平成23年度

筑波大学情報学群情報科学類

卒業研究論文

題目
モダンテクニックの動きを取り入れた
抽象的表現を可能にする描画インタフェース

主専攻　　情報システム主専攻

著者　　枝松 ちさと

指導教員　高橋伸 志築文太郎 三末和男 田中二郎

要　旨

近年，コンピュータを利用したデジタル絵画の制作が普及しており，一般ユーザからプロに至るまで盛んにペイントソフトによる制作が行われている．これらのソフトに搭載される多くの機能は，実際の絵画手法を再現しており，ユーザはアナログ絵画を制作するのと同じように，制作に必要な要素を調整しながらデジタル絵画を制作することが出来る．しかしながら，既存のペイントソフトでの抽象的な表現に関しては，既に用意されているテクスチャをマウスなどで変形し，キャンバスに貼り付けているという手法であることがほとんどである．このため，制作される絵画は単調なものになりやすく，また実際の絵画手法を再現できていないという問題点がある．

本研究では，この問題を解決するために実際に抽象的な表現を行う際の動きを取り入れたハンドジェスチャによる描画を可能にするインタフェースを提案した．そして，この提案を基にしたシステムの実装を行った．これにより，ユーザは手の位置を調整し，ハンドジェスチャを行うことで，その位置や動きに応じた変化のある抽象的な表現が可能となる．

目 次

図 目 次

第1章　序論

1.1　研究の背景

1.1.1　デジタルペイントソフトでの絵画制作手法

現在，デジタルペイントソフトを用いたデジタル絵画の制作が，一般ユーザからプロのデザイナーに至るまで盛んに行われている．一般によく利用されるデジタルペイントソフトの例としては，Painter[1]，SAI[2]，Pixia[3] などが挙げられる．

上で述べたこれらのソフトには，線を描く，色を塗るなどの基本的な描画機能や，水彩，油彩などの絵具媒体を選択できる機能などが搭載されている．これらの機能は実際の絵画手法を再現しており，描画時の動作も実際の絵画制作時のものに則している．

例えば，線を描くにはストロークと太さという 2 つのパラメータがある．制作者はこれらを調整しながら，キャンバスに自分のイメージしたものを描いて制作を行う．実際の制作においては，入力デバイスを筆とした場合，制作者は画筆の筆跡でストロークを，キャンバスにどれだけ画筆を押し付けるかで太さを調整して制作を行っている．これをデジタルペイントでの制作にあてはめてみると，入力デバイスをスタイラスとした場合，画筆の筆跡をスタイラスの軌跡に，画筆の押し付け具合を筆圧センサの情報に置き換えられており，制作者は実際にアナログ絵画を描くのと同じように制作することが出来る．

このように，デジタルペイントソフトに搭載されている機能を用いて描画する際でも，実際の動作に則しているため，ユーザはアナログ絵画を制作するのと同じように，描画に必要なパラメータを調整しながらデジタル絵画を制作することができる．

1.1.2　モダンテクニックを用いた実際の絵画制作

線の描画などの基本的な技法以外に，実際の絵画の制作には表現手段として様々な描画技法がある．その中でも，抽象的なデザインや表現を行うための技法としてモダンテクニックというものがある．これは，単に人物画や静物画のような具体的な物を模写する絵画を制作する際に用いる技法ではなく，偶然生じる模様や不規則な形を利用して絵画を制作する技法である．

---

[1] Corel Painter (URL : `http://www.corel.com/`)
[2] Easy Paint Tool SAI (URL : `http://www.systemax.jp/ja/sai/`)
[3] Pixia (URL : `http://www.pixia.jp/`)

モダンテクニックが偶然性という特徴をもつ一方で，どの位置に配置するのか，どの方向に描画を行うか，あるいはどの程度の範囲に描画するのかなどは制作者に委ねられる．このため，ある程度の必然性も持ち合わせており，制作者の表現意図もある程度反映されるという特徴もある．また，この技法は抽象表現のみの絵画だけでなく，具体的な物が描かれた絵画にニュアンスを出したり，変化を加えるために用いられることもある．

1.1.3　ペイントソフトでのモダンテクニックの表現に関する問題点

ほとんどのデジタルペイントソフトでは 1.1.2 項で述べたモダンテクニックや他の抽象表現を，あらかじめ準備されたテクスチャを選択し，キャンバスの任意の位置でスタイラスをタップすることで行っている (図 1.1)．しかしながら，このような手法にはいくつかの問題点が存在する．

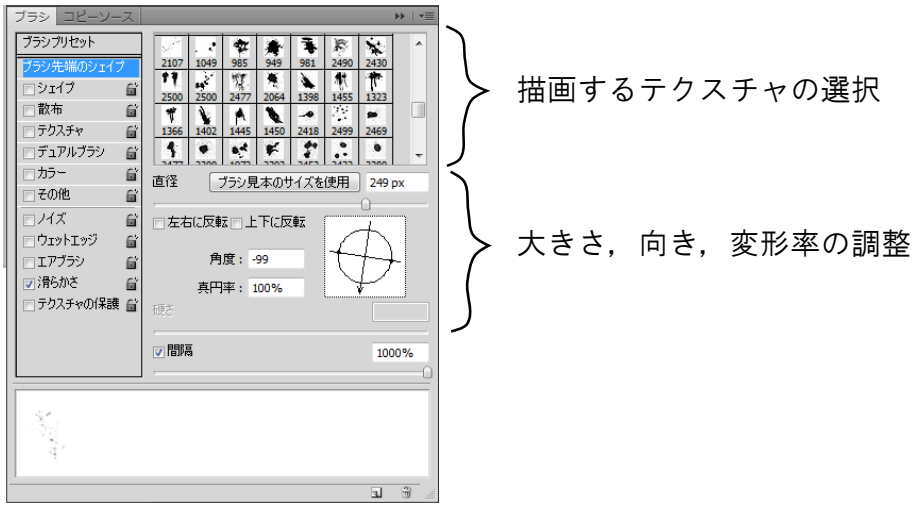


図 1.1: デジタルペイントソフトでのモダンテクニックの実現例

まず一つ目は，この操作によって制作される絵画は単調なものになりがちであるということである．すなわち，スタンプを何度もキャンバスに押したような描画になるため，変化を出しづらいという問題が挙げられる．先にも述べたが，これはテクスチャベースでの手法であることに起因している．

二つ目に，偶然性を活かした描画であるとは言えないということである．描画の方向および範囲は，図 1.1 のようにスライダによる拡大縮小，具体的な数値を入力して角度の指定を行うことによって決定されている．ユーザの意図を反映するという観点から見れば，任意の向きやサイズを指定すれば描画が可能であることは利点であるが，モダンテクニックの特徴である偶然性が活かされていないと考えられる．

三つ目に，実際の制作における動作という視点では，スライダによる調整は不自然であるということが挙げられる．このため，アナログ絵画と同じように描画に必要なパラメータを

調整し，デジタル絵画を制作できていないと考えられる．

1.2 本研究の目的

本研究では，モダンテクニックに必要なパラメータを考慮した，変化のある描画を可能にするインタフェースの提案を目的とする．そして，アナログ絵画を制作するのと同じように，制作者の動作から表現に必要なパラメータを調整して，デジタル絵画を制作できることを目指す．

1.3 アプローチ

本研究のアプローチとして，モダンテクニックに必要な要素に“高さ”を追加する．本研究では，この要素を追加するために，実際に描画を行う際の動きを取り入れた空中でのハンドジェスチャによって描画を可能にするインタフェースを実装する．これにより，ユーザは実際の絵画制作を行うのと同じような操作でモダンテクニックを用いたデジタル絵画を制作することが可能になる．さらに，操作の位置を調整することで，ユーザはその調整に応じた変化のある描画を行うことが出来る．

1.4 本論文の構成

本論文は本章を含め 6 章で構成されている．第 2 章では関連研究について述べる．第 3 章では，インタフェースを検討するにあたり，モダンテクニックの技法を用いて実際に制作を行う．そして，この制作過程から考察を行い，この考察に基づいたインタフェースの検討を行う．第 4 章では開発したシステムの実装について述べる．第 5 章ではそのシステムに関する評価および議論を行う．最後に，第 6 章で結論を述べる．

第2章　関連研究

本章では本研究と関連のある研究として，描画を行うための入力手段に関する研究，描画のフィードバックに関する研究，および偶然性を活かした描画手法に関する研究の 3 つの分野について述べる．

2.1　描画を行うための入力手段に関する研究

Vandoren らのシステム [1] では，描画を行う入力手段として実際の画筆を用いている．このシステムにおいて，ユーザはまず画筆を水に浸し，テーブルトップ上に表示されたカラーパレットで混色を行う．その後，同テーブルトップに表示されたキャンバス上で描画を行う．このシステムでは，FTIR 方式を用いて水に濡れた画筆とキャンバスの接触を検知している．このため，丸筆や平筆といった画筆の種類や，画筆に含ませた水分量に応じた変化のある描画が行われる．これにより，ユーザは実際に水彩画を制作しているかのような体験ができる．岩井らによる Thermo Painter[2] は，熱画像を用いたタブレット型入力装置を用いた描画システムである．この研究では描画の入力手段として温度を用いた表現手法を提案しており，体温などを利用して描画を行えるシステムとなっている．Hornof らによる EyeDraw[3] は，視線の動きをトラッキングすることで絵を描画できるシステムである．また，Harada らの Voicedraw[4] は，声のみで描画を行うシステムである．描画機能はコマンドを話すことで選択し，カーソルやストロークの動きは，母音[1]を上下左右などの方向に割り当て，動かしたい方向の母音を発音することで制御する．また，視線と声の組み合わせで入力を行うシステムもある. Kamp らは，カーソルやストローク操作を視線の動きで行い，描画の開始/終了などの機能の選択を音声で行う描画システムを開発した [5]．Chen らが開発したシステム [6] では，両手を入力手段として用いる．テーブルトップ上に両手を置き，左手のジェスチャ操作でメニューを選択し，右手で描画を行う. Horace らの Body-Brush[7] は，人間の身体そのものをブラシとして扱い，描画を行うシステムである．このシステムは，赤外線照明および赤外線カメラで 3 次元空間における身体の動きをキャプチャし，そこからユーザの動きや姿勢を分析して，動きに同期した描画を行っている．

本研究では，特別に道具を用いることはせず，片手の動きを入力手段として扱う．また，[3][4][5] の研究は，身体が不自由な障害者に焦点を当てたシステムである．これに対し本研究で提案するシステムは，マウスなどによる数値調整での抽象表現を行うことを苦手とするユーザに焦点を当てており，本研究とは異なる．

[1] cat の “a”，law の “aw”，feet の “ee” など．

2.2 描画のフィードバックに関する研究

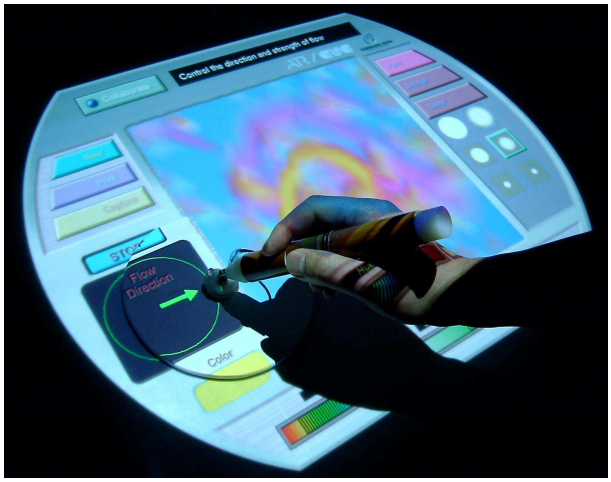


図 2.1: Sumi-Nagashi

櫻井らによる Sequential Graphics[8] は，描画時の臨場感を再現するペイントソフトである．このシステムは目の前で絵画を制作する様子を見せるライブペインティングのように，視覚的に制作過程の臨場感，および制作者の心景を感じ取ることを目指している．また Yoshida らは，日本の伝統的な絵画手法である “墨流し”をモチーフとしたデジタルペイントツールを開発した [9]．このシステムでは，逐次的に変化する絵画の流れを視覚的に得ることが出来る．さらに，色に “重さ” という概念を定義し，この感触を力覚ディスプレイである Proactive Desk[10] を用いることでユーザに提示している (図 2.1)．一方，描画シミュレータである DAB[11] や Gregory らの研究 [12] は，力覚ディスプレイである PHANToM [2] を利用することによって，ユーザへのフィードバックとして実際にキャンバスと筆が接触しているような感覚を提示している．

これらのシステムは全てペンや力覚ディスプレイによる描画を行っているが，本研究ではこのようなデバイスを用いた描画は行わない．また，これらのうち [9][11][12] はキャンバスと画筆との接触を想定した力覚フィードバックを提示する研究となっており，キャンバスと接することなく描画を実現する本研究とは異なる．

2.3 偶然性を活かした描画手法に関する研究

Lee ら [13] は，絵具を垂らして描画を行う技法を，流体シミュレーションによって再現するペイントソフトを開発した．Ryokai らによる I/O Brush[14] は，筆型デバイスに取り付けられたカメラでキャプチャした画像を “インク” として扱い，描画を行うペイントソフトである

[2] Sensable PHANToM (URL : `http://www.sensable.com/haptic-phantom-desktop.htm`)

図 2.2: I/O Brush

(図 2.2)．また，草地らによる Roll Canvas[15] は，キャンバスに回転をもたせることによって描画にある程度の偶然性を導入した．ユーザが意図して描いたストロークに回転が加わることで，半主体的な絵が生成される．

I/O Brush は，日常の風景がインクとなっており，絵具のように単色をインクとして扱う本研究とは異なる．Lee らのシステムは，絵具を滴らせるという技法に着目した点では本研究と類似するが，RollCanvas と同様，ユーザの操作で扱うパラメータが x, y の 2 次元のみとなっており，z 軸の情報を加えた本システムとは異なる．

第3章　インタフェース設計

本章では，まず複数あるモダンテクニックのうち，本研究で取り扱う技法について述べる．次に，これらを用いて実際にアナログによる制作を行ったことについて述べる．そして，その制作過程について考察を行い，これを基にしたインタフェースを設計する．

3.1　本研究で取り扱うモダンテクニック

モダンテクニックは，具体的な絵を制作するときに用いる技法と比べ，制作者の技術や能力を必要としないことが特徴として挙げられる．すなわち，ほとんど誰でも行える動作であり，幼児から大人まで幅広い層がこの技法を扱うことが可能である．モダンテクニックの代表的な技法としては，ドリッピング [16][17]，スパッタリング [17]，マーブリング [17]，デカルコマニー [18]，フロッタージュ[19] などが挙げられる．

本研究ではこれらのモダンテクニックのうち，絵画制作でよく用いられるドリッピング，スパッタリングの 2 つの技法に焦点を当てる．以下に各技法の詳細について述べる．

3.1.1　ドリッピング

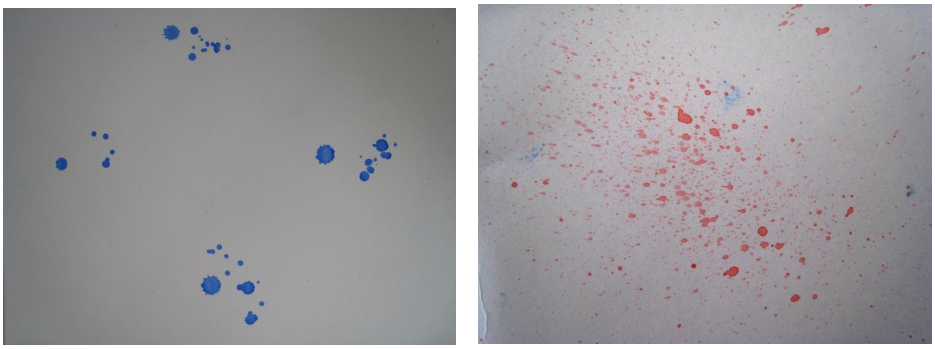
ドリッピング (dripping) は，床や机の上に置いたキャンバスに向かって絵具を含ませた画筆を空中から振り下ろし，絵具を滴らせることによって描画を行う技法である．これにより図 3.1(a) に示すような円状のしみのようなものが散っている絵が得られる．この技法は，小中学校の美術教育で利用されることが多い．

この技法を用いて絵画を制作した代表的な画家に Jackson Pollock[20] が挙げられる．Pollock は，ドリッピングに加えて絵具を撒く技法 (ポーリングと呼ばれる) を用いて，数多くの抽象絵画を生み出した．また，制作している様子を観客に見せるライブペインティングを行う際に用いられることもある．

3.1.2　スパッタリング

スパッタリング (spattering) は，床や机の上に置いたキャンバスに向かって絵具を含ませた画筆やブラシを空中から指ではじき，絵具のしぶきを飛ばすことによって描画を行う技法である．しぶきを散らすには，指以外にも筆の柄ではじく方法や，金網でブラシをこする方法もある．これにより，図 3.1(b) に示すような細かい点が無数に散らばった絵が得られる．こ

の特徴を活かして，風景画における砂や草花が群生している様子などを表現することもできる[1]．また，ドリッピングと同様に小中学校の美術教育や，水彩，アクリル，油彩，木製品などに絵具を塗るトールペイントなどで利用されている．



(a) ドリッピング　　(b) スパッタリング

図 3.1: それぞれの技法で得られる描画

3.2　予備実験

本研究でインタフェースを提案するにあたり，実際に 3.1.1 項，3.1.2 項で述べた技法を用いて制作を行う予備実験を行った．以下に実験内容と結果，および考察について述べる．

3.2.1　実験内容

およそ 5cm，15cm，30cm の高さから，それぞれドリッピングおよびスパッタリングを用いて制作を行う．その際，それぞれの技法の動作にはどのような特徴があるのか，また高さによってどのように描画に違いが生まれるのかを調べる．なお，制作の様子はデジタルカメラの動画撮影で記録を行った．

道具には以下のものを用いた．

- 絵具
  - サクラクレパス マット水彩 (12ml) あお [2]
  - サクラクレパス マット水彩 (12ml) しゅいろ [2]
- 描画を行う道具

[1]初心者も描ける アクリル絵の具の使い方を写真で解説
URL:http://flower77777.blog95.fc2.com/blog-entry-46.html
[2]URL:http://www.craypas.com/products/lineup/detail/36.php

- スポイト (ドリッピングの高さによる違いを調べるために使用)
- 画筆 15 号 (ドリッピングに使用)
- 市販されている歯ブラシ (スパッタリングに使用)

- キャンバス
  - マルマン 図案シリーズ スケッチブック S120 B4 画用紙 並口[3]

3.2.2 実験結果

以下に示す図は，実際に制作した絵の一部である．まず，高さによる円の半径の違いを調べるためにスポイトで一定量の絵具を落とした．この結果の一部が図 3.2 である．次に，ドリッピングの動作を行ったときに得られた結果の一部を図 3.3 に示す．そして，スパッタリングの動作を行ったときに得られた結果の一部を図 3.4 に示す.

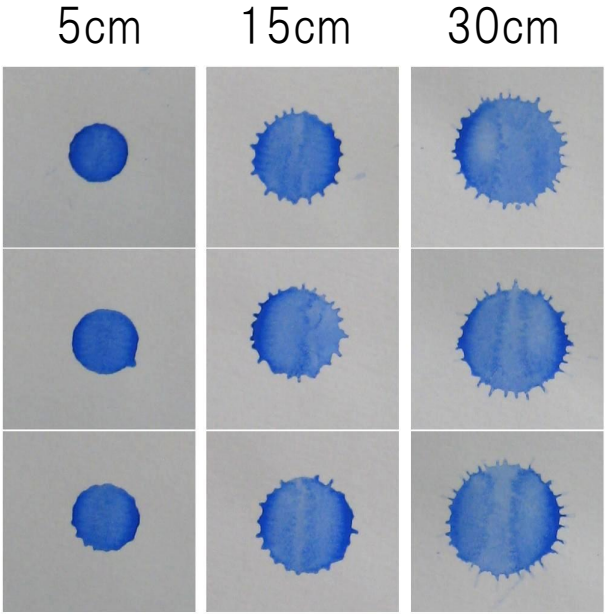


図 3.2: スポイトにおける高さの違いによる描画の違い

3.2.3 考察

3.2.2 項での結果から，以下ではそれぞれの技法に関する考察について論じる．また，動作についての考察についても述べる．

[3]URL:http://www.e-maruman.co.jp/products/sketchbook/zuan_book/

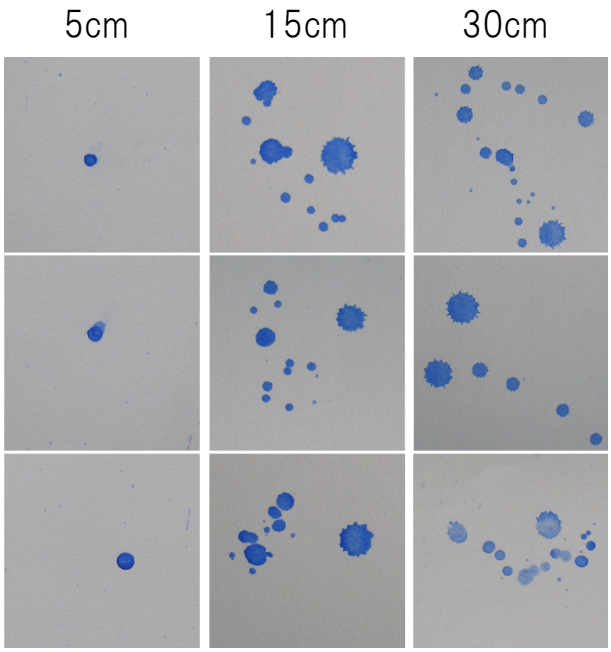


図 3.3: ドリッピングにおける位置の違いによる描画の違い

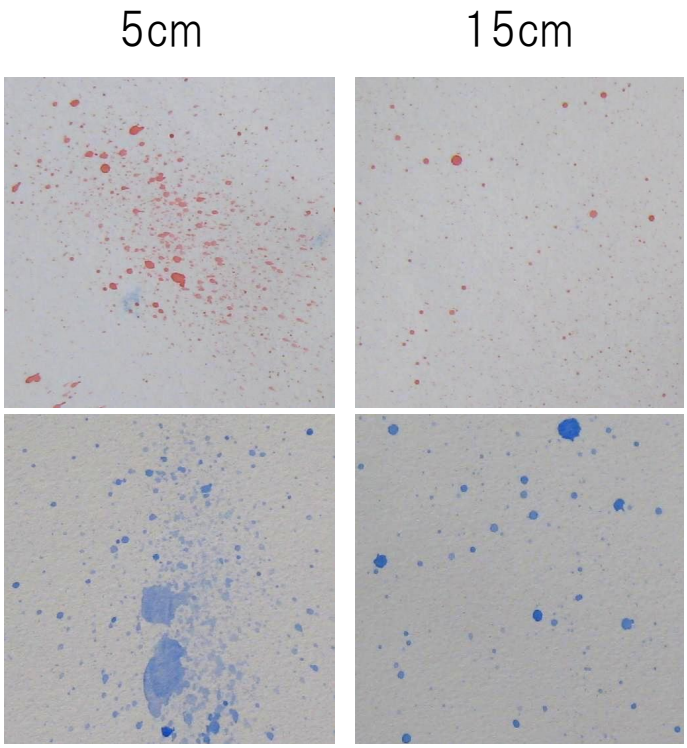


図 3.4: スパッタリングにおける位置の違いによる描画の違い

ドリッピング

- 図 3.2 より，この動作を行う位置が低ければ低いほど描画される円の直径は小さくなる．逆に，この動作を行う位置が高ければ高いほど，描画される円の直径は大きくなる．
- 図 3.3 より，この動作を行う位置が低い場合，描画される円は少ない．逆に，位置が高い場合，描画される円の数は多い．
- 図 3.3 より，この動作を行う位置が高ければ高いほど，描画される円の間隔が広くなる．
- ドリッピングの動作は，キャンバス面に対して垂直方向 (z 軸方向) に行っており，キャンバス面に対して水平方向への動き (x, y 軸方向) はそれほど見られない．

スパッタリング

- 図 3.4 より，この動作を行う位置が低ければ低いほど描画される範囲は狭くなる．逆に，この動作を行う位置が高ければ高いほど，描画される範囲が広がる．
- 図 3.4 より，描画される細かな円の半径は均一ではない．
- この動作を行う勢いが強ければ強いほど，描画される範囲が広がる．勢いが弱い場合には，描画される範囲が狭くなる．
- スパッタリングの動作は，キャンバス面に対して水平方向 (x, y 軸方向) に行っており，キャンバス面に対して垂直方向 (z 軸方向) はそれほど見られない．

3.3　インタフェースの設計

本節では，3.2 節で論じたことを基に，本研究で提案するインタフェースの設計について述べる．まず，ユーザはどのような操作をするのかについて述べ，次に描画に変化をつけるためにユーザはどう操作を変えれば良いかについて述べる．

3.3.1　ユーザが行う操作

本研究で提案するインタフェースでは特別な道具は用いず，手の動きを操作として扱う．それぞれの操作は，実際の技法の動きと近くなるようにするのが良いと考えた．これは，実際の動きに似ている方がユーザにも覚えやすい操作になり，なおかつアナログ絵画の制作と同じように制作できると考えたからである．以下で各技法でのユーザの操作について述べる．

ドリッピング

本研究で提案するインタフェースにおいて，ドリッピングを用いた描画を行うには，ユーザは描画したい位置で手を空中からキャンバスに向かって真下に振り下ろす動作を行えば良い．本システムでのドリッピングの動作イメージを図 3.5 に示す．

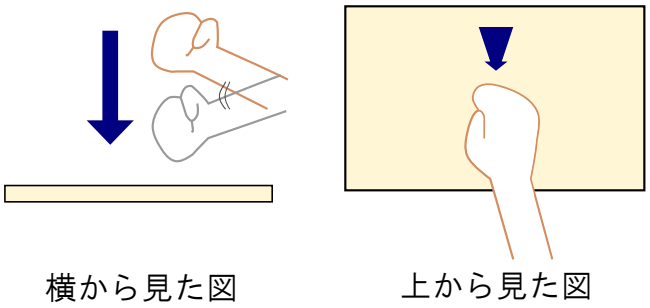


図 3.5: ドリッピングの動作イメージ図

スパッタリング

本研究で提案するインタフェースにおいて，スパッタリングを用いた描画を行うには，ユーザは描画したい範囲にキャンバスと平行に手を振る動作を空中で行えば良い．本システムでのスパッタリングの動作イメージを図 3.6 に示す．

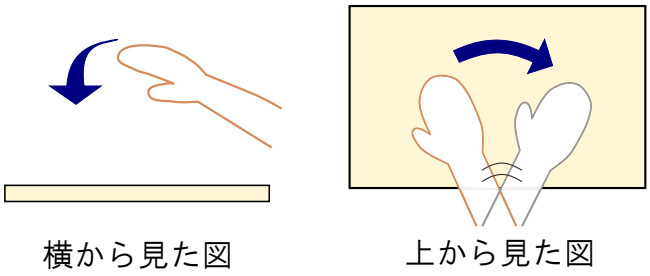


図 3.6: スパッタリングの動作イメージ図

3.3.2　操作位置の違いによる描画の違い

ユーザが高さや移動距離を変えてそれぞれの技法の操作を行えば，その高さや移動距離に応じた変化のある描画が可能となる．描画に変化をもたらすために，ユーザはどのような操作の変化を加えれば良いのかについて以下に述べる．

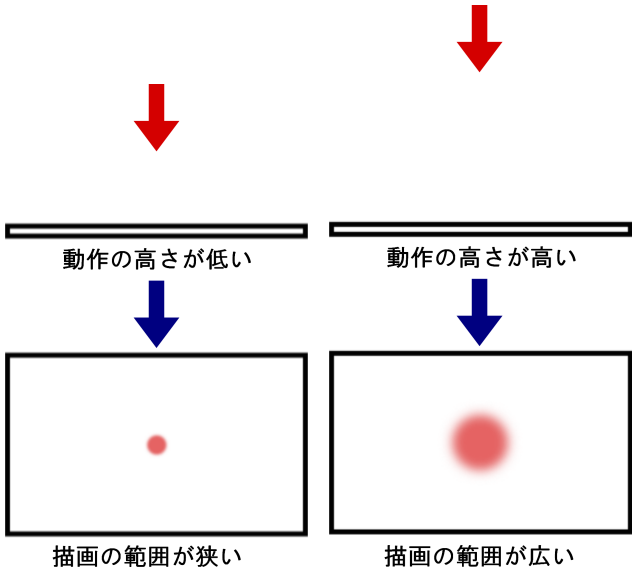
ドリッピング

ドリッピングでの描画において，ユーザが狭い範囲で描画したい，あるいはあまり多くの円を飛ばしたくないと考えた場合，低い位置でドリッピングの操作を行えば良い．逆に，広い範囲に描画したい，あるいは多くの円を飛ばしたいとユーザが考えた場合には，高い位置でドリッピングの操作を行えば良い．操作の位置に応じて描画範囲が異なるイメージ図を図 3.7 に示す．

スパッタリング

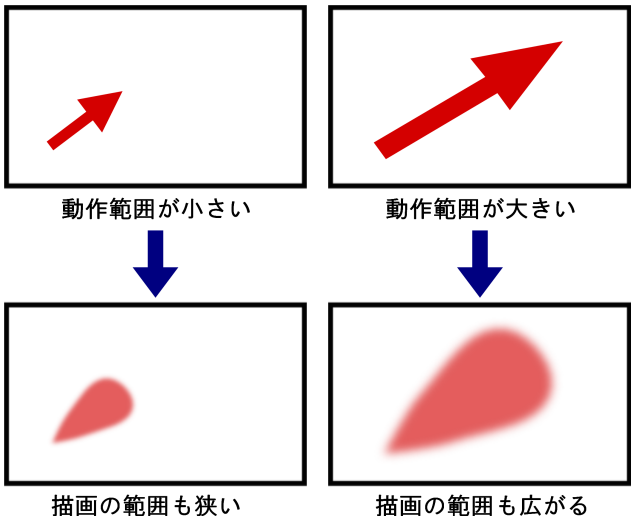
スパッタリングでの描画において，ユーザが狭い範囲で描画したいと考えた場合，スパッタリングの操作を短い距離で行えば良い．逆に，広い範囲に描画したいと考えた場合には，ユーザはこの操作を長い距離で行えば良い．

また，どの方向に向かって動作を行ったのかをユーザが分かるように，動作の始点から終点に向かって扇状に描画が広がるようにする．例えば，ユーザがスパッタリングの操作を下から上に向かって行った場合，描画は下から上へ扇状に広がったものとなる．距離の長さ，操作の方向に応じて描画範囲が異なるイメージ図を図 3.8 に示す．



(a) 位置が低いときの描画範囲 (b) 位置が高いときの描画範囲

図 3.7: 動作の位置に応じて変化する描画範囲 (ドリッピング)



(a) 距離が短いときの描画範囲 (b) 距離が長いときの描画範囲

図 3.8: 動作の距離の長さに応じて変化する描画範囲 (スパッタリング)

第4章　実装

本章では，前章で論じたインタフェース設計を基に，実際に開発したシステムの実装について述べる．

4.1　開発環境

本システムは位置情報解析部と描画部の 2 部で構成されており，各部合わせて 2 台の計算機を用いた．次に各々の開発環境について述べる．

位置情報解析部

位置情報解析部の開発環境として，OS は Windows7 Professional，CPU は Intel Core i7 2820QM 2.30GHz であり，IDE は Visual Studio 2010 を使用した．開発言語には C++を用いた．Kinect センサ[1] からの画像取得および解析を行うためのライブラリとして Prime Sensor 社の OpenNI[2] を利用した．

描画部

描画部の開発環境として，OS は Windows Vista Home Basic，CPU は Intel Core2 Quad 2.50GHz であり，IDE は Visual Studio 2010 を使用した．開発言語には C++を用い，ライブラリとして openFrameworks[3]を利用した．

4.2　ハードウェア構成

本システムのハードウェア構成のイメージ図を図 4.1 に示す．描画を行うキャンバスとして液晶タブレットをディスプレイ面が上を向くように設置する．そして，キャンバス面とユーザの手の動きを認識できる位置にカメラを設置する．実際に本システムを設置した例を，図 4.2 に示す．

[1] Kinect for Xbox 360 (URL : `http://www.xbox.com/ja-JP/kinect`)
[2] OpenNI. PrimeSense Sensor Module (URL : `http://www.openni.org/`)
[3] openFrameworks (URL : `http://www.openframeworks.cc/`)

絵を表示するディスプレイには Wacom 社の液晶タブレット Cintiq 12WX [4]を用いる．カメラは RGB カメラと深度カメラを備えた Microsoft 社の Kinect センサを用いる．Kinect センサは位置情報解析部の計算機，液晶タブレットは描画部の計算機に接続されており，処理は別々に行われる．

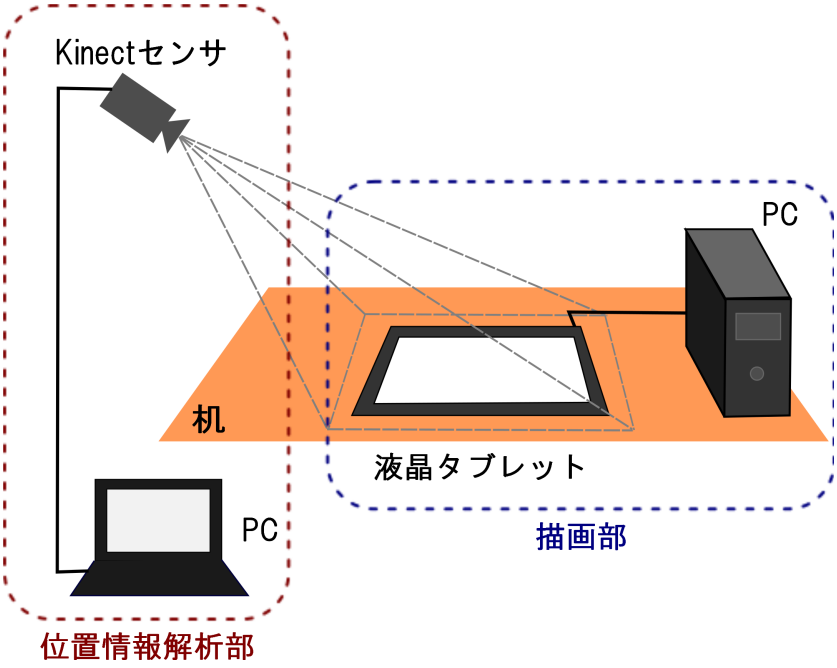


図 4.1: ハードウェア構成のイメージ

4.3 ソフトウェア構成

本システムのソフトウェアは，位置情報解析部，描画部の 2 つの部分から構成されている．処理の流れのイメージ図を図 4.3 に示す．以下で各部の詳細を説明する．

4.3.1 位置情報解析部

位置情報解析部では，キャンバス面に対する手の 3 次元情報を求める．大まかな流れとしては，まずキャンバス面の位置認識を行い，次に手の位置の認識を行う．そして，この 2 つの情報を基に手の位置情報をキャンバス面の座標系へ変換し，描画部へとその情報を送信する．以下に各々の詳細を述べる．

[4]Wacom Cintiq 12WX (URL : `http://wacom.jp/jp/products/cintiq/cintiq12/`)

図 4.2: システムの設置例

キャンバス面の位置認識

本モジュールで得られる RGB 画像および Depth 画像を図 4.4 として示す．Kinect の深度カメラで得られる Depth 画像から，液晶タブレットの画面内における四隅の点を選択することで，キャンバスとなる位置を認識する．ロバストな点の選択を可能にするために，再帰性反射材を貼付した調整用の用紙を作成した (図 4.5)．この用紙の四隅の角と液晶タブレットの画面の四隅の角を合わせて用紙を乗せ (図 4.4(a))，反射して投影される角 (図 4.4(b)) をクリックして選択する．選択し終えたら，用紙を液晶タブレットから外す．

手の位置認識

本システムでは，ハンドジェスチャの認識による手のトラッキングを用いた．これにより，マーカなどの特別なものを使用することなくトラッキングを行うことが出来る．OpenNI のミドルウェア部を担当する NITE ライブラリに，指定のハンドジェスチャ(以降，フォーカスジェスチャと呼ぶ) を認識し，この動作がトリガとなって任意の処理を行えるものがある．今回の実装では，このライブラリを使用して手の位置のトラッキングを行った．これにより，ユーザがフォーカスジェスチャを行うだけで手の位置を把握することが出来る．

今回，フォーカスジェスチャには“手を振る”動作を登録し，この動作が認識されると手のひらの中心点の座標を返す処理を行うように実装した．手を振る動作の検出には，`XnVWaveDetector`

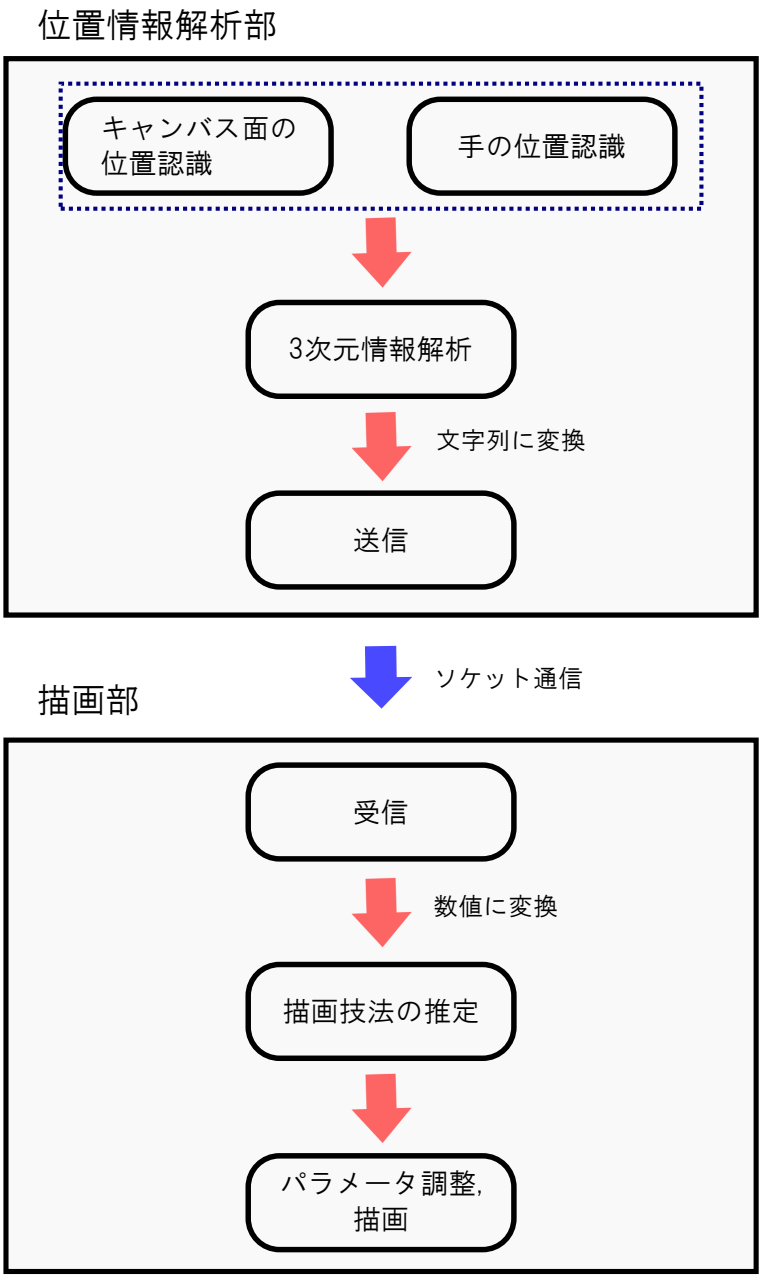
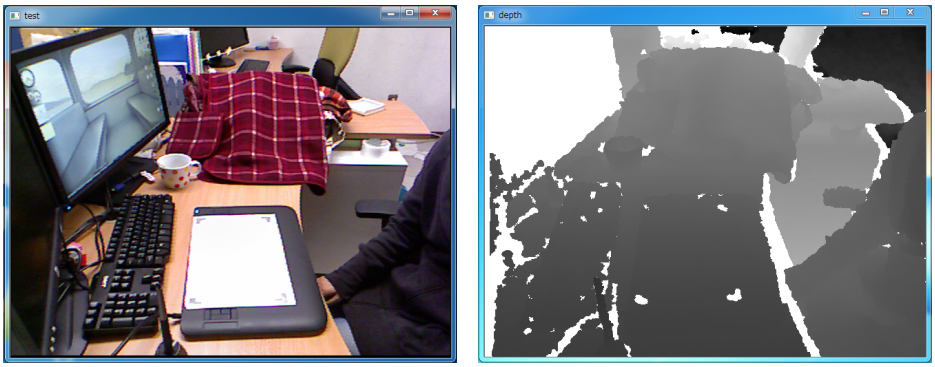
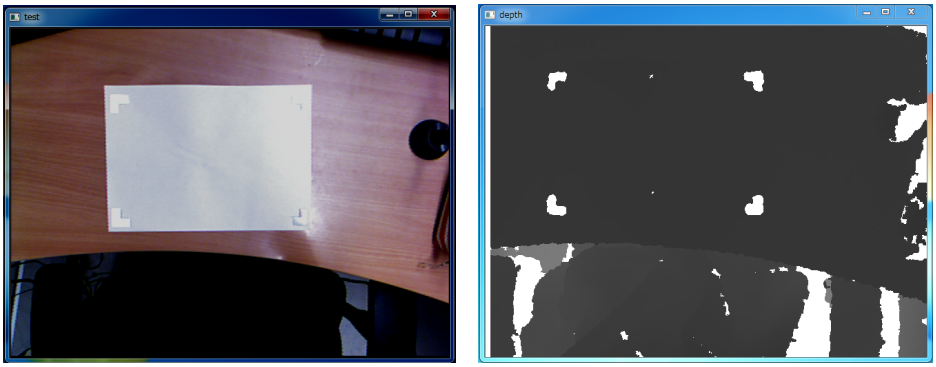


図 4.3: ソフトウェア構成のイメージ



(a) RGB 画像　　(b) Depth 画像

図 4.4: 情報取得モジュールで表示される 2 画像



(a) RGB 画像　　(b) Depth 画像

図 4.5: 調整用用紙

クラスを利用した．手のひらの中心点およびその位置座標は，手を振る動作を認識したときに OpenNI によって算出される．なお，OpenNI が返す位置座標の単位はミリメートルである．図 4.6 の大きな白い点は，算出された手のひらの中心である．

万一カメラが手の位置を認識できなくなった場合でも，手を振るというフォーカスジェスチャを行うことで再び手の位置をトラッキングすることが可能である．

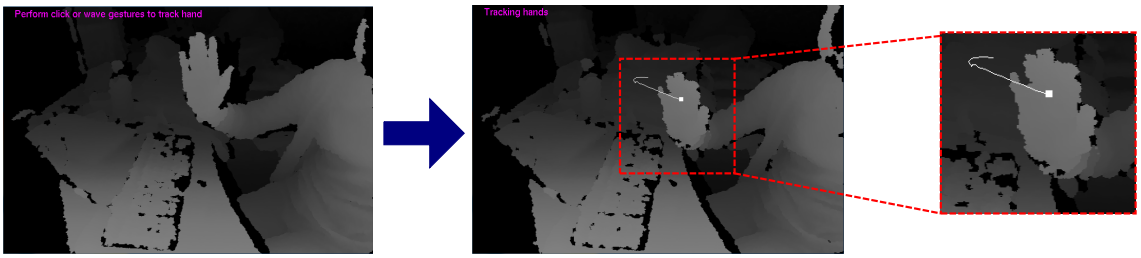


図 4.6: フォーカスジェスチャの認識による手のトラッキングのイメージ

キャンバス面に対する手の 3 次元情報の解析

カメラから得られる手の位置情報を基に，キャンバス面に対する 3 次元情報を計算する．得られるキャンバス面の位置情報および手の位置情報はカメラの中心を原点とする座標系であるため，キャンバス面の左上隅を原点とする座標系での位置を求める必要がある (図 4.7)．キャンバス面からの手の 3 次元位置を求める手順を以下に示す．

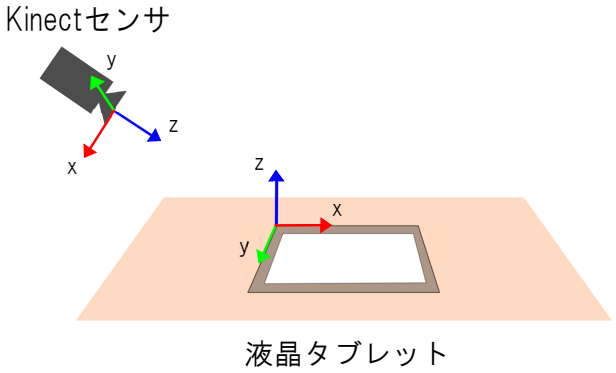


図 4.7: カメラの中心を原点とする座標系とキャンバス面の左上隅を原点とする座標系の違い

1. 法線ベクトルの計算

   図 4.8 のようにキャンバス面の左上隅の座標を P_0，左下隅の座標を P_1，右下隅の座標を P_2，右上隅の座標を P_3 とする．また，手のひらの中心の座標を P とする．P_0 から

P_1, P_3 へのベクトルを $\boldsymbol{v}_0, \boldsymbol{v}_1$ とすると，キャンバス面に対する法線ベクトル $\boldsymbol{n}$ は，$\boldsymbol{v}_0$ と $\boldsymbol{v}_1$ のベクトル積で求められる (図 4.8).

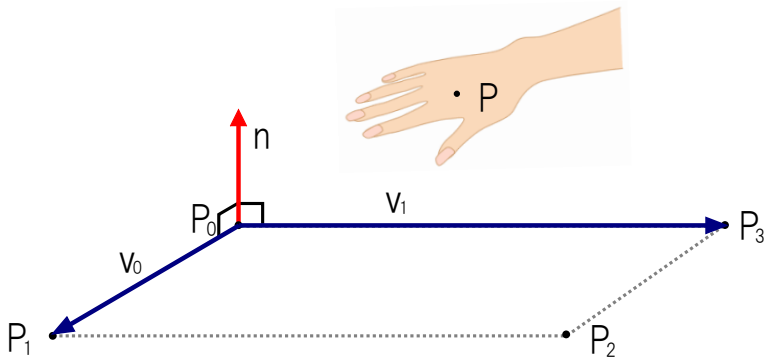


図 4.8: 法線ベクトルの計算

2. 平面の生成
   1 で求めた法線ベクトルを基に，平面を生成する．この平面を H とする (図 4.9).

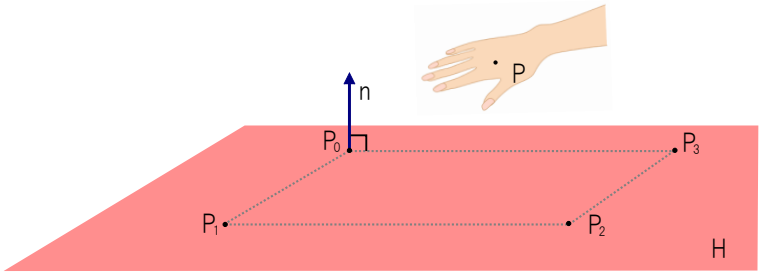


図 4.9: 平面の生成

3. 平面と手の距離，および交点の計算
   手の位置 P から 2 で求めた平面 H に垂線を下ろし，この距離 h を計算する．これがキャンバス面からの高さ，すなわち z 座標の値となる．また，P から H に垂線を下ろした時の交点を I とする (図 4.10).

4. キャンバス面と交点の距離の計算
   $\boldsymbol{v}_1$ を法線ベクトルとし，P_0 を通る平面 H' を生成する．そして 3 で求めた I から H' に垂線を下ろし，この距離 h' を計算する．これがキャンバス面の左隅を原点とした座標系における x 座標の値となる (図 4.11(a)). 同様にして $\boldsymbol{v}_0$ を法線ベクトルとし，P_0 を通る平面 H'' を生成し，H'' と I の距離 h'' を計算する．これがキャンバス面の左隅を原点とした座標系における y 座標の値となる (図 4.11(b)). なお，x, y 成分は距離を求めた後，それぞれ正規化を行っている．これは描画部での処理をしやすくするためである．

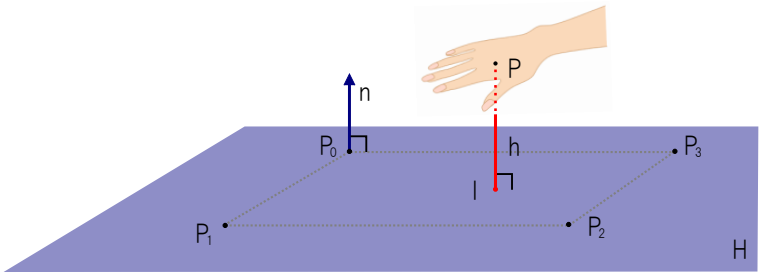
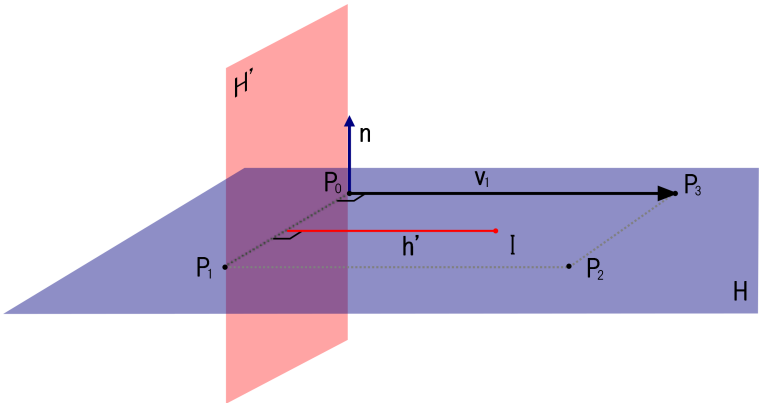
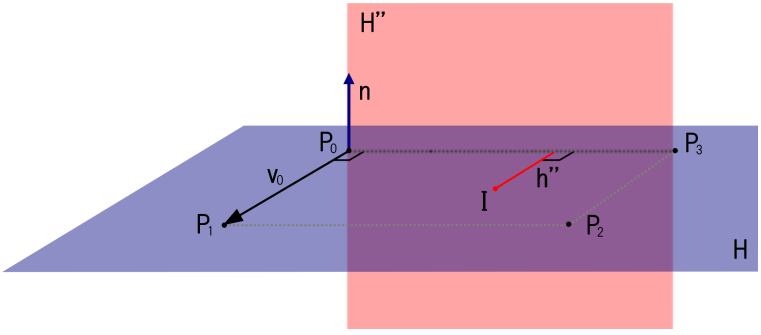


図 4.10: 平面と手の距離，および交点の計算



(a) キャンバス面と交点の x 座標の計算



(b) キャンバス面と交点の y 座標の計算

図 4.11: キャンバス面と交点の距離の計算

位置情報の送信

本システムでは，位置情報解析を実行する計算機と，描画を実行する計算機が異なっている．そこで，本システムでは 2 台の計算機間の通信に TCP/IP プロトコルによるソケット通信を用いた．今回の実装では，位置情報解析部を実行する計算機をクライアントとして構成を行った．解析した情報は数値であるが，サーバ側となる描画部での通信が文字列のみのサポートとなっているため，数値で送信することができない．したがって，解析した情報を文字列に変換した後，サーバへと送信する．描画部での受信の遅れを考慮し，データはフレームごとではなく一定時間ごとに送信している．今回の実装では 65msec おきに送信している．

4.3.2 描画部

描画部では，描画技法や手の位置情報に応じた描画を行う．大まかな流れとしては，位置情報解析部から送られてくる情報を受信した後，その情報を基に描画技法の推定を行う．そして，推定された技法で手の位置や動作方向などに応じた描画をディスプレイに表示する．以下に各々の詳細を述べる．

位置情報の受信

位置情報解析部と同様，通信には TCP/IP プロトコルによるソケット通信を用いて情報を受信する．本システムでは，描画部を実行する計算機をサーバ側として構成を行った．サーバは常にクライアントからのメッセージを受信できるよう待機している．サーバプログラムには openFrameworks のアドオンである ofxNetwork を利用した．このアドオンは文字列での通信のみのサポートとなっている．受信した文字列は，技法の推定を可能にするために数値に変換する．

描画技法の推定

描画技法の推定は，速度による推定とした．ここでいう速度は，速さと向きを持つベクトル量である．前回取得した情報からどれだけ手が移動したかを計算して推定する．3.2.2 項での知見を基に，ユーザがどの描画技法を行っているかを推定する．以下に推定方法と疑似コード 1 を示す.

まず，前回取得した 3 次元情報 (すなわち 65msec 前の情報) と現在取得した 3 次元情報をもとに，高さの差分 (すなわち z 座標の差分) および移動距離 (すなわち 2 点の x, y 座標間の距離) を計算する．x, y 成分は位置解析を行った際に正規化しているので，この値にキャンバスの縦横サイズをそれぞれ乗じて新たな (x, y) とする．今回の実装では，キャンバスのサイズは 1280 × 780 である．したがって，正規化してある x, y にそれぞれ 1280，780 を乗じてから移動距離を計算する．なお，以降に出てくる (x, y) 座標とは，正規化した値ではなく，キャン

バスの縦横サイズをそれぞれ乗じたときの値である．求めた高さの差分を h_{diff}，移動距離を D とする．

h_{diff} がある閾値より大きく，なおかつ移動距離が短いとき (すなわち D が別の閾値未満である場合)，この動作はドリッピング技法であると推定する．逆に，高さの差分がそれほどなく (すなわち h_{diff} がある一定の範囲内の差分である場合)，なおかつ移動距離がある閾値より大きい場合，この動作はスパッタリング技法であると推定する．なお，高さの差分は (現在の z 座標) – (前回取得時の z 座標) で求めているため，万一手を真上に振り上げてもドリッピングであるという推定を行わないようになっている．

Algorithm 1 描画技法の推定

```
h_diff ⇐ 前回取得時と現在の z 座標の差分
D ⇐ 前回取得時からの移動距離
if h_diff > 閾値 1 then
  if D < 閾値 2 then
    ドリッピング
  end if
else if 閾値 3 < h_diff < 閾値 4 then
  if D > 閾値 5 then
    スパッタリング
  end if
end if
```

描画

推定された描画技法に応じて，ディスプレイに描画を行う．描画の際に必要となるパラメータは，推定する際に算出した情報や，現在の位置情報などを利用する．以下に各技法での描画をどのように実装したかについて述べる．

ドリッピング　ドリッピングの描画は，現在の 3 次元情報を利用し，(x, y) 座標の位置を中心に描画を行う．今回の実装では，スポイトのように一定量の液滴を滴らせて 1 つの円を描画するのではなく，画筆を用いて複数の液滴が滴ることを想定して実装している．すなわち，振り下ろす位置が高い場合には，複数の円が描画されることになる．周囲に飛び散る円の間隔や，それぞれの円の半径は現在の高さを基に決定する．以下に描画アルゴリズムの疑似コード 2 を示し，詳細を述べる．

まず，現在の 2 次元座標を $Pos.x$, $Pos.y$，スライダで指定した色を col，現在の z 成分 を 10 で割った値を rad にそれぞれ代入する．z 成分 の単位はミリメートルであるので，ここでセンチメートルに換算する．3.2 節において，3 段階の高さで実験を行った．この実験結果に基づいて，今回の実装でも高さを 3 段階に分けて描画を変化させるようにした．もし，10cm

Algorithm 2 ドリッピングの描画

$Pos.x \Leftarrow$ 現在の位置の x 座標
$Pos.y \Leftarrow$ 現在の位置の y 座標
$col \Leftarrow$ スライダから取得した RGBA 値 (以降の描画での色はこの色となる)
$rad \Leftarrow$ 現在の位置の z 座標/10
$i \Leftarrow 0$
if $rad <= 10$ **then**
  半径が rad の円を $(Pos.x, Pos.y)$ の位置に描画.
else if $10cm < rad <= 20$ **then**
  半径 rad の円を $(Pos.x, Pos.y)$ の位置に描画.
  while $i <$ 描画する円の個数 **do**
    $drawRad \Leftarrow$ 1 から rad 未満の乱数
    $drawDist \Leftarrow$ 0 から rad*(定数) 内の乱数
    $drawDirection \Leftarrow$ 0 から 2π 内の乱数
    $(Pos.x, Pos.y)$ から $drawDirection$ 方向,
    $drawDist$ 離れた距離に半径 $drawRad$ で円を描画
    $i \Leftarrow i + 1$
  end while
else if $rad < 20$ **then**
  半径 rad の円を $(Pos.x, Pos.y)$ の位置に描画.
  while $i <$ 描画する円の個数 **do**
    $drawRad \Leftarrow$ 1 から rad/(定数) の乱数
    $drawDist \Leftarrow$ 0 から rad*(定数) 内の乱数
    $drawDirection \Leftarrow$ 0 から 2π 内の乱数
    $(Pos.x, Pos.y)$ から $drawDirection$ 方向,
    $drawDist$ 離れた距離に半径 $drawRad$ で円を描画
    $i \Leftarrow i + 1$
  end while
end if

以下の高さで動作を行った場合，位置はそれほど高くないので $(Pos.x, Pos.y)$ の位置に円を 1 つだけ描画する．このときの半径は rad であり，高さを反映するようになっている．

動作を行った高さが 10cm よりも高く 20cm 以下の場合には，複数の円を飛ばす．初めに，10cm 以下のときと同様に $(Pos.x, Pos.y)$ の位置に半径 rad の円を 1 つ描画する．これを中心円と呼称する．次に，飛ばす円それぞれに対し，円の半径，中心円からの距離，方向を決定する．円の半径は，rad よりも小さい値でランダムに定める．これを $drawRad$ とおく．中心円からの距離であるが，今回の実装では，0 から rad を定数倍した範囲内でランダムに決定する．これを $drawDist$ に代入する．そして方向に関しては，中心円の周囲 2π，すなわち 360 度に飛び散るものと仮定して実装する．0 から 2π の間でランダムに値を決定し，これを $drawDirection$ とする．最後に，$(Pos.x, Pos.y)$ から角度 $drawDirection$ 方向，$drawDist$ 離れた距離に半径 $drawRad$ で円を描画する．なお，描画する円の数は任意に定めることが出来る．

動作を行った高さが 20cm より高い場合にも，複数の円を飛ばす．基本的には 10　～20cm の範囲における描画と同じであるが，20cm より高い場合には $drawRad$ と $drawDist$ の決定方法が異なっている．具体的には，$drawRad$ の決定が 10cm～20cm の場合には rad より小さな値でランダムに行われていたのに対し，この場合には rad を定数で割って，その値以内でランダムに決定する．このようにした理由は，10cm～20cm の範囲と同じようにすると，大きな円ばかりが描画されがちになり，不自然であると考えたからである．また，$drawDist$ は 0 から rad を定数倍した範囲内でランダムに決定する．このときの定数は 10cm～20cm よりも大きい値となっている．この高さの範囲においても描画する円の数は任意に定めることが出来る．

図 4.12 は高さの違いによって描画が異なる様子を示している．動作が完了した時点での位置，すなわち現在の位置が低かった場合，図 4.12(a) のように円が周囲に飛び散らないが，現在の位置が高かった場合は，図 4.12(b) のように多くの円が周囲に飛び散る．

スパッタリング　スパッタリングの描画は，前回取得した 2 次元情報と現在の 2 次元情報，および先の推定で求めた移動距離 D を利用する．前回取得した (x, y) 座標を始点，現在の (x, y) 座標を終点とし，始点から終点に向かって描画範囲が扇状に広がるように実装した．以下に描画アルゴリズムの疑似コード 3 を示し，この詳細について述べる．

まず，前回取得した 2 次元座標を $Pos.x$, $Pos.y$，技法推定の際に算出した距離を D，スライダで指定した色を col にそれぞれ代入する．描画方向は，前回の 2 次元情報と現在の 2 次元情報の 2 点を利用し，アークタンジェントを用いて求めることが出来る．これを $direction$ に代入する．次に，描画する各点に対して，それぞれ点の大きさ，距離，方向を定める．点の大きさは 0.5px から 1.5px の範囲でランダムに決定する．しかし，1.0px 以上のサイズの点が多すぎると描画結果が不自然になってしまうと考えたため，1.0px 以上の点は一定数以上現れないように実装した．これを $drawRad$ に代入する．距離は，始点から終点までの距離内でランダムに決定する．すなわち，0 から D までの範囲でランダムに決定を行っている．これを $drawDist$ とする．そして，扇状に描画するために，角度 $drawDirection$ を決定する．これ

(a) 低い位置で振り下ろした場合

(b) 高い位置で振り下ろした場合

図 4.12: 実装したドリッピングの描画

Algorithm 3 スパッタリングの描画

$Pos.x \Leftarrow 65msec$ 前の位置の x 座標
$Pos.y \Leftarrow 65msec$ 前の位置の y 座標
$col \Leftarrow$ スライダから取得した RGBA 値 (以降の描画での色はこの色となる)
$D \Leftarrow$ 前回取得時からの移動距離
$direction \Leftarrow$ 描画方向 (角度)
$i \Leftarrow 0$
while $i <$ 描画する点の個数 **do**
  $drawRad \Leftarrow 0.5px$ から $1.5px$ 内の乱数
  $drawDist \Leftarrow 0$ から D 内の乱数
  $drawDirection \Leftarrow direction$ からある一定角度範囲内の乱数
  $(Pos.x, Pos.y)$ から $drawDirection$ 方向,
  $drawDist$ 離れた位置に半径 $drawRad$ で点を描画
  $i \Leftarrow i + 1$
end while

は，先ほど求めた $direction$ を中心とし，ある一定範囲の角度内をランダムに指定して代入している．最後に，$(Pos.x, Pos.y)$ から角度 $drawDirection$ 方向，$drawDist$ 離れた距離に半径 $drawRad$ で点を描画する．なお，描画する点の数は任意に定めることが出来る．

図 4.13 は移動距離の違いによって描画が異なる様子を示している．移動距離が短い場合，図 4.13(a) のように描画範囲は狭くなるが，移動距離が長い場合には，図 4.13(b) のように描画範囲が広くなる．

(a) 移動距離が短い場合

(b) 移動距離が長い場合

図 4.13: 実装したスパッタリングの描画

4.3.3 その他の機能

その他の機能として，色の調整機能と制作したデジタル絵画の保存機能，およびキャンバスの全クリア機能を実装した．デフォルトの色は黒に設定している．色の調整は RGBA 値のスライダで行い，ユーザはスタイラスを用いてスライダの値を調整することにより色を変更することが可能である (図 4.14)．また，このスライダはキーボードの Space キーで表示/非表示の切り替えを行うことが出来る．デジタル絵画の保存はキーボードの Enter キーを押すことで可能となる．このとき，ファイルの上書きを防ぐために，ファイル名は現在の年月日および時刻とした．また，保存形式は PNG である．キャンバスを白紙に戻したい場合は，キーボードの “c” をタイプすることで全てを消去することが出来る．

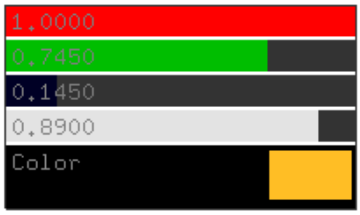


図 4.14: 色を調整するためのスライダ

第5章　評価

本章では，実際に本システムを用いて自由制作の実験を行ったことについて述べる．実験の目的は本システムの使いやすさや問題点，および要求を発見することである．

5.1　試用から得られた知見・課題

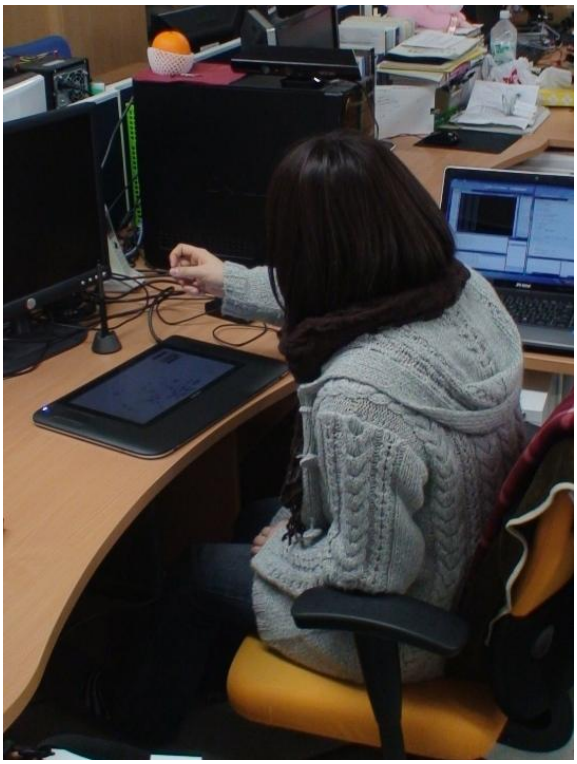
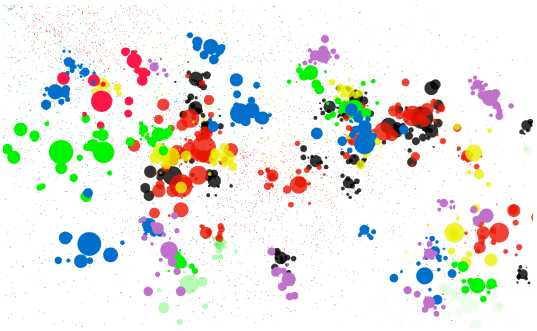
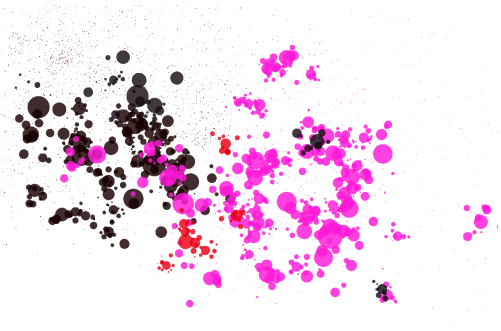


図 5.1: 実装したシステムを試用している被験者

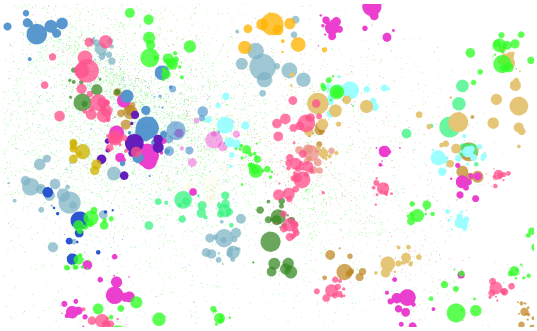
被験者に，4 章で実装したシステムを用いて自由に制作を行ってもらった．被験者は情報系の大学生 3 名である．うち 1 名に筆者を含む．被験者には，操作等について 5 分程度の説明を受けた後，自由な制作を 10 分程度行ってもらった．図 5.1 は本システムを試用している被験者の様子である．図 5.2 にそれぞれの被験者が制作したデジタル絵画を示す．



(a) 被験者 A が制作したデジタル絵画



(b) 被験者 B が制作したデジタル絵画



(c) 被験者 C が制作したデジタル絵画

図 5.2: 本システムを試用した被験者が制作したデジタル絵画

5.1.1 試用した被験者からのコメント

試用した被験者から，以下のようなコメントが得られた．

- 操作についてのコメント
  - 手だけで操作出来るのは楽である．
  - 色の変更に関して，スタイラスを用いてスライダで行うのはやりやすかった．
  - 実際の描画の動作に似ていると思う．
  - 手で制作するのは楽しいが，手の操作性の良さ・柔軟さを活かしてもっと細かい操作が出来ると良い．
  - 空中で操作し続けるのは少し疲れる．
- 描画に関してのコメント
  - ランダムに描画されるため，どういう描画になるのか分からないのが楽しい．
  - 高さに応じて描画に変化があるのは良い．
  - スパッタリングの描画が細かすぎて描画されたのかが分かりにくい．このため，実際に則しているのかも疑問がある．
  - アルファ値もスライダで調整出来るが，アルファ値が低いときの描画結果は実際の描画に近いとは言えないかもしれない．
  - 円のみの描画では物足りない．さらに実際に則した偶然性があると良い．
  - 精度がもう少し良い方が良い．
  - Undo 操作が出来ると良い．
- その他のコメント
  - 最初は制作するもののアイデアがあまり固まらないまま制作を始めたが，ある程度のランダム性のある制作を行っていくうちに，アイデアが固まっていったような感覚や，新しいアイデアが生まれたような感覚があって楽しい．

操作については，手のみで操作できることの簡単さや楽しさを感じてもらえた．しかしながら，手の操作の柔軟性をもう少し活かせると良いという意見も得られた．現在は手のひらの中心 1 点の位置情報のみを解析しているが，指先 5 点の位置情報を解析することが出来れば，さらに細かいインタラクションが行えると考える．

描画に関しては，ある程度の偶然性があることが楽しいというコメントが得られた．しかし一方で，スパッタリングの描画の細かさについて，全被験者がネガティブなコメントをしていた．細かすぎる描画であるため，しばしば画面を覗き込んで描画できたかどうかを確認している場面もしばしば見受けられた．このことに関しては，描画したことが一目で分かる程度まで，一つ一つの点のサイズを大きくすることで改善されるのではないかと考えられる．

5.1.2 観察から見られた問題点

被験者が制作を行っている間に，いくつかの問題点が見られた．以下にその問題点について述べる．

データ受信の遅延

今回実装したシステムにおいて，データ受信の遅延が見られた．これは，位置情報解析部で手の位置を捉えられなくなり，再度フォーカスジェスチャを行って位置を取得していたときに生じていた問題である．遅延が生じていると描画も遅れてしまうため，動作と描画にもタイムラグが生まれてしまう．これが原因で，何度も描画動作を行ってしまい，余計に描画されるという問題も発生していた．

予期しない描画

位置情報解析部が手の位置を把握している限り，キャンバスに対する手の位置情報は常にサーバに送信されている．このため，意図せずキャンバス上で手を動かした際に描画の推定が行われてしまい，ユーザの意図を反映しない描画が行われることがあった．

5.2 今後の課題

現在のシステムでは，生成されたキャンバス面が固定されるため，液晶タブレットを動かすと基準点がずれてしまうという問題がある．この問題に対しては，現在調整用用紙に取り付けた再帰性反射材を液晶タブレットに取り付け，このマーカを OpenCV[1] のようなライブラリを用いてトラッキングし，座標変換を行うことで改善されると考えられる．

このシステムの応用として，具体的な物を描くこととの組み合わせが考えられる．今回実装したシステムでは，線の描画の機能を実装していない．線の描画機能を追加すれば，具体的な絵画に変化をつけることが可能になると考えられる．

[1] OpenCV (URL : `http://opencv.jp/`)

第6章　結論

本研究では，デジタル絵画制作において，モダンテクニックの動きを取り入れた描画手法を提案し，その手法を実現するインタフェースの開発を行った．描画手法を提案するにあたり，実際にモダンテクニックを用いて制作を行った．そして，その制作過程から描画の違いや動きについて考察し，実装するインタフェースの設計を行った．開発したインタフェースは，ユーザが特定の描画技法の手の動きをすることにより，デジタル絵画における抽象的な表現が可能である．また，その動きを行った位置に応じて描画に違いが出るよう工夫した．

今後は，試用から得られた課題を基にシステムの拡張を行いたい．また，精度実験など本システムの定量的な評価を行い，そこから得られたフィードバックを基に本システムの改善を図りたい．

謝辞

本研究を進めるにあたり，指導教員である高橋伸准教授，田中二郎教授をはじめ，三末和男准教授，志築文太郎講師には適切なご指導をいただきました．ここに深く御礼申し上げます．また，インタラクティブプログラミング研究室の皆様には，ゼミなどを通じて数々のご意見をいただきました．特に，ユビキタスチームの皆様にはチームゼミ以外にも研究生活において多くのご意見やご指摘をいただきました．この場を借りて厚く御礼申し上げます．そして，精神的・経済的にに支えてくれた両親がいなければ，大学生活をここまで実りあるものには出来なかったと思います．ここに感謝の意を表します．最後に，大学生活を共に過ごした友人やサークルの先輩・同期・後輩，学生生活の中でお世話になった全ての方々に心より感謝いたします．本当にありがとうございました．

参考文献

[1] Peter Vandoren, Luc Claesen, Tom Van Laerhoven, Johannes Taelman, Chris Raymaekers, Eddy Flerackers, Frank Van Reeth. FluidPaint: an interactive digital painting system using real wet brushes. *ITS '09*: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, pp.53-56, 2009.

[2] 岩井 大輔，金谷 一朗，日浦 慎作，井口 征士，佐藤 宏介． Thermo Painter：熱画像を用いたタブレット型入力装置をそのインタラクティブ描画システム． 情報処理学会論文誌 46(7)，pp.1582-1593，2007.

[3] Anthony J. Hornof, Anna Cavender. EyeDraw: enabling children with severe motor impairments to draw with their eyes. *CHI '05*: Proceedings of the SIGCHI conference on Human factors in computing systems, pp.161-170, 2005.

[4] Susumu Harada, Jacob O. Wobbrock, James A. Landay. Voicedraw: a hands-free voice-driven drawing application for people with motor impairments. *Assets '07*: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility, pp.27-34, 2007.

[5] Jan van der Kamp, Veronica Sundstedty. Gaze and voice controlled drawing. *NGCA '11*: Proceedings of the 1st Conference on Novel Gaze-Controlled Applications, Article No.9, 2011.

[6] Xinlei Chen, Hideki Koike, Yasuto Nakanishi, Kenji Oka, Yoichi Sato. Two-handed drawing on augmented desk system. *AVI '02*: Proceedings of the Working Conference on Advanced Visual Interfaces, pp.219-222, 2002.

[7] Horace H. S. Ip, Young Hay, Alex C. C. Tang. Body-Brush: a body-driven interface for visual aesthetics. *MULTIMEDIA '02*: Proceedings of the tenth ACM international conference on Multimedia, pp.664-665, 2002.

[8] 櫻井 稔，江渡 浩一郎． Sequential Graphics: 描画時の臨場感を再現するペイントソフト. *WISS 2008*: 16th Workshop on Interactive Systems and Software, pp.29-34, 2008.

[9] Sumi-nagashi: creation of new style media art with haptic digital colors. Shunsuke Yoshida, Jun Kurumisawa, Haruo Noma, Nobuji Tetsutani, Kenichi Hosaka. *MULTIMEDIA '04*: Proceedings of the 12th annual ACM international conference on Multimedia, pp.636-643, 2004.

[10] Proactive Desk: 力覚提示が可能なデスクトップ操作環境．吉田俊介，柿田充弘，野間春生，鉄谷信二．インタラクション 2003, pp.211-212, 2003.

[11] Bill Baxter, Vincent Scheib, Ming C. Lin, Dinesh Manocha. DAB: interactive haptic painting with 3D virtual brushes. *SIGGRAPH '01*: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp.461-468, 2001.

[12] Arthur D. Gregory, Stephen A. Ehmann, Ming C. Lin. inTouch: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface. *IEEE Virtual Reality*, pp.45-52, 2000.

[13] Sangwon Lee, Sven C. Olsen, Bruce Gooch. Interactive 3D Fluid Jet Painting. *NPAR '06*: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering, pp.97-104, 2006.

[14] K. Ryokai, S. Marti, H. Ishii. I/O brush: drawing with everyday objects as ink. *CHI '04*: Proceedings of the SIGCHI conference on Human factors in computing systems, pp.303-310, 2004.

[15] 草地 映介，渡邊 淳司．表現意図と偶然性を併せ持つ “Minimal Drawing”の提案．日本バーチャルリアリティ学会論文誌，12(3)，pp.389-392，2007.

[16] 美術手帖．現代アート事典．美術出版，2009.

[17] 美術資料．秀学社.

[18] 武蔵野美術大学．デカルコマニー.
`http://zokeifile.musabi.ac.jp/contents/decalco/decalco.pdf`

[19] 武蔵野美術大学．フロッタージュ.
`http://zokeifile.musabi.ac.jp/contents/frotta/frotta.pdf`

[20] 世界美術大事典 GRANDE ENCICROPEDIA dell'ARTE, 5, pp.269. 小学館, 1989.