修士論文

顔認識の為のリアルタイム特徴抽出

平成 15 年 2 月 20 日受理

島根大学大学院 総合理工学研究科

数理・情報システム学専攻 計算機科学講座

指導教官 田中 章司郎

貫目 洋一

目次

第１章　序論

現在、個人認証の手段として、多種多様の生体情報をコンピュータでの個人認証に使う時代が到来している。例えば虹彩、声紋、顔、DNA、指紋などが列挙できる。その中で1990 年代後半から顔認識が実用領域に入ってきた。本研究ではこの顔画像を用いた個人認証システムの為の前処理に関しての研究である。

顔認証とは生体認証技術のひとつである。生体認識とは物理的な特性、属性に基づいて人間を自動的に認識するコンピュータ化された手法のことである。生体認識技術は高い認証精度を持っている。特に、DNA、指紋はほぼ 100％に近い認識率である。本研究で顔認識をターゲットにした理由として、我々が個人の認証に長年、顔の写真を利用してきたことにより親近性があるということである。また、その他の生体情報を使用した方法よりも速度、費用面において有利である。そのほかにも、既存のデータベースが使え、容易に使用でき、データ採取時に対象者に意識させる必要がないことや、立ち止まる必要が無い為に抵抗感が少ないこと、単純で人間によるバックアップが可能であるということである。これが DNA や指紋や虹彩のデータの場合には人間では個人を識別できない。さらに顔画像は保存データにより、追跡調査が可能であり遠隔からの監視も可能となる。

現在では海外はもちろん、日本でも NEC、富士通、東芝、沖電気、NTT データ等で顔認識の研究[1][2]が盛んに行われている。顔認識の短所としては事前に対象者の顔を撮影しなければならないこと、不特定多数の顔データを採取してしまうことである。このことはプライバシーの保護の問題に発展する可能性がある。データ採取時には撮影環境を管理する必要があること、また他の生体認証技術と同様に万能ではないことが挙げられる。顔認識はカメラにおける顔の発見である。これは画像全体から背景を取り除き、顔領域を抽出して顔の特徴を切り出すことである。顔の計測は顔の特徴部分の位置関係が重要な要素になる。人間はいつも同じ表情をして、同じ方向を向いているわけではないので、表情による認識や顔の向き、その変化の大きさからの認識も必要となる。認識方法として単純な背景からの顔の切り出しを行うことは簡単である。しかし、いつも単純な背景ばかりではない為、対象者が複数いる場合、複雑な背景だと顔の切り出しが難しくなる。これらのことはコンピュータ上では容易でない。

本研究では、100％に近い認識を目指すのではなく、不特定多数の対象顔画像の中から目的の顔画像を発見できるように、コンピュータによる顔認識の為の前処理にターゲットを絞った。認識環境が低価格で整えられるように、一般に広く普及している WDM ビデオキャプチャアーキテクチャを採用して、CCD カメラから送信されるストリーミングデータからリアルタイムに人物の顔領域を検出し、向き、光の加減を考慮して最も適しているデータを選び特徴点の抽出を行う手法について調査・検討した。

開発環境・実験環境

本研究で使用したソフトウェア、ハードウェア、言語及び OS は

・ハードウェア
CPU　　：Pentium4　1.5GHz
Memory：512MB

Camera：CMOS イメージセンサー 10 万画素 解像度 352×288 USB1.1 接続
　　：CCD イメージセンサー 30 万画素 解像度 640×480 PCI キャプチャボード接続
　　： デジタルカメラ FinePix50i 220 万画素 解像度 720×576 USB1.1 接続

キャプチャカード ：BT878 搭載 PCI キャプチャボード
　　：IEEE1394 準拠 PCI カード

・ソフトウェア
開発環境
 Microsoft
　Visual Studio 6.0 Enterprise
　DirectX SDK 8.1
 OS
　Microsoft Windows XP Professional

以上である。対象 OS に Windows を選択した理由は一般に最も普及している OS だからである。

第２章　個人認証技術

２．１　個人認証とは

個人認証とは、あらかじめ本人であることを登録しておき、その証拠を示すことによって本人であることを確認することを言う。

近年、アメリカで起きた同時多発テロに見られるように犯罪は増加の一途をたどっている為、これまで人で行ってきた個人の特定には限界がある。よってコンピュータ等を使用して不特定多数の対象から目的の人物を判別する方法が必要になってきた。例えば、銀行の現金自動預払機、空港でのテロリズム防止のための人物検査、コンピュータのセキュリティ管理、犯罪捜査などである。

現在一般的に行われている個人認証を、表 2.1 に示す。

表 2.1　個人認証の種類

方法	内容	例
所有物による方法	本人であることを証明するものを発行し，それを携帯する者を本人とみなす方法	運転免許証
知識による方法	本人しか知り得ない情報の提示により本人とみなす方法	暗証番号
生体情報を利用した方法	人間の身体，あるいは行動の特徴に基づいて，個人を自動的に認証する方法	指紋

２．２　個人認証の方法

コンピュータに個人を識別させる方法として、これまでに虹彩（アイリス）、声紋、顔、DNA、指紋といった様々な方法が考案され実用化されている。

・虹彩（アイリス）認証・・・ 沖電気工業「アイリスパス」

・声紋認証・・・・富士通「VoiceGATE」

・顔認証・・・・・東芝「FacePass」

・DNA 認証・・・ NTT データ「DNA 実印 IC カード」

・指紋認証・・・・NEC「SecureFinger」

各認証方法の特徴を比較したものを表 2.2 に示す。最も正確な方法は DNA を使う方法であるがコストがかかり容易に使うことができない。手軽に個人認証を使用するには扱いが簡単な虹彩か、顔を使った方法となる。扱いやすさに加えて認識される側の抵抗感が少ないものは顔となる。

表 2.2　各認証方法の比較

方式	扱いやすさ	経年変化耐性	偽造のしにくさ	本人拒否の少なさ	他人拒否率	コスト
虹彩	◎	○	◎	○	○	○
声紋	○	×	×	△	×	○
顔	◎	×	○	×	△	○
DNA	△	◎	◎	◎	◎	×
指紋	△	◎	◎	○	○	○

そこで本研究では、この中でも非接触で実現でき、認識される側の抵抗感が少なく、認識装置の故障時にもオペレータがチェックすることが可能で、不特定多数を対象とすることができる顔画像を用いた個人認証に注目した。

２.３　顔認識とは

人間の顔をコンピュータで認識させるという試みは、顔のパターン認識の問題として古くから行われてきた。コンピュータの性能向上とともに画像処理技術が発達し、近年ではリアルタイムでの画像の認識と合成がある程度可能になったため、顔画像の認識と生成技術を統合した顔情報処理システムに関する多くの研究が行われている。

顔の認識には、顔の発見、個人識別、表情認識などの課題がある。

２.３.１　顔認識の複雑さ

人間は、何の苦労もなく瞬時に人の顔を認識しているが、これをコンピュータに処理させた場合は難しい問題となる。

顔認識には処理すべき要素が２つあり、それぞれに課題がある。最初の障害が顔の検出である。システムは、画面の中に顔が存在するかしないかを判断しなければならない。その上で顔があれば、顔領域を切り出し、それが使えるかを判定する。次に、識別の為の特徴抽出の問題である。顔領域を抽出した後、更に特徴を抽出して数値化し、データベースに格納された顔データと整合する必要がある。

顔の検出や認識が困難である理由は、これらが固定の画像パターンで表されないことで

ある。対象者は、いつ、どんな距離で、またいろいろな速度で、様々な照明条件や背景のもとで現われるか分からない。さらに写真に示すように顔の表情、向き、髪型、顔の毛、（眉毛やひげ)、眼鏡のリードなど時間の経過につれて変化し、同じ顔でもおびただしい量の多様性が生じる(図 2.1)。

図 2.1　顔の多様性

以上のように、認識のたびに、同じ顔の向き、表情にすることは通常困難な為、顔の向き、表情の影響を受けにくい方法を採用することが不可欠である。更に顔認識に要求される条件は、実時間で認識できることである。

顔認識の要求条件をまとめると次のようになる。

１．実時間認識
２．顔の向きや表情変化への対応
３．照明変動への対応
４．経年変化への対応

これまで開発されている顔画像認識法は、顔の“幾何学的な情報に基づく方法”と“顔パターンに基づく方法”に大きく分類できる。前者は、瞳、目がしら、目じり、口端などの特徴点の幾何学関係をパラメータ化したものを特徴量とする。後者では、顔全体を濃淡パターンとしてとらえ、これをベクトルに展開したものを特徴量とする。顔認識の研究が始まった当初は幾何学的な方法が主流だったが、最近ではパターンに基づく方法が主流となって

いる。しかしこの方法は、顔の変動の影響を受けやすく、パターンから正面の検出が困難な為、本研究では前者の方法を採用する。

２.３.２　顔の認識

人間は画像がぼやけていても、照明条件や背景が大きく変化しても、目に入ってくる情報の中から顔を即座に見つけ出す能力がある。また、髪型や化粧、見る角度、表情が異なっても、個人を識別することができる。更に、笑った顔、怒った顔など基本的な表情は相手が変わっても共通に理解することができる。顔の認識をコンピュータで行う際に問題となる顔の発見、個人識別などについて以下に述べる。

室内など背景のあるシーン中の顔認識では、まず画面中の顔がある場所と他の物体とを区別して切り出す必要がある。顔の発見は画像処理の対象領域を決定するための前処理であるが、これを容易にするために単色の背景を使用したり、カメラとの位置関係を制限したりすることが多かった。顔の発見の手がかりとして、肌の色、動き、形、大きさなどが考えられる。この中で、色はカメラからの位置によらずに比較的簡単に使えるので頻繁に利用されている。つまり、肌の色の部分を画像から抜き出して次の処理の対象にするわけである。コンピュータで取り込むカラー画像は、赤、緑、青（RGB）の３成分に分解されているが、肌色抽出を問題にする場合はこの 3 成分での表現はあまり有効ではない。なぜならば、同じ色でも明るさが変化すると RGB の値が大きく異なってくるからである。そこで、本研究では色相、色彩、明度（HSV）の 3 成分による色表現に変換し、人の肌の色相を抽出する方法を利用する。

顔は目、鼻、口等の部品から構成され、形状や位置関係は個人による多少の違いはあっても、ほぼ共通した特徴を持つ。これらの人の顔が共通して持つこれらの一般的な特徴を用いて、個人認識の為のパラメータとする。

２.４　従来手法との比較

本研究の手法では、2.2 節で紹介した手法と比較して以下のような特徴がある。

・対象者が必ずしも立ち止まる必要がない。
・低価格化が進んだ USB 接続の CCD カメラを使用したことで、USB ポートのあるパソコンなら簡単に接続でき低価格で認識環境を整えられる。
・PC と WDM ドライバを使用する CCD カメラのみでシステムが構築可能。
・空港や駅の改札などに設置し、不特定多数の対象者からデータマイニングできる。
・目・鼻・口の位置などの特徴をとらえてリアルタイムにデジタルデータ化できる。
・抽出時の顔画像が残る為、アクセス履歴の監査確認作業が容易である。

２.５　提案手法の位置づけ

提案手法は、100%の認識率を目指す従来の手法と違い、不特定多数の対象の中から、目的の顔画像に近いものを抽出していくデータマイニングの要素が大きい。その為、他人誤認率はこれまでの方法よりも高いが、立ち止まる必要がない為、認識されているということを対象者に意識させなくてよい。

第３章　顔の選択

３.１　ビデオキャプチャ

ビデオキャプチャとは、パソコンに動画を取り込む事である。その用途により、様々な取り込み方があるが、その中でも現在最も普及している Windows Driver Model (WDM) 規格[3]に準拠したビデオキャプチャシステムを特徴点抽出に使用した。

３.１.１　ＷＤＭビデオキャプチャ

WDM ビデオキャプチャは、1992 年から使用されている Video for Windows アーキテクチャの固有の問題を解決するためにマイクロソフトによって開発された[4]。WDM ビデオキャプチャの主な利点は次のとおりである。

・32 ビットドライバである。
・DirectShow(3.13 節参照)が使用できる。
・ビデオキャプチャデバイスと DVD/MPEG デバイスの間で共有される、ハードウェア(ビデオポートやチップセットなど)用の単一クラスドライバアーキテクチャである。
・テレビチューナーと入力選択、およびフィールド、バーティカルブランクインターバル(VBI)、ビデオポートエクステンション(VPE)がサポートされる。
・Windows プラットフォームで動作する 1 種類のドライバである。

図 3.1 で、WDM キャプチャアーキテクチャの 3 つの基本コンポーネントを示す。

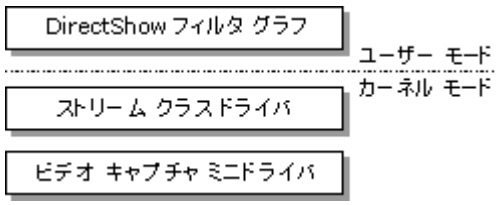


図 3.1　WDM の基本コンポーネント

WDM キャプチャアーキテクチャは DirectShow とスムーズに統合できるように設計されているため、WDM キャプチャフィルタを使用して DirectShow で簡単にキャプチャグラフを構築できる。WDM キャプチャフィルタは DirectShow からストリーミングクラスに制御メッセージを送信する。Windows デバイスドライバキット(DDK)に含まれる KsTune.ax、KsXbar.ax、及び KsCap.ax フィルタを使用すると、USB カメラ、1394 DV デバイス、TV、デスクトップカメラのデータ等の WDM ストリーミングデータをストリームクラスによって制御し、DirectShow キャプチャグラフに送信することができる。図 3.2 で、これらのコンポーネントがどのように基本アーキテクチャに統合されているかを示す。

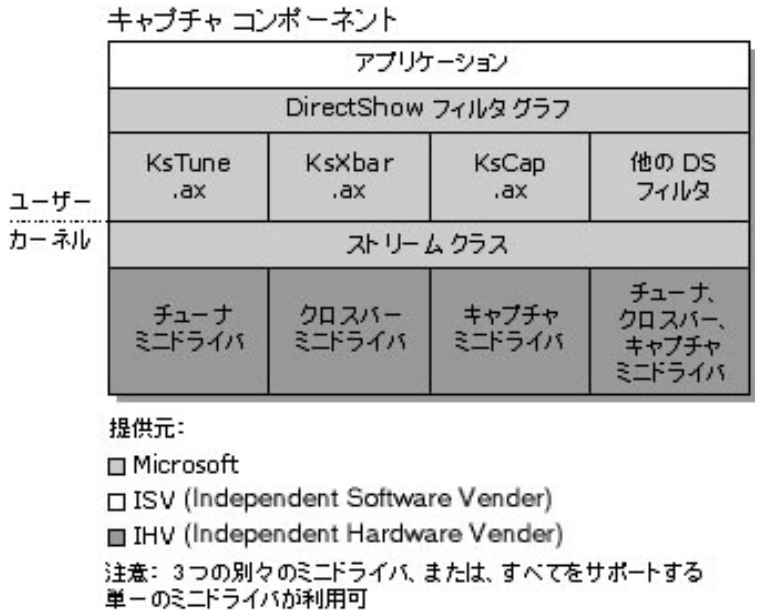


図 3.2　WDM Video Capture Architecture

図 3.2 では、KsTune.ax、KsXbar.ax、KsCap.ax、およびその他の DirectShow フィルタが、ストリームクラスと直接通信することを示す。

KsTune.ax、KsXbar.ax、及び KsCap.ax フィルタには、それぞれをサポートするミニドライバ（または 3 つすべてをサポートするミニドライバ）が必要である。ビデオキャプチャミニドライバは、ストリームクラスのクライアントであり、ビデオイメージと関連データのストリームを生成するハードウェアデバイスを制御する。

３.１.２　ＤｉｒｅｃｔＸ

DirectX とは Microsoft 社の「画像」「音」「通信」などのマルチメディア・データを扱うプログラムに対して、ハードウェアへの高速なアクセスを提供するＡＰＩ群の総称である[5]。種類はランタイム版と SDK(Software Development Kit)がある。前者は DirectX で作られたアプリケーションを動作させる為に必要で後者はそのアプリケーションを作るために必要である。

DirectX は開発環境や開発言語ではなく、COM(Component Object Model)のひとつである。COM オブジェクトは、インターフェイスを介して様々な開発環境から使うことが可能である。そのため、DirectX は Visual C++ からでも、Visual Basic からでも使用できる。DirectX を使えば、描画処理が他のプログラムに干渉されないため、高速なパフォーマンスが期待できる。

DirectX は、描画やサウンド、マルチメディアなどに応じて表 3.1 のような各要素に分かれている。

表 3.1　DirectX の要素[5]

名前	役割
DirectXGraphics	グラフィックスプログラミングのための API を提供する DirectX7 以前の DirectDraw と Direct3D を統合している
DirectXAudio	サウンドプログラミングのための API を提供する DirectX7 以前の DirectSound と DirectMusic を統合している
DirectInput	入力装置に関する API を提供する
DirectPlay	マルチプレイヤーネットワークプログラミングをサポートする
DirectShow	高品質なマルチメディア再生をサポートする
DirectSetup	DirectX コンポーネントのワンコールインストールを提供する

本研究ではこの中の映像や音声のストリーミング技術を提供する API、DirectShow をウィンドウモードで使用している。

3.1.3　DirectShow

DirectShow は、アプリケーションがフィルタと呼ばれる一連の相互に接続されたオブジェクトを通してデータをストリーミングできるようにするストリーミングアーキテクチャである[5]。DirectShow フィルタの集まりはフィルタグラフと呼ばれる。

DirectShow フィルタは、ソースフィルタ・変換フィルタ・レンダラの主に 3 つのカテゴリに分類される。ソースフィルタはデータを作成し、これを次のフィルタに送り込む。変換フィルタはデータを受け取って転送する。場合によっては複数のスレッド上で処理を行う。レンダラはデータの受け取りのみを行っている。

すべての DirectShow フィルタは、ピンと呼ばれる接続ポイントを少なくとも 1 つは持っている。フィルタはピンを介して他のフィルタに接続される。メディアデータは、ピン接続を通って、フィルタ間で移動する。

フィルタグラフは、停止・ポーズ・実行中・遷移の 4 つの状態を取る。遷移状態では、グラフは 1 つの状態から別の状態に変化しつつあり、DirectShow のマルチスレッド性のために、まだその変化を完了していない。

ほとんどのフィルタは、ポーズと実行中の状態は同じものとなる。ソースフィルタは新しいデータを生成し、変換フィルタは処理のために新しいデータを受け付ける。この規則の例外は、ライブキャプチャフィルタとレンダラフィルタである。ライブキャプチャフィルタは実行中にのみデータを送信し、ポーズ中にはデータを送信しない。レンダラフィルタはポーズ中にデータのレンダリングを停止し、新しいデータは受け付けない。

停止したフィルタは、データの処理や新しいデータの受け付けは行わない。ワーカースレッドをシャットダウンし、使用中の他のリソースをすべて解放する。

DirectShow のフィルタは、カーネルモードのストリームクラスドライバからデータを操作する強力で比較的安全な方法となっている。DirectShow は柔軟なため、様々なフィルタ構成が可能である。図 3.3 は、ビデオのプレビューとディスクへのキャプチャを同時に行うユーザーモード DirectShow フィルタの構成例を示す。

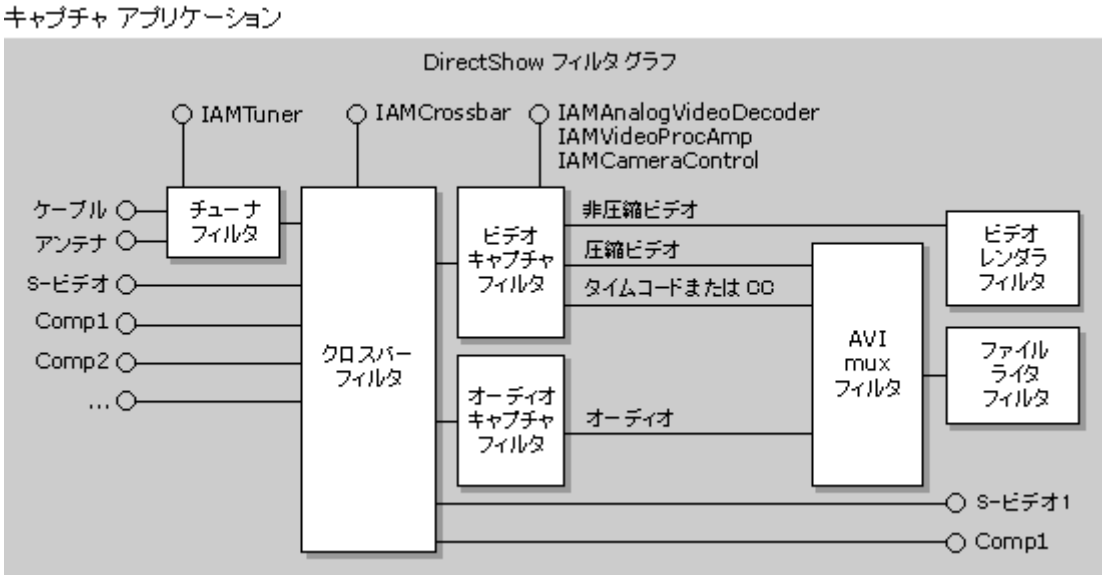


図 3.3　DirectShow フィルタの構成例

図 3.3 は、受信 TV 信号がチューナーフィルタを使用してチューニングされ、クロスバーフィルタを使用してルーティングされる様子を示す。フィルタグラフは、ディスクに保存する様々なストリームのデータをビデオまたはオーディオキャプチャフィルタに渡す。これには、オーディオストリーム、ビデオストリーム、および SMPTE タイムコードやクローズキャプションデータなど､さまざまなストリームのその他の付加データが含まれる。

３.１.４　キャプチャ処理の流れ

USB カメラから画像情報を表示、キャプチャする処理の流れ[6]を図 3.4 に示す。

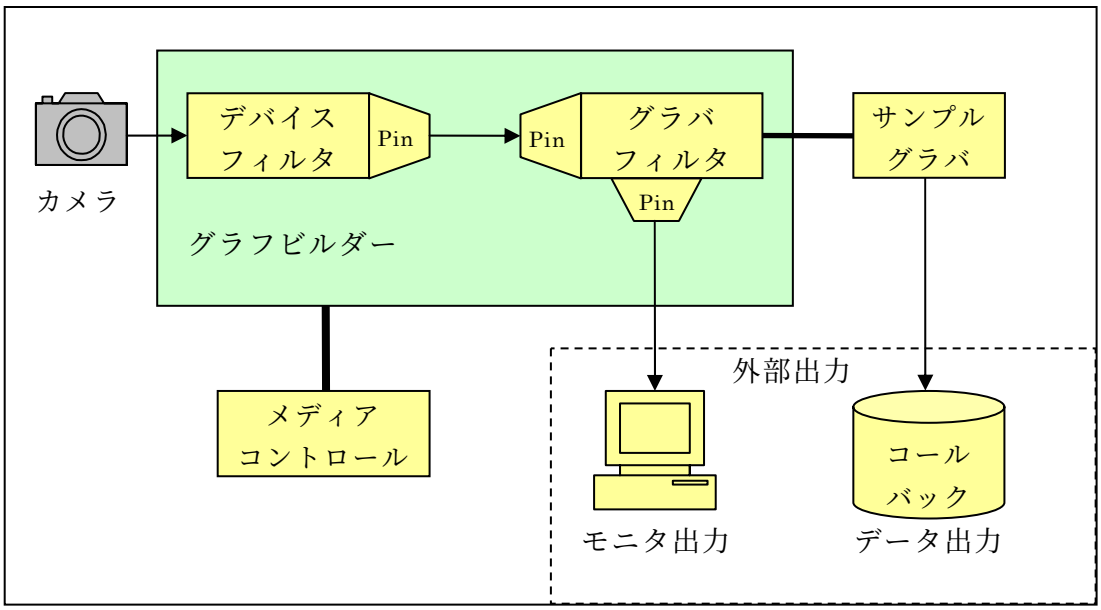


図 3.4　キャプチャ処理の流れ

基本となる DirectShow オブジェクトは以下の 5 つとなる。

①グラフビルダー　　(IGraphBuilder)
②デバイスフィルタ　　(IBaseFilter)
③グラバフィルタ　　(IBaseFilter)
④サンプルグラバ　　(ISampleGrabber)
⑤メディアコントロール (IMediaControl)

USB カメラから入力された画像が“②デバイスフィルタ”→“③グラバフィルタ”→ディスプレイへと出力される。また“④サンプルグラバ”は“③グラバフィルタ”からデータを取得し、BYTE 型の画像データをコールバックとして外部出力する。この“コールバック”を参照することで、キャプチャ画像を取得することができる。(コールバックは裏で何回も呼ばれる処理で、スレッドと OnTimer を組み合わせたような動作を行う。)
“②デバイスフィルタ”と“③グラバフィルタ”とディスプレイは“①グラフビルダー”によって接続され、“⑤メディアコントロール”によって全体のスイッチが入り、USB カメラ画像が全体に流れる。

以下で処理の流れを細分化し、それぞれの処理をコード化する。
まず、DirectShow ヘッダをインクルードする。

表 3.2　DirectShow ヘッダ

```
#include <dshow.h>
#include <streams.h>
#include <qedit.h>
```

以下の 5 つの DirectShow オブジェクトがキーワードとなる。

グラフビルダー	(IGraphBuilder)	m_pGraph
デバイスフィルタ	(IBaseFilter)	pSrc
グラバフィルタ	(IBaseFilter)	pGrabber
サンプルグラバ	(ISampleGrabber)	m_pGrabberInterface
メディアコントロール	(IMediaControl)	pMediaControl

これらを生成、接続していくのが基本的なプログラムの流れとなる。

表 3.3　ビデオキャプチャの基本的流れ

1. フィルタグラフ作成　　CoCreateInstance(CLSID_FilterGraph)
2. キャプチャデバイスの取得 ICreateDevNum, IEnumMoniker, IMoniker, IBaseFilter
3. キャプチャビルダグラフ　ICaptureGraphBuilder2
3-1. キャプチャビルダグラフ作成 CoCreateInstance(CLSID_CaptureGraphBuilder2)
3-2. フィルタグラフへの関連付け ICaptureGraphBuilder2::SetFilterGraph((IFilterGraph)pGraph)
4. ファイルライタフィルタ　IFileSinkFilter
4-1. フィルタグラフへの追加　ICaptureGraphBuilder2::SetOutputFileName(...)
4-2. ストリームのレンダリング設定　ICaptureGraphBuilder2::RenderStream(...)
5. キャプチャ開始　IMediaControl::Run()

表 3.4　デバイスの列挙

2-1.デバイスの列挙子を取得 ICreateDevEnum CoCreateInstance によってインスタンス生成
2-2.カテゴリの列挙子を取得 IEnumMoniker　ICreateDevEnum::CreateClassEnumerator (カテゴリ名, ...) によって入手
2-3.インターフェイスを取得　IMoniker　IEnumMoniker::Next によって列挙
2-4.フィルタにバインド　IBaseFilter　IMoniker::BindToObject(..., (void **)pSrc)

表 3.5　フィルタプロパティの取得

1.フィルタ列挙子の取得	IEnumFilters　IFilterGraph::EnumFilters
2.フィルタの列挙	IBaseFilter　IEnumFilters::Next(1, (IBaseFilter **), (ULONG *))
3.フィルタ名の取得	FILTER_INFO　IBaseFilter::QueryFilterInfo(FILTER_INFO *)

ストリーミングデータからフレームを切り出して特徴を抽出するには DirectShow でスチルキャプチャをする必要がある。スチルキャプチャは、ストリーミングデータの流れの中に網を仕掛けておくイメージで、ビデオストリームの中にサンプルグラバを挿入する。これによって、フィルタを通過していくサンプルを獲得できる。この操作は SampleGrabber フィルタを作成し、ISampleGrabber インターフェイスを用いて行う。

具体的には、ISampleGrabberCB インターフェイスを実装したクラスを作成した後に、ISampleGrabber::SetCallBack()によって関連付ける。

基本的流れは表 3.3 のビデオキャプチャと同様であるが、スチルキャプチャでは、受け取るサンプル毎にコールバックを呼び出すことができるので、これを利用してバッファの内容をファイル、メモリに書き出す。コールバック関数としては BufferCB を実装する。BufferCB は、データへのポインタや長さを直接取得できる。

表 3.6　スチルキャプチャの基本的流れ

0. ISampleGrabberCB インターフェイスの実装
1-3　表 3.3 と同様の処理
4-1. サンプルグラバの生成
CoCreateInstance(CLSID_SampleGrabber, ..., IID_IBaseFilter, ...)
IBaseFilter::QueryInterfase(IID_ISampleGrabber, ...)
4-2. メディアタイプの設定　ISampleGrabber::SetMediaType(AM_MEDIA_TYPE *)
4-3. フィルタグラフへ追加　IFilterGraph::AddFilter(IBaseFilter *)
4-4. サンプルグラバの接続
4-5. グラバのモードを適切に設定　SetBufferSample(FALSE), SetOneShot(FALSE)
SetCallback()：コールバックメソッドを指定
第 1 引数：ISampleGrabberCB 実装クラスのインスタンスへのポインタ
第 2 引数:どちらのメソッドを用いるか. 0 なら SampleCB, 1 なら BufferCB
5. キャプチャ中にグラブを行う　ISampleGrabber::GetCurrentBuffer(...)

キャプチャを行うにはピンを得る必要がある。

表 3.7　ピンの列挙

1. ピン列挙子の取得	IEnumPins	IBaseFilter::EnumPins
2. ピンの列挙	IPin	IEnumPins::Next(1, (IPin **), 0)
3. ピンの方向(IN,,OUT)の取得	FILTER_INFO	IPin::QueryDirection(PIN_DIRECTION *)

表 3.8 にピンを得る関数を示す。

表 3.8　ピンを得る関数

```
IPin *GetPin(IBaseFilter *pFilter, PIN_DIRECTION PinDir)
{
  BOOL          bFound = FALSE;
  IEnumPins   *pEnum;
  IPin          *pPin;
  pFilter->EnumPins(&pEnum);
  while(pEnum->Next(1, &pPin, 0) == S_OK)
    {
      PIN_DIRECTION PinDirThis;
      pPin->QueryDirection(&PinDirThis);
      if (bFound = (PinDir == PinDirThis)) // 引数で指定した方向のピンなら break
        break;
      pPin->Release();
    }
  pEnum->Release();
  return (bFound ? pPin : 0);
}
```

コールバックインターフェースを実装するには、ISampleGrabberCB インターフェイスを継承してメソッドを実装する。ただし、これは COM に基づいて実装する必要がので、簡略化のため CUnknown も継承する。これによって、IUnknown の AddRef、 Release、QueryInterface といったメソッドを最初から実装する手間が省ける。その代わり、NonDelegatingQueryInterface を実装する。

表 3.9　コールバックインターフェースを実装したスチルキャプチャ部分

```
class CGrabCB: public CUnknown, public ISampleGrabberCB
{
public:
  DECLARE_IUNKNOWN;

  STDMETHODIMP NonDelegatingQueryInterface(REFIID riid, void **ppv)
  {
    if( riid == IID_ISampleGrabberCB ){
      return GetInterface((ISampleGrabberCB*)this, ppv);
    }
    return CUnknown::NonDelegatingQueryInterface(riid, ppv);
  }

  // ISampleGrabberCB のメソッド
  STDMETHODIMP SampleCB(double SampleTime, IMediaSample *pSample)
  {
    return E_NOTIMPL;
  }

  STDMETHODIMP BufferCB(double SampleTime, BYTE *pBuffer, long BufferLen)
  {
    cerr << "Sample time: " << SampleTime  << "\t";
    cerr << "BufferLen: " << BufferLen;
    cerr << endl;
    SaveOneImage("C:\\tmp.yuv", pBuffer, BufferLen); // 後に定義する
    return S_OK;
  }
  // コンストラクタ
  CGrabCB( ) : CUnknown("SGCB", NULL){
}
};
```

３.２　人領域の検出

リアルタイムに人物の特徴を抽出するにはビデオストリームの中から人物のいるフレームを検出する必要がある。また、人物のいない部分に対して処理を行っても無意味である為、その間は処理を止める。

実験では人領域の抽出の為に、カメラは固定されていることを前提とし、何かがカメラの前を通る時は、フレーム間に差分があることを利用して、フレーム間の動き差分をリアルタイムに求め、あらかじめ決めた値以上の差分が生じた際に人が通ったと判断し、認識処理を開始する。

３.２.１　フレーム間差分の抽出

3.1.3 節で説明した DirectShow では CaptureGraphBuilder コンポーネントを使ってビデオデバイスからのキャプチャプログラムを作ることがでる。

この場合、FilterGraph コンポーネントは CaptureGraphBuilder の配下で動作することになる。CaptureGraphBuilder 配下の FilterGraph にグラバを組み込むことによって、キャプチャ中のフレームデータを直接 DIB フォーマットのデータとして得ることができ、得られたデータと直前のサンプルとの差分をとることで動きをとらえることができる。[7]

つまり、ビデオストリームから現在のフレームと、前のフレームとの差分をメモリ上で取り、作成した差分フレームを新しいバッファに代入することにより、差分画像を取得する。

作成したフレーム間差分をとるソフトの実行画面を図 3.5 に示すが、差分のプレビューは、黒くなり人間の目では分かりにくいので、分かりやすいように反転した画像も表示している。

元画像　　差分画像　　差分画像の反転したもの

図 3.5　フレーム間差分の取得例

以下の表 3.10 にフレーム間差分取得部分のソースを示す。

表 3.10　フレーム間差分の検出

```
while( !(img00.hwnd) );                                    // ウィンドウの表示待ち
    pGrab -> SetBufferSamples(TRUE);          // グラブ開始
    pMC -> Run();                                          // レンダリング開始
    double s = 0.0;
    while(1){
        hr = pGrab -> GetCurrentBuffer( &n, (long *)buffer );
    // グラブ
        s =0.0;
        for( i = 0;i < n ;i++){
            s = s +  ( buf2[i]= fabs(buffer[i] - buf3[i] ));
        }
        if( s/n > 10 )                          //差分判定
            Color Extraction();
        InvalidateRect( img00.hwnd, NULL, FALSE);
        InvalidateRect( img01.hwnd, NULL, FALSE);
        if( kbhit() ){
            getch();                            // kbhit で取得したキーの破棄
            hr = pMC -> Pause();
            if(getch() == 'q')  break;
            pMC -> Run();
        }
        CopyMemory( buf3, buffer ,n);
    }
    pMC -> Stop();
    pGrab -> SetBufferSamples( 0 );
```

以上の操作を行いビデオストリームの動きを検出し、次項の顔領域の検出へと進む。

３.３　顔領域の検出

顔画像を用いた個人認証を行う為には、3.2 節の方法で取り出した人を含むと思われるビデオストリームの中から人の顔の部分を抜き出さなければならない。もし、顔領域がない場合は抜き出したビデオストリームに人はいないものとして破棄する。

顔領域を精度よく抽出し、顔領域外のデータに対して特徴抽出を行わないようにすることにより、誤った特徴点を取りにくいようにする。顔の抽出には肌色情報を使用した。

３.３.１　表色系

カラー画像の表現方法を総称して表色系(color specification system)[8]とよぶ。表色系の代表的なものに、RGB 表色系がある。この表色系は、人間の網膜上にある色を感じる 3 種類の細胞（錐体細胞）が、それぞれ波長 435.8nm の青(Blue)、546.1nm の緑(Green)、700.0nm の赤(Red) に感度のピークを持つことから、これらの波長の単色光を 3 原色と定め、他の色をこれらの 3 原色の混合で表現するものである。コンピュータのディスプレイなどは、この RGB 表色系を元にして色表現をしている。

しかし、RGB 表色系では以下のような問題点がある。

1. 人間の色表現特性は、色相、明度、彩度の各々の変化に対して違った尺度を持っているので各情報は分離して考えなくてはならないが、RGB 空間においては R・G・B の３属性間の相関が強く、各情報を独立に扱うことが難しい。

2. RGB 空間において２つの色を２点で表した場合、２点間の距離と人間の感じる色差は場所によって異なり比例しない。

上記の問題点は、コンピュータで画像を処理する場合にも生じてしまう。これらの問題点を解決するために考案された表色系が HSV 表色系である。

HSV 表色系は赤→黄→緑→青→紫を基調色相とし、これらの中間色とともに円周上に並べた色相環に基づく色相(Hue)、無彩色から有彩色への鮮やかさの度合いを表した彩度(Saturation)、色の明るさを表す明度(Intensity Value) からなっている。カラーテレビの色調整がこれと似ている。

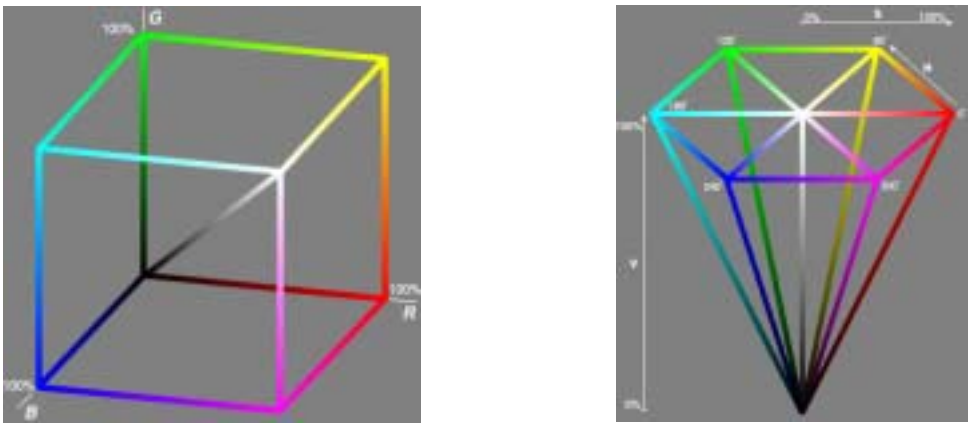


図 3.6　RGB 表色系と HSV 表色系の色立体

RGB 表色系、HSV 表色系とも 3 つの数字を使って色を表す表現方式である。各表色系で表現できる主な色を図 3.7、図 3.8 に示す。

	R	G	B
Black	0	0	0
Red	255	0	0
Green	0	255	0
Yellow	255	255	0
Blue	0	0	255
Magenta	255	0	255
Cyan	0	255	255
White	255	255	255

図 3.7　RGB 表色系の基本色

	R	G	B
000度	255	0	0
060度	255	255	0
120度	0	255	0
180度	0	255	255
240度	0	0	255
300度	255	0	255
360 度	255	0	0

図 3.8　HSV 表色系の色円環(色相)

３.３.２　チャンネル分解

図 3.9 のイメージを RGB 表色系、HSV 表色系で分解した。
このイメージは色相を円に沿って 1 周させたものを外側は白く、内側は黒くしたものである。

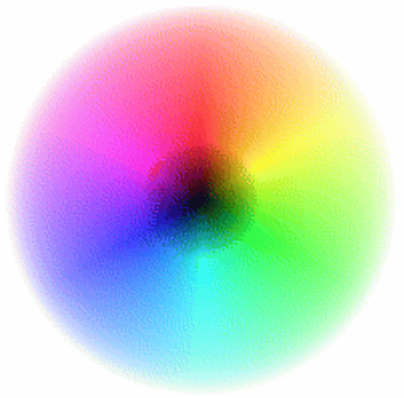


図 3.9　色円環

・RGB 表色系で分解
　白い部分が対応する色の箇所で、黒い部分にはその色はない。
図の外側が白いので全ての外側は白くなり、中心は全て黒くなる。

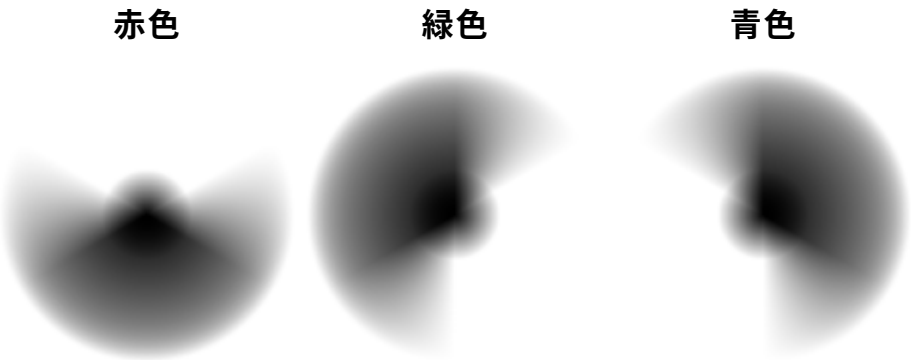


図 3.10　色円環の RGB 表色系分解

・HSV 表色系で分解
　色相が円に沿ってきれいに一周しているのが分かるほか、彩度と明度についても図 3.11 の様になっているのかが分かる。

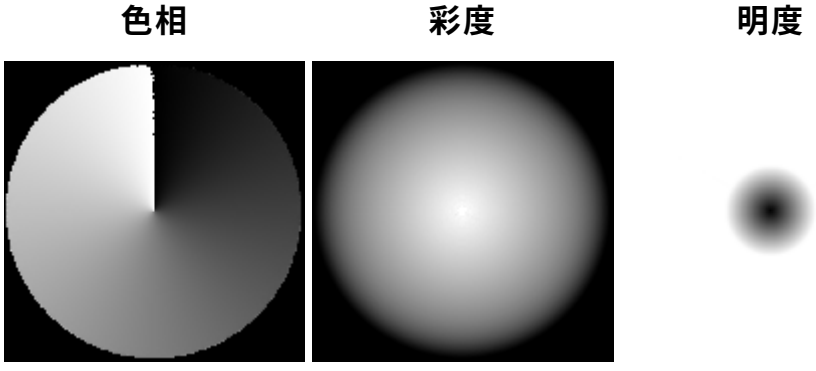


図 3.11　色円環の HSV 表色系分解

プログラム中で使用した RGB 表色系→HSV 表色系の変換式[8]は以下のとおりである。

表 3.11　RGB 表色系→HSV 表色系の変換式

・Hue(色相)の求め方

```
max = max (r, g, b);
min = min (r, g, b);

if (g == max) {
   hue= (b - r) / ( max - min ) * 60 + 120;
}
else if (b == max) {
  hue= (r - g) / ( max - min ) * 60 + 240;
}
else if (g < b) {
  hue= (g - b) / ( max - min ) * 60 + 360;
}
else {
  hue= (g - b) / ( max - min ) * 60;
}
```

・Saturation(色彩)の求め方

```
max = max (r, g, b);
min = min (r, g, b);
Saturation = ( max - min ) / max * 100;
```

・Value(明度)の求め方

```
Value = max (r, g, b) / 2.55
```

・HSV の値の範囲

H = 0～359　S = 0～100 V = 0～100

３.３.３　肌色検出

作成したプログラムでは人間がいるかどうかの判別に肌色を用いた。肌色とは人の肌の色のことであるが､人により千差万別で一人一人違いがある｡同じ人物の肌でも手のひら、手の甲､顔､首､細かい所では血管の部分などでそれぞれ肌の色は異なって見えてしまう。これらの色がひとまとまりにして肌色と呼ばれている。

カメラなどから得られたストリーミング映像から、プログラムに肌色を認識させる為には、プログラムに肌色とは何かということを判別させなければならない。

そこで、デジタルカメラ(FinePix50i)で撮影した人物写真の肌の RGB 値を調べてみた。

表 3.12　撮影場所による RGB 表色系、HSV 表色系の肌色の値

肌の色										
R（赤）	180	208	141	173	159	184	136	206	246	105
G（緑）	139	145	96	120	101	125	89	171	223	52
B（青）	119	130	77	112	94	121	61	151	217	44
H（色相）	19	11	17	8	10	4	22	21	12	8
S（色彩）	34	38	45	35	40	34	57	26	12	58
V（明度）	71	82	58	68	62	72	53	81	97	41

使用した写真サンプルは数日に分けて撮影し、晴れや曇り、野外や室内、暗い所や明るい所などが混在してある為、RGB 値では値の差が激しい事がよく分かる。表 3.12 に RGB 表色系の値を上段に HSV 表色系の値を下段に示して比較した。

HSV 表色系は色相「H」と明度「V」が独立しているため、写真の明るさを気にせずに色合いを調べることができる｡上の表では H の値が 0～359 の範囲であるにもかかわらず、どの肌色もあまり変わらない大きさになっていることが分かる。

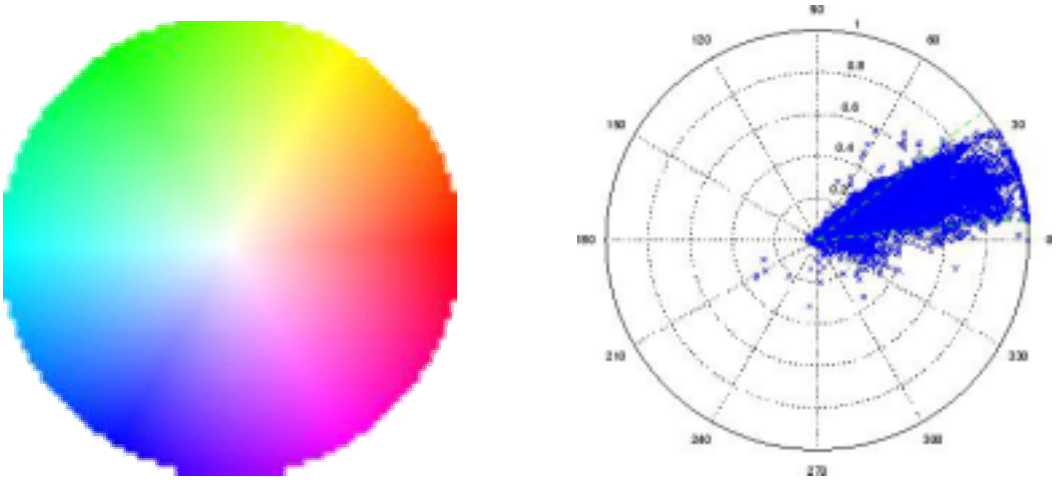


図 3.12　多数のサンプルの H、S 値を色相関にプロットしたもの[9]

図 3.12 は多数の肌のサンプルを H と S の値を色相環にプロットしたものである。色相環（H[0～359]）は 3 時方向を 0 とし、反時計回りに値が大きくなる。S[0～100]は色彩を表し、色相環の中心からの距離で表している。V[0～100]は明度であり、色相環の円に直行する軸を取る。V の値が小さいほど色が暗くなっていく。

H の値は、1～8 が日本人、9 が白人種、10 が黒人種の肌データとなる。先行研究[9]では H の範囲を 6～38 としているが､本研究では H の値の 0～30 の間を肌色として扱った。

３.３.４　ＲＧＢ値を使用した肌色検出

肌色検出には HSV 値を使用した方法が最も有効と考えられるが、他の方法とも比較する為に RGB 値をした方法についても検証してみた。

検証方法は､3.2.1 節の方法で人が含まれていると思われる領域をメモリ内に切り出した後、肌色(#FFCC99)情報を参考にして、図 3.13 のように RGB 毎にしきい値を設けて各成分を 2 値化するフィルタリングを行う。フィルタリングによって肌色の部分を白(#FFFFFF)に変化させ、最も大きな白色領域を顔として検出した。

しかし、肌の色には個人差があるのでフィルタリングを全自動とすることはできず、一部手動操作が必要となった。また、肌色に近い服装、手を上げている場合なども領域を誤認識し、背景によっては全く検出できなかった。

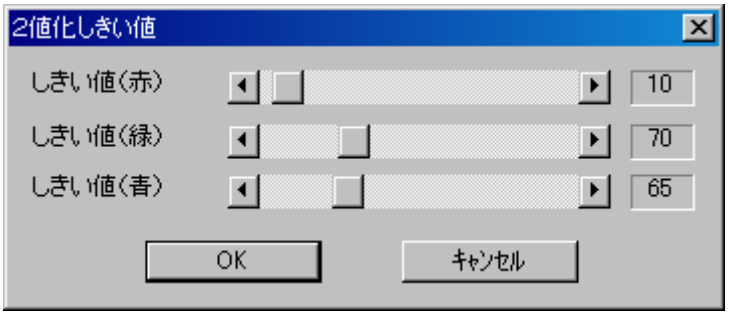


図 3.13　RGB 各成分の 2 値化しきい値ダイアログ

図 3.14　肌色領域の検出結果

３.３.５　ＨＳＶ値を使用した肌色検出

RGB 値を使用する方法は、前節で述べたような問題が発生することが確認できた。変換式は表 3.11 を使用し、H が 0～30 の画素を白(#FFFFFF)、残りの画素は黒(#000000)としている。

図 3.15-3.17 に HSV 値を使用して様々なテストデータに対して抽出を行った結果を示す。

図 3.15　屋外・晴れ

図 3.16　屋外・曇り

図 3.17　室内・照明

結果から RGB 表色系で行った時に比べて照明、天候、背景、服装を気にする必要がなくなり、手動でしきい値を詮索する必要がなくなった。図 3.16 では RGB 表色系での処理が苦手とする白色のシャツ、車がうまく除外できている。また、肌色検出処理の前にスムージング処理を行うことでノイズを除去することができ、より正確な肌色領域を得ることができた。

３.４　顔領域の選択実験

顔領域の切り出しは、連続した肌色領域を検出した場合そこを始点にして最後の連続領域を終点とする。得られた始点、終点を含む四角形を顔領域として抽出した。図 3.18、図 3.19 共に左図は HSV 値を用いた肌色検出で抽出した結果で、右図は左図に対して行った顔領域の選択実験をして得た領域を元の写真に適用したものである。

３.４.１　肌色ピクセルの連続に注目した選択実験

X 軸の連続した肌色ピクセルに注目し、あらかじめ決めたピクセル以上の肌色領域を顔領域の始点と終点に採用した場合、図 3.18 のような結果となった。顔領域を含む部分は選択できているが、X 座標の連続した肌色ピクセルのみに注目したため、首も顔領域として認識してしまい、顔の横幅も正確に得ることができなかった。

図 3.18　肌色ピクセルの連続に注目した実験結果

３.４.２　肌色面積に注目した選択実験

3.4.1 節の方法は肌色領域の上下が線として連続しているかを見ていたので、顔領域を正確に得ることできなかった。そこで、肌色領域が面として連続しているかという条件に変更した。結果、図 3.19 で示すように目、鼻、口を含む顔領域の選択に成功した。

図 3.19　肌色面積に注目した実験結果

図 3.20 に顔領域検出の様子をフローチャートで示す。

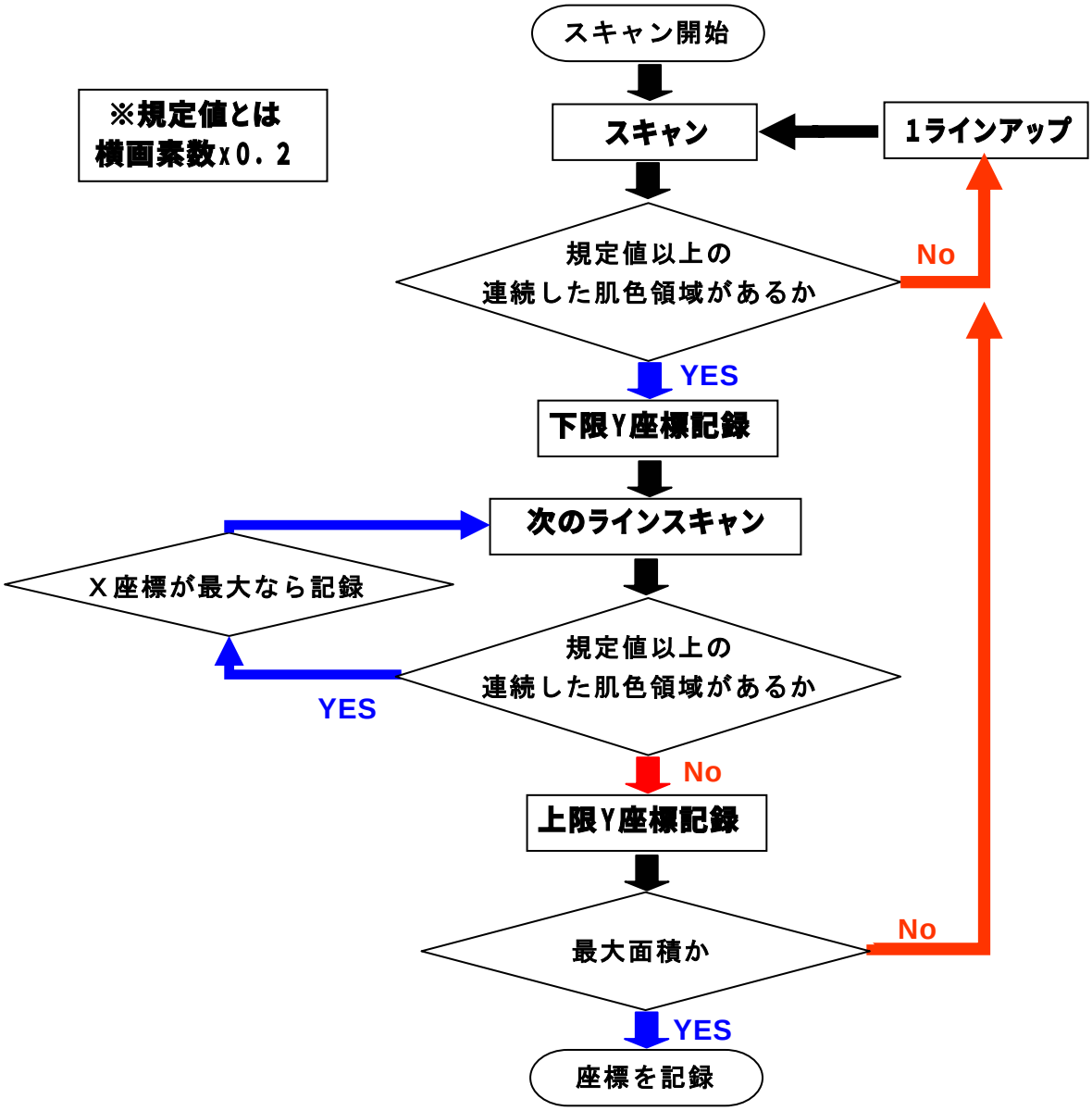


図 3.20　顔領域選択のフローチャート

図 3.21 に顔領域検出処理の様子を示す。

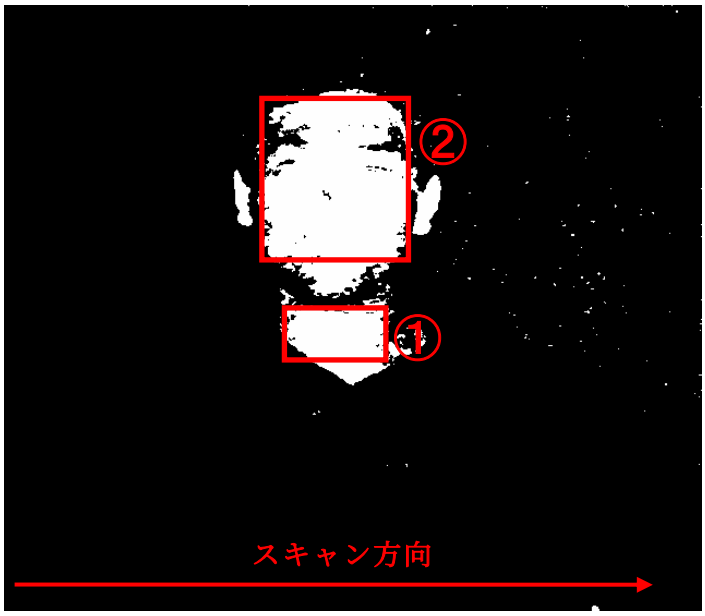


図 3.21　顔領域検出の様子

HSV 値を使って肌色検出をした画像に対して図 3.21 のように下から上に順次スキャンしていく。図 3.20 のフローチャートに従って 64 ピクセル以上の肌色領域を検出したらそこを始点として、64 ピクセル以上の肌色領域がなくなるまでスキャンする。顔領域の候補として得る座標はスキャンを始めた Y 座標、スキャンの過程で最も小さな X 座標と大きな X 座標、最後に終点の Y 座標である。得られた座標の面積を計算して大きな方を顔領域とした。図 3.21 の例では①と②の 2 つの領域が候補をして挙げられるが、面積の大きな②を顔領域として採用する。

キャプチャのタイミングによって切り出される顔領域の大きさが違うため、後の認識時の為に画素数は正規化する。

第４章　　特徴点の選択

４.１　特徴点選択の目的

特徴点はコンピュータに人物認識をさせる上でもっとも必要な要素である。2.3.1 節で述べたように実際の環境下で撮影した顔画像は環境や対象者の変化の影響を大きく受ける。従っていかにしてロバスト性を持たせるかが実際の環境下での顔検出の成功のポイントとなる。人間の認識能力と比較した場合、未知の変化や複雑なパターンに対する認識能力の点でコンピュータは著しく劣っている。コンピュータによる認識能力を向上させる方法の１つは人間の脳内での情報処理を取り入れることである。本研究では、人間がいかにして有効な特徴点の抽出選択をしているかに注目しプログラムによる認識処理の向上を目指した。

人間は対象を認識する際、対象の特徴的な部分に目を向けて情報を獲得する。そして、それらの注目点から得られた情報を統合することにより個人認識を実現していると考えられる。この情報の選択を行えることが、人間の高速かつ安定した認識の本質的な部分であると考えた。

もしコンピュータの認識処理に人間と同じように利用する情報、特徴点の選択を導入することができれば画像内の全ての特徴点を利用するよりも未知の変化や複雑なパターンに対する処理能力の向上が期待できる。特に、本研究のターゲットである人間の顔は、個人間で多少の変動はあるが大局的に見れば同じような構造を持っている。従って認識に有効な特徴点と有効でない特徴点が明確にできる可能性があり、顔の識別に有効な特徴点だけを選択をすることができれば少ない特徴点で認識能力の向上が期待できる。

４.２　特徴点の抽出方法

人間の顔に対する特徴点の抽出方法はこれまでに様々な方法が提案されている。認識に有効な場所を選択した研究として、K.Ohba and K.Ikeuchi [10] は対象のエッジに注目し、局所的なエッジパターンの中で他に類似したパターンがないもの、つまり間違え難いエッジパターンを選択した。また、L.Itti 等[11] は色エッジ輝度等のボトムアップ情報のみから自律的に注視点を選択する方法を提案した。

このような、何らかの基準により注目点を選択させる手法を参考にしてプログラムを作成した。本研究では認識時に最も正面からの標本が欲しい為、特徴点の中から目、鼻、口を選択したい。しかし複数の特徴点から目的の特徴点を選択するには使用する特徴点抽出方法や識別方法が問題となる。

特徴点抽出の基準であるが、この基準の善し悪しが認識結果に影響を与えると考えられる。認識処理で有効な特徴点を選択するにはパターン認識があるが、事前パターンの設定によっては、認識精度の信頼性が著しく低くなってしまう。

本研究では大量のデータを処理する必要がある為、プログラムではエッジとエッジの色情報を元に特徴点を抽出した後で、認識に利用する特徴点の選択を導入することにより認識能力の向上と処理速度の高速化を目指した。

４.３　エッジ検出

エッジとは物体の外縁を表す線のことであり、画像処理では画像を特徴づける線要素という。複数の物体を１つの画像に収めた場合、物体と物体の境界が濃淡の境界に大体一致する。そのため濃淡変化が大きいところは人工的な模様や物体の輪郭、ライトアップによる明暗の境界などが考えられる。これを利用してエッジを検出するのが主なエッジ検出処理である。

エッジ検出は画像処理のなかでも基本的な操作の１つで、特定の物体を取り出したり、複雑な画像を認識したりと、画像理解の為の手がかりとして使われている。エッジは信号において急激な変化がある場合に領域を分割する要素として表現される。

４.３.１　１次微分によるエッジ検出

エッジは信号が変化をする部分であるから、信号の変化分を取り出す微分を用いるのが一般的な方法である[8]。

２次元画像の微分によるエッジ検出は以下のようになる。

・各方向の微分量算出

$fx = s(i+1,j) - s(i,j)$

$fy = s(i,j+1) - s(i,j)$

・画像のエッジの強さ

$$y(i,j) = \sqrt{fx \times fx + fy \times fy}$$

又は

$y(i,j) = |fx| + |fy|$

・画像のエッジの方向

$\theta = \tan^{-1}(fy/fx)$

実際に１次元の信号で確認をすると図 4.1 のようになる。

原信号　　微分

図 4.1　微分方によるエッジ検出

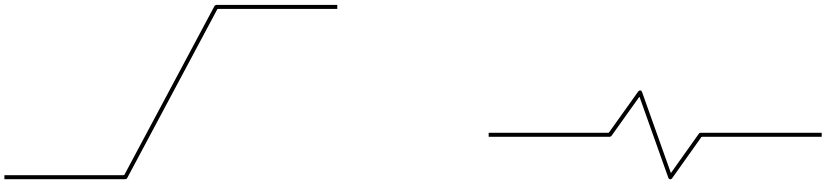
このように微分によって変化分が分かり、これがエッジとして検出される。

４.３.２　２次微分によるエッジ検出

エッジの強さだけを検出する方法として２次微分を用いるラプラシアン(Laplacian)がある。これは次式のように示される。

$$L(i,j) = x(i,j)\times 4 - \{ x(i-1,j)+x(i+1,j)+x(i,j-1)+x(i,j+1) \}$$

１次元の信号では図 4.2 のようになる。



原信号　　２次微分

図 4.2　２次微分によるエッジ検出

画像に適用する場合は、処理点と近傍領域によるオペレータを使用してフィルタ処理を行う。

エッジ検出は雑音に非常に弱く、雑音成分までエッジとして検出してしまう。その為、雑音除去フィルタを前処理としてかける方法があるが、細部信号等が劣化する可能性がある。先行研究[12]に、雑音除去をしながらエッジ検出を行う非線形フィルタ等がある。

４.４　エッジ検出フィルタ

エッジ抽出フィルタは異なった色を持つ領域の間に存在する境界を探し出す。つまり、画像の中にあるオブジェクトの輪郭を取り出すことができるということである。

輪郭、ラプラシアン、ソーベルフィルタの場合には背景は黒か透明であり、輪郭は白か原画像と同じ色となる。このフィルタを適用した結果、画像に映っている物体の輪郭（境界）や光と影の境界、人工的な模様の輪郭などを検出できる。エッジ抽出フィルタは特徴点の選択を簡単にする為に利用する。

画像のマスクフィルタリングとは、画像上の任意のピクセルの新しい輝度値をその画素の近傍画素の輝度値を用いた演算により決定する方法である。この時、各ピクセルに対して行う演算は全く独立であり、各画素が次の自分の輝度を決めるに際に用いる近傍の大きさは、画像全体と比較して十分に小さいものであることが期待される。

図4.3で､プログラムに使用した3×3マスクを用いたフィルタリングの詳細を説明する。

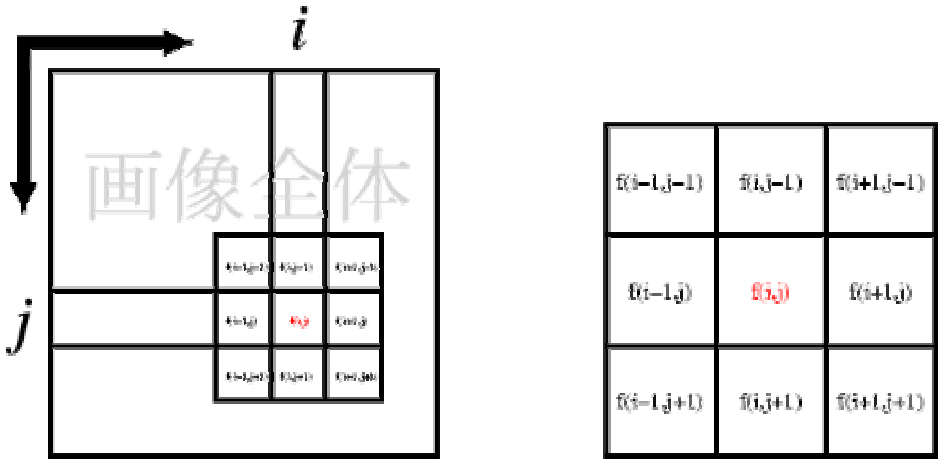


図 4.3　画像上の座標値(i,j)を中心とする 3×3 マスク

図 4.3 において、3×3 格子の中心にある画素が、今注目している画素である。その画素の輝度を f(i,j)と表す。ただし、ここで注目画素は画像全体の中で、左から i 個目、上から j 個目のところに位置しているものと仮定する。つまり画像の左上画素を(0,0)という座標値で表したとき、この注目画素の座標値は(i,j)である。

3×3 マスクを用いたフィルタリングにおいては、画像処理後の注目画素(i,j)の新しい輝度 g(i,j)は、画像処理前の画像における注目画素を取り巻く 3×3 マスク上の画素の輝度情報、(i-1,j-1)、f(i,j-1)、f(i+1,j-1)、f(i-1,j)、f(i,j)、f(i+1,j)、f(i-1,j+1)、f(i,j+1)、f(i+1,j+1)の 9 個の値を用いて決定される。

４.４.１　２値化画像の細線化処理

2 値化画像の細線化処理とは抽出したカラー画像をグレースケールに変換した後に 2 値化し、白色または黒色の濃度に変換する処理である[8]。その理由は、情報量減少により処理負荷を軽減し得る事、幾何学的諸概念の適用が容易であることである。２値化を行うためには白色、黒色の境となるしきい値は全画面について一定値を採用する。

２値化画像を線図形化することで特徴点抽出が行いやすくなる。この方法は文字認識やパターン認識の前処理として多く使用されている。また、イメージの白い領域を細線化するか、黒い領域を細線化によっても結果が変化する。

細線化の定義
（１）中心線の線幅が１である。
（２）細線化した線が中心線の元の画像の中心である。
（３）途中で切断されたり、孔が生じたりしない（連結性の保存）。
（４）不必要な「ひげ」が生じない。
（５）中心線が必要以上に縮まない。
（６）交差部において中心線がひずまない。

アルゴリズム

細線化の形態として 4 連結を採用する。また、4 連結を調べるために 3×3 のマスクを用いる。画像は左上から走査し、注目点が黒であるとき、その 8 近傍を調べ、注目点が白になるか調べる。

図 4.4 の場合、注目点は端点であるので、消去できる。これら以外の場合は連結点になるので、注目点を消去すると図形が切断される。

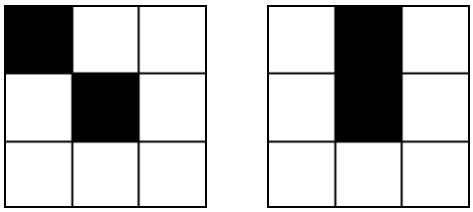


図 4.4　注目点の周囲に黒画素が 1 つの場合

図 4.5 の場合、注目点はひげであるので消去できる。これら以外の場合は連結点になってしまうので、注目点を消去すると図形が切断される。

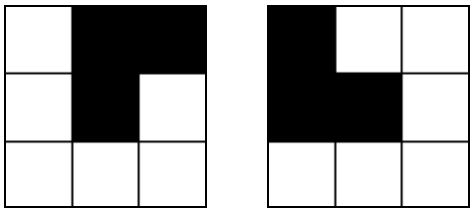


図 4.5　注目点の周囲に黒画素が 2 つの場合

図 4.6 の場合も、注目点を消去することができる。

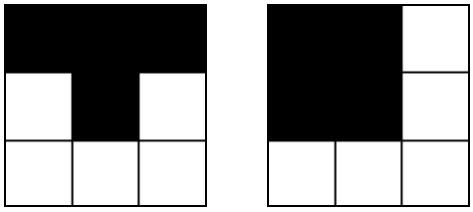


図 4.6　注目点の周囲に黒画素が 3 つの場合

図 4.4～図 4.6 のような操作を繰り返し行うことで最線化をすることができる。プログラムではステップ数を決めて処理回数を制限している。

４.４.２　１次微分フィルタ

このフィルタは１次微分フィルタの基本的なもので、注目したピクセルと回りの画素２画素間の差分を求める輪郭検出である。しかし、雑音があると雑音も一様に抽出してしまう欠点がある。

0	0	0
0	-1	-1
0	1	1

水平方向 f_x

0	0	0
0	1	-1
0	1	-1

垂直方向 f_y

注目の画素数は $f = \sqrt{(f_x{}^2 + f_y{}^2)}$

図 4.7　１次微分

４.４.３　ソーベルフィルタ

ソーベルフィルタとは縦と横の勾配よりエッジを検出する処理である。エッジをより強調する手法として知られており、中央画素の周囲の画素濃度に係数を乗じて加えたものを中央画素の濃度とする。なおソーベルフィルタは平滑化の操作を含んでいるので、一般の微分と異なり、雑音に対して強いという特徴がある。

ソーベルフィルタはラプラシアンフィルタと同じ効果を得ることができる。ただし、エッジは多少広めで、また多少ぼやけたものになる。ソーベルフィルタを水平方向だけに適用する場合、得られる画像はより高いコントラストと多くの色を持つことになる。ソーベルフィルタを垂直方向にだけ適用する場合には、多少暗めでコントラストの低い画像を得ることができる。

-1	0	1
-2	0	2
-1	0	1

水平方向 f_x

-1	-2	-1
0	0	0
1	2	1

垂直方向 f_y

注目の画素数は $f = \sqrt{(f_x{}^2 + f_y{}^2)}$

図 4.8　ソーベルフィルタ

４.４.４　プレウィットフィルタ

プレウィットフィルタは 4.4.3 節のソーベルフィルタと同様の考え方で、異なる係数を適用する。

-1	0	1
-1	0	1
-1	0	1

水平方向 f_x

-1	-1	-1
0	0	0
1	1	1

垂直方向 f_y

注目の画素数は $f = \sqrt{(f_x{}^2 + f_y{}^2)}$

図 4.9　プレウィットフィルタ

４.４.５　ロバーツフィルタ

ロバーツフィルタとは近接ピクセルを使用した輪郭検出で、濃度の変化の勾配を与える処理である。特に斜め方向の勾配を与えるので、斜めのエッジの検出に有効である。ロバーツフィルタは処理結果が一般に低輝度となる。

0	0	0
0	1	0
0	0	-1

水平方向 f_x

0	0	0
0	0	1
0	-1	0

垂直方向 f_y

注目の画素数は $f = \sqrt{(f_x{}^2 + f_y{}^2)}$

図 4.10　ロバーツフィルタ

４.４.６　ラプラシアンフィルタ

ラプラシアンフィルタは、エッジの方向に依存しない強調が可能である。ラプシアンフィルタは非常に細い(1 ピクセルの幅の)色付きの輪郭線をもった黒い画像を生成する。

ラプラシアンは一般にノイズに弱く、エッジ強調時にノイズも強調してしまう為、最良の結果を得るには、フィルタをかける前の画像にガウシアンぼかしフィルタをかけると良好な結果が得られる。

図 4.11 のフィルタは X 軸方向に対して適用する。

0	-1	0
-1	4	-1
0	-1	0

強さ 1

-1	-1	-1
-1	8	-1
-1	-1	-1

強さ 2

1	-2	1
-2	4	-2
1	-2	1

強さ 3

図 4.11　ラプラシアンフィルタ

４.５　各方法の評価

ビデオストリームに対して直接操作を行ったのでは細かい違いを検証することができない。そのため、ビデオストリームから取り出したフレームを静止画として保存した後で、顔領域を抜き出し、各方法を適用し評価する。

４.５.１　評価方法

まず、ビデオストリームより１フレーム抽出(図 4.12)し、ノイズ除去を行った後で。肌色フィルタを適用する(図 4.13)。

図 4.12　評価画像

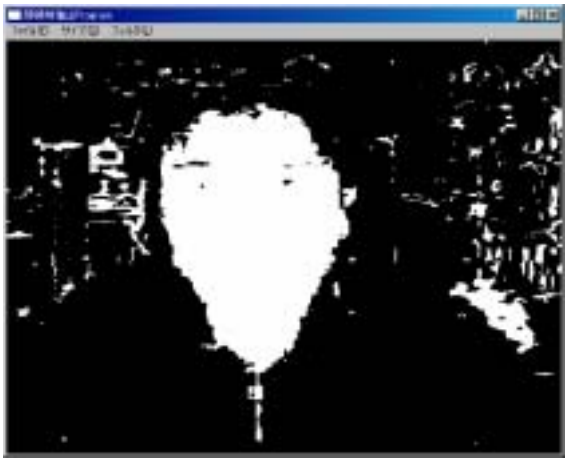


図 4.13　肌色フィルタ適応

3.4.2 節の方法で顔領域を抽出し(図 4.14)、顔部分のみをファイルに保存する(図 4.15)。

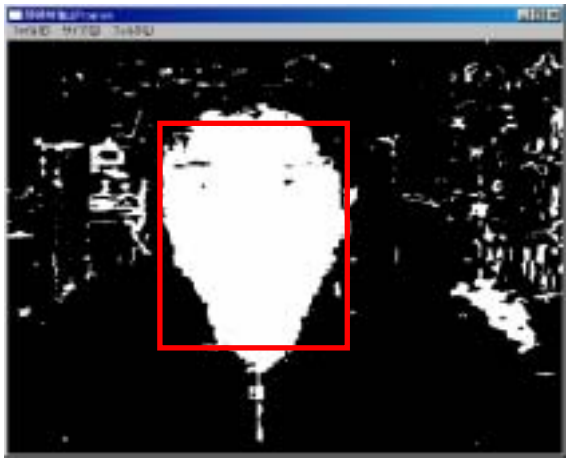


図 4.14　顔領域選択

図 4.15　顔部分

この処理によって得られた図 4.15 に各フィルタを適用した後、エッジの強調を行い特徴点抽出処理を行った結果を評価する。

図 4.16 に顔領域X軸方向の特徴点検出の様子をフローチャートで示す。

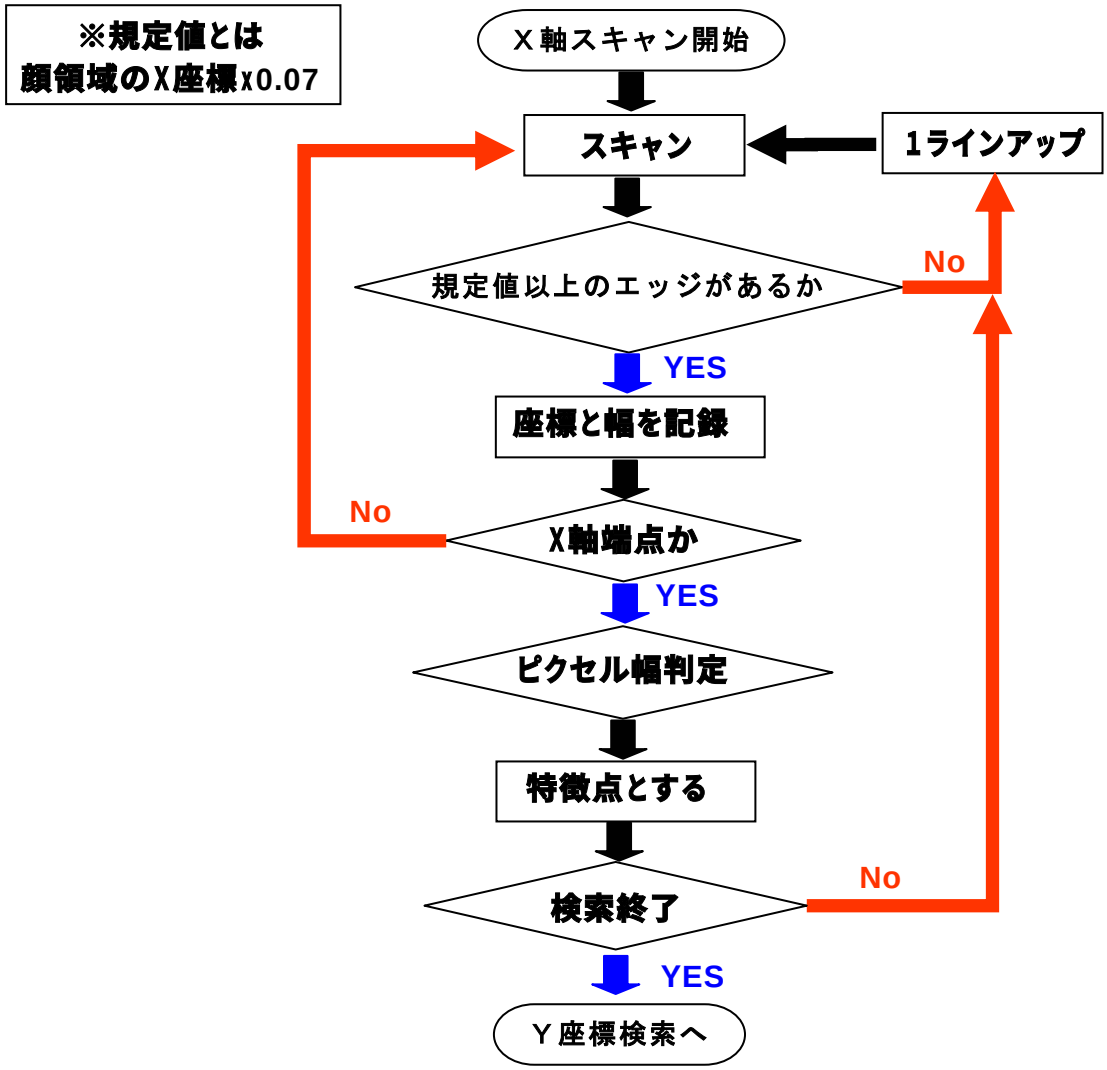


図 4.16　顔領域X軸方向の特徴点検出のフローチャート

X 軸は目、鼻、口の特徴点が線として強調される。実験を繰り返した結果、経験値として規定値は顔領域 X 座標の 7%とした。

図 4.17 に顔領域Ｙ軸方向の特徴点検出の様子をフローチャートで示す。

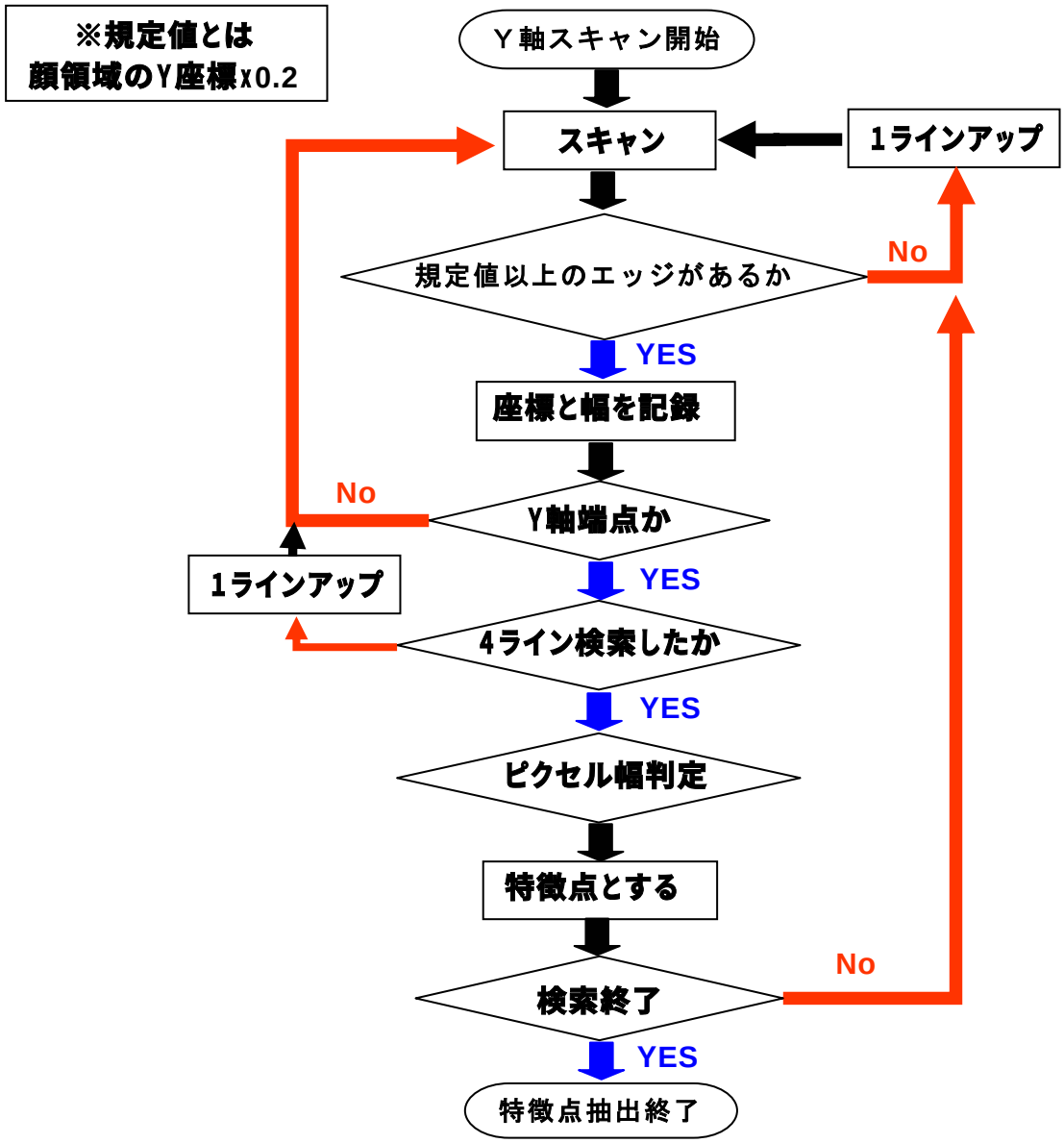
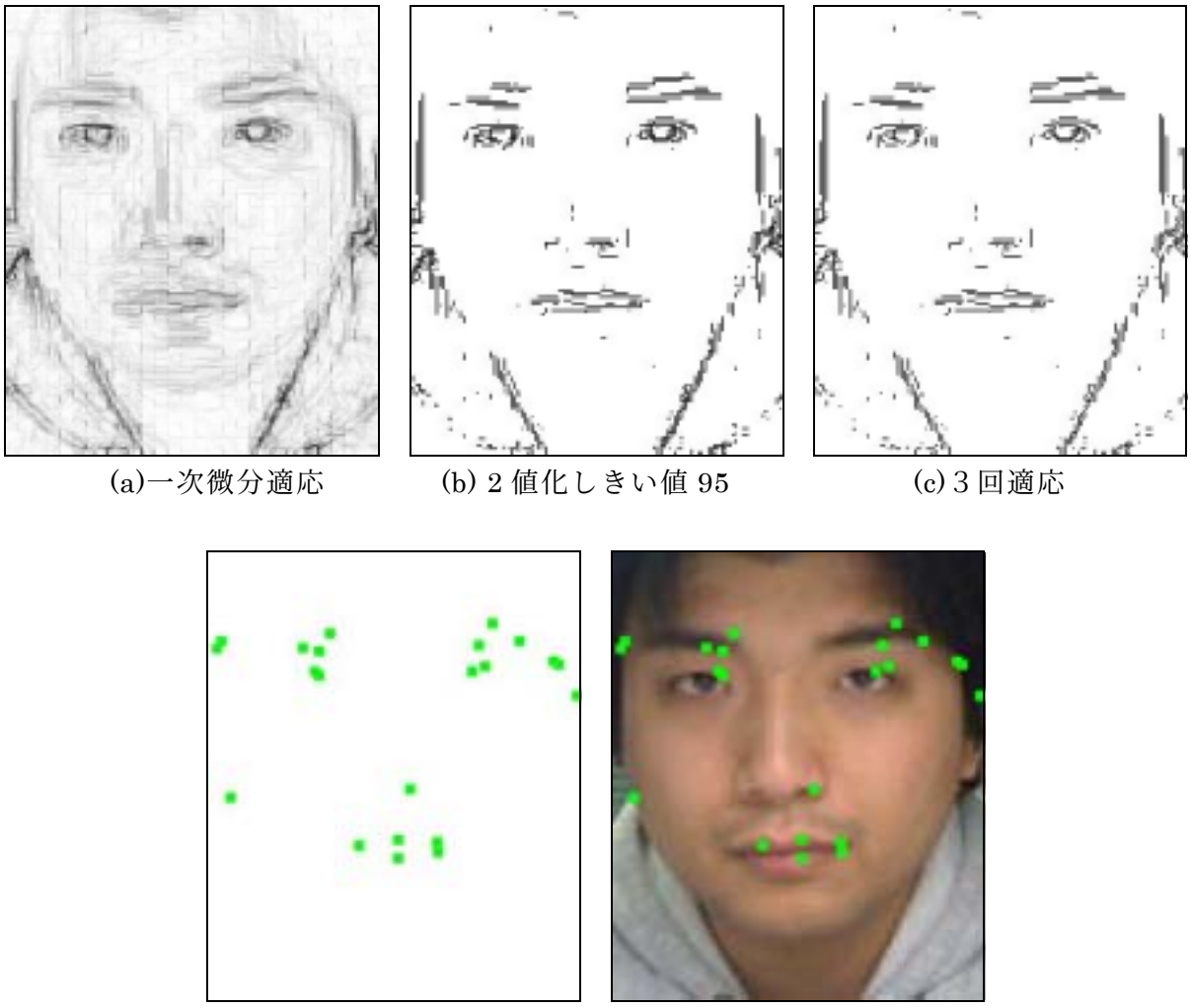


図 4.17　顔領域Ｙ軸方向の特徴点検出のフローチャート

Ｙ軸は輪郭線を示す場合が多く、細線化の際も複数出現する為、4 ラインで１つの特徴とし、規定値もＸ軸の場合より大きく、顔領域Ｙ座標の 20%とした。

４.５.２　一次微分の適応結果



(a)一次微分適応　(b) 2 値化しきい値 95　(c) 3 回適応

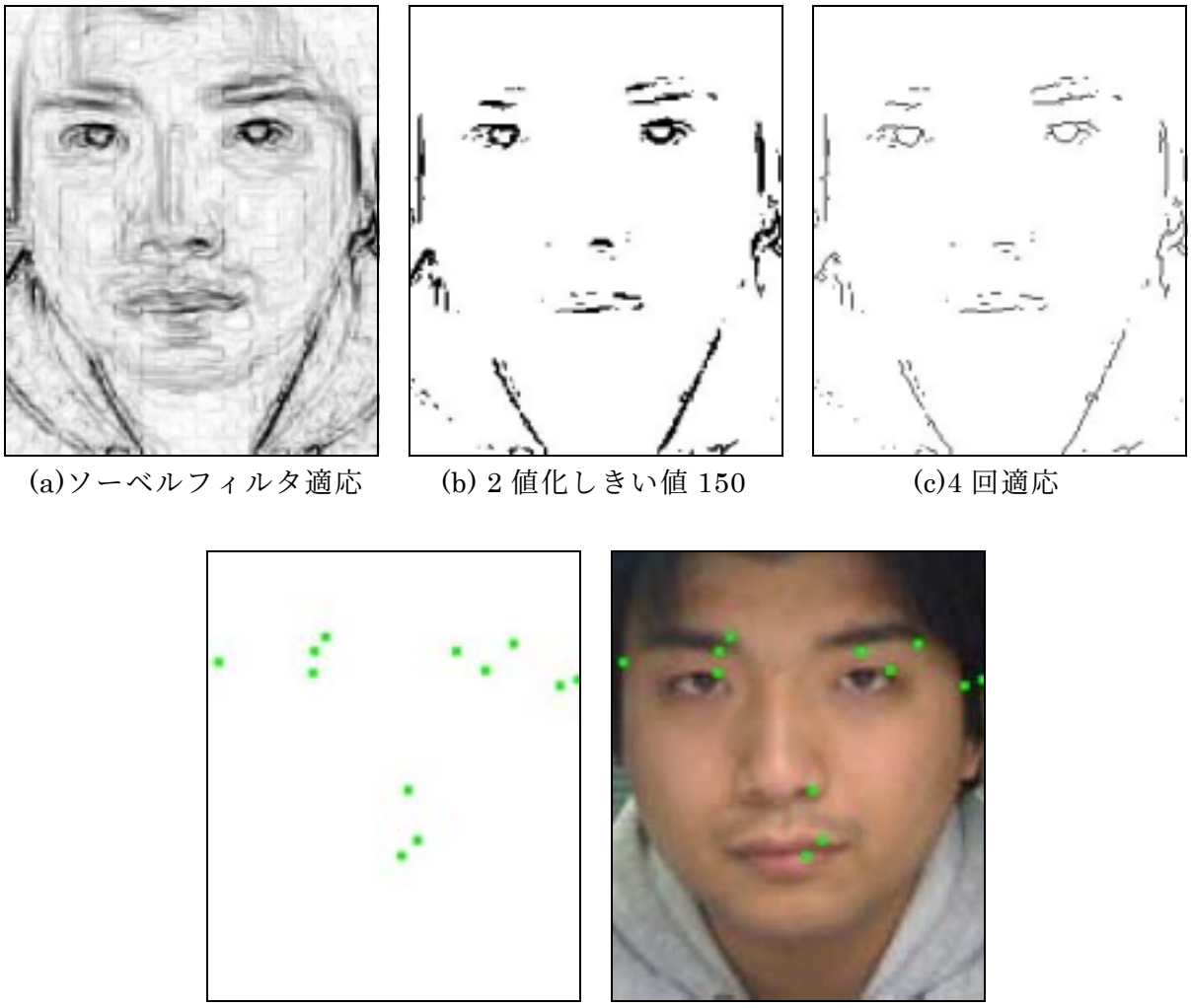
(d)特徴点　(e)特徴点の適応

図 4.18　一次微分を使った特徴点抽出

１次微分フィルタ(4.4.2 節)を適応した結果である。プログラムではフィルタに係数 2 を掛けて使用した。図 4.18(a)の適応結果をみると、エッジの強さが弱く、雑音を拾いすぎている。これは、しきい値 95 で 2 値化した図 4.18(b)を見れば分かる。実験の結果、有効なしきい値は 65～120 の範囲であったので 95 を使用した。フィルタは３回以上かけても効果がないことが分かった。

特徴点を抽出する際のフィルタの回数を１～３回それぞれにおいて試した結果、ほとんど差が見られなかった。よってプログラムにはフィルタは１回で採用した。この結果、図 4.18(e)に示すように、ノイズの影響を大きく受けておりノイズも特徴点として拾っている。その為、目、鼻、口以外にも多くのポイントをとってしまうので特徴点抽出用のフィルタには向かない。

４.５.３　ソーベルフィルタの適応結果



(a)ソーベルフィルタ適応　(b) 2 値化しきい値 150　(c)4 回適応

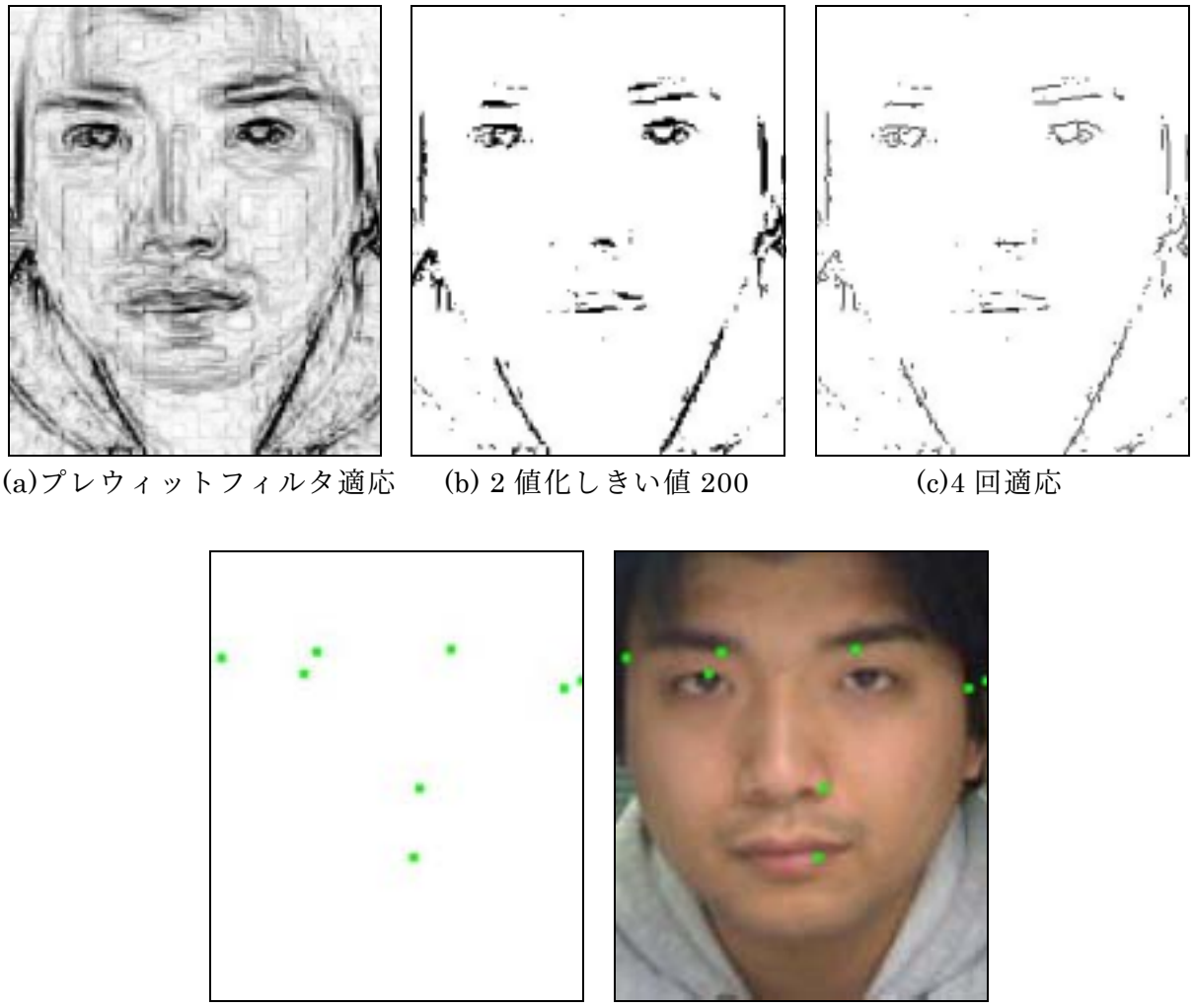
(d)特徴点　(e)特徴点の適応

図 4.19　ソーベルフィルタを使った特徴点抽出

ソーベルフィルタ(4.4.3 節)を適応した結果である。図 4.19(a)の適応結果をみると、エッジははっきりと強調されており、雑音がうまく平滑化されている。これは、しきい値 100 で 2 値化した図 4.19(b)をみれば分かる。実験の結果、有効なしきい値は 100～200 の範囲であったので 150 を使用した。フィルタは 4 回以上かけても効果がないということが分かった。

特徴点を抽出する際のフィルタの回数を 1～4 回それぞれにおいて試した結果、フィルタの回数を増やした場合、輪郭が細線化されすぎてうまく特徴点が取れず、フィルタ 2 回が最も結果がよかった。よってプログラムにはフィルタは 2 回で採用した。この結果、図 4.19(e)に示すように目、鼻、口を含むポイントが取れ、余分な特徴点も少なかった。よって、ソーベルフィルタは特徴点抽出用のフィルタに向いている。

４.５.４　プレウィットフィルタの適応結果



(a)プレウィットフィルタ適応　(b) 2 値化しきい値 200　(c)4 回適応

(d)特徴点　(e)特徴点の適応

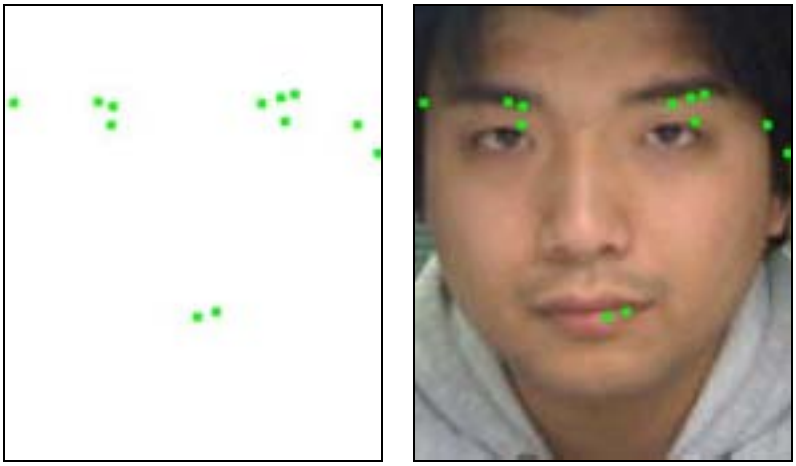
図 4.20　プレウィットフィルタを使った特徴点抽出

プレウィットフィルタ(4.4.4 節)を適応した結果である。プレウィットフィルタはソーベルフィルタ(4.4.3 節)と同様の処理で係数だけが違うものなので似たような結果が得られた。図 4.20(a)の適応結果をみると、ソーベルフィルタより更にエッジが強調されているが、雑音も多めに強調されている。この為、有効なしきい値は高めで 165～240 の範囲であった。しきい値 200 で 2 値化した結果の図 4.20(b)を見ると、ソーベルフィルタよりノイズが多い。フィルタの回数はソーベルフィルタと同様に 4 回以上増やしても効果がないことが分かった。

特徴点を抽出する際のフィルタの回数を 1～4 回それぞれにおいて試した結果、フィルタ回数 2 回が最も結果がよかった。フィルタをかける回数を増やした場合でも、雑音は小さくなるが残ってしまった。よってプログラムにはフィルタは 2 回で採用した。この結果、図 4.20(e)に示すように目、鼻、口を含むポイントが取れる良好な結果が得られたので、ソーベルフィルタと同じく特徴点抽出用のフィルタに向いている。

４.５.５　ロバーツフィルタの適応結果

(a)ロバーツフィルタ適応　(b) 2 値化しきい値 160　(c) 3 回適応



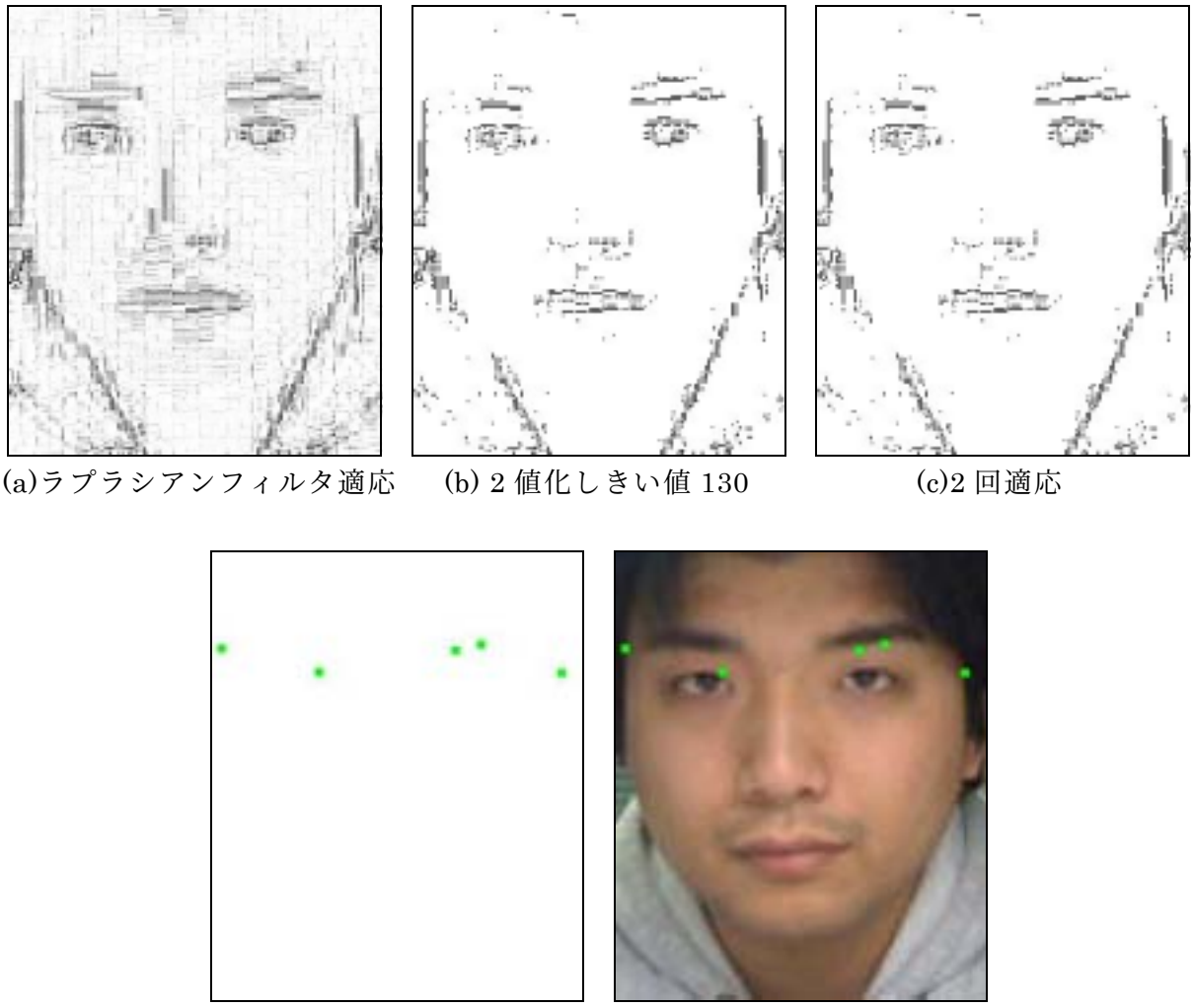
(d)特徴点　(e)特徴点の適応

図 4.21　ロバーツフィルタを使った特徴点抽出

ロバーツフィルタ(4.4.5 節)を適応した結果である。図 4.21(a)の適応結果をみると、エッジが細かい横線として強調されている為、鼻や、口と特徴が弱い。実験の結果、有効なしきい値は 110～210 の範囲であったので 160 を使用した。しきい値 160 で 2 値化した図 4.21(b)をみると、輪郭以外は横線として強調されたことが分かる。フィルタは 3 回以上かけても効果がないことが分かった。

特徴点を抽出する際のフィルタの回数を 1～3 回それぞれにおいて試した結果、いずれの場合も大差なかったので処理速度向上の為、プログラムにはフィルタ 1 回で採用した。エッジの強調では 2 値化の時点で鼻のエッジが細かすぎた為、図 4.21(e)に示すように全く特徴として取れていない。よって、顔の特徴抽出には向かない。

４.５.６　ラプラシアンフィルタの適応結果



(a)ラプラシアンフィルタ適応　(b) 2 値化しきい値 130　(c)2 回適応

(d)特徴点　(e)特徴点の適応

図 4.22　ラプラシアンフィルタを使った特徴点抽出

ラプラシアンフィルタ(4.4.6 節)を強さ 2 で適応した結果である。図 4.22(a)の適応結果をみると、エッジが非常に細く、点、線として強調されている為、全体の特徴がやや弱い。実験の結果、有効なしきい値は 110～170 の範囲であったので 140 を使用した。フィルタは 2 回以上かけても効果がないことが分かった。

特徴点を抽出する際のフィルタの回数を 1～2 回それぞれにおいて試した結果、変化がみられなかったのでプログラムにはフィルタは 1 回で採用した。エッジの強調では細線化の時点で特徴が点、線ノイズとしてなくなっているので、図 4.22(d)に示すようにほとんどの特徴が取れていない。よって、顔の特徴抽出には向かない。

４.６　特徴点の抽出実験

4.5.2 節～4.5.6 節で各種フィルタの評価を行った結果、ソーベルフィルタとプレウィットフィルタが特徴点の抽出において良好な結果を示した。両者を比べた結果、ソーベルフィルタの方がより雑音への耐性に優れていたので、作成したプログラムのエッジ検出の部分にはソーベルフィルタを採用した。

3.2.1 節、3.3.5 節、3.4.2 節、4.5.2 節～4.5.6 節の実験で得られた結果を利用して作成したプログラムの実行結果を図 4.23 に示す。フレームレートは 320×240 で 5fps、320×240 で 3fps、640×480 で 1.5fps であった。320×240 までが実用的であるといえる。

(a)長髪・照明(明るい)　　(b)眼鏡・照明(普通)

(c)照明(暗い)

図 4.23　特徴点抽出実験

図 4.23 に示した実験から分かるように、対象者、眼鏡の有無、髪の長さ、照明の違い、服の色を変えてテストした結果、いずれの場合も目、鼻、口を含む特徴点をリアルタイムに抽出することができた。よって、この抽出した特徴点の中から目、鼻、口を代表とする認識に必要な特徴点を明確に選択することができればリアルタイム顔認識に有効であるといえる。

第５章　　結論

５.１　まとめ

本論文は、生体情報を用いた個人認証の手段として顔認証に注目し、認証システム、セキュリティーシステムに広く応用できると期待されているリアルタイム顔認識の為の特徴抽出手法について、認識システムの基盤を作成、特徴点の抽出手法を提言して、その有効性を確認したものである。

実験では、WDM ビデオキャプチャデバイスから得られたビデオストリームから動き検出に成功し、HSV 表色系を利用した肌色検出の有効性について実証することができた。また、3.3.5 節の肌色検出の結果から 3.4.2 節で提案した手法を用いて顔領域を抽出できることも確認した。

特徴点の抽出においては、ぼかしフィルタ、エッジ強調フィルタ、2 値化処理、最線化処理を組み合わせ、4.5.1 節で提案した特徴検出手法を使って特徴点が抽出できることを 4.5.2 節～4.5.6 節の結果より確認した。

リアルタイム特徴抽出プログラムを作成するにあたっては、多数のサンプルを使用し、全てのフィルタ、処理に関して係数の変更、マスク幅の変更、ステップ数の変更をしながら実験を行い、得られた結果をもとにプログラムを作成した。4.6 節での実験結果で示すように、作成したリアルタイム特徴抽出プログラムは、多様性のある顔、眼鏡の有無、髪の長さ、照明の違い、服の色の違いの中からリアルタイムに目、鼻、口を含む特徴点を得ることができた。

現在、様々な研究機関からリアルタイム顔認証システムが発表されているが、本研究でリアルタイム顔認証の基礎部分を作成したことで、高度な認証システムを開発する為の第一歩が踏み出せたと思う。

プログラムの仕様上、対象者が複数人の場合には一人分しか画像と特徴点を得ることはできないが、現時点でも無人の場所に設置し、防犯に使用することは十分に可能であるといえる。

５.２　今後の課題

今後の課題としては、第一に､特徴点から人物認識のパラメータとして利用できる目・鼻・口、その他のパラメータを選択できるようにすることである。目・鼻・口の検出は、本研究で採取できた各特徴点周辺の画素において、色情報の利用やパターンマッチングの利用に加えて新たなアルゴリズムを考案することにより検出できるようになるのではないかと思われる。目・鼻・口が選択できれば本研究の次のステップである正面検出に進むことができ、これまで多数開発されている顔認証システムを使用することができる。

本研究では実験するまでに至らなかったが、目・鼻・口が抽出できたなら顔の輪郭座標と目・鼻・口の位置情報を用いて正面検出、斜め画像から正面画像への復元を行うことが可能となる。このような処理を行うことで、多人数を認識にする場面において認識漏れを少なくすることができる。

作成したリアルタイム特徴抽出プログラムは、WDM ドライバを使用するキャプチャデ

バイスであれば全て使用することができるというメリットがある。しかし、デメリットとして各キャプチャデバイスの性能を考慮しなければならない。使用するカメラによってはノイズを多く拾う物もある。その場合はメディアンフィルタのように、エッジ保存、インパルス性ノイズの除去を同時に成し遂げられる、非線形で順序統計量に基づくフィルタ処理をする必要がある。メディアンフィルタでは各画素をソートする必要がある為、リアルタイムに行う処理への採用は、計算量が増え欠点となる。その為、高速に処理できエッジを保存しつつノイズを除去できるフィルタを考案する必要がある。

また、作成したシステムでは対象が同時に複数人いる場合でも、一人分しか採取することができない為、得られた顔領域候補全てに対して顔検出が行えるように、顔の検出アルゴリズムを改良することが必要である。

謝辞

本研究にあたり、最後まで熱心な御指導をいただきました田中章司郎教授には、心より御礼申し上げます。

また、田中研究室の辰己圭介君、高木明君、学部生のみなさん、岡本研究室の田中文崇君には､本研究に関して数々の御協力と御助言をいただきました｡厚く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等の全ての知的財産権を、本研究の指導教官である田中教授に譲渡致します。

参考文献

[1] 松下満次,他：アイリス個人認識システム，沖電気研究開発 第 175 号, Vol.64, No.3, pp.107-110，1997

[2] 福井和広, 山口修:顔画像を用いた個人認証技術, 東芝レビューVol.56 No.7, pp.14-17, 2001

[3] ChrisCant：WDM デバイスドライバ，翔泳社，2000

[4] Microsoft Corporation, MSDN Library

[5] Microsoft Corporation, DirectX 8.1 SDK Document

[6] 星正明：DirectX8 実践プログラミング，工学社，2001

[7] Intel Corporation, Open Source Computer Vision Library Reference Manual

[8] 中嶋正之, 他：技術編 CG 標準テキストブック, 財団法人画像情報教育振興協会, 1999

[9] Jamie Sherrah and Shaogang Gong, “Skin Colour Analysis”, University of Edinburgh, May, 2001.

[10] K.Ohba and K.Ikeuchi,“Detectability, uniqueness and veliability of eigen-windows for stable verification of partially occluded objects,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.19, No.9, pp. 1043-1047, Sep., 1997.

[11] L.Itti, C.Koch, and ENiebur, “A Model of saliency-based visual attention for rapid scene analysis,” IEEE Trans. Pattern Analysis and Machine Intelligence,Vol.20, No.11, pp. 1254–1259, Nov., 1998.

[12] 橋本有平，梶川嘉延，野村康雄，"画像信号の方向性を考慮したインパルス性ノイズ除去手法，"電子情報通信学会論文誌 A, Vol. J84-A, No. 6, pp. 759-768, Jun., 2001.