

スケッチインタフェースの研究動向

五十嵐健夫

手書きスケッチを計算機アプリケーションへの入力として用いるスケッチインタフェースが注目を集めている。スケッチインタフェースは、従来の GUI と比較して特に種々のデザイン活動の初期段階における創造的な作業の支援に有効であると考えられている。本稿では、これまでに発表されてきた代表的な手法やアプリケーションシステムを紹介しながら、スケッチインタフェースの可能性と将来像について論じる。具体的には、基礎技術である認識アルゴリズムやインタラクション手法について紹介したあと、スケッチ入力を利用した 2 次元のデザインシステムや 3 次元グラフィックスのオーサリングシステムについて紹介する。効果的なスケッチインタフェースを実装する上での注意点などについても議論する。

Sketching interfaces are getting popular in user interface and graphics research community. They are expected to be effective for creative exploration in the early stages of intellectual activities. This paper reports on the recent trends in such sketching interfaces. We first describe basic recognition techniques and interaction techniques for sketching systems. We then introduce some representative application systems both in 2D and 3D domains. We also discuss some design issues to build effective sketching systems.

1 はじめに

現在主流のインタフェースは、マウスを用いて画面上のボタンやメニューを操作するという通常グラフィカルユーザインタフェース (GUI) と呼ばれているものである。このようなインタフェースは、それ以前のすべてのコマンドをキーボードから入力するコマンドラインインタフェースと比較して、特に初心者にとって非常に使いやすいものとなっている。しかし、ボタンやメニューによる操作は、依然として「ユーザが計算機に対して細かく指示を出して操作する」というモデルに基づいており、ユーザが本来作業の対象としているタスクに思考を集中することを防いでいると考えられる。この問題は、特に知的生産活動の初期段階である創造的な作業の支援において重大な問題となってくる。このような問題を解決し、自由な発想を防

げることなく創造的な活動を支援するインタラクション手法として、手書きスケッチを入力として用いるスケッチインタフェースが注目を集めている。ここでいうスケッチとは、紙の上にペンで絵や図を走り書きすることを指している。このようなスケッチは、通常の GUI を利用した編集作業と比較して非常に手早く自分の考えを表現したり変更したりすることができるため、特に種々のデザイン活動の初期段階において、頭の中の曖昧な概念を整理して具体化していく作業に適している。従来紙の上で行われていたこのようなスケッチ作業を計算機によって支援する、あるいは計算機での編集作業にスケッチの考え方を取り入れようとするのがスケッチインタフェースである。

手書きの図面等を計算機に認識させようという取り組みは古くから存在しているが、主に人工知能や画像認識の分野において、入力された複雑な図形を一度に解析する認識アルゴリズムの開発に重点がおかれていた。近年研究が盛んに行われているものは、手書き入力を総合的に活用したアプリケーションシステムを構築しようとしているものであり、特にシス

Recent Trends in Sketching Interfaces.

Takeo IGARASHI, 東京大学, The Univ. of Tokyo.

コンピュータソフトウェア, Vol.16, No.5 (1999), pp.78-83.

[小論文] xxxx 年 yy 月 zz 日受付.

テムの対話的側面を重視している点が異なっている。すなわち、単にスケッチを認識して終わりにするのではなく、結果をどう提示しそれに対してさらにユーザがどう行動すべきかまで考慮にいれている点が最近のスケッチインタフェースの特徴であるといえる。

本稿では、スケッチインタフェースに関わる研究事例について紹介し、スケッチインタフェースの可能性と将来像について論じてみたい。まず、スケッチインタフェースの基礎技術である、手書き図形の認識アルゴリズム、およびペンを用いた様々なインタラクション手法について紹介する。次に、スケッチインタフェースを利用した様々なアプリケーションシステムについて、2次元ベースのものとは3次元ベースのものに分けて紹介する。さらに、これらのアプリケーションシステムの紹介を踏まえた上で、スケッチインタフェースを実装する上で重要な要素について議論する。最後に、まとめと将来の研究課題について述べる。

2 認識技術

スケッチインタフェースにおいて、ユーザからの入力大きく2種類に分けることができる。一つは、紙の上においてペンで書きつける作業に相当するものであり、ペンの動きがそのままストロークとして画面に記録される。もう一つは、紙には存在しない制御コマンドであり、スケッチシステムでは多くの場合、ジェスチャーとして実装されている。ジェスチャーとは、特定の簡単なペンの動きによってコマンドを起動するものであり、例えば、点を打ったら選択、丸で囲んだら範囲選択、ぐじゃぐじゃと書いたら削除、といったジェスチャーが考えられる。

ペンで書かれたものの認識は、オンライン文字・図形認識に相当する処理である。ここでいうオンラインとは、紙に書いたものをスキャンして処理するのではなく、元からストローク単位で表現されたベクトル情報を扱うという意味である。要素技術としては、文字と図形を区別する処理、文字認識、個々の図形の認識、複数の図形からなる図の認識、などがあげられる。ここでは、最近の研究のうち、複数の図形からなる図の認識を対象としたものを紹介する。

Gross らは、図形の認識においては、個々の図形の形状だけでなく周囲に描かれている図形との位置関係が重要であるとし、そのような考え方に基づいた認識システムを提案している [8]。例えば、同じ2つの円が並んで描かれていたとしても、それがより大きな円の中にあれば目と考えるべきであるし、箱の下についていれば車輪と考えるべきである。このシステムのアルゴリズムは、基本的にはボトムアップとトップダウンの組み合わせである。まずジェネリックな円という形状が認識された後、上位レベルで周囲のコンテキストからそこに顔あるいは車が存在することが導き出され、さらにその情報を下におろすことによって最初ただの円として認識されたものが目あるいは車輪とラベル付けされる (図1)。

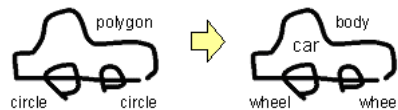


図1 状況に応じたプリミティブの認識の例 [8].

Alvarado らは、同様にボトムアップな認識とトップダウンな探索を組み合わせることで、認識誤りを低減する手法を提案している [2]。通常利用される方法は、単純にボトムアップに要素をまとめていくか、トップダウンに当てはまる要素を探索していくかの2通りである。しかし、ボトムアップだけだと途中で認識誤りがあるとそれ以上進めなくなり、逆にトップダウンだけだと対象とできる図の自由度が低くなるという欠点がある。この研究では、基本的にはボトムアップに探索しながら構造に対する仮説を徐々に構成していき、かつその仮説で仮定された構造を満たすような要素が存在しているか確認するといった処理を行っている。具体的な例として、家系図を認識するシステムと電子回路を認識するシステム (SketchRead) を紹介している (図2)。

ジェスチャーに関しては、形状の認識アルゴリズムを追求するアプローチと、インタフェースを工夫していこうとするアプローチの大きく2つに分けられる。後者については次節で詳しく述べる。ジェスチャー形

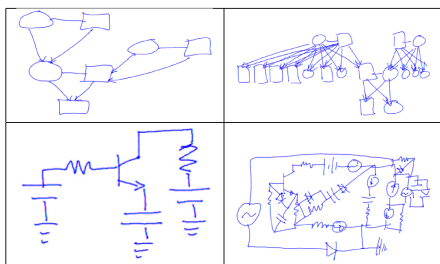


図 2 SketchRead における図の認識の例 [2].

状の認識アルゴリズムとしては、Rubine のアルゴリズム [23] が非常に多くのシステムで利用されている。これは、例として与えておいたジェスチャーとユーザの描いたジェスチャーを、共に特徴パラメータ空間にマッピングしその空間上で入力ジェスチャーに最も近いサンプルジェスチャーを認識結果として返す、というものである。特徴量としては、始点・終点での線の角度や始点と終点の距離、ストロークの全長などが使われている (図 3)。

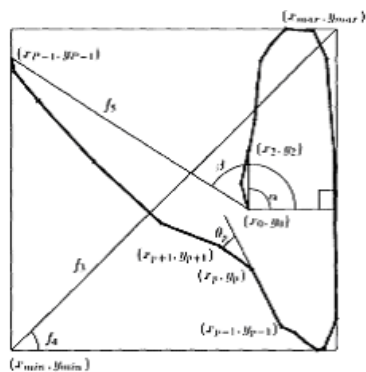


図 3 Rubine のアルゴリズムにおける特徴量 [23].

3 インタラクション手法

スケッチシステムにおいて、ほとんどの操作はただ単純に「書く」というものであり、実物のペンを使った作業と変わらない。しかし、削除や undo といった計算機の機能を利用するためには、なんらかのコマンド処理が必要となる。そのためのインタフェースとして、通常のボタンやメニューに代わる様々な手法が

提案されている。ここでは、それらについて簡単に紹介する。

Pie Menu は、通常ポップアップメニューにおいて縦に並んで表示される項目を放射状に並べたものである [4]。放射状に並べることによって、それぞれの項目を位置でなく方向で区別することが可能となる (図 4 左)。これは特に熟練したユーザにとって有利となる。すなわち、初心者の方はメニューの内容を目で確認したあとに項目を選択するが、熟練してくるとメニューを目で見なくてもペン先を動かす方向を変えるだけで項目を選択することができるようになる。さらに熟練したユーザ向けには、メニューそのものを表示せず、ペンの動きだけで該当する項目を選択させることが可能となる。このようなものは Marking menu と呼ばれている [16] (図 4 右)。Pie menu, Marking menu とともに階層化することが可能である。例えば、Pie menu で下方向に「ファイル」という項目があり、そちらを選ぶと「保存」「新規作成」などのメニューが現れ、右方向を選ぶと「保存」が選ばれる、といった具合になる。これを Marking menu で操作する場合には、単に「L」という形状を描けば同じコマンドが実行される。Pie menu, Marking menu は、ペンに適したインタフェースであるという他に、初心者から熟練者へのスムーズな移行をサポートするという点でも重要である。



図 4 Pie menu (左) [4] と Marking menu (右) [16].

ペン入力に関する研究で最近注目を集めているものに、crossing による操作がある。これは、通常のマウスにおけるクリック操作に対応するものとして、ペンで 1 点をたたきタッピングの代わりにペンで目

標物を「横切る」動作を利用しようとするものである(図 5). マウスであれば目標物の上にカーソルを持っていてからクリックする間にポインタが移動することはあまりないが, ペンの場合にはペン先をおろしていく過程で(視差などの影響もあり)どうしても位置がずれてしまうために小さな目標を正確にヒットすることが難しい. それに対して, すでにペン先を画面に押し付けた上で目標を横切るという動作はより安定に行うことができる. Accot らは実験によって crossing の有効性を示しており [1], Apitz らは crossing のみを用いたアプリケーションの試作を行っている [3].

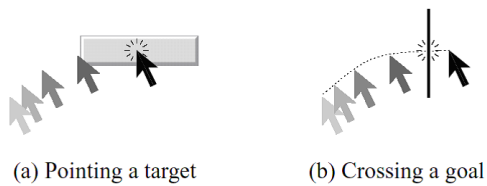


図 5 Pointing (左) と Crossing (右) [1].

その他, 大きな丸で対象を囲った後に最後に豚のしっぽのような小さな丸を描くとメニューが現れる Scriboli というインタフェース [11] や, ペンを画面に押し付けずに空中で特定のジェスチャーをするとコマンドが起動される hover widgets[10], marking menu のように一つの長いストロークでジェスチャーを指定するのではなく短いストロークを何本か組み合わせる multi-stroke menu[29], 圧力を感知できるペンを利用した pressure widgets[22] などが提案されている. なお, ジェスチャーを利用する場合には, ジェスチャー入力と通常の描画操作とを区別する手段について考慮が必要である. 通常使われる方法は, ペンの側面についているボタンを押しながら描いた場合にジェスチャーとする, といったものであるが, ボタンがない場合などは, 描き始める前にペン先をしばらく停留させる, あるいはジェスチャーを描いた後にタップする, などの方法が利用されている.

4 2D スケッチ

この節では, スケッチインタフェースを利用した 2 次元のアプリケーションシステムの例をいくつか紹介する. 対象としているのは, GUI や WEB の画面レイアウトのようなデザインの初期段階の支援や, 物理や数学の問題を説明するための補助というように, 簡単に作成できて結果をすぐに見ることができる点が重要なタスクであり, この目標を達成するための手法として, 手書きスケッチが有効に活用されている.

スケッチインタフェースによるインタラクティブなデザイン支援という考え方ははじめに提唱したのが, Landay らによる SILK システムである [17]. これは GUI の画面デザインをスケッチによって行うシステムである. 通常このようなラフな画面デザインは, インタフェースビルダーを利用した計算機上での詳細な編集作業に先立って, 紙とペンを利用して行われている. SILK は手書きスケッチを利用することでこのデザイン作業を計算機で行えるようにしたものであり, 単に見かけのデザインだけでなく簡単な動作も試してみることを可能している. 例えば, 画面上にスクロールバーを描くとそれで画面をスクロールしたり, ボタンを描くとそれを押して別の画面に移ったり, といったことができる(図 6). 認識は行わが整形は行わずに手書きの絵をそのまま表示するので, ユーザはスケッチ画のまま操作を試してみることができる. あえて整形せずにスケッチのまま表示することで, まだ最終決定ではなく案の段階であることが表現され, 様々な可能性についてより柔軟に検討することが可能となっている.

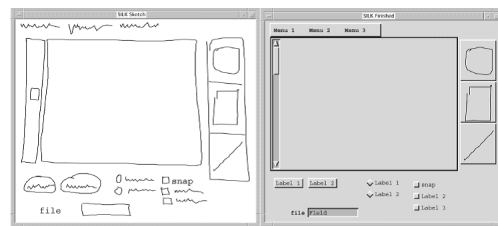


図 6 SILK の画面例 (右は認識結果) [17].

DENIM は、同様にスケッチ画を利用して Web サイトのデザインを行うシステムである [19]。最初にサイト構造を編集するモードで画面上に個々のページを表すラベルを配置し、さらにその間のリンク構造を表す線で結ぶ (図 7 右)。典型的には、トップページの下にいくつか関連ページがぶら下がっているような構造になる。次に、ページをデザインするモードに入り、個々のページの画面イメージを手書きスケッチで描く (図 7 左)。更に、ページ内の任意の場所から他のページへ向かう線を描くことで、ハイパーリンクを張ることができる。ある程度デザインができたあとで「実行」モードに入ると、擬似的な Web ブラウザの上で、これらのスケッチ画で表現されたページ群をハイパーリンクをたどりながらブラウズすることができ、気に入らなければ即座に手書きスケッチで内容を訂正することができる。本システムは、WYSIWYG エディタなどで実際のページ作成を行う前の全体的なサイト構造のデザインの段階で使うことを想定しており、SILK の場合と同様にあえて手書きのままで表示することで様々なデザインを手軽に試すことを後押ししている。

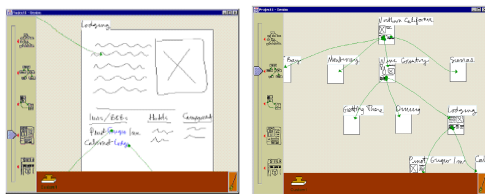


図 7 DENIM の画面例 [19]。ページビュー (左) とサイトビュー (右)。

Music Notepad は、五線譜に手書きで音符を描いていくことで楽譜を作成するシステムである [9]。ただし、通常の音符をそのまま描くのではなく、ペンで簡単に入力できてかつ認識しやすいようなジェスチャーが設定されていて、それが入力として利用される (図 8)。記入した音符の列の上をペンでなぞることで、その部分を再生してみたりすることもできる。本システムは、通常よくある楽譜入力インタフェースと比較して、より楽譜に書き込むのに近い感覚で入力でき、

また手軽に入力内容を確認することができるので試行錯誤しながら楽曲を作成する作業に適している。



図 8 MusicNotepad におけるジェスチャー例 [9]。

スケッチインタフェースは、上記のようなデザイン活動だけでなく、講義中に先生と生徒の間で対話的に行われる、黒板を利用した説明などにも利用されている。五十嵐らは、算数や理科の授業ででてくるような図の板書を補助するものとして、手書きで描かれた線を受け取って平行や直角・対称といった幾何学的な関係を満たすように清書する Peagsus というシステムを提案している [12] (図 9)。Davis らは、力学を説明するためのツールとして、手書きスケッチで描いた車輪や傾斜面・バネや重りといった要素に対して物理シミュレーションを適用できるシステムを提案している [7] (図 10)。手書きスケッチを利用することで、様々な初期条件を与えるとどのような結果の違いにつながるのかを簡単に見せることが可能となる。LaViola らは、手書きで描いた図が同じく手書きで書かれた数式に従ってアニメーションする、といったシステムを提案している [18] (図 11)。例えば、 $x_0 = t$, $x_1 = t^2$ といった数式を書き、さらに丸を二つ描いてそれぞれを x_0 , x_1 に関連付けると、二つの丸が式に従って移動する (はじめは x_0 の方が先行するが、後から x_1 が早くなって追い抜く) といったアニメーションが示される。このようにすることで、一見ただけではわかりにくい式の意味の理解を促進することができる。

手書き入力を利用した様々なアプリケーションをサポートする基盤システムを目指したものとして Flatland システムがある [20]。このシステムは手軽にメモを取るための電子ホワイトボードシステムとしてデザインされており、画面上に複数のアプリケーションが共存しているときにそれらの間で重なりが起

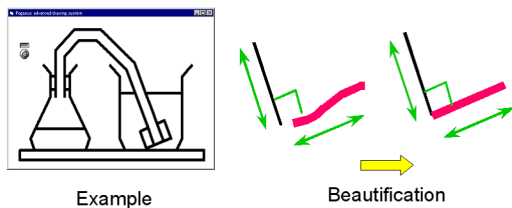


図 9 Pegasus の画面例と整形操作 [12].

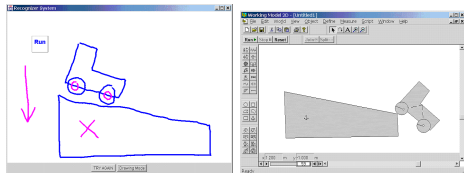
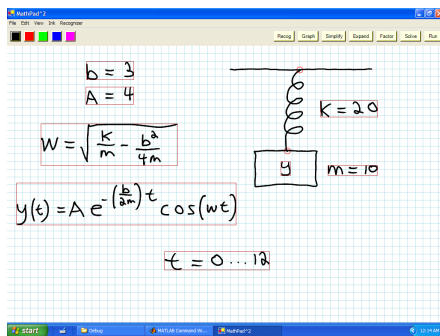


図 10 スケッチへの力学シミュレーションの適用 [7].

図 11 MathPad² の画面例 [18].

こらないように調整をする空間管理機構、画面上の特定の領域においてアプリケーションを起動して個別の作業を支援する支援機構、および履歴をすべて自動的にデータベースに保存し過去の状態をいつでも復元できるようにする履歴管理機構などを提供している。具体的なアプリケーションとして、手書きで一本の線を引くと自動的に2本に増やして道路にしてくれる(交差点などがあれば自動的にその処理も行う)地図描画機能、手書きで数式を書くと自動的に計算してくれる計算機能、手書きの図を描くと自動的に整形してくれる整形機能などが実装されている(図12)。実装上の特徴として、本システムではすべての入力と表示がストロークとして表現されており、アプリケーションはその構造を解釈して変換を行う一種のフィル

ターと動作するように設計されている。

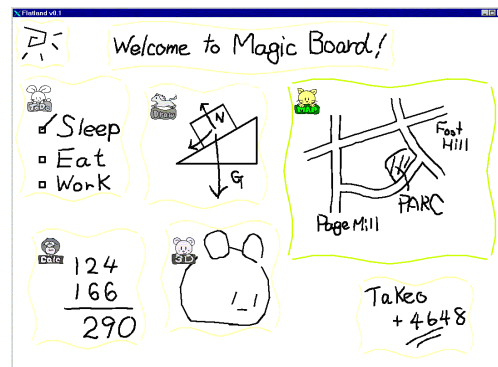


図 12 Flatland の画面例 [20]. 様々なアプリケーションが共存している。

5 3D スケッチ

スケッチインタフェースが注目を集めているアプリケーション分野の一つとして、3次元グラフィックスがある。通常の3次元グラフィックスの作成環境は、訓練を受けた専門家が映画や広告に使う高品質の画像を注意深く作成するようにデザインされており、多くの制御点を設定したり複雑な編集コマンドを組み合わせ使用したりしなくてはならないといった問題点がある。このような複雑なインタフェースはデザインが確定した後にそれを計算機上の表現へと変換する目的には適しているが、アイデアを膨らませてデザインを試行錯誤しながら確定させていくプロセスには適さない。そのため、通常このようなデザイン作業は、紙とペンを利用して計算機の外で行われている。スケッチインタフェースは、このようなデザインプロセスに計算機の計算能力を活用して行うものであり、複雑な操作を通じて詳細な表現を時間をかけて作成するのではなく、簡単な操作でラフな表現を即座に作成することを目的としている。また、操作を簡単にすることによって、初心者や一般のユーザが、自分の考えを整理したり他人に伝えたりするための道具として使うことも可能となる。以下、いくつかのシステムを紹介する。

インタラクティブな3次元スケッチングの可能性

を最初に示したのが Zeleznik らによる SKETCH というシステムである [28]. それ以前にも, 手書きの線画から 3 次元形状を自動生成するという取り組みは画像認識や人工知能の分野で長く行われてきていたが, あくまでも計算機は線画を入力として受け取って 3 次元形状を出力として返すというバッチ处理的な考え方に基づくものであった. SKETCH は対話性を重視しており, 計算機上で視点を変更したりしながら徐々に形状を作り上げていくことを特徴としている. 基本的には, 用意された直方体や円柱・円錐といった 3 次元のプリミティブを 3 次元空間中に配置していくというシステムになっている (図 13). プリミティブ毎に 2 次元のジェスチャーが設定されており, ユーザが画面上にジェスチャーを描くと, その場所に対応する 3 次元プリミティブが配置される. 重要な点は, 2 次元入力に含まれていない奥行き情報をどのように決定するかであるが, 本システムの場合は「あらゆるオブジェクトはすでにあるオブジェクトの上に乗っている (宙に浮いていない)」という暗黙の前提を利用することで, この問題を解決している. どうしても宙に浮かせたい場合には, 物体から少し離れた位置に影を描くことで宙に浮いている状態を表現することができる. 本システムは建物のような空間の表現に適しており, 実際の細かい設計に入る前のデザインの検討などに利用されることを想定している.

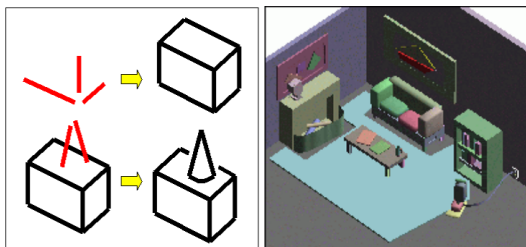


図 13 SKETCH の操作の例とモデリング例 [28].

SKETCH が建物のように平面で囲まれた形状を対象としているのに対し, Teddy と呼ばれるシステムは生物のように自由曲面で囲まれた形状を表現することを目的としている [13]. ユーザが画面上に描きたい 3 次元形状の輪郭をなぞるような線を描くと, 計

算機がその輪郭に対応するような 3 次元形状を自動的に生成して提示する (図 14). 計算機内部においては, 輪郭に囲まれた領域を前後に「膨らます」ような処理を行っている. できあがった 3 次元形状に対してはさらに切断処理や突起生成処理などを適用することができる. これらの処理はすべて手書きスケッチを画面上に描くことによって行われる. Teddy のような手法を用いることにより, 3 次元グラフィクスに関する経験のない子供でも簡単に表現力豊かな 3 次元モデルを作成することが可能となっている. また, その場で簡単に 3 次元形状を作成できるという機能は 3 次元的な概念を伝えるための道具としても有用であり, 医者が患者に対して疾患の状況を説明したり, 学校の先生が生徒に対して立体的な概念の説明をしたりする場合に利用することができる.

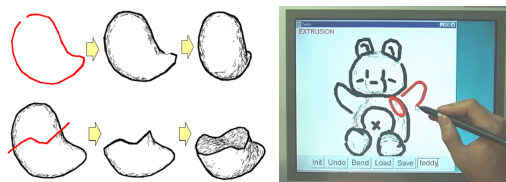


図 14 Teddy の操作の例 (左) とモデリング例 (右) [13].

上記の 2 つのシステムは, 内部表現自体は標準的な 3 次元モデルとなっており, 出力結果は完全な 3 次元形状データとして利用することも可能である. しかし, スケッチを純粋にデザインのための中間表現と考えた場合には, そのような完全な 3 次元表現を作成しなくても様々な効果的な処理を実現することができる. Projective Strokes と呼ばれるシステム [26] は, 自分の周囲を見回すように視点を変えながらスケッチすることで, 自分を中心とした 3 次元空間を擬似的に表現するシステムである (図 15). 内部的には, 画面上に描かれた手書きストロークを視点を中心とした球面上に投影することによって, このような効果を実現している. Harold と呼ばれるシステムは, 3 次元世界の中でユーザが描いたひとかたまりのストロークを一枚のビルボードに投影して表現する [5]. ビルボードとは, 常に視線方向を向くように制

御されたオブジェクトのことであり、この上に絵を描くことで木や建物といったものを実際の3次元構造を作成することなく表現することが可能となる。



図 15 周囲を見回すように視点を変えながら描いた絵を、視点を囲む円柱面へ投影した例 [26].

その他、対象を限定することによってより複雑な表現を実現しているシステムがいくつかある。まず、植物を対象としているものとして、樹木をモデリングするものと花をモデリングするものが発表されている。樹木をモデリングするシステムでは、ユーザはまず幹や枝に表現した2次元スケッチを描き、これをシステムが自動的に3次元化する[21]。この際、枝同士がなるべく離れた場所に配置されるように、奥行き方向の形状が決定される(図16)。また、すべての枝を手で描くのは労力がかかるので、このシステムでは、既に描かれた枝を例として類似した枝を大量に自動生成する機構なども提供している。花のモデリングシステムでは、花びらやおしべ・めしべなどそれぞれに応じたパーツごとに専用のスケッチインタフェースを提供することで効率のよいモデリングを実現している[15]。例えば、花びらの作成では最初に3本の線を引くことで基本形状を作成し、さらに追加の変形ストロークを描くことで3次元的な形状が作成される(図17)。またこのシステムでは、花のもつ複雑な構造を扱いやすくするために、植物学の分野で伝統的に用いられてきた花式図や花序と呼ばれる表現を利用している。

特定のアプリケーションを対象としたものとして、他に衣服のデザインを対象としたものがある。五十嵐らのシステムは、2次元のパターンとしてデザインされた衣服を3次元キャラクターに着せ付けるためのインタフェースとして、2次元のパターンと3次元キャラクターの上に対応するストロークを描くマーキングインタフェースを提案している[14]。システムは、与えられたストロークがなるべく一致するように衣服を3

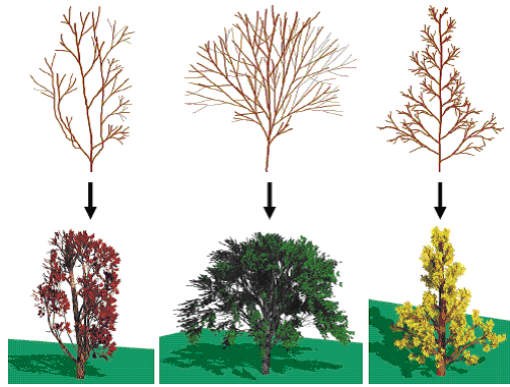


図 16 樹木のモデリング例 [21].

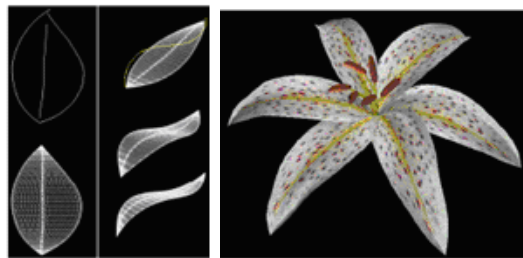


図 17 花のモデリング例 [15].

次元キャラクターの体の上に配置する(図18)。このようにすることで、3次元空間中で移動したり回転したりするという複雑で面倒な操作を行うことなく、衣服を簡単に着せ付けることが可能となっている。一方、Truquinらのシステムは、2次元のパターンを描くのではなく、人体を表現した2次元の絵の上に直接服の絵を描くインタフェースを提供している[27]。与えられた服のスケッチ画は、システムによって自動的に3次元的な衣服の形状に変換され、3次元キャラクターの上に着せ付けられる(図19)。両者とも、衣服の形状を厳密に表現するためのものではないが、服の着せ付け方を検討したり、服のデザインの概略を検討したりする目的に適したシステムとなっている。

スケッチを利用してアニメーションを作成するシステムとして、Motion Doodlesと呼ばれるものがある[25]。これは、画面上に動きを表現する線を描くことで、簡単にアニメーションを作成するというものである。対象としている動作は、歩行や走行・ジャンプといった、地面の上を一方方向に向かって進んでいく動作

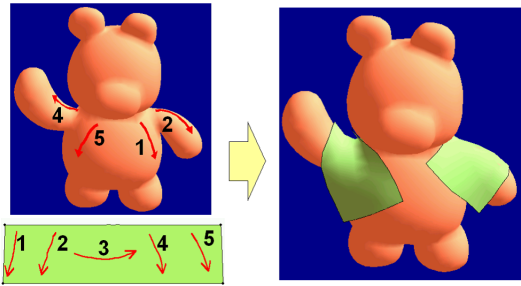


図 18 対応する線を描くことによって衣服の着せ付けるインタフェース [14].

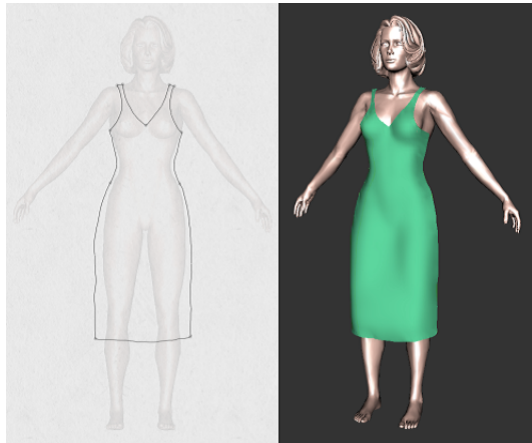


図 19 衣服の概形を描いて着せ付けるインタフェース [27].

のみであり、あらかじめ用意された動きをスケッチの形にあわせて組み合わせたり速度を変えたりして構成している (図 20)。また、人型のキャラクタのポーズを腕や脚などを一本の線で表した簡単な絵で設定し、それを連続して与えることでアニメーションを作成するシステムも提案されている [6]。この際問題となるのが、個々の線に対応した体のパーツの奥行き方向の傾きである。このシステムでは、あらかじめ与えておいた関節の可動範囲を考慮して自動的に決定する方法と、複数の候補を提示してユーザに選ばせる方法を組み合わせてこの曖昧性を解消している (図 21)。

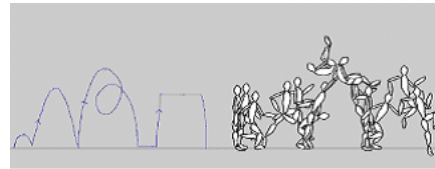


図 20 MotionDoodles の例 [25]。軌跡を描く。



図 21 棒人間を描くことによるポーズの設定 [6].

6 議論

効果的なスケッチインタフェースを作成するには、いくつか注意すべき重要な要素がある。ここではそのうちの何点かについて説明する。

まず、スケッチインタフェースにおいて創造的な活動を効果的に促進するためには、画面に表示されているのはあくまでも仮のデザインであり最終的な生産物ではない、ということをユーザにはっきりわかるように提示することが重要である。そのためには、わざと正確さを落として画面に表示する方法が有効である。すなわち、通常の GUI アプリケーションのように完全な直線や完全な陰影の着いた画像を表示しようとユーザはそれがすでに完成されたものと感じてしまうので、それを更に改良したり別の方法を探る意欲をなくしてしまいがちである。それに対し、わざとくずした表現を行うことで、それがまだ未完成であり曖昧性を含んでいることが明示され、瑣末な詳細でなくもっとも本質的な部分に注意を向けてデザインを改良する行為が促進される。例えば、SILK では認識した widgets を完全に変換して表示することはせず、敢えて手書きのままで表示している。SKETCH や Teddy においても、3次元データを画面に表示するときに、完全な輪郭線や陰影を描くのではなく、敢えて手書き風に崩して表現する手法 (Non-photorealistic rendering) が使われている。このような「崩した表現を利用することで自由な発想が促進される」という

点について実際に実験で示した例も報告されている[24].

他に、スケッチを利用したインタフェースを設計する上で注意が必要な点として、入力曖昧性への対処がある。ボタンやメニューによる入力には曖昧性がないので、システムは与えられた命令をそのままこなせばよい。しかし、スケッチ入力のように認識技術を利用したシステムでは、ユーザの入力に常に曖昧性が伴うので、それをどう解決するか工夫が必要となる。前述の表示上の工夫は、認識における曖昧性の影響を低減するという意味でも有効であるが、それ以外に、認識率を上げたり曖昧性を解消するインタフェースを提供したりといった工夫が可能である。

まず、認識率を上げる方法として、対象とするドメインをうまく限定すること、および入力が行われた時点での状況を曖昧性の解消に利用する方法などが利用されている。ドメインの限定については、すでに個々のアプリケーションの紹介で述べたとおりであり、限られた入力で複雑な操作を行うためにもっとも効果的な方法である。状況を曖昧性の解消に利用しているものとしては、SKETCH における、入力されたオブジェクトの位置はそのオブジェクトが描かれた位置に既にある他のオブジェクトの上に乗るように決定される、という 3 次元位置の決定アルゴリズムや、Teddy における、描かれた線が物体の上であれば単なる描画だが物体を横切っていれば切断である、というオペレーション決定手法などが挙げられる。

曖昧性を解消するインタフェースの工夫としては、複数の結果を提示して後からユーザに選ばせる方法や、操作を一旦取り消してもう一度行くと前段での認識結果と違う結果を提示する、といった方法が利用されている。たとえば、2 次元の図形整形システムである Pegasus[12] では複数の整形結果が考えられる場合にはそれらを表示してユーザに選ばせるといった手法が使われており(図 22)、2 次元の棒人間の絵から 3 次元のポーズを生成するシステム[6]でも同様に表示された複数の候補から選択する手法が使われている。入力結果を取り消して再入力すると別の認識結果を提示するという工夫は、Music notepad[9]やSketch[28]で利用されている。

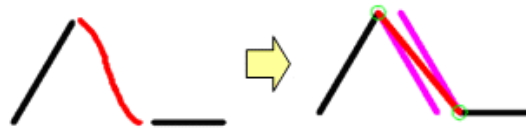


図 22 Pegasus における複数の認識結果の提示 [12]. クリックで好きなものを選ぶことができる。

7 まとめと今後の研究課題

本稿では、スケッチインタフェースに関する最近の研究事例について紹介し、また効果的なスケッチインタフェースを作成するために考慮すべき点について簡単な議論を行った。スケッチインタフェースは、もともと計算機があまり得意としていなかった「曖昧な情報の取り扱い」を実現しようとするものであり、単に実用的な側面だけでなくソフトウェアサイエンスの側面からもいろいろと面白い要素が含まれているものと考えている。本稿をきっかけに、この分野に興味をもっていただければ幸いである。

なお、スケッチインタフェースに関して残されている課題の一つとして、はじめて使うユーザに対する操作方法の提示をどうするかという問題がある。通常の GUI では、ユーザが行える操作はボタンやメニューとして一覧表示されているので、初めて見たユーザがどのような操作が行えるのかを理解するのはそれほど難しくない。しかし、スケッチインタフェースでは、極端な場合には、完全に真っ白なスクリーンを画面に提示するだけ、という状態であるので、初めて見たユーザは何をどう操作すればよいのかまったく見当がつかないといった問題が生じる。これは、自由な描画を入力として用いるというスケッチインタフェースにとっては根源的な問題であるともいえる。ジェスチャー一覧を提供したり、チュートリアルやマニュアルを工夫したり、といったことは考えられるものの、実際にはマニュアルを見ずに直接動かしたがるユーザが多いのが現実である。初めて触ったユーザに操作方法をどう提示し、その後の学習プロセスをどう支援していくのかといった点については、今後さらなる研究が必要である。

参考文献

- [1] Accot, J. and S. Zhai.: More than dotting the i's — foundations for crossing-based interfaces, Proc. Human Factors in Computing Systems, CHI'03 (2003), pp. 73–80.
- [2] Alvarado, C. and Davis, R.: SketchREAD: A Multi-Domain Sketch Recognition Engine, Proc. UIST 2004 (2004), pp. 23–32.
- [3] Apitz, G. and Guimbretiere, F.: CrossY: a crossing-based drawing application, UIST 2004 (2004), pp. 3–12.
- [4] Callahan, J., Hopkins, D., Weiser, M. and Shneiderman, B.: An Empirical Comparison of Pie vs. Linear Menus, Proc. CHI'88 (1988), pp. 95–100.
- [5] Cohen, J.M., Hughes, J.F., Zeleznik, R.C.: Harold: a world made of drawings, NPAR 2000 (2000), pp. 83–90.
- [6] Davis, J., Agrawala, M., Chuang, E., Popovic, Z., and Salesin, D.: A Sketching Interface for Articulated Figure Animation, SCA 2003 (2003), pp. 320–328.
- [7] Davis, R. 2002. Sketch understanding in design: Overview of work at the MIT AI lab. Sketch Understanding, Papers from the 2002 AAAI Spring Symposium (2002), pp. 24–31.
- [8] Gross, M.D. and E. Do.: Ambiguous Intentions: A paper-like interface for creative design, Proc. ACM Conf. on User Interface Software Technology (UIST)'96 (1996), pp. 183–192.
- [9] Forsberg, A., Dieterich, M., and Zeleznik, R.: The Music Notepad, Proc. of UIST '98 (1998), pp. 203–210.
- [10] Grossman, T., Hinckley, K. Baudisch, P., Agrawala, M., and Balakrishnan, R.: Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices, to appear in CHI 2006.
- [11] Hinckley, K., Baudisch, P., Ramos, G., and Guimbretiere, F.: Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli, CHI 2005 (2005), pp. 451–460.
- [12] Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H.: Interactive Beautification: A Technique for Rapid Geometric Design, (1997), pp. 105–114.
- [13] Igarashi, T., Matsuoka, S., and Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design, SIGGRAPH '99 (1999), pp. 409–416.
- [14] Igarashi, T. and Hughes, J.F.: Clothing Manipulation, Proc. of UIST'02 (2002), pp. 697–697.
- [15] Ijiri, T., Okabe, M., Owada, S., and Igarashi, T.: Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints, SIGGRAPH 2005, Vol. 24, No. 3, (2005), pp. 720–726.
- [16] Kurtenbach, G. and Buxton, W.: User Learning and Performance with Marking Menus, In Proceedings of CHI'94 (1994), pp. 258–264.
- [17] Landay, J. A. and Myers, B. A.: Sketching Interfaces: Toward More Human Interface Design, In IEEE Computer, Vol. 34, No. 3, (2001), pp. 56–64.
- [18] LaViola, J. and Zeleznik, R.: MathPad2: A System for the Creation and Exploration of Mathematical Sketches, ACM Transactions on Graphics (Proc. of SIGGRAPH 2004), Vol. 23, No. 3 (2004), pp. 432–440.
- [19] Lin, J., Newman, M. W., Hong, J. I. and Landay, J. A.: DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design, Proc. of CHI 2000 (2000), pp. 510–517.
- [20] Mynatt, Elizabeth D., Igarashi, T., Keith Edwards, W., and LaMarca, A.: Flatland: New Dimensions in Office Whiteboards, ACM CHI'99 (1999), pp. 346–353.
- [21] Okabe, M., Owada, S., and Igarashi, T.: Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing, Eurographics 2005 (2005), pp. 487–496.
- [22] Ramos, G., Boulos, M., Balakrishnan, R.: Pressure Widgets, Proc. of the SIGCHI conference on Human factors in computing systems, (2004), pp. 487–494.
- [23] Rubine, D.: Specifying Gestures by Example, In Proc. of ACM SIGGRAPH '91 (1991), pp. 329–337.
- [24] Schumann, J., Strothotte, T., Raddb, A., and Laser, S.: Assessing the Effect of Nonphotorealistic Rendered Images in CAD, Proc. of CHI'96 (1996), pp. 35–41.
- [25] Thorne, M., Burke, D., and Panne, M.: Motion doodles: an interface for sketching character motion, SIGGRAPH 2004, Vol. 23, Vol. 3, (2004), pp. 424–431.
- [26] Tolba, O., Dorsey, J., and McMillan, L.: Sketching with Projective 2D Strokes, UIST '99: Proc. of the 12th Annual ACM Symposium on User Interface Software & Technology (1999), pp. 149–157.
- [27] Turquin, E., Cani, M-P., and Hughes, J.F.: Sketching garments for virtual characters, Eurographics Workshop on Sketch-Based Interfaces and Modeling 2004 (2004), pp. ??–??.
- [28] Zeleznik, R.C., Herndon, K.P., and Hughes, J.F.: SKETCH: An interface for sketching 3D scenes, SIGGRAPH '96 (1996), pp. 163–170.
- [29] Zhao, S., P., Agrawala, M., Hinckley, K., and Baudisch, P.: Zone and Polygon Menus: Using Relative Position to Increase Breadth of Multi-Stroke Marking Menus, to appear in CHI 2006.