずれた場合の自動キャリブレーションとして実装している。修正は操作をしていない時のみに実行するため、まず ave_pts と now_pts の差分の2次ノルムを算出する。2次ノルムが一定(1.5)以下の時は一次 IIR フィルタを通して平均的な base_pts へと修正する。

ave_pts 算出

基本的には now_pts に IIR フィルタをかけて平均的な座標を求めて、ave_pts の値としている。現在の実装では過去値の重み 0.65 にしているが、カメラの fps によって調整されるべきである(fps が高い場合は過去値重みもわずかに高い)。また対応点がない場合は対応点を見失っている可能性があるので、ゆっくりと base_pts へと修正する(過去値重み 0.95)。base_pts に戻る事でユーザが操作をやめた時にマーカ追尾を再開できる可能性が高まる。

6. マウス処理

mouseMode 関数ではマーカの移動方向にマウスカーソルを動かす。マーカの移動方向には単純に now_pts(ソースでは tmp_pts と記述)から base_pts を引いた値の平均値を利用している。同時に押し込み量の尺度として、標準偏差(std)を算出している。平均値の2次ノルムにくらべ標準偏差が十分に大きい場合はクリックと判断できる。ただし現在クリック機能は実装していない。

7. 圧力マップ処理

press 関数では 50pix 間隔の格子点上での圧力を表示する。ここでの圧力は厳密な意味での圧力ではなく、実際にはマーカ点密度である。マーカ密度が低くなると圧力が高いと判断する。マーカ密度はマーカ移動方向の差分から求める。press 関数では getDirec で各格子点でのマーカ移動方向と 5pix ずれた点のマーカ移動方向をとり、その差分を利用して基準状態からの密度変動量としている。基準状態の密度は一定とみなしている、すなわち暗にゲルに埋め込まれたマーカがほぼ均一な分布で配置されている事を期待している。getDirec では各マーカの移動方向を補完して各格子点での移動量を推定している。補完にあたっては格子点からの距離に応じてガウス関数重みをかけて平均値を取っている。ここでガウス関数のσ幅は R で指定されているが、この R はゲルに埋め込んだマーカの平均間隔に比例させて設定するべきである。

8. さわさわセンサ処理

sens 関数は優しくタッチした事を検出する。タッチに関わる場所はゲルの中心部分である事が多いので、ase_pts 座標を見て画像中央に近い nofCenter 個のマーカを選んで利用する。

次に選んだマーカの平均移動量を高周波成分と低周波成分にわけて取得し、低周波成分が小さいのに高周波成分が大きい事をもってタッチしていると判定する。高周波成分としては now_pts から ave_pts を引いた平均値を用いる。低周波成分としては ave_pts から base_pts を引いた平均値を用いる。高周波・低周波成分の2次ノルムを記録しておき、過去5フレームの値を使って判断する。高周波は5フレームの平均、低周波は過去5フレームの最大値を用いて高周波成分が0.2より大きく、低周波成分が2未満の時にタッチされたと判断している。

9. インスパイア元について

ゲル+光学マーカという発想は i-tokyo で見た「GelForceⁱⁱⁱ」がきっかけです。「おっぱい」と「ゲルを用いた触覚センサ」を融合させるというアイディアについては「光弾性と LCD を用いた柔らかいタ

タッチパネル^{iv}」の展示を見学した際に聞いたアイデアでした。本作品はこの二つを組み合わせ、さらに物理的定量値の精度は重要でなく、操作者が「やろうとした事」を判断するという事が重要なのではないかと提案するためにさわさわセンサ機能を追加しました。

ⁱ <http://www.nicovideo.jp/watch/sm22397923>

ⁱⁱ <https://drive.google.com/folderview?id=0B2MvWvAFndBdOTRfWXpqMDgwa2c>

ⁱⁱⁱ <http://tachilab.org/modules/projects/gelforce.html>

^{iv} <http://www.youtube.com/watch?v=t516V6ml4hU>