動画像理解技術とその応用

藤吉 弘亘

hf@cs.chubu.ac.jp

中部大学工学部情報工学科

http://www.vision.cs.chubu.ac.jp/VU/

2009, 3, 7

平成 21 年 3 月 7 日

目 次

第1章　画像理解技術を用いた次世代ビデオ監視システム:VSAM

近年，犯罪発生率の急増に伴い，ビデオ監視システムに関する研究への期待が高まっている [1]. 従来の重要施設の入退出管理を目的としたビデオ監視システムは，監視カメラ映像を記録するものや，監視員が複数のカメラ映像を同時にモニタリングするものが多い．監視する範囲が広く 24 時間監視となると監視員への負担が大きくなり，問題とされている．これに対し，米国では 1997 年より 2000 年の 3 年間，国防総省の研究開発部門である高度研究計画局 (DARPA:Defence Advanced Research Projects Agency) の下，画像理解技術を用いたビデオ監視システムの研究プロジェクト VSAM (Video Surveillance and Monitoring) が行われた [2]．VSAM プロジェクトには，カーネギーメロン大学 (CMU)，MIT をはじめとする 10 大学と，Sarnoff Corporation，Texas Instruments(TI) 等の 2 企業が参加した．CMU では，キャンパスに 12 台のカメラを配置し VSAM テストシステムを構築した [3]．VSAM システムは，動画像理解技術により検出した侵入物体を複数のカメラが協調してトラッキングし，その状況をリアルタイムで監視員に提示する．これにより，監視員の負担軽減と作業効率化に大きく貢献でき，新しいビデオ監視システムとして期待されている．

本章では，画像理解技術を用いた自動ビデオ監視システムとして，VSAM プロジェクトの概要，動画像理解技術，抽出結果の表示方法，及び複数カメラによる協調動作について述べる．

1.1　VSAM システムの構成とその特徴

1.1.1　分散協調による状況理解

1 台のカメラセンサで監視できる範囲は限られているので，実時間で広域監視を行うためには，複数のカメラセンサを効果的に配置し，ユーザ（監視員）からのタスクをネットワークに接続された複数のカメラセンサが協調して動作する必要がある．VSAM プロジェクトでは，このようなシステムの実現を目指している．

図 1.1 に VSAM システムの概要を示す．個々のセンサは，屋外カメラからの映像より実時間で物体の検出・追跡・識別を行い，自動検出した物体情報を監視センタに送信する．各カメラセンサからの情報はサイトモデルと呼ばれる共通の 3D データ空間で統合され，マップ上に表示される．ユーザは，人や自動車がどこを移動しているか等の広域の動的状況を実時間でモニタすることが可能となる．ユーザがシステムにタスクを与えると，複数のカメラセンサは協調して動作するため，1 台のカメラでは許容できない広範囲における侵入物体のトラッキングを可能とし，その行動軌跡を知ることができる．さらに，コンピュータグラフィックス (CG) を用いて合成映像を生成することにより，監視領域で実際に起った動的シーンを仮想的に再現することが可能となる．

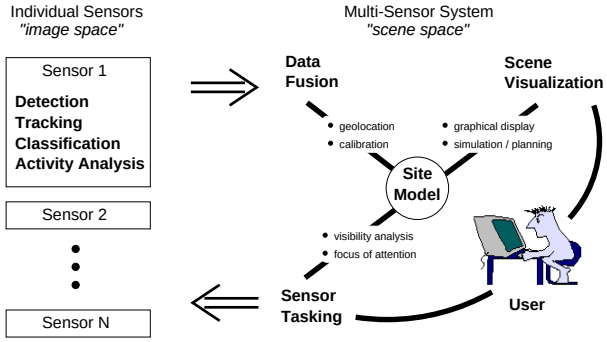


図 1.1: システムの概要

1.1.2　3D サイトモデル

VSAM システムの特徴の一つは，サイトモデル (3D 地形データ) の使用である．サイトモデルを用いることで，単眼カメラでも対象物体の三次元位置の推定が可能となる．米国における地形データは，USGS (U.S.Geological Survey) から DEM (Digital Elevation Map) データを購入することができるが，その解像度は 1 ピクセル $= 30m^2$ と低い．そこで，VSAM プロジェクトでは CMU のキャンパスと周辺の地形データに建物や道路等の情報を加えたより高解像度の DEM データを作成した (図 1.2 参照)．これをサイトモデルと呼ぶ．また，GPS を用いて測定した複数のランドマークより各カメラの位置を計算した [4]．図 1.2(c) は，実際のカメラビューをサイトモデルと計算したカメラ位置から合成したものである．(b) の実画像と (c) の CG 画像は，似ているものではないが，カメラ画像内に背景として何が写っているか，例えば道路や建物の位置を正確に知ることができることを表している．

(a) 3D site model of CMU campus　(b) real image　(c) synthsized image

図 1.2: サイトモデル

1.1.3 VSAM システムの構成

CMU キャンパスの約 $0.4km^2$ の領域を監視対象とし，12 台のカメラを建物の屋上や壁面に死角領域が少なくなるように配置した (図 1.3(a) 参照)．各カメラは Pan/Tilt/Zoom 機能を持つ．テストシステムは，図 1.3(d) に示すように，SPUs (Sensor Processing Units), OCU (Operator Contorol Unit), GUI (map-based Graphical User Interface), VIS (Visualization nodes) から構成される．

SPU は，pan/tilt/zoom 機能を持つアクティブカメラと動画像処理を行う PC で構成されたセンサである．SPU はカメラ映像から自動的に物体を検出・識別し，その結果を記号データとしてネットワークを介して OCU に送信する．OCU は全ての SPU からの情報を受け取りデータベースに登録し，GUI のマップ上にその結果を表示する．

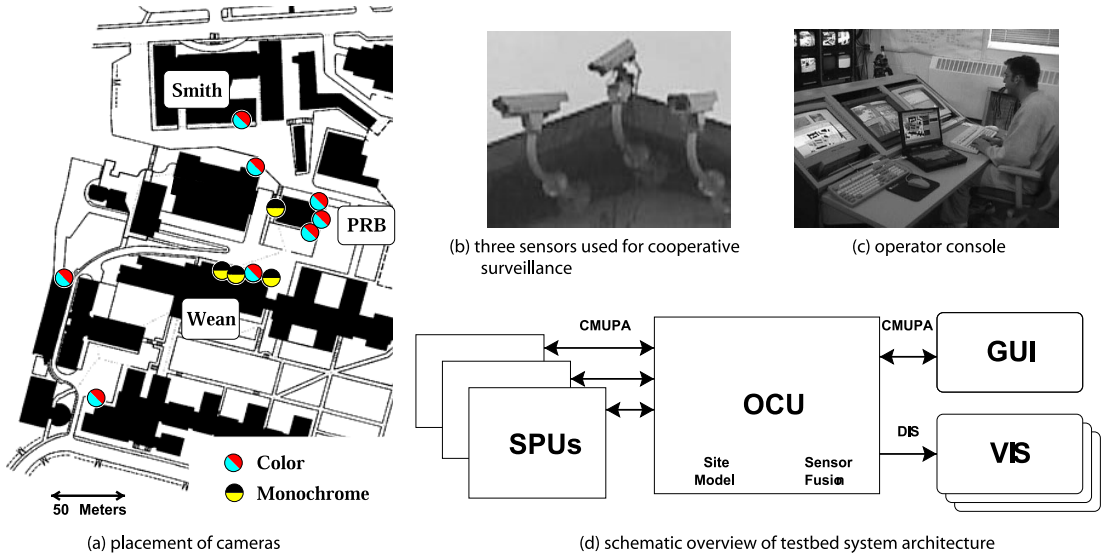


図 1.3: VSAM システムの構成

1.2 動画像理解技術

動画像理解技術として，物体検出，識別，アクティビティ認識のアルゴリズムと評価実験結果について述べる．

1.2.1 物体検出・追跡

侵入物体の検出には，検出すべき物体が存在しない背景画像を予め用意しておき，入力画像と背景画像の差分を計算する背景差分処理が多く用いられている [5]．人と自動車のアクティビティを認識するには，画像上の複数物体の重なりを検出する必要があるが，背景差分処理と領域クラスタリングを組み合わせた手法では重なった複数物体を一つの領域として検出するという問題がある．

そこで VSAM では，複数物体の重なりを理解するレイヤー型検出法を提案した [6]．レイヤー型検出法[1]は，ピクセル分析とリージョン分析の二つの処理からなる．ピクセル分析では，各ピクセルの輝度値の時間変化を観測し，その変化軌跡によりピクセルの状態を静もしくは動と判定する．ある時間幅の変化軌跡を用いることで，太陽光等の環境変化の影響を受けにくくなる．リージョン分析では，動とラベル化されたピクセル領域を移動物体と判定する．静とラベル化された領域は静止物体と判定し，背景上のレイヤーとして記憶する．一度停止した物体は，再び動き出すまでレイヤーとして登録されているため，レイヤー上を通過する移動物体を区別して検出することが可能となる．

図 1.4 に，停止した 2 台の自動車とその手前を移動する物体 (人) の検出例を示す．比較的交通量が多い屋外駐車場の 2 箇所を視野とするカメラで撮影した約 8 時間のビデオ映像を用いて，本検出法の評価実験を行った結果，背景差分法は約 83%，レイヤー型検出法は 92%の検出率を得た．検出後，各物体はカルマンフィルタを拡張したトラッキングにより固有の ID が付けられる [3]．

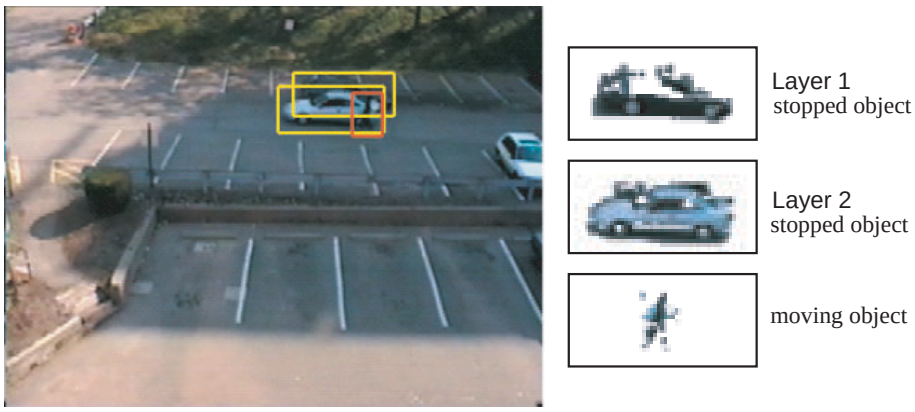


図 1.4: 物体検出例

1.2.2　物体識別

検出した物体は，以下の要因によりその「見え」は逐次変化し不安定であるため，人と自動車の識別は容易でない．

- 屋外環境を対象物体が移動
- 天候による照明変化
- カメラの位置

VSAM システムでは，12 台のカメラを見えの大きく異なるカメラ毎にグループ化し，それぞれの識別器を作成することで対処した．識別にはニューラルネットワークを用いた．また，物体の詳細情報として対象の色，車種 (形) を識別するために判別空間による手法を用いた．

[1] レイヤー型検出については，3.3 にてアルゴリズム等の詳細を述べる．

ニューラルネットワークを用いた識別

見えの大きく変わるカメラ毎に約1000 枚の学習サンプルを作成した．検出した画像から計算した形状の複雑度 (周囲長^2/面積)，面積，縦横比，カメラのズーム倍率をニューラルネットワークへの入力パラメータとした．出力は，人 (一人), 人のグループ (二人以上), 自動車, その他の 4 クラスとした．ニューラルネットワーク[2]は 3 層の階層型で，学習にはバックプロパゲーション法を用いた．物体検出は，入力映像中の対象に ID をつけてトラッキングする．そこで，トラッキングシーケンスを通じて各フレーム毎に出力されたニューラルネットワークの結果から各クラスのヒストグラムを作成し，最大となったクラスを最終識別結果とした．小規模なニューラルネットワーク (入力層 4, 中間層 16, 出力層 4) で構成しているため，リアルタイムでの処理を実現している．

種別 (形状) の識別

学習用サンプル画像約 2000 枚に対し，オペレータが目視で種別 { セダン，バン，トラック，小型搬送車，人，その他 } のラベルをつけた．サンプル画像が予め定めた特定対象（実験では FedEx 社の集配車，郵便車，パトカーとした）である場合には，その旨のラベルも加えた．学習用の各画像からその長方領域の幅と高さ (2 個)，濃淡画像部分の幅と高さ方向の 1,2,3 次モーメントの総計 ($2 \times 3 = 6$ 個)，濃淡画像部分の重心 (幅，高さ方向に 2 個)，濃淡画像部分の面積 (1 個) の 11 次元の特徴ベクトルを算出し，種別識別用の判別空間を構成する．種別の識別は，識別対象の画像の特徴ベクトルを種別識別用の判別空間に射影し，k 近傍法で判定する [7].

対象物の色の推定

まず様々な条件下で撮影した色サンプル (晴天時の映像から約 1500，曇天時の映像から約 1000) にオペレータが自らの印象に基づいて 6 種類の色ラベルをつけ，それをシステムにそのまま学習させた．個々の色サンプル画像から 3 次元の色 [8] の特徴ベクトルを算出し，これをクラスとして線形判別空間を構成した．色の推定は，推定対象画像の色ベクトルをこの判別空間に射影し，k 近傍法で判定する．

図 1.5 は稼働中の本システムの処理画面例である．天候と太陽の位置の双方を考慮し，晴天／曇天の朝から日没の間の映像により評価を行った結果，種別と色の平均識別率は約 90% であった．

1.2.3 アクティビティ認識

監視領域で起った事象を知るためには，人と自動車のインタラクションを認識することが必要となる．このような行動認識は，テロ防止対策や駐車場管理等への応用が考えられる．

[2]ニューラルネットワークによる識別については，7.3 にて学習方法等の詳細を述べる．

図 1.5: 物体識別例

人の動き分析

人の動きを認識するには，人体の各部位を追跡する手法が多く提案されている [9]．我々は，リアルタイムで検出した人の画像領域から，スター型スケルトンを抽出する手法を提案した [10]．スター型スケルトンは，検出領域の重心と輪郭線上の自動的に抽出された特徴点で構成される (図 1.6 参照)．スケルトン形状から頭部と足部の特徴点を決定し，姿勢の前傾度合いと足部の時間的変化を周波数分析することにより，人間が歩いているか走っているかを識別することが可能となる．

人と自動車のアクティビティ認識

移動物体間の相互作用 (インタラクション) を含む行動 (アクティビティ) を推定する．物体検出・識別によって得られた情報を基に，各検出領域が内部状態として物体の種別，アクション (Appearing, Moving, Stopped, Disappearing), インタラクション (Near, MovingAwayFrom, MovingTowards, NoInteraction) の組を持つことを仮定し，それら内部状態の組同士の確率的関係に基づいてアクティビティを推定する．詳細については，文献 [11] を参照いただきたい．これにより，以下の 6 種類のアクティビティを認識することが可能となる．また，これらの結果の表示方法は 1.4.3 に述べる．

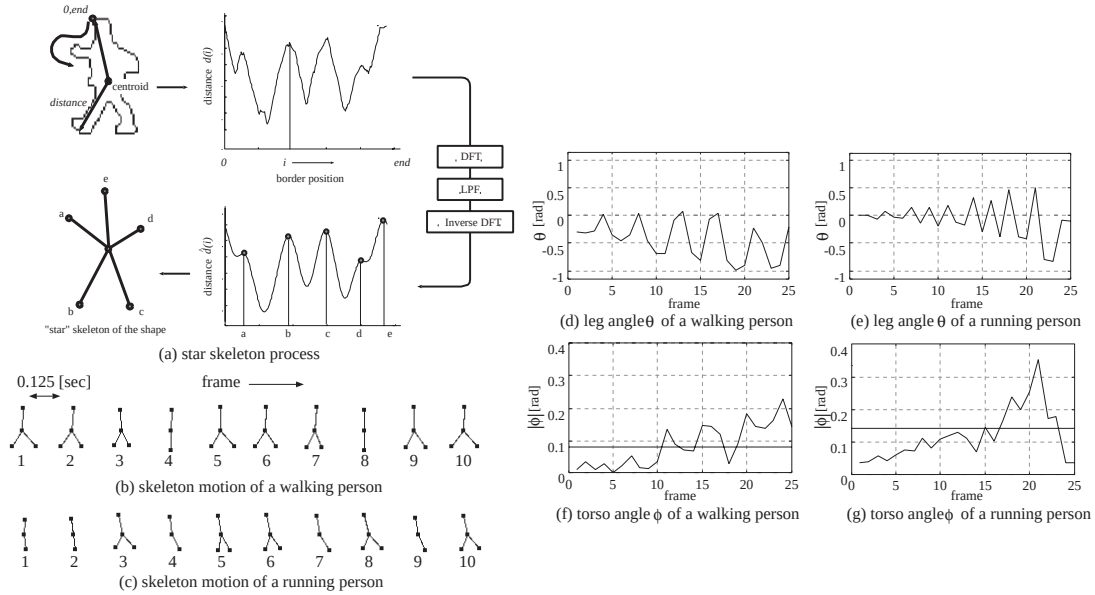


図 1.6: スター型スケルトンによる動きの分析

human entered a vehicle.	(人が車に乗車した)
human exited a vehicle.	(人が車から降車した)
human entered a building.	(人が建物に入った)
human exited a building.	(人が建物からでてきた)
a vehicle parked.	(車を駐車した)
human rendevouz.	(2 人以上の人が集合している)

このような日常の人や車のアクティビティを認識することは，テロ等に繋がる非日常的なイベントを抽出するキーとなる．また，認識したアクティビティの対象物に類似した物体をデータベースから検索することで，監視領域内の別の場所や異なる時間帯に観測された同一対象物体の行動を知ることが可能となる．

1.3 対象物体の 3 次元位置の推定

検出された対象物体の 3 次元位置の推定は，通常複数のカメラを用いたステレオ手法が多く用いられるが，監視領域が広く物体数が多い場合，常に一つの物体を 2 台以上のカメラセンサでトラッキングできるとは限らない．VSAM では，監視領域の地形データであるサイトモデルを用意しているため，対象物体 (人や自動車) が地面上に接しているという仮定により，画像上の検出直方領域の下辺部の中心点を通過する 3 次元空間上の光線が地形データ (DEM) の地面と交差する点を対象物の 3 次元位置と推定できる[3](図 1.7 参照)．

[3]単眼カメラによる 3 次元位置推定については，5.2 で述べる．

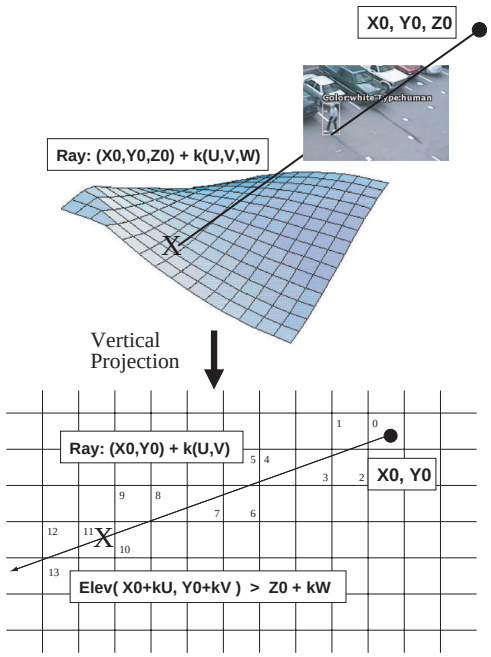


図 1.7: 位置の推定

Leica 社製のレーザトラッカーセオドライトを用いて，本手法の位置推定精度の評価を行った．評価実験は，駐車場を二周したときのセオドライトによる測量座標とカメラセンサにより光線交差法を用いて自動推定した座標間の距離を測定した．実験の結果，カメラから対象までの距離が約 65m のとき，平均誤差が 0.6m と良好な精度を得た．

1.4　動的シーンの表示

監視システムでは，リアルタイムに情報をユーザに伝える必要がある．また，ユーザが一度に複数のモニタを監視することは難しいため，複数の情報源からの情報を一括して提示する必要がある．ここでは，動画像理解技術により得られた情報 (識別クラス, 3D 位置座標) をユーザに提示する手法として 2 次元平面地図を用いたマップ型 GUI, CG による 3D 動的シーン表示，WWW を用いたアクティビティの要約表示について述べる．

1.4.1　マップ型 GUI

SPU から集められた対象物体の情報を 2 次元平面地図の上にリアルタイムで表示するマップ型 GUI を開発した (図 1.8 参照)．マップ上には，現在の全ての対象物の位置に物体を示す記号 (人は丸，自動車は四角) とカメラの状態として位置と FOV (Field of view) が表示される．ユーザがマウスによりマップ上のカメラをクリックすると，そのライブカメラ映像をモニタすることができ

る．マップ型 GUI は広域に配置された複数のカメラからの情報を一括して表示するため，広域監視領域の状況が容易に監視できる．

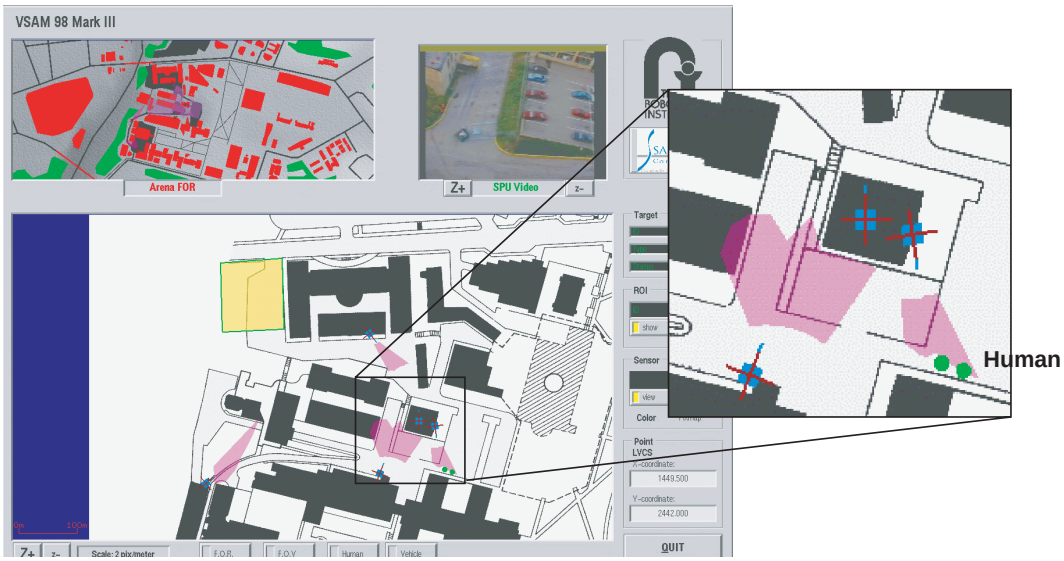


図 1.8: マップ型 GUI

1.4.2 CG による動的シーンの 3D 表示

監視結果は全てデータベースに登録されているため，それらのデータとサイトモデルにより CG を用いて動的な合成映像を自動生成しユーザに提示した．CG を用いることで実際に起った動的シーンを，任意の視点からの映像として再現することができる．交通事故等を様々な角度から再現することで事故の検証に有効であると考えられる．その他の応用例としては，サッカー等のスポーツにおけるゲームシーンの解析と再現が挙げられる．

1.4.3 WWW によるアクティビティの要約表示

人と自動車のアクティビティと識別結果は，全てデータベースに登録されている．ユーザは，WWW を介してサーバ上のデータベースを参照することで，監視場所で起こったアクティビティの要約を確認することができる．WEB サーバは，ユーザ (WEB ブラウザ) の要求に対して CGI(Common Gateway Interface) によりデータベースからアクティビティの要約結果を自動生成し表示する [12].

WWW によるアクティビティの要約結果の表示例を図 1.10 に示す．(a) は，アクティビティレポートの例である．アクティビティレポートは，“A Human got out of a Vehicle”, “A Vehicle parked” 等のイベント結果を時刻順に表示する．もし，ユーザがアクティビティに関与した物体の詳細を知りたいとき，物体の識別タイプや色等の情報とその画像をハイパーテキストリンクにより表示することができる．

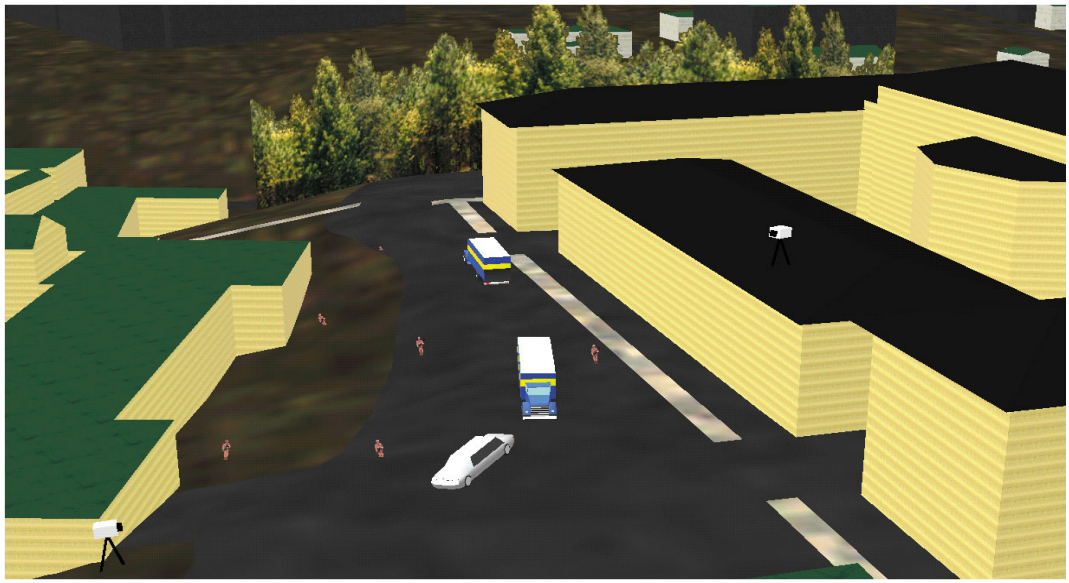


図 1.9: CG による動的シーンの再現

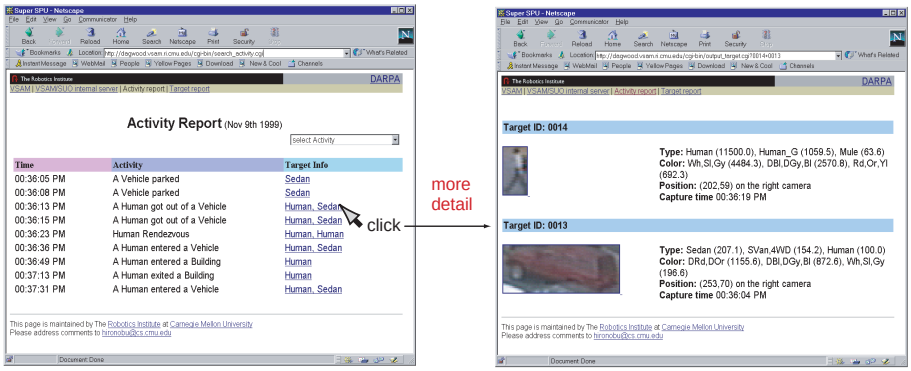
(b) はオブジェクトリポートの例である．オブジェクトリポートは，システムによって検出された全ての物体の一覧を表示する．観測された物体数が多いとき，ユーザは全ての物体を確認することが不可能であるため，識別タイプ毎に表示することができる．また，ユーザがある特定の物体をマウスクリックにより指定すると，システムはデータベースに登録された他の物体との類似度を識別タイプと色情報から計算し，類似度の高い順に表示する．これにより，監視領域内の別の場所や異なる時間帯に観測された同一対象物体の長時間の行動を推定することができる．

1.5　複数のカメラによる協調監視

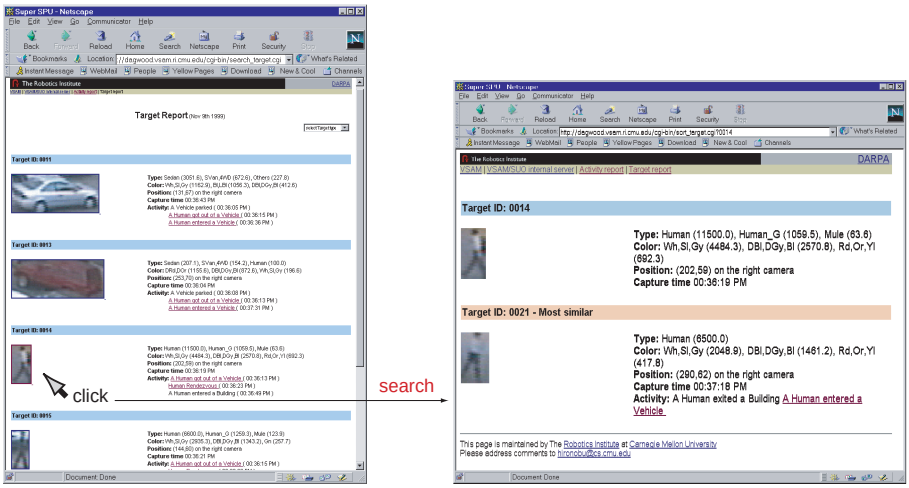
複数のカメラを用いることで，対象監視領域の拡大，同一対象の長時間に渡る追跡，侵入物体の多方向画像の取得が可能となる．本節では，複数カメラの協調動作におけるカメラ構成とその処理技術について述べる．

1.5.1　マスター・スレーブによる自動追尾

固定カメラ (マスター) と旋回型カメラ (スレーブ) を組み合わせた自動追尾システムでは，まず監視領域内に侵入した物体をマスターカメラより 2. で述べた背景差分等の移動体検出法用いて検出する．次に，検出した画像座標に対応する実空間の位置を推定する．推定した位置情報を基に複数のスレーブカメラのパン・チルト・ズームを制御して，多方向から対象物の映像を取得する．2 次元カメラ画像座標から実世界の位置推定には，1.3 で述べたように，カメラキャリブレーション [13] により得られたカメラの内部・外部パラメータから，図 1.11 に示すようなカメラ光学中心と



(a) Activity report



(b) Object report

図 1.10: WWW による表示と検索例

カメラ画像上の座標を通る世界座標空間における直線を求め，その直線が床面に交差する点を対象物の位置として計算する [14]．スレーブカメラの台数を増やすことで，物体の詳細映像を多方向から同時に取得することができるため，防犯上重要な情報を見逃しにくい撮影システムとして有効である．

1.5.2 ハンドオフによる協調追尾

カメラの設置場所により木や建物等によるオクルージョン領域が発生し，1 台のカメラセンサのみで特定の侵入物体を長時間に渡り追跡することは困難である．この問題を解決するには，ネットワーク化された複数のアクティブカメラの協調動作が必要となる．入場ゲートで捉えた侵入物体をカメラセンサがパン・チルトを制御して追跡し，他のカメラ視野に移動した際，そのカメラセンサ

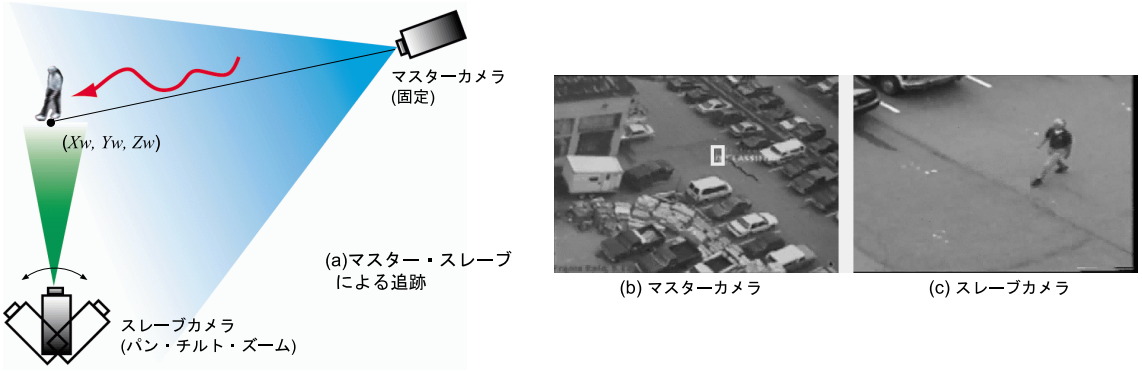


図 1.11: マスター・スレーブによる自動追尾システム

に追跡タスクの受け渡し (ハンドオフ) を行う．その際，異なる位置に設置されたカメラにおける物体の見えは異なるため，同一対象であるかの判定が重要となる．各カメラ画像の対象領域内の色ヒストグラムと 3.1 の手法で求めた位置情報を基に整合度を算出し，同一対象であるか判定する．カメラ間で同一対象と判定されれば，対象物の移動速度，大きさを考慮してパン・チルト・ズームを制御して追跡する．図 1.12 は，1 台の自動車が約 400[m] の距離を約 3 分で移動したとき 3 台のカメラによるハンドオフ動作例である．

1.6 VSAM 画像理解技術の応用

カメラの組み合わせに応じた画像理解技術とその応用として複数のカメラセンサによるビデオ監視システムについて述べた．動画像理解技術による監視カメラの自動化は欠かせない技術であり，今後は 24 時間への対応やステレオ視による三次元情報の活用，ネットワーク化した複数カメラによる監視が重要になると思われる．VSAM プロジェクトの詳細については，参考文献や Web site(http://www.cs.cmu.edu/~vsam/) を参照いただきたい．現在，本章で解説した VSAM 画像理解技術は，監視用途だけでなく EyeVision システム [15] 等のエンターテイメントや，歩行者 ITS(Intelligent Transport System)[16][17] へ応用されている．

謝辞 CMU VSAM Project: Takeo Kanade, Robert T. Collins, Alan J. Lipton, David Duggins, Raju S. Patil, Osamu Hasegawa, Nobuyoshi Enomoto, Yanghai Tsin の諸氏に深く感謝する．

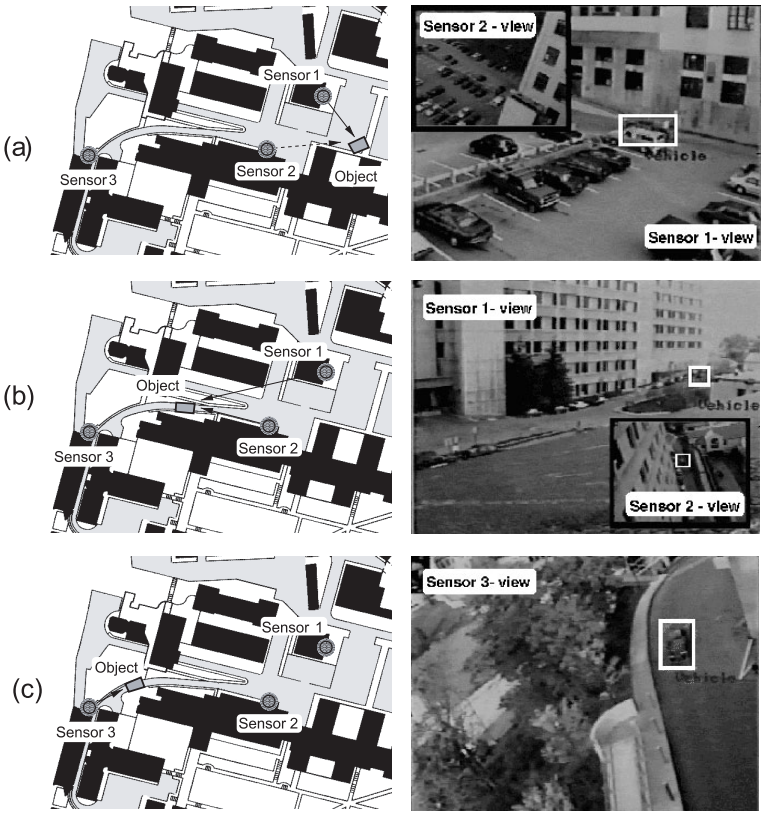


図 1.12: ハンドオフ動作例

参考文献

[1] 東京・新宿広場に防犯カメラ設置計画, 毎日新聞 2002 年 3 月 6 日 (2002).

[2] VSAM: “Section I, video surveillance and monitoring”, Proc. of the 1998 DARPA Image Understanding Workshop, Vol.1, pp. 1–400 (Nov. 1998).

[3] R. Collins, A. Lipton, H. Fujiyoshi and T. Kanade: “Algorithms for cooperative multi-sensor surveillance”, Proc. of IEEE Special Issue on ”Video Communications, Processing and Understanding for Third Generation Surveillance Systems”, (Oct. 2001).

[4] R. Collins and Y. Tsin: “Calibration of an outdoor active camera system”, Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 528–534 (Jun. 1999).

[5] A. Lipton, H. Fujiyoshi and R. Patil: “Moving target detection and classification from real-time video”, Proc. of the 1998 Workshop on Applications of Computer Vision, pp. 8–14 (Oct. 1998).

[6] 藤吉弘亘, 金出武雄: “複数物体の重なりを理解するレイヤー型検出法”, 第 7 回画像センシングシンポジウム論文集, pp. 369–374 (2001-6).

[7] 長谷川修, 金出武雄: “一般道路映像中の対象物のオンライン識別”, 第 7 回画像センシングシンポジウム論文集, pp. 221–226 (2001-6).

[8] Y. Ohta and T. Kanade: “Color information for region segmentation”, Computer Graphics and Image Processing, Vol. 13, No. 3, pp. 222–241 (1980).

[9] J.K. Aggarwal and Q. Cai: “Human motion analysis: A review”, Computer Vision and Image Understanding, Vol. 73 No. 3, pp. 428–440 (Mar. 1999).

[10] H. Fujiyoshi and A. Lipton: “Real-time human motion analysis by image skeletonization”, Proc. of the 1998 Workshop on Applications of Computer Vision, pp. 14–21 (Oct. 1998).

[11] N. Enomoto, T. Kanade, H. Fujiyoshi and O. Hasegawa: “A method for monitoring activities of multiple objects by using stochastic model”, IEICE Transactions on Information and Systems, Vol. 84-D No. 12 (Dec. 2001).

[12] 藤吉弘亘, 榎本暢芳, 長谷川修, 金出武雄: “アクティビティモニタリング–屋外監視映像の要約と WWW 上表示・検索システム–”, 第 7 回画像センシングシンポジウム論文集, pp. 423–428 (2001-6).

[13] R. Y. Tsai: “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses”, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, pp.323〜344 (1987).

[14] 小川雄三, 藤吉弘亘: “実空間に対応した Master-Slaving による追尾カメラシステム”, 第 9 回画像センシングシンポジウム, pp.211-216 (2003).

[15] http://www.eyevision.com/

[16] 矢入 (江口) 郁子，猪木誠二: “高齢者・障害者の自立移動を支援する Robotic Communication Terminals(3)”，人工知能学会論文誌，Vol.18, No.1, pp.29-35(2003)

[17] 藤吉弘亘，小村剛史，矢入 (江口) 郁子，香山健太郎，吉水宏: “歩行者 ITS のためのフレーム間差分による移動体検出法とその評価”，情報処理学会論文誌，Vol.45 No.SIG13(CVIM10), pp.11-20(2004)

第2章　移動体検出:Object Detection

コンピュータの処理能力の高速化に伴い，動画像を実時間で処理することが可能となりつつある．特に，動画像内における人等の移動体追跡の実現が期待されている．移動体の追跡は，画像内から移動体領域を検出し，各フレームにおいて同一対象がどこにいるかを求めることであり，ビデオ監視システムをはじめ，さまざまな分野に対して応用されている．本章では，移動体追跡の第一段階である動画像からの移動体検出法について述べる．従来，移動体検出手法には以下の 3 つの手法が多く用いられている．

- フレーム間差分 (Temporal Differecing)
  フレーム間差分法は，現在の入力画像と前回の画像との差分を計算し，差分値の大きい領域を移動物体として検出する．動的な環境変化に適応的であるが，一般に移動体全ての領域を抽出することは不可能である．

- 背景差分法 (Background Subtraction)
  背景差分法は，検出すべき物体が存在しない背景画像をあらかじめ用意しておき，入力画像と背景画像の差分を計算する手法である．ほぼ完全な移動体領域を検出することができるが，天候等による環境変化に対して背景画像を更新する必要がある．

- オプティカルフロー (Optical Flow)
  オプティカルフロー法は，カメラが動いている状態でも，カメラの動きと異なるフローを求めることで動物体を検出することができる．しかし，計算量が多いため特別なハードウェアを使用しない場合，リアルタイムでの演算は困難である．

本章では，まず最初に固定カメラ映像からの移動体検出として，フレーム間差分法と背景差分法について述べ，次に移動カメラからの移動体検出としてオプティカルフローについて述べる．フレーム間差分や背景差分は，異なる時刻に撮影された 2 枚の画像の差を観察することによって，変化情報を得る．2 枚の画像において同じ位置にある画素の差を差分画像という (図 2.1)．この差分画像を用いて移動体検出は行われる．

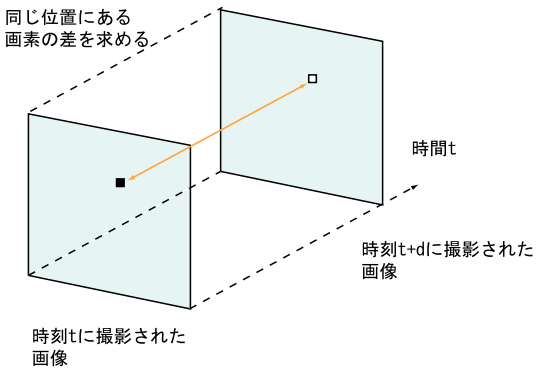


図 2.1: 差分画像

2.1　フレーム間差分 【Source Code】

フレーム間差分法は時刻 t と時刻 $t-1$ の画像の差分により移動物体を検出する手法である．フレーム間差分は次式により求められる．

$$\Delta_t = |I_t - I_{t-1}| \tag{2.1}$$

I_t は現在の入力画像，I_{t-1} は 1 フレーム前の画像とする．移動物体の移動速度が遅い場合，1 フレーム間では変化が少ないため次式に示すように現在の入力画像と n フレーム前までの画像と差分の最大値を使用する．

$$\Delta_t = \max\{|I_t - I_{t-j}|\}, \forall_j \in [1, n]\} \tag{2.2}$$

急激な輝度値の変化がピクセル上に生じたとき，変化量 Δ_t の値は大きくなる．ここで，ピクセル状態を表す M は，輝度変化を処理することにより，次式に示すように物体 (1) と背景 (0) に判定する．Th は，急激な変化を判定する閾値である．

$$M_t(u,v) = \begin{cases} 1 & ,\Delta t(u,v) > Th \\ 0 & ,\text{Otherwise} \end{cases} \tag{2.3}$$

図 2.2 に表すように，フレーム間差分は，変化のあるピクセルのみを検出し，対象とする移動体全体の領域のピクセルを検出することはできない．

2.1.1　カラー画像の差分計算

カラー画像に対して差分処理を行うには，RGB をグレースケールに変換する必要がある．RGB をグレースケールに変換する方法に RGB の平均を計算する単純平均法がある．

$$\mathrm{Y} = \frac{(\mathrm{R} + \mathrm{G} + \mathrm{B})}{3} \tag{2.4}$$

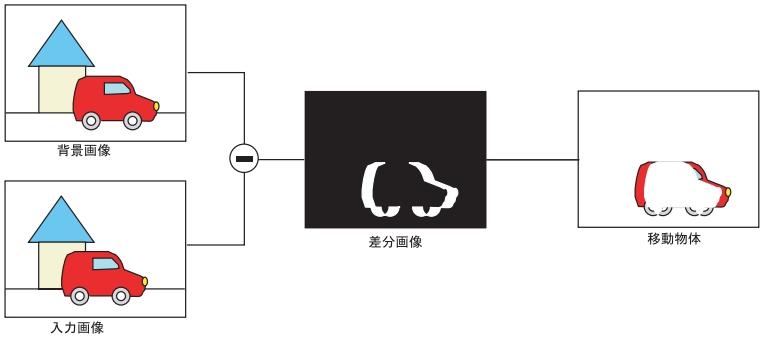


図 2.2: フレーム間差分法

しかし，単純平均法でグレースケール変換した画像は，緑が暗く，青が明るい不自然なグレー画像になる．より自然なグレースケール画像を得るためには，次式に示す NTSC 加重平均法が用いられている．

$$\mathrm{Y} = 0.298912\mathrm{R} + 0.586611\mathrm{G} + 0.114477\mathrm{B} \tag{2.5}$$

カラー画像を単純平均法と NTSC 加重平均法によりグレースケール化した結果を図 2.3 に示す．図 2.3 から，単純平均を用いて変換した場合では赤色，緑色，青色が同じ色に変換されているが，NTSC 加重平均を用いると緑色が明るく青色が暗く変換されていることがわかる．図 2.4 に NTSC 加重平均法を用いてグレースケール変換した画像に対してフレーム間差分を施した結果を示す．

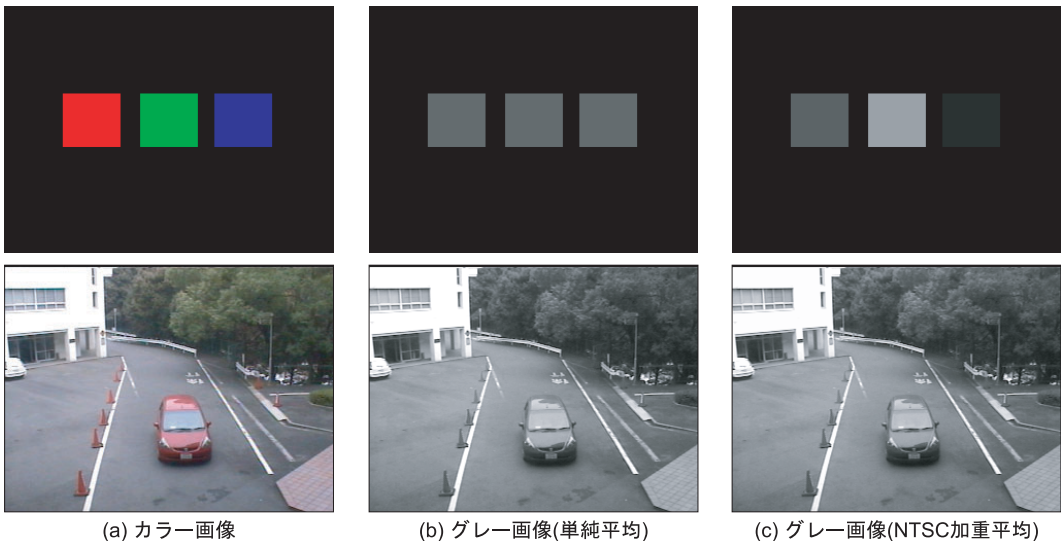


図 2.3: 単純平均法と NTSC 加重平均法の比較

図??より，車両の色が同一色で，テクスチャーがない場合，車両位置が移動しても，ボンネットなどの領域は変化が生じないため，検出されない場合がある．


(a) 時刻tの画像　(b) 時刻t-1の画像　(c) フレーム間差分結果

図 2.4: カラー画像をグレースケールに変換してフレーム間差分した結果

2.1.2　RGB ベクトル間の角度を用いた差分計算

24 ビットのカラー画像の場合，RGB のそれぞれの要素が 0-255 の値を取る．そこで RGB の値を要素とした 3 次元ベクトルを考え，これを RGB ベクトルと定義する (図 2.5) [2].

RGB ベクトル法ではベクトルの長さが輝度値を表し，ベクトルの方向が色彩を表す．時刻 t における RGB ベクトル $\overrightarrow{I_1}(R_1, G_1, B_1)$ と $\overrightarrow{I_2}(R_2, G_2, B_2)$ のなす角を θ とすると，$\overrightarrow{I_1}$ と $\overrightarrow{I_2}$ の内積から，$\cos\theta$ を次式より求めることができる．

$$\cos\theta = \frac{\overrightarrow{I_1} \cdot \overrightarrow{I_2}}{\mid \overrightarrow{I_1} \mid\mid \overrightarrow{I_2} \mid} = \frac{R_1R_2 + G_1G_2 + B_1B_2}{\sqrt{R_1^2 + G_1^2 + B_1^2}\sqrt{R_2^2 + G_2^2 + B_2^2}} \tag{2.6}$$

人為的に物体に当たる照明の強さを変化させた映像に対して，輝度差と RGB ベクトルのなす角

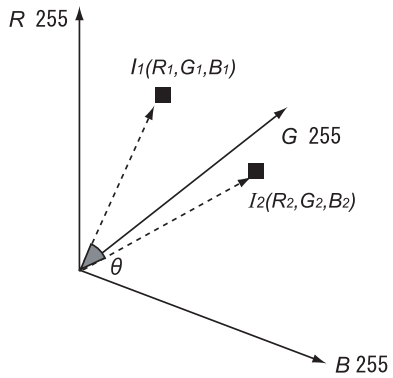


図 2.5: RGB ベクトル

を用いてフレーム間差分した結果を図 2.6 に示す．輝度差を用いた場合，照度変化により人形の全体を誤検出しているが，RGB ベクトルのなす角を用いた場合では人形の移動した部分のみを検出できている．これは，照度変化が生じても RGB ベクトルの大きさが変化するだけであり，ベクトルの方向は変化しないためである．

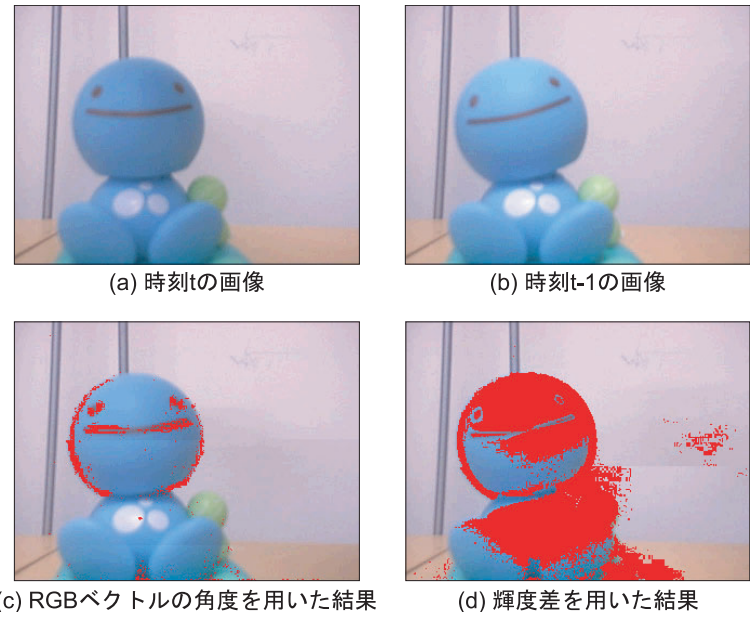


図 2.6: 輝度差と RGB ベクトルを用いたフレーム間差分結果

2.1.3 しきい値の決定

ピクセルが物体もしくは背景であるかを判定するためのしきい値には，固定しきい値と変動しきい値がある．固定しきい値法は，その名の通り 2 値化を行うしきい値 Th に固定値を用いる方法である (図 2.7)．しきい値を $Th = 5$，$Th = 15$，$Th = 30$ に設定したときのフレーム間差分結果を図 2.8 に示す．図 2.8 から，しきい値を 30 にすると移動物体により生じる輝度変化に対してしきい値が高く設定されるため，移動体領域を十分に検出できない．しかし，しきい値を 5 まで下げると微小な輝度変化を検出してしまうため，背景領域で生じる輝度変化を検出してしまう．そのため，図 2.8 の場合ではしきい値を 15 程度に設定すると適度に移動体領域が検出されることがわかる．

2.1.4 変動しきい値の設定

しきい値 Th には背景の変化に適応し，かつ画像内の各領域毎にしきい値を設定する必要がある．各フレーム毎にしきい値を設定するには，各ピクセル毎に過去数フレーム間の輝度変化の分散をしきい値に反映させることが考えられる [1]．過去 K フレーム間の輝度値の分散値 $S_t^2(u,v)$ は次式より計算する．

$$S_t^2(u,v) = \frac{K\sum_{i=1}^{K} I_{t-i}^2(u,v) - (\sum_{i=1}^{K} I_{t-i}(u,v))^2}{K(K-1)} \tag{2.7}$$

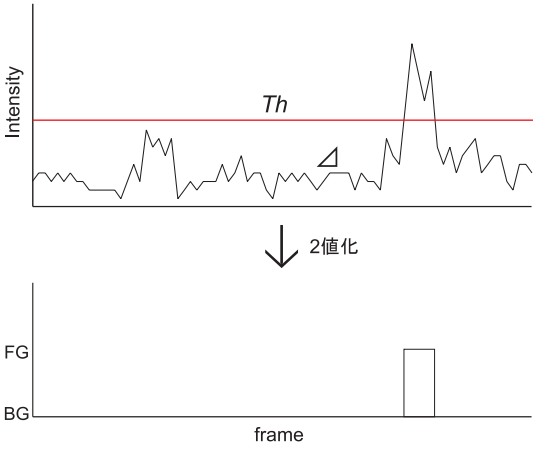


図 2.7: 固定しきい値法

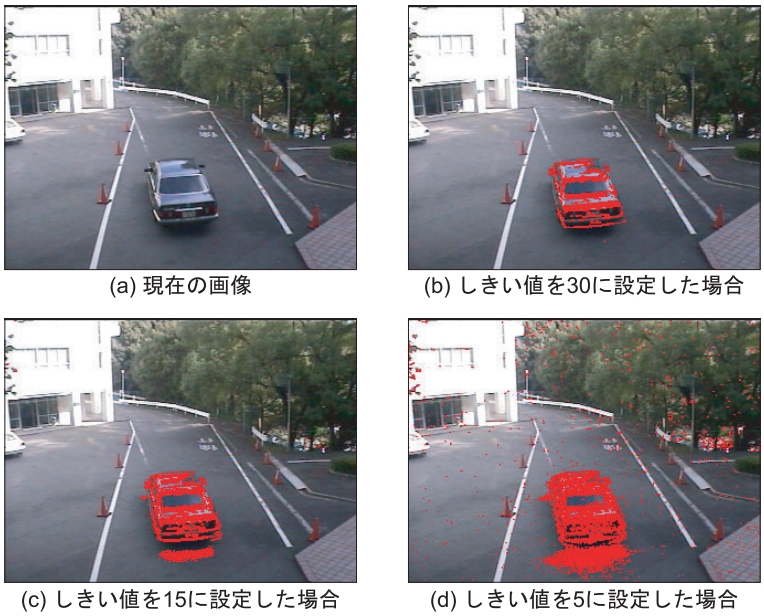


図 2.8: しきい値変更にともなう画像変化

これはフレーム $t-1$ から $t-K$ までの分散[1]であり，輝度変化が安定した状態のとき分散値は小さくなる．図 2.9 に示すように過去の短期間（K フレーム）における輝度値の分散をしきい値に反映させることで，天候・時間帯等の背景変化に対してその影響を受けにくくなり，検出性能の向上が期待できる．また式 (2.7) による分散は，平均を求める必要がなく，計算コストを抑えることが可能である．(2.7) 式により求めた分散値を用いて，各フレームにおいて変動しきい値を次式によ

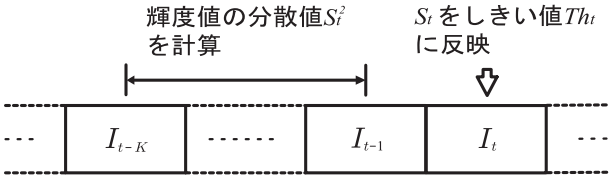


図 2.9: 分散の計算

り更新する．$S_t(u,v)$ の係数 α はしきい値の大きさを決定する値である．

$$Th_t(u,v) = \alpha \cdot S_t(u,v) \tag{2.8}$$

しかし，分散の変動が激しい領域では，Th_t の変動も激しく移動物体を検出できない場合がある．そこで，しきい値の変動を抑制するために過去のしきい値の重み付けによる更新を行う．しきい値 $Th_t^{'}(u,v)$ を次式により毎フレーム更新する．

$$Th_t^{'}(u,v) = \beta \cdot Th_t(u,v) + (1-\beta)Th_{t-1}^{'}(u,v) \qquad (0<\beta<1) \tag{2.9}$$

ただし，$M_{t-1}=1$ の場合は $Th_t^{'} = Th_{t-1}^{'}$ とする．β は，新しく計算した分散値 $S_t(u,v)$ を最新フレームにおけるしきい値 Th_t にどれだけ反映させて更新するかを決定する定数である．

図 2.10 に，あるピクセルにおける輝度の変動（差分値 Δ）としきい値を示す．移動物体は 33〜55 フレームに通過している．移動体通過フレームを全て検出できていないのは，移動物体の面積が大きい場合，フレーム間差分では変化が生じるエッジ領域のみ検出するからである．Th_t の場合，37 フレーム目の輝度変化に追従しており，40 フレーム目を検出できていない．$Th_t^{'}$ では，分散値の変動が抑えられ 40 レーム目の検出が可能である．日中における各領域での輝度変化を表す差分値 Δ と分散しきい値 Th_t' の時間変化を図 2.11 に示す．(a) は影領域，(b) はエッジ領域での差分値としきい値の変動である．日中の影領域においては，日向とのコントラストにより影が濃くなり移動体通過時には微小な輝度変化しか観測されない．図 2.11 (a) より，しきい値を 7 程度に設定しなければ，77〜84,96〜101 フレームの移動物体を検出できないことがわかる．一方，エッジや木の揺れが生じる領域においては輝度変化が激しく，しきい値を 30 程度に設定しなければ誤検出が発生する．この様な状況では，固定しきい値で対応することは困難である．分散しきい値では，各ピクセル毎に過去の輝度変化を反映して設定するため，検出可能となる．図 2.12 に影領域における移動体検出例を示す．固定しきい値においては，エッジ部分や木の揺れなどの誤検出が発生するためしきい値を低く設定できない．そのため，影領域における移動物体をほとんど検出できていない．一方，分散しきい値では，輝度変化の小さいピクセルはしきい値が低く設定されるため，検

[1]通常の分散式からの変形は付録 B.1 に示す．

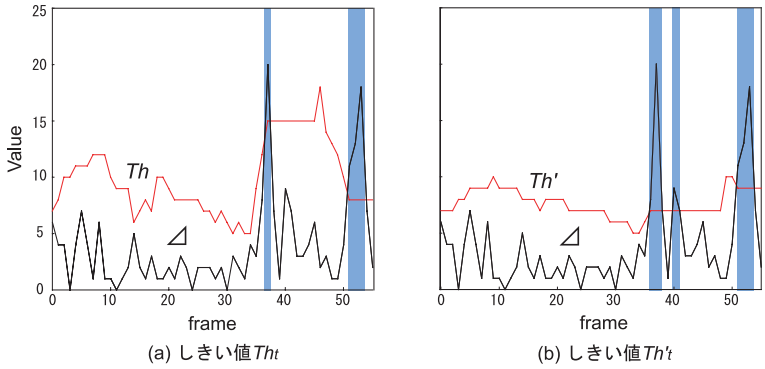


図 2.10: しきい値の遷移

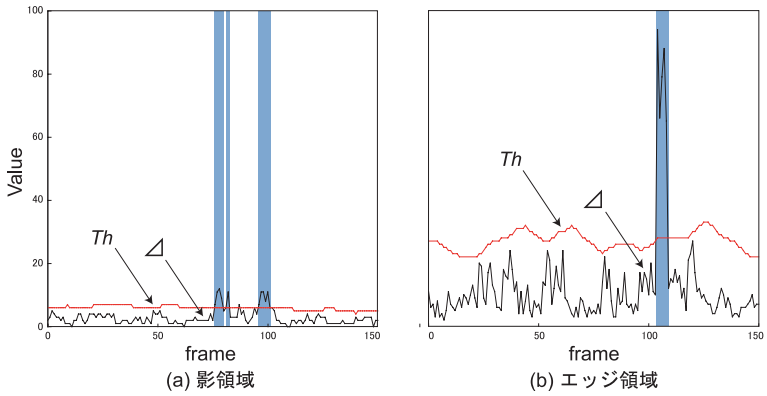


図 2.11: 影とエッジ領域におけるしきい値

出感度を上げることになり影領域での検出が可能となる．変化の大きいピクセルはしきい値が高く設定され，検出感度を下げることになり木の揺れ等の誤検出が抑制される．

2.1.5　3 枚の画像を用いたフレーム間差分法

通常，フレーム間差分法を用いて移動体検出を行うと，移動体領域だけでなく背景領域も検出される．フレーム間差分を用いてより正確な移動体領域を検出する手法として，3 枚の画像を用いた手法が提案されている．図 2.13 に示すように 3 枚の画像を用いたフレーム間差分法では，まず，A と B，B と C の差分画像 AB と BC を作成し，しきい値処理を施し 2 値画像を得る．次に 2 値画像 AB と BC の論理積処理 (AND 処理) を行い，AB と BC の共通領域をとり出すことで，画像 B における移動体領域を得ることができる．この手法による差分結果を図 2.14 に示す．図 2.14(f) から移動体領域のみが検出されていることがわかる．

(a) 時刻tの画像　(b) 固定しきい値(*Th*=12)　(c) 分散しきい値

図 2.12: 影領域における移動体検出

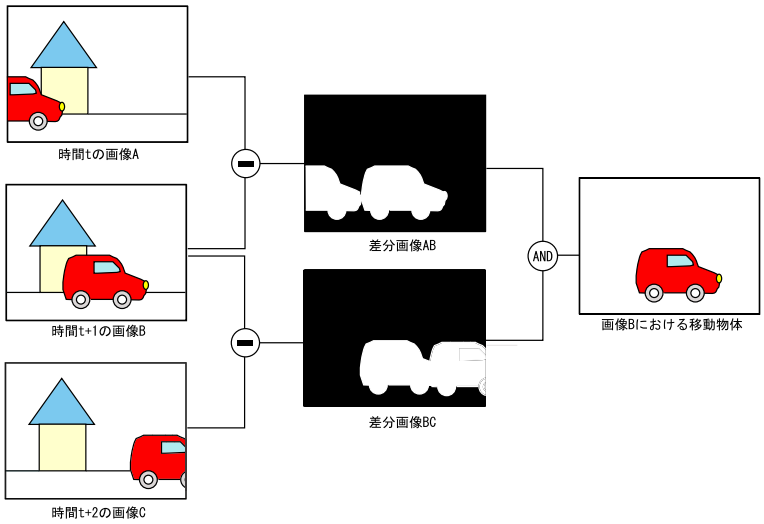


図 2.13: 3 枚の画像によるフレーム間差分

2.2 背景差分法 【Source Code】

背景差分法は，入力画像とあらかじめ用意した背景画像 (モデル) と現時刻 t との画像の差分により物体を検出する手法である (図 2.15)．2.1 で述べたフレーム間差分法では，移動体領域が分断される，静止物体を検出できないという問題があるが，背景差分は物体領域全体を検出することが可能である (図 2.17)．背景差分は，次式により求めることができる．

$$\Delta_t = |I_t - B_t| \tag{2.10}$$

ただし，I_t は時刻 t の画像，B_t は背景画像とする．急激な輝度値の変化がピクセル上に生じたとき，変化量 Δ_t の値は大きくなる．ここで，ピクセル状態を表す M は，輝度変化を処理することにより，次式に示すように物体 (1) と背景 (0) に判定する．Th は，急激な変化を判定する閾値で

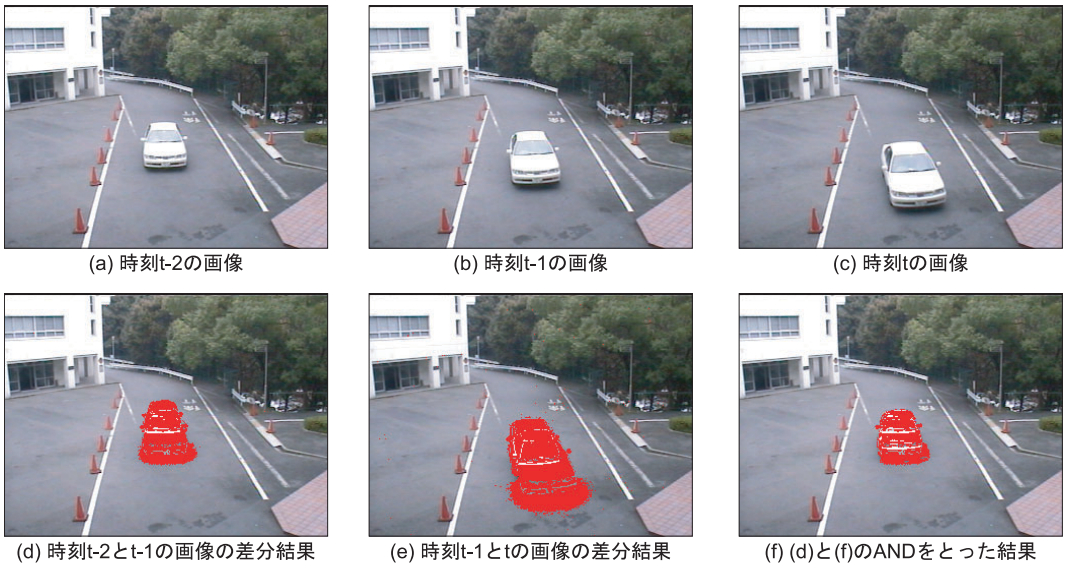


図 2.14: 3 枚の画像によるフレーム間差分結果

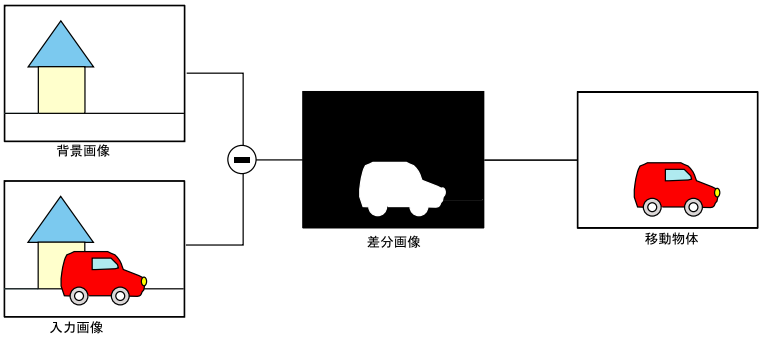


図 2.15: 背景差分法

ある．

$$M_t(u,v) = \begin{cases} 1 & ,\Delta t(u,v) > Th \\ 0 & ,\text{Otherwise} \end{cases} \tag{2.11}$$

背景差分法は背景画像を必要とするため，特に屋外環境のような背景が変化するシーンにおいては以下の問題に対処する必要がある．

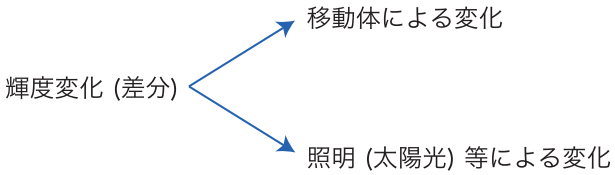


図 2.16: 背景変化における問題

1．照明変化 照明変化 (照明強度や照明色の変化) や影による背景物体の見えの変化 [5][6][11]．

2．背景物体の変動 木や旗の揺れなどの背景物体の変動 [7][8][3]．

これらの問題に対してユークリッド距離では，距離計算に基づく効果的な背景差分法が提案されている．

(1) 背景差分による差分結果

(2) フレーム間差分による差分結果

図 2.17: 背景差分とフレーム間差分の比較

2.2.1　背景差分における距離計算

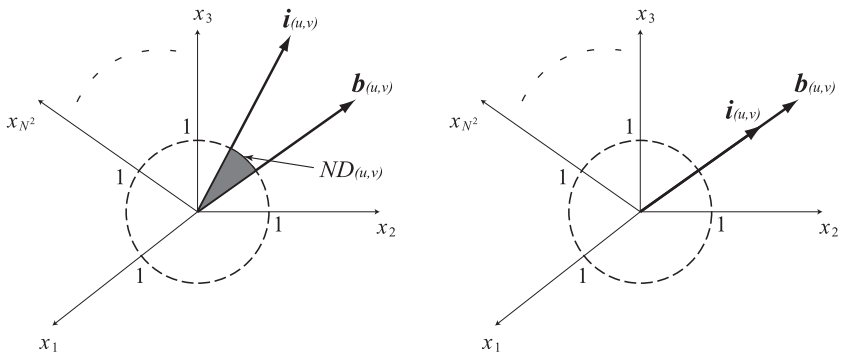
正規化距離を用いた背景差分法

長屋ら [5] は，1. の照明変化による背景物体の誤検出を防ぐ手法として，正規化距離を用いた背景差分法を提案している．照明変化の影響を受けにくい正規化距離を利用することにより，移動物体による輝度変化のみを検出することができる．

正規化距離を計算するためには，まず，画像を $N \times N$ 画素のブロックに分割し，そのブロックを N^2 次元のベクトルで表現する．ここで，ベクトルの要素はそれぞれブロック中の輝度値に対応している．$\boldsymbol{i}_{(u,v)}$ と $\boldsymbol{b}_{(u,v)}$ をそれぞれ入力画像と背景画像中の同一位置のブロックに対応するベクトルとする．すると，正規化距離は次式で表される．

$$ND(\boldsymbol{i}_{(u,v)}) = \left| \frac{\boldsymbol{i}_{(u,v)}}{|\boldsymbol{i}_{(u,v)}|} - \frac{\boldsymbol{b}_{(u,v)}}{|\boldsymbol{b}_{(u,v)}|} \right| \tag{2.12}$$

ただし，$|\cdot|$ はベクトルの大きさを表している．図 2.18 に示すように，照度変化が生じた場合に $ND(\boldsymbol{i}_{(u,v)})$ はほぼゼロになるため，移動物体と照明変化による輝度変化を判別することができる．松山ら [6] は，正規化距離による背景差分を改良し，照明条件の推定に基づく背景差分と組み合わせている．



(a) 物体の通過　　(b) 照明変化

図 2.18: 正規化距離

2.2.2　背景画像の生成

全加算平均による背景画像の生成

背景画像を生成する最も簡単な方法として全加算平均による背景画像の生成法が用いられている．

移動物体の存在しない映像 S_0 から，各ピクセルにおいて輝度 $I_0(u, v; t)$ の平均値を計算し，背景画像 $I_b(u, v)$ とする．

$$I_b(u, v) = \frac{1}{N} \sum_{t=0}^{N-1} I_0(u, v) \tag{2.13}$$

ただし，N は映像 S_0 のフレーム数とする．全加算平均を用いると，簡単に背景画像を生成することができるが，照明変化や背景物体の変動により生じる問題に対処することは困難である．

事後確率を用いた背景差分法 【Source Code】

木の揺れ等の背景物体が変動するシーン対しては，画素値の出現頻度から背景モデルを生成する手法が提案されている [3].

背景に属する画素のクラスを ω_0 とし，移動物体に属する画素のクラスを ω_1 とする．ある画素の画素値が I のとき，その画素が ω_0 に属する確率を $p(\omega_0|I)$ とし，ω_1 に属する確率を $p(\omega_1|I)$ とする．ここで，ベイズの定理より，次式のように書き直すことができる．

$$p(\omega_0|I) = \frac{p(\omega_0)p(I|\omega_0)}{p(I)} \tag{2.14}$$

$$p(\omega_1|I) = \frac{p(\omega_1)p(I|\omega_1)}{p(I)} \tag{2.15}$$

ただし，$p(I)$ は次式により計算される．

$$p(I) = p(\omega_0)p(I|\omega_0) + p(\omega_1)p(I|\omega_1) \tag{2.16}$$

一定時間，背景画像としてある画素の画素値を観測するとき，画素値 I の度数を $q(I)$ とすると，背景画像の画素値の確率密度関数は次式で求めることができる．

$$p(I|\omega_0) = \frac{q(I)}{\sum_{I=0}^{255} q(I)} \tag{2.17}$$

ここで階調数は 256 であるとする．

つぎに，移動物体の画素の画素値は未知であるため，画素値の現れ方が一様であると仮定すると次式となる．

$$p(I|\omega_1) = \frac{1}{256} \tag{2.18}$$

各画素の画素値 I を観測したときに，$p(\omega_0|I)$ と $p(\omega_1|I)$ を計算し，値が大きいほうの事象がより起こりやすい．すなわち，$p(\omega_0|I)$ が大きければその画素は背景に属し，$p(\omega_1|I)$ が大きければ移動物体に属する．値の大小は，次式を計算することで求めることができる．

$$\frac{p(\omega_0|I)}{p(\omega_1|I)} = \frac{p(\omega_0)p(I|\omega_0)}{p(\omega_1)p(I|\omega_1)} \tag{2.19}$$

ここで，$p(\omega_0)$ はその画素が背景である事前確率であり，$p(\omega_1)$ はその画素が移動物体である事前確率であり，$p(\omega_0) + p(\omega_1) = 1$ となる．$p(\omega_0)$ と $p(\omega_1)$ の初期値はそれぞれ 0.5 とし，$p(\omega_1)$ の更新は次式により行う．

$$p_{t+1}(\omega_1) = p_t(\omega_1|I) \tag{2.20}$$

ただし，誤検出を減らすため $p(\omega_1)$ の最大値は 0.5 とする．背景で木が揺れている映像に対して，全加算平均を用いて背景画像を生成する手法と出現頻度から背景モデルを生成する手法を用いて背

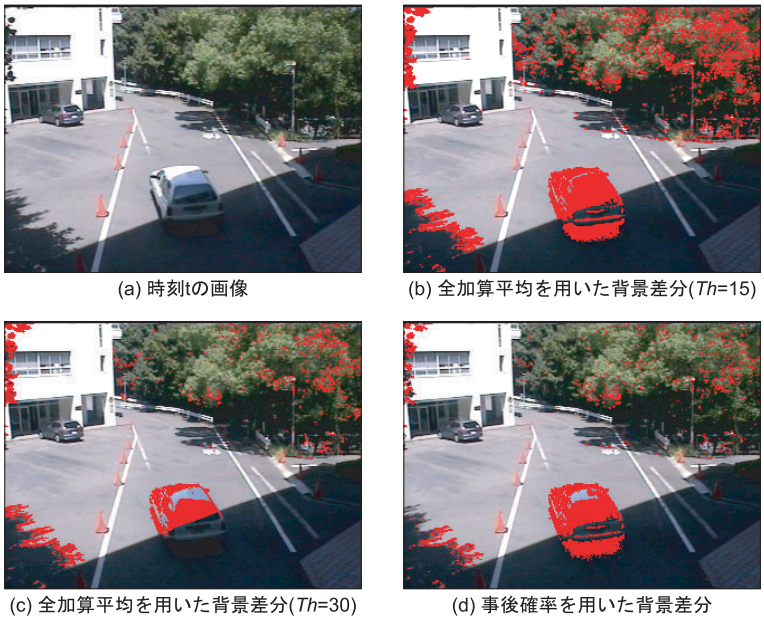


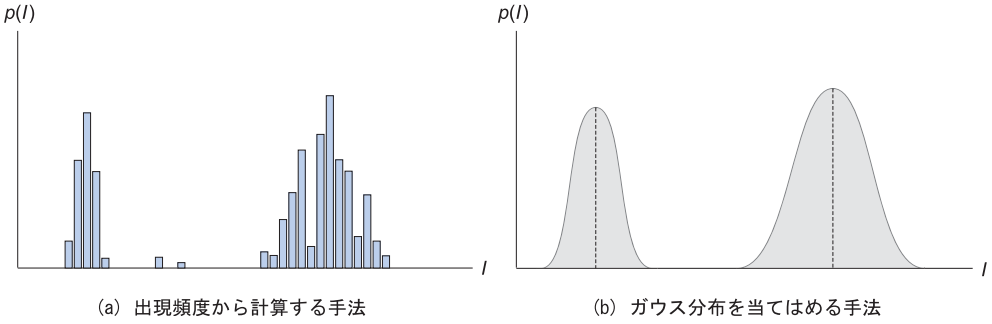
図 2.19: 木が揺れているシーンでの背景差分結果

景差分をした結果を図 2.19 に示す．図 2.19(b)，(c) から，全加算平均を用いた場合，しきい値を 15 に設定すると移動物体の検出はできるが，木や木の影の揺れを誤検出する．しきい値を 30 に上げると，背景物体の誤検出は抑制されるが，影領域内の車を検出できなくなる．図 2.19(d) から，出現頻度から背景モデルを生成することにより，背景物体の変動による誤検出を抑制し，移動物体を検出できていることがわかる．背景物体の誤検出が完全に抑制されていないのは，確率密度関数 $p(\omega_0|I)$ を各階調ごと独立にを計算しているため，学習時と違う木の揺れ方をした場合に物体を判定するためである．

混合ガウス分布を用いた背景変動のモデル化

Wren ら [7] は，木の揺れなどにより生じる輝度変化を各ピクセルごとに正規分布を用いてモデル化している．しかし，背景で生じる輝度変化は複数に分布する場合があり単一のガウス分布でモデル化できるとは限らない．Stauffer ら [8] は，複数のガウス分布を用いることにより，複雑な背景変化をモデル化する手法を提案している．

事後確率を用いた背景差分法では，背景輝度の頻度分布から個々の階調値ごとに生起確率を計算するため，背景モデルは頻度分布に依存する (図 2.20(a))．そのため，学習時と違う背景変動が生じると対応できない．一方，ガウス分布を用いてモデル化を行う手法は，学習時のデータにガウス分布を当てはめて背景変動をモデル化するため，学習データが不完全でもガウス分布で近似することができる (図 2.20(b))．図 2.21 に，木の影が揺れているピクセルの約 1 分間の輝度値をプロットした結果を示す．注目ピクセル上には，道路の色と木の影の色が現れるため輝度値の分布は 2 つに分かれている．これらの各分布を 2 つのガウス分布で表現できれば，入力パターンを背景と物体に



(a) 出現頻度から計算する手法　(b) ガウス分布を当てはめる手法

図 2.20: 背景の輝度変動のモデル化手法

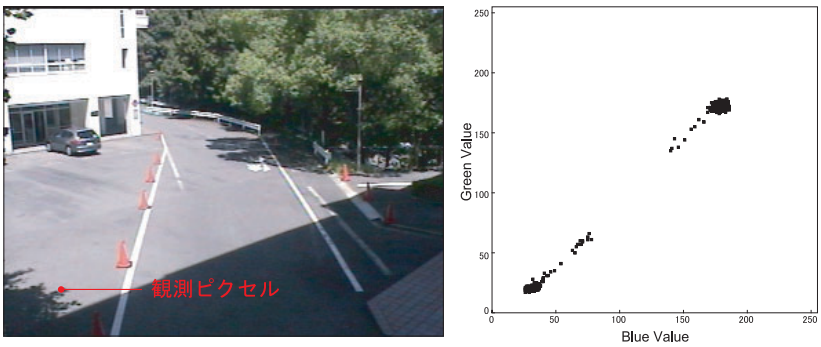
判定が可能となる．

各ピクセルにおける背景輝度の分布 $\{I_1, \cdots, I_t\}$ を混合ガウス分布によりモデル化する．

$$P(\boldsymbol{I}_t) = \sum_{i=1}^{K} \omega_i * \eta(\boldsymbol{I}_t, \boldsymbol{\mu}_{i,t}, \boldsymbol{\Sigma}_{i,t}) \tag{2.21}$$

ただし，K はガウス分布の数，$\omega_{i,t}$ は重み係数，$\boldsymbol{\mu}_{i,t}, \boldsymbol{\Sigma}_{i,t}$ はガウス分布の平均と共分散行列とする．η は，ガウス分布の確率密度関数であり，次式で表される．

$$\eta(\boldsymbol{I}_t, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^M |\boldsymbol{\Sigma}|}} \cdot \exp\{-\frac{1}{2}(\boldsymbol{I}_t - \boldsymbol{\mu}_t)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{X}_t - \boldsymbol{\mu}_t)\} \tag{2.22}$$



(a) 画像中の観測ピクセル　(b) 観測ピクセル上の輝度値を約1分間プロットした結果

図 2.21: 木の影が揺れているピクセルにおける輝度の分布

ロバスト統計に基づいた背景推定 【Source Code】

背景画像を生成するためには，移動物体を含まないシーンの画像が必要となる．しかし，道路や人通りが多い場所などを対象とする場合，移動物体が含まれていない画像を得ることが困難となる．このような場合において，ロバスト統計を用いて背景画像を推定する手法が提案されている [9].

図 2.22(c)(d)(e) は道路を撮影した映像中の 1 フレームを表している．この映像における注目ピクセル (図中の十字の交点) の輝度値の変化を図 2.22(a) に示す．図 2.22(a) に示されるような輝度値の時間的変化は，人や車 (移動物体) が道路の表面 (背景) をしばしば隠すために起こる．このような各画素の輝度値が時間的に変化するような時系列画像データから背景とみなす部分の輝度値を推定するためには，自動車のような移動物体によって引き起こされる輝度値の急激な変動を無視し，時間的にあまり変化しないような輝度値を推定する必要がある．つまり，移動物体により引き起こされる急激な輝度値の変化の影響をあまり受けないような推定の仕組みが必要となる．ロバスト推定法を用いると例外値をある程度含むようなデータからでも安定なモデル推定が可能となる．

最も単純なロバスト推定法の一つとしてメジアンを利用したものがある．これは，データのヒストグラムが与えられたときにその累積確率が $\frac{1}{2}$ となるような点を求める方法で，メジアンにより推定された値は平均値とは異なり例外値の影響を受けにくいものとなっている．

図 2.22(b) は，注目ピクセルにおける 1〜1100 フレーム間の輝度値のヒストグラムを示したものである．このヒストグラムから，輝度値の最頻値は 81〜87 あたりであり，この部分が背景の輝度値を表しているものと考えることができる．このとき，このヒストグラムに対して平均値とメジアンをそれぞれ求めると，平均値は 79.8 となり低輝度の例外値の影響を受けて最頻値から左の方へとずれている．一方，メジアンは 83.0 で，輝度値の最頻値とほぼ一致していることがわかる．ここで用いた道路シーンの場合，ヒストグラム中の最頻値が道路の表面を表していると考えることができるので，ロバスト推定を用いることにより移動物体を含む画像列から背景の輝度値を推定することが可能となる．

しかし，メジアンを用いたロバスト推定法は，オンラインで用いることを考えると実用的ではない．メジアンの計算にはヒストグラムが必要であり，画像中の全画素についてのヒストグラムを更新するには計算コストが大きすぎるためである．そのため，M 推定を利用して背景の推定値を逐次的に更新できる手法が提案されている [10].

2.2.3　背景画像の更新

生成した背景画像はシーンの変化に対して適応的に更新する必要がある．最も単純な方法としては，一定時間ごとに背景画像を取り込む方法があるが，背景を更新するタイミングを誤ると誤検出が生じるという問題がある．そのため，逐次入力される画像に対して背景画像を徐々に更新する手法が提案されている [11].

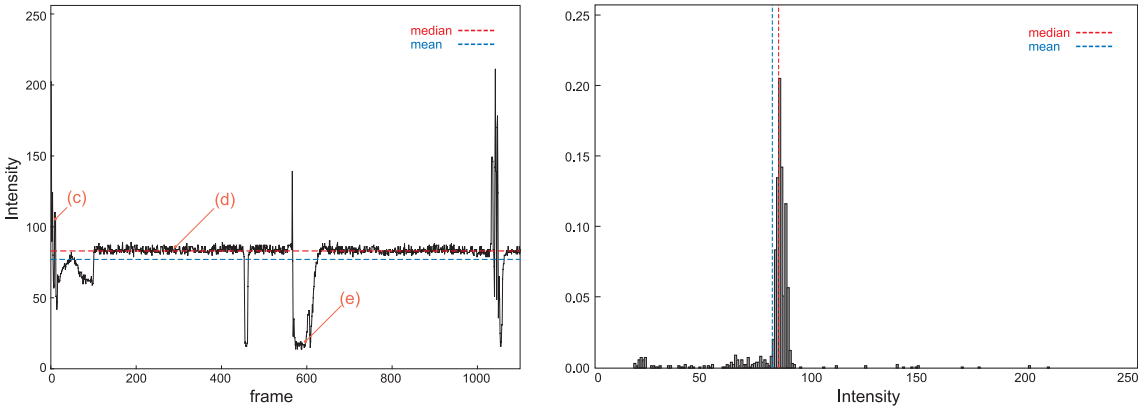
時刻 t の入力画像を $I(t)$，背景画像を $B(t-1)$ とし，以下の式を用いて環境変化に追従するように更新を行う．

$$B(t) = \alpha \cdot I(t) + (1-\alpha)B(t-1) \qquad (2.23)$$
$$(0 < \alpha < 1)$$

ただし，物体と判定されたピクセルの背景画像は更新しない．α は背景と判定されたピクセルを背景画像にどれだけ反映させて更新するかを決定する定数である．

連続時間システムにおける初期背景画像の時間変化を次式に示す.

$$\bar{P}(t) = F(t)\bar{P}_0 \qquad (2.24)$$
$$F(t) = e^{-\frac{t}{\tau}} \qquad (2.25)$$



(a) 注目ピクセルにおける輝度値の時間変化　(b) 注目ピクセルにおける輝度値のヒストグラム

(c) 10フレーム目の画像　(d) 300フレーム目の画像　(e) 600フレーム目の画像

図 2.22: 道路シーンの注目ピクセルにおける輝度値の時間変化

e の係数 $-\frac{1}{\tau}$ が負の値をとるとき $0 < e^{-\frac{t}{\tau}} < 1$ となり，$\bar{P}(t)$ は指数関数的に減少する．このため，初期背景画像 $\bar{P}_0$ の情報は時間とともに減少する．連続時間システムから離散時間システムへの変換を考えるために，信号を時間間隔 T でサンプリングする．時間 t は，離散的に進行することから，現在の時刻 t を $t_k = kT(k = 0, 1, 2, \cdots)$ と表すことができる．ここに定義したサンプル時刻を表す記号 t_k を用い $\bar{P}(t_k)$ を $\bar{P}_k$ と書けば

$$\bar{P}_k = e^{\frac{t_k}{\tau}} \bar{P}_0 = e^{-\frac{kT}{\tau}} \bar{P}_0$$

となる．また t_k につづくサンプル時刻 t_{k+1} においては

$$\bar{P}_{k+1} = e^{-\frac{T}{\tau}} \bar{P}_k$$

とななる．なお，このとき $e^{-\frac{T}{\tau}}$ は一定の値をとるため定数 α とおくことができる．

$$\bar{P}_{k+1} = \alpha \bar{P}_k \tag{2.26}$$

式 (2.26) に現在の入力 P_{k+1} を与えると

$$\bar{P}_{k+1} = \alpha \bar{P}_k + qP_{k+1}$$

となる．ここでは $q = 1 - \alpha$ とする．

$$\bar{P}_{k+1} = \alpha \bar{P}_k + (1 - \alpha) P_{k+1} \tag{2.27}$$

τ の値を 1 ～ 6 まで変化させた場合の $e^{-\frac{T}{\tau}}$ の減衰の様子を図 2.23 に示す.

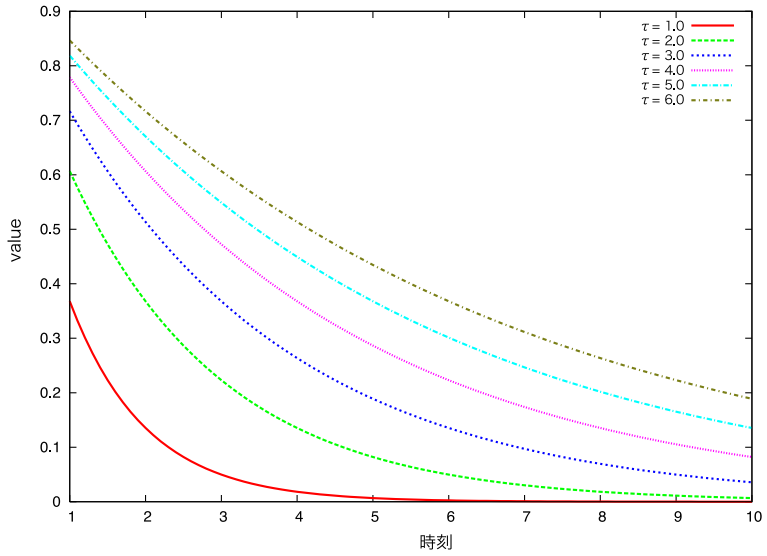


図 2.23: τ の値における $e^{-\frac{T}{\tau}}$ の変化

α の変化による背景画像の変化

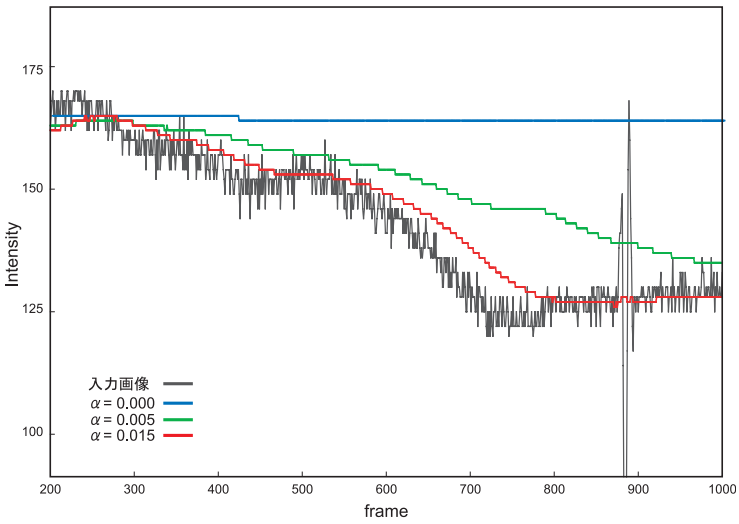
太陽が雲に隠れ背景が変化がする映像に対して，式 (2.23) の α を 0.000，0.005，0.015 に設定した場合での背景画像の変化を図 2.24 に示す．図 2.24(a) が背景変化前の画像であり，図 2.24(b) が太陽が雲で隠れ背景が変化した後の画像である．図 2.24(c) に注目ピクセルにおける入力画像と背景画像の輝度値の変化を示す．青は $\alpha = 0.000$，緑は $\alpha = 0.005$，赤は $\alpha = 0.015$ に設定した場合の背景画像の輝度値である．$\alpha = 0.000$ に設定した場合，背景画像は更新されないため時間が経過しても輝度値は変化しない．図 2.24(e) に示す差分値は時間と共に上昇し，600 フレーム付近から背景を誤検出している．$\alpha = 0.005$ に設定した場合においても，背景変化に対して背景画像を更新する速度が遅く，図 2.24(e) に示すように 700～800 フレームの間で誤検出が生じている．$\alpha = 0.015$ に設定した場合は，背景画像の更新が背景変化に追従しており，移動体通過フレームのみを検出できている．

2.3　オプティカルフロー

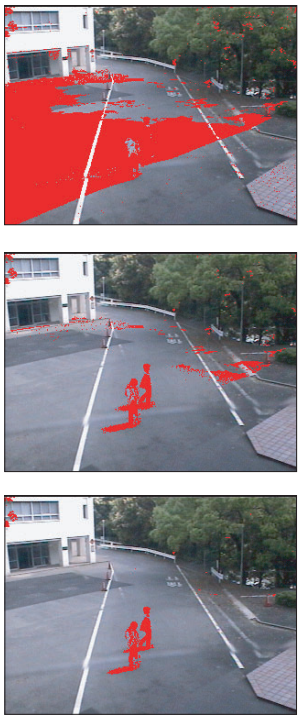
オプティカルフローとは画像間における濃淡パターンを対応付けし，その移動量をベクトル表現したものである [13]．求めたベクトルの向きと大きさから，移動カメラの場合でも，カメラの動きと異なるフローを求めることで，移動体の検出が可能となる．ここではオプティカルフローを求める代表的な手法である，ブロックマッチング法と勾配法について述べる．

(a) 200フレーム目の入力画像

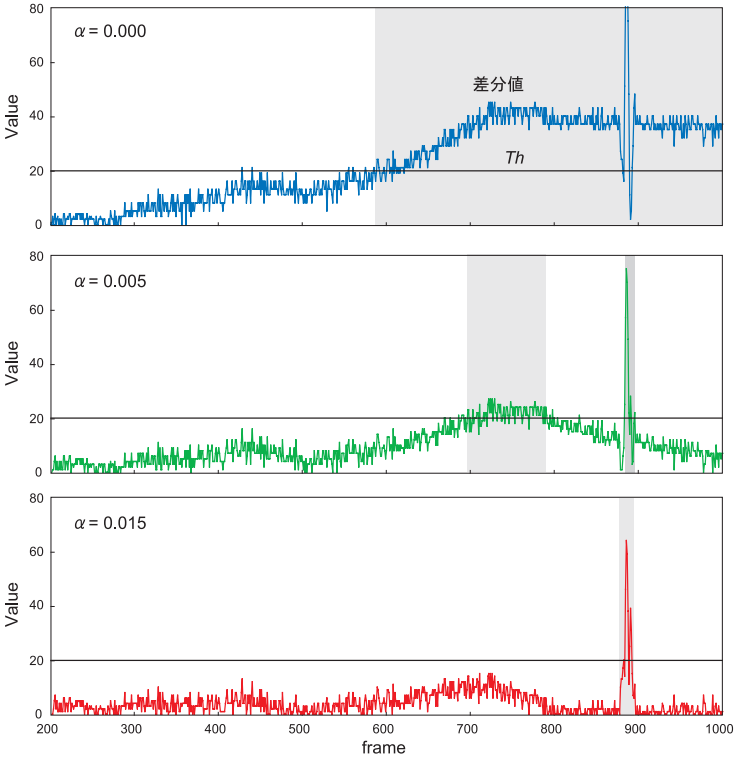
(b) 700フレーム目の入力画像



(c) 注目ピクセルにおける入力画像と背景画像の輝度値の時間変化



(d)700フレーム目の背景差分結果



(e) 注目ピクセルにおける差分値の時間変化

図 2.24: αを変化させた場合での背景画像の時間変化

2.3.1　ブロックマッチング法 【Source Code】

ブロックマッチング法 (別名：領域ベース法) はテンプレートマッチングを用いてオプティカルフローを求める方法である．図 2.25 に示すように，画像 A の a 位置にある画素が画像 B のどの位置に移動したかを調べる．その際，a の画素を左上とした小領域 ($N \times N$px) をテンプレートのブロックとし，画像 B の b 位置 (a と同位置) を左上にして，ある範囲を探索することになる．オプティカルフローを求めるためは，まず前フレーム (画像 A) の評価するテンプレートと現フレーム (画像 B) の注目ブロックの濃淡パターンがどれほど合致しているかを計算する．濃淡パターンの合致を決定する方法として絶対値差分和や相関係数値等を利用することも出来るが，今回は計算コストが小さい残差逐次検定法について記載する．

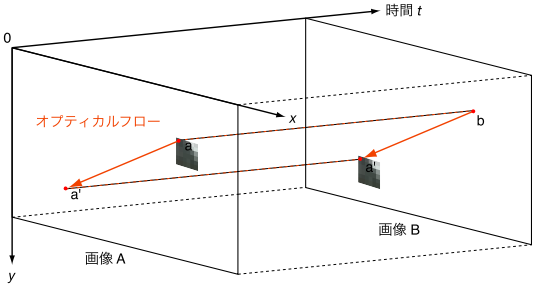


図 2.25: ブロックマッチング法

探索画像 I 上で $N \times N$ の画素のテンプレート画像 T を動かし, 濃淡パターンの照合を行うとする．(x, y) は入力画像内におけるテンプレート画像の左上位置を示す．このとき式 (2.28) に示すテンプレート画像と探索画像との残差が最小になる (x, y) の位置を求めることにより, 探索画像とテンプレート画像との一致を決定する．

$$R(x,y) = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} |I(x+m, y+n) - T(m,n)| \tag{2.28}$$

計算時間短縮のために, 残差を加算した値がある閾値を超えた場合に計算を終了し，次の注目領域での計算に移行する措置をとっている．

時系列の画像に対し，フレーム t と $t-1$ のオプティカルフローをブロックマッチング法により計算した結果を図 2.26 に示す．

2.3.2　勾配法

勾配法は，連続する 2 枚の画像において移動体の特徴を表す画像関数 (濃淡分布) の移動量は微小であるという仮定を基にオプティカルフローを求める手法である．

画像上の画素 (x, y) の時刻 t における画素値を $E(x, y, t)$ とし，時間 Δt の間にその画素が $(\Delta x, \Delta y)$ だけ移動したとする．画素 (x, y) と $(x + \Delta x, y + \Delta y)$ の画素値は変化しないと仮定すると，次式が成り立つ．

$$E(x, y, t) = E(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.29}$$

図 2.26: ブロックマッチング法により推定されたオプティカルフロー

次に，画素値は滑らかに変化していると仮定して微小変化を式で表現する．微小な値を近似する方法としてテイラー展開を用い，一次近似すると次式になる．

$$E(x,y,t) \fallingdotseq E(x,y,t) + \Delta x\frac{\partial E}{\partial x} + \Delta y\frac{\partial E}{\partial y} + \Delta t\frac{\partial E}{\partial t} \tag{2.30}$$

両辺を Δt で割ると，

$$\frac{\Delta x}{\Delta t}\frac{\partial E}{\partial x} + \frac{\Delta y}{\Delta t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0 \tag{2.31}$$

ここで Δt は限りなく 0 に近づくと考えると次式を得る．

$$\frac{\partial x}{\partial t}\frac{\partial E}{\partial x} + \frac{\partial y}{\partial t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0 \tag{2.32}$$

ここで，画素 (x,y) におけるオプティカルフローの x 成分を $\frac{\partial x}{\partial t} = u$，$y$ 成分を $\frac{\partial y}{\partial t} = v$ とし，画像上の画素値の勾配を E_x, E_y，画素値の時間微分を E_t とすると，勾配法の拘束条件 (式 (2.33)) を得ることが出来る．

$$E_x u + E_y v + Et = 0 \tag{2.33}$$

上式を解くことで，オプティカルフローを推定することが出来るが，推定すべき未知数は u，v の 2 つであるのに対し，式は一つしか与えられないため，解を一意に決定することは出来ない．そこでいくつかの仮定を導入することで u，v を決定する手法が提案されている．仮定は “ある注目画素の近傍での動きは滑らかである”(ローカル (局所) 法)，“物体内の動きは滑らかに変化する”(グローバル (大域) 法) を空間勾配，時間勾配と関係づけることで取り決める．例えば空間的局所最適化法の一種である Lucas-Kanade 法は，“同一物体の濃淡パターン上の局所領域において，得られるオプティカルフローの拘束方程式は同一の解を持つ”，と仮定することで (u,v) の値を決定している．詳しい算出法は第 2.3.2 章に記載する．しかし，どの拘束条件を用いても，大元にある仮定において時間的，空間的に滑らかな画像関数の存在を前提としているため，仮定が成り立たない場合にフローの精度が落ちる．

Lucas-Kanade (LK) 法 【Source Code】

空間的局所最適化法の一種である Lucas-Kanade 法は，“同一物体の濃淡パターン上の局所領域において，得られるオプティカルフローの拘束方程式は同一の解を持つ”，と仮定することで (u, v) の値を決定する [15].

これは局所領域においてそのパターンの並び方は変化しないと言うことを指す．例えば局所領域のサイズを 3×3 とすると，オプティカルフローの拘束方程式は次のように表すことが出来る.

$$\begin{aligned} E_{x1}u + E_{y1}v &= -E_{t1} \\ E_{x2}u + E_{y2}v &= -E_{t2} \\ &\vdots \\ E_{x9}u + E_{y9}v &= -E_{t9} \end{aligned} \tag{2.34}$$

(u, v) にそれぞれかかる係数をまとめ E とする．

$$E = \begin{bmatrix} E_{x1} & E_{y1} \\ E_{x2} & E_{y2} \\ \vdots & \vdots \\ E_{x9} & E_{y9} \end{bmatrix} \tag{2.35}$$

E は 9×2 の行列式となる．同様に，$t = (E_{t1}, E_{t2}, \cdots, E_{t9})^T$，$P = (u, v)^T$ とするとこれらの式は $EP = t$ とまとめることが出来る．この両辺に E の転置行列をかけると次式が得られる．

$$E^T EP = E^T t \tag{2.36}$$

これにより $E^T E$ は 2×2 の行列，$E^T t$ は 2×1 のベクトルとなる．

$$\begin{bmatrix} \sum E_x E_x & \sum E_x E_y \\ \sum E_x E_y & \sum E_y E_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum E_x E_t \\ \sum E_y E_t \end{bmatrix} \tag{2.37}$$

総和の上限は局所領域のサイズとなる．これは先に入力する必要があるが，式 (2.38) より一意に (u, v) の値を定めることが出来るようになる．

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum E_x E_x & \sum E_x E_y \\ \sum E_x E_y & \sum E_y E_y \end{bmatrix}^{-1} \begin{bmatrix} \sum E_x E_t \\ \sum E_y E_t \end{bmatrix} \tag{2.38}$$

時系列の画像に対し，フレーム t と $t-1$ のオプティカルフローを LK 法により計算した結果を図 2.27 に示す.

2.3.3　ブロックマッチング法と LK 法の比較

一般的にブロックマッチング法よりも LK 法の方が高速にオプティカルフローを求めることができることが知られている．両手法に対し，局所領域のサイズを変更させ、オプティカルフローの計算時間を計測した結果を表 2.1 に示す．また計測環境を表 2.2 に示す.

図 2.27: LK 法により推定されたオプティカルフロー

表 2.1: 各手法のパラメータ変化による計算コスト [msec]

局所領域サイズ	ブロックマッチング法	勾配法 (LK 法)
3 × 3	2708.46	105.98
5 × 5	8731.69	116.05
7 × 7	19076.43	133.50
9 × 9	33545.50	160.14

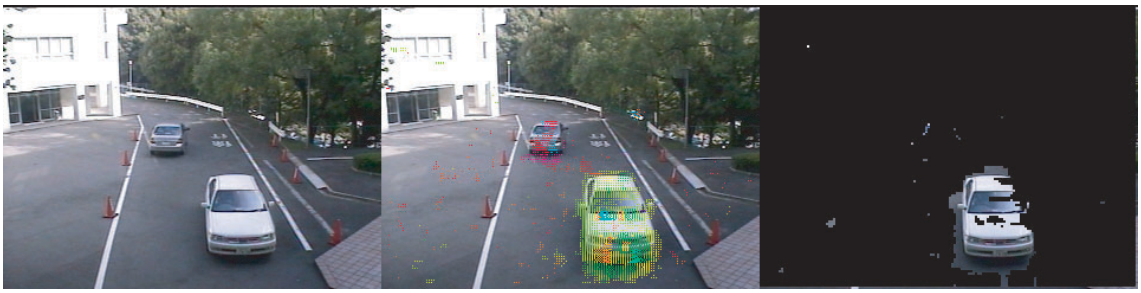
表 2.2: 計測環境

CPU	PowerPC G5 dual 2.5GHz
OS	Mac OS X 10.3.9
Memory	1.5GB DDR SDRAM
Video Momory	128MB

結果からもわかる通り，ブッロクマッチング法に比べ，LK 法は圧倒的に速い速度で計算を終えることができる．またブロックマッチング法では，局所領域のサイズを大きくするにつれ，計算時間が指数関数的に増大していることがわかる．この理由として，ブロックマッチング法では，局所領域の移動を計測するために，1 ピクセル毎のマッチングを行っていることが挙げられる．それに対し，LK 法では，マッチングをとるのではなく，式 (2.35)〜式 (2.38) の行列計算を行っているだけなので，局所領域を大きくしても，計算コストの増加は，式 (2.36) だけであるため，計算時間が肥大することは無い．

2.3.4　オプティカルフローからの移動体検出

オプティカルフローを用いて移動体検出を行うことができる．カメラが固定されている場合，移動体領域のオプティカルフローは非ゼロのベクトル値をもつ．そのベクトルをもつ画素を集めることで移動体検出を行うことができる．また，オプティカルフローから移動方向を知ることができるため，ある方向に向かって移動している物体のみを検出することも可能である．図 2.28 にオプティカルフローを用いて移動体を検出した例を示す．また図 2.28 では，下方向のフローで校正される物体のみを検出対象とした．



入力画像　　オプティカルフロー　　移動体検出例

図 2.28: オプティカルフローを用いた移動体検出の例

2.4　移動カメラ (旋回型) による移動体検出

監視対象領域が広く，かつ対象物体をより高解像度で撮影するためには，ズーム旋回カメラを使用することが考えられる．その際，カメラが旋回している状態では，画像全体に動きを持つため，2.1，2.2 で述べたフレーム間差分や背景差分による移動体領域の検出は不可能である．このような旋回型カメラより得られる動画像からの移動体検出には，画像の幾何変換による手法が提案されている [12]．幾何変換による手法では，図 2(a) のようにパン・チルト角における複数の背景画像をあらかじめ用意しておく．現在のカメラのパン・チルト角より最も近い背景画像 (c) を選択し，現在の入力画像を背景画像が重なるように幾何変換を施す．選択された背景画像と幾何変換後の入力画像 (d) の差分を求める．その結果，(e) に示すように旋回カメラから得られる画像内の移動体領域を検出することが可能となる．

(a) パン・チルト角における複数の背景画像

(b) 入力画像 (c) 選択された背景 (d) 幾何変換 (e) 差分結果

図 2.29: 画像幾何変換による移動体検出

2.5 移動体検出の評価方法

図 2.30 に移動体検出法の評価法を示す．ある物体が対象領域に現れてから消えるまでの間に，対象物体を検出できた時間を tp[sec]，対象物体を検出できなかった時間を fn[sec] とする．木の揺れや静止物体など本来検出すべき対象ではないものを誤検出した時間を fp[sec] とする．複数の誤検出パターンが出現した場合，出現時間が同時なものについては，それを一括して扱うこととし，その全てが消えるまでの時間を計測する．また，夜間，朝，夕暮に車のヘッドライトで照らし出された領域については，自動車本体を含んでいても誤検出とする．図 2.31 に tp，fn，fp の関係を示す．システムがなんらかの物体を検出した時間は tp+fp，対象とする本来検出されるべき物体が出現してからカメラの視野から外れるまでの時間は tp+fn となる．

- 移動体を検出した時間 : tp [sec]
- 移動体を検出できなかった時間 : fn [sec]
- 移動体以外を誤検出した時間 : fp [sec]

移動体検出の評価は，人や自動車などの対象物体の検出だけでなく，外乱による誤検出に対しても行う必要があることから，以下に示す移動体検出率と誤検出率を計算し評価を行う．本来検出されるべき時間内において，実際に検出できた時間の割合を移動体検出率とし，次式で算出する．

$$Detection = \frac{\Sigma tp}{\Sigma tp + \Sigma fn} \times 100[\%]$$

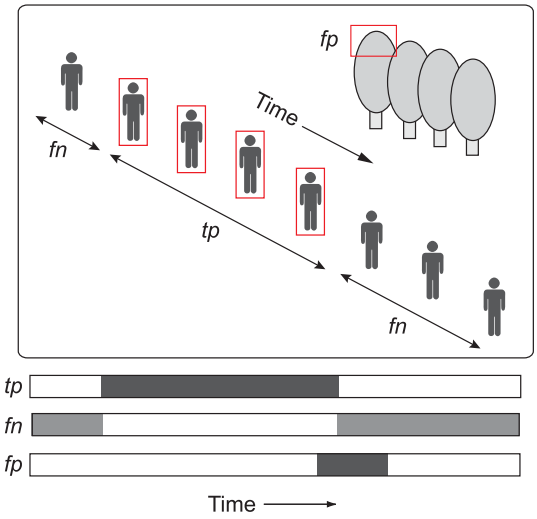


図 2.30: 検出時間に基づく評価法

システムが検出した時間において，検出すべきでないものの時間の割合を誤検出率とし，次式で算出する．

$$FalseDetection = \frac{\Sigma fp}{\Sigma tp + \Sigma fp} \times 100[\%]$$

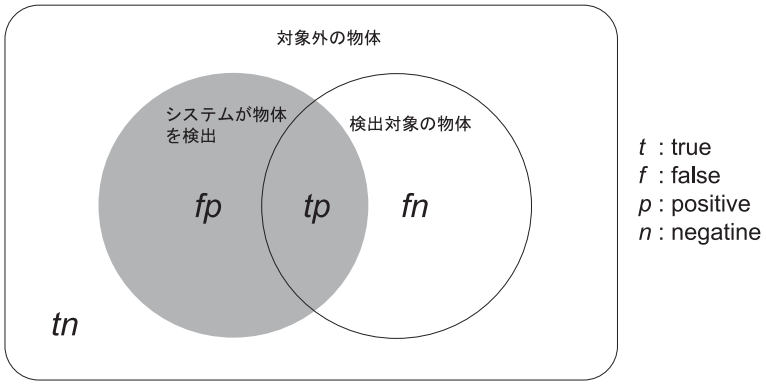


図 2.31: 検出の定義

参考文献

[1] 藤吉弘亘, 小村剛史, 矢入 (江口) 郁子, 香山健太郎, 吉水宏: “歩行者 ITS のためのフレーム間差分による移動体検出法とその評価”, 情報処理学会誌:コンピュータビジョンとイメージメディア, Vol. 45, No. SIG 13 (2004).

[2] 木村俊, 荒谷真一, 菅泰雄: “カラー情報による物体の抽出と特徴点検出による 3 次元形状の認識”, 第 8 回画像センシングシンポジウム講演論文集, H8, pp. 381-384 (2002).

[3] 中井宏章: “事後確率を用いた移動物体検出手法”, 情処学研報, SIG-CV90-1 (1994).

[4] 藤吉弘亘, 金出武雄: “複数物体の重なりを理解するレイヤー型検出法”, 第 7 回画像センシングシンポジウム, pp. 369-374 (2001).

[5] 長屋茂喜, 宮武孝文, 藤田武洋, 上田博唯, 伊藤敬一: “時間相関型背景判定法による移動物体検出” , 信学論 (D-II), Vol. J79-D-II No. 4, pp. 568-576 (1996).

[6] 松山隆司, 和田俊和, 波部斉, 棚橋和也: “照明変化に頑強な背景差分” , 信学論 (D-II), Vol. J84-D-II, No. 10, pp. 2976-2985 (2001).

[7] Wren, Christopher R., Ali Azarbayejani, Trevor Darrell, and Alex Pentland: “Pfinder: Real-time tracking of the human body”, In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 780-785 (1997).

[8] C.Stauffer and W.E.L. Grimson: “Adaptive background mixture models for real-time tracking”, In Computer Vision and Pattern Recognition 1999(CVPR99), Vol. II, pp. 246-253 (1999).

[9] 栗田多喜夫, 梅山伸二, 島井博行: “適応的なロバスト推定を用いた背景のモデリング” , 1998 信学シ大, D-12-92, p. 314 (1998).

[10] 島井博行, 栗田多喜夫, 梅山伸二, 田中勝, 三島健稔: “ロバスト統計に基づいた適応的な背景推定法” , 信学論 (D-II), Vol. J86-D-II, No. 6, pp. 796-806 (2003).

[11] A. Lipton, H. Fujiyoshi, and R.S. Patil: “Moving target detection and classification from real-time video”, In Proceedings of the 1998 Workshop on Applications of Computer Vision (1998).

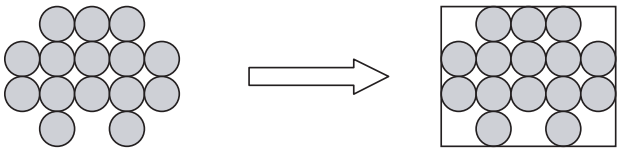
[12] F. Dellaert and R. Collins: “Fast image-based tracking by selective pixel integration”, ICCV 99 Workshop on FrameRate Vision (1999).

[13] BKP Horn, BG Schunck : “Determining optical flow”, Artificial Intelligence, 17, pp.185-203, (1981).

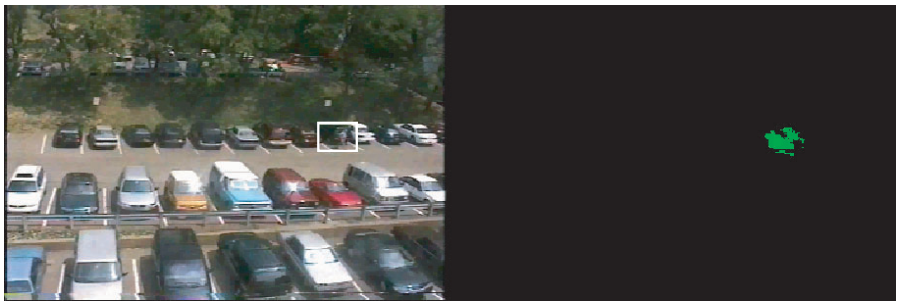
[14] B Lucas, T Kanade: “An iterative image registration technique with an application to stereo vision”, Proceedings of Imaging Understanding Workshop, pp.121-130(1981).

第3章　領域クラスタリング:Spatial Clustering

第 2 章で述べた背景差分法を用いることで，図 3.2 のように “車” や “人” など移動体が存在する領域を検出することができた. しかし，移動体を検出したといっても画素単位でのことであり，どの画素がどのオブジェクトを構成する画素であるかとはわからない．そこで，画素単位で検出された “車” や “人” など 1 つのオブジェクトを構成しているそれぞれの点を，1 つの集合としてまとめる処理である領域のセグメント化を行う.



オブジェクトを構成している点　　全部で 1 つのオブジェクト



領域のセグメント化　　背景差分結果

図 3.1: 背景差分と領域のセグメント化

図 3.2: 背景差分と領域のセグメント化

本章では，領域のセグメント化を実現するために，クラスタリングに基づく手法について述べる．また，グラフカットと Mean-Shift による領域セグメンテーション手法について述べる.

3.1　Nearest Neighbor 法による領域クラスタリング

Nearest Neighbor 法は，入力サンプルに対して学習用のデータを検索し，最も近いデータが属するクラスに入力サンプルを分類する手法である．ここでは，Nearest Neighbor 法を領域クラス

タリングに応用する手法について述べる．背景差分等の物体検出の結果，移動体領域の候補点 P_1 〜P_N (1〜N はラスタ走査の順とする) が検出されたとする．

1. 初期点 P_1 を代表点とする初期クラス C_1 に属するとし，次の注目点 $(i = 2)$ へ移動する．
2. 注目点 P_i と，既存の各クラスに属している全ての点とのユークリッド距離を求める．クラス C_j までのユークリッド距離 D_{ij} が最小であった場合，
   (a) $D_{ij} < \mathrm{Th}$ ： P_i をクラス C_j に属するとし，代表点を増やす．
   (b) $D_{ij} > \mathrm{Th}$ ： P_i を代表パターンとする新しいクラスを作る．
3. $i = N$ なら処理を終了．それ以外は i を次の注目点において (2) の処理を繰り返す．

NN 法による領域クラスタリングの流れを図 3.3 に示す．

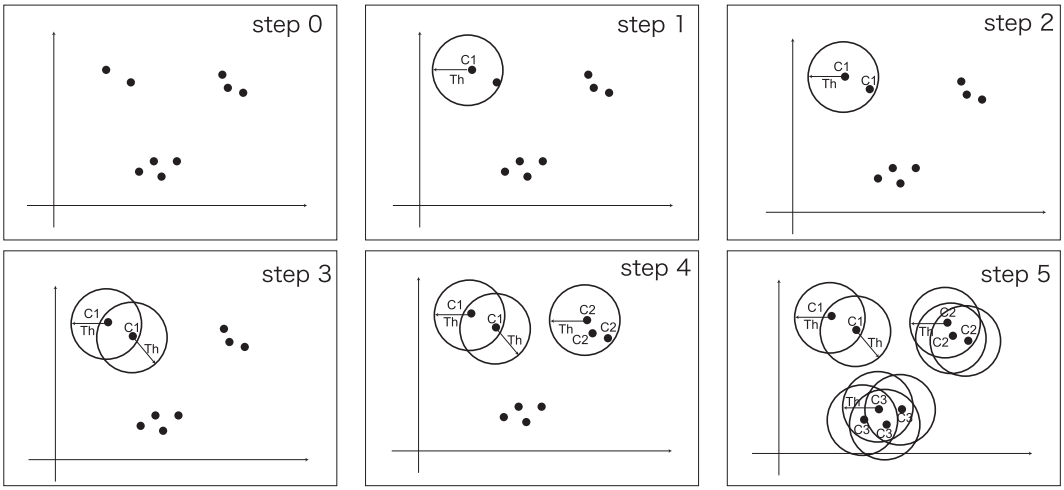


図 3.3: NN 法によるクラスタリング処理

3.2　改良型 NN 法による領域クラスタリング　【Source Code】

3.1 の NN 法による領域クラスタリングでは，突発的なスパイクノイズも検出領域の 1 つとしてクラスタリング対象となり，本来の検出領域より広くセグメントされたり，ノイズのみで構成されるクラスターを多く出力するという問題がある．そこで，条件を設定してクラスの併合や，消滅を行うことで，対象とする移動体領域のみをセグメントする．移動他領域の候補点を P_1〜P_N とする．

1. 注目点 P_1 を代表点とするクラス C_1 に属するとし，次の距離 $(i = 2)$ へ移動する．
2. 点 P_i と，既存の各クラスの代表点との X 方向と Y 方向の距離を調べる．クラス C_j までの X 方向の距離を DX_{ij}，Y 方向の距離を DY_{ij}，としたとき
   (a) $DX_{ij} < \mathrm{ThX}$ ： P_i をクラス C_j に所属させる．

(b) $DY_{ij} < \mathrm{ThY}$ ： P_i をクラス C_j に所属させる．

(c) どこにも属さない場合 ：P_i を新規クラスとして登録する．

3. $i = N$ なら処理を (4) へ．それ以外は i を 1 増やして次の注目点において (2) の処理を繰り返す．

4. 各クラスの矩形領域が接触している場合，1 つのクラスに併合する．

5. クラスの横幅，縦幅，画素数，面積比が閾値以下ならそのクラスはノイズ領域として消滅させる．

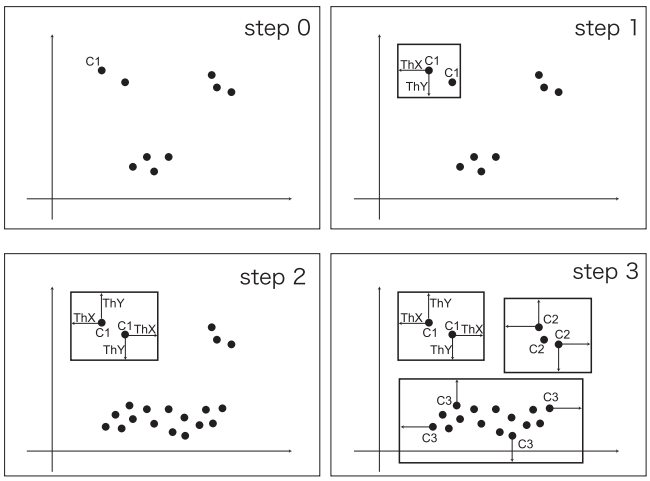


図 3.4: 改良型 NN 法によるクラスタリング処理

NN 法と改良型 NN 法によるクラスタリングの処理結果を図 3.5 に示す．図 3.5 を見て明らかなように，NN 法では背景差分のノイズまでオブジェクトとして領域をセグメント化している．

(a) 検出結果　(b) NN法　(c) 改良NN法

図 3.5: NN 法と改良型 NN 法クラスタリング

クラスタリングの限界

NN 法に基づく領域クラスタリング手法では，カメラから見て車と人の領域が重なっていたり，車と車が重なっている場合など，これらのオブジェクトを個々に区別してセグメントすることは不

可能である．

図 3.6: 車と人が重なった例

3.3　複数物体の重なりを理解するレイヤー型検出

“人が自動車から降りた” 等の人と自動車間の行動 (アクティビティ) を認識するには，人が自動車から降りた瞬間を観測する必要がある．その際には，人と自動車が画像上で重なっている場合が多く，従来の物体検出と領域クラスタリングを組み合わせた手法では，それぞれの領域を分けて検出することが難しい．

本節では，複数物体の重なりを理解する物体検出法として，ピクセル分析とリージョン分析の二つの処理からなるレイヤー型検出法について述べる．本手法は，一度静止した物体を背景上のレイヤーとして登録するため，その後，そのレイヤー上を通過する移動物体を区別して検出することが可能となる．

3.3.1　レイヤー型検出アルゴリズム

レイヤー型検出法は，ピクセル分析とリージョン分析の二つの過程からなる．ピクセル分析の目的は，各ピクセルの時間的変化を観測することにより，そのピクセルの状態を静 (stationary) もしくは 動 (transient) と判定することである．リージョン分析の目的は，ピクセルの集まりを移動物体 (moving object) もしくは静止物体 (stopped object) と判定することである．図??はピクセル分析とリージョン分析の概念を示す．あるピクセルにおいて，その輝度値の状態は初期値である背景から始まり，物体が通過するとき transient の状態へ遷移する．その物体が停止したときは，stationary の状態へ遷移する．

図 3.8 は，検出過程を示す．本検出アルゴリズムは，各ピクセルが transient であるかどうかを決定するために，ある時間幅における輝度値の振舞を観測する必要がある．ピクセル分析により transient，stationary と判定されたピクセル群は，クラスタリングにより各領域にまとめられる．リージョン分析は，領域が transient ピクセルの場合移動物体，stationary ピクセルの場合静止物体と判定する．レイヤーマネージメントは，静止物体と判定された領域を新規レイヤーとして登録する．再び静止物体が動き始めるとレイヤーの削除を行う．

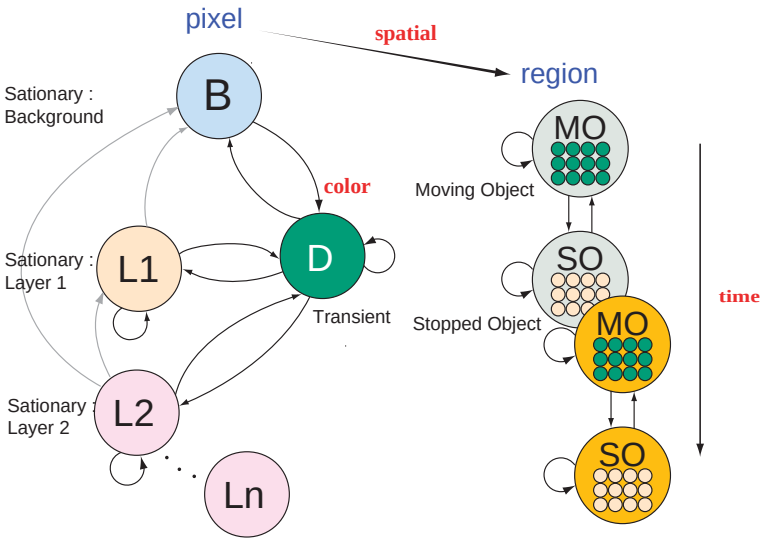


図 3.7: ピクセル分析とリージャン分析の概念

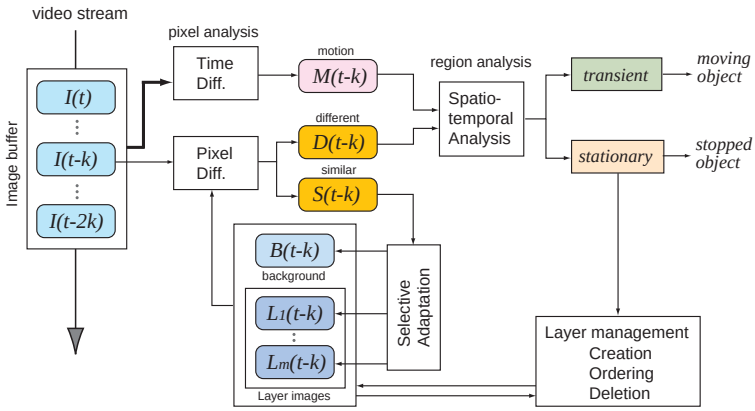


図 3.8: レイヤー型検出の過程

ピクセル分析 【Source Code】

屋外カメラで撮影した画像上のピクセルの輝度変化は，状況に応じて図 3.9 に示すように三つの特徴に大別できる．

- 物体がピクセル上を通過するとき，そのピクセルの輝度値は急激な変化を伴う．その後，一時的に不安定な状態が続き，再度急激な変化の後，背景である元の輝度値に戻る (図 3.9(a) 参照)．
- 物体がピクセル上で停止したとき，そのピクセルの輝度値は急激な変化の後，一時的に不安定な状態が続き，最終的には物体の輝度値に安定する (図 3.9(b) 参照)．
- 太陽が雲に隠れた等の環境変化が生じたとき，輝度値は図 3.9(c) に示すように緩やかに変化する．

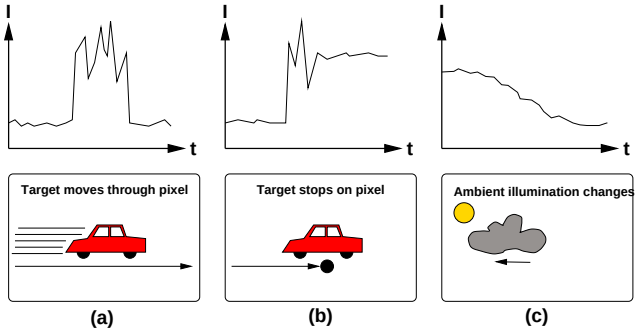


図 3.9: ピクセルの輝度変化

これらの特徴を捉えるためには，以下の二つの変化に着目する必要がある．

1. 輝度値の急激な変化
2. 不安定な状態から安定した状態への遷移

まず最初に，急激な輝度変化を捉えるモーショントリガーを考える．ここで，I_t を現在のフレームから k フレーム前の輝度値とする．輝度値の変化量 T を求めるには，t よりも前（過去）のフレームに着目する．変化量 T は次式により求められる．

$$T = \max \left\{ \left| I_t - I_{(t-j)} \right| , \forall j \in [1,5] \right\} \tag{3.1}$$

急激な輝度値の変化がピクセル上に生じたとき，T の値は大きくなる．

次に，ピクセルの安定度[1]を表す S について考える．安定状態の検出には t より後 (未来) のフレームに着目する．安定度 S は次式により求められる．

$$S = \frac{k \sum_{j=0}^{k} I_{(t+j)}{}^2 - (\sum_{j=0}^{k} I_{(t+j)})^2}{k(k-1)} \tag{3.2}$$

[1]安定度 S は分散であり，通常の分散を求める式とは異なる．通常の分散式からの式の展開は付録 B.1 を参照のこと

これはフレーム t から $t+k$ までの分散であり，安定した状態のとき S は小さくなる．

ここで，各ピクセルの状態を表す M は，背景のとき bg，transient のとき tr，stationary のとき st をとるように，以下に示すアルゴリズムにより決定される．I は，時刻 t における輝度値とする．Th_t は急激な変化を判定する閾値，Th_s は安定性を判定する閾値である．background intensity は背景画像として，予め用意しておく．

```
if ((M = st OR M = bg) AND (T > Th_t))
   M = tr
if ((M = tr) AND (S < Th_s)) {
   if (I = background intensity)
      M = bg
   else
      M = st
}
```

この操作を全ピクセルについて行う．背景と判定されたピクセルは，次式に示す IIR フィルタにより環境変化に追従するよう適応的に更新する [3]．

$$B(t) = \alpha I(t) + (1-\alpha)B(t-1) \qquad (0 < \alpha < 1) \tag{3.3}$$

α は，背景と判定されたピクセルを背景画像にどれだけ反映させて更新するかを決定する定数である．

リージョン分析

リージョン分析では，ピクセル分析によりラベル化された背景以外のピクセル領域について検討する．まず最初に，背景以外のピクセル群に対して，領域クラスタリングを行う．クラスタリングには，近隣領域の連結性を用いた手法が多いが，本研究では，ある半径 r_c 以内の距離に存在するピクセル同士を同一クラス $R(i)$ に属することとした．i はクラス番号である．このとき，各領域 $R(i)$ は以下の三状態に分けられる．

1. 領域が全て transient ピクセルから構成されている場合，その領域は moving object と判定できる．

2. 領域が全て stationary ピクセルから構成されている場合，その領域は stopped object と判定できる．その領域が登録されていなければ新規レイヤーとして登録する．

3. 領域が transient と stationary ピクセルの混合として構成されている場合，その領域内に stopped object と moving object が含まれていると考えられる．そこで，レイヤー操作によりレイヤーとして登録されていない領域を抽出し，再度領域クラスタリングを行う．クラスタリング後の領域が stationary ピクセルのみの場合は，stopped object と判定でき，新規レイヤーとして登録する．クラスタリング後の領域が transient ピクセルの場合は moving object と判定できる．

すべての領域 $R(i)$ は，以下に示すアルゴリズムにより，stopped object，moving object と判定される．ここで，L(j) はレイヤー画像，j は既に登録されているレイヤーの数とする．

```
if (R = tr) {
    R -> moving object
}
elseif (R = st) {
    %remove all pixels already assigned
    %to any layer
    R = R - (L(0) + L(1) + .. + L(j))
    %if anything is left, make a new
    %layer out of it
    if (R != 0) {
        make new layer L(j+1) = R
        R -> stopped object
    }
else {
    %R contains a mixture of t and s
    R = R - (L(0) + L(1) + .. + L(j))
    SR(i) = spatial_clustering(R)
    for (each region SR(i)) {
        if (SR(i) = tr) {
            SR(i) -> moving object
        }
        if (SR(i) = st) {
            make new layer L(j+1) = SR
            SR(i) -> stopped object
        }
    }
}
```

レイヤーマネージメントは，領域が全て stationary ピクセルにより構成されているとき，その領域を新規レイヤーとして背景上に登録する．再び，そのレイヤー領域である物体が動き出したときはそのレイヤーを削除する．

3.3.2　レイヤー型検出結果

図 3.10 に，監視カメラ映像のある一点の輝度変化とそのピクセル分析例を示す．この映像シーケンスには，次に示す人と自動車によるアクティビティが含まれている．

1. 自動車 A が監視領域に侵入し，ピクセル上で停止．

2. 別の自動車 B が監視領域に侵入し，自動車 A のカメラからみて手前側に停止.

3. 自動車 B から人が手前側に降車して，ピクセル上を通過.

4. 同じ人が自動車に戻るため，再度ピクセル上を通過.

5. 自動車 B が発進してピクセル上を通過.

6. 自動車 A が発進してピクセル上を通過.

これらの各事象に対して，図 3.9(a)，(b) に示した変動特徴と同じ軌跡を観測することができる．また，ピクセル分析が正確にピクセルの状態を transient と stationary に判定しているのが分かる．

図 3.11 は，上記の 4. の状態におけるレイヤー型検出によるリージョン分析例を示す．これは，三つの物体が重なっているにも関わらず，二つの stopped object と一つの moving object をそれぞれ検出した例である．静止物体は一時的に背景上のレイヤーとして登録されているため，その上を通過する移動物体である人の領域を区別して検出することが可能である．また，レイヤー操作により他の物体が重なっているために隠れているオクルージョン領域 (青色で示した領域) を知ることができる．

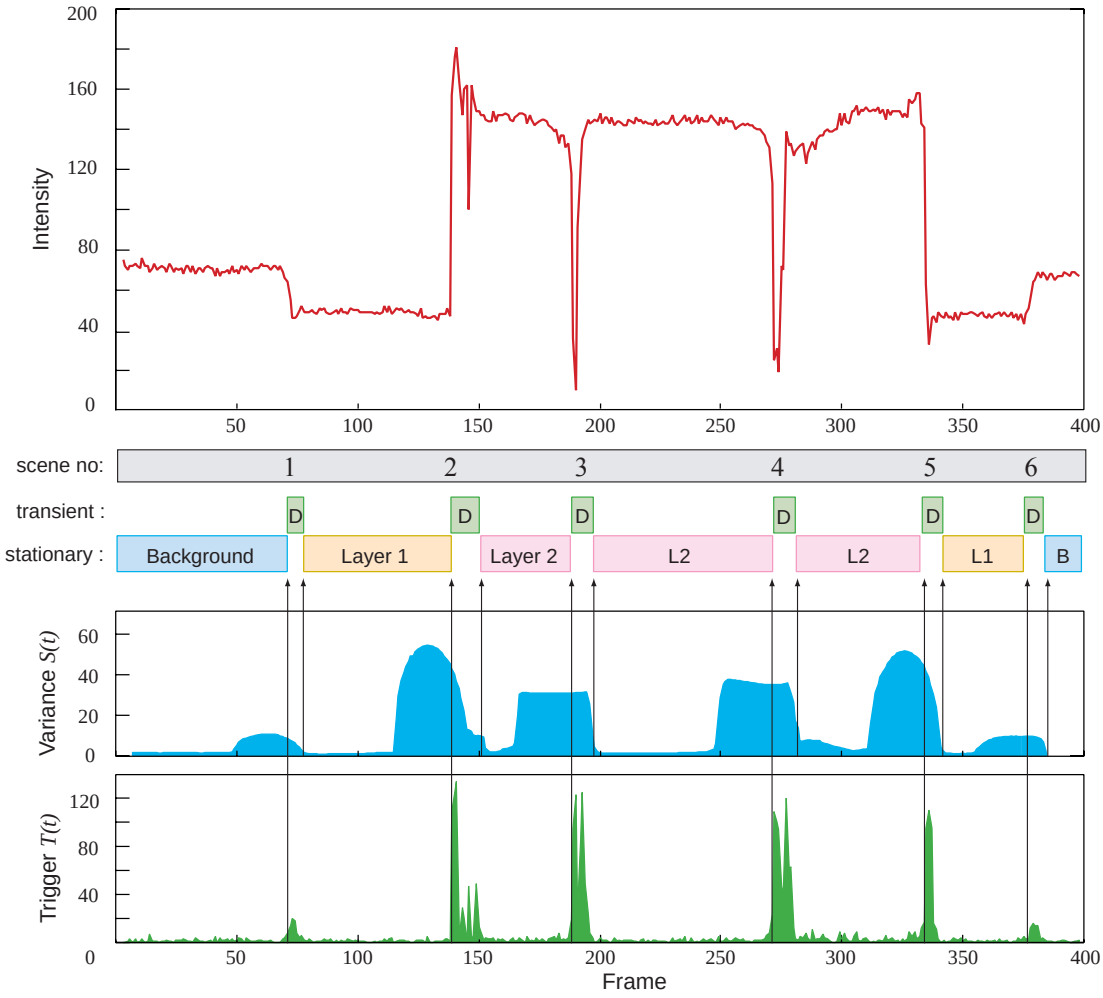


図 3.10: 輝度変化とそのピクセル分析例

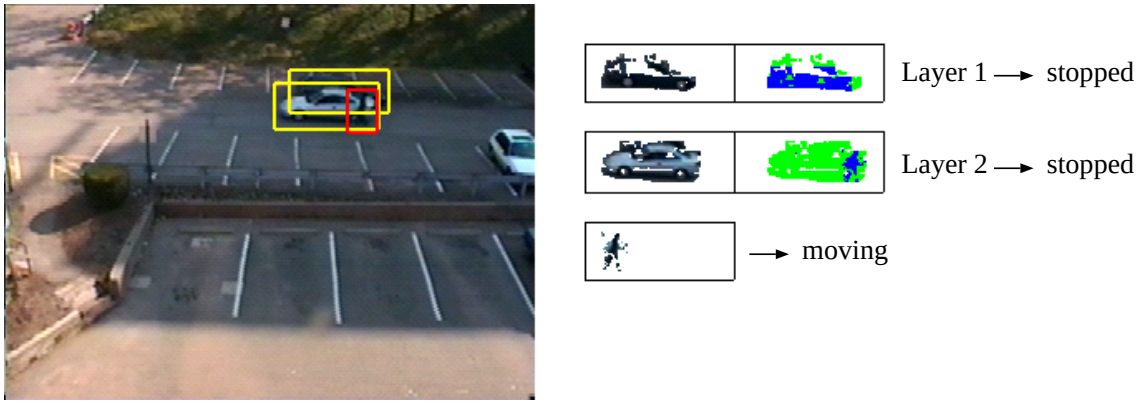


図 3.11: リージョン分析による検出例

3.4 グラフカットによる領域セグメンテーション

コンピュータビジョンの問題として，エネルギーを最小化する問題がある．例えば，画像をピクセル単位で輝度，視差，領域分割などでラベリングをする問題が上げられる．実画像に対してラベリングを行う場合，画像のノイズなどの影響により不明確な部分が存在する可能性があり，最適なラベリングを行うことは困難である．そこでエネルギー最小化問題に対して，Boykov らは minimum cut/maximum flow algorithms を利用してエネルギーを最小化する方法 [5] を提案している．彼らの手法では，問題設定に合うようにグラフを作成し，そのグラフの minimun cut を求めることにより，エネルギー関数の最小化を行う．この minimum cut は max flow algorithm を用いることにより，効率的に計算することが可能である．近年，minimum cut/maximum flow algorithms に基づいた手法が提案されており，image restoration [6, 7, 8, 9], stereo and motion [10, 11, 12, 13], image segmentation [14, 15], multi-camera reconstruction [16] などさまざまな分野に応用されている．本章では，ラベリング問題，グラフカットアルゴリズムについて述べ，そして，グラフカットアルゴリズムを用いた領域のセグメンテーション，グラフカットステレオについて述べる．

3.4.1 ラベリング問題

ラベリング問題とは，画像 P の各ピクセル p に対して，ラベル L_p をつけていく問題である．このラベリングを何に対して行うかは，求める問題により異なる．例えば，ステレオでは depth に対してラベリングをし，セグメンテーションでは各物体に対してラベリングする．図 3.12 では，輝度値に対してラベリングを行った結果である．このようなラベリング問題に対しての，解法の１つ

図 3.12: ラベリング問題

としてエネルギー関数 $E(L)$ を以下のように定義し，これを最小値を求める方法がある．

$$E(L) = \sum_{p \in P} D_p(L_p) + \sum_{(p,q) \in N} V_{(p,q)}(L_p, L_q) \tag{3.4}$$

ここで，N は近傍領域のピクセル，$D_p(L_p)$ は観測データに対してのペナルティー関数，$V_{(p,q)}(L_p, L_q)$ は近傍ピクセルとの相互関係を表した関数である．

エネルギー最小化モデル

エネルギー関数 E は，同一のもののラベリングや境界が不連続なものなど，どのような問題に対して行うかにより決定される．このエネルギー関数のモデルとして， Potts Interaction Energy

Model と Liner Interaction Energy Model の 2 つが提案されている．

Potts Interaction Energy Model

Potts Interaction Energy Model では，エネルギー関数を以下のように定義している．

$$E(I) = \sum_{p \in P} ||I_p - I_p^{\circ}|| + \sum_{(p,q) \in N} K_{(p,q)} \cdot T(I_p \neq I_q) \tag{3.5}$$

ここで，$I = \{I_p | p \in P\}$ は正解データの集合で，ここでは，ラベルの事を指す．$I^{\circ} = \{I_p^{\circ} | p \in P\}$ はノイズが混入している観測データである．この項では，入力されたデータとの差が小さくなるようなラベルが付けられるようにしたものである．また，$K(p,q)$ は近傍ピクセルとのラベルの不連続性を表したペナルティー関数，$T(I_p, I_q)$ は I_p と I_q を比較する指標関数であり，I_p と I_q が異なる場合は 1, それ以外は 0 を返す．この $K(p,q)$ は，p と q が似ている場合は大きくなる関数であるが，同じラベルが付いている場合は $T(I_p, I_q)$ が 0 となる為反映されない．そのためこの項では，近傍で異なるラベルがつけられており，かつ似ていないほど小さくなるといえる．この Potts Energy は 2 値のラベリングでは max flow を用いることにより最適解を得ることが可能であるが，多値のラベリングでは NP hard 問題となるため，最適解を得られない可能性がある．

Liner Interaction Energy Model

次に，Liner Interaction Energy Model は以下のように定義される.

$$E(I) = \sum_{p \in P} ||I_p - I_p^{\circ}|| + \sum_{(p,q) \in N} A_{(p,q)} \cdot |I_p - I_q| \tag{3.6}$$

このモデルと Potts Model では $A_{(p,q)}$ と $K_{(p,q)}$ の項が異なる．$A_{(p,q)}$ は近傍ピクセルである p と q の重要性を表す関数である．このエネルギー関数は，Potts Energy よりも多くのラベルを扱う際に有効である．

3.4.2　グラフカットアルゴリズム

定義したエネルギー関数 E に基づきグラフを作成し，min-cut/max-flow algorithm より最適解を求める．このようにして最適解を求める手法をコンピュータビジョンでは Graph Cuts Algorithm と呼ばれている．もともとの min-cut/max-flow algorithm は以下のようなネットワークに対して，どの程度の量を最大流すことができるかを求める手法である．この章では，min-cut/max-flow algorithm について述べる．

グラフ

重み付き有向グラフを $G = (E,V)$ と定義する．このとき，V はノード (node)，E はエッジ (edge) である．このグラフをコンピュータビジョンの問題で使用する場合はピクセルやボクセルな

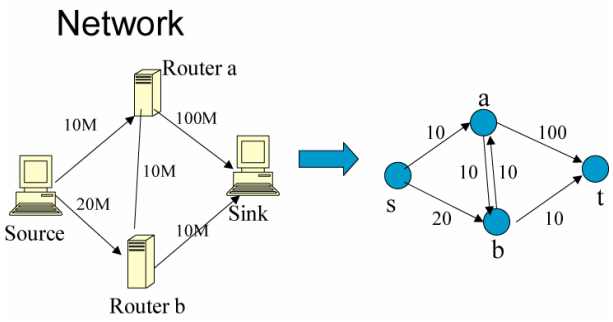


図 3.13: ネットワーク

どがノードとなる．またピクセルやボクセル以外のノードとして，ソース (source)$s \in V$ とシンク (sink)$t \in V$ と呼ばれる特殊なターミナル（ラベル）を追加する．エッジはノード間の関係を表現しており，周辺のピクセルやボクセルとの関係を表したものを n-link，各ピクセルやボクセルと s，t との関係を表したものを t-link と呼ぶ．n-link のコストは，周辺ピクセルとの連続性を表現したペナルティ関数による決定され，t-link のコストは各ピクセルがそのラベルである確率を表したペナルティ関数により決定される．n-link と t-link は，式 (3.4) の $V_{p,q}$ と D_p に相当する．作成されるグラフをネットワークグラフと呼ぶ（図 3.14(a)）．

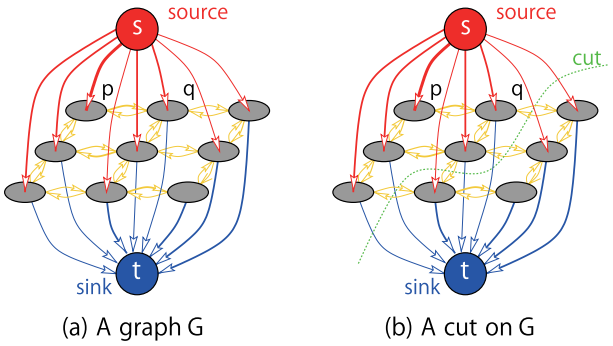


図 3.14: グラフ例

最小カットと最大フロー

ネットワークグラフを $s \in S$ と $t \in T = V - S$ となる 2 つのグラフへの分割を考える．分割する際に切断されるエッジを s-t カットと呼ぶ．その切断されるカットのうち S から T へ接続されているエッジのコストの総和を s-t カットの容量という．この s-t カットの容量が最小となる s-t カット（最小カット）を求めたい．この最小カットは，最大フロー最小カットの定理を用いることによって，ネットワークグラフの最大フローから求めることが可能である．

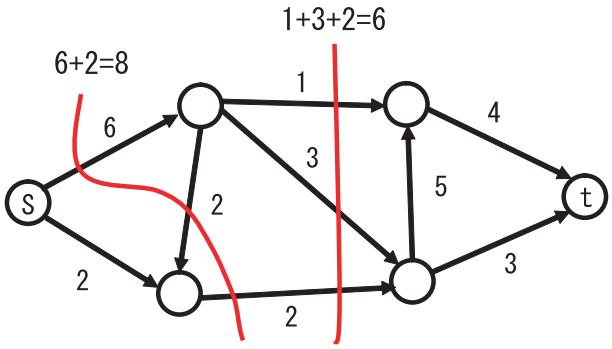


図 3.15: s-t cut

最大フローを求める手法の代表的なものとして，Ford-Fulkerson Method[17] と Push-Relabel Method[18] がある．Ford-Fulkerson Method はフローの増分路が存在しないとき，そのフローは最大であるという考えに基づいている．はじめに，フローを 0 として s から t への増分路を探索し，フローを増加させる．この作業を増分路がなくなるまで繰り返すことにより，最大フローを求める．Push-Relabel Method では，s から近傍のノードへ可能な限りフローを流し，それをすべてのノードに対して繰り返すことにより最大フローを求める．

3.4.3　領域のセグメンテーション 【Source Code】

グラフカットアルゴリズムを用いて領域のセグメンテーションを行う．作成するグラフは，Potts Interaction Energy Model に従い以下のように定義する．

$$E(L) = \lambda \cdot R(L) + B(L) \tag{3.7}$$

$$R(L) = \sum_{p \in P} R_p(L_p) \tag{3.8}$$

$$B(L) = \sum_{\{p,q\} in N} B_{\{p,q\}} \cdot \delta(L_p, Lq) \tag{3.9}$$

$$\delta(L_p, L_q) = \begin{cases} 1 & \text{if} \quad L_p \neq L_q \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

ここで，$R(L)$ は領域情報，$B(L)$ は境界情報，λ は $R(L)$ のパラメータ係数である．このエネルギー関数 $E(L)$ に基づき，グラフのエッジのコストを計算する（表 3.1）．

ここで，

$$R_p(\text{"}obj\text{"}) = -\ln \Pr(I_p|\mathcal{O}) \tag{3.11}$$

$$R_p(\text{"}bkg\text{"}) = -\ln \Pr(I_p|\mathcal{B}) \tag{3.12}$$

$$B_{\{p,q\}} \propto \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p,q)} \tag{3.13}$$

$$K = 1 + \max_{p \in P} \sum_{q:\{p,q\} \in N} B_{p,q} \tag{3.14}$$

表 3.1: エッジに与える重み

edge	weight (cost)	for
$\{p,q\}$	$B_{\{p,q\}}$	$\{p,q\} \in N$
$\{p,S\}$	$\lambda \cdot R_p(\text{"}bkg\text{"})$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p,T\}$	$\lambda \cdot R_p(\text{"}obj\text{"})$	$p \in P, p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

となる．このとき，$\mathcal{O}, \mathcal{B}$ はそれぞれ前景と背景の正解ラベルである．図 3.16 にグラフカットによるセグメンテーション例を示す．

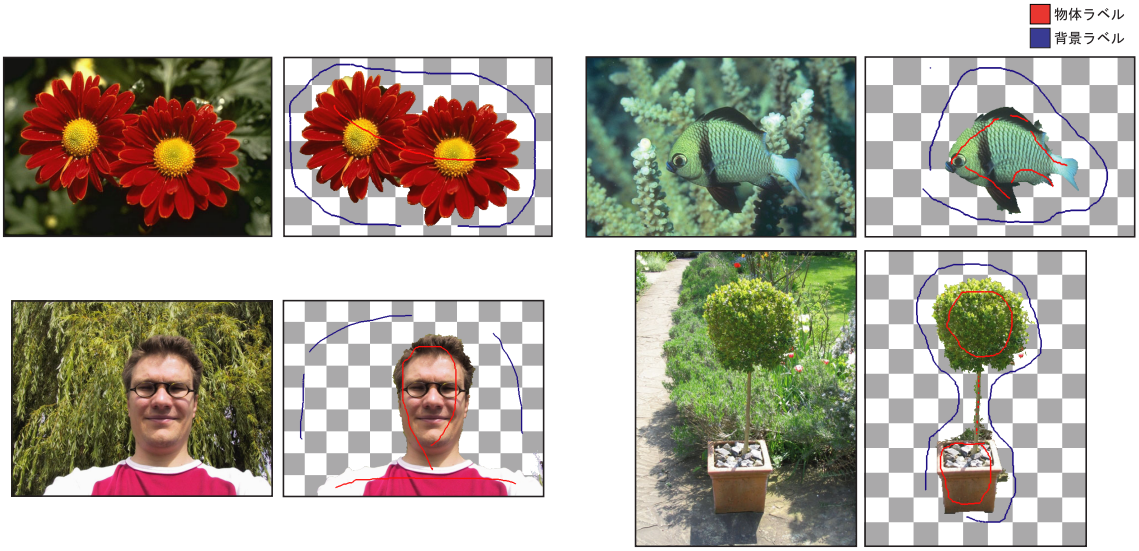


図 3.16: セグメンテーション例

3.4.4 グラフカットによるステレオ

ステレオはもっともシンプルな 3 次元復元の手法である．ステレオでは 2 台のカメラから得られる画像間のピクセルの対応により視差を求める．この画像間のピクセルの対応を求める問題は，テクスチャやカメラの設置などの要因で正確に求めることが困難である．そこでグラフカットによるステレオではピクセルの対応に対してラベリングを行うことで視差を求める．このとき，ステレオではオクルージョンが発生する為すべての対応が取れるわけではないため，グラフカットによる

ステレオでは新たなエネルギー関数を以下のように定義する．

$$E(L) = E_{data}(L) + E_{occ}(L) + E_s(L) \tag{3.15}$$

$$E_{data}(L) = \sum_{l(p,q)=1} D_{(p,q)} \tag{3.16}$$

$$E_{occ}(L) = \sum_{p \in P_1 \cup P_2} C_p \cdot T(p\ is\ occluded) \tag{3.17}$$

$$E_s(L) = \sum_{\{(p,q),(p',q')\} \in N} K_{\{(p,q),(p',q')\}} \cdot T(l_{(p,q)} \neq l_{(p',q')}) \tag{3.18}$$

このとき，L はピクセル対に対してのラベルを表す．$D_{(p,q)}$ はピクセル p と q の輝度差，C_p はピクセル p のオクルージョンを表すペナルティー関数，$K_{\{(p,q),(p',q')\}}$ は周辺のピクセルとの滑らかさを表している．

例として図 3.17 の 1 列の画像の対応について考える．はじめに，実線で描かれているような適当なピクセル対のラベル L^0 をつける．これに新たなラベルの追加を行う．新たなラベルとして，破線のようなピクセル対 L^α を見つけ，図 3.17 左のようなグラフを作成する．このときのコストは，表 3.2 に基づいて決定される． a はあるピクセル対であり，$D_{occ}(a)$ はピクセル対のオクルー

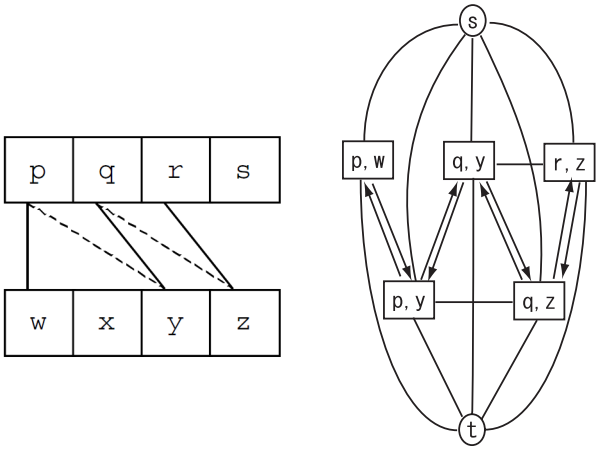


図 3.17: ピクセル対応とそのグラフ

ジョンのペナルティ関数，$D(a)$ はピクセル対の輝度差，$D_{smooth}(a)$ はピクセル対同士の滑らかさを表す関数である．また，$K_{\{(p,q),(p',q')\}}$ は以下の式が用いられる．

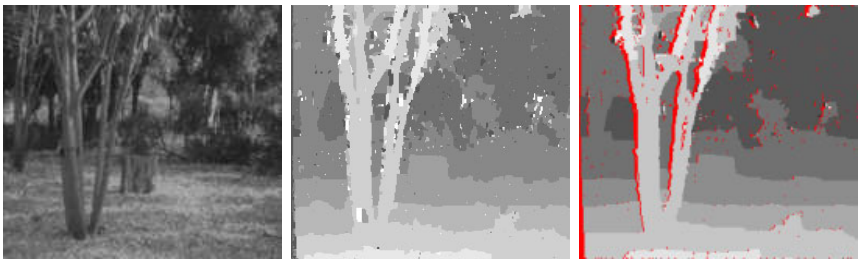
$$K_{\{(p,q),(p',q')\}} = \begin{cases} \lambda & \textbf{if } \max(|I_p - I_{p'}|, |I_q - I_{q'}|) < 8 \\ 3\lambda & \textbf{otherwise} \end{cases} \tag{3.19}$$

作成したグラフの最小カットを求めることにより，追加したラベルがどこにつけられるかが決定される．この作業を繰り返し行うことにより，多値のラベリングを行うことが可能になる．以下に，ステレオの例を示す．

表 3.2: エッジに与える重み

edge	weight(cost)	for
(s, a)	$D_{occ}(a)$	$a \in L^0$
(a, t)	$D_{occ}(a)$	$a \in L^\alpha$
(a, t)	$D(a) + D_{smooth}(a)$	$a \in L^0$
(s, a)	$D(a)$	$a \in L^\alpha$
$(a1, a2), (a2, a1)$	$K_{a1,a2}$	$\{a1, a2\} \in N, a1, a2 \in \tilde{L}$
$(a1, a2)$	∞	$a1 \in L^0, a2 \in L^\alpha$
$(a2, a1)$	C_p	$a1 \in L^0, a2 \in L^\alpha$

(a) Left image of Head pair (b) Potts model stereo (c) Stereo with occlusions
Disparity maps obtained for the Head pair



(d) Left image of Tree pair (e) Potts model stereo (f) Stereo with occlusions
Disparity maps obtained for the Tree pair

(g) Random pair (h) Potts model stereo (i) Stereo with occlusions
Disparity maps obtained for the Random pair

図 3.18: ステレオの結果例 (文献 [5] より)

3.5　Mean-Shift クラスタリング

多次元データの最頻値 (モード) の解析方法の 1 つに Mean-Shift Analysis が提案されてる．従来，多次元データの最頻値を求める際には，全てのデータの 1 次微分値を求める必要があった．Mean-Shift では， 1 次微分値を直接求めることなく，周辺のサンプル点の勾配値を計算することでデータの最頻値を求めることができる．また， Mean-Shift は，解析する特徴空間を変えることで，多くの分野に応用することができる．

3.5.1　Mean-Shift の理論

Mean-Shift とは，図 3.19 の多次元データの最頻値を探索する最頻値解析法であり， 1975 年に Fukunaga ら [19] により提案された (当時，“Mean-Shift“という名は付いていなかった)．その後，1995 年に Cheng[22] により “Mean-Shift“という名で，最頻値解析法，クラスタリング手法の 1 つとして一般化された．以降， Mean-Shift によるフィルタリング（平滑化），セグメンテーション [21, 22]，トラッキング [23, 3] のように， Mean-Shift を応用した様々な手法が提案されている．

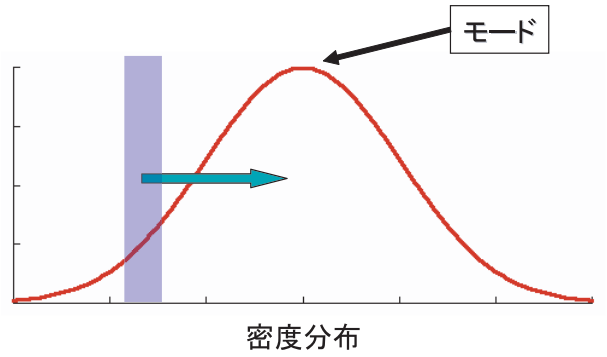


図 3.19: 多次元データの最頻値探索

3.5.2　核密度推定 (Kernel Density Estimation)

入力される n 個の d 次元データを $\{\mathbf{x}_i\}_{i=1,\ldots,n}$ とする．これらの多次元データの密度を計算する方法として， 核密度推定法 (Kernel Density Estimation Method) がある．これは， Parzen 推定とも呼ばれている．この 核密度推定法の一般式は以下のように定義されている．

$$\hat{f}_{h,K}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{3.20}$$

式 (3.20) を用いることにより，カーネル関数 K に対する $\mathbf{x}$ の確率密度 $\hat{f}_{h,K}(\mathbf{x})$ を求めることができる．このとき，h はバンド幅のパラメータを意味しており，核密度推定法によって得られる推定

結果は，バンド幅 h に依存する．ここで，カーネル関数 $K(\mathbf{x})$ が左右に対称ならば，式 (3.20) は以下のように表すことができる．

$$K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2) \tag{3.21}$$

このとき，$c_{k,d}$ は正規化定数である．ここで，カーネル関数 $K(\mathbf{x})$ は，以下の Normal Kernel が一般的に用いられる．

$$k_N(x) = \exp\left(-\frac{1}{2}x\right) \qquad x \geq 0 \tag{3.22}$$

$$K_N(\mathbf{x}) = (2\pi)^{-d/2}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \tag{3.23}$$

また，以下の Epanechnikov Kernel なども用いられる．

$$k_E(x) = \begin{cases} 1-x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \tag{3.24}$$

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

よって，式 (3.20) は以下のように変形することができる．

$$\hat{f}_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.26}$$

3.5.3 密度勾配推定 (Density Gradient Estimation)

多次元データの最頻値を求めるために，確率密度 $\hat{f}(\mathbf{x})$ の最大値を求める．求めたい最頻値は，確率密度 $\hat{f}(\mathbf{x})$ の最大値であるため，その勾配 $\nabla f(\mathbf{x})$ は 0 となる．よって，式 (3.26) から，密度勾配 $\hat{\nabla} f_{h,K}(\mathbf{x})$ は以下の式のようになる (付録 A 参照)．

$$\hat{\nabla} f_{h,K}(\mathbf{x}) \equiv \nabla \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}-\mathbf{x}_i)k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.27}$$

ここで，カーネル関数 $k(x)$ の微分である $k'(x)$ を以下のように定義する．

$$g(x) = -k'(x) \tag{3.28}$$

$$G(\mathbf{x}) = c_{g,d}g\left(\|\mathbf{x}\|^2\right) \tag{3.29}$$

ここで，$c_{g,d}$ は正規化定数である．この式 (3.28) を式 (3.27) に代入する (付録 B 参照)．

$$\begin{aligned} \hat{\nabla} f_{h,K}(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right] \end{aligned} \tag{3.30}$$

式 (3.26) より，カーネル関数 G に対する $\mathbf{x}$ の確率密度 $\hat{f}_{h,G}(\mathbf{x})$ は以下の式のようになる．

$$\hat{f}_{h,G}(\mathbf{x}) = \frac{c_{g,d}}{nh^d}\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.31}$$

ここで， Mean-Shift Vector $\mathbf{m}_{h,G}(\mathbf{x})$ を以下のように定義する．

$$\mathbf{m}_{h,G}(\mathbf{x}) = \frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \tag{3.32}$$

この式 (3.31), (3.32) を式 (3.30) に代入する．

$$\hat{\nabla} f_{h,K}(\mathbf{x}) = \hat{f}_{h,G}(\mathbf{x})\frac{2c_{k,d}}{h^2 c_{g,d}}\mathbf{m}_{h,G}(\mathbf{x}) \tag{3.33}$$

ここから，以下のように変形する．

$$\mathbf{m}_{h,G}(\mathbf{x}) = \frac{1}{2}h^2 c\frac{\hat{\nabla} f_{h,K}(\mathbf{x})}{\hat{f}_{h,G}(\mathbf{x})} \tag{3.34}$$

ここで，c は正規化定数である．式 (3.34) より，カーネル関数 G から計算される Mean-Shift Vector $\mathbf{m}_{h,G}(\mathbf{x})$ は，カーネル関数 K から得られる密度勾配に比例するため， Mean-Shift Vector は常に極大値の方向を向く．よって， 注目点を Mean-Shift Vector の方向へ繰り返し移動させることで極大値を求めることができる．故に， Mean-Shift Vector を計算することよって，多次元データの最頻値を求めることが可能となる．

ここで，式 (3.32) により定義された Mean-Shift Vector $\mathbf{m}_{h,G}(\mathbf{x})$ は，図 3.20 のような関係となる．

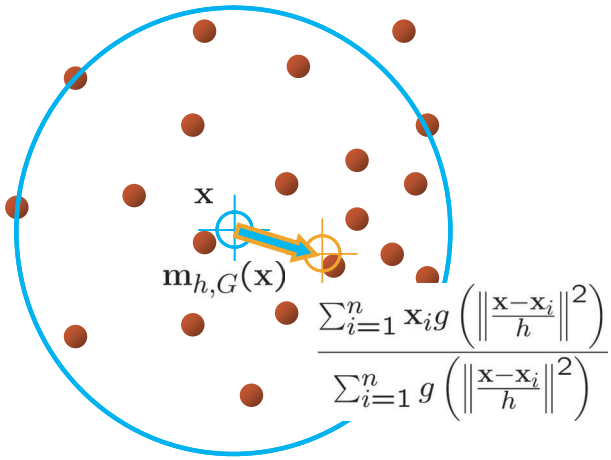


図 3.20: 式 (3.32) の関係

故に，注目点 $\{\mathbf{y}_j\}_{j=1,2,\ldots}$ を以下のように定義することができる．

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right\|^2\right)} \tag{3.35}$$

$\mathbf{y}_j$ から算出される $\mathbf{y}_{j+1}$ は， Mean-Shift Vector の特性により，常に極大方向へ移動する．これを繰り返し行うことにより，多次元データの最頻値を探索することができる．

図 3.21 に， Mean-Shift による最頻値探索の結果を示す．

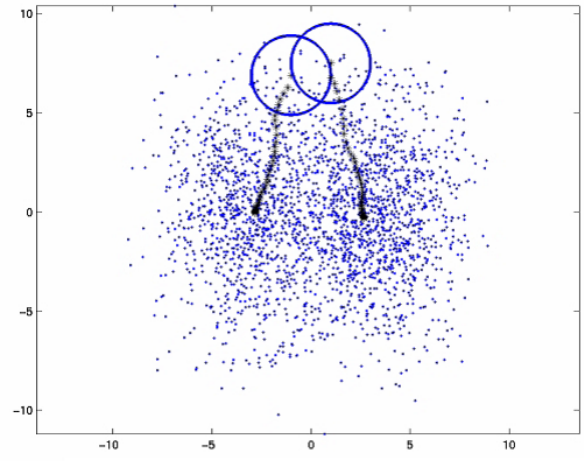


図 3.21: Mean-Shift による最頻値探索 (文献 [22] より)

3.5.4 Mean-Shift クラスタリング

Mean-Shift クラスタリング [19, 22] はクラスタ数を指定することなく，クラスタリングすることができるという特徴がある (与えるパラメータはバンド幅 h)．

入力データを $\{\mathbf{x}_i\}_{i=1,\ldots,n}$，注目点を $\{\mathbf{y}_j\}_{i=1,2,\ldots}$，クラスタリング結果を $\{\mathbf{z}_i\}_{i=1,\ldots,n}$ としたとき， Mean-Shift クラスタリングは以下の処理手順により行う．

1. $\mathbf{y}_1 = \mathbf{x}_i$
2. 式 (3.36) より，注目点 $\mathbf{y}_{j+1}$ を計算
3. 注目点 $\mathbf{y}_{j+1}$ の移動量が 0 に収束するまで 2.〜3. を繰り返す
4. $\mathbf{z}_i = \mathbf{y}_{j+1}$ とし， $i = n$ になるまで 1.〜4. を繰り返す

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right\|^2\right)} \tag{3.36}$$

図 3.22 に，Mean-Shift クラスタリングの結果を示す．

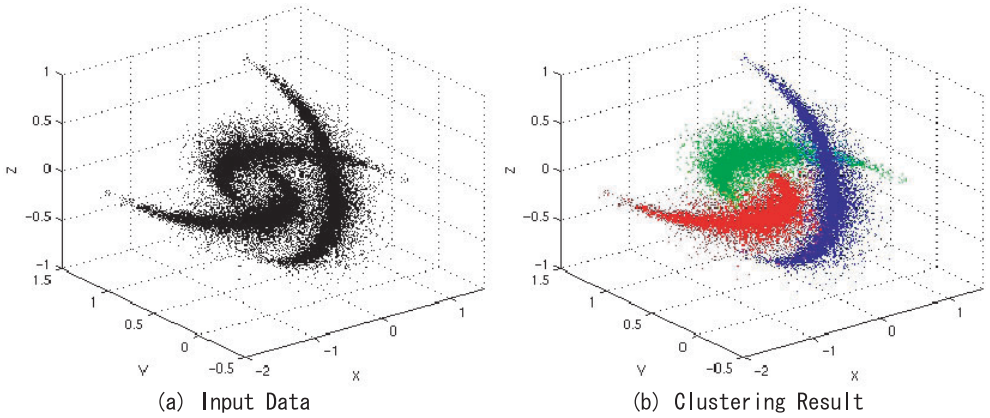


図 3.22: Mean-Shift クラスタリング (文献 [22] より)

3.5.5　Mean-Shift フィルタリング 【Source Code】

一般的な平滑化手法として Gaussian Filter がある．Gaussian Filter での平滑化では，画像全体が平滑化されるため，エッジ情報が減少してしまう．Mean-Shift フィルタリング [21, 22] による平滑化手法は，画像のエッジ情報を保持した平滑化を行うことが可能となる．エッジ情報を保持した平滑化方法には， Bilateral Filter がある．Bilateral Filter では，繰り返し処理により平滑化度合いを変化させるが， Mean-Shift フィルタリングは 1 回のフィルタリングで Bilateral Filter と類似した結果を得ることができる．

入力画像を $\{\mathbf{x}_i\}_{i=1,\ldots,n}$，注目点を $\{\mathbf{y}_j\}_{i=1,2,\ldots}$，フィルタリング結果を $\{\mathbf{z}_i\}_{i=1,\ldots,n}$ とする．ここで，$\mathbf{x}_i, \mathbf{z}_i$ はそれぞれ空間情報と色情報の情報を持っている．このとき， Mean-Shift フィルタリングは以下の処理手順により行う．

1. $\mathbf{y}_1 = \mathbf{x}_i$
2. 式 (3.36) より，注目点 $\mathbf{y}_{j+1}$ を計算
3. 注目点 $\mathbf{y}_{j+1}$ の移動量が 0 に収束するまで 2.〜3. を繰り返す
4. $\mathbf{z}_i = (\mathbf{x}_i^s, \mathbf{y}_i^r)$ とし， $i = n$ になるまで 1.〜4. を繰り返す

ここで，s は空間情報，r は色情報を意味する．

Mean-Shift を行う特徴空間は，空間情報の 2 次元と色情報の p 次元である．このとき色情報をカラーで扱う場合，L*a*b* または L*u*v* 色空間を用いるのが良い．これは， Mean-Shift を行うにはユークリッドな空間が必要であることが挙げられる．一般的な RGB 色は非線形空間で構成されているため， Mean-Shift に適していない．一方，L*a*b* または L*u*v* は色の心理的距離に基づいたユークリッドな空間となっているため， Mean-Shift に適している．

Mean-Shift はすべての次元を同等に扱うため，性質が違う空間情報と色情報を $d = 2 + p$ 次元として扱うのは不適切である．そこで，以下のような，空間と色情報を別々にしたカーネル関数 K_{h_s,h_r} を用いる必要がある．

$$K_{h_s,h_r}(\mathbf{x}) = \frac{C}{h_s^2 h_r^p} k\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|^2\right) \tag{3.37}$$

ここで，h_s は空間情報のバンド幅，h_r は色情報のバンド幅，C は正規化定数である．図 3.23 に，空間情報のバンド幅 h_s，色情報のバンド幅 h_r のパラメータを変化させたときの Mean-Shift フィルタリングの結果を示す．赤色の線は注目点の軌跡，青色の点は収束点である．図 3.23 から，h_s を大きくすると，画像座標上での探索範囲が大きくなることで，小さな領域が大きな領域に統合されるため，エッジ以外でより平滑化されているのがわかる．一方，h_r を大きくした場合，色空間での探索範囲が大きくなるため，輝度差が小さなエッジは平滑化される．

3.5.6 Mean-Shift セグメンテーション

Mean-Shift フィルタリングの発展手法として， Mean-Shift セグメンテーション [21, 22] がある．Mean-Shift セグメンテーションは，各ピクセルに対して Mean-Shift クラスタリングを行い，そのクラスタを小領域とすることで画像のセグメンテーションを行う．

入力画像を $\{\mathbf{x}_i\}_{i=1,\ldots,n}$，フィルタリング結果を $\{\mathbf{z}_i\}_{i=1,\ldots,n}$，各ピクセルのラベルを $\{\mathbf{L}_i\}_{i=1,\ldots,n}$ としたとき， Mean-Shift セグメンテーションは以下の処理手順により行う．

1. Mean-Shift フィルタリング処理を行う
2. フィルタリング結果から，$\mathbf{z}_i$ をクラスタ $\{C_p\}_{p=1,\ldots,m}$ にクラスタリング (m はクラスタ数)
3. $i = n$ となるまで，$L_i = \{p|\mathbf{z}_i \in C_p\}$
4. 空間領域において，M pixel 以下の領域を削除 (近傍クラスタに吸収させる)

ここで， Mean-Shift セグメンテーションでは，パラメータとして空間情報のバンド幅 h_s，色情報のバンド幅 h_r，削除領域範囲 M を与える必要がある．図 3.24，3.25 に， Mean-Shift セグメンテーションの結果を示す．

(a) 入力画像

(b) (hs, hr) = (10, 6)

(c) (hs, hr) = (30, 6)

(d) (hs, hr) = (10, 30)

(e) (hs, hr) = (30, 30)

図 3.23: Mean-Shift フィルタリング例

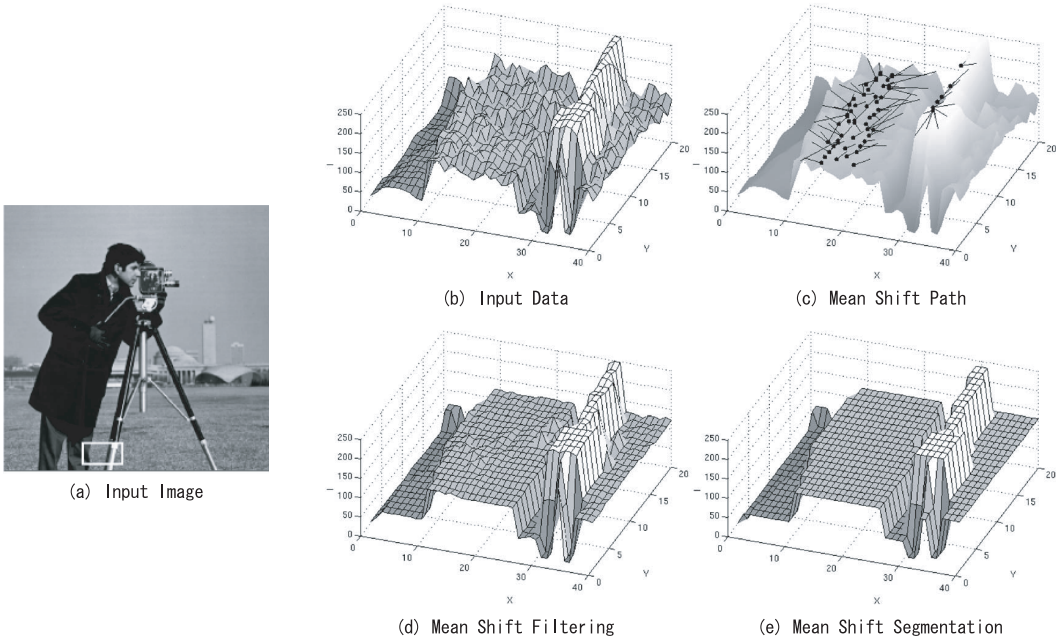


図 3.24: Mean-Shift セグメンテーション (文献 [22] より)

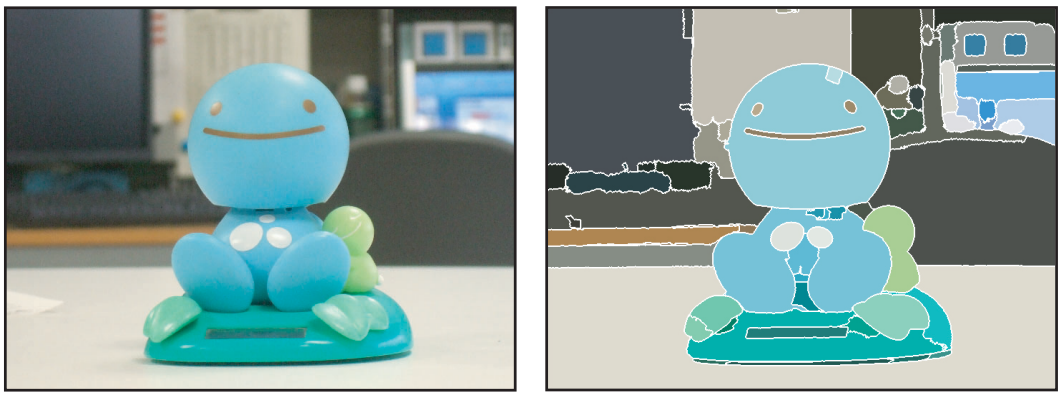


図 3.25: Mean-Shift セグメンテーション例

参考文献

[1] “ディジタル画像処理”, 財団法人 画像情報教育振興協会 (CG-ARTS 協会).

[2] 藤吉弘亘, 金出武雄: “複数物体の重なりを理解するレイヤー型検出法”, 第 7 回画像センシングシンポジウム論文集, pp. 369–374 (Jun. 2001-6).

[3] A. Lipton, H. Fujiyoshi, and R.S. Patil: “Moving target detection and classification from real-time video”, In *Proceedings of the 1998 Workshop on Applications of Computer Vision*, pp. 8–14 (Oct. 1998).

[4] H. Fujiyoshi, and T. Kanade: “Layered Detection for Multiple Overlapping Objects”, In IEICP TRANSACTIONS on Infromation and Systems, vol. E87-D, No.12, pp.2821-2827(December 2004).

[5] Y. Boykov, V. Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”, PAMI, vol. 26, no. 9, pp. 1124-1137, Sept. 2004.

[6] D. Greig, B. Porteous, A. Seheult, “Exact Maximum A Posteriori Estimation for Binary Images”, J. Royal Statistical Soc., Series B, vol. 51, no. 2, pp. 271-279, 1989.

[7] Y. Boykov, O. Veksler, R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts”, Proc. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp. 1222-123, 2001.

[8] Y. Boykov, O. Veksler, R. Zabih, “Markov Random Fields with Efficient Approximations”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 648-655, 1998.

[9] H.Ishikawa, D. Geiger, “Segmentation by Grouping Junctions”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 125-131, 1998.

[10] H. Ishikawa, D. Geiger, R. Zabih, “Occlusions, Discontinuities, and Epipolar Lines in Stereo”, Proc. Int ’l Conf. Computer Vision, pp. 1033-1040, 2003.

[11] J. Kim, V. Kolmogorov, “Visual Correspondence Using Energy Minimization and Mutual Information”, Proc. Int ’l Conf. Computer Vision, pp. 508-515, 2001.

[12] V. Kolmogorov, R. Zabih, “Visual Correspondence with Occlusions Using Graph Cuts”, PhD thesis, Stanford Univ., Dec. 2002.

[13] M.H. Lin, “Surfaces with Occlusions from Layered Stereo”, Int ’l J. Computer Vision, vol. 1,no. 2, pp. 1-15, 1999.

[14] Y. Boykov, M. Jolly, “Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images” , ICCV, vol. I, pp. 105-112, 2001

[15] Y. Boykov, V. Kolmogorov, “Computing Geodesics and Minimal Surfaces via Graph Cuts”, Proc. European Conf. Computer Vision, pp. 232-248, 1998.

[16] V. Kolmogorov, R. Zabih, “Multi-Camera Scene Reconstruction via Graph Cuts”, Proc. European Conf. Computer Vision, vol. 3, pp. 82-96, 2002.

[17] L. Ford, D. Fulkerson, “Flow in Networks.”, Princeton University Press, 1962.

[18] A. V. Goldberg, R. E. Tarjan, “A new approach to the maximum flow problem”, Journal of the Association for Computing Machinery, vol 35, no. 4, pp 921-940, Oct 1988.

[19] K. Fukunaga and L. D. Hostetler, “The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition”, IEEE, Trans. Information Theory, Vol.21, pp.32-40, 1975.

[20] Y. Cheng, “Mean Shift, Mode Seeking, and Clustering”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.17, pp.790-799, 1995.

[21] D. Comaniciu and P. Meer, “Mean Shift Analysis and Applications“, Proceedings of the International Conference on Computer Vision, Vol.2, pp.1197-1203, 1999.

[22] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis“, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.24, No.5, pp.603-619, 2002.

[23] D. Comaniciu and P. Meer, “Real-Time Tracking of Non-Rigid Objects using Mean Shift“, IEEE Conference on Computer Vision and Pattern Recognition, Vol.2, pp.142-149, 2000.

[24] R. Collins, “Mean-shift Blob Tracking through Scale Space“, IEEE Conference on Computer Vision and Pattern Recognition, Vol.2, pp.234-240, 2003.

第4章　物体追跡:Object Tracking

本章では，物体の見えであるアピアランス特徴を用いて画像全体から対象物を探索するテンプレートマッチング，計算コストの高いテンプレートマッチングの高速化手法として，残差逐次検定法，疎密探索法，アクティブ探索法について，局所領域のみを探索する Mean-shift と Particle Filter 用いたトラッキング手法について述べる．次に，特徴点追跡として，一般的な手法である LK 法及び，その発展型である KLT 法と SIFT を用いた手法について述べる．

4.1　物体追跡とは

物体追跡とは，与えられた動画像から，指定した対象が画像上で，どのように移動するかを推定する問題である．物体追跡は，通常，図 4.1(a) のように，被写体である対象の初期位置 $\boldsymbol{x}_0$ が，次フレームにおいてどのように変化したかという移動量 $\Delta\boldsymbol{x}$ を順方向に探索することである．この処理を毎フレーム繰り返すことで，現在のフレームにおける対象物体の位置を知ることができる．このような処理を逐次処理による物体追跡と呼ぶ．逐次処理による物体追跡は，各フレームごとの処理が高速であれば，リアルタイムに対象物体の追跡処理が可能となる．そのため，このような逐次処理による物体追跡は，デジタルカメラやデジタルビデオのオートフォーカスや自動カメラワークといった撮影時の処理に利用することができる．

一方，図 4.1(b) のように，動画像を時空間として捉え，対象領域をボリュームとして抽出する追跡アプローチも考えられる [20][21]．このような処理を一括処理による物体追跡と呼ぶ．一括処理による物体追跡では，あるフレームで追跡対象位置 $\boldsymbol{x}_0$ を指定し，追跡対象が動画像中でどのように移動したかを，動画像全体を用いて順方向と逆方向から最適化することで移動量 $\Delta\boldsymbol{x}_0, \ldots, \Delta\boldsymbol{x}_n$ を求める．そのため，逐次処理よりもノイズやオクルージョンに対して頑健に追跡することができる．一括処理による物体追跡は複数の時系列画像からなるフレームバッファに対して行うため，画像合成や動画像編集等の撮影後の処理として利用される．被写体の追跡をするには，追跡対象領域を指定する必要がある．ここでは，ユーザが指定した矩形領域から得られるカラーヒストグラムを特徴量として追跡する手法について考える．領域ベースの追跡には図 4.2 に示すように，全探索による手法と局所探索による手法がある．

全探索の追跡手法として代表的なものは，テンプレートマッチングである．テンプレートマッチングは探索領域内をラスタスキャンしてマッチング位置を求める．そのため，回転やスケール変化などを考慮した場合，非常に計算量が多くなるという問題がある．この問題を解決するアプローチとして，アクティブ探索法 [7] が提案されている．

一方，探索領域内の局所探索を効率的かつ高速に行う手法として，Mean Shift が提案されている．また，全探索と局所探索の中間的な手法として，確率的に物体位置を推定する Particle Filter[10] が提案されている．

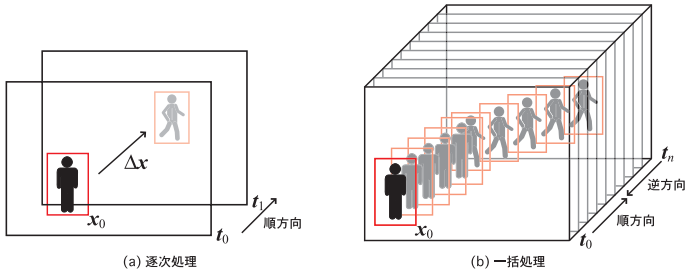


図 4.1: 物体追跡の種類

これらの追跡手法では，特徴量にカラーヒストグラムを用いることが一般的である．カラーヒストグラムは歩行中の人のように形状が変化しても，その色分布は変化しないため，追跡に用いると非剛体や物体の見えの変化に対して頑健で高速な追跡が可能となる．以下に，探索領域全体を効率的に探索するアクティブ探索法，近傍領域を効率的に探索する Mean Shift と，確率的に次の物体位置を推定する Particle Filter について述べる．

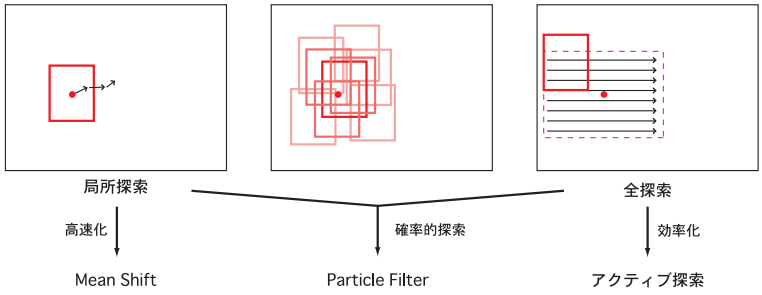


図 4.2: 領域ベースの追跡手法

4.2　テンプレートマッチング

テンプレートマッチングでは，予め標準パターンをテンプレートとして用意しておき，このテンプレートを用いて入力画像とのマッチングを行う．テンプレートマッチングを行うことで，用意したテンプレートと同じパターンが入力画像中のどこに存在するかを知ることができる．図 4.3 に示すように，テンプレートを画像全体に対してラスタ操作し，それぞれの位置でテンプレートと入力画像の類似度を求める．

4.2.1　類似度・相違度の計算

テンプレートマッチングでは，2 つの画像間の類似度 (similarity measure) または相違度 (dissimilarity measure) を調べるために，SSD(Sum of Square Difference) や SAD(Sum of Absolute

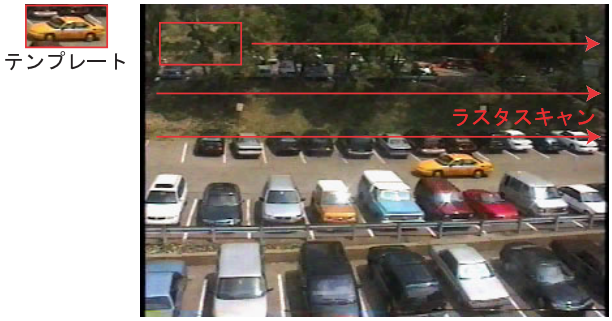


図 4.3: テンプレートマッチングの例

Difference) を用いる．テンプレートの大きさを $M \times N$ とし，テンプレートの位置 (i, j) における画素値を $T(i, j)$，テンプレートと重ね合わせた対象画像の画素値を $I(i, j)$ とした場合，SSD，SAD はそれぞれ次式により表される．

- SSD(差の 2 乗和)

$$R_{SSD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (I(i, j) - T(i, j))^2 \tag{4.1}$$

- SAD(差の絶対値和)

$$R_{SAD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} |I(i, j) - T(i, j)| \tag{4.2}$$

SSD や SAD は画像がテンプレートと完全に一致したときに値 0 をとり，相違が大きいほど大きな値をとるため，相違度を表している．

SSD や SAD の他に正規化相互相関 NCC(normalized cross-corelation) を用いることもある．

- NCC(正規化相互相関)

$$R_{NCC} = \frac{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (I(i, j)T(i, j))}{\sqrt{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} I(i, j)^2 \times \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} T(i, j)^2}} \tag{4.3}$$

SAD や SSD の相違度，NCC の類似度は，次のように理解することができる．図 4.4 に示すように，テンプレート $T(i, j)$ と対象画像 $I(i, j)$ を $M \times N$ 要素のベクトル $\boldsymbol{T}$ と $\boldsymbol{I}$ と考えると，$M \times N$

次元空間内において，SAD はベクトル $\boldsymbol{T}$ と $\boldsymbol{I}$ の先端間の市街地距離，SSD は先端間のユークリッド距離の 2 乗，NCC はベクトルのなす角の余弦 (cos) となっている．ベクトル先端間の距離は小さいほど類似性が高く，ベクトルのなす角の余弦は 1 に近いほど類似性が高い．

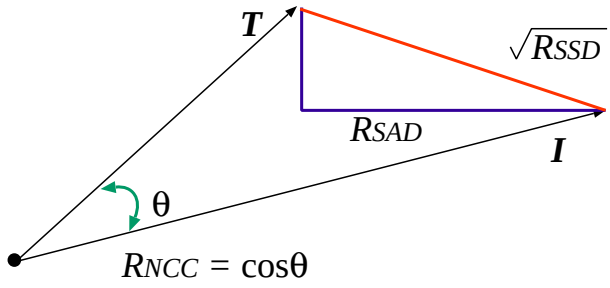


図 4.4: SSD，SAD，NCC の意味

また，テンプレートと画像を画素値に関する統計値と考えて，次の相互相関関数を類似度として利用することもある．

$$R_{ZNCC} = \frac{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1}((I(i,j)-\bar{I})(T(i,j)-\bar{T}))}{\sqrt{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1}(I(i,j)-\bar{I})^2 \times \sum_{j=0}^{N-1}\sum_{i=0}^{M-1}(T(i,j)-\bar{T})^2}} \tag{4.4}$$

$$\bar{I} = \frac{1}{MN}\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} I(i,j), \qquad \bar{T} = \frac{1}{MN}\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} T(i,j) \tag{4.5}$$

相互相関関数は正規化されており，テンプレートと画像が完全に一致した場合，最大値の 1 をとる．

4.3　更新テンプレートによるトラッキング 【Source Code】

初期テンプレートは，検出した領域とし，次フレームの画像中にテンプレートと類似するパターンを保持する．時間変化とともに移動体の形状は変化するためテンプレートを更新する必要がある．IIR フィルタによるテンプレートの更新例を図 4.5 に示す．フィルタによるテンプレートの更新を次式に示す [1].

$$T_t(i,j) = \alpha T_t(i,j) + (1-\alpha)T_{t-1}(i,j) \qquad (0 < \alpha < 1) \tag{4.6}$$

4.4　テンプレートマッチングの高速化

探索範囲に対してすべての位置で類似度を計算する全探索では，探索範囲が大きいと多くの計算時間が必要になる．そのため，探索を効率化する手法が提案されている．

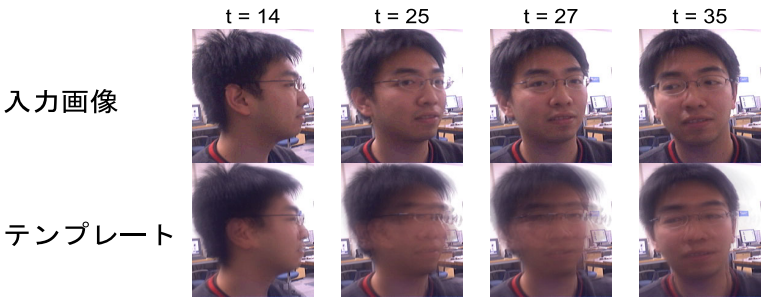


図 4.5: IIR フィルタによるテンプレートの更新

4.4.1 残差逐次検定法

相違度を計算するとき，領域内の計算結果を可算している．このとき，入力画像とテンプレート画像が目標検出位置ではない場所で重ね合わせをすると，残差が急激に増大する．そこで，可算の途中で残差があるしきい値を超えたら検出位置ではないと判断し，次の位置での計算に移る方法を残差逐次検定法 (SSDA: Sequential Similarity Detection Algorithm) という．

SSDA で用いるしきい値の決定には，しきい値自動決定法が提案されている．最初は，しきい値なしで最後まで可算し，そのときの残差を初期しきい値とする．以降，領域内の可算が最後までしきい値を超えることなく終了したら，その残差を次回からのしきい値とし，可算途中でしきい値こ超えたら次の領域へ探索場所を移す．SSDA は，多くの場合途中で可算が打ち切られるため，大幅に計算回数と処理時間を短縮することが可能となる．

4.4.2 疎密探索法

入力画像を何段階かの解像度で表現し，解像度の疎密性を利用することで，類似度や相違度の最大値位置や最小値位置を効率的に探索することができる．この探索方法を疎密探索法 (coarse-to-fine search) という．

疎密探索法は，最初に入力画像を段階的に縮小し，図 4.6 のようにイメージピラミッドを構成する．探索は，イメージピラミッドの低解像度画像から順に行う．低解像度画像を探索して探索位置が決定したら画像解像度を 1 段階上げ，次からの処理は探索位置に相当する周辺の領域のみを探索すればよい．最終的に入力画像の解像度における位置が高速に探索することが可能となる．

4.5 アクティブ探索法 【Source Code】

アクティブ探索法とは村瀬らによって提案された画像の高速探索手法である [7]．アクティブ探索法は，物体の形状変形に安定な色ヒストグラムを特徴として利用し，入力画像中のある位置の類似値からその近傍の類似値の上限値を計算する．上限値が探索値より小さければ，その領域での探索が省略できるため，照合回数を大幅に減らすことが可能となる．

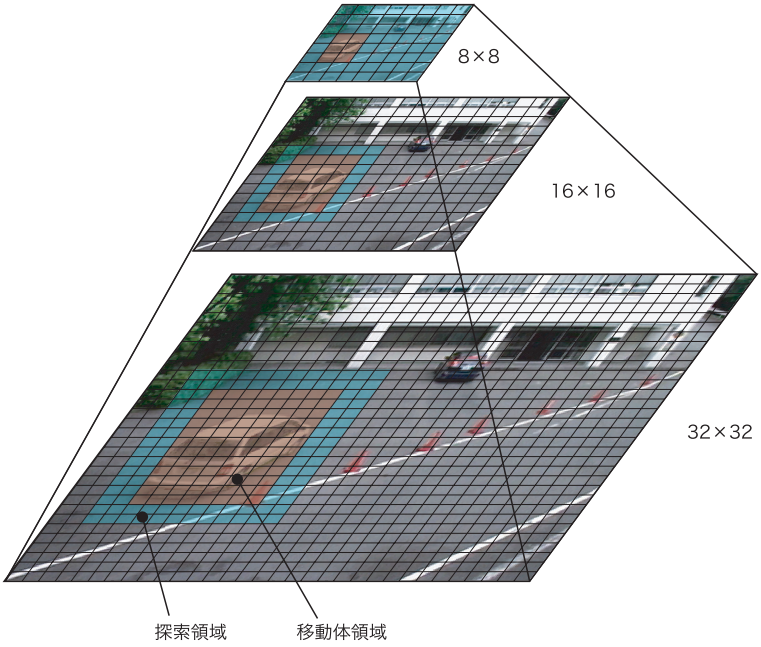


図 4.6: イメージピラミッド

4.5.1　参照画像の色ヒストグラムの作成

参照画像として探索すべき物体の画像から矩形領域を切り出し，この領域の色ヒストグラムを計算する．色ヒストグラムの計算では，RGB 空間の各軸を Q 分割した 3 次元ヒストグラムを作成する．参照画像の色ヒストグラムを $M_i^{'}$ $(i = 1, 2, \ldots, I), I = Q^3$ とすると，$\sum_i M_i^{'}$ の値は矩形領域の画素数となる．次に，各ヒストグラムの要素の値を画素数で割った正規化ヒストグラムを作成する．

$$M_i = \frac{M_i^{'}}{\sum_j M_j^{'}} \tag{4.7}$$

なお，$\sum_{i=0}^{I} M_i$ は 1.0 に正規化されている．参照画像と入力画像の例を図 4.7 に，参照画像と探索領域の色ヒストグラムの例を図 4.8 に示す．

4.5.2　類似値

入力画像中から矩形領域により抽出された領域の正規化ヒストグラムを H，参照画像の正規化ヒストグラムを M とすると，ヒストグラムの類似値 S_{HM} は，

$$S_{HM} = \sum_{i=1}^{I} \min(H_i, M_i) \tag{4.8}$$

によって与えられる．S_{HM} は正規化されているため，0 から 1 の値をとる．

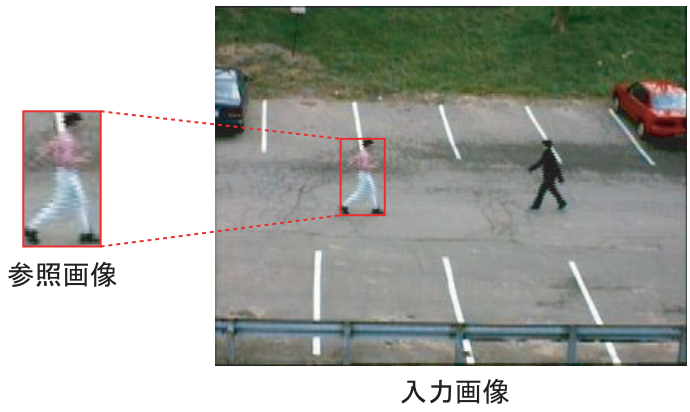


図 4.7: 参照画像と入力画像

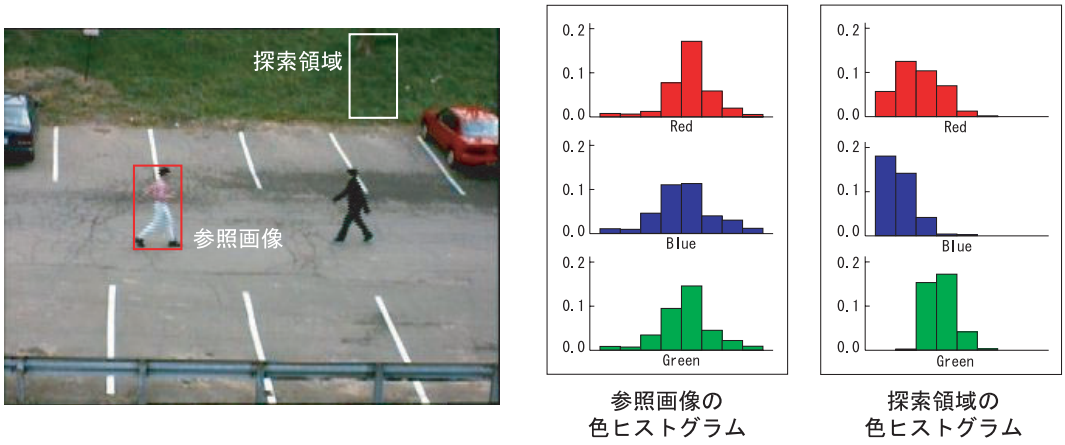


図 4.8: 参照画像と探索領域の色ヒストグラム

4.5.3 上限値

一般に，入力画像中のある位置での部分領域と参照画像との類似値は，その位置の近傍での類似値に類似している．図 4.9 の A の領域と B の領域とのヒストグラムの違いは，A と B の重なりあった部分の部分ヒストグラムは同じであるため，図の差分部分だけであるという関係を利用することが可能である．

図 4.9 に示すように，A，B を画像中の任意の領域，M を参照画像とする．S_{AM} を A と M の類似値，S_{BM} を B と M の類似値とする．$A \cap B$ を A と B の共通領域，$A - B$ を A の領域から B の領域除いた領域，$|A|$ を A の画素数とする．$|A| \leq |B|$ の場合には，

$$S_{AM} = \frac{\min(S_{BM}|B|, |B \cap A|) + |A - B|}{|A|} \tag{4.9}$$

という関係が成り立つ．右辺は S_{BM} が計算されたときの S_{AM} の上限値である．この関係を利用

すれば，ある位置での類似値が計算されればその近傍の類似値の上限値がわかることになるため，もし探索している領域の類似値よりの上限値が小さければ，この領域での類似値の計算をする必要はない．

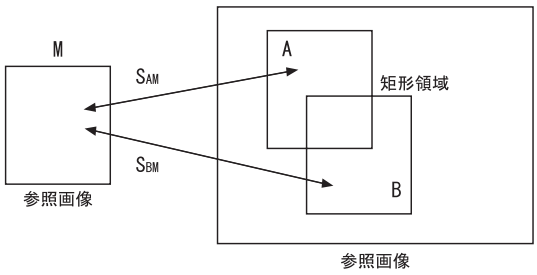
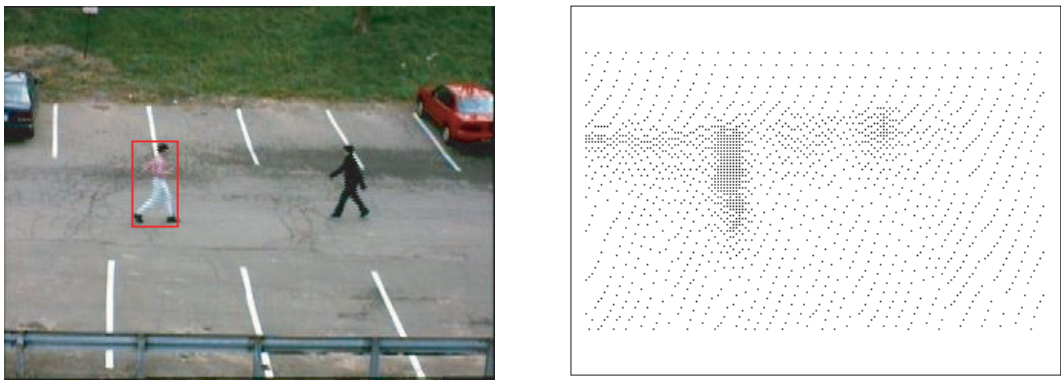


図 4.9: 参照画像と二つの局所領域との関係

4.5.4　アクティブ探索法による計算量の削減

アクティブ探索法は，総当たり法と同じ結果を保証しながら計算量と計算時間を大幅に削減できる．図 4.10 に実際に参照画像と照合を行った箇所を示す．対象領域では密な探索を行い，それ以外の領域では疎に探索していることがわかる．
また，総当たり法，残差逐次検定法，アクティブ探索法の 3 つの手法の処理時間の比較を行った．その計算時間の結果を図 4.11 に示す．なお，処理には CPU : Intel Pentium4 1.5GHz，Memory : 1GB の PC を用いた．320 × 240 サイズの探索に，アクティブ探索法は 0.21 sec で終了していることがわかる．



入力画像　　探索箇所

図 4.10: 入力画像と照合した箇所

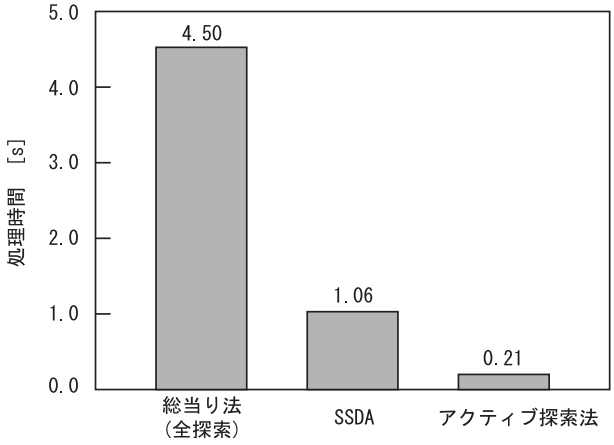


図 4.11: 処理時間の比較

4.6 Mean-Shift によるトラッキング 【Source Code】

Mean-Shift アルゴリズムは，ある関数 $f(x)$ の初期値周辺のある区間における傾きにより，$f(x)$ の値が大きくなる方向へ区間中心の移動を繰り返すことで，初期値周辺において $f(x)$ が極大となる点を求める方法である．Mean-Shift を用いたトラッキングでは，図 4.12 のように追跡対処を囲む窓 (以下ウィンドウと呼ぶ) 内の特徴を求め，それを次フレーム以降で Mean-Shift により求められる移動方向にウィンドウの位置を移動させることにより行う．

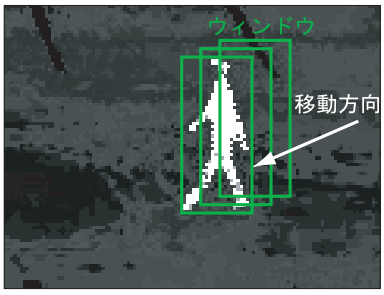


図 4.12: 移動方向とウィンドウ移動

4.6.1 特徴の表現法

ウィンドウ内の特徴として，R, G, B それぞれの 1 次元カラーヒストグラムを用いることにする．初期ウィンドウ内のピクセル $\mathbf{x}_i^*(i = 1, \cdots, n)$ における輝度 u の特徴および，現在のウィンドウ中心 $\mathbf{x}_0$ の周辺ピクセル $\mathbf{x}_i(i = 1, \cdots, n)$ の特徴は式 (4.10),(4.11) に示すように R, G, B それぞれの

正規化ヒストグラムとして得られる．

$$
\begin{aligned}
m_u^R &= \frac{\sum_{i=1}^{n} \delta_K[b_R(\mathbf{x}_i^*) - u]}{n}, \\
m_u^G &= \frac{\sum_{i=1}^{n} \delta_K[b_G(\mathbf{x}_i^*) - u]}{n}, \\
m_u^B &= \frac{\sum_{i=1}^{n} \delta_K[b_B(\mathbf{x}_i^*) - u]}{n}
\end{aligned} \tag{4.10}
$$

$$
\begin{aligned}
d_u^R(\mathbf{x}_0) &= \frac{\sum_{i=1}^{n_c} \delta_K[b_R(\mathbf{x}_i) - u]}{n}, \\
d_u^G(\mathbf{x}_0) &= \frac{\sum_{i=1}^{n_c} \delta_K[b_G(\mathbf{x}_i) - u]}{n}, \\
d_u^B(\mathbf{x}_0) &= \frac{\sum_{i=1}^{n_c} \delta_K[b_B(\mathbf{x}_i) - u]}{n}
\end{aligned} \tag{4.11}
$$

ここで，$b_R(\mathbf{x}), b_G(\mathbf{x}), b_B(\mathbf{x})$ はそれぞれ $\mathbf{x}$ における各 R, G, B での輝度，δ_K は Kronecker のデルタ関数 (if $x = 0$ then $\delta_K[x] = 1$, otherwise $\delta_K[x] = 0$) を表す．

4.6.2　重みの計算

得られたカラーヒストグラムから，現在のウィンドウ中心 $\mathbf{x}_0$ の周辺ピクセル $\mathbf{x}_i$ の R, G, B における重み w_i を計算する．

$$
\begin{aligned}
w^R(\mathbf{x}_i) &= \sqrt{m^R_{b_{R(\mathbf{x}_i)}} / d^R_{b_R(\mathbf{x}_i)}(\mathbf{x}_0)}, \\
w^G(\mathbf{x}_i) &= \sqrt{m^G_{b_{G(\mathbf{x}_i)}} / d^G_{b_G(\mathbf{x}_i)}(\mathbf{x}_0)}, \\
w^B(\mathbf{x}_i) &= \sqrt{m^B_{b_{B(\mathbf{x}_i)}} / d^B_{b_B(\mathbf{x}_i)}(\mathbf{x}_0)}
\end{aligned} \tag{4.12}
$$

各ピクセルの重みは，ピクセルの持つ R, G, B 値より求められた重みの総和となる．

$$
w(\mathbf{x}_i) = w^R(\mathbf{x}_i) + w^G(\mathbf{x}_i) + w^B(\mathbf{x}_i) \tag{4.13}
$$

重みは R, G, B のそれぞれにおける輝度ごとに求められる．また m, d はどちらも正規化された色ヒストグラムであるため，初期ウィンドウと現在のウィンドウ位置周辺における輝度 u の割合を表す．よって，初期ウィンドウ内において多く存在し，現在のウィンドウ位置周辺において少ない色が大きな重みを持つこととなる．

4.6.3 移動量の計算

Mean-Shift による移動量 (Mean-Shift ベクトル)$\Delta\mathbf{x}$ は次のように求める．

$$\Delta\mathbf{x} = \frac{\sum_{i=1}^{n} K\left(\mathbf{x}_i - \mathbf{x}_0, \sigma\right) w_i(\mathbf{x}_i)(\mathbf{x}_i - \mathbf{x}_0)}{\sum_{i=1}^{n} K\left(\mathbf{x}_i - \mathbf{x}_0, \sigma\right) w_i(\mathbf{x}_i)} \tag{4.14}$$

ここで，$\mathbf{x}_i (i = 1, \cdots, n)$ はウィンドウ中心 $\mathbf{x}_0$ の周辺ピクセルで，w_i は初期ウィンドウ内の特徴と現在のウィンドウ内およびその周辺の特徴より求められた $\mathbf{x}_i$ における重みを表す．つまり，求められたウィンドウ位置周辺の重み分布を基に，ウィンドウを移動させる．また，$K(\mathbf{x}, \sigma)$ はカーネル関数である．Mean-Shift トラッキングにおけるカーネル関数は以下の式により求める．

$$K(\mathbf{x}, \sigma) = \exp\left(\frac{-\left(x^2 + y^2\right)}{2\sigma^2}\right) \tag{4.15}$$

初期ウィンドウから離れるほど $K\left(\mathbf{x}, \sigma\right)$ の値が小さくなるため，初期値周辺の重みの高い分布となる．また，σ はカーネル関数の分散を表しており，σ を大きくすることで移動量の大きなウィンドウに対しても追跡が可能となる．

4.6.4 トラッキング手順

以上までで述べた Mean-Shift アルゴリズムにより，トラッキングを行う手順を示す．前フレームにおいて求められたウィンドウ中心を $\hat{\mathbf{x}}_0$ の初期値として以下の手順で行う．

1. 現在のフレームにおける $\hat{\mathbf{x}}_0$ を中心としたウィンドウ内およびその周辺における特徴 $d(\hat{\mathbf{x}}_0)$ を求める
2. 式 (4.13) より，ウィンドウ内およびその周辺の各ピクセルの重み $w(\mathbf{x}_i)(i = 1, \cdots, n_c)$ を求める．
3. 式 (4.14) より，移動量を計算し，$\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}_0 + \Delta\mathbf{x}$ とする．
4. 移動量 $\Delta\mathbf{x}$ から求めた距離が予め設定しておいた ϵ よりも大きい場合は，$\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{x}}_1$ として step.1 に戻る．

以上の手順により，最終的に得られた $\hat{\mathbf{x}}_1$ を現在のフレームにおけるウィンドウ中心とする．

4.6.5 Mean-Shift によるトラッキングの特徴

Mean-Shift アルゴリズムを用いたトラッキング結果を図 4.13 に示す．図より，オクルージョンが発生した状態 (300，390 フレーム) においてもトラッキングが可能であることがわかる．Mean-Shift アルゴリズムを用いたトラッキングでは，ウィンドウ位置の周辺において，初期ウィンドウでの特

徴に類似したピクセル分布のある方向へのウィンドウを移動させるため，局所的な探索によってトラッキングを実現する．また，ピクセル分布の特徴 (色，テクスチャなど) を基に探索するため，様々な形状の物体をトラッキングすることに適しており，部分的なオクルージョンや混雑状態，および，カメラの位置変化に対して頑健である．また，ITS の分野では，ピクセル分布の特徴の他に，エッジの強度を用いるものや [8]，ロバスト性を向上させるために，差分画像を適用する手法も提案されている [9].

Mean-Shift の改良として，文献 [3] では，Mean-Shift 法の探索ウィンドウサイズを可変して対象物のスケールの変化に適応させている．また Mean-Shift により求められた移動体の移動量を基に，常に対象物がカメラ中心となるようにカメラの Pan/Tilt を制御するアクティブカメラが報告されている [4].

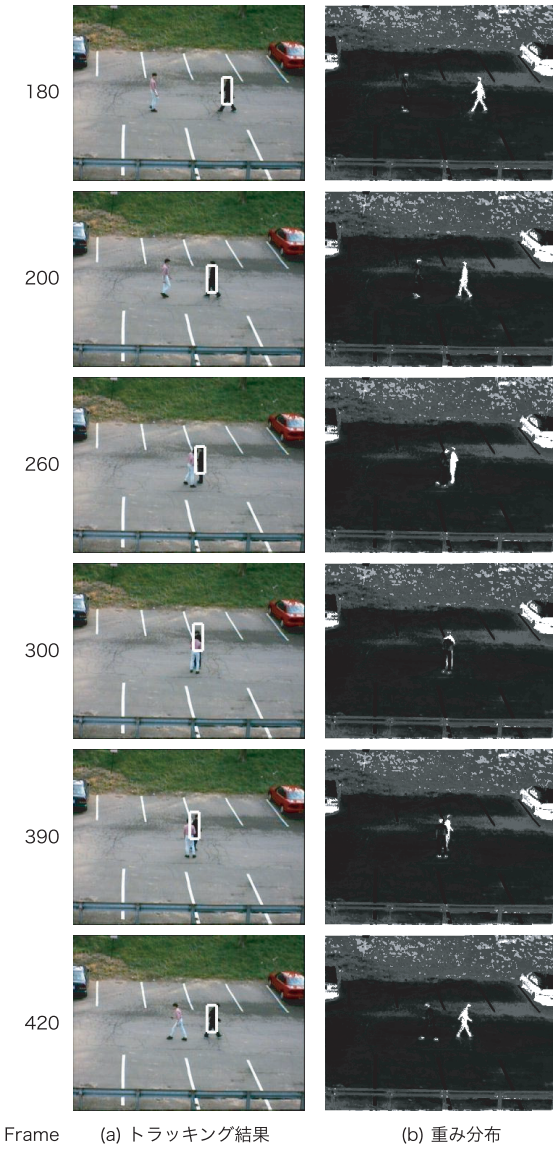


図 4.13: Mean-Shift によるトラッキング結果

4.7　Particle Filterによるトラッキング 【Source Code】

パーティクルフィルタの基本アイデアは非常にシンプルで，事前確率分布，事後確率分布をこれらの条件付分布に従う多数のパーティクルを用いて近似を行う手法である．追跡する対象を状態量と重み (尤度) を持つ多数のパーティクルにより状態空間内全体の確率分布を近似することで，ノイズや環境の変動に対して頑健な追跡を行うことができる．

4.7.1　処理の流れ

パーティクルフィルタでは，時刻 t における事後確率密度$p(\boldsymbol{x}_t|\boldsymbol{z}_t)$を，状態 $\boldsymbol{x}$ の仮説とその重みの N 個の組から成るパーティクル群

$$\boldsymbol{s}_t^{(i)} = \{\boldsymbol{x}_t^{(i)}, \boldsymbol{\pi}_t^{(i)}\}(i = 1, \cdots, N) \tag{4.16}$$

により近似することで，対象物体を追跡する手法である．ここで時刻 t における i 番目の追跡対象の状態量$\boldsymbol{x}_t^{(i)}$，$\boldsymbol{\pi}_t^{(i)}$ はその尤度である重みを示している．
時刻 t において画像から観測値$\boldsymbol{z}_t$が得られると，追跡対象の状態$\boldsymbol{x}_t$を確率変数とする確率密度は，事後確率密度$p(\boldsymbol{x}_t|\boldsymbol{z}_t)$として表される．この$p(\boldsymbol{x}_t|\boldsymbol{z}_t)$はベイズの法則を用いて以下のように表すことができる．

$$p(\boldsymbol{x}_t|\boldsymbol{z}_t) = \alpha p(\boldsymbol{z}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{t-1}) \tag{4.17}$$

ここで，α は正規化のための定数である．

一般的に用いられている CONDENSATION アルゴリズム [11] は，各フレームに対して対象物体を追跡する際，その繰り返し計算を主に予測 (prediction)，観測 (obseration)，選択 (selection) の 3 つのステップに分けて実行している．このアルゴリズムを考慮したパーティクルフィルタによる移動体の追跡アルゴリズムを以下に示す (図 4.14 参照)．

Step1. 初期化 対象物体を指定し，N 個のパーティクルを生成

Step2. 予測 パーティクルを状態遷移モデルに基づき移動

Step3. 観測 各パーティクルについて矩形を作成，矩形内の色ヒストグラムを取得

Step4. 尤度計算 各パーティクルより取得した色ヒストグラムと前状態の対象物体より取得した参照ヒストグラムとを比較，パーティクルの尤度 $L_t^{(i)}$ を取得

Step5. 対象推定 各パーティクルの尤度を重みとした重み付き平均を求め，対象物体を推定

Step6. 選択 尤度が高い順番で尤度の高さに比例する確率で N 個のパーティクルを選択

Step7. 次状態推定 $t+1$ として Step2 の予測を実行

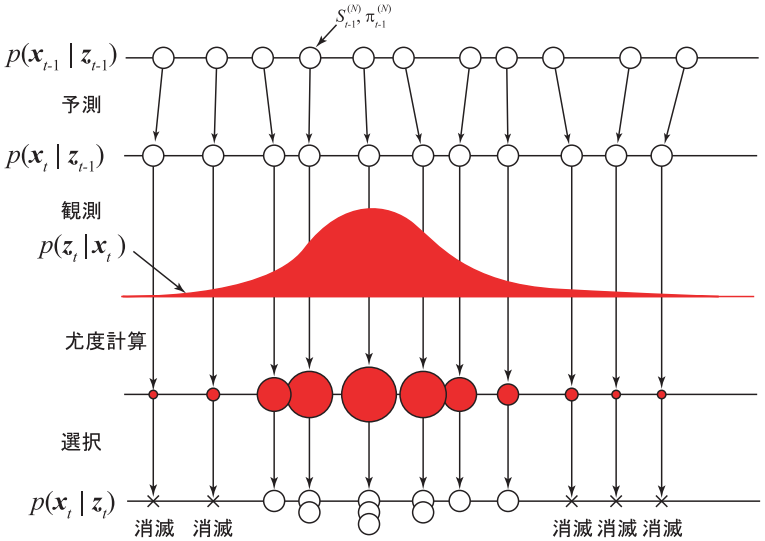


図 4.14: パーティクルフィルタのアルゴリズム

初期フレームより対象物体を矩形を用いて指定し，その矩形の重心位置を状態量として N 個のパーティクルを生成する．この時，次状態推定のための参照ヒストグラムを対象物体と指定した矩形より取得する．次に，各パーティクルについて状態遷移モデルに沿い遷移させ，対象物体の次状態位置を予測する．各パーティクルの状態量より矩形を作成し，矩形内の色ヒストグラムを観測する．そして観測した各パーティクルの色ヒストグラムと前状態の対象物体より取得した参照ヒストグラムを比較することより、遷移したパーティクルの位置に対象物体が存在する確率を尤度計算より求め，その確率を尤度と表現する．各パーティクルの尤度と状態量から重み付き平均を求め，対象物体の位置を推定する．この対象物体の位置から矩形を次状態の尤度計算に用いる参照ヒストグラムを取得する．そして，パーティクルの尤度を考慮し，尤度が高ければ尤度をそのパーティクルの状態量が等しいものを尤度に比例する確率で複製する．この処理をパーティクル総数が N 個となるまで選択し，新たなパーティクル群を作成，次状態の推定を行う．

状態ベクトル

パーティクルフィルタを用いて物体の重心位置の推定を行い，矩形を用いて対象物体を示す．従って，時刻 t における対象の位置は (x_t, y_t) の二次元である．対象位置と同様に各パーティクルに状態ベクトル $\boldsymbol{x}_t$ を持たせる (式 (4.18))．

$$\boldsymbol{x}_t = (x_t, y_t)^T \tag{4.18}$$

パーティクルの状態遷移モデル

対象物体の位置を推定する際，物体の運動を考慮することにより，効果的なパーティクルの遷移が可能となる．しかし，対象の運動が不規則である場合，運動モデルを定義することにより追跡に

失敗しやすいという欠点がある．本研究では，物体の運動は不規則であると仮定し，正規乱数を用いてパーティクルを遷移させる．

$$\boldsymbol{x}_t^{(i)} = \boldsymbol{x}_{t-1}^{(i)} + \omega_t \tag{4.19}$$

ここで，ω_t はガウシアンノイズである．

尤度計算

本稿では，尤度計算に色ヒストグラムの類似度を利用する．各パーティクルの位置座標に基づき配置した矩形内の色特徴量として HSV ヒストグラムを作成する．色ヒストグラムを特徴量とする利点として，他の特徴と比べて対象物体の回転や歪み，スケールの変動に対して特徴量の変化が少ないことがあげられる．前状態で推定された対象の HSV ヒストグラムを参照ヒストグラムとし，現状態の各パーティクルの観測ヒストグラムとの類似度から尤度 $L_t^{(i)}$ を求める．ここでは，この二つのヒストグラムの類似度として，Bhattacharyya 係数を用いている．二つの分布$\boldsymbol{p}$と$\boldsymbol{q}$との Bhattacharyya 係数 S_c は以下のように表すことができる．

$$S_c = \sum_{u=1}^{m} \sqrt{p_u q_u} \tag{4.20}$$

m は成分数であり，ここではヒストグラムの長さである．$\sum_{u=1}^{m} p_u = \sum_{u=1}^{m} q_u = 1$ となる．この係数は二つのヒストグラム分布が似ているほど大きな値になり，完全に一致すると $S_c = 1$ となる．求めた類似度から，求めた類似度から各パーティクルの尤度を式 (4.21) で計算する．

$$L_t^{(i)} = \exp(kS_c) \tag{4.21}$$

ここで k は定数である．

対象物体の推定

各パーティクルの尤度を重みとして現状態の対象物体位置を推定する．ここで，各パーティクルが持つ尤度を合計 1 になるように正規化をした $\boldsymbol{\pi}_t^{(i)}$ を作成する．各パーティクルの状態量と重みを考慮した重み付き平均を用いて対象物体の位置を推定する．

$$X_t = \sum_{i=1}^{N} \boldsymbol{\pi}_t^{(i)} \boldsymbol{x}_t^{(i)} \tag{4.22}$$

パーティクルの選択

次状態に継承するパーティクルは重みに従い抽出する．パーティクルの尤度 $\boldsymbol{\pi}$ と総数 N を考慮した以下の選択基準を定義する．

$$n_t^{(i)} = N\boldsymbol{\pi}_t^{(i)} \tag{4.23}$$

ここで，パーティクルの尤度が大きい順に選択基準を用いてパーティクルを $n_t^{(i)}$ 増やす．この処理を合計パーティクル数が N 個になるまで繰り返す．もし，選択基準を用いたパーティクルの増加処理の結果が N 個以下の場合，N 個になるまで対象物体推定位置にパーティクルを発生させる．この処理を行うことにより，次状態のパーティクルは現状態の対象物体付近に存在することになり，その位置からパーティクルを遷移させることにより精度が高い次状態の推定を行うことが出来る．

4.7.2 パーティクルフィルタによるトラッキングの特徴

パーティクルフィルタを用いたトラッキング結果を図 4.16 に示す．図 4.16(a) はパーティクルの数を 10 に (b) はパーティクルの数を 50 にしたときのトラッキング例である．赤い矩形はトラッキング結果であり，その他の矩形は白が最大尤度となる各パーティクルの尤度を表示したものである．パーティクルフィルタでは，ちりばめられた個々のパーティクルにおいて予測により対象物の追跡を行い，尤度を求めることにより，対象物のトラッキングを行う．そのため，尤度の高いパーティクル群が対象物となるため，ノイズを含んだシーンにおいて効果をより発揮する．

Mean Shift は繰り返し的に極大値を探索する手法で，Particle Filter は重み付き平均により確率的に推定する手法である．そのため，探索する尤度分布が図 4.15 (a) のように，ピークの周辺に極大値が存在する場合は，Mean Shift では最適解に収束しない場合がある．一方，Particle Filter は，ランダムサンプリングにより確率的に推定する手法であるため，最適解の近似値を得ることができる．しかし，図 4.15 (b) のような鋭いピークを持つ場合，Particle Filter ではサンプリングの密度が十分でないと正確な推定が困難であるが，Mean Shift では勾配方向へ移動させる手法のため，精度よく最適解を求めることが可能である．

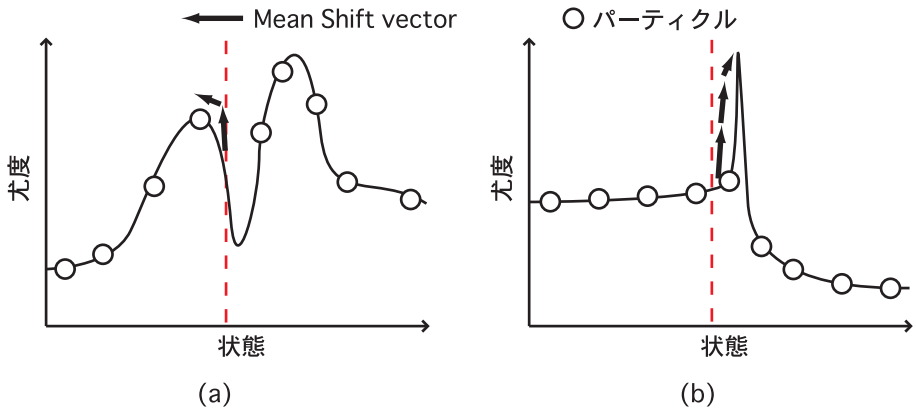


図 4.15: 尤度分布

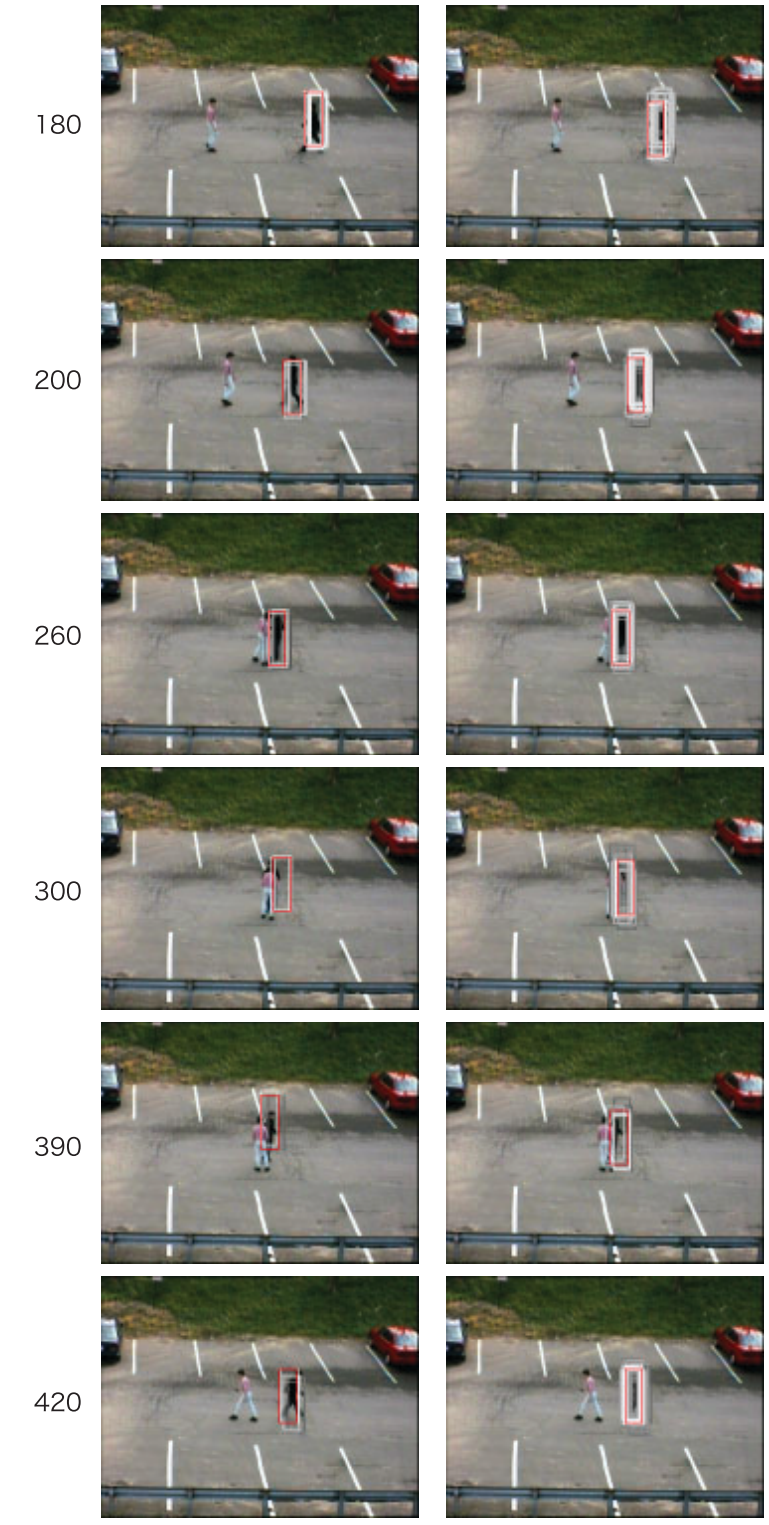


図 4.16: パーティクルフィルタによるトラッキング結果

4.8 類似物体に頑健なパーティクルフィルタ

追跡対象の色特徴を用いたパーティクルフィルタでは，対象物体と類似する物体が探索領域範囲内に侵入すると追跡が困難となる．これは，パーティクルの遷移範囲内に類似パターンを持つ物体が観測された場合，尤度が各物体毎に高くなるためである．このときの尤度分布は図 4.17 に示すように物体数分の山を持つ形状となる．類似物体を含む尤度分布から対象物体の位置を求めた場合，その結果は対象物体から外れ，追跡が破綻する．そこで，尤度分布が対象物体と類似物体の尤度で構成されていることに着目する．各物体の尤度を混合正規分布モデルを用いて判別することにより，追跡の破綻を防ぐ [12].

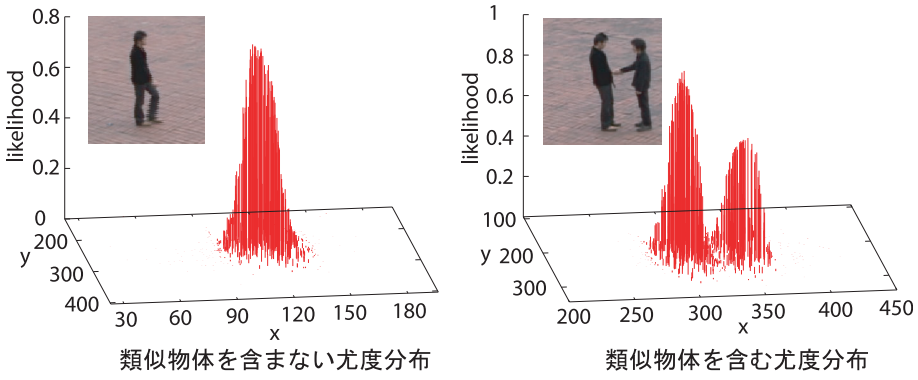


図 4.17: 尤度分布

4.8.1 混合正規分布モデルによる物体判別

混合正規分布モデルとパラメータ推定

類似物体の情報を含む尤度分布から尤度が高いパーティクルの選択を行う．尤度が高いパーティクル群の座標を特徴ベクトルから，K 個の正規分布の重み付き線形結合による混合正規分布モデルを推定する．K 個の成分からなる 2 次元の混合正規分布パラメータを $\Phi = (\alpha_j, \phi_j)_{j=1}^{K}$ ，$\phi = (\boldsymbol{\mu}, \sum)$ とし，尤度が最大となるパラメータを ϕ_{ML} を EM アルゴリズムにより推定する．

$$\Phi_{ML} = \arg\max_{\Phi} \sum_{j=1}^{K} \alpha_j \eta(\boldsymbol{x}; \phi_j) \tag{4.24}$$

$$\eta(\boldsymbol{x}; \phi_j) = \frac{1}{\sqrt{(2\pi)^2 |\sum_j|}} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_j)^T \sum_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j) \right\} \tag{4.25}$$

$\eta(\boldsymbol{x}; \phi_j)$ は平均ベクトル $\boldsymbol{\mu}_j$，共分散行列 $\sum_j$ の正規分布成分である．α_j はそれらの混合比であり，$\sum_{j=1}^{K} \alpha_j = 1 (0 \leq \alpha_j < 1)$ を満たす．

正規分布の統合

物体が K 個以下の場合，同じ領域に複数の正規分布が当てはめられることになる．そこで，同領域に当てはめられている同じパラメータを持つ正規分布を同一物体の状態量に属するものとして統合する．これにより，物体数に対応した正規分布の数を求めることができる．

対象尤度分布の作成

各正規分布の中心と前状態の対象物体推定位置の中心座標との距離を測定し，最も近い正規分布を対象物体と判別する．選択された正規分布成分 $\eta(\boldsymbol{x}_t;\phi_d)$ と尤度分布を掛け合わせることにより，対象物体のみを考慮した尤度 $\pi_t^{(i)}$ を次式により算出する (図 4.18 参照)．

$$\pi_t^{(i)} = L_t^{(i)}\eta(\boldsymbol{x}_t;\phi_j) \tag{4.26}$$

この対象物体尤度分布の尤度 $\pi_t^{(i)}$ を重みとしたパーティクルの重み付き平均より対象物体の推定値を求める．

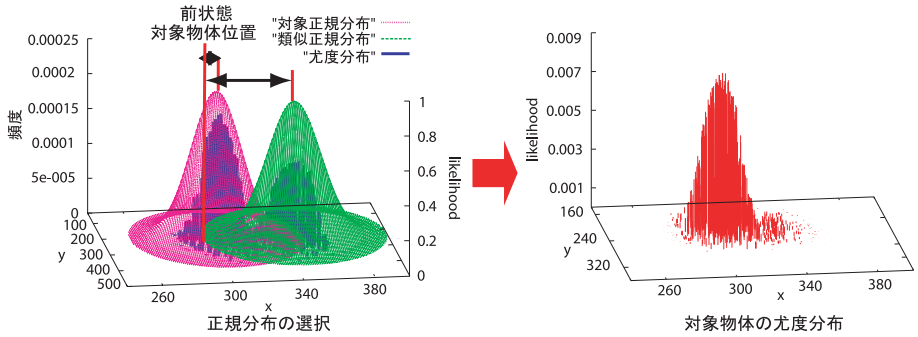


図 4.18: 対象物体尤度分布の作成

4.8.2　類似物体を含むシーケンスのトラッキング

図 4.19 に類似物体が存在する際のトラッキング例を示す．このように，混合正規分布モデルにより類似物体の判別を行うことにより，パーティクルフィルタを用いて物体を追跡することが可能となる．

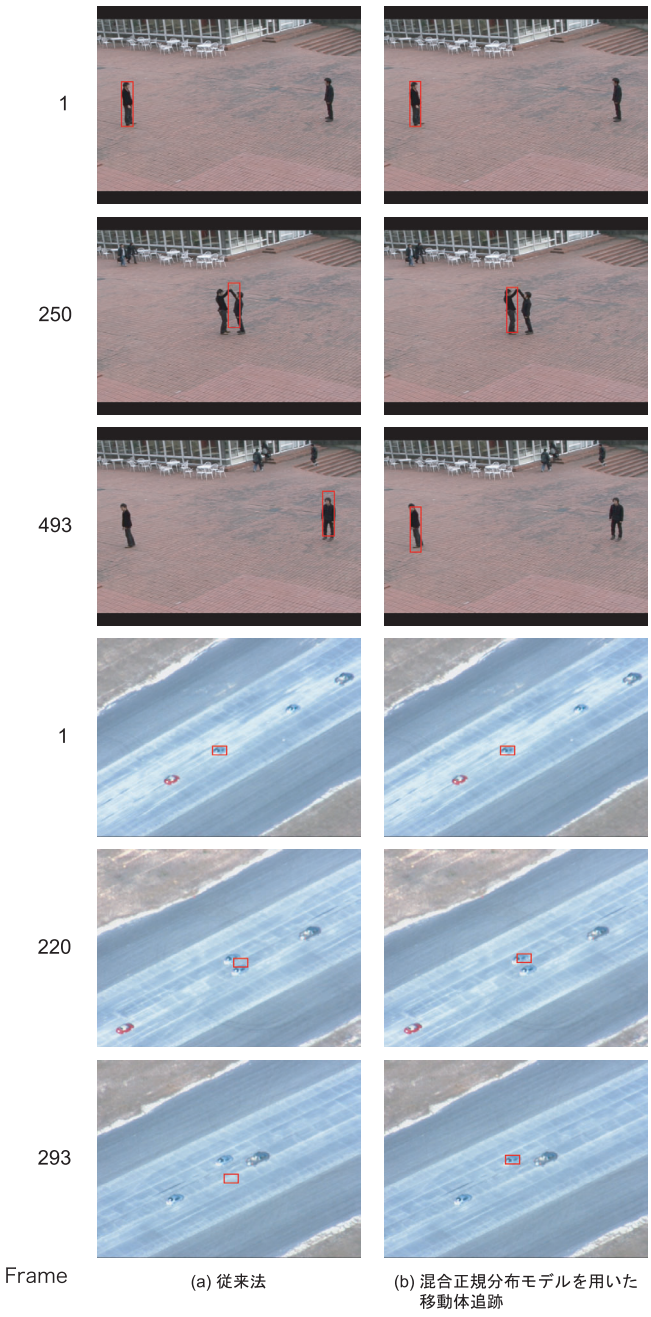


図 4.19: 類似物体を含む移動体追跡

4.9　Kanade-Lucas-Tomasi 法による特徴点追跡

特徴点追跡の一般的な手法として Lucas-Kanade アルゴリズム (以下 LK 法) 及び，その発展型である Kanade-Lucas-Tomasi 法 (以下 KLT 法) がある [10][14]．KLT 法による移動体追跡は，特徴点追跡に最適化された特徴点を選択し，それを 2 画像間の特徴点間に Newton-Raphson の最小化法を用いて追跡を行う手法である．

4.9.1　Lucas-Kanade アルゴリズム

ある 2 枚の画像に同一の領域 R が写っているとする．片方の画像を基準 F としたとき，もう一方の画像 G では R がどれだけずれているか（移動しているか）の量 $\boldsymbol{h}$ を求めるには以下のいずれかの値を最小化するような $\boldsymbol{h}$ を見つければよい．

$$\|L_1\| = \sum_{\boldsymbol{x}\in R} |F(\boldsymbol{x}+\boldsymbol{h}) - G(\boldsymbol{x})| \tag{4.27}$$

$$\|L_2\| = \Big(\sum_{\boldsymbol{x}\in R} [F(\boldsymbol{x}+\boldsymbol{h}) - G(\boldsymbol{x})]^2\Big)^{1/2} \tag{4.28}$$

$$L_3 = \frac{-\sum_{\boldsymbol{x}\in R} F(\boldsymbol{x}+\boldsymbol{h})G(\boldsymbol{x})}{\Big(\sum_{\boldsymbol{x}\in R} F(\boldsymbol{x}+\boldsymbol{h})^2\Big)^{1/2}\Big(\sum_{\boldsymbol{x}\in R} G(\boldsymbol{x})^2\Big)^{1/2}} \tag{4.29}$$

ここで $\boldsymbol{x}$ はピクセルの位置であり，$F(\boldsymbol{x}), G(\boldsymbol{x})$ は各画像における $\boldsymbol{x}$ での輝度値を表す．

1 次元への置き換え

式 (4.27),(4.28),(4.29) の最小化は 2 次元の画像での問題ではなく，1 次元の波形のずれの問題として置き換えることができる．そこで，1 次元の問題とした場合の h の求め方について考える (図 4.20)．ある波形 F より h 進んだ同一形状の波形 G があったとき，2 つの波形の間には以下の式が成り立つ．

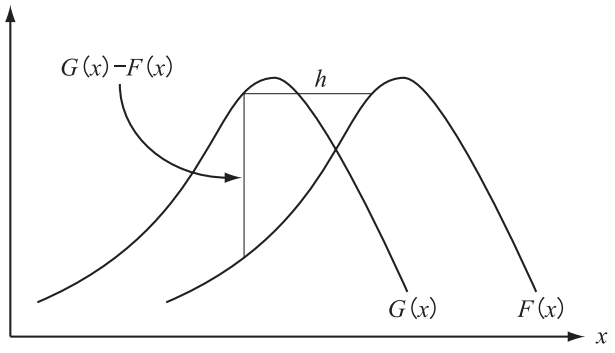


図 4.20: 1 次元の波形のずれ

$$F(x+h) = G(x) \tag{4.30}$$

ここで h を微小変化と仮定すると以下の近似が成り立つ．

$$\begin{aligned} F'(x) &\approx \frac{F(x+h) - F(x)}{h} \\ &= \frac{G(x) - F(x)}{h} \end{aligned} \tag{4.31}$$

また，式 (4.31) より以下の式が求められる．

$$h \approx \frac{G(x) - F(x)}{F'(x)} \tag{4.32}$$

ここで，式 (4.31) は近似式であるので誤差を含む．そのため，いくつかの点における値の平均を求めることでより真値に近い値を求めることができる．平均を求める式は以下のようになる．

$$h \approx \sum_x \frac{G(x) - F(x)}{F'(x)} / \sum_x 1 \tag{4.33}$$

この式で h を定義する事ができた（$F'(x)$ の求め方については後述）．ただし，先に述べたように式 (4.31) は近似式であるため誤差を含む．式 (4.33) では平均を求めることで誤差の影響を緩和したが，誤差の小さいと考えられる点に対して重み付けを行う事でさらに精度を良くする事が期待できる．式 (4.31) は 2 点の変化率の平均を求めることで微分の近似を得ているといえる．つまり，2 点の変化率の差が小さければ誤差の小さい値を求めることができるが，逆に差が大きければ誤差は大きくなると考えられる．2 点における変化率の差は，変化率の変化率，つまり 2 次微分で評価する事ができる．2 次微分の絶対値が大きければ 2 点間の変化が大きく，小さければほとんど変化のない値となる．2 次微分は以下の式で近似することができる．

$$F''(x) \approx \frac{G'(x) - F'(x)}{h} \tag{4.34}$$

ここで h は x によらない値のため，省略しても大小関係に影響はない．また，$F''(x)$ の絶対値が大きいほど重みを小さくしたいため，重み関数は以下のようになる．

$$w(x) = \frac{1}{|G'(x) - F'(x)|} \tag{4.35}$$

この重みを式 (4.33) に加えると以下のようになる．

$$h \approx \sum_x \frac{w(x)[G(x) - F(x)]}{F'(x)} / \sum_x w(x) \tag{4.36}$$

式 (4.36) を以下のように繰り返し計算によって解く事で波形のずれ h を求めることができる．

$$\begin{aligned} h_0 &= 0 \\ h_{k+1} &= h_k + \sum_x \frac{w(x)[G(x) - F(x+h_k)]}{F'(x+h_k)} / \sum_x w(x) \end{aligned} \tag{4.37}$$

式の拡張

1 次元の h は式 (4.37) で求めることができるが，この式には $F'(x)$ を分母とした分数をいくつか足し合わせる必要がある．分数は分母が 0 になる場合は計算不能となるため，一点でも $F'(x) = 0$ となる点を含むと計算ができなくなる．そこで，式を別の形で定義する事でこの問題を回避する．式 (4.31) より以下の近似が得られる．

$$F(x+h) \approx F(x) + hF'(x) \tag{4.38}$$

ここで最小化すべき関数である式 (4.28) を用いる．ただし，式 (4.28) では全体を $\frac{1}{2}$ 乗しているが，大小関係には影響がないため，省略可能である．

$$E = \sum_x [F(x+h) - G(x)]^2 \tag{4.39}$$

式 (4.39) を最小とするような h を求めるため，式 (4.39) を h で偏微分し，0 となる点を求める．

$$\begin{aligned} 0 &= \frac{\partial E}{\partial h} \\ &\approx \frac{\partial}{\partial h} \sum_x [F(x) + hF'(x) - G(x)]^2 \\ &= \sum_x 2F'(x)[F(x) + hF'(x) - G(x)] \end{aligned} \tag{4.40}$$

よって

$$h \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2} \tag{4.41}$$

式 (4.41) は分数の合計がなくなり，計算不能となる可能性が大幅に減少した．ここで，式 (4.37) と同様に重み関数の導入と繰り返し計算を行うと以下のようになる．

$$\begin{aligned} h_0 &= 0 \\ h_{k+1} &= h_k + \frac{\sum_x w(x)F'(x+h_k)[G(x) - F(x+h_k)]}{\sum_x w(x)F'(x+h_k)^2} \end{aligned} \tag{4.42}$$

微分値の計算

これまでの方法は全て $F'(x)$ が求められるという前提であった．しかし，実際は式 (4.31) で示されているように $F'(x)$ を求めるには h が必要となる．そこで実際の計算では $F'(x)$ は以下のように求める．

$$F'(x) \approx \frac{F(x+\Delta x) - F(x)}{\Delta x} \tag{4.43}$$

Δx は微小量である．また，$G'(x)$ も同様に求める．

多次元への展開

1 次元における移動量 h を求める事方法を示したが，同様の考え方を多次元に展開する．最小化すべき関数は以下のようになる．

$$E = \sum_{\boldsymbol{x} \in R} [F(\boldsymbol{x} + \boldsymbol{h}) - G(\boldsymbol{x})]^2 \tag{4.44}$$

多次元では全ての要素はベクトルとなる．ここで式 (4.38) より

$$F(\boldsymbol{x} + \boldsymbol{h}) \approx F(\boldsymbol{x}) + \boldsymbol{h} \frac{\partial}{\partial \boldsymbol{x}} F(\boldsymbol{x}) \tag{4.45}$$

ただし，

$$\frac{\partial}{\partial \boldsymbol{x}} = \Big[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_n}\Big]^T \tag{4.46}$$

また，式 (4.40) より

$$\begin{aligned} 0 &= \frac{\partial}{\partial \boldsymbol{h}} E \\ &\approx \frac{\partial}{\partial \boldsymbol{h}} \sum_{\boldsymbol{x}} [F(\boldsymbol{x}) + h \frac{\partial F}{\partial \boldsymbol{x}} - G(\boldsymbol{x})]^2 \\ &= \sum_{\boldsymbol{x}} 2 \frac{\partial F}{\partial \boldsymbol{x}} [F(\boldsymbol{x}) + \boldsymbol{h} \frac{\partial F}{\partial \boldsymbol{x}} - G(\boldsymbol{x})] \end{aligned} \tag{4.47}$$

よって

$$\boldsymbol{h} = \Big[\sum_{\boldsymbol{x}} \Big(\frac{\partial F}{\partial \boldsymbol{x}}\Big)^T [G(\boldsymbol{x}) - F(\boldsymbol{x})]\Big] \Big[\sum_{\boldsymbol{x}} \Big(\frac{\partial F}{\partial \boldsymbol{x}}\Big)^T \Big(\frac{\partial F}{\partial \boldsymbol{x}}\Big)\Big]^{-1} \tag{4.48}$$

式 (4.48) を式 (4.42) と同様に重み関数の導入と繰り返し計算によって解くことで，多次元の場合の移動量 $\boldsymbol{h}$ を求めることができる．

さらなる一般化

これまで移動量は各軸の移動量のベクトルである $\boldsymbol{h}$ を対象としていた．しかし，画像における移動量はこのような平行移動成分のみでなく，回転，サイズの変化（スケーリング）が含まれる．しかし，このような場合でも同様の考え方を適用できる．回転，スケーリングの変化を与える行列を $\boldsymbol{A}$ とおくと，以下の関係が成り立つ．

$$G(\boldsymbol{x}) = F(\boldsymbol{x}\boldsymbol{A} + \boldsymbol{h}) \tag{4.49}$$

また，ステレオ画像を対象とした場合では 2 枚の画像間でコントラストや明るさに差がある場合がある．その補正として以下の式を満たすような α, β を導入する．

$$F(\boldsymbol{x}) = \alpha G(\boldsymbol{x}) + \beta \tag{4.50}$$

これを最小化すべき式の形にすると

$$E = \sum_{\boldsymbol{x}} [F(\boldsymbol{x}\boldsymbol{A} + \boldsymbol{h}) - (\alpha G(\boldsymbol{x}) + \beta)]^2 \tag{4.51}$$

式 (4.51) をこれまでと同様に重み関数の導入と繰り返し計算によって解くことで，$\boldsymbol{h}$ と $\boldsymbol{A}$ を求めることができる．

4.9.2　Kanade-Lucas-Tomasi 法

LK 法は動画像，ステレオ画像どちらにも適用でき，平行移動だけでなく回転，スケーリングにも対応した非常に一般性のある手法である．それに対し，KLT 法は動画像中のトラッキングのみを対象とし，さらにいくつかの仮定を導入した手法である．動画像中の移動物体の速度に対し，画像間の撮影間隔が十分に小さければ画像間での移動量は微小となり，それらは平行移動のみで近似する事ができる．つまり式 (4.48) を求めればよい．また，基準となる画像 F は追跡後に G に更新する．これにより，蓄積していく回転とスケーリングの成分を打ち消す事ができる．

特徴点の選択

どれだけ優れた追跡法を用いても，対象となる点がほとんど特徴を持っていなければ追跡は難しい．つまり特徴点の追跡は点の選択も重要である．KLT 法では移動量 $\boldsymbol{h}$ 求める際の係数の行列

$$D = \begin{bmatrix} \frac{\partial F}{\partial x}^2 & \frac{\partial F}{\partial x}\frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial x}\frac{\partial F}{\partial y} & \frac{\partial F}{\partial y}^2 \end{bmatrix} \tag{4.52}$$

の固有値 λ_1, λ_2 を基準に特徴点を選択する．固有値 λ_1, λ_2 はそれぞれ x 軸方向，y 軸方向への輝度値の分散を表している．輝度値の分散が大きければ変化が激しく特徴的な点といえるため，追跡を行いやすいといえる．よって，以下の式を満たす点を追跡点として選択する．

$$\min(\lambda_1, \lambda_2) > \lambda \tag{4.53}$$

ただし λ はしきい値である．

4.10 SIFT 特徴量を用いた特徴点追跡

都築らは，回転やスケール変化に対して頑健な，SIFT 特徴量を用いた Mean Shift 探索による特徴点追跡法 [19] を提案している．この手法では，SIFT により記述された特徴量を用いて Mean Shift による画像空間とスケール空間を交互に極値探索を行うことにより特徴点の移動量とスケールを探索して追跡を行う．図 4.21 に，SIFT 特徴量を用いた特徴点追跡と KLT 法の追跡例を示す．各点は 50 フレームの軌跡を表している．SIFT による追跡手法は SIFT 特徴量にスケールが含まれているため，KLT 法に比べ，より多くの点を長時間にわたり追跡できていることが分かる．

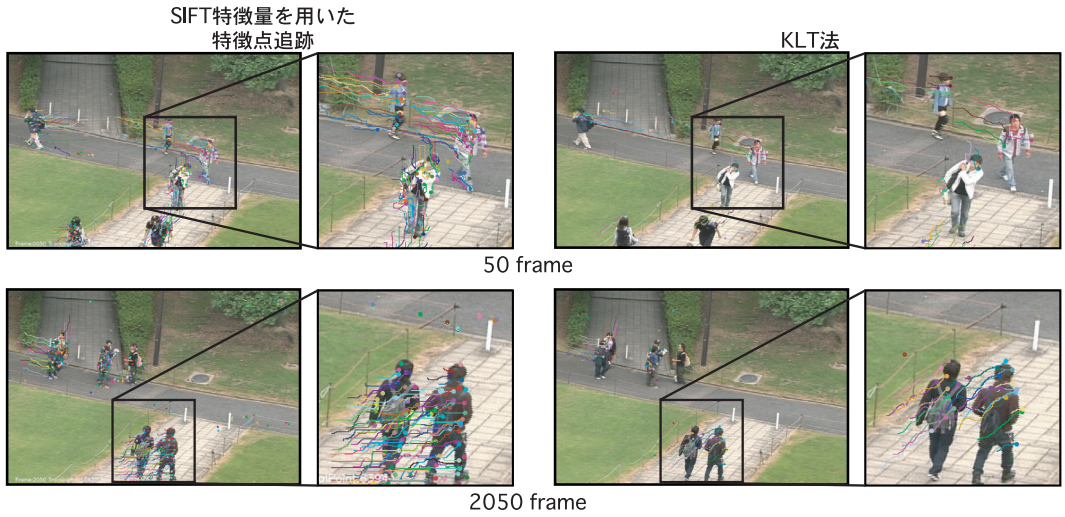


図 4.21: SIFT 特徴量を用いた特徴点追跡と KLT 法の追跡例

4.11 一括処理による物体追跡

一括処理による物体追跡は，これまでに述べた逐次処理の追跡手法に比べオクルージョンに対して頑健な追跡が可能となる．以下に，時空間画像から対象物体の追跡技術として，順方向と逆方向から追跡する Bi-Directional Tracking，対象物体をボリュームとして抽出する Graph Cuts について述べる．

4.11.1 Bi-Directional Tracking

Bi-Directional Tracking [20] は，Sun らにより提案された手法で，初期フレームと最終フレームで対象領域を指定し，順方向と逆方向から追跡する手法である．Bi-Directional Tracking は，事後確率最大化 (MAP) により動画像全体を最適化することで，オクルージョンや重なりに対しても頑健に追跡することができる．

対象物体の座標 p，スケール s としたとき，対象物体の状態を $x = \{p, s\}$ と定義する．状態 x のヒストグラムを $h(x)$ としたとき，Bhattacharyya 距離は以下のようになる．

$$B(h(x_0), h(x_i)) = 1 - \sum_{j=1}^{B} \sqrt{h_j(x_0) h_j(x_i)} \tag{4.54}$$

ここで，初期フレームと最終フレームに入力された初期状態を $\{x_0, x_T\}$，シーケンスを $Y = \{y_0, y_1, \ldots, y_T\}$，推定する物体の状態を $X = \{x_2, \ldots, x_{T-1}\}$ としたとき，その事後確率を以下の式で表現する．

$$\begin{aligned} P(X|Y, x_1, x_T) &= \frac{1}{Z} \prod_{i=2}^{T-1} \psi(y_i|x_i, x_1, x_T) \\ & \quad \prod_{i=2}^{T-1} \psi(x_i, x_{i+1}) \end{aligned} \tag{4.55}$$

ここで，$\psi(y_i|x_i, x_1, x_T)$ は小区間の軌跡で次式で表現する．

$$\begin{aligned} \psi(y_i|x_i, x_1, x_T) &\sim \exp(-\min(B(h(x_i), h(x_1)), \\ & \quad B(h(x_i), h(x_T)))/2\sigma_h^2) \end{aligned} \tag{4.56}$$

σ_h^2 は分散パラメータである．また，$\psi(x_i, x_{i+1})$ はポテンシャル関数で次式で表現する．

$$\psi(y_i|x_i, x_1, x_T) \sim \exp(-D(x_i, x_{i+1})/2\sigma_p^2) \tag{4.57}$$

$$\begin{aligned} D(x_i, x_{i+1}) &= \|p_i - p_{i+1}\| + \\ & \quad \beta \|s_i - s_{i+1}\| \end{aligned} \tag{4.58}$$

ここで，σ_p^2 は分散パラメータ，β は重みである．この事後確率を最大化することで物体の軌跡を求める．

はじめに，各フレームごとに，状態の絞込みを行う．各状態に対して 式 (4.56) を計算し，それらを Mean Shift クラスタリング [22] により候補点を選び出す (図 4.22)．次に，選び出された候補

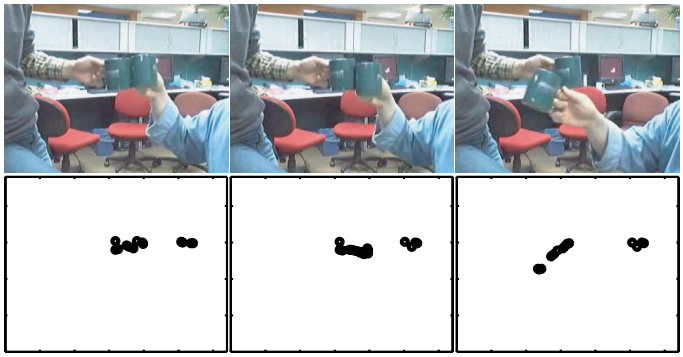


図 4.22: Mean Shift クラスタリングによる候補点 (文献 [20] より)

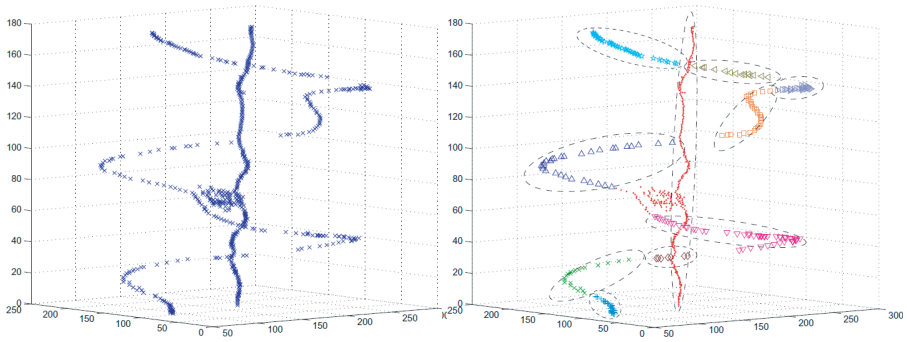


図 4.23: 部分軌跡の作成 (文献 [20] より)

点に対して，spectral clustering により部分軌跡を作成する (図 4.23)．作成された部分軌跡の内，オクルージョン領域を B-スプラインにより補間することで，対象物体の軌跡を求める．図 4.24 に，オクルージョンがある場合での追跡例を示す．

図 4.24: 追跡結果 (文献 [20] より)

4.11.2 Graph Cuts による対象領域の抽出

動画像全体をグラフで表現し，エネルギー関数の最小化により最適化することで物体領域を時系列に抽出する手法として，Interactive Graph Cuts [21] [23] が提案されている．Interactive Graph Cuts は，本来はセグメンテーション手法であり，対象物体を指定することで，動画像中の対象物体領域をボリュームデータとして抜き出すことが可能である．そのため，物体追跡として利用することができる．

Graph Cuts では，物体か背景というラベルに対するエネルギー関数を定義し，これを最小化するようなラベルを付けることにより最適化を行う手法である．画像 P に対する各ピクセルを $p \in P$ としたとき，ラベルを $\boldsymbol{L} = \{L_1, L_2, \ldots, L_p, \ldots, L_{|P|}\}$ とし，各 L_p には物体 (“obj”) か背景 (“bkg”) のラベルが与えられる．また，p の近傍ピクセルを $q \in N$ とする．このとき，エネルギー関数は以下のように定義される．

$$E(\boldsymbol{L}) \quad = \quad \lambda \cdot R(\boldsymbol{L}) + B\boldsymbol{L}) \tag{4.59}$$

ここで，$R(\boldsymbol{L})$ は領域情報 (t-link)，$B(\boldsymbol{L})$ は境界情報 (n-link)，λ は $R(\boldsymbol{L})$ のパラメータ係数である．このエネルギー関数を Graph Cuts Algorithm を用いて最小化することで，各ピクセルにラベル付けを行う．具体的には，図 4.25 のようにグラフを作成し，このグラフに対して max flow / min cut algorithm によりグラフを分割することで求める．

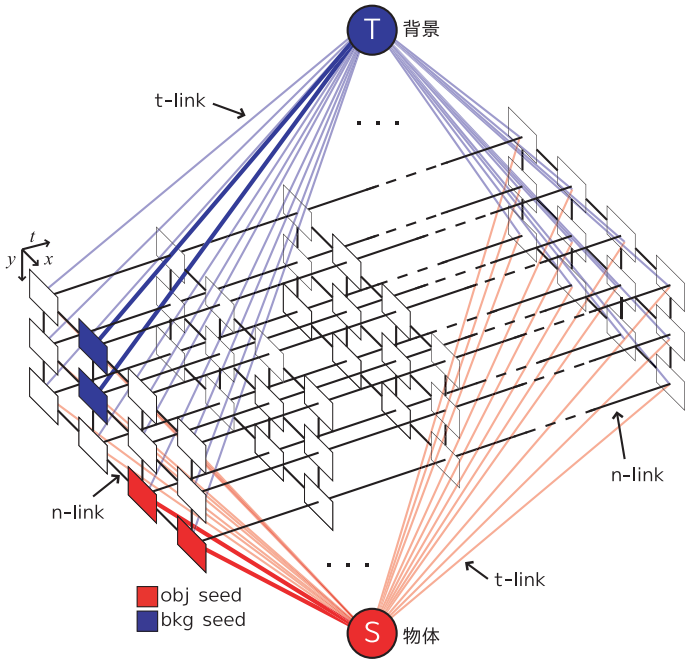


図 4.25: グラフの作成

しかし，動画像に対してグラフを作成した場合，各ピクセルをノードとするため，大量のノードが必要となる．そのため，長時間のシーケンスには対応することができない，計算コストの増加などの問題が生じる．このような問題に対しての一般的な解決法として，前処理で，各ピクセルをクラスタリングすることでスーパーピクセルを作成し，それらを 1 つのノードとする手法が用いられる．図 4.26 に，Graph Cuts による対象物体を抽出している例を示す．

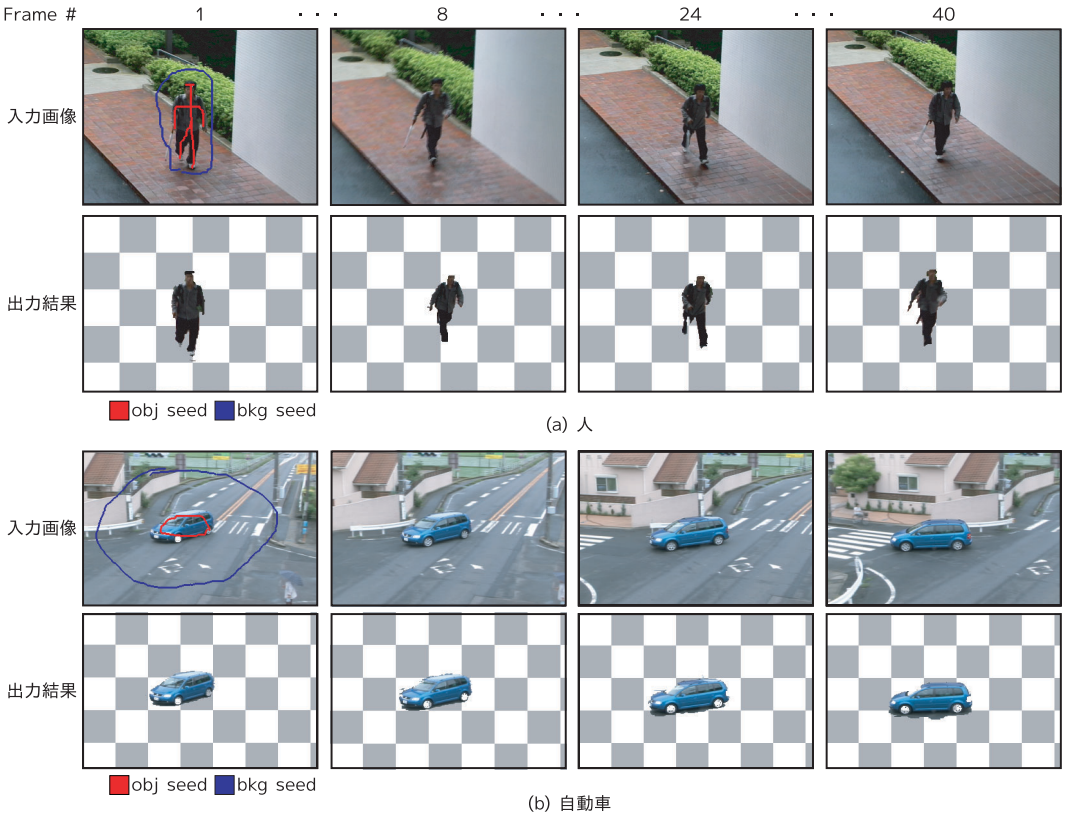


図 4.26: Graph Cuts による物体抽出例

参考文献

[1] A. Lipton, H. Fujiyoshi and R. Patil: “Moving Target Detection and Classification from Real-time Video”, Proc. of the 1998 Workshop on Applications of Computer Vision, pp. 8–14 (Oct. 1998).

[2] D. Comaniciu, V. Ramesh, and P. Meer: “Real-time Tracking of Non-rigid Objects using Mean Shift”, in IEEE Computer Vision and Pattern Recognition, pp. II:142-149 (2000).

[3] R. Collins: “Mean-shift Bolb Tracking Through Scale Space”, in IEEE Computer Vision and Pattern Recognition (Jun. 2003).

[4] R. Collins,O. Amidi, and T. Kanade: “An Active Camera System for Acquiring Multi-view Video”, Proceedings of the 2002 International Conference on Image Processing (ICIP ’02), pp. 517 - 520 (Sep. 2002).

[5] “ディジタル画像処理”, 財団法人 画像情報教育振興協会 (CG-ARTS 協会), 2004.

[6] 安居院猛, 長尾智晴 : “画像の処理と認識”, 昭晃堂, 1992.

[7] 村瀬洋, V. V. Vinod : “局所色情報を用いた高速物体探索 ーアクティブ探索法ー”, 電子情報通信学会論文誌 D-II, Vol. J81 D-II, No.9, pp.2035-2042, 1998.

[8] K. She, G. Babis, H. Gu, R. Miller, “Vehicle Tracking using On-line Fusion of Color and Shape Features”, IEEE Conference on Intelligent Transportation System, Oct 2004.

[9] 小関亮介，箕浦良文，藤吉弘亘，秋田時彦，柿並俊明，“協調的な複数の Mean-Shift トラッカによる後方車両追跡”，画像の理解・認識シンポジウム (MIRU), pp.419-426, July 2005

[10] 加藤丈和: “パーティクルフィルタとその実装法”, 情報処理学会 コンピュータビジョンとイメージメディア研究会報告 2007-CVIM-157-22, pp. 161-168, Jan 2007.

[11] Isard, M. and Blake, A.: “CONDENSATION - Conditional Density Propagation for Visual Tracking, International Journal of Computer Vision, Vol. 29, No. 1, pp. 5-29, 1998.

[12] K. Okuma, A. Taleghani, N. Freitas, J. Little and D. Lowe, “A Boosted Particle Filter: Multitarget Detection and Tracking”, European Conference on Computer Vision, Vol. 3021 of LNCS, pages 28–39. Springer, 2004.

[13] Jianbo Shi, Carlo Tomasi: “Good Features to Track” , 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR’94), 1994, pp. 593 - 600.

[14] Carlo Tomasi, Takeo Kanade: “Detection and Tracking of Point Features” , CMU Tech. Rep. CMU-CS-91-132, April 1991.

[15] Bruce D. Lucas, T Kanade: “An Iterative Image Registration Technique with an Application to Stereo Vision” , Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI ’81), April, 1981, pp. 674-679.

[16] 深尾隆則, 金出武雄: “2 段階特徴点追従アルゴリズム” , 情報処理学会研究報告, 2003-CVIM-141, pp. 103-110.

[17] 加藤丈和, 深尾隆則, 羽下哲司: “対象追跡 -フレーム間の類似度に着目した手法から動きのモデルに着目した手法まで-” , 情報処理学会研究報告, 2005-CVIM-150, pp. 185-198.

[18] 金澤靖, 金谷健一,: “コンピュータビジョンのための画像の特徴点の抽出” , 電子情報通信学会誌, vol.87, no.12, pp.1043–1048, 2004 年 12 月.

[19] 都築勇司, 藤吉弘亘, 金出武雄: “SIFT 特徴量に基ずく Mean-Shift 探索による特徴点追跡” , 情報処理学会論文誌, vol.49, No.SIG6, pp.35-45, 2008.

[20] Jian Sun, Weiwei Zhang, Xiaoou Tang, and Heung-Yeung Shum: “Bi-directional Tracking using Trajectory Segment Analysis” , In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV’05), Vol.1, pp.717-724, 2005.

[21] Yuri Boykov and Gareth Funka-Lea: “Graph Cuts and Effecient n-d Image Segmentation” , Int. J. Comput. Vision, Vol.70, No.2, pp.109-131, 2006.

[22] Y. Cheng, “Mean Shift, Mode Seeking, and Clustering” ,IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.17, pp.790-799, 1995.

[23] Yuri Boykov and Marie-Pierre Jolly: “Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in n-d Images” , ICCV2001, Vol.1, p.105, 2001.

第5章　画像から実空間へのマッピング:Mapping to world coordinate

検出した対象物の世界座標系における位置を求めるため，通常 2 台以上のカメラを用いたステレオ視を用いる．しかし，屋外等におけるカメラの設置やコストの面から，1 台のカメラが望ましい．そこで本章では，検出・追跡された移動体の画像座標から実空間座標 (世界座標) での位置情報を得る手法として，平面射影行列 [1] とカメラキャリブレーションにより得られたカメラパラメータを用いた光線交差法について述べる．

5.1　平面射影変換によるマッピング

5.1.1　平面射影行列 (Homography)

3 次元空間中の物体の位置を求める手法として，射影変換を用いた手法がある．この手法では，2 つの平面間における対応する点を採取し，その対応点群から平面射影行列 H を求める．この平面射影行列を用いることにより，一方の平面上の点に対応したもう一方の平面上の座標を計算することが可能となる．

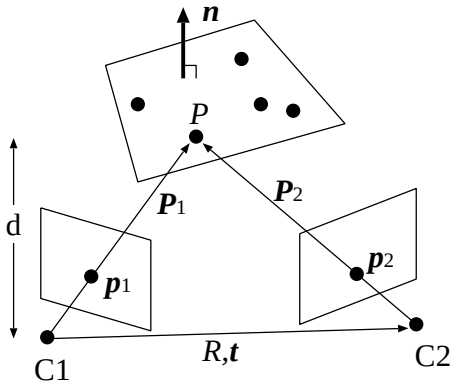


図 5.1: 平面射影変換の概念

図 5.1 に 3 次元空間とカメラ位置の関係を示す．ここでは，2 台のカメラ間の平面射影行列を求める手法について述べる．カメラ C1 のカメラ座標を基準とする．この平面のカメラ C1 のカメラ

座標での法線ベクトルを $\boldsymbol{n}$，カメラ中心 C1 からの距離，すなわち C1 からこの平面に下ろした垂線の長さを d とする．

この平面上の点 P を指すカメラ C1 のカメラ座標での位置ベクトルを $\boldsymbol{P}_1$ とすると

$$\boldsymbol{n}^\top \boldsymbol{P}_1 = d \tag{5.1}$$

である．

C1 から見た第二のカメラ C2 のカメラ座標への回転を R，平行移動を $\boldsymbol{t}$ とすると，点 P を指すカメラ C2 のカメラ座標での位置ベクトル $\boldsymbol{P}_2$ は，$\boldsymbol{P}_2 = R\boldsymbol{P}_1 + \boldsymbol{t}$ であるので，$\boldsymbol{n}^\top \boldsymbol{P}_1/d = 1$ であることを用いると次式が成り立つ．

$$\begin{aligned} \boldsymbol{P}_2 &= R\boldsymbol{P}_1 + \boldsymbol{t}\frac{\boldsymbol{n}^\top \boldsymbol{P}_1}{d} && (5.2)\\ &= \left[R + \frac{\boldsymbol{t}\boldsymbol{n}^\top}{d}\right]\boldsymbol{P}_1 && (5.3) \end{aligned}$$

点 P のそれぞれのカメラでの画像点のカメラ座標は，$\boldsymbol{p}_1 = \boldsymbol{P}_1/z_1$，$\boldsymbol{p}_2 = \boldsymbol{P}_2/z_2$ で与えらので，

$$\frac{z_2}{z_1}\boldsymbol{p}_2 = \left[R + \frac{\boldsymbol{t}\boldsymbol{n}^\top}{d}\right]\boldsymbol{p}_1 \tag{5.4}$$

このそれぞれの点の画像座標を (u_1, v_1)，(u_2, v_2) とすれば，この式は次の様に表すことができる．

$$\alpha \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{5.5}$$

従って，この平面上に存在する点の夫々の画像への投影点の画像座標の組はすべて，式 (5.5) で表すことができ，平面射影行列が既知であれば，ステレオ画像の一方の画像点からもう一方の画像上の対応点の位置が決定できる．

5.1.2　平面射影行列の決定 【Source Code】

3 次元空間内のある平面上に存在する n 個の点について，二つの画像間で対応が与えられているとする．画像 1 における i 番目の点を (u_{1i}, v_{1i})，同様に画像 2 における i 番目の点を (u_{2i}, v_{2i}) とする．

さらに，この点についての式 (5.5) の定数 α を α_i とおけば

$$\alpha_i \begin{bmatrix} u_{2i} \\ v_{2i} \\ 1 \end{bmatrix} = H \begin{bmatrix} u_{1i} \\ v_{1i} \\ 1 \end{bmatrix} \tag{5.6}$$

は，H の (ij) 成分を H_{ij} と表し，$\boldsymbol{h} = (H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33})^\top$ とすることで，次のように書ける．

$$\left[\begin{array}{cccccccc|c} u_{1i} & v_{1i} & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & u_{1i} & v_{1i} & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & u_{1i} & v_{1i} & 1 \end{array}\right| \left.\begin{array}{c} -u_{2i} \\ -v_{2i} \\ -1 \end{array}\right] \begin{bmatrix} \boldsymbol{h} \\ \alpha_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.7}$$

左辺の行列の左の 3×9 の部分 U_i，残りの 1 列を-$\boldsymbol{u}_i$ で表すと，n 個の対応点が取れている場合，$\boldsymbol{h}$ は次式を満たすことになる．

$$\begin{bmatrix} U_1 & -u_{21} & 0 & \cdots & \cdots & \cdots & 0 \\ U_2 & 0 & -u_{22} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ U_n & 0 & \cdots & \cdots & \cdots & 0 & -u_{2n} \end{bmatrix} \begin{bmatrix} \boldsymbol{h} \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = (0) \tag{5.8}$$

これを解いて $\boldsymbol{h}$ を求め，平面射影行列 H を得る．式 (5.8) を展開すると以下のようになる．

$$\begin{cases} u_{11}H_{11} + v_{11}H_{12} + H_{13} - u_{21}\alpha_1 = 0 \\ u_{11}H_{21} + v_{11}H_{22} + H_{23} - v_{21}\alpha_1 = 0 \\ u_{11}H_{31} + v_{11}H_{32} + H_{33} - \quad \alpha_1 = 0 \\ u_{12}H_{11} + v_{12}H_{12} + H_{13} - u_{22}\alpha_2 = 0 \\ u_{12}H_{21} + v_{12}H_{22} + H_{23} - v_{22}\alpha_2 = 0 \\ u_{12}H_{31} + v_{12}H_{32} + H_{33} - \quad \alpha_2 = 0 \\ \qquad\qquad \vdots \\ u_{1i}H_{11} + v_{1i}H_{12} + H_{13} - u_{2i}\alpha_i = 0 \\ u_{1i}H_{21} + v_{1i}H_{22} + H_{23} - v_{2i}\alpha_i = 0 \\ u_{1i}H_{31} + v_{1i}H_{32} + H_{33} - \quad \alpha_i = 0 \end{cases} \tag{5.9}$$

このとき，$\alpha_i = u_{1i}H_{31} + v_{1i}H_{32} + H_{33}$ となり，式 (5.9) は α を消去した式へ変更が可能である．

$$\begin{cases} u_{11}H_{11} + v_{11}H_{12} + H_{13} - u_{21}u_{11}H_{31} - u_{21}v_{11}H_{32} - u_{21}H_{33} = 0 \\ u_{11}H_{21} + v_{11}H_{22} + H_{23} - v_{21}u_{11}H_{31} - v_{21}v_{11}H_{32} - v_{21}H_{33} = 0 \\ u_{12}H_{11} + v_{12}H_{12} + H_{13} - u_{22}u_{12}H_{31} - u_{22}v_{12}H_{32} - u_{22}H_{33} = 0 \\ u_{12}H_{21} + v_{12}H_{22} + H_{23} - v_{22}u_{12}H_{31} - v_{22}v_{12}H_{32} - v_{22}H_{33} = 0 \\ \qquad\qquad \vdots \\ u_{1i}H_{11} + v_{1i}H_{12} + H_{13} - u_{2i}u_{1i}H_{31} - u_{2i}v_{1i}H_{32} - u_{2i}H_{33} = 0 \\ u_{1i}H_{21} + v_{1i}H_{22} + H_{23} - v_{2i}u_{1i}H_{31} - v_{2i}v_{1i}H_{32} - v_{2i}H_{33} = 0 \end{cases} \tag{5.10}$$

そして，この式を$\boldsymbol{A}\ \boldsymbol{h} = \boldsymbol{0}$の形の行列式にすると，

$$\begin{bmatrix} u_{11} & v_{11} & 1 & 0 & 0 & 0 & -u_{21}u_{11} & -u_{21}v_{11} & -u_{21} \\ 0 & 0 & 0 & u_{11} & v_{11} & 1 & -v_{21}u_{11} & -v_{21}v_{11} & -v_{21} \\ u_{12} & v_{12} & 1 & 0 & 0 & 0 & -u_{22}u_{12} & -u_{22}v_{12} & -u_{22} \\ 0 & 0 & 0 & u_{12} & v_{12} & 1 & -v_{22}u_{12} & -v_{22}v_{12} & -v_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{1i} & v_{1i} & 1 & 0 & 0 & 0 & -u_{2i}u_{1i} & -u_{2i}v_{1i} & -u_{2i} \\ 0 & 0 & 0 & u_{1i} & v_{1i} & 1 & -v_{2i}u_{1i} & -v_{2i}v_{1i} & -v_{2i} \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} = (\boldsymbol{0}) \tag{5.11}$$

(5.12)

が得られる．$\boldsymbol{h}$ は $\boldsymbol{A}^T\boldsymbol{A}$ の最小固有値に対応する固有ベクトルとして求められる．

5.1.3　射影変換行列による実世界へのマッピング 【Source Code】

射影変換行列を求める際に，8 点以上の対応点が必要である．図 5.2 に示した 640×480 画素の画像内の床平面と，世界座標系の上から見たマップ平面への射影行列を求めるために，実際に用いた画像座標と世界座標の対応点群を以下に示す．これらの対応点より，前節で述べた手法を用いて射影変換行列を求める．平面射影行列 $\boldsymbol{h}$ を求めるには，$\boldsymbol{A}^T\boldsymbol{A}$ の最小固有値に対応する固有ベクトルを求めるプログラムを作成する必要があるが，金谷らにより，より精度の良い H の推定法 [2] が提案され，射影変換行列の推定プログラムが提供されている [3].

対応点群 (123 点)

u	v	x	y	u	v	x	y	u	v	x	y	u	v	x	y
449	111	1000	500	433	247	1500	6500	355	179	2500	4500	303	146	3500	3000
450	117	1000	1000	434	271	1500	7000	351	191	2500	5000	297	156	3500	3500
453	123	1000	1500	435	302	1500	7500	346	208	2500	5500	290	166	3500	4000
454	130	1000	2000	435	336	1500	8000	341	224	2500	6000	282	179	3500	4500
456	138	1000	2500	436	379	1500	8500	334	247	2500	6500	274	191	3500	5000
458	146	1000	3000	436	430	1500	9000	327	270	2500	7000	263	208	3500	5500
460	156	1000	3500	399	117	2000	1000	318	300	2500	7500	253	224	3500	6000
463	166	1000	4000	399	123	2000	1500	307	333	2500	8000	240	245	3500	6500
466	179	1000	4500	398	130	2000	2000	295	374	2500	8500	226	268	3500	7000
469	192	1000	5000	396	137	2000	2500	280	422	2500	9000	291	124	4000	1500
473	209	1000	5500	396	145	2000	3000	347	117	3000	1000	285	131	4000	2000
477	226	1000	6000	394	155	2000	3500	344	124	3000	1500	279	140	4000	2500
481	248	1000	6500	393	165	2000	4000	341	130	3000	2000	272	148	4000	3000
486	272	1000	7000	391	178	2000	4500	336	139	3000	2500	264	157	4000	3500
492	303	1000	7500	390	190	2000	5000	333	146	3000	3000	257	166	4000	4000
498	336	1000	8000	387	208	2000	5500	328	156	3000	3500	246	179	4000	4500
506	380	1000	8500	385	225	2000	6000	323	165	3000	4000	237	192	4000	5000
512	429	1000	9000	382	247	2000	6500	317	179	3000	4500	224	208	4000	5500
425	111	1500	500	379	270	2000	7000	310	191	3000	5000	211	224	4000	6000
425	117	1500	1000	375	301	2000	7500	302	208	3000	5500	195	245	4000	6500
426	123	1500	1500	370	334	2000	8000	295	224	3000	6000	179	266	4000	7000
426	130	1500	2000	363	377	2000	8500	284	246	3000	6500	270	124	4500	1500
427	138	1500	2500	356	426	2000	9000	274	268	3000	7000	261	132	4500	2000
428	147	1500	3000	373	116	2500	1000	261	297	3000	7500	254	140	4500	2500
429	156	1500	3500	372	123	2500	1500	246	329	3000	8000	245	148	4500	3000
429	166	1500	4000	370	130	2500	2000	228	369	3000	8500	235	158	4500	3500
430	179	1500	4500	368	137	2500	2500	322	117	3500	1000	226	167	4500	4000
431	192	1500	5000	365	145	2500	3000	319	124	3500	1500	214	180	4500	4500
432	208	1500	5500	362	155	2500	3500	314	131	3500	2000	203	192	4500	5000
432	225	1500	6000	359	165	2500	4000	308	139	3500	2500	188	209	4500	5500
												175	224	4500	6000
												156	245	4500	6500
												139	266	4500	7000

平面射影行列を求める際には，最低 8 点が必要であるが，対応点を自動抽出した場合，観測誤差や外れ値 (outlier) が含まれることがある．このような場合は，多数対応点セットからランダムに 8 対応点を選び平面射影行列を求め，各対応点における変換誤差を評価し，最も誤差が小さくなる H を求める RANSAC(Random Sample Consensus) 法を用いると良い．

平面射影行列

```
-0.2009162849677137 0.11490048622241648 87.57374592297015
0.012018491810747465 0.9095423573851311 -100.09895045679708
2.5415015989855735e-6 7.130851020567426e-5 0.0012054633141478372
```

式 (5.5) を展開し，対応点を世界座標 $(x_w, y_w, z_w = 0)$ とし，α を消去すると次式のように変換できる．

$$x_w = \frac{uH_{11} + vH_{12} + H_{13}}{uH_{31} + vH_{32} + H_{33}}, \quad y_w = \frac{uH_{21} + vH_{22} + H_{23}}{uH_{31} + vH_{32} + H_{33}} \tag{5.13}$$

式 (5.13) に，観測した画像座標 (u, v) を入力し，対応点から予め求めておいた平面射影行列を用いて，画像座標 (u, v) に対応する世界座標 (x_w, y_w) を求めることにより，図 5.2 のように画像座標を世界座標へマッピングすることが可能となる．

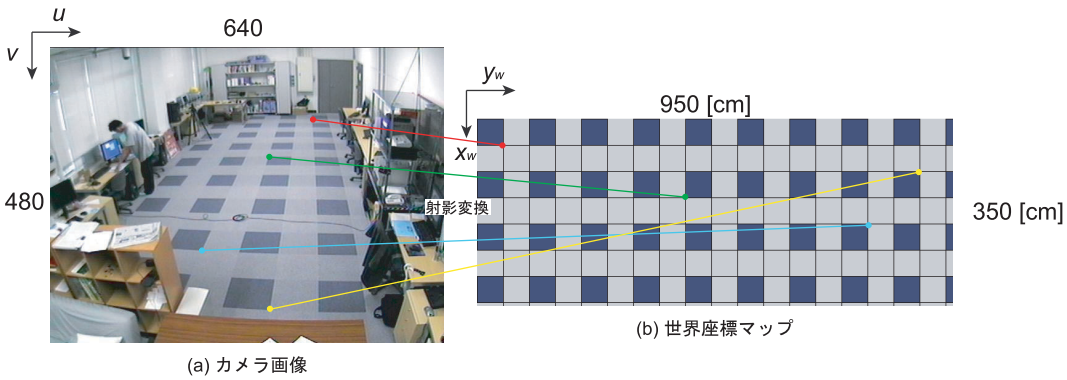


図 5.2: 世界座標マップへのマッピング

5.2　光線交差法によるマッピング

5.2.1　カメラキャリブレーション

カメラキャリブレーションとは，実世界のカメラとコンピュータ内部に設定したカメラモデルとの対応付けを行い，各種のカメラパラメータを決定することをいう．図 5.3 にピンホールカメラモデルを示す．画像平面 I から距離 f のところに I に平行な面 F を置き，その上の点 c にピンホールをあける．物体からくる光はピンホール，すなわち点 c を通り，画像平面に像を結ぶ．物体の点とピンホール，画像平面上の像は一直線上にある．このような射影は中心射影と呼ばれる．点 c をレンズ中心，または，焦点，面 F を焦点面，レンズ中心から画像平面までの距離 f を焦点距離という．点 c を通り画像平面に垂直な線を光軸といい，それと画像平面との交点 C を光軸点という．

空間中の点の 3 次元座標とその 2 次元像との間に，以下の関係が成り立つ．

$$x = f\frac{x}{z}, \qquad y = f\frac{y}{z} \tag{5.14}$$

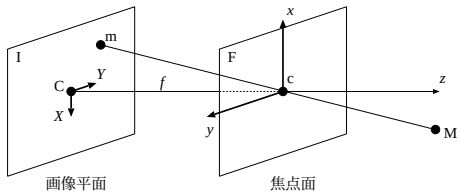


図 5.3: ピンホールカメラモデル

図 5.4 に示すように，焦点面の後ろにある画像面を，焦点の前に移動し，反転させても式 (5.14) の関係が成り立つ．コンピュータビジョンでは，この図が一般的に用いられている．

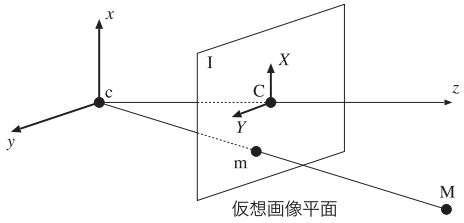


図 5.4: 仮想の画像平面をレンズ中心に置いたピンホールカメラモデル

今回のカメラキャリブレーションは Tsai の提案した手法で行う [5]．このとき，世界座標 (x_w, y_w, z_w) とカメラ座標 (x, y, z) の関係は次式で表される．$\boldsymbol{R}$ は 3 行 3 列の回転行列，$\boldsymbol{T}$ は平行移動ベクトル (T_x, T_y, T_z) である．

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{R}\left(\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} - \boldsymbol{T} \right) \tag{5.15}$$

カメラモデルを図 5.5 に示す．カメラキャリブレーションは，以下に示すカメラの内部・外部パラメータを求めることである．

- 内部パラメータ:
  焦点距離 f
  画像座標の中心 $O(u_c, v_c)$
  レンズ歪み係数 k_1，k_2
  画像座標の縦横比 S
- 外部パラメータ:
  平行移動ベクトル $\boldsymbol{T}$
  回転ベクトル $\boldsymbol{R}$

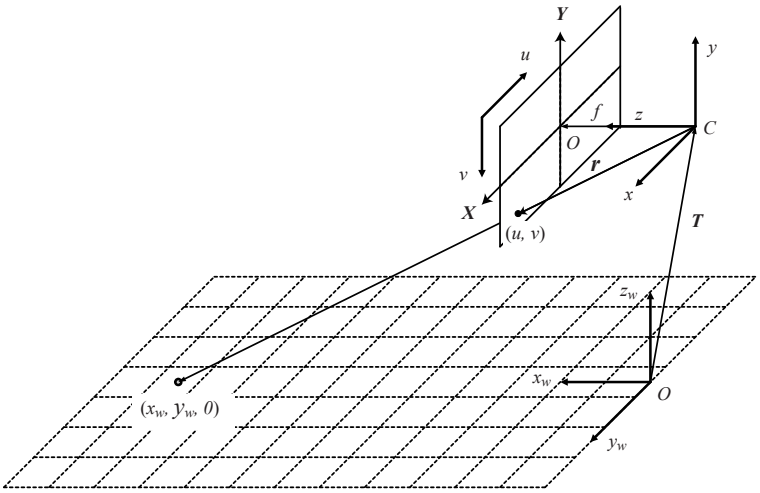


図 5.5: カメラモデルと各座標系

ここでは，カメラの内部・外部パラメータの求め方については，文献 [4, 5, 6] を参照してほしい．また，カメラキャリブレーションプログラムが WEB 上に提供されている．

5.2.2　カメラパラメータによる光線交差法

以下にカメラキャリブレーションにより得られたカメラパラメータによる光線交差法による 3 次元位置を求める手順を示す．求める世界座標は床面上 ($z_w = 0$) の点とする．

Step1 画像座標 $X - Y$ 上の点 (X_d, Y_d) は以下の式で表される．(u_c, v_c) は画像座標の中心である．

$$X_d = u_c - u, \qquad Y_d = v_c - v \tag{5.16}$$

Step2 レンズ歪みを修正した点 (X_u, Y_u) は次式となる．r は画像中心からの距離である．

$$X_u = SX_d(1 + k_1 r^2 + k_2 r^4), \qquad Y_u = Y_d(1 + k_1 r^2 + k_2 r^4) \tag{5.17}$$

Step3 図 5.5 において，カメラ座標の原点から画像上の任意点 p に通過するベクトル (x_v, y_v, z_v) は以下の式となる．

$$\begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \boldsymbol{R}^\top \begin{bmatrix} X_u \\ Y_u \\ f \end{bmatrix} \tag{5.18}$$

Step4 これを 3 次元空間上の直線式で表すと以下のようになる．

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \alpha \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} + \boldsymbol{T} \tag{5.19}$$

Step5 ここで地面が $x_w - y_w$ 平面に平行であり，$z_w = 0$ であると仮定すると，直線が平面と交わるときの α を決定できる．

$$\alpha = -\frac{T_z}{z_v} \tag{5.20}$$

Step6 α の値を式 5.19 に代入することで，画像座標点 (u, v) に対応した世界座標点 $(x_w, y_w, 0)$ を求めることができる．

$$x_w = -\frac{T_z}{z_v}x_v + T_x, \qquad y_w = -\frac{T_z}{z_v}y_v + T_y \tag{5.21}$$

5.3 実空間座標へのマッピング結果

5.3.1 平面射影変換とカメラパラメータによる光線交差法の比較

図 5.6 にキャリブレーションデータの (u, v) より位置を推定した結果を示す．また，表 5.1 に各手法の誤差を示す．表 5.1 より，カメラパラメータを用いた光線交差法は，カメラレンズ歪みを考慮した位置推定を行うため，精度の良い推定が可能であることがわかる．

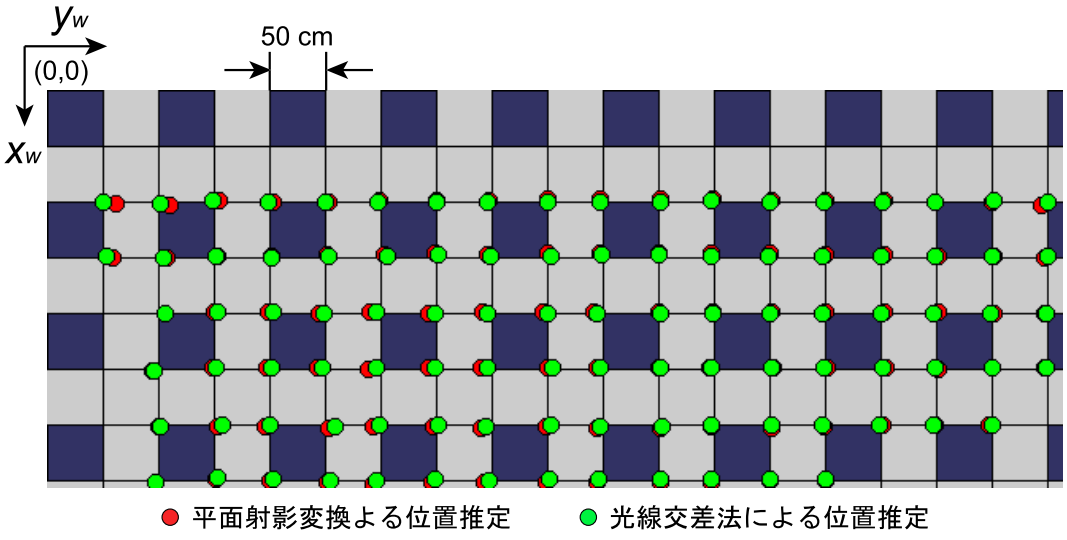


図 5.6: キャリブレーションデータ座標を推定した結果

5.3.2 位置推定精度の検証

このように，カメラキャリブレーション時に用いた特徴点に対しては位置推定誤差を求めることができるが，それ以外の点は対応する世界座標の真値が分からないため，誤差を求めることができない．そこで本研究では，図 5.7 のように，画像座標 (u, v) が 4 近傍に 1 画素ずれた際に対応する

表 5.1: 各手法による推定誤差 [mm]

	誤差
平面射影変換	39.3
光線交差法	25.3

世界座標点 $(x_w, y_w, 0)$ 間のユークリッド距離の平均値を求め，これを空間分解能とし，推定誤差の指標とする．この空間分解能を，画像上で床面に相当する領域について算出した結果を図 5.8 に示す．図 5.8 より，カメラからの距離が離れるほど空間分解能が低くなり，最大で 1 画素のずれが約 80[mm] に対応していることがわかる．このように予めカメラの画像座標に対応した実空間の分解能を計算し，目的に応じた空間分解能が得られるか確認する必要がある．

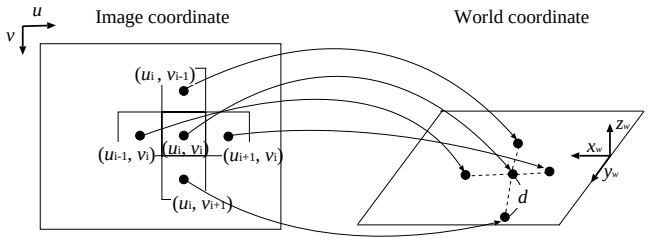


図 5.7: 空間分解能の計算法

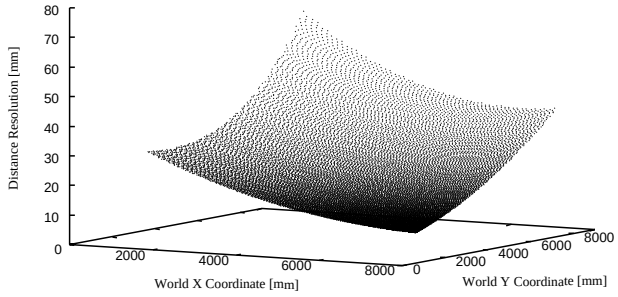


図 5.8: 空間分解能の分布

5.4　対象物の高さ計測

カメラパラメータを用いた光線交差法の応用として，対象物の高さ (身長) の推定と，高さ情報を用いたマスタースレーブによる自動追尾システムについて述べる．

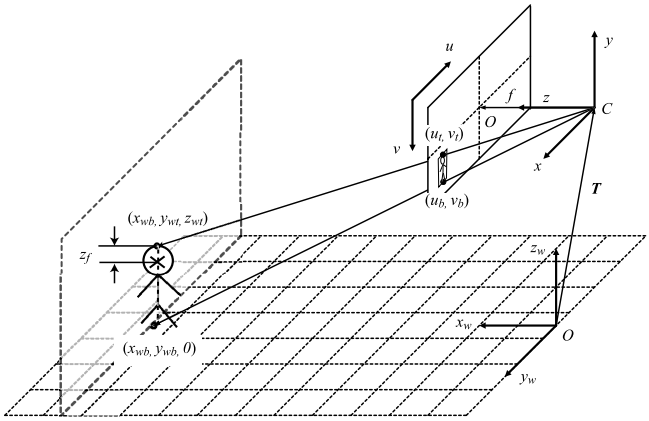


図 5.9: 高さの取得

5.4.1 光線交差法による高さ計測

5.2.2 の光線交差法により求めた床面上の位置情報を基に，$y_w - z_w$ 平面を対象とした光線交差を求める [7]．以下にそのアルゴリズムを示す (図 5.9 参照)．

Step1 検出領域の下端部中央点 (u_b, v_b) が，地面に接していると仮定し，5.2.2 で述べた手法により，対応する世界座標点 $(x_{wb}, y_{wb}, 0)$ を求める．

Step2 検出した対象物が地面から垂直に立っていると仮定し，世界座標点 $(x_{wb}, y_{wb}, 0)$ を通る $y_w - z_w$ 平面を考える．

Step3 人物の頭頂部にあたる点の画像座標点 (u_t, v_t) を通る光線を式 (5.19) より求める．ここでは，式 (5.19) における実数 α を β とする．その光線が Step2 の $x_w - z_w$ 平面と交差する β を以下より決定する．

$$\beta = \frac{(x_{wb} - T_x)}{x_v} \tag{5.22}$$

Step4 β の値を式 (5.19) に代入することで，画像座標点 (u_t, v_t) に対応した世界座標点 (x_{wb}, y_{wt}, z_{wt}) を求めることができる．

$$\begin{aligned} y_{wt} &= \frac{y_v(x_{wb} - T_x)}{x_v} + T_y, \\ z_{wt} &= \frac{z_v(x_{wb} - T_x)}{x_v} + T_z \end{aligned} \tag{5.23}$$

この z_{wt} は，検出した人物の実世界における床面からの高さであり，身長を示している．

Step5 人物の顔は頭頂部の鉛直下方向にあるので，頭頂部の世界座標点 (x_{wb}, y_{wt}, z_{wt}) から一定値 z_f 減算した世界座標点 $(x_{wt}, y_{wt}, z_{wt} - z_f)$ を，Slave カメラで狙うべき世界座標点とする．

5.4.2 高さ情報を用いたマスタースレーブによる自動追尾

マスターカメラ (固定) から得られた移動体の位置と高さ情報を基に Slave カメラを制御する．Slave カメラのパン・チルト角はカメラと対象物の位置関係から計算する．

図 5.10: 追尾カメラシステムの動作例

Slave カメラと推定された物体の位置までの距離からズームパラメータを決定する．このとき，物体が一定の大きさに映るように，物体までの距離を基にズームパラメータを調整する．さらに本システムでは，推定位置毎の空間分解能を考慮し，対象物が空間分解能の低い領域にいるときは Slave カメラ視野から物体が外れないように，ズームを少し引くようにパラメータを調節する．このように，Slave カメラのパン・チルト・ズーム量は世界座標点を基に計算するため，任意位置に設置された Slave カメラを，任意の世界座標点に向けて制御することができ，対象物を多方向から撮影することが可能となる．

移動体の自動追尾における本システムの動作例を図 5.10 に示す．図 5.10(a) が Master カメラ映像で，長方形で囲まれた領域が検出領域を示している．検出領域から推定した位置を基に制御した Slave カメラ映像を (b) と (c) に示す．(d) は対象領域を真上から見下ろしたマップであり，検出した物体の軌跡，Slave カメラの位置とその視野等の状況をリアルタイムで提示する．この動作例における対象物は，Slave カメラ 1 の方向に移動しているため，Slave カメラ 1 は顔正面を捉えるように制御されている．一方，Slave カメラ 2 は，カメラと対象物の位置関係より対象物全体を捉えるように制御されている．このように，本システムは適応的に Slave カメラの注視点を位置関係から決定するため，常に正面顔領域をズームした状態の映像を取得することが可能である．

図 5.11 に，フレーム毎の検出した物体の高さを示す．また，その際の Slave カメラ画像を上に示す．図 5.11 より，しゃがんだ人物の高さを約 1[m] と検出していることがわかる．また，高さ情報を用いて Slave カメラを制御しているため，しゃがんだ人物の全体像を捉えることができている．このように，検出物体の実世界における高さを推定することで，対象物の身長に合わせた追尾を実現することができる．

また，しゃがんでいない状態である 200 フレームまでの高さが周期的に約 100[mm] の幅で変化していることがわかる．これは，物体がカメラに向かって歩いているため，進行方向に足を歩み出した際に，対象画像領域が大きく検出されるからである．このように，位置精度は物体検出の能力に大きく左右される．特に，複数物体が重なったときのオクルージョンや誤検出は，実際の運用上大きな問題となる．

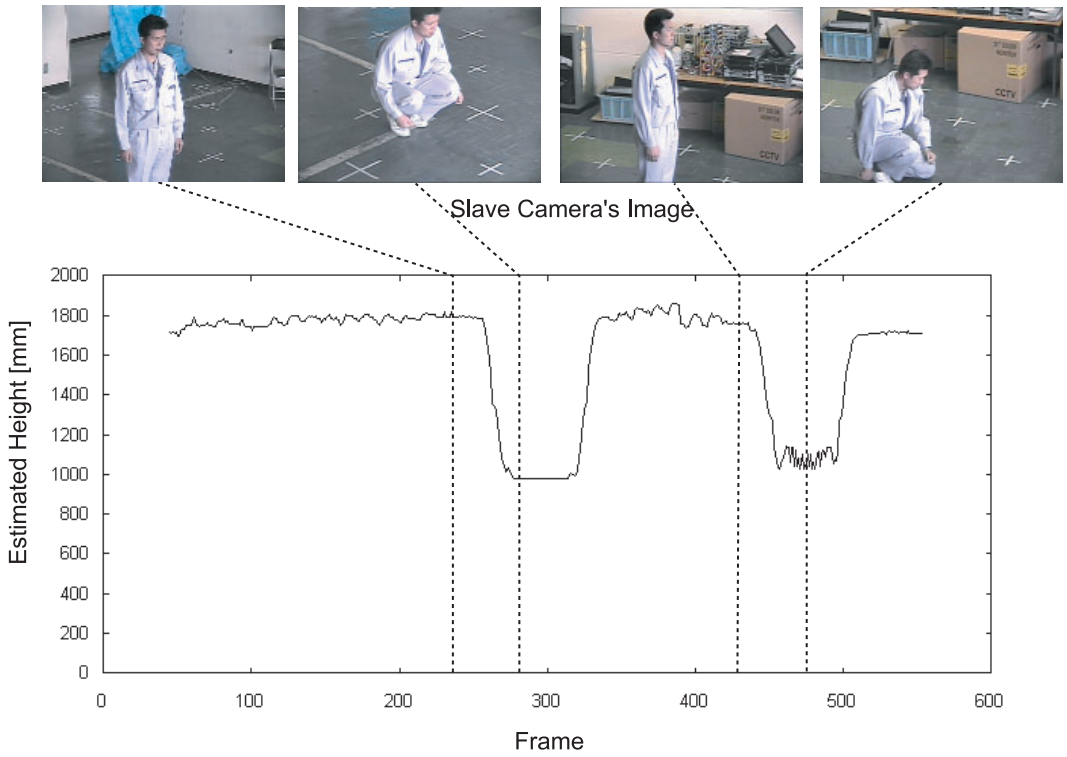

Slave Camera's Image
Estimated Height [mm]
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
Frame

図 5.11: 高さの検出例

参考文献

[1] 出口光一郎: “ロボットビジョンの基礎”, コロナ社, (2000).

[2] 清水慶行, 太田直哉, 金谷健一, “信頼性評価を備えた最適な射影変換の計算プログラム”, 情報処理学会研究報告 98-CVIM-111-5, pp. 33-40, (May. 1998).

[3] 金谷健一, 太田直哉: “幾何学的計算プログラム”:
http://www.ail.cs.gunma-u.ac.jp/Labo/program.html

[4] “A flexible new technique for camera calibration”:
http://research.microsoft.com/˜ zhang/Calib/

[5] R. Y. Tsai: “A versatile camera calibration technique for high-accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses” IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, pp. 323-344 (Aug. 1987)

[6] Tsai Camera Calibration Software:
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

[7] 小川雄三, 藤吉弘亘: “実空間に対応した Master-Slaving による追尾カメラシステム”, 第 9 回画像センシングシンポジウム, pp. 211〜216 (2003).

第6章　特徴量抽出:Feature Extraction

画像中の特徴的な点の抽出を行うことにより，画像間の対応関係 (F-matrix) の算出，高度な画像処理や画像認識を行うことが可能となる．このとき，画像中から特徴点をいかに正確に効率よく検出するかが問題となる．本章では，gradient ベースの特徴抽出法として SIFT(Scale-Invariant Feature Transform) と HOG(Histgrams of Oriented Gradients) について述べる．SIFT は，画像の拡大・縮小，回転などの変化に頑健な局所特徴量を記述するための手法である．HOG は，人検出に有効な局所領域における輝度の勾配方向をヒストグラム化した特徴として用いられている．最後に，アピアランスとモーション特徴を同時に捉える時空間特徴として Space-Time Patch について述べる．

6.1　Scale-Invariant Feature Transform(SIFT)

特徴点追跡や対応点探索などに用いられる局所的な特徴は，照明変化や画像の拡大・縮小，回転などの変化に頑健であることが望ましい．このような局所特徴を記述する手法に SIFT[1] がある．SIFT の処理は，特徴点 (以下，キーポイントと呼ぶ) の検出 (detection) と特徴量の記述 (description) の 2 段階からなり，各処理は以下の流れとなる．

detection
- 1. スケールとキーポイント検出
- 2. キーポイントのローカライズ

description
- 3. オリエンテーションの算出
- 4. 特徴量の記述

1. スケールとキーポイント検出では，DoG 処理によりスケールとキーポイントを検出し，2. キーポイントのローカライズでは，1. で検出されたキーポイントから特徴点として向かない点を削除し，その後サブピクセル推定を行う．3. オリエンテーションの算出では，回転に不変な特徴を得るためにキーポイントのオリエンテーションを求める．4. 特徴量の記述では，3. で求めたオリエンテーションに基づいてキーポイントの特徴量を記述する．以下に各処理の詳細を述べる．

6.1.1　スケールとキーポイント検出

第 1 段階のキーポイント検出では，DoG 処理を用いてスケールスペースにおける極値探索をすることで，キーポイントの位置とスケールを決定する．

LoG によるスケール探索

特徴点のスケール探索には，ガウス関数が有効であることが Koenderink[3] や Lindeberg[4] により報告されている．Lindeberg は，ガウシアンカーネルを用いたスケールスペースとして Scale-normalized Laplacian-of-Gaussian(以下，LoG と呼ぶ) を提案している．LoG は，画像にスケール σ を変化させながら式 (6.1) で表される LoG オペレータ (図 6.1) を適用し，その極大位置を特徴点のスケールと決定する．

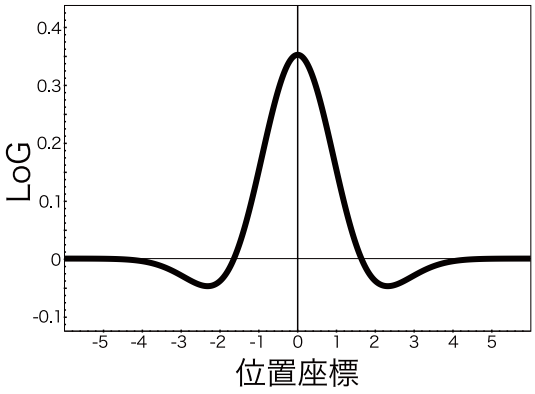


図 6.1: LoG オペレータ

$$LoG = f(\sigma) = -\frac{x^2+y^2-2\sigma^2}{2\pi\sigma^6}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{6.1}$$

σ はガウシアンフィルタのスケール，x と y は注目画素からの距離である．LoG は計算コストが高いという問題があり，Lowe[2] によってより効率的な極値検出法として Difference-of-Gaussian(DoG) を用いる手法が提案されている．DoG と LoG の関係は次式の拡散方程式から導かれる．

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G \tag{6.2}$$

G はガウス関数であり，右辺はガウス関数の 2 次微分 (LoG) である．式 (6.2) は次式のように表すことができる．

$$\frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma)-G(x,y,\sigma)}{k\sigma-\sigma} \tag{6.3}$$

式 (6.2) と式 (6.3) から次式が成り立つ．

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma)-G(x,y,\sigma)}{k\sigma-\sigma} \tag{6.4}$$

したがって次式が得られる．

$$(k-1)\sigma^2\nabla^2 G \approx G(x,y,k\sigma)-G(x,y,\sigma) \tag{6.5}$$

ここで，$\sigma^2\nabla^2 G$ は LoG であるので，式 (6.5) は DoG が LoG の近似であることを表している．LoG と DoG から得られる結果がほぼ同じであるため，SIFT では計算効率の良い DoG を用いる．

Difference-of-Gaussian 処理

キーポイント候補点は，スケールの異なるガウス関数 $G(x, y, \sigma)$ と入力画像 $I(u, v)$ を畳み込んだ平滑化画像 $L(u, v, \sigma)$ の差分 (DoG 画像) から求める．それぞれ以下の式により求める．

$$L(u, v, \sigma) = G(x, y, \sigma) * I(u, v) \tag{6.6}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{6.7}$$

DoG の結果の画像を $D(u, v, \sigma)$ とすると，DoG 画像は次式で求まる．

$$D(u, v, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(u, v)$$

$$= L(u, v, k\sigma) - L(u, v, \sigma) \tag{6.8}$$

この処理を σ_0から k 倍ずつ大きくした 異なるスケール間で行い，図 6.2 に示すような複数の DoG 画像を求める．σ が一定の割合で増加し続けると，ガウシアンフィルタのウィンドウサイズが大きくなり，処理できない端領域の拡大と計算コストの増加という問題が発生する．この問題に対し，SIFT では画像のダウンサンプリングにより σ の変化の連続性を保持した平滑化処理を実現している．

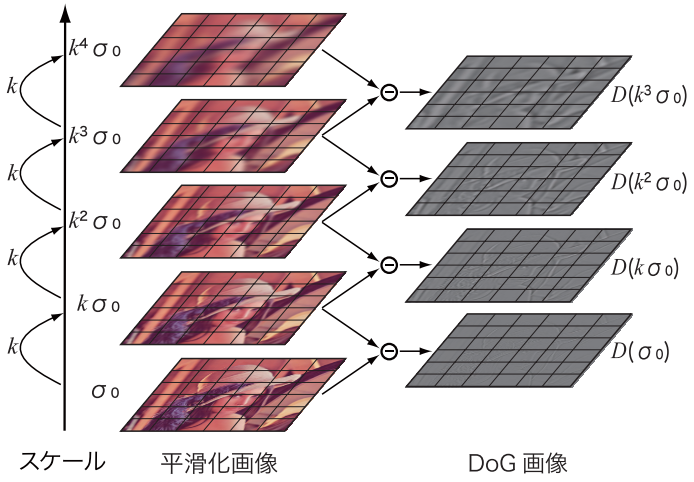


図 6.2: DoG 処理の流れ

σ の連続性を保持した平滑化処理

σ の連続性を保持した効率的な平滑化処理を図6.3 に示す．はじめに，入力画像を初期値である σ_0 で平滑化を行い，平滑化画像 $L_1(\sigma_0)$ を得る．次に σ_0 を k 倍した値 $k\sigma_0$ で平滑化を行い $L_1(k\sigma_0)$ を得る．同様の処理により，σ の異なる複数の平滑化画像を得る．ここまでの処理 1 セットを 1 オクターブとする．次に，複数生成された平滑化画像の中から $2\sigma_0$ で平滑化された画像 $L_1(2\sigma_0)$ を 1/2 のサイズにダウンサンプリングする．1 オクターブにおける処理回数については増加率 k の設

定とともに後述する．1/2 のサイズにダウンサンプリングされた画像 $L_2(\sigma_0)$ と，$2\sigma_0$ で平滑化を行った画像 $L_1(2\sigma_0)$ には以下のような関係が成り立つ．

$$L_1(2\sigma_0) \approx L_2(\sigma_0) \tag{6.9}$$

この関係を利用することで，σ の最大値を制限することができるため，ガウシアンフィルタのウィンドウサイズによる計算量の増加を防ぐことができる．

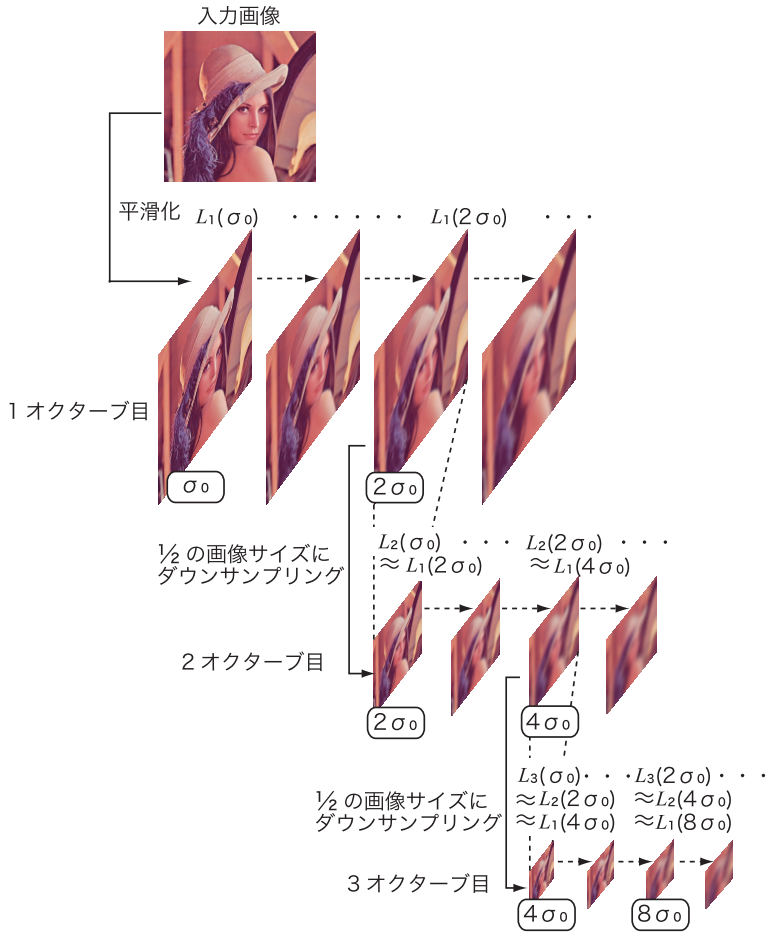


図 6.3: σ の連続性を保持した効率的な平滑化処理

増加率 k

σ の増加率 k は，1 オクターブにおけるスケールスペースの分割数により決定する．スケールスペースの分割数を s とした場合，1 オクターブでは，スケールスペースは σ_0 から $2\sigma_0$ まで増加するため，σ の増加率 k は $k = 2^{1/s}$ となる．図 6.4 に示す DoG 処理の例では，$s = 2$(分割数 2) であるため，$k = 2^{1/2} = \sqrt{2}$ となる．極値探索には DoG 画像を 3 枚 1 組で処理する必要があるため，s

枚の極値検出の対象となる画像を得るためには $s+2$ 枚の DoG 画像が必要となる．さらに，$s+2$ 枚の DoG 画像を得るためには $s+3$ 枚の平滑化画像が必要になる．したがって，1 オクターブにおける平滑化の回数は $s+3$ 回となる．ここで求まる極値検出対象画像は次章で行う極値検出に用いるものである．文献 [1] では，実験により分割数 s=3，初期値 $\sigma_0 = 1.6$ のとき，最適なキーポイントを得ることができると報告されている．

1 枚の入力画像に対するオクターブ数は，入力画像のサイズに依存する．入力画像は，処理が進むにつれて 1/2 のサイズにダウンサンプリングされる．ダウンサンプリングを続けた結果，画像の一辺のサイズがある値以下になったとき処理を終了する．この値は任意で決定する．この値が大きければ，少ないオクターブ数になり，小さければ多いオクターブ数になる．例えば 640×480 ピクセルの画像に対して，ダウンサンプリング後の最小のサイズを 10 とした場合，6 回目のダウンサンプリングで一辺が 7 ピクセルとなり処理が終了するため，オクターブ数は 6 となる．

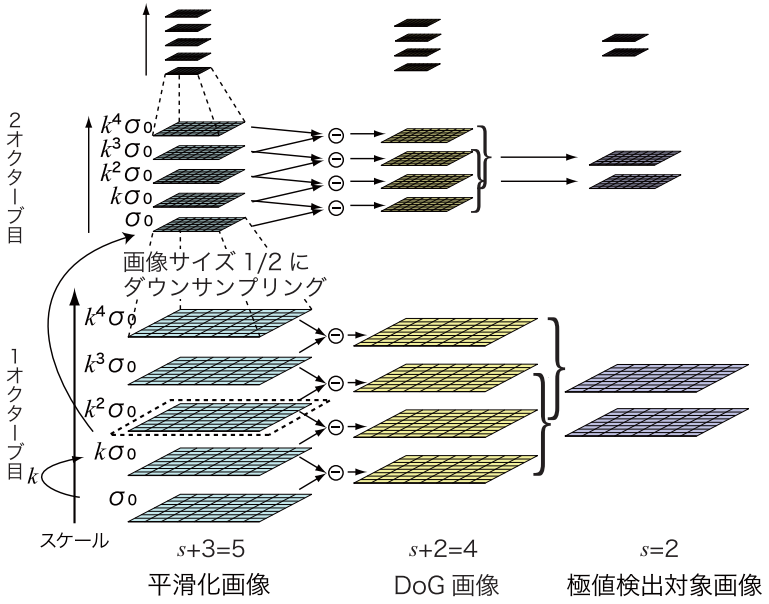


図 6.4: $s = 2(k = \sqrt{2})$ のときの DoG 処理例

DoG 画像からの極値検出

DoG は異なるスケールによる平滑化画像の差分のため，DoG の値が大きくなる σ では，スケールの変化領域にエッジ等の情報量を多く含んでいるといえる．そこで，DoG 画像から極値を検出し，キーポイントとスケールを決定する．極値の検出は，図 6.5 のように DoG 画像 3 枚一組で行う．DoG 画像 (図 6.5 中の点線で囲まれた画像) の注目画素 (図 6.5 中の黒色領域) と，その周りの 26 近傍 (図 6.5 中の灰色領域) を比較し，極値であった場合，その画素をキーポイント候補点として検出する．このような極値検出は，σ の値の小さい DoG 画像から行う．一度極値が検出された画素は，より大きなスケールで極値が検出されてもキーポイント候補点としない．この処理をスケールの異なる DoG 画像の全画素に対して行う．

次に，スケールスペースの極値の性質について述べる．画像中のある座標におけるスケール変化と DoG 出力値の推移を図 6.6 に示す．実線の円で示すスケールサイズのとき，右に示すグラフから DoG 出力が最大値 (極大値) となることがわかる．図 6.6(a) の原画像を 2 倍に拡大した (b) においても，実線で示すスケールにて DoG の値が最大となる．このとき，図 6.6(a) の DoG の極値を σ_1，図 6.6(b) を σ_2 とすると，$\sigma_2 = 2\sigma_1$ の関係が成り立つ．このように，画像サイズが 2 倍になると，DoG の極値探索により検出されたキーポイントのスケール σ も比例して 2 倍となる．SIFT は，特徴を最も含むスケール σ を自動的に決定するため，空間的に同範囲の領域から特徴量を記述することで，拡大・縮小に不変な特徴量となる．

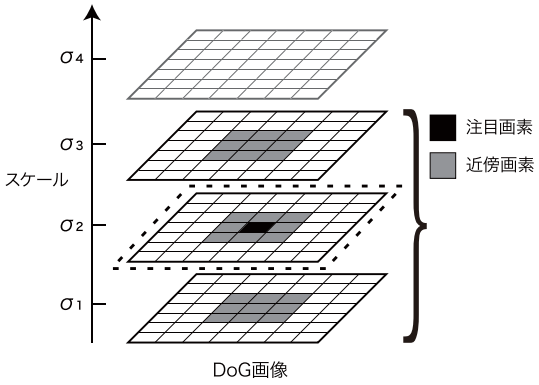


図 6.5: 極値検出の流れ

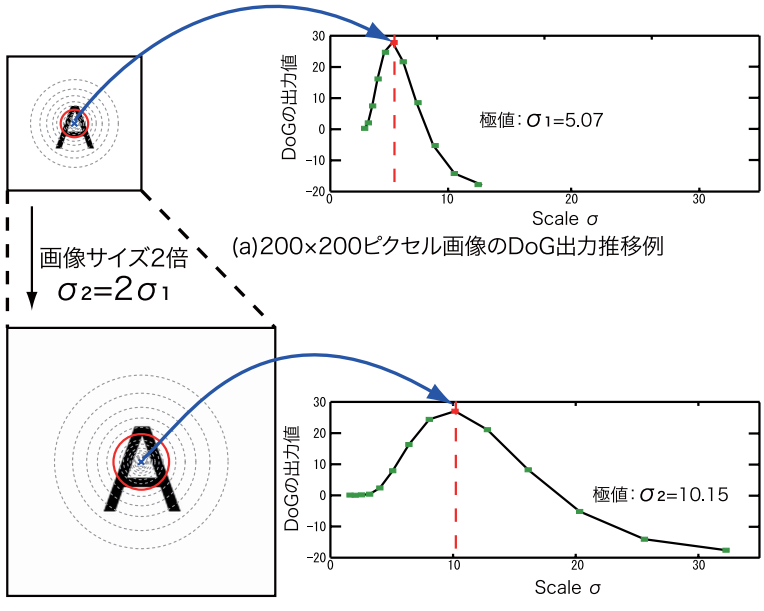
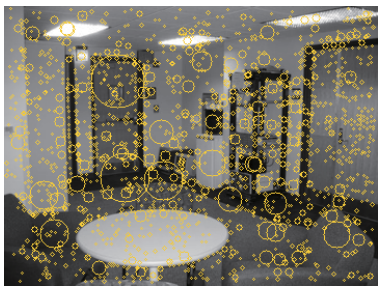
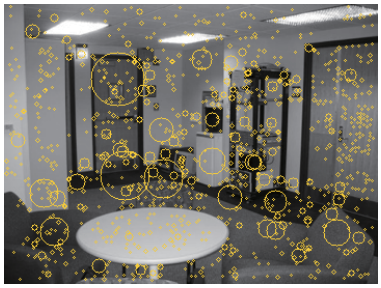


図 6.6: スケールと DoG 出力の関係



(a) 全キーポイント候補点
(キーポイント候補 1895 点)



(b) エッジ上のキーポイントを削除
(キーポイント候補 1197 点)

(c) 低コントラストのキーポイントを削除
(キーポイント候補 421 点)

図 6.7: キーポイント候補点の絞り込み

6.1.2 キーポイントのローカライズ

6.1.1 により検出されたキーポイント候補点の中には，DoG 出力値が小さい点 (low contrast) やエッジ上の点が含まれており，これらの点はノイズや開口問題に影響を受け易いという問題がある．そこで，キーポイント候補点の中から， 主曲率とコントラストにより安定したキーポイントに絞り込む．さらに，キーポイントのサブピクセル推定により位置とスケールを算出する．

主曲率によるキーポイントの絞り込み

エッジ上に存在するキーポイント候補点の削除方法について述べる．キーポイント候補点における 2 次元ヘッセ行列 $\mathbf{H}$ を次式により計算し，主曲率を求める．

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{6.10}$$

行列内の導関数は，キーポイント候補位置での DoG 出力値の 2 次微分から得られる．ここで，ヘッセ行列から求められる第 1 固有値を α，第 2 固有値を $\beta(\alpha > \beta)$ とする．このときヘッセ行列の対角成分の和 $\mathrm{Tr}(\mathbf{H})$ と行列式 $\mathrm{Det}(\mathbf{H})$ は次のように計算できる．

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \tag{6.11}$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{6.12}$$

さらに，γ を第 1 固有値と第 2 固有値の比率とし，$\alpha = \gamma\beta$ とすると次式のようになる．

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma} \tag{6.13}$$

この値は固有値そのものではなく，固有値 α,β の比率で決まる．したがって，固有値を求めずにエッジ上の点であるか判別することが可能となる．この値を次式に示すようにしきい値処理することで，不要なキーポイント候補点を削除する．

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(\gamma_{th}+1)^2}{\gamma_{th}} \tag{6.14}$$

式 (6.14) を満足するような点をキーポイント候補とする．しきい値は γ_{th} により決定する．この処理は，ハリスのコーナー検出に良く似たもので，固有値の比率がしきい値より大きい点，つまりエッジ上に存在する点が削除される．文献 [1] では $\gamma_{th} = 10$ を採用しており，しきい値は 12.1 となる．

図 6.7(a) は検出された全キーポイント候補点を表している．図中の円の中心がキーポイント位置，円の半径がキーポイントの持つスケールである．図 6.7(b) では，主曲率によりドア等のエッジ上の点が削除されていることがわかる．

キーポイントのサブピクセル位置推定

3 変数 (x, y, σ) の 2 次関数をフィッティングすることで，キーポイント候補点のサブピクセル位置とスケールを算出する．ある点 $\mathbf{x} = (x, y, \sigma)^T$ での DoG 関数 $D(\mathbf{x})$ をテイラー展開すると次式のようになる．

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{6.15}$$

式 (6.15) について $\mathbf{x}$ に関する偏導関数を求め，0 とすると次式が得られる．

$$\frac{\partial D}{\partial \mathbf{x}} + \frac{\partial^2 D}{\partial \mathbf{x}^2}\hat{\mathbf{x}} = 0 \tag{6.16}$$

このとき $\hat{\mathbf{x}}$ はキーポイント候補点 (極値) のサブピクセル位置を表している．この式を変形し次式を得る．

$$\frac{\partial^2 D}{\partial \mathbf{x}^2}\hat{\mathbf{x}} = -\frac{\partial D}{\partial \mathbf{x}} \tag{6.17}$$

この式は次のように表される．

$$\begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial xy} & \frac{\partial^2 D}{\partial x\sigma} \\ \frac{\partial^2 D}{\partial xy} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y\sigma} \\ \frac{\partial^2 D}{\partial x\sigma} & \frac{\partial^2 D}{\partial y\sigma} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix} \begin{bmatrix} x \\ y \\ \sigma \end{bmatrix} = - \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \sigma} \end{bmatrix} \tag{6.18}$$

式 (6.18) をキーポイント候補点のサブピクセル位置 $\hat{\mathbf{x}}$ を得るために変形する．

$$\begin{bmatrix} x \\ y \\ \sigma \end{bmatrix} = - \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial xy} & \frac{\partial^2 D}{\partial x\sigma} \\ \frac{\partial^2 D}{\partial xy} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y\sigma} \\ \frac{\partial^2 D}{\partial x\sigma} & \frac{\partial^2 D}{\partial y\sigma} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \sigma} \end{bmatrix} \tag{6.19}$$

得られた式 (6.19) を解くことにより，キーポイント候補点のサブピクセル位置 $\hat{\mathbf{x}} = (x, y, \sigma)$ を得る．

コントラストによるキーポイントの絞り込み

サブピクセル位置での DoG 出力を算出し，コントラストによるキーポイントの絞り込みを行う．式 (6.19) は次のように表される．

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{6.20}$$

式 (6.20) を式 (6.15) に代入すると次式が得られる．

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}} \tag{6.21}$$

D は DoG 関数であり，$\hat{\mathbf{x}}$ はサブピクセル位置を表しているため，式 (6.21) はサブピクセル位置での DoG 出力値となる．この DoG の値からキーポイント削除の判別を行う．文献 [1] では，しきい値として 0.03 を用いている．サブピクセル位置での DoG 出力の絶対値がしきい値より小さい場合 (つまり，コントラストが低い場合) ノイズに影響されやすいため削除する．図 6.7(c) にコントラストにより絞り込まれたキーポイントを示す.

6.1.3 オリエンテーションの算出

検出したキーポイントに対して，第 2 段階の処理である特徴量の記述を行う．まず，検出された各キーポイントのオリエンテーションを求める．オリエンテーションはキーポイントにおける方向を表し，特徴量記述の際にオリエンテーションにより向きの正規化を行うことで，回転に不変となる．キーポイントのオリエンテーションを求めるには，まずキーポイントが検出された平滑化画像 $L(u,v)$ の勾配強度 $m(u,v)$ と勾配方向 $\theta(u,v)$ を以下の式により求める．

$$m(u,v) = \sqrt{f_u(u,v)^2 + f_v(u,v)^2} \tag{6.22}$$

$$\theta(u,v) = \tan^{-1}\frac{f_v(u,v)}{f_u(u,v)} \tag{6.23}$$

$$\begin{cases} f_u(u,v) = L(u+1,v) - L(u-1,v) \\ f_v(u,v) = L(u,v+1) - L(u,v-1) \end{cases} \tag{6.24}$$

局所領域における勾配強度 $m(x,y)$ と勾配方向$\theta(x,y)$ から図 6.8 に示すような重み付方向ヒストグラム h を以下の式により作成する．

$$h_{\theta'} = \sum_x \sum_y w(x,y) \cdot \delta\left[\theta', \theta(x,y)\right] \tag{6.25}$$

$$w(x,y) = G(x,y,\sigma) \cdot m(x,y) \tag{6.26}$$

ここで，h_θ は，全方向を 36 方向に量子化したヒストグラムである．$w(x,y)$ はある局所領域の画素 (x,y) での重みであり，キーポイントが持つスケールサイズのガウス窓 $G(x,y,\sigma)$ と勾配強度 $m(x,y)$ から求める．δ は Kronecker のデルタ関数で，勾配方向 $\theta(x,y)$ が量子化した方向 θ' に含まれるとき 1 を返す．また，このときのガウス窓にはキーポイントが持つスケールを用いる．ガウス窓による重み付けにより，キーポイントに近い特徴量がより強く反映される．この 36 方向のヒストグラムの最大値から 80% 以上となるピークをキーポイントのオリエンテーションとして割り当てる．

図 6.8 の例ではキーポイントに割り当てられるオリエンテーションは 1 方向のみであるが，図 6.9 のようにコーナーのようなキーポイントでは 2 方向となり，2 つのオリエンテーションを持つ．このように，1 つのキーポイントに対して複数のオリエンテーションが割り当てられる場合がある．

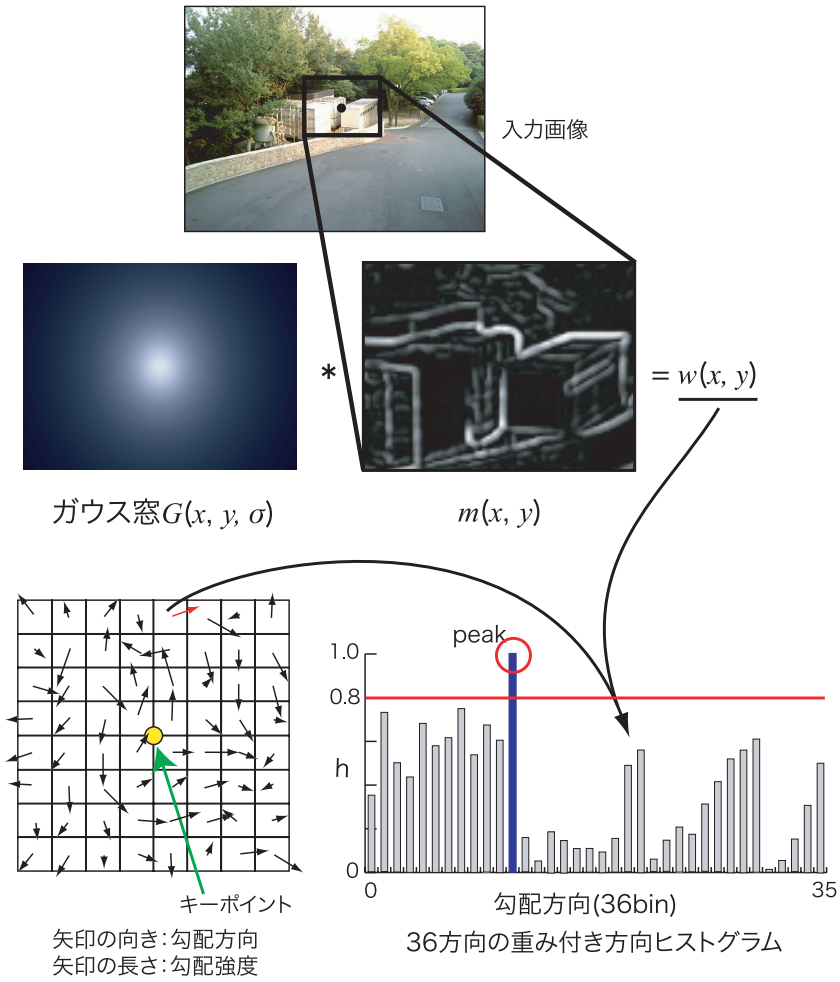


図 6.8: ヒストグラム作成の流れ

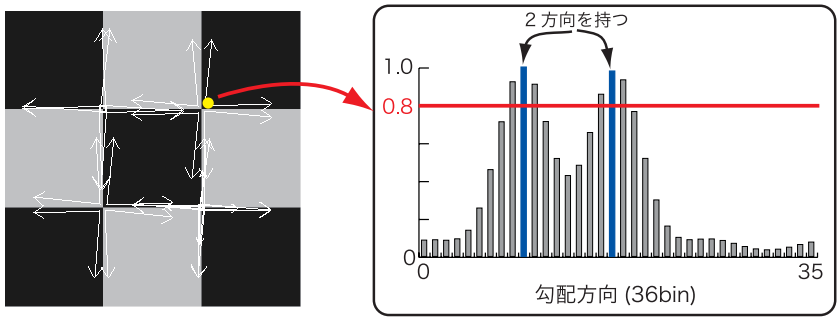


図 6.9: 2 方向のオリエンテーションを持つキーポイント

6.1.4 特徴量の記述

検出したオリエンテーションを基に，SIFT descriptor により 128 次元の特徴量を記述する．まず，図 6.10 に示すようにキーポイントのオリエンテーション方向に回転する．特徴量の記述には，キーポイント周辺領域の持つ勾配情報を用いる．使用する勾配情報は，キーポイントを中心とし，そのキーポイントが持つスケールを半径とした円領域内から求める (図 6.10 中のガウス窓内の領域)．周辺領域を一辺を 4 ブロックの計 16 ブロックに分割し，図 6.11 に示すようにブロックごとに 8 方向 (45 度ずつ) の勾配方向ヒストグラムを作成する．この勾配方向ヒストグラムは，キーポイントのオリエンテーションを算出したときに作成したヒストグラムと同様の手法で求める．

図 6.11 の例では $4 \times 4 = 16$ ブロックに各 8 方向のヒストグラムを作成するため，$4 \times 4 \times 8 = 128$ 次元の特徴ベクトルとしてキーポイントの特徴を記述する．このように，キーポイントが持つオリエンテーション方向に座標軸をあわせた領域で特徴を記述するため，回転に不変な特徴量となる．また，128 次元の各特徴ベクトルの長さはベクトルの総和で正規化する．これにより，キーポイントは照明変化に対して影響の少ない特徴量となる．

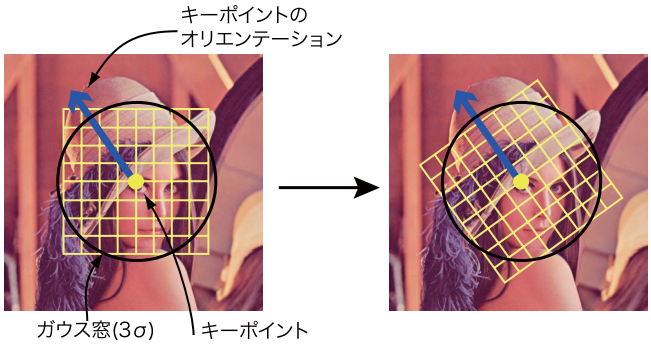


図 6.10: 特徴量を記述する領域

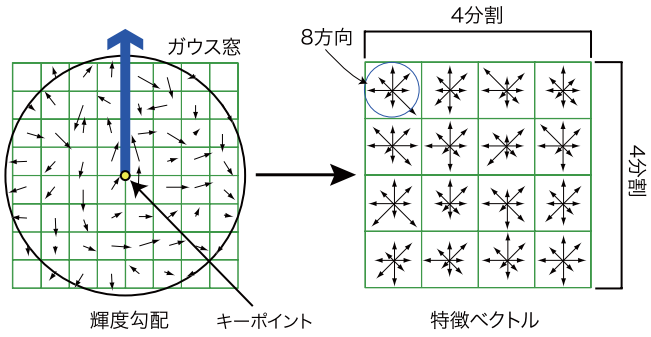


図 6.11: ブロックごとの特徴量記述

6.1.5　画像変化に対する SIFT 特徴量

画像の回転やスケール変化に対する SIFT 特徴量の影響について検討する．まず，参照用画像から SIFT によりキーポイントを検出する．次に，参照用画像に対して，回転・スケール変化・照明変化・アフィン変化・JPEG 圧縮の 5 種類の画像を作成する．参照用画像から検出されたあるキーポイント 1 点に注目し，画像変化に対する特徴量の影響について比較する．

図 6.12 に各変化に対する SIFT 特徴量 (128 次元) を示す．図 6.12 中の円の中心がキーポイントの位置であり，円はスケールであり特徴を記述する範囲である．また，円の中心から伸びる直線の方向は，そのキーポイントのオリエンテーションである．図 6.12(b), (c), (d), (e) に示す回転，スケール変化，照明変化，JPEG 圧縮 (ノイズの付加) が発生した画像と原画像とのユークリッド距離はどれも小さく，同じような特徴量を記述していることがわかる．しかし，図 6.12(f) に示すアフィン変化の場合は，画像に歪みの変化が含まれるため，スケールと方向を正規化して特徴を記述するだけでは不十分であり，ユークリッド距離が大きくなる．微小なアフィン変化に対してはある程度頑健ではあるが，不変ではないということに注意する必要がある．この問題に対し，Mikolajczyk らはアフィン変化に対応した領域検出器を用いることで，SIFT と同様の特徴記述を行う手法を提案している [5].

6.2　SIFT を用いたアプリケーション

scholar.google.com[6] で SIFT 論文 [1] の引用件数を調査したところ，911 件であった[1]．そのうち 291 件が SIFT を用いたアプリケーションに関する論文であり，その応用は以下の 4 つに大別できる．

- 対応点探索による画像のマッチング (142 件，49%)
- 特定画像を用いた物体認識 (71 件，24%)
- 画像分類 (73 件，25%)
- 特徴点追跡 (5 件，2%)

括弧内の数値は，SIFT を参照している論文数と，合計引用数 291 に対する各アプリケーションの割合である．本章では，上記の 4 つの SIFT を用いたアプリケーションについて述べる．

6.2.1　対応点探索による画像のマッチング 【Source Code】

SIFT により異なる画像間で抽出された各キーポイントの特徴量を比較することで，画像間の対応点探索が可能となる．以下に，対応点探索の流れを示す.

画像 $I1$ 中のあるキーポイント k_{I1} と画像 $I2$ 中のあるキーポイント k_{I2} の特徴量をそれぞれ $\mathbf{v}^{k_{I1}}$, $\mathbf{v}^{k_{I2}}$ とすると，特徴量間のユークリッド距離 d は次式により算出される.

$$d(\mathbf{v}^{k_{I1}}, \mathbf{v}^{k_{I2}}) = \sqrt{\sum_{i=1}^{128} (v_i^{k_{I1}} - v_i^{k_{I2}})^2} \tag{6.27}$$

[1] 2007 年 5 月 22 日時点

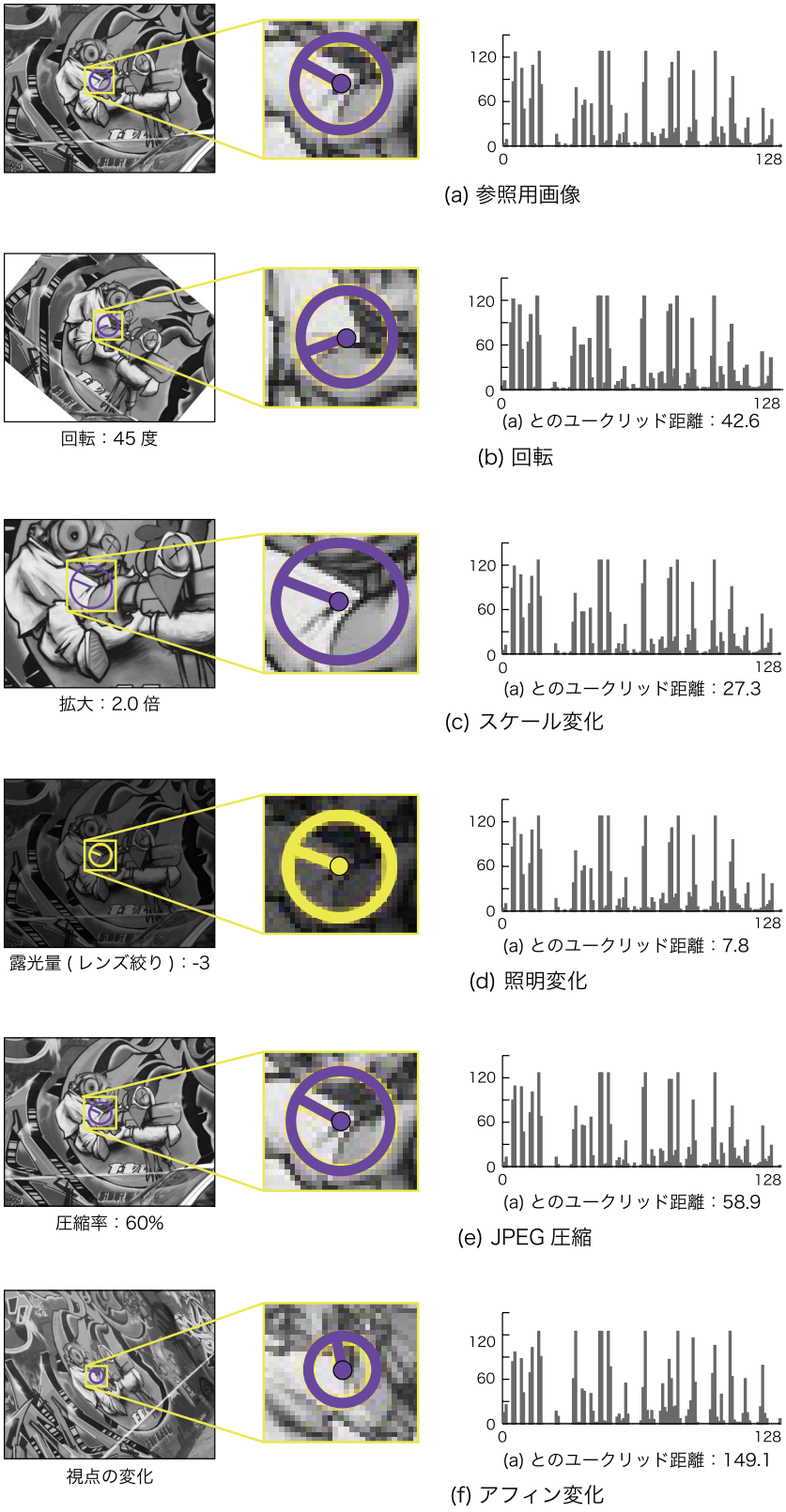


図 6.12: 画像変化に対する SIFT 特徴量

ここで，SIFT 特徴の次元数は 128 次元である．図 6.13 に示すように，あるキーポイント 1 点に対して，一方の画像中に含まれる全キーポイントとの特徴量間の距離 d を算出し，その中で最も d が最小となる距離を d_1，二番目に最小となる距離を d_2 とすると，d_1 と d_2 の関係から，式 6.28 を満たす組み合わせを対応点として検出することができる．このとき，r が 0 に近い程，信頼性が高い対応点となる．

$$d_1 < r \times d_2 \quad (0 < r < 1) \tag{6.28}$$

図 6.14 に同じ方向から距離を変えて撮影した画像のマッチング例を示す．このように，SIFT 特徴量を用いると，スケール変化に影響を受けず，対応点の検出が可能であることがわかる．

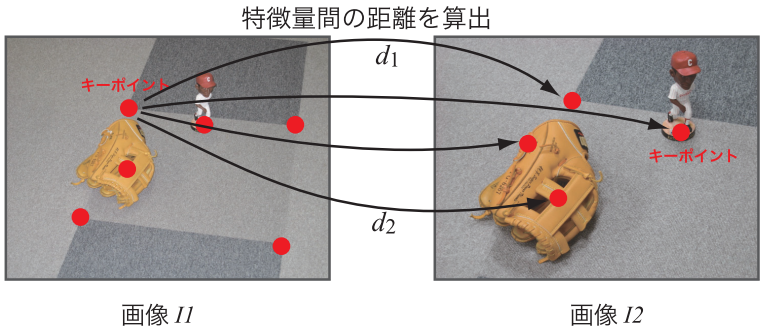


図 6.13: 対応点探索

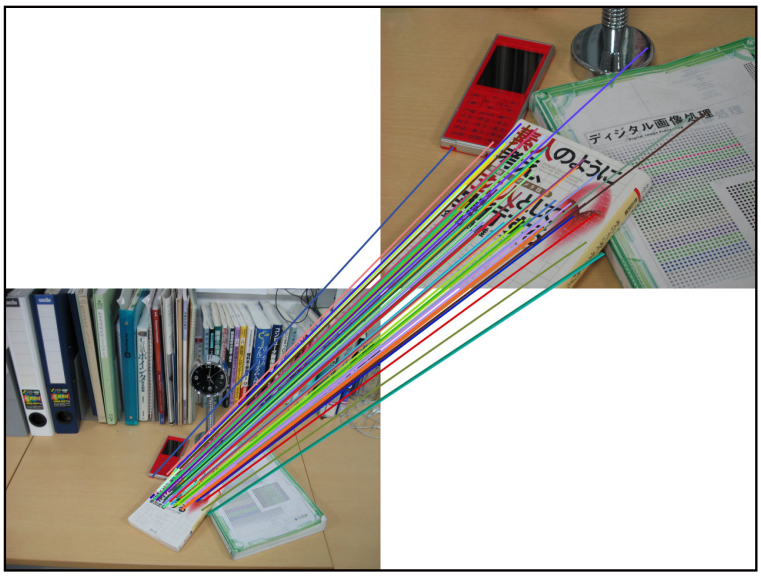


図 6.14: 対応点探索例

Autostitch[7], [8] は，SIFT を利用したモザイク画像を自動生成するフリーウェアである．Autostitch では，図 6.15 に示す 1600 × 1200 画素の 3 枚の画像からモザイク画像を約 5 秒で生成する．

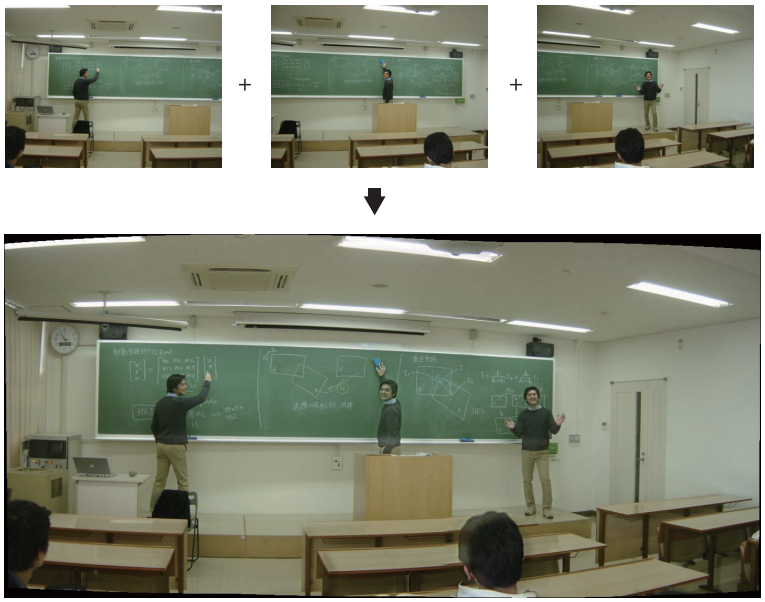


図 6.15: Autostitch で生成したモザイク画像

6.2.2 特定画像を用いた物体認識

あらかじめ参照画像から SIFT キーポイントを求めておき，入力画像と対応点探索を行うことで，物体認識が可能となる．Lowe[1] が提案した手法では，テンプレートから検出された各対応点の持つ 2 次元座標，スケール，オリエンテーションの 4 つのパラメータから一般化ハフ変換を用いて投票を行う．3 点以上の投票点からテンプレートと入力画像間のアフィンパラメータを算出する．同様に高木らは，車載カメラによって撮影された前方実環境画像から，SIFT 特徴量を用いてイラストパターンの道路標識と実画像のマッチングにより標識を認識する手法を提案している [9]．得られた対応点のスケール，オリエンテーションから，中心位置座標に投票して標識の認識を実現している．図 6.16 に標識検出・認識結果例を示す．

6.2.3 特徴点追跡

従来，特徴点追跡には KLT 法 [10] が用いられている．KLT 法は，局所領域における各点の動きは同一であると仮定し，弛緩法により目的関数を最小化する手法である．微小時間における領域は，平行移動のみしかしない，照明の変化による輝度値の変化がない，という状態を仮定して特徴点の移動先を求める．そのため，対象物体の運動に回転やスケール変化を含む場合や，照明の変化による輝度値の変化が激しい場合，特徴点の追跡に失敗することがある．一方，SIFT は画像の

(a) 入力画像　　(b) 認識結果画像

図 6.16: 道路交通標識の認識例

回転・スケール変化・照明変化に頑健な特徴量を記述することが可能である．したがって，SIFT により抽出された特徴点を追跡対象に用いることで，KLT 法では困難であった回転・スケール変化・照明変化が含まれる場合でも頑健に追跡を行うことが可能である．SIFT を用いた手法として，都築らは SIFT 特徴量の類似度を重みとした Mean-Shift 探索による特徴点追跡を提案し，非剛体の追跡へ応用している [11]．非剛体の追跡例として，図 6.17 に SIFT を用いた Mean-Shift 探索による人の追跡と KLT 法による人の追跡の結果を示す．図中の各点は特徴点の軌跡を表している．SIFT を用いた特徴点追跡法は KLT 法に比べ，より多くの点を長時間にわたり追跡できていることがわかる．

(a)SIFT+Mean-Shift　　(b)KLT 法

図 6.17: SIFT を用いた Mean-Shift 探索による特徴点追跡

6.2.4　Bag-of-Keypoints による画像分類

SIFT は局所領域のマッチングを頑健に行うことができるため，特定物体の同定には有効である．しかし，一般物体認識問題などのクラス分類には，SIFT 特徴量をそのまま使用することは困難である．そこで，SIFT を用いた一般物体認識・画像分類手法として，Bag-of-Keypoints[12] というアプローチが提案されている．Bag-of-Keypoints は，文書分類手法である Bag-of-words [13] を画像に適用した手法であり，Bag-of-words で文章を単語の集合と見なし，単語の語順を無視してそ

の頻度で文章の分類を行うのと同様に，画像を局所特徴量 (keypoint) の集合と見なし，その位置情報を無視して画像の認識を行う．具体的には，事前にすべての学習画像から SIFT 特徴量を抽出し，その局所特徴量をベクトル量子化する．このベクトル量子化された特徴量は visual word や visual alphabet などと呼ばれる．1 枚の入力画像から得られた visual word のヒストグラムをその画像の特徴量として，識別器を構築する．図 6.18 に Bag-of-Keypoints の流れを示す．

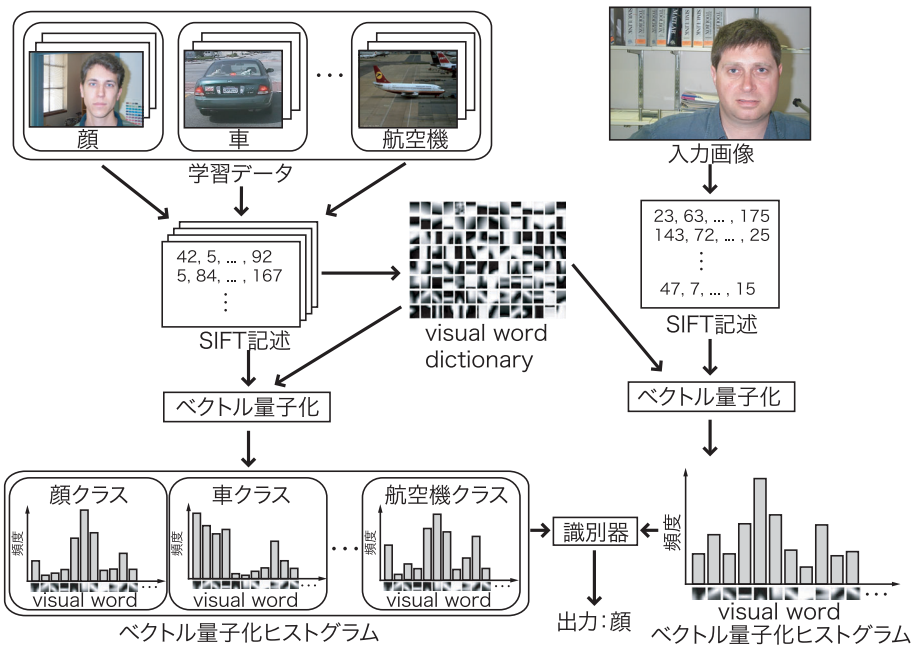


図 6.18: Bag-of-Keypoints の流れ

Manning ら [12] はキーポイントの検出に Affine Invariant keypoint [5] を利用し，領域をアフィン変換して SIFT descriptor により特徴量を記述することで，アフィン変換に頑健な特徴量に基づく物体認識を実現している．Fei-Fei ら [14] は風景画像など 13 クラスの画像分類法を提案している．この手法では，自然風景シーンではエッジやコーナーといった特徴点の抽出が困難であるため，画像を等間隔に分割し，ランダムに決定したスケールで SIFT 特徴量を記述している．キーポイントをグリッドに分割することで，DoG 極値を用いる場合よりも高精度な分類が可能であることを報告している．Agarwal ら [15] は，画像中の visual word のヒストグラムを局所領域で作成し上位階層の特徴量を計算し，これを繰り返すことで階層的な特徴量の記述を行う Hyperfeature を提案している．Nagahashi ら [16] は，visual word を構造ごとに分割した領域で作成することで，識別率を向上させている．Sivic ら [17] は，ビデオ中の各フレームに含まれる visual word を利用し，テキスト検索の手法を応用することで，視点の異なる同一シーンを高速に検索できる Video Google を提案している．このように，Bag-of-Keypoints アプローチにおける特徴量として，SIFT が用いられている．

6.3　SIFT の拡張

SIFT を拡張した手法が多く提案されている [18]-[21]．PCA-SIFT[18] は，SIFT によって検出された局所領域から得られる勾配情報を，PCA を用いて次元圧縮を行い特徴量を記述する手法である．BSIFT[19] は，背景情報 (Background) の影響を軽減して，対象とする物体の SIFT 特徴を記述する手法である．CSIFT[20] は，照明変化による物体の見えの変化の影響を除去し，照明変化に頑健な特徴を記述する手法である．さらに，SIFT を N 次元 (3 次元や 3 次元+時間) に拡張した N-SIFT[21] が提案されている．本章では，SIFT の拡張として PCA-SIFT[18] と BSIFT[19] について述べる．

6.3.1　PCA-SIFT

PCA-SIFT は，SIFT で検出した局所領域の勾配情報に対して主成分分析 (PCA) を適用し，SIFT 特徴の頑健性や識別性を向上させる手法である．PCA-SIFT では特徴点の検出とオリエンテーションの算出までの処理は SIFT と全く同じであり，特徴量記述のみ異なる．

特徴量記述

SIFT では，キーポイントのスケールに対応した領域を 4×4 のブロックに分割し，ブロックごとに 8 方向の方向ヒストグラムを作成することで 128 次元の特徴量を記述する．一方，PCA-SIFT では，キーポイントのスケールに対応した領域を 41×41 のパッチにリサンプリングする．リサンプリングしたパッチの水平・垂直方向の勾配を算出し，$39 \times 39 \times 2$=3,042 次元の特徴量を得る．求めた 3,042 次元の特徴量に PCA を適用し，次元圧縮を行う．PCA に用いる射影行列は，あらかじめ学習画像を用いて算出しておく．文献 [18] では，21,000 個のキーポイントから算出したパッチを用いて射影行列を求めている．圧縮する次元数は，実験により 36 次元が最も有効であることが報告されている．

ノイズが加えられた画像，回転・スケール変化がある画像，視点の変化がある画像，輝度を低減させた画像に対して，SIFT と PCA-SIFT の比較実験の結果，どの場合でも PCA-SIFT の性能が SIFT の性能を上回っていることが報告されている．また，PCA-SIFT の特徴量は低次元に圧縮されているため，SIFT よりも高速にマッチングを行うことが可能である．

6.3.2　BSIFT(Background and Scale Invariant Feature Transform)

SIFT は対象物体以外の背景画素を含んだ特徴量記述を行うため，背景の影響を受けやすいという問題がある．これは特徴点検出と特徴量記述に使用するガウス窓が，物体領域と背景領域を含むからである．Stein ら [19] は，物体と背景の境界情報を用いることで，物体領域のみの情報による特徴点の検出と特徴量を記述する手法である BSIFT を提案している．BSIFT では，境界情報が重要であり，物体の境界情報が既知であるか，もしくはデプス画像等から境界情報が得られた画像を対象とする．

特徴点検出

SIFT は DoG の極値を特徴点として検出する．しかし，この方法では背景情報を含んだまま特徴

点の検出を行う．そのため，同一物体でも異なる位置に特徴点が検出される場合がある．そこで，BSIFT では DoG のための平滑化として，ガウス関数ではなく以下の式を用いる．

$$I^{k+1}(u,v) \leftarrow I^{k}(u,v) + \tau \nabla^2 I^{k}(u,v) \tag{6.29}$$

$$\nabla^2 I(u,v) = \frac{\partial^2 I}{\partial u^2} + \frac{\partial^2 I}{\partial v^2} \tag{6.30}$$

k は繰り返し回数である．式 (6.30) は，ラプラシアンフィルタであり，式 (6.29) は 2 次微分を用いた平滑化である．式 (6.29) を用いることで，物体の境界に注目した平滑化を行うことができる．ここで τ を 0 に近づけたとき，$\sigma = \sqrt{2k\tau}$ でのガウシアンフィルタによる平滑化と同等の結果となる．文献 [19] では，$\tau = 0.2$ としている．

特徴量記述

SIFT による特徴量の記述は，キーポイントを中心としたガウス分布を重みとして輝度勾配ヒストグラムを作成する．しかし，この方法では背景画素の情報も含んだ特徴を記述してしまう．特に境界付近においては背景の影響が大きいため，マッチング失敗の原因となる．BSIFT では，キーポイントを中心としたガウス分布と境界情報を用いて距離変換した値を掛け合わせて重み分布とする (図 6.19)．したがって，背景領域の影響を軽減した SIFT 特徴量を得ることができる．

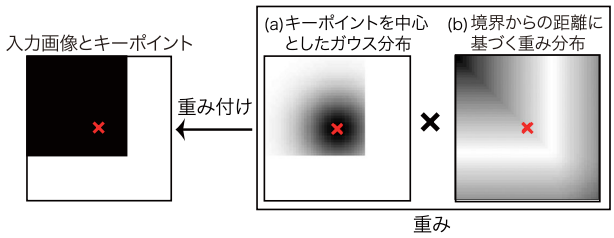


図 6.19: BSIFT で用いる重み分布

6.4 Histograms of Oriented Gradients(HOG)

一般物体認識のための gradient ベースの特徴量として，Histograms of Oriented Gradients(HOG)[14] が提案されている．HOG は，SIFT と同様に局所領域における輝度の勾配方向をヒストグラム化した特徴量である．SIFT と類似した特徴量の記述を行うが，SIFT は特徴点に対して特徴量を記述するのに対し，HOG ではある一定領域に対する特徴量の記述を行う．そのため，大まかな物体形状を表現することが可能であり，人検出 [14]-[26] や車検出 [24] 等の一般物体認識等に用いられている．

6.4.1 HOG 特徴量の算出

HOG 特徴量を算出するためには，画像から輝度勾配を算出し，算出された勾配強度と勾配方向から輝度の勾配方向ヒストグラムを作成し，正規化を行う．以下に HOG 算出アルゴリズムについて述べる．

輝度勾配算出

各ピクセルの輝度から SIFT と同様に勾配強度 m と勾配方向 θ を次式より算出する．

$$m(u,v) = \sqrt{f_u(u,v)^2 + f_v(u,v)^2} \tag{6.31}$$

$$\theta(u,v) = \tan^{-1}\frac{f_v(u,v)}{f_u(u,v)} \tag{6.32}$$

$$\begin{cases} f_u(u,v) = I(u+1,v) - I(u-1,v) \\ f_v(u,v) = I(u,v+1) - I(u,v-1) \end{cases} \tag{6.33}$$

セルによるヒストグラム化

図 6.20 に示すように，算出された勾配強度 m と勾配方向 θ を用いて，5×5 ピクセルをセルとした領域において輝度の勾配方向ヒストグラムを作成する．輝度の勾配方向ヒストグラムは，$0^\circ - 180^\circ$ を 20° ずつに分割するため，9 方向の勾配方向ヒストグラムとなる．

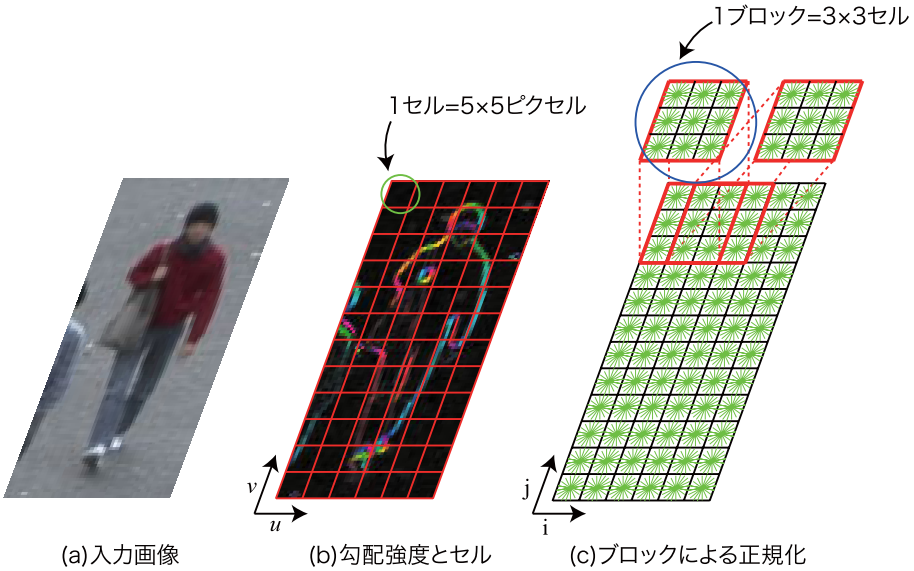


図 6.20: HOG で用いる領域の構造

ブロックによる正規化

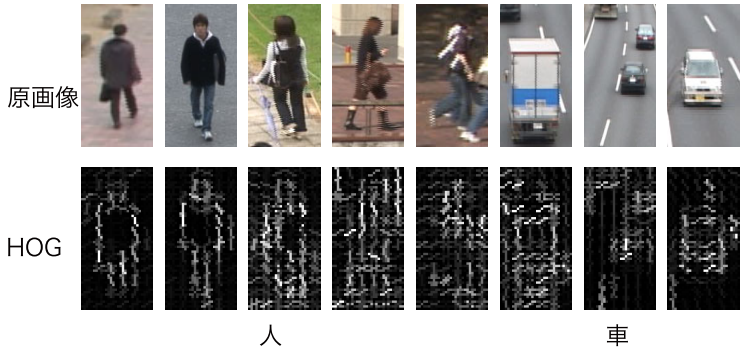
各セルで作成した輝度の勾配方向ヒストグラムを 3×3 セルを 1 ブロックとして正規化を行う．i 行 j 列のセル (i,j) の特徴量 (9 次元) を $F_{ij} = [f_1, f_2, \cdots, f_9]$ とすると，k 番目のブロックの特徴量 (81 次元) は $\mathrm{V}_k = [\mathrm{F}_{i\ j}, \mathrm{F}_{i+1\ j}, \mathrm{F}_{i+2\ j}, \mathrm{F}_{i\ j+1}, \mathrm{F}_{i+1\ j+1}, \mathrm{F}_{i+2\ j+1}, \mathrm{F}_{i\ j+2}, \mathrm{F}_{i+1\ j+2}, \mathrm{F}_{i+2\ j+2}]$ と表すことができる．正規化後の特徴量を v としたとき，次式より正規化する．

$$v = \frac{f}{\sqrt{||\mathrm{V_k}||_2^2 + \epsilon^2}} \qquad (\epsilon = 1) \tag{6.34}$$

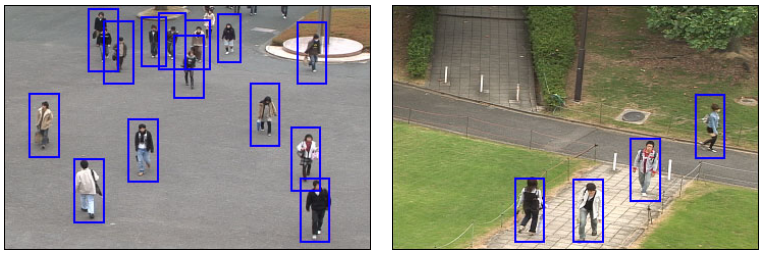
正規化は，図 6.20(c) のようにブロックを 1 セルずつ移動させることによって正規化を行う．そのため，特徴量 f は異なるブロックの領域によって何度も正規化される．入力画像を 30×60 ピクセルとした場合，横方向に 4 ブロック，縦方向に 10 ブロック，合計 40 ブロックに対して正規化を行う．各ブロックごとに正規化された HOG 特徴量は，40 ブロック $\times$ 81 次元 $= 3,240$ 次元となる．

6.4.2 HOG を用いた一般物体認識 【Source Code】

HOG 特徴量は，回転やスケール変化に不変ではないが，局所的な幾何学的変化と明度変化に不変であり，図 6.21(a) に示すように，大まかな形状特徴量を表現可能なため，一般物体認識に用いられている．Dalal らは，HOG 特徴量を算出し，SVM(Support Vector Machine) を用いて人検出する方法を提案した [14]．HOG は，PCA-SIFT や Shape Contexts を特徴量とした手法よりも高精度な人検出が可能であることが報告されている．図 6.21(b) に HOG を用いた人検出の結果を示す．さらに，アピアランス特徴である HOG に動きの情報としてオプティカルフローを用いた人検出法 [23] や，HOG と時空間特徴の共起を表現することで人検出精度を向上させる手法 [26] が提案されている．



(a)HOG 特徴量抽出例



(b)HOG を用いた人検出例

図 6.21: HOG 特徴量とその応用例

6.5　時空間特徴 Space-Time Patch

E.Shechtman らによって提案されている Space-Time Patch(ST-patch) 特徴 [27] は，画像を時間方向に重ねた 3 次元データである時空間画像の局所領域から得られる特徴である．図 6.22 に ST-patch の概念を示す．x，y は画像の座標軸，t は時間軸，3 つのラインは個々の画素の動き，$[u\ v\ w]^{\mathrm{T}}$ は ST-patch 中の動き方向ベクトル，∇P_i は個々の画素の勾配方向ベクトルを表している．

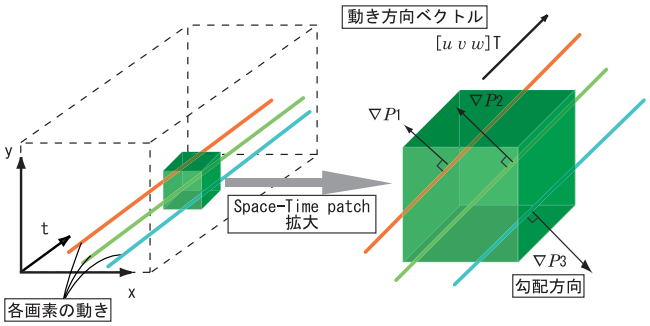


図 6.22: ST-patch の概要

6.5.1　ST-patch 特徴の算出

ST-patch 特徴を得るため，時空間画像においての x 軸，y 軸，t 軸の勾配を求める．画像中の動きが一定の場合，各軸に対するある画素 i の勾配 $\nabla P_i = (P_{x_i}, P_{y_i}, P_{t_i})$ は，画素の動き方向ベクトル $[u\ v\ w]^{\mathrm{T}}$ に対して垂直となる．よって，式 (6.35) の関係が成り立つ．

$$\nabla P_i \begin{bmatrix} u \\ v \\ w \end{bmatrix} = 0 \tag{6.35}$$

画素数が n の場合，式 (6.35) は式 (6.36) となる．

$$\begin{bmatrix} P_{x_1} & P_{y_1} & P_{t_1} \\ P_{x_2} & P_{y_2} & P_{t_2} \\ \vdots & \vdots & \vdots \\ P_{x_n} & P_{y_n} & P_{t_n} \end{bmatrix}_{n\times 3} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \tag{6.36}$$

ST-patch 中の n 画素の ∇P_i からなる $n \times 3$ の行列を $\mathbf{G}$ とし，行列 $\mathbf{G}^{\mathrm{T}}$ を掛けると式 (6.37) となる．

$$\mathbf{G}^{\mathrm{T}}\mathbf{G} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3\times 1} \tag{6.37}$$

この様に，行列 $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ は 3×3 の行列となる．ここで，行列 $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ を $\mathbf{M}$ とすると式 (6.38) のように表すことができる．

$$\mathbf{M} = \mathbf{G}^{\mathrm{T}}\mathbf{G} = \begin{bmatrix} \sum {P_x}^2 & \sum P_x P_y & \sum P_x P_t \\ \sum P_y P_x & \sum {P_y}^2 & \sum P_y P_t \\ \sum P_t P_x & \sum P_t P_y & \sum {P_t}^2 \end{bmatrix} \tag{6.38}$$

式 (6.38) より求められる行列 $\mathbf{M}$ が 1 つの ST-patch から得られる特徴となる．行列 $\mathrm{M}(3 \times 3)$ には，左上の 2×2 の行列にアピアランスの情報，3 行目と 3 列目にはモーションの情報を持つ．

6.5.2 ST-patch の性質

2 つの ST-Patch の一致

2 つの ST-patch P1，P2 間に共通のベクトル $\boldsymbol{u} = [u \ v \ w]^{\mathrm{T}}$ が存在する場合，P1 と P2 間の動きは一致し，各 ST-patch における行列 G_1 と G_2 は $\mathrm{G}_1\boldsymbol{u} = 0$ かつ $\mathrm{G}_2\boldsymbol{u} = 0$ となる．ここで，$\mathrm{G}_1\boldsymbol{u} = 0$ と $\mathrm{G}_2\boldsymbol{u} = 0$ をまとめると式 (6.39) となる．

$$\mathrm{G}_{12} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \mathrm{G}_1 \\ \mathrm{G}_2 \end{bmatrix}_{2n \times 3} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{2n \times 1} \tag{6.39}$$

ここで，行列 G_{12} は ST-patch の P1 と P2 の両方の勾配を含む．式 (6.37)，式 (6.38) と同様に，式 (6.39) に行列 G_{12} の転置行列 $\mathrm{G}_{12}^{\mathrm{T}}$ を掛け，$\mathrm{M}_{12} = \mathrm{G}_{12}^{\mathrm{T}}\mathrm{G}_{12}$ とすると式 (6.40) となる．

$$\mathrm{M}_{12} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3 \times 1} \tag{6.40}$$

また，ここで求められる M_{12} は式 (6.41) により求めることもできる．

$$\mathrm{M}_{12} = \mathrm{M}_1 + \mathrm{M}_2 = \mathrm{G}_1^{\mathrm{T}}\mathrm{G}_1 + \mathrm{G}_2^{\mathrm{T}}\mathrm{G}_2 \tag{6.41}$$

ランク制約による状態判定

画素の動き方向ベクトル $[u \ v \ w]^{\mathrm{T}}$ に対して垂直となる勾配 ∇P_i は，動きが一定の場合，全て 2 次元平面上に現れる．2 つの ST-patch の動きが一致している場合，それぞれの ∇P_i は共に同一 2 次元平面上に現れ，行列 rmM_{12} のランクは $rank(\mathrm{M}_{12}) \leq 2$ となる (図 6.23(a))．2 つの ST-patch の動きが不一致の場合，∇P_i は別々の 2 次元平面上に現れ，行列 M_{12} のランクは $rank(\mathrm{M}_{12}) = 3$ となる (図 6.23(b))． ここで，行列 M の 2×2 の左上の行列 $\mathrm{M}^{\diamondsuit}$ を式 (6.42) のように求める．

$$\mathrm{M}^{\diamondsuit} = \begin{bmatrix} \sum P_x^2 & \sum P_x P_y \\ \sum P_y P_x & \sum P_y^2 \end{bmatrix} \tag{6.42}$$

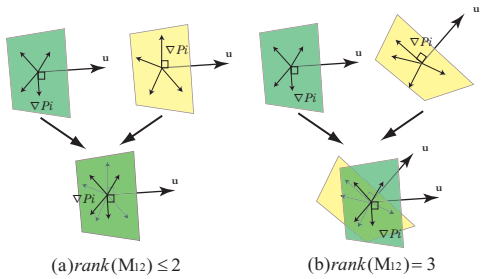


図 6.23: 勾配とランクの関係

ST-patch の空間の特性と式 (6.42) で求めれれる行列 $\mathrm{M}^{\diamondsuit}$ を用いて，ランク制約による ST-patch の動きの状態判定を行う．ST-patch の空間の特性から，単一の動きの ST-patch は，$rank(\mathrm{M}) = rank(\mathrm{M}^{\diamondsuit})$ のようなランクの状態となる．そして，複数の動きを含む ST-patch では，1 つの行と列だけが M からへ $\mathrm{M}^{\diamondsuit}$ 変換されているので，ランクの違いが 1 となる．よって，式 (6.43) のような関係が成り立つ．

$$\begin{aligned} \Delta r &= rank(\mathrm{M}) - rank(\mathrm{M}^{\diamondsuit}) \\ &= \begin{cases} 0 : \text{single motion} \\ 1 : \text{multiple motions} \end{cases} \end{aligned} \tag{6.43}$$

同様に，2 つの異なる ST-patch においても，動きが互いに一致しているかを式 (6.44) により求めることができる．

$$\begin{aligned} \Delta r &= rank(\mathrm{M}_{12}) - rank(\mathrm{M}_{12}^{\diamondsuit}) \\ &= \begin{cases} 0 : \text{consistent} \\ 1 : \text{inconsistent} \end{cases} \end{aligned} \tag{6.44}$$

6.5.3　ランク増加値 Δr

行列 M と行列 $\mathrm{M}^{\diamondsuit}$ の固有値を用いてランク増加値 Δr(Continues Rank Increase Measure) を定義し，2 つの異なる ST-patch の類似度を求める．

$\lambda_1 \geq \lambda_2 \geq \lambda_3$ を行列 M の固有値，$\lambda_1^{\diamondsuit} \geq \lambda_2^{\diamondsuit}$ を行列 $\mathrm{M}^{\diamondsuit}$ の固有値とする．それぞれの固有値の関係は $\lambda_1 \geq \lambda_1^{\diamondsuit} \geq \lambda_2 \geq \lambda_2^{\diamondsuit} \geq \lambda_3$ となる．これより式 (6.45) の関係が成り立つ．

$$\lambda_1 \geq \frac{\lambda_1 \cdot \lambda_2 \cdot \lambda_3}{\lambda_1^{\diamondsuit} \cdot \lambda_2^{\diamondsuit}} \geq \lambda_3 \geq 0 \tag{6.45}$$

式 (6.45) を λ_1 で割ると式 (6.46) のようになる．

$$1 \geq \frac{\lambda_2 \cdot \lambda_3}{\lambda_1^{\diamondsuit} \cdot \lambda_2^{\diamondsuit}} \geq \frac{\lambda_3}{\lambda_1} \geq 0 \tag{6.46}$$

式 (6.46) から，ランク増加量 Δr を式 (6.47) のように定義する．

$$\Delta r \quad = \quad \frac{\lambda_2 \cdot \lambda_3}{\lambda_1^{\diamondsuit} \cdot \lambda_2^{\diamondsuit}} \qquad (0 \leq \Delta r \leq 1) \tag{6.47}$$

ここで求められたランク増加量 Δr の値は，ST-patch 内の動きベクトルが一定のとき $\Delta r \approx 0$ となり，ST-patch 内の動きベクトルの変化が激しいとき $\Delta r \approx 1$ となる．

6.5.4 2 つの ST-Patch の相互関係

式 (6.47) から求められたランク増加量 Δr より，2 つの ST-patch P1 と P2 の動きの類似度を求めることができる．P1 のランク増加量を Δr_1 ，P2 のランク増加量を Δr_2 ，P1 と P2 をまとめたランク増加量を Δr_{12} とすると，式 (6.48) により動きの類似度が求められる．

$$m_{12} \quad = \quad \frac{\min(\Delta r_1, \Delta r_2)}{\Delta r_{12}} \tag{6.48}$$

ここで求められた類似度 m_{12} は，$m_{12} \approx 1$ の場合は類似度が高く，$m_{12} \approx 0$ の場合は類似度が低くいことを意味する．

6.5.5 ST-patch 特徴を用いた動作識別 【Source Code】

図 6.24 に ST-patch 特徴を用いて，時系列テンプレートと同じ動きをしている物体をビデオシーケンス内から識別した例を示す．ここでは，ある小領域の時系列テンプレートとビデオシーケンスの動きが一致しているかどうかを求めるために，時系列テンプレートとビデオシーケンスから得られた 2 つの ST-patch の特徴の類似度を算出し，求められた 2 つの ST-patch の特徴の類似度を複数用いて総合的に動きを評価することにより物体の動きの識別を行っている．

6.5.6 ST-patch 特徴を用いた移動方向識別

村井らは，学習サンプルより作成した ST-patch 特徴のコードブックを用いることで物体の移動方向識別を行う手法を提案している [28]．以下に，手法の流れを述べる．

Step1 入力画像のラスタスキャンにより抽出されたパッチの ST-patch 特徴を抽出

Step2 入力 ST-patch 特徴を $\mathbf{e}$，作成したコードブックの各クラスタの特徴ベクトルを $\mathbf{c}$ とし，2 つ特徴ベクトル間のユークリッド距離を計算し，ユークリッド距離が最小のクラスタ $\mathbf{C}$ を式 (6.49) により算出

$$\mathbf{C} = \underset{\mathbf{c}}{\operatorname{argmin}} \parallel \mathbf{e} - \mathbf{c} \parallel^2 \tag{6.49}$$

Step3 コードブックのクラスタのラベル含有率に基づいて移動方向の識別

Step4 画像全体の処理が終了するまで **Step1**〜**Step3** を繰り返す

以上の処理により，図 6.25 に示す結果を得ることができる．

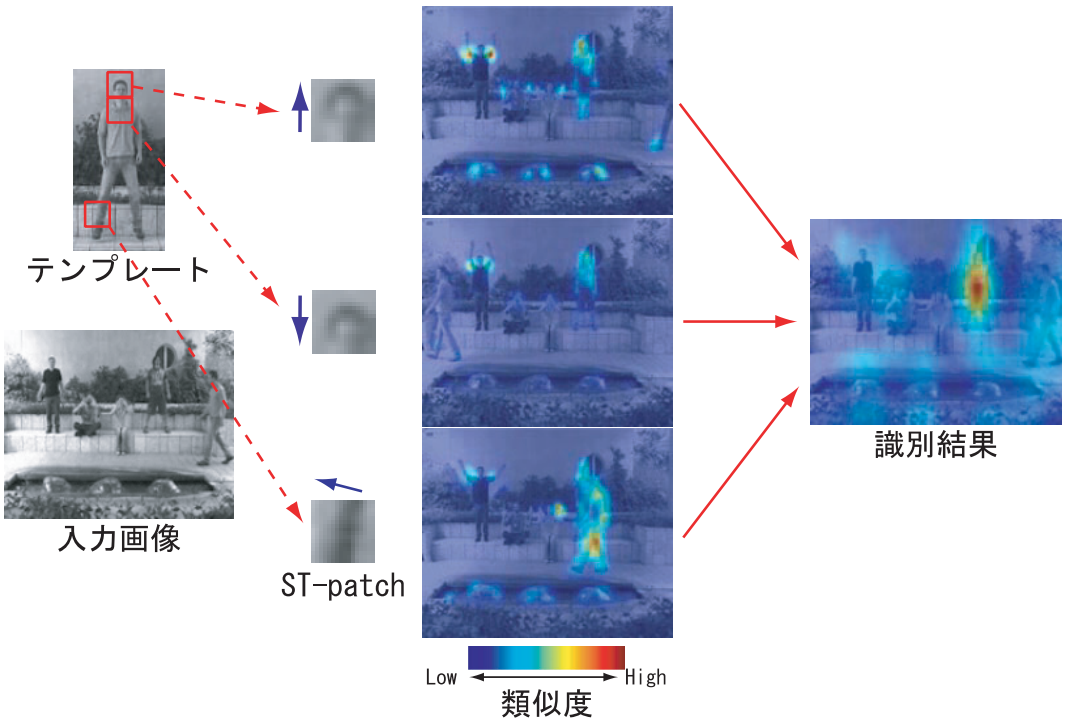


図 6.24: ST-patch を用いた動作識別 (文献 [27])

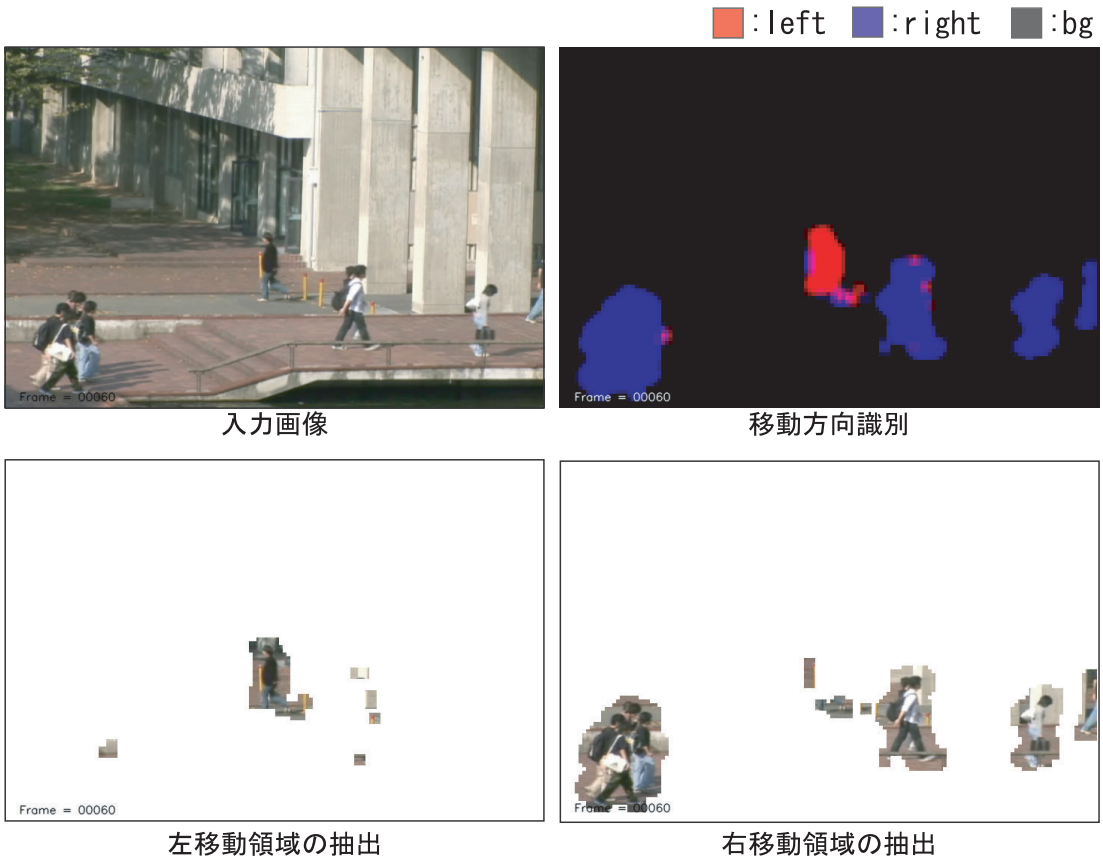


図 6.25: ST-patch を用いた物体の移動方向識別例文献 [27])

参考文献

[1] D. Lowe, “Distinctive image features from scale-invariant keypoints”, Proc. of International Journal of Computer Vision (IJCV), 60(2), pp. 91-110, 2004.

[2] D. G. Lowe, “Object recognition from local scale-invariant features”, Proc. of IEEE International Conference on Computer Vision (ICCV), pp. 1150-1157, 1999.

[3] J. J. Koenderink, “The structure of images”, Proc. of Biological Cybernetics, vol. 50, pp. 363-370, 1984.

[4] T. Lindeberg, “Scale-space theory: A basic tool for analysing structures at different scales”, Proc. of Journal of Applied Statistics, 21(2), pp. 224-270.

[5] K. Mikolajczyk, and C. Schmid, “An affine invariant interest point detector”, Proc. of European Conference on Computer Vision (ECCV), pp. 128-142, 2002.

[6] http://scholar.google.com/

[7] M. Brown and D. G. Lowe, “Recognising panoramas”, Proc. of IEEE International Conference on Computer Vision (ICCV), pp. 1218-1225, Nice, France, October, 2003.

[8] http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

[9] 高木雅成, 藤吉弘亘, “SIFT 特徴量を用いた交通道路標識認識”, 第 13 回画像センシングシンポジウム SSII07, LD2-06, 2007.

[10] C. Tomasi and T. Kanade, “Detection and tracking of point features”, Technical report, CMU-CS-91-132, 1991.

[11] 都築勇司, 藤吉弘亘, 金出武雄, “SIFT 特徴量に基づく Mean-Shift 探索による特徴点追跡”, 情報処理学会 研究報告 CVIM157, pp. 101-108, 2007.

[12] G. Csurka, C.R. Dance, L. Fan, and C. Bray, “Visual categorization with bags of keypoints”, Proc. of European Conference on Computer Vision (ECCV), pp. 1-22, 2004.

[13] C. D. Manning, and H. SchFutze, “Foundation of statistical natural language processing”, The MIT Press, 1999.

[14] L. Fei-Fei, and P. Perona, “A bayesian hierarchical model for learning natural scene categories”, Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol 2, pp. 524 - 531, 2005.

[15] A. Agarwal, and B. Triggs, “Hyperfeatures – multilevel local coding for visual recognition”, Proc. of European Conference on Computer Vision (ECCV), vol. 1, pp 30-43, 2006.

[16] T. Nagahashi, H. Fujiyoshi, T. Kanade, “Object type classification using structure-based feature representation”, MVA2007：IAPR Conference on Machine Vision Applications, pp. 142-145, May, 2007.

[17] J. Sivic, and A. Zisserman, “Video google: A text retrieval approach to object matching in videos”, Proc. of IEEE International Conference on Computer Vision (ICCV), vol. 2, pp. 1470-1477, 2003.

[18] Y. Ke, R. Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors”, Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 511-517, 2004.

[19] A. Stein, M. Herbert, “Incorporating background invariance into feature-based object recognition”, Proc. of IEEE Workshop on Applications of Computer Vision (WACV), pp. 37-44, January, 2005.

[20] Alaa E. Abdel-Hakim and Aly A. Farag, “CSIFT: A SIFT descriptor with color invariant characteristics”, Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1978-1983, 2006.

[21] W. Cheung and G. Hamarneh, “N-dimensional scale invariant feature transform for matching medical images,” Proc. of IEEE International Symposium on Biomedical Imaging (ISBI), pp. 720-723, 2007.

[22] N. Dalal, B. Triggs, “Histograms of oriented gradients for human detection”, Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 886-893, 2005.

[23] N. Dalal, B. Triggs and C. Schmid, “Human detection using oriented histograms of flow and appearance”, Proc. of IEEE European Conference on Computer Vision (ECCV), vol. 2, pp. 428-441, May, 2006.

[24] F. Han, Y. Shan, R. Cekander, H. S. Sawhney and R. Kumar, “A two-stage approach to people and vehicle detection with HOG-based SVM”, Proc. of Workshop on Performance Metrics for Intelligent Systems, pp. 133-140, 2006.

[25] F. Suard, A. Broggi, “Pedestrian detection using infrared images and histograms of oriented fradients”, Proc. of IEEE Intelligent Vehicles Symposium (IV), pp. 206-212, Jun, 2006.

[26] 山内悠嗣, 藤吉弘亘, Bon-Woo Hwang, 金出武雄, “アピアランスと時空間特徴の共起に基づく人検出”, 第 10 回画像の認識・理解シンポジウム (MIRU2007), pp. 1492-1497, Jul, 2007.

[27] E.Shechtman and M.Irani, “Space-Time Behavior Based Correlation”, Computer Vision and Pattern Recognition, vol.1, pp.405- 412, 2005.

[28] 村井泰裕, 藤吉弘亘, 金出武雄, “Space-Time Patch を用いた物体の移動方向識別とセグメンテーション”, 情報処理学会論文誌, vol.1, No.2, pp21-31, 2008.

第7章　物体識別:Object Classification

検出した移動体領域が人もしくは車であるかを識別するには，検出領域内から特徴量を抽出し事前に学習した必要がある．本章では，VSAM で用いた形状特徴の抽出法と識別器の構成方法について述べる．次に，ニューラルネットワーク，SVM，Adaboost による物体識別法について述べ，最後に特徴量と統計的学習法の動向について述べる．

7.1　形状特徴量の抽出

背景差分により検出された画像領域が人か自動車であるかを判定するためには，各クラスの特徴を最大限に表す特徴量を選択する必要がある．しかし，屋外に設置したカメラ映像内の自動車や人は，図 7.1 に示すように同一クラス内においてもその進行方向により見えは異なる．そのため，このような見えの違いに依存しない特徴量を検討する必要がある．

7.1.1　形状の複雑度

輪郭線の凹凸に着目した形状の複雑度は次式より求められる．

$$\text{複雑度 } (dispersedness) = \frac{\text{周囲長}^2}{\text{面積}} \tag{7.1}$$

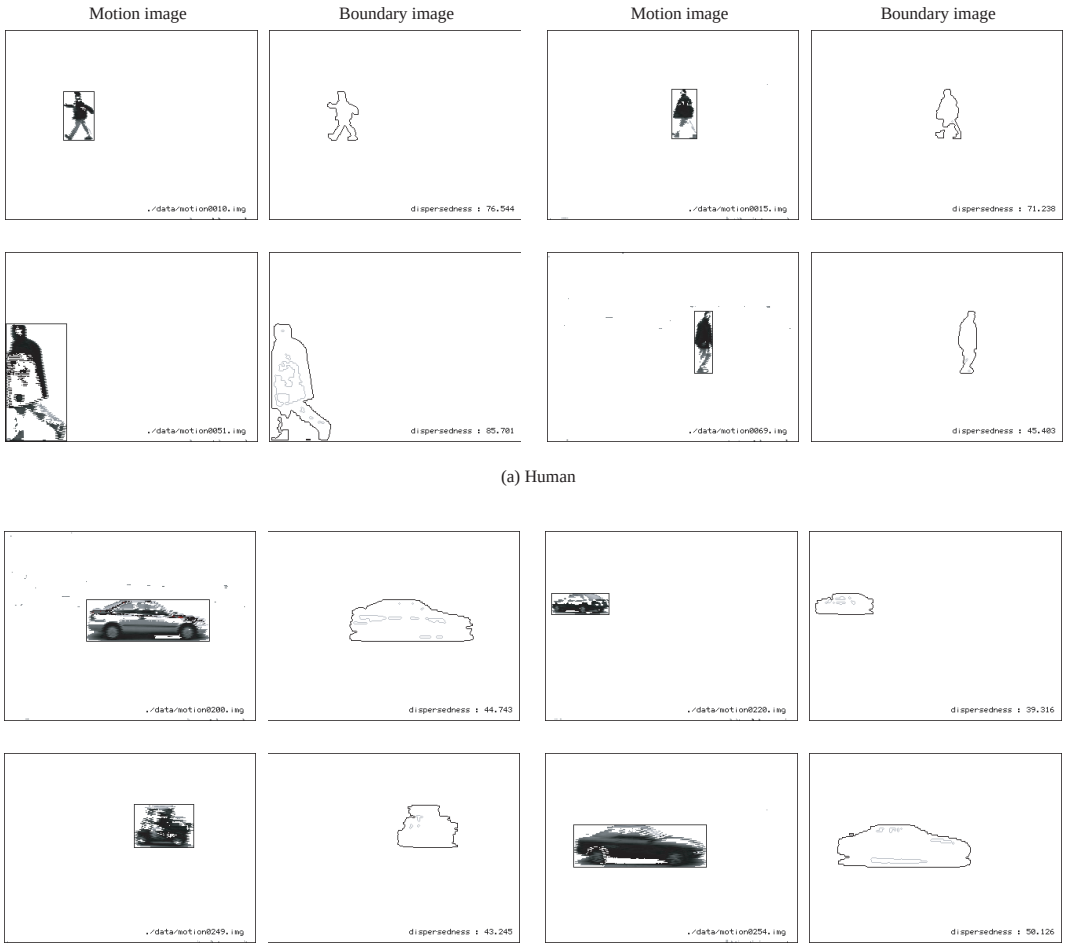
周囲長は，検出したピクセル群から輪郭線を抽出し，その個数もしくは市街地距離の総和として求める．面積は検出ピクセル数とする．複雑度の最小値は円形状のときであり，その値は 4π となる．

図 7.2 に，車が 44 パターン，人が 169 パターンに対する複雑度のヒストグラムを示す．車の輪郭形状は，人の輪郭形状より複雑でないため低い値に分布し，その分散も小さいことがわかる．一方，人の場合は歩行の際に，足や手の振りによりその形状の複雑度が変化するため複雑度が低い値から高い値まで分布していることがわかる．

7.1.2　しきい値処理による識別

複雑度を用いた簡単な識別をオンラインで行うには，検出された未知入力サンプルに対して複雑度を求め，予め設定したしきい値より複雑度が大きい場合を人クラス，小さい場合を自動車クラスと判定する．

$$\begin{aligned} dispersedness &< th \quad : \quad \text{人クラスに属する} \\ dispersedness &\geq th \quad : \quad \text{自動車クラスに属する} \end{aligned}$$



(a) Human

(b) Vehicle

図 7.1: 検出結果と輪郭線特徴

図 7.3 にしきい値を変化させたときの各クラスの識別率を示す．グラフから，しきい値が 25 のとき，83.7%の識別率を得た．複雑度は人と車の形状の違いを表す一特徴として有効であることが分かる．

7.2　識別器の構築

7.1 で述べた 1 変数による簡単なしきい値処理による物体識別では，しきい値を手動で決定する必要がある．さらに，1 つの特徴量のみを使用するため，識別能力が低いという問題がある．より高性能な識別器を実現するには，クラス間の違いを表すより効果的な他の特徴量の追加や，多くの学習サンプルデータを用いた識別器の構築が考えられる．ここでは，線形判別関数とマハラノビス距離による識別手法について述べる．

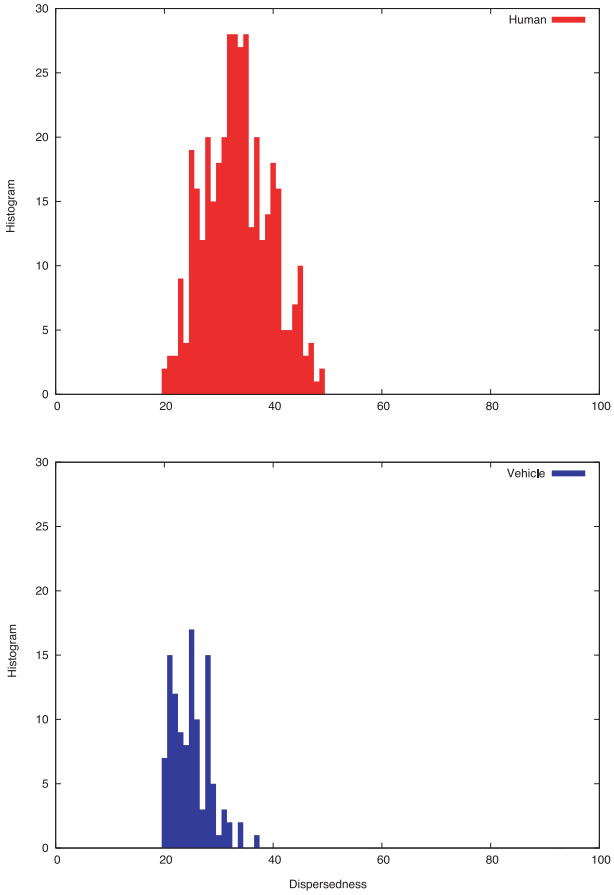


図 7.2: 複雑度に対するヒストグラム

7.2.1 線形判別関数

図 7.4 は，縦軸を複雑度，横軸を面積（検出ピクセル数）とした場合の各クラスの分布である．図 7.2 の 1 変数 (複雑度) によるヒストグラムでは，境界線によるクラスの分離が難しいことが推測できるが，図 7.4 より新しい特徴量 (面積) を追加することで，その判別が容易になることが期待できる．

そこで，学習用サンプルデータから各クラスをより良く分離する境界線を求める手法として線形判別関数 (Linear Discriminant Function) を用いる．線形判別関数における境界線は，図 7.5(a) に示すように直線であり，1 次式による判別である．一方，図 7.5(b) に示すマハラノビス距離 (mahalanobis' generalized distance) は，2 次式で表される非線形な識別境界線となる．

特徴量となる 2 変数 (x_1, x_2) による線形判別関数は次式となる．

$$z = a_1 x_1 + a_2 x_2 + a_0 \tag{7.2}$$

a_1, a_2, a_0 は，予め用意した学習サンプルより，次式に示すように求める．この過程をオフライ

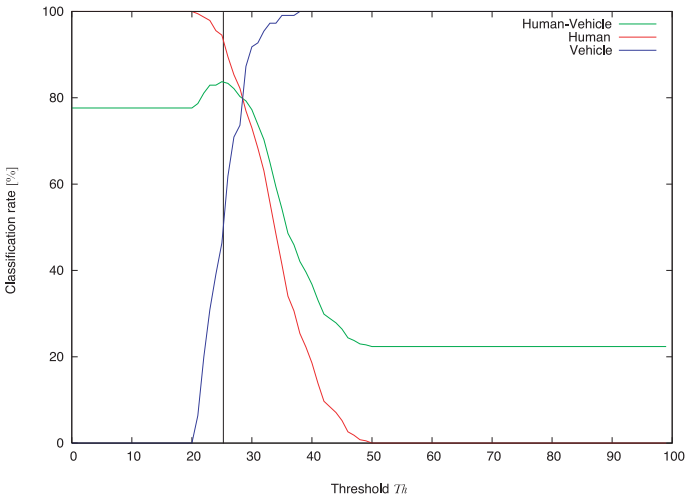


図 7.3: 複雑度による線形判別結果

ン学習と呼ぶ.

$$\begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} \sum x_{1i}^{(1)2} - \frac{(\sum x_{1i}^{(1)})^2}{N_1} & \sum x_{1i}^{(1)} x_{2i}^{(1)} - \frac{(\sum x_{1i}^{(1)} \sum x_{2i}^{(1)})^2}{N_1} \\ \sum x_{1i}^{(1)} x_{2i}^{(1)} - \frac{(\sum x_{1i}^{(1)} \sum x_{2i}^{(1)})^2}{N_1} & \sum x_{2i}^{(1)2} - \frac{(\sum x_{2i}^{(1)})^2}{N_1} \end{bmatrix} \tag{7.3}$$

$$\begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} = \begin{bmatrix} \sum x_{1i}^{(2)2} - \frac{(\sum x_{1i}^{1})^{(2)}}{N_2} & \sum x_{1i}^{(2)} x_{2i}^{(2)} - \frac{(\sum x_{1i}^{(2)} \sum x_{2i}^{(2)})^2}{N_2} \\ \sum x_{1i}^{(2)} x_{2i}^{(2)} - \frac{(\sum x_{1i}^{(2)} \sum x_{2i}^{(2)})^2}{N_2} & \sum x_{2i}^{(2)2} - \frac{(\sum x_{2i}^{1})^{(2)}}{N_2} \end{bmatrix} \tag{7.4}$$

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} + w_{11}^{(2)} & w_{12}^{(1)} + w_{12}^{(2)} \\ w_{21}^{(1)} + w_{21}^{(2)} & w_{22}^{(1)} + w_{22}^{(2)} \end{bmatrix} \tag{7.5}$$

$$\begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} = \begin{bmatrix} \frac{w_{11}}{N_1+N_2-2} & \frac{w_{12}}{N_1+N_2-2} \\ \frac{w_{21}}{N_1+N_2-2} & \frac{w_{22}}{N_1+N_2-2} \end{bmatrix} \tag{7.6}$$

$$a_1 = \frac{1}{s_{11}s_{22} - s_{12}^2} \left\{ s_{22} \left(\frac{\sum x_{1i}^{(1)}}{N_1} - \frac{\sum x_{1i}^{(2)}}{N_2} \right) s_{12} \left(\frac{\sum x_{2i}^{(1)}}{N_1} - \frac{\sum x_{2i}^{(2)}}{N_2} \right) \right\} \tag{7.7}$$

$$a_2 = \frac{1}{s_{11}s_{22} - s_{12}^2} \left\{ s_{12} \left(\frac{\sum x_{1i}^{(1)}}{N_1} - \frac{\sum x_{1i}^{(2)}}{N_2} \right) s_{11} \left(\frac{\sum x_{2i}^{(1)}}{N_1} - \frac{\sum x_{2i}^{(2)}}{N_2} \right) \right\} \tag{7.8}$$

$$a_0 = -\frac{1}{2} \left(\frac{a_1 \sum x_{1i}^{(1)} + a_2 \sum x_{2i}^{(1)}}{N_1} + \frac{a_1 \sum x_{1i}^{(2)} + a_2 \sum x_{2i}^{(2)}}{N_2} \right) \tag{7.9}$$

$x_{1i}^{(1)}, x_{2i}^{(1)}, x_{1i}^{(2)}, x_{2i}^{(2)}$ は両クラスにおける各特徴量，N_1, N_2 は両クラスのサンプル数である．この結果，各変数 (x_1, x_2) に対する重み a_1, a_2 が決定され，オンラインで識別する際には，検出した移動体領域内の特徴量を計算し，次式に代入して z を求める．

$$z = -0.002382592 \times area + 0.027149614 \times dispersedness - 0.44778073 \tag{7.10}$$

上式の係数は実際に人クラス 80 パターン，車クラス 80 パターン計 160 パターンの学習用サンプ

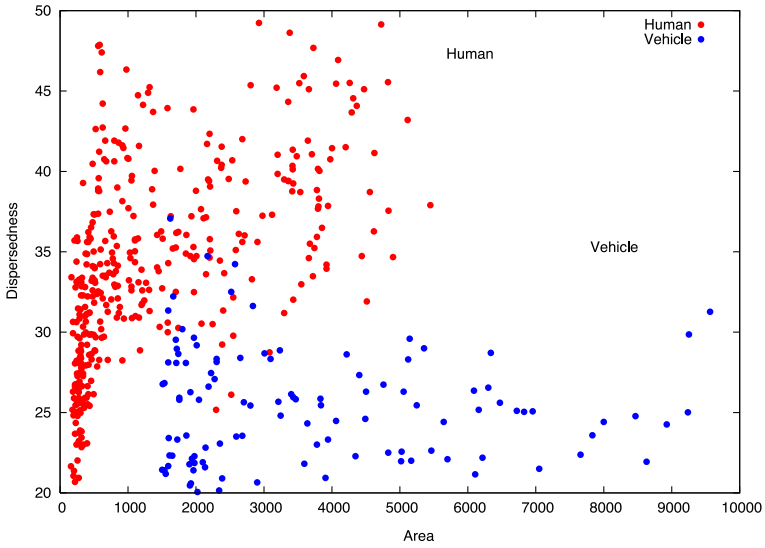


図 7.4: 各クラスの分布

ルから計算したものである．クラスの判定は z の値が 0 より大きいか小さいかで，以下に示す条件で判別を行う．

$$\begin{cases} z > 0 & : \quad \text{クラス } 2(human) \text{ に属する} \\ z < 0 & : \quad \text{クラス } 1(vehicle) \text{ に属する} \end{cases}$$

7.2.2 マハラノビス距離

より高度な識別を実現するには，非線形 (2 次式) な境界線で判別するマハラノビス距離が有効である．マハラのビス距離の算出法を以下に示す．

n 個の確率変数をもつパターンを $\boldsymbol{X}$ とし，$\boldsymbol{X}$ の分散共分散行列を作成する．

$$\sum_i = \begin{bmatrix} \frac{1}{N}\sum_{p=1}^{N}(x_{1p}-\mu_1)^2 & \cdots & \frac{1}{N}\sum_{p=1}^{N}(x_{1p}-\mu_1)(x_{np}-\mu_n) \\ \vdots & \ddots & \vdots \\ \frac{1}{N}\sum_{p=1}^{N}(x_{np}-\mu_n)(x_{1p}-\mu_1) & \cdots & \frac{1}{N}\sum_{p=1}^{N}(x_{np}-\mu_n)^2 \end{bmatrix} \tag{7.11}$$

となる.N はサンプル数で $\mu_1, \mu_2, \cdots, mu_n$ は平均である．ここで $\sum_i$ としたときマハラノビス距離を用いた確率密度関数 $f_i(X)$ は,

$$f_i(X) \quad = \quad \frac{1}{(2\pi)^{\frac{1}{2}}|\sum_i|^{\frac{1}{2}}} exp(-\frac{1}{2}(X-\bar{\mu})\sum\nolimits_i^{-1}(X-\bar{\mu})) \tag{7.12}$$

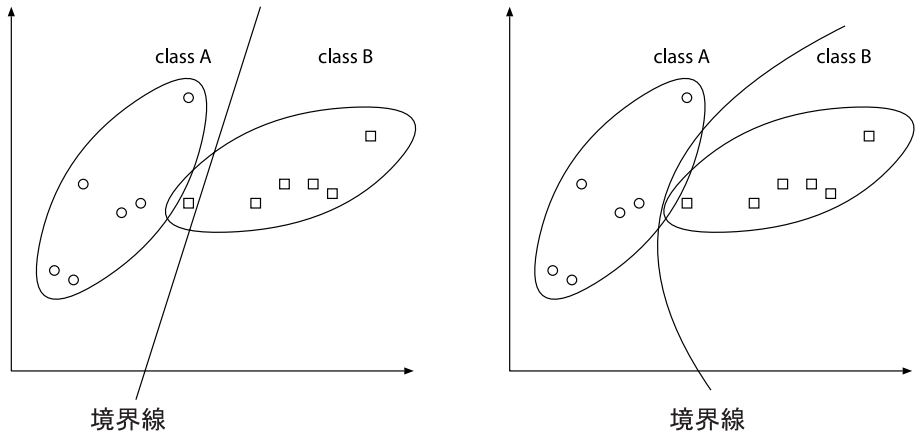


図 7.5: 線形判別関数とマハラノビス距離による境界線

となる. また確率密度関数 $f_i(X)$ は,

$$\int_{-\infty}^{\infty} f_i(X) \quad = \quad 1 \tag{7.13}$$

となる. よって確率密度関数とは, 確率変数 X が入力されたとき,X がクラスについてのメンバらしさを求めるものである. よって全クラスについて $f_i(X)$ を計算したとき, その最大値 f_m であればパターン X は m に所属すると判断する. 識別関数 $f_i(X)$ は, 計算の便宜上 $\ln f_i(X)$ を用いる. ln を用いても相互の大小関係は変わらないためである.

$$\ln f_i(X) \quad = \quad -\frac{n}{2}(2\pi) - \frac{1}{2}\ln|\sum{}_{\mathrm{i}}| - \frac{1}{2}(\mathrm{X}-\bar{\mu})\sum{}_{\mathrm{i}}^{-1}(\mathrm{X}-\bar{\mu}) \tag{7.14}$$

ここで N はサンプル数，$\mu_1, \mu_2, \cdots, \mu_n$ は $x_1, x_2, \cdots, x_n$ の平均値である．ここで, 分散共分散が $|\sum_i| = |\sum|(i = 1, 2, 3, \cdots, n)$ であったとき $f_i(X)$ の比較は,

$$g_i(X) \quad = \quad -(X-\bar{\mu})^{\top}\sum{}_{i}^{-1}(X-\bar{\mu}) \tag{7.15}$$

$$\tag{7.16}$$

となる. これがマハラノビス距離を用いた識別関数である.

7.2.3　線形判別関数とマハラノビス距離による識別結果 【Source Code】

図 7.6 に，複雑度と面積分布における線形判別関数とマハラノビス距離による識別境界を示す. このときの識別結果を表 7.1 に示す．マハラノビス距離を用いた識別は，非線形な識別境界による識別を行うため，より精度の高い識別が実現できていることがわかる.

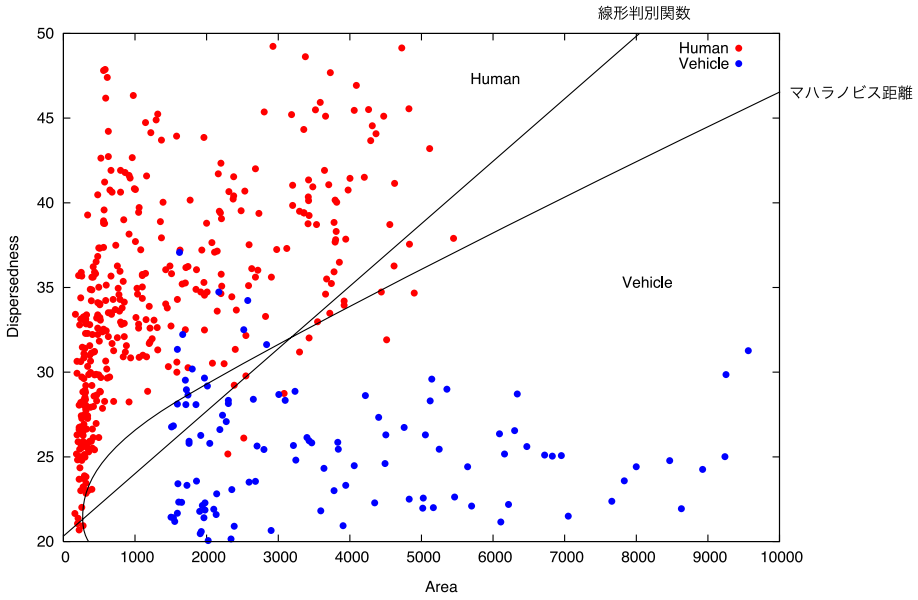


図 7.6: 線形判別関数とマハラノビス距離による識別境界

表 7.1: 線形判別関数とマハラビス距離による識別結果 [%]

method	human	vehicle	total
線形判別関数	95.3 (364/382)	83.6 (92/110)	89.5
マハラノビス	93.7 (358/382)	90.0 (99/110)	92.9

7.3 ニューラルネットによる物体識別

階層型のニューラルネットワークは，多数のサンプルによる学習により，高精度な識別器を構築することができる．また，その設計も比較的容易である．以下に，ニューラルネットワークによる物体識別への適用方法について述べる．

7.3.1 ニューラルネットワークについて

3 層ニューラルネットワークにおける逆誤差伝搬法 (バックプロパゲーション) の学習アルゴリズムについて簡単に述べる．バックプロパゲーションは，Rumelhart らにより 1986 年に提案された教師付き学習法である [2].

ネットワーク構成は，入力層，中間層，出力層からなる階層型ネットワークである．各ユニットへの入力は，式 7.17 に示すように下層ユニットの出力と結合係数を掛けたものの総和にしきい値を加えた値となる．中間層，出力層におけるユニットの出力関数は，式 7.18 に示すシグモイド関

数を用いる．μ はシグモイドの傾きである．ここで，使用する記号を以下のように定義する．

$$X_j = \sum_i W_{ji} \cdot Y_i + \theta_j \tag{7.17}$$

$$Y_j = \frac{1}{1 + e^{-\frac{1}{\mu} X_j}} \tag{7.18}$$

X_j : 上層ユニット j の入力総和
Y_i : 下層ユニット i の出力
W_{ji} : 下層ユニット i から上層ユニット j への結合係数 (重み)
θ_j : 上層ユニット j のオフセット
T_j : 出力層ユニット j における教師信号
Y_j : 上層ユニット j の出力

バックプロパゲーションでは，あるパターンを入力したとき出力層における出力値 Y_j と，教師信号 T_j の二乗誤差の総和により，ネットワークの性能を評価する．

$$E = \frac{1}{2} \sum_j (T_j - Y_j)^2 \tag{7.19}$$

この二乗誤差 E を減少する方向に，次式に従って結合係数を修正する．

$$\Delta W_{ji} = -a \frac{\partial E}{\partial W_{ji}} \tag{7.20}$$

a : 学習レート

7.20 を変形すると，

$$\begin{aligned} \Delta W_{ji} &= -a \frac{\partial E}{\partial X_j} \cdot \frac{\partial X_j}{\partial W_{ji}} \\ &= -a \frac{\partial E}{\partial X_j} \cdot Y_i \end{aligned} \tag{7.21}$$

となる．したがって ΔW_{ji} は次式のようになる．

$$\Delta W_{ji} = a \cdot \sigma_j \cdot Y_i \tag{7.22}$$

$$\sigma_j = -\frac{\partial E}{\partial X_j} \tag{7.23}$$

ここで，ユニット j が出力層の場合，σ_j は次式のようになる．

$$\sigma_j = -\frac{\partial E}{\partial Y_j} \cdot \frac{\partial Y_j}{\partial X_j} \tag{7.24}$$

$$= (T_j - Y_j) \cdot \frac{\partial Y_j}{\partial X_j} \tag{7.25}$$

ただし，7.18 より次式が成り立つ．

$$\frac{\partial Y_j}{\partial X_j} = \frac{1}{\mu} Y_j(1 - Y_j) \tag{7.26}$$

次に，ユニット j が中間層の場合，7.24 は次式のようになる．

$$\begin{aligned} \sigma_j &= -\sum_k \left(\frac{\partial E}{\partial X_k} \cdot \frac{\partial X_k}{\partial Y_j}\right) \cdot \frac{\partial Y_j}{\partial X_j} \\ &= \sum_k (\sigma_k \cdot W_{kj}) \cdot \frac{\partial Y_j}{\partial X_j} \end{aligned} \tag{7.27}$$

ここで，k はユニット j が結合している一段上層のユニットである．中間層ユニット j は，上層の全ユニットと結合しているため，伝搬される誤差の総和を取らなければならない．

学習アルゴリズム

1. ネットワークの状態を決める結合係数 とオフセットを乱数値で初期化する．
2. 最初のパターンを学習パターンとする．
3. 学習パターンの値を入力層ユニットに入れ，出力層ユニットでの出力値を計算する．
4. 学習パターンの教師信号と出力層との誤差から，7.257.27 より，結合係数とオフセットを修正する．
5. 学習パターンを変更するときは 3. ～ 4. を繰り返す．
6. 学習繰り返し回数を更新して，制限回数以下であれば 2. ～ 5. を繰り返す．もしくは，二乗誤差の総和があるしきい値以下であれば学習を終了する．

7.3.2 ネットワークモデルと学習

物体識別に用いるニューラルネットワークは図 7.7 に示すような 3 層の階層型で，学習には 7.3.1 で述べたバックプロパゲーション法を用いる．ニューラルネットワークへの入力は，複雑度 (dispersedness)，面積 (area)，縦横比 (aspect ratio)，カメラのズームパラメータ (zoom magnification) の 4 パラメータの値を 0〜1 に正規化し，各入力ユニットへ入力する．出力層は人 (single human)，複数の人 (Human Group)，車 (Vehicle) の 3 ユニットとなる．教師信号には，入力が single human の場合，対応する出力ユニットを 1，その他を 0 とする．また，抑制パターンとして木の揺れ等の誤検出領域をリジェクトするように全ての出力ユニットを 0 となる教師信号を与え学習を行う．

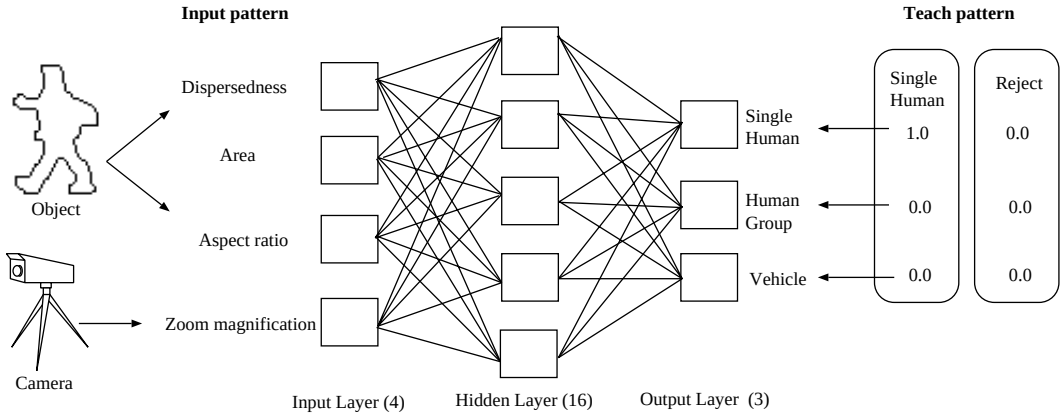


図 7.7: ニューラルネットワークによる物体識別

7.3.3　ニューラルネットワークによる識別

学習したニューラルネットワークに，検出した領域から複雑度，面積，縦横比を求め，カメラから得られるズーム率を入力する．各出力ユニットの出力を計算し，最大値を求める．その最大値により識別する．

$$\begin{cases} \text{最大値} > th & \text{リジェクト} \\ \text{最大値} < th & \text{最大値となる出力ユニットに対応するクラスに属する} \end{cases}$$

評価用データベースに対して，しきい値 th が 0.42 としたとき表 7.2 の結果を得た．また，その際のコンフュージョンマトリクスを表 7.3 に示す．Human Group は複数の人の配置によって Single Human もしくは Vehicle に誤識別する場合が他のクラスより多いことがわかる．

ニューラルネットワークの出力計算は，積和計算であるためハードウェア化に適している．より汎化能力の高いニューラルネットワークを構築するには，入力特徴量の選別や学習時に乱数を取り入れる等の工夫が必要である [3].

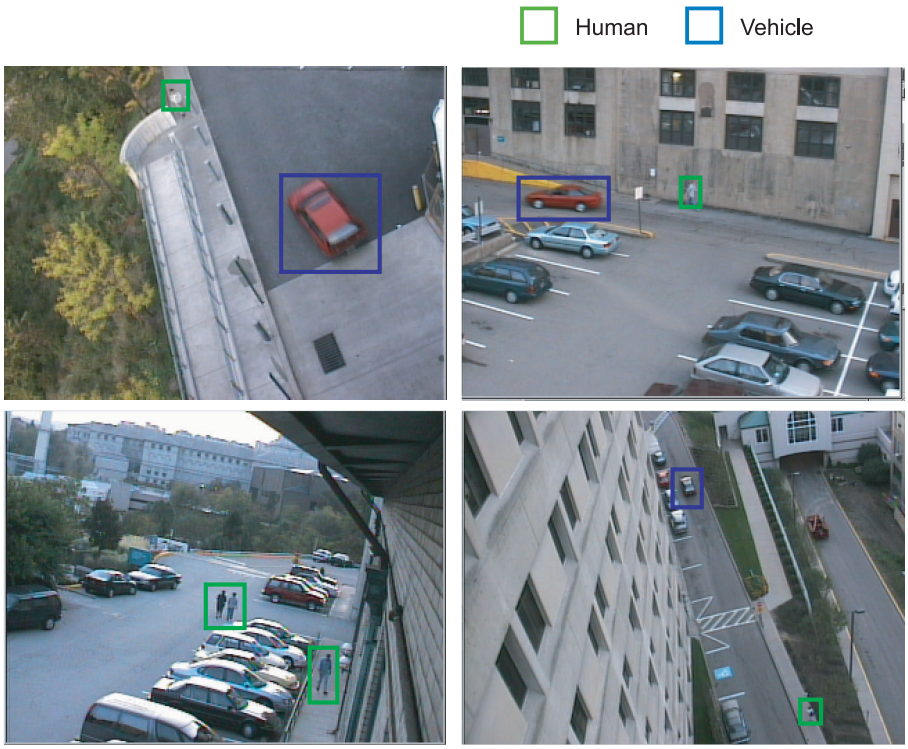


図 7.8: ニューラルネットワークによる物体識別

表 7.2: 識別結果

	Single Human	Human Group	Vehicle	Reject
識別率	99.0	85.0	99.2	67.4
個数	1149/1160	171/201	1019/1027	58/86

表 7.3: confusion matrix

	Single Human	Human Group	Vehicle	Reject
Single Human	-	3	0	8
Human Group	8	-	5	16
Vehicle	4	2	-	2
Reject	23	1	4	-

7.4 Support Vector Machine(SVM) 【Source Code】

SVM は 1960 年代に Vapnik 等が考案した Optimal Separating Hyperplane を起源とし，1990 年代になってカーネル学習法と組み合わせた非線形の識別手法へと拡張された．カーネルトリックにより非線形の識別関数が構成できるように拡張した SVM は，現在知られている手法の中で最もパターン認識性能の優秀な学習モデルの一つである．SVM は，基本的に 2 クラスのパターン認識器を構成する手法である．2 クラスのパターン認識器を構成するということは，学習パターンに対して最適な識別境界を決定することと同義である．まず，図 7.9(a) に示すような，線形分離可能な問題について考える．線形判別的関数を用いてこのような分布に識別境界を引く際，図 7.9(b) のように複数の識別境界を考えることができ，学習時においてこれらの境界は等価である．しかし，実際には図 7.9(b) 中において境界線 (1) や境界線 (4) よりも境界線 (2)，更には境界線 (3) の方がよい結果を示すことが多い．これらの境界線の差は，真の分布がサンプルで完全には表現されていないために発生する．境界線 (3) はサンプルから識別境界への距離（マージン）が大きく，そのようなサンプルと実データの差を飲み込むため認識性能が高くなる．このような，実データに対する頑健性を汎化性能という．SVM は，識別境界とサンプル群の距離を最大とすることで汎化性能の高い識別境界を構築する．また，図 7.9(c) のように非線形な問題には，カーネルトリックにより線形判別可能な問題へと変換することで認識を可能とする．以下では，まず線形な問題に対する

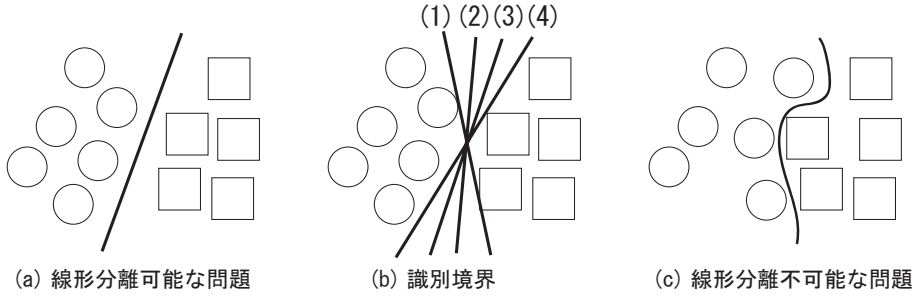


図 7.9: 2 クラスのパターン認識問題

マージン最大化 SVM について述べ，その後，非線形な問題に対するための拡張法としてカーネルトリックについて簡単に述べる．

7.4.1 線形学習マシンとマージン

線形クラス分類は以下のように書くことができる．

- 入力空間を $\mathbf{X}$，出力定義域を $\mathbf{Y}$ と記述する
- 入力を $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in \mathbf{X}$ とする
- 出力定義域は 2 値クラス分類に対して $\mathbf{Y} = \{-1, 1\}$ であるとする

識別境界

$$f(x) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \tag{7.28}$$

$$= \sum_{i=1}^{n} w_i x_i + b \tag{7.29}$$

決定関数

$$sgn(f(\mathbf{x})) = \begin{cases} \geq 0, & 1 \\ < 0, & -1 \end{cases} \tag{7.30}$$

この際の $f(\mathbf{x})$ が決定境界である．ここで，$\mathbf{x}$ は $\mathbf{x} = \{x_1, x_2, ..., x_i\}$ なる入力サンプル，y はラベルである．また，$\mathbf{w}$ は重みベクトル（weight vector），b はバイアス (bias) である．
線形分離可能な場合では，一般的に学習パターンをすべて正しく識別する超平面が複数存在する．片一方のクラスのデータに非常に近い判別関数を用いると，そのクラスに対する汎化性能が低下する．逆に，非常に遠い判別関数を用いればもう一方のクラスに対する汎化性能が低下する．この問題を，SVM は「マージンの最大化」を用いて解決し，汎化性能の高い識別境界を決定する．

7.4.2 線形 SVM：最大マージンクラス分類器 (maximal margin classifier)

最も単純にマージンを最大化する線形識別関数を求める，最大マージンクラス分類器について述べる．
与えられた線形分離可能なトレーニング標本 $S = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n))$ に対し，超平面 $(\mathbf{w}, b)$ は最適化問題

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{n} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1] \tag{7.31}$$

$$= \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \sum_{i=1}^{n} \alpha_i \tag{7.32}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{7.33}$$

を解くことで得られる．ここで，パラメータ α は 2 次最適化問題として (12) 式を最大化するような最適解とする．ただし，バイアス b の値は双対問題には現れないので，b^* は主問題の制約を用いて発見する．すなわち

$$b^* = -\frac{max_{y_i=-1}(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle) + min_{y_i=1}(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle)}{2} \tag{7.34}$$

また，ここで，カルッシュ - クーン - タッカー相補条件により，最適な解 α^* と $(\mathbf{w}^*, b^*)$ が，

$$\alpha_i^* [y_i (\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle + b^*) - 1] = 0, \quad i = 1, ..., n \tag{7.35}$$

を満たさなければならないという制約が存在する．これは，入力 $\mathbf{x}_i$ に対してのみマージンが 1 であり，したがって，対応するゼロでない α_i^* が超平面に最も近接したものとなることを表している．

これら近接したもの以外のパラメータ α は全て 0 である．したがって，重みベクトルの決定にこれらの点以外は関わらない．このことから，これらの点はサポートベクトル (support vectors) と呼ばれる．このようにして得られた重み集合が最適な識別関数となる．
したがって，命題に対する最適解 α^*, b^* により与えられる重みベクトル $\mathbf{w} = \sum_{i \in sv}^{n} y_i \alpha_i^* \mathbf{x}_i$ は，マージンを γ とすると

$$\gamma = \frac{1}{||\mathbf{w}||_2} = \left(\sum_{i \in sv} \alpha_i^* \right)^{-1/2} \tag{7.36}$$

を持つ最大マージン超平面を実現することとなる．

この問題は一般に，SMO アルゴリズムのような高速な逐次最適化アルゴリズムを用いて解くことが多いが，ここでは再急降下法によって解くことを考える．すると，アルゴリズムは

SVM

学習サンプル：$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$，ラベル：$y = \{-1, 1\}$
ラグランジ乗数$\alpha \leftarrow O$
`for` $t \leftarrow 1$ `to` T
`for` $i \leftarrow 1$ `to` m

$$\alpha_i \leftarrow \alpha_i + \eta_i (1 - y_i \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_i^t \mathbf{x}_i)$$

$$if \alpha_i < 0 \ then \ \alpha \leftarrow 0$$

`end for`

として，適当な停止基準を満たすまで最適化を行うことで実現できる．その際の重みベクトル $\mathbf{w}$ は

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{7.37}$$

で求められる．
最大マージン SVM を用い，図 7.10 のサンプルを分類した例を図 7.11 に示す．

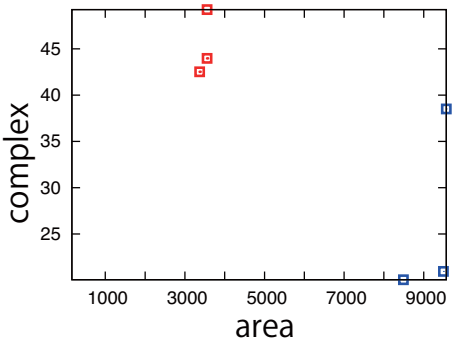


図 7.10: 線形分離可能な学習サンプル

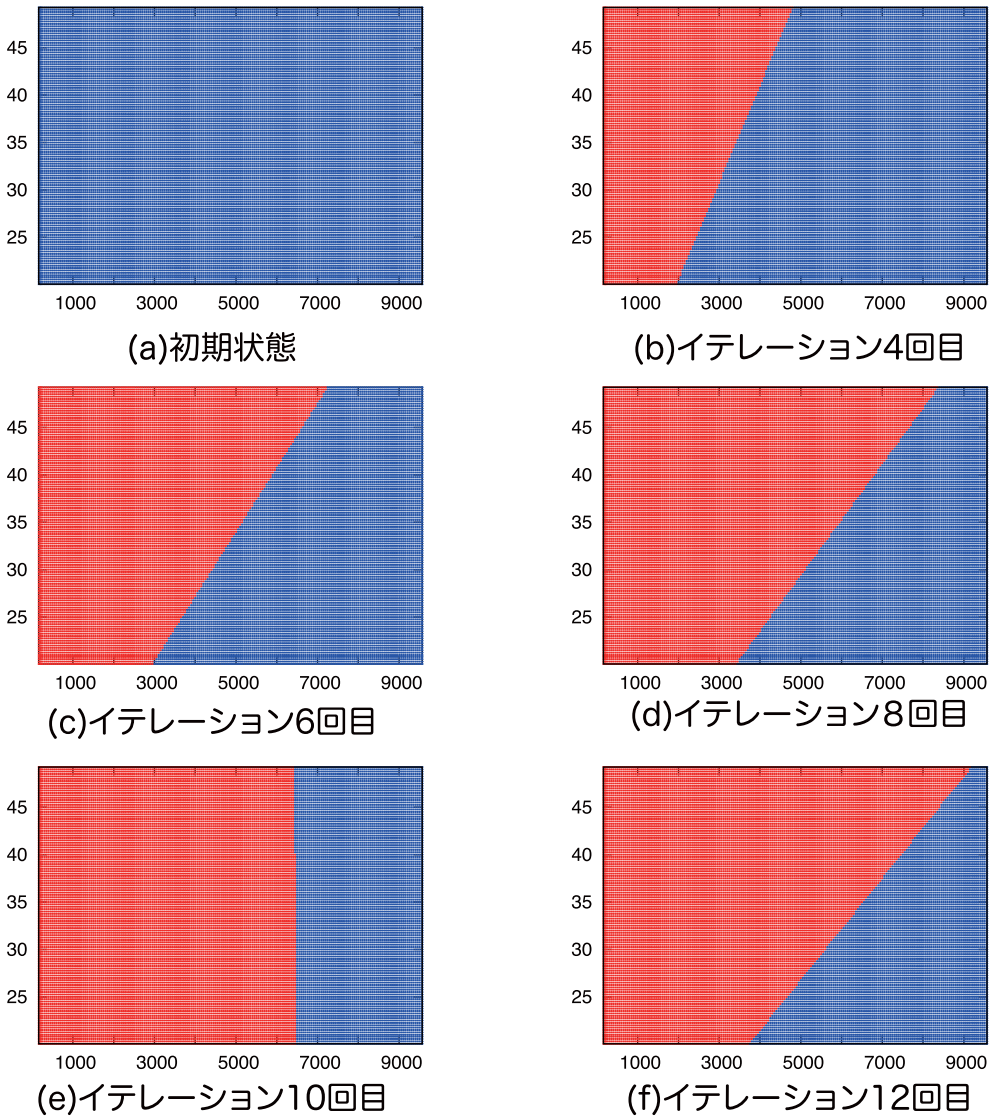


図 7.11: 最大マージン SVM による識別境界

逐次更新の中で両サンプルからの距離を最大にするよう識別境界が構築されていることがわかる.

7.4.3　線形 SVM：ソフトマージン最適化

最大マージンクラス分類器は，実世界問題に用いることができない．理由は，実世界問題は一般的に特徴空間で線形分離が不可能だからである．データが完全に分離されない限り，前節で述べたマージンは負の量を持つことになる．マージンに依存しすぎると，システムが少数の特異な点の影響を受けすぎる危険を招く．そこで，ノイズが存在する実世界問題に対応するため，ある程度の誤差を許して境界を設定するソフトマージンの概念を導入する．また，ソフトマージンに対して前節までの，誤差を許さないマージンをハードマージンとも呼ぶ．

マージンスラック変数 (margin slack variable) の導入

ソフトマージン最適化においては，ハードマージンと異なり，マージン最大化を行いながらも幾つかの標本が境界を超えて反対側に存在することを許す．この際の合致できなかった点，すなわちマージン γ が $\gamma \leq \frac{1}{||\mathbf{w}||_2}$ である点がどの程度であるかを測る尺度として，マージンスラック変数 (margin slack variable) を導入する．マージンスラック変数は

$$\xi((\mathbf{x}_i, y_i, (\mathbf{w}, b), \gamma) = \xi_i = max(0, \gamma - y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b)) \tag{7.38}$$

と定義する．この量が 0 を超える値を持つとき，標本 $\mathbf{x}_i$ は $(\mathbf{w}, b)$ で与えられる識別平面によって誤識別されることを表している．この値のユークリッドノルム $||\xi||_2$ は，全ての標本について考慮した分類ミスの量を表現することとなる．ハードマージンにおけるマージン最大化の主ラグランジアンは以下であった．

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1] && (7.39) \\ &= \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \sum_{i=1}^{n} \alpha_i && (7.40) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle . && (7.41) \end{aligned}$$

そこに，マージンスラック変数 ξ を導入し，解くことで

$$\begin{aligned} L(\mathbf{w}, b, \alpha, \mathbf{r}) &= \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{n} \xi_i \sum_{i=1}^{n} -\alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi] - \sum_{i=1}^{l} r_i \xi_i && (7.42) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle && (7.43) \\ \frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i = \mathbf{0} && (7.44) \\ \frac{\partial L}{\partial \xi} &= C\xi - \alpha = \mathbf{0} && (7.45) \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^{n} y_i \alpha_i = 0 && (7.46) \end{aligned}$$

を得る．この問題を α について最大化することで最適解を得る．

この問題をハードマージン同様に再急降下法によって解くことを考える．すると，アルゴリズムは

ソフトマージン SVM

学習サンプル：$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$，ラベル：$y = \{-1, 1\}$

ラグランジ乗数$\alpha \leftarrow O$

`for` $t \leftarrow 1$ `to` T

`for` $i \leftarrow 1$ `to` m

$$\alpha_i \leftarrow \alpha_i + \eta_i(1 - y_i \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_i^t \mathbf{x}_i)$$

$if \alpha_i < 0\ then\ \alpha \leftarrow 0$

$if \alpha_i > C\ then\ \alpha \leftarrow C$

`end for`

として，適当な停止基準を満たすまで最適化を行うことで実現できる．何のことは無い，$\alpha \leq C$ を強要するだけのことである．つまり，ソフトマージンの導入はアルゴリズム的にはラグランジ乗数 α の上限を設けるに過ぎない．しかしこれにより，入り組んだサンプルから適切にマージン最大化を行える．なお，ここで紹介したのは 1 －ノルムソフトマージンであり，2 －ノルムソフトマージンをはじめとした他の評価尺度は文献 [7] を参照されたい．

ソフトマージン SVM により，前節までで扱ってきた人，車両分類問題を解いた例を図 7.12 に示す．このような線形分離不可能な問題に対して，ハードマージン SVM ではそもそも判別不可能な境界しか引けない．しかし，ソフトマージンを導入することで線形分離不可能な問題に適当な解を与えることができる．

表 7.4: ソフトマージン SVM による識別率 [%]

合計	車両	人
81.5	72.7	84. 0

7.4.4 非線形 SVM：カーネルトリック

ソフトマージン法を用いることで，線形分離可能でない場合に対してもパラメータを求めることができるようになる．しかし，ソフトマージン法を用いたとしても，本質的に非線形で複雑な識別課題に対しては，必ずしも良い性能の識別器を構成できるとは限らない．本質的に非線形な問題に対応するための方法として，特徴ベクトルを非線形変換して，その空間で線形の識別を行う「カーネルトリック」と呼ばれている方法が知られている．この方法を用いることでサポートベクターマシンの性能，応用範囲は劇的に向上する．

一般に，線形分離可能性はサンプル数が大きくなればなるほど難しくなり，逆に，特徴空間ベクト

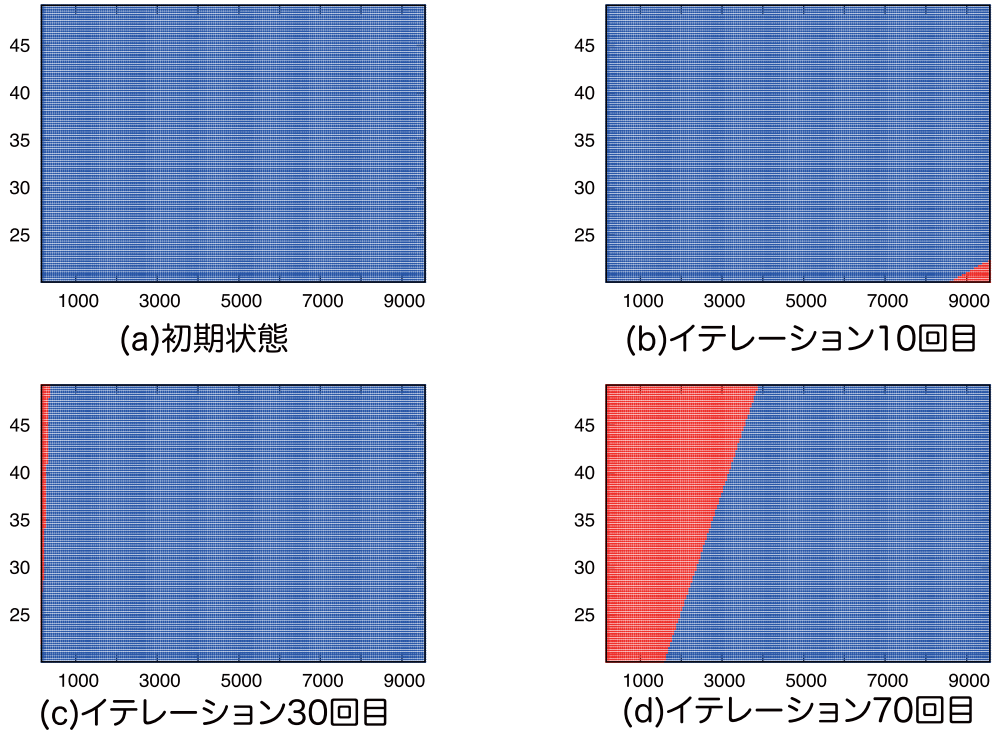


図 7.12: ソフトマージン SVM による識別境界

ルの次元が大きくなるほど易しくなる．例えば，特徴ベクトルの次元が標本の数よりも大きいなら，どんなラベル付けに対しても線形分離可能である．しかし，高次元への写像を行うと，次元の増加に伴い汎化能力が落ちてしまう．また，難しい問題を線形分離可能にするためには，標本と同程度の大きな次元に写像しなければならないので，結果的に膨大な計算量が必要となってしまう．ここで，入力データ $\mathbf{x}$ の特徴空間への写像を $\Phi(\mathbf{x})$ で表現する．すると，識別関数の決定規則は

$$f(\mathbf{x}) \quad = \quad \sum_{i=1}^{n} y_i \alpha_i \left\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \right\rangle + b \tag{7.47}$$

と表すことで評価できる．この特徴空間内の内積 $\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \rangle$ は標本点数と特徴次元 d の増加に伴い計算量が爆発的に増加し，一般には計算が困難である．そこで，この内積を入力点の関数として直接計算できるもの，カーネルの計算で代替するのがカーネルトリックである．すなわち

$$\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) \rangle = K(\mathbf{x}_i, \mathbf{x}) \tag{7.48}$$

となるような $K()$ がカーネルである．カーネルとしては，

多項式カーネル

$$K(\mathbf{x}_1, \mathbf{x}_2) = (1 + \langle \mathbf{x}_1 \cdot \mathbf{x}_2 \rangle)^p \tag{7.49}$$

Gauss カーネル

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(\frac{-||\mathbf{x}_1 - \mathbf{x}_2||_2}{2\sigma^2}\right) \tag{7.50}$$

などが用いられる．実際には，標本の分布に対応したものを実験的に求めて設定することが多い．これにより，非線形な決定境界を SVM によって設定することができる．

カーネルトリックは，極論すれば写像空間を特徴空間と見なす手法である．即ち，前節までで取り上げてきた識別時に用いる重みベクトル $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ は写像空間を考慮して写像数分の $\mathbf{w} = \{w_1, w_2, ..., w_m\}$ と見なすことになる．ラグランジ乗数更新アルゴリズムは基本的にカーネル $K(\mathbf{x}_i, \mathbf{x}_j)$ を内積の代わりに用い，

SVM

学習サンプル：$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$，ラベル：$y = \{-1, 1\}$

ラグランジ乗数$\alpha \leftarrow O$

`for` $t \leftarrow 1$ `to` T

`for` $i \leftarrow 1$ `to` m

$$\alpha_i \leftarrow \alpha_i + \eta_i(1 - y_i \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_i^t \mathbf{x}_i)$$

$$if \alpha_i < 0 \ then \ \alpha \leftarrow 0$$

$$if \alpha_i > C \ then \ \alpha \leftarrow C$$

`end for`

として，適当な停止基準を満たすまで最適化を行うことで実現できる．

また，サポートベクトル sv 以外による写像空間は識別平面に対し意味を持たないため

$$w_{i \in sv} \quad = \quad \alpha^t_{i \in sv} y_{i \in sv} \tag{7.51}$$

の集合というサポートベクトルの個数と同数次元のベクトルとなる．

また，当然線形 SVM も線形カーネル

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^t \mathbf{x}_2) \tag{7.52}$$

を用いたカーネル SVM として解くことができる．ガウシアンカーネルを用いたカーネル SVM により，人，車両分類問題を解いた例を図 7.13 に示す．図中 (a) はパラメータ σ が大なる場合，(b) は小なる場合である．それぞれ滑らかな曲線として境界が描かれており，識別性能も高い．比較すると，σ の値を小さくとることでより詳細な境界を引くことができるが，過学習に陥り易くなることが見て取れる．

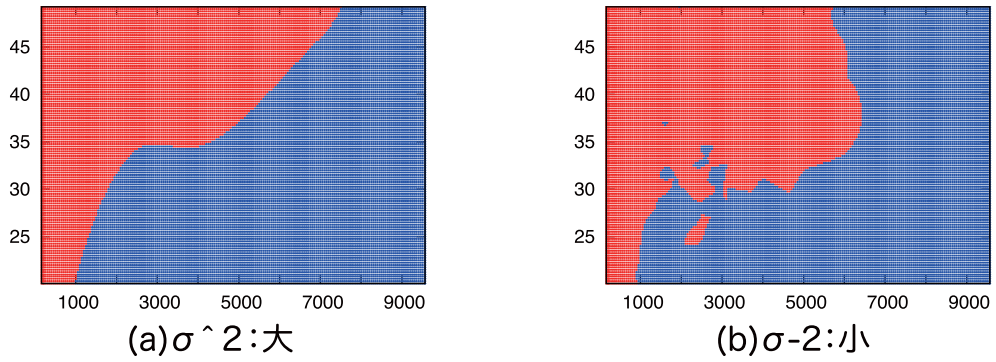


図 7.13: カーネル SVM による識別境界

表 7.5: カーネル SVM による識別率 [%]

	合計	車両	人
σ：大	94.5	95.4	94.2
σ：小	95.5	89.0	97.3

7.5　Boosting による識別

Boosting とは，少なくとも randomguess よりは高性能 (正解率 50% 以上) であるが，識別性能のニーズを満足するほどではない学習アルゴリズム（弱学習アルゴリズム）を複数用い，これらの集合からより正確な識別関数を生成する手法である．図 7.14 に Boosting による識別の概念図を示す.

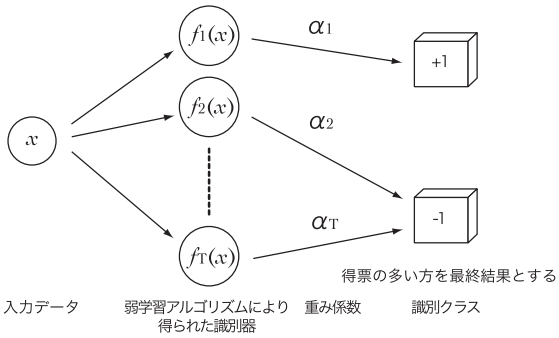


図 7.14: Boosting により得られる最終的な識別関数のイメージ

7.5.1 AdaBoost の学習 【Source Code】

AdaBoost とは，Boosting による識別の一種である [5]，AdaBoost の基本的な流れは，まず学習データから学習アルゴリズムによって識別関数を生成し，識別関数が誤識別を起こしたデータを重視して再学習を行う．この処理を反復した後，これらの識別関数のアンサンブルによって最終的な識別関数を生成する．最終的な識別関数は次式で表される．

$$F(x) = \sum_{t}^{T} \alpha_t f_t(x) \quad f_t(x) = \begin{cases} +1 \\ -1 \end{cases} \tag{7.53}$$

式（1）において x は入力ベクトル，$f_t(x)$ は t 番目の弱学習アルゴリズムによって得られた識別器，α_t はその重み係数である．以下に α_t の求め方を示す．まず，入力データ $x_i, (i = 1, \cdots, N)$ の重みを $D_{t,i}$ とし，次のように初期化する．

$$D_{t,i} = \frac{1}{m}, (t = 1, \cdots, m) \tag{7.54}$$

$$\sum_{i=1}^{m} D_t(i) = 1, D_{t,i} \geq 0 \tag{7.55}$$

次に，以下に示す手順によって T 個の弱学習器とそれらにかかる重み係数 α_t を求める．

AdaBoost

学習サンプル：$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$，ラベル：$y = \{-1, 1\}$, 識別関数 : $f(\mathbf{x})$

$D_i(i)$: データ i の重み $(\sum_{i=1}^{m} D_t(i) = 1, D_t(i) \geq 0)$

初期値 $D_t(i) \leftarrow \frac{1}{m}, \epsilon_t = 0.0$

`for` $t \leftarrow 1$ `to` T

- D_tを用いて f_tを学習
- 重み付き誤差ϵ_tを計算 ($\epsilon_t < 0.5$ が成り立つとする)

$\epsilon_t \leftarrow 0$

`for` $i \leftarrow 1$ `to` m

`if` $f_t(x_i) \neq y_i$

$\epsilon_t \leftarrow \epsilon_i + D_t(t)$

- $\alpha_t \leftarrow \frac{1}{2}\ln(\frac{1 - \epsilon_t}{\epsilon_t})$
- $D_{t+1}(i) \leftarrow D_t(i) e^{-\alpha_t f_t(x_i) y_i} / Z_t$
- 正規化定数：$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i f_t(x_i))$

`end for`

このような逐次過程において，初期の弱学習器は入力データ全体に対して識別を行う識別関数が得られ，t が大きくなるに従って誤識別を起こしたデータに重みが加えられ，識別が困難なデータに対応した識別関数が得られる．

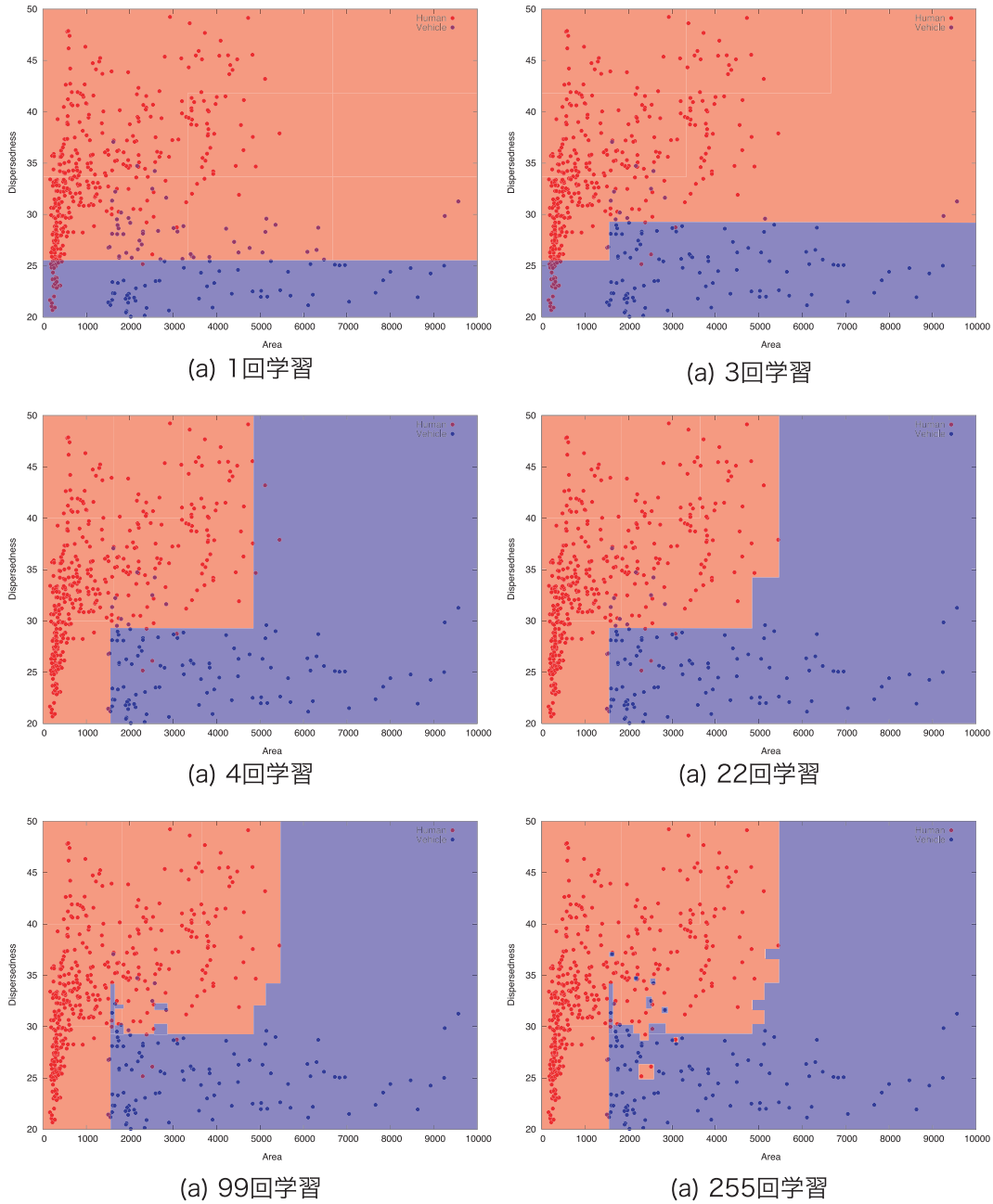
7.5.2　識別結果

図 7.15 に，複雑度と面積分布によって車両，人を判別する際の学習過程を示す．また各学習過程における識別率を表 7.6 に示す.

表 7.6: 各学習過程における識別率 [%]

Round	human	vehicle	total
1	95.5	50.0	84.3
3	93.2	90.9	92.7
4	98.4	92.7	97.2
22	99.0	92.7	97.4
99	99.0	96.4	98.3
255	100	100	100

図 7.15 より，AdaBoost では，学習の回数が少ない場合，単純な線形判別関数での結果しか得られないが，学習を繰り返すことで，線形識別関数が組み合わさり，高精度な識別器を構築することが可能となることがわかる.



(a) 1回学習　(a) 3回学習

(a) 4回学習　(a) 22回学習

(a) 99回学習　(a) 255回学習

図 7.15: AdaBoost による識別境界の遷移

7.5.3 Real AdaBoost

AdaBoost で用いる弱識別器の出力は 2 値（binary value）である．弱識別器の出力を実数 (real value) に拡張したものを Real AdaBoost[6] と呼び，区別のため 2 値判別器を用いた AdaBoost を Descreate AdaBoost と呼ばれる．Real AdaBoost は，Descreate AdaBoost よりも少ない弱識別器数においても同程度の精度を得ることができる．

弱識別器は，決定木（Decision tree）等が挙げられるが，ここでは確率密度関数を用いたものについて紹介する．まず，サンプル集合 $\mathbf{X}$ をラベル $y \in \{+1, -1\}$ に応じてポジティブサンプル $\mathbf{X}_+$，ネガティブサンプル $\mathbf{X}_-$ に分割する．次に，それぞれのサンプル集合に対し，任意の BIN 数を持つ 1 次元ヒストグラムにより表現される確率密度関数 $W_\pm$ を作成する．

$$W_+^{\mathrm{j}} = \sum_{i:\,\mathrm{j}\ \in J \wedge y_i = +1} D_t(i) \tag{7.56}$$

$$W_-^{\mathrm{j}} = \sum_{i:\,\mathrm{j}\ \in J \wedge y_i = -1} D_t(i) \tag{7.57}$$

ここで，t は学習回数，i は学習サンプルの番号，j は 1 次元ヒストグラムの BIN の番号，$D(i)$ は学習サンプルの重みを表す．弱識別器の出力 $h(\boldsymbol{x})$ は，次式より作成した確率密度関数の対応する BIN の値に応じた出力を得ることができる．

$$h(\boldsymbol{x}) = \frac{1}{2} \ln \frac{W_+ + \epsilon}{W_- + \epsilon} \tag{7.58}$$

ここで，ϵ は 0 除算を防ぐための定数である（例. $\epsilon = 10^{-7}$）．

次に，弱識別器の選択は，次式より得られる確率密度関数のポジティブクラスとネガティブクラスの分離度を表す評価値 Z を用いる．

$$Z = 2 \sum_j \sqrt{W_+^j W_-^j} \tag{7.59}$$

全ての弱識別器候補から，評価値を算出し，次式より最も小さな評価値の確率密度関数に対応する弱識別器を選択する．

$$h_t(\boldsymbol{x}) = \underset{h \in H}{\mathrm{argmin}} \mathrm{Z} \tag{7.60}$$

これは，2 つの分布 W_+, W_- が最もよく分離する分布を選択するに等しい．

ここで，弱識別器中に $\frac{1}{2} \ln$ が含まれていることに注意されたい．これは弱識別器は確率的な出力を行うため，分布比 $\frac{W_+ + \epsilon}{W_- + \epsilon}$ を指数評価したものになる．そして，弱識別器の出力は学習サンプルの重み D_t の更新に利用される．Descreate AdaBoost では，サンプルに対して 2 値に判別した結果を用いて，弱識別器の重み α により一律にサンプルの重み付けを行っていた．Real AdaBoost では，弱識別器の出力が実数値であるため，弱識別器の重み α が必要なくなり，次式より弱識別器の出力に応じてサンプルの重み付けを行う．

$$D_{t+1}(i) = \frac{D_t(i) e^{-h_t(\boldsymbol{x}) y_i}}{\sum_{m=1}^{M} D_t(m) e^{-h_t(\boldsymbol{x}) y_i}} \tag{7.61}$$

それにより，ひとつひとつのサンプルに対して最適な重みを付けることが可能なため学習が早く収束し，少ない弱識別器で高精度な識別が可能となる．

最終識別器は，弱識別器の線形和として次式により表される．

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} h_t(\boldsymbol{x}) \tag{7.62}$$

これより導かれる Real AdaBoost アルゴリズムは以下である．

Real AdaBoost

M 個の学習サンプル：$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M$，ラベル：$y = \{+1, -1\}$

$D_t(i)$: データ i の重み $(\sum_{i=1}^{m} D_t(i) = 1, D_t(i) \geq 0)$

初期値 $D_t(i) \leftarrow \dfrac{1}{m}$

`for` $t \leftarrow 1$ `to` T

−確率密度関数 W の作成

$$W_+^{\mathrm{j}} = \sum_{i:\,\mathrm{j}\ \in J \wedge y_i = +1} D_t(i)$$

$$W_-^{\mathrm{j}} = \sum_{i:\,\mathrm{j}\ \in J \wedge y_i = -1} D_t(i)$$

−評価値 Z の算出

$$Z = 2 \sum_j \sqrt{W_+^j W_-^j}$$

−弱識別器の選択

$$h_t(\boldsymbol{x}) = \underset{h \in H}{\mathrm{argminZ}}$$

−学習サンプルの重みの更新

$$D_{t+1}(i) = \frac{D_t(i) e^{-h_t(\boldsymbol{x}) y_i}}{\sum_{m=1}^{M} D_t(m) e^{-h_t(\boldsymbol{x}) y_i}}$$

`end for`

7.6 AdaBoost と SVM の比較

AdaBoost と SVM は共にマージンを最大化する識別器として知られる．SVM は陽にマージン γ を最大化し，AdaBoost はマージンの関数である Z を最大化する．これにより構築される境界を比較すると，図 7.16 の様になる．ここでの AdaBoost はハードマージンであり，弱識別器の性能次第で対象の判別誤差についての最適化が可能であることがわかる（ソフトマージン AdaBoost も亜種として存在するが主流ではない）．一方，ハードマージン SVM は再急降下法を用いた本ケースでは収束が遅く，完全に分離するには至らない．しかしながら，弱識別器に依存する分明確な境界が引かれている AdaBoost，カーネル空間への写像による滑らかな境界が引かれている SVM は本質的に同様の問題を解いていることが，識別境界から見て取れる．これらの手法は共にサンプルに

依存が大きく，ノイズに対し過学習に陥り易いという危険性を持つため，学習パラメータや学習進度の選定には注意が必要である．

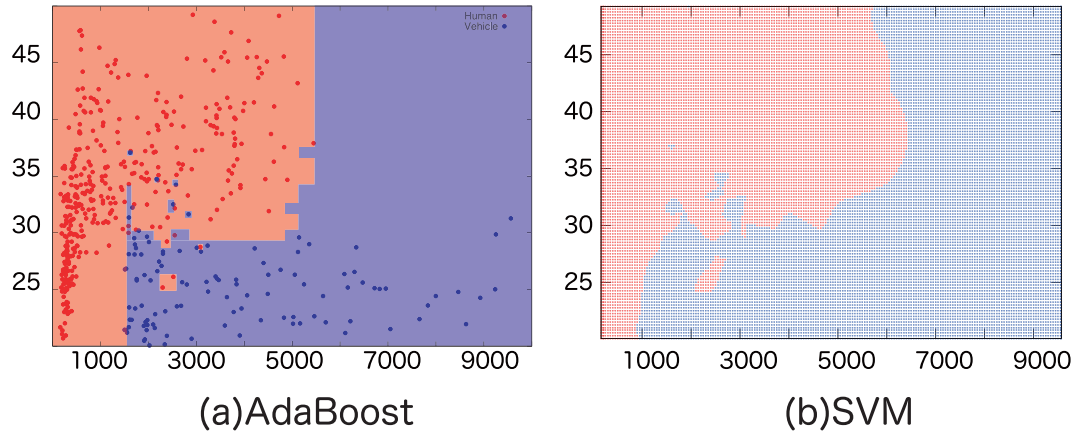


図 7.16: AdaBoost と SVM

7.7 識別器と特徴量の変遷

第 1 章で述べた VSAM における動体検出法は，入力画像と背景画像の差分を計算する背景差分ベースの手法であった．このような動体検出をベースとした動画像理解のアプローチは，移動体同士が画像上で重なった場合にセグメンテーションに失敗するため，その後の処理である物体識別が不可能となる問題があった．図 7.17 に背景差分による動体検出の失敗例を示す．図 7.17 の A，B，C のように人同士が画像上で重なった場合，1 つの動体として検出される．

(a) 入力画像　　(b) 背景差分画像

図 7.17: 背景差分による動体検出の失敗例

これに対して，Viola と Jones は Haar-like(図 7.18(a)) と呼ばれる局所特徴と統計的学習の組み合わせによる高速かつ高精度な顔検出法 [11] を提案した．この手法は，入力画像に対して検出ウィ

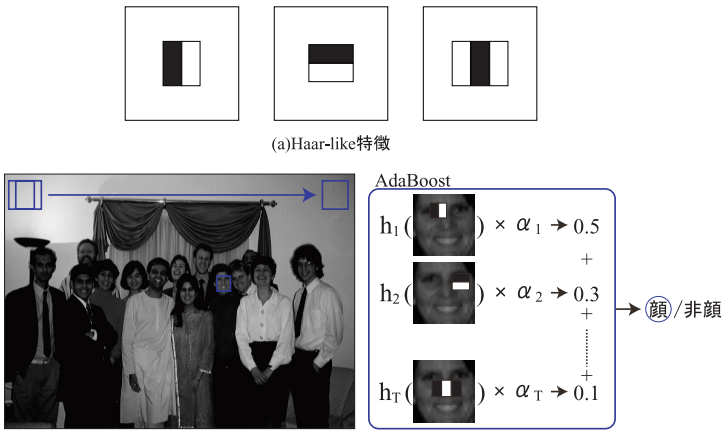


図 7.18: ラスタスキャンと AdaBoost による顔検出

ンドウを図 7.18(b) のようにラスタスキャンし，図 7.18(c) に示すように，AdaBoost を用いて検出ウィンドウ内の画像が顔/非顔として判別するため，前処理として動体検出を必要としない．顔検出の分野では，ラスタスキャン方式による手法が多く提案され実用化された．近年では検出対象が顔から人へと変わり，形状変化が大きいため検出がより難しいとされている人検出の研究が盛んになっている．本章では，近年のアプローチである局所特徴量と統計的学習を組み合わせた人検出法と，高精度化のための有効な特徴量の捉え方について述べる．

7.7.1　識別器と特徴量の変遷

図 7.19 に物体認識に用いられる識別器と特徴量の変遷を示す [12]．物体認識に用いられる第 1 世代の特徴量は，輝度分布や wavelet[13] など画像全体から得られるものや，研究者・開発者が特徴量の種類を決定し，2.1.2 の VSAM における物体識別で述べたような形状の複雑度のような意味のある特徴量を抽出していた．これは，リアルタイム性を求めるために，ニューラルネットワーク全体のサイズ (結合数重み) をコンパクトにするために，入力ユニットの数を小さくする必要があり，そのため，低次元数の特徴量が用いられていた．それに対し，2000 年以降に提案された第 2 世代では，Haar-like 特徴や HOG 特徴などの low-level な局所特徴量から，統計的学習法であるブースティングを用いて特徴選択を行うため，高次元 (数千〜数十万パターン) の特徴量を扱うことが可能である．さらに，第 3 世代 (2005 年以降) では，第 2 世代の low-level 特徴量を AdaBoost の特徴選択を利用して組み合わせ，識別に有効な mid-level 特徴量を自動生成する Joint Haar-like[15] や Joint-HOG[16]，Shapelet[17] が登場した．

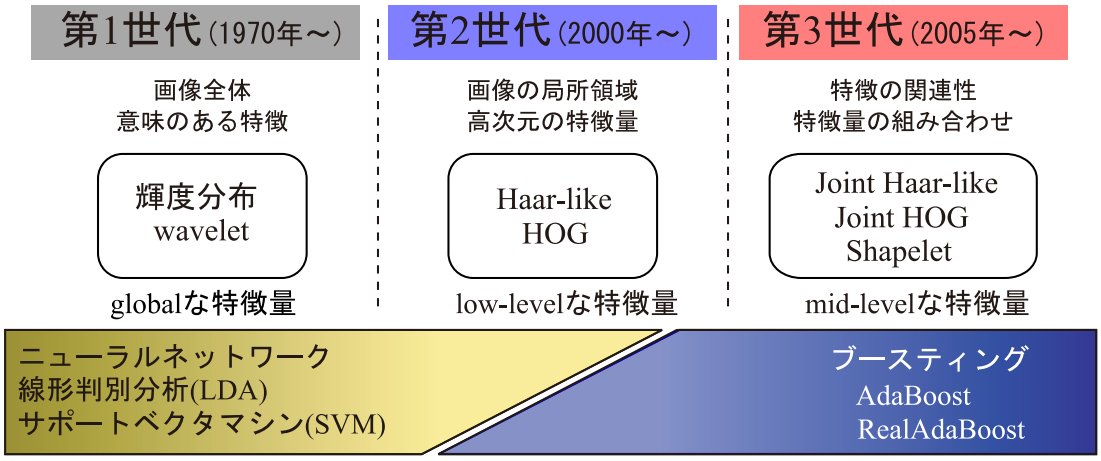


図 7.19: 識別器と特徴量の変遷

7.7.2　局所特徴量 (HOG) と統計的学習手法による人検出

人は動きとともに形状が変化する非剛体な物体であるため，顔検出と比較して難しい問題である．また，画像中での人同士の重なりによるオクルージョンの発生や衣服の違い，照明や影の影響も検出を困難とする要因である．このような問題に対して，Dalal 等により局所領域における勾配方向をヒストグラム化した Histograms of Oriented Gradients(HOG) 特徴量と統計的学習手法を組み合わせた人検出法 [14] が提案された．HOG 特徴量は，照明の変動による影響が少なく，局所的な幾何学的変化に頑健であるため高精度な人検出を可能とした．

図 7.20(a) に AdaBoost による HOG 特徴量の捉え方を示す．AdaBoost の弱識別器により 1 個の HOG 特徴量が選択され，最終的に多数ある弱識別器の重み付き多数決により人と人以外に判別する．

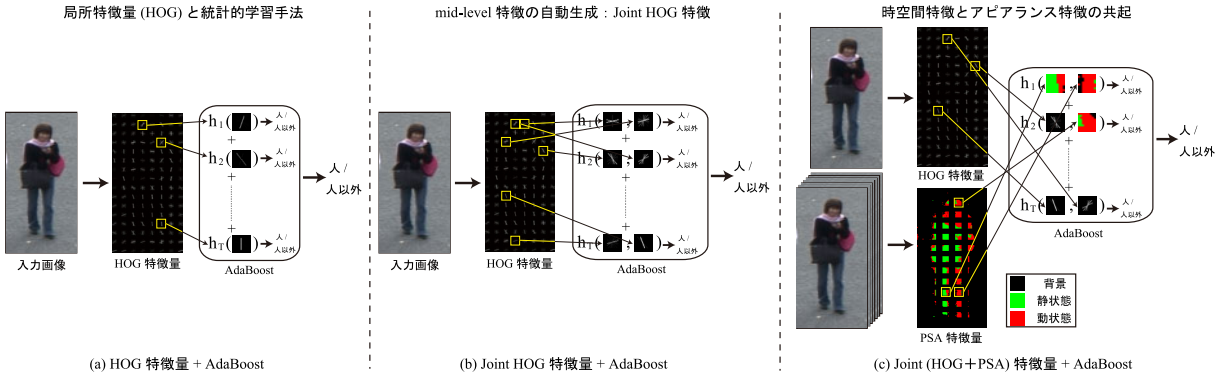


図 7.20: 人検出における特徴量の捉え方

7.7.3 mid-level 特徴の自動生成：Joint HOG 特徴

人には，形状の左右対称性や連続したエッジがあり，これらの特徴を捉えることで検出精度を向上させることができると考えられる．我々は，人独特の形状を捉えるために，複数の HOG 特徴量を組み合わせた Joint HOG 特徴量と，2 段階に構築した AdaBoost による学習法 [16] を提案した．複数の low-level な特徴量である HOG 特徴量を AdaBoost により組み合わせることで mid-level な特徴量である Joint HOG 特徴の POOL を自動的に生成し，この Joint HOG 特徴を入力とした 2 段階目の AdaBoost により，識別に有効な Joint HOG 特徴を自動的に選択して最終識別器を構築する (図 7.21)．図 7.20(b) に，AdaBoost による Joint HOG 特徴の捉え方を示す．HOG 特徴量で

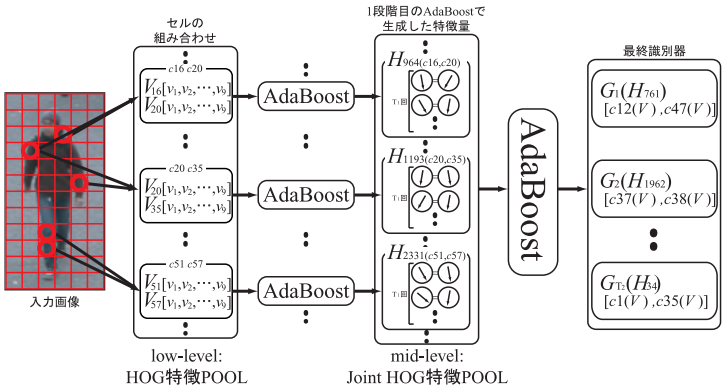


図 7.21: 2 段階 AdaBoost による Joint HOG

は，1 個の弱識別器が 1 個の HOG 特徴量を用いて識別したのに対し，Joint HOG 特徴では，1 個の弱識別器が位置の異なる 2 つの領域内に含まれる複数の HOG 特徴量を用いて識別を行う．これにより，従来の単一の HOG 特徴量のみでは捉えることができない物体形状の対称性や連続的なエッジを自動的に捉えることが可能となり，高精度な人検出法を実現した．図 7.22(a) に人の平均勾配画像，図 7.22(b)，(c) に AdaBoost により選択された HOG 特徴量を可視化した結果を示

す．HOG 特徴量の勾配方向を 9 方向で表現しており，輝度が高いほど AdaBoost における弱識別器の重みが高いことを表す．1 段階目で選択された HOG 特徴量 (図 7.22(b)) は全ての局所領域において選択されているが，2 段階目で選択された HOG 特徴量 (図 7.22(c)) では，人の輪郭に沿った HOG 特徴量が自動的に選択され，高い重みを持つことがわかる．

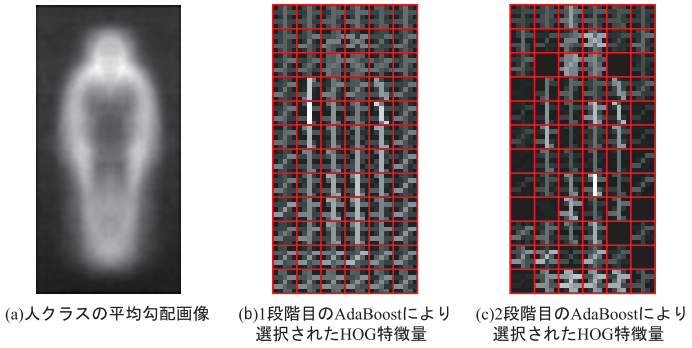


図 7.22: 選択された HOG 特徴量の可視化

7.7.4　時空間特徴とアピアランス特徴の共起

Joint HOG 特徴のフレームワークでは，人のアピアランスを表す HOG 特徴量 (図 7.23(b)) に，他の特徴を追加することが可能である．我々は，従来動体検出に用いられてきた，時空間特徴に基づく特徴量として図 7.23(c) に示すピクセル状態分析 (PSA) の結果を加えることにより，より高精度な人検出を達成した [26]．ピクセル状態分析とは，第 3 章で示したレイヤー型検出に用いられた

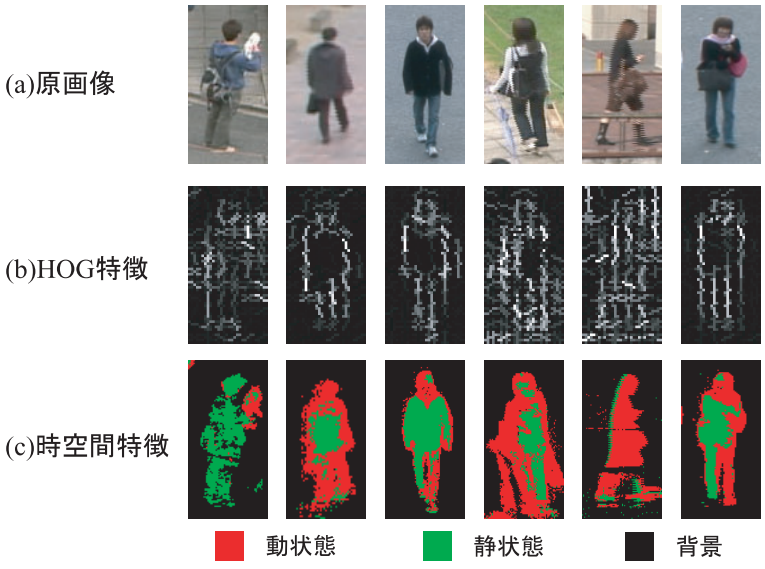


図 7.23: HOG 特徴とピクセル状態分析の例

手法であり，ピクセル状態の時間変化をモデル化し，各ピクセルを背景と動状態，静状態に判別する手法である．この時空間特徴とアピアランス特徴を図 7.20(c) に示すように同時に捉える手法を示す．時空間特徴を加えることにより，AdaBoost の弱識別器は，人のアピアランスと動きの情報を捉えることが可能となる．これにより，アピアランスの情報のみでは誤検出する人に似た物体に対して誤検出を抑制することができる．

AdaBoost により選択された特徴量に着目するために，図 7.24 に各学習ラウンドにおける HOG 特徴量と PSA 特徴量の選択された割合と，その際に選択された特徴量の可視化の例を示す．

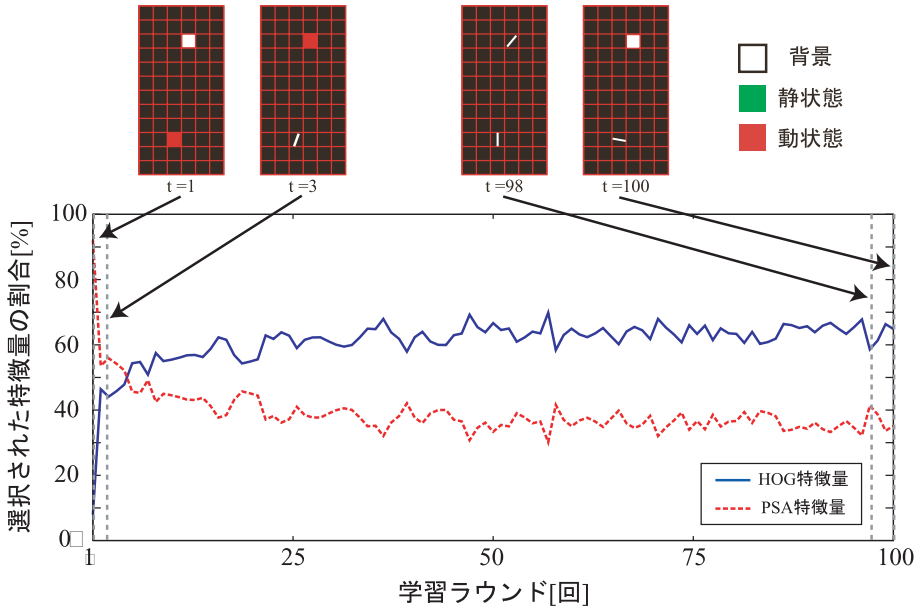


図 7.24: 選択された特徴量の割合

初期ラウンドにおける弱識別器では PSA 特徴が多く選択され，学習ラウンド数が進むにつれて HOG 特徴量が選択される割合が多い．これは，まず物体の動きを表すことが可能な PSA 特徴により，大まかに人と人以外を判別し，この後アピアランスの情報をもつ HOG 特徴量を用いて，より細かな識別境界を形成していると考えられる．

図 7.25 に本手法による人検出の例を示す．人の大きさの変化や人同士の画像上での重なりによる部分的な隠れに対しても高精度な人検出が可能である．

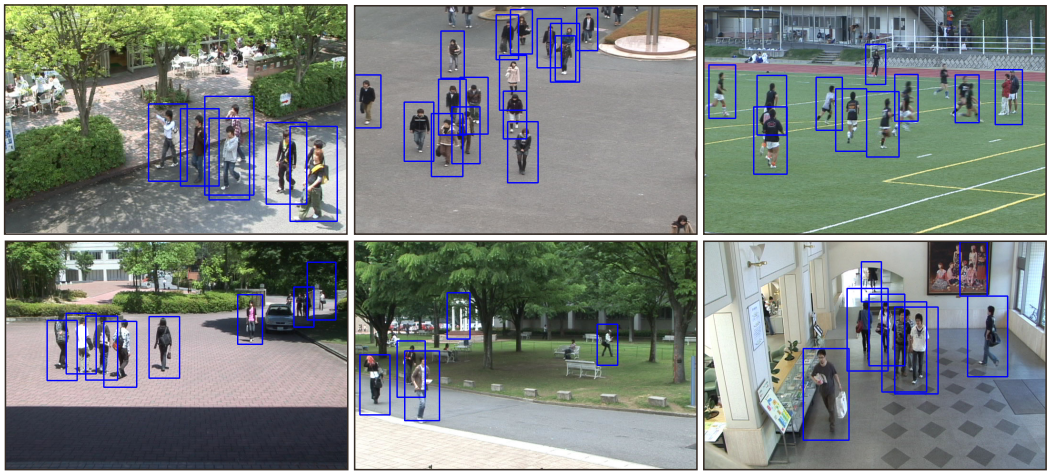


図 7.25: 人検出例

図 7.26 は，評価実験の結果を示す DET(Detection Error Tradeoff) カーブであり，原点に近いほど識別器の性能が高いことを示す．従来人検出に用いられている HOG 特徴量のみよりも，Joint HOG 特徴の方が識別性能が高いことがわかる．また，人のアピアランス特徴と時空間特徴量を同時に捉えることで，さらに高精度な人検出ができている．従来の HOG 特徴量に比べて Joint HOG+PSA 特徴は，誤検出率 5.0%において検出率を 99%まで向上させることができた．

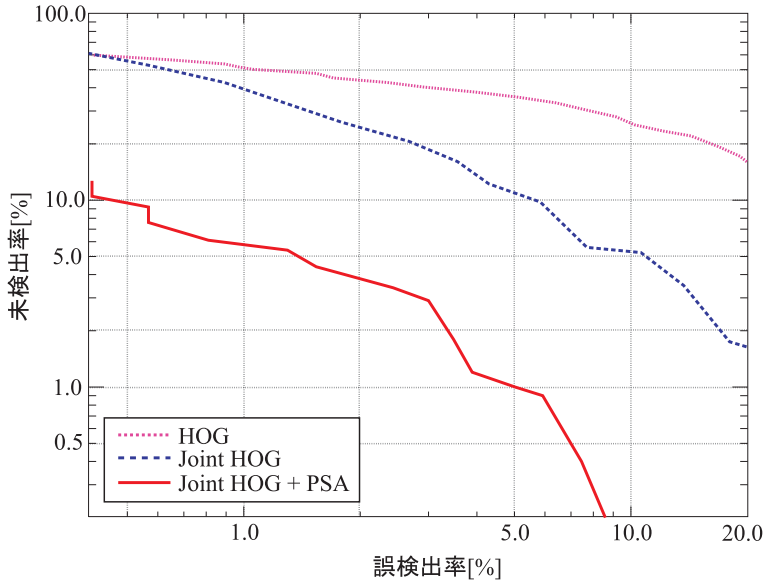


図 7.26: DET カーブ

参考文献

[1] 長谷川修, 金出武雄: “一般道路映像中の移動物体の識別・色の推定と特定対象の検出”, 情報処理学会論文誌, Vol. 44 No. 7, pp. 1795-1807 (2003).

[2] Rumellhart d.E.,McClelland J.L.and the PDP Research Group: “Paralell distributed processing”, 1, The MIP Prss (1986).

[3] 藤吉弘亘, 梅崎太造, 今村友彦, 金出武雄: “ニューラルネットワークによるナンバープレートの位置検出”, 電子情報通信学会, J80-D-II, 5, pp. 1169-1177 (1997).

[4] Y. Ohta and T. Kanade: “Color information for region segmentation”, Computer Graphics and Image Processing, Vol. 13, No. 3, pp. 222–241 (1980).

[5] Y Freund and R Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. Journal of Computer and System Sciences, Vol.23, No. 1, pp.119-139, (January 2001).

[6] R. E. Schapire and Y. Singer,“Improved Boosting Algorithms Using Confidence-rated Predictions”, Machine Learning, No. 37, pp. 297-336, 1999.

[7] Nello Cristianini (原著), John Shawe - Taylor (原著), 大北 剛 (翻訳): ”サポートベクターマシン入門”, 共立出版, 2005.

[8] B. Leibe, A. Leonardis, and B. Schiele, “Interleaved Object Categorization and Segmentation”, British Machine Vision Conference, Norwich, pp.759–768, Sept, 2003.

[9] B. Leibe, A. Leonardis, and B. Schiele, “Combined Object Categorization and Segmentation with an Implicit Shape Model”, European Conference on Computer Vision, Prague, pp.496-510, May, 2004.

[10] D. Comaniciu and P. Meer, “Mean Shift Analysis and Applications”, International Conference on Computer Vision, vol.2, pp.1197-1203, 1999.

[11] P. Viola and M. Jones, “Robust Real-Time Face Detection”, Int. Journal of Computer Vision, 57(2), pp. 137-154, (2004).

[12] 山下隆義, “統計的学習法を用いた物体認識における特徴量の進化”, 第 14 回画像センシングシンポジウム SSII08, OR4-01, (2008).

[13] H. Schneiderman, T. Kanade , “A statistical method for 3d object detection applied to faces and cars”, IEEE Computer Vision and Pattern Recognition, pp. 746-751, (2000).

[14] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection”, IEEE Computer Vision and Pattern Recognition, pp. 886-893, (2005).

[15] T. Mita, T. Kaneko, B. Stenger and O. Hori, “Discriminative Feature Co-occurrence Selection for Object Detection”, IEEE Pattern Analysis and Machine Intelligence, vol. 30, no. 7, pp. 1257-1269, 2008.

[16] 三井相和, 山内悠嗣, 藤吉弘亘, “Joint HOG 特徴を用いた 2 段階 AdaBoost による人検出”, 第 14 回画像センシングシンポジウム SSII08, IN1-06, (2008).

[17] P. Sabzmeydani and G. Mori, “Detecting Pedestrians by Learning Shapelet Features”, IEEE Computer Vision and Pattern Recognition , pp1-8, 2007.

第8章　おわりに

本テキストでは，1997年〜2000年に行われたVSAM(Video Surveillance and Monitoring) プロジェクトにおいて開発された動画像理解技術のアルゴリズムを中心に述べ，その実現例を見てきた．VSAM動画像理解技術は，2000年当時の高性能PC(Pentium III 600MHz) でリアルタイム処理されており，VSAMプロジェクトが終了してから既に7年も経っている今，組み込み系への実装が可能な段階にあるといえる．また，実際にVSAM技術が斬新なアプリケーションとして，Object Video社等の企業から販売されている段階にある．
最後に，本テキストが動画像理解技術を用いた応用に役立てば幸いである．

付録

A.1 Processing による動画像処理

A.1.1 Processing とは

Processing(http://processing.org/) はインタラクティブなグラフィックデザインを作成する過程を通じてコンピュータプログラミングの基礎を学ぶための環境であり，アイディアを描き試すためのスケッチブックでもある．アプリケーション本体は Java で，ユーザに用意されているプログラムコード自体も Java と同様の形式となる．そのため Java 言語を練習する環境として利用することも可能である．さらに，Java の性質を生かして Java アプレットへ出力する機能も備えられているため，作品をそのまま Web 上で公開できる簡易さも備えている．WEB カメラや QuickTime ムービーからの読み込みが簡易であり，簡単な動画像処理プログラミングにも適しているといえる．

A.1.2 操作インターフェイス

図 A.1 は Processing のインターフェイスを示している．

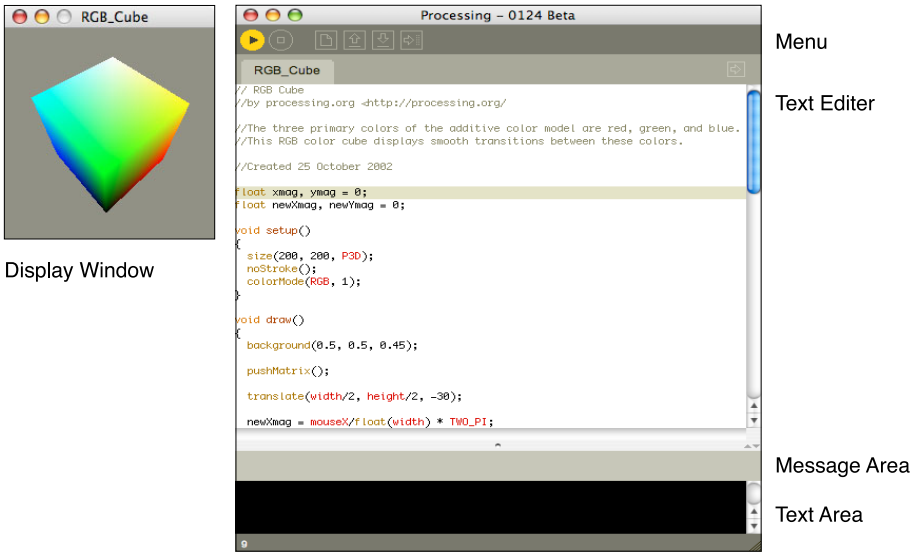


図 A.1: Processing のインターフェイス

図 A.2 にインターフェイス上部の 6 つのボタンの解説を示す.

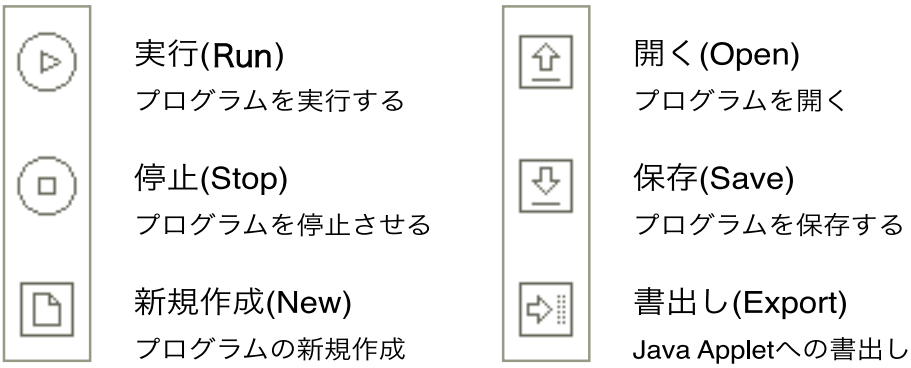


図 A.2: 各ボタンの役割り

書き出し (Export) ボタンをクリックすると，表示されているスケッチを Java アプレットとして出力する．またその際に Java アプレットを表示するために必要な最低限の HTML タグを書き出す．Processing に関する他の環境や命令についてのリファレンスは，下記の URL を参照してほしい.

英語: `http://processing.org/reference/index.html`

Processing は現在 β バージョンが無料で配布されている.

また，上記のサイトには Processingβ 版のユーザ間で情報交換を行うメッセージボードも設置されている.

A.1.3 画像の描画

画像を Processing を用いて表示するのはとても簡単である．Processing では，.gif, .jpg, .tga, .png のいずれかの形式の画像を読み込むことが可能である．woman.jpg という JPEG ファイルをウィンドウディスプレイ上に表示するプログラム image を作成するには，image という名のスケッチを保存する．これにより，図 A.3 のように My Documents フォルダ内 (Mac では Documents フォルダ内) の Processing フォルダに image という名のフォルダが作成される．画像を利用するためには，Processing のメニューバーの Sketch → Add File... を選択し，表示したい画像を読み込む必要がある．読み込んだ画像は，image フォルダ内の data フォルダにコピーされる.

画像ファイル woman.jpg を data フォルダ内に読み込めば描画プログラムの前準備が完了する:

静止画像 (woman.jpg) を表示するプログラムは次のようになる.

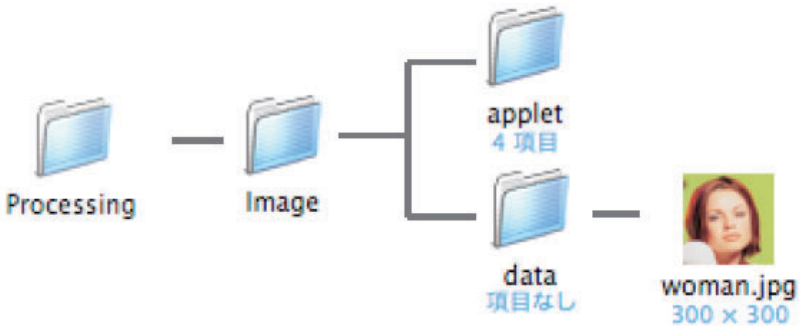


図 A.3: フォルダの構成

静止画再生用のプログラム

```
PImage b;

void setup() {
   b = loadImage("woman.jpg");
   size(b.width, b.height);
}
void draw() {
   background(255);
   image(b, 0, 0);
}
```

PImage は画像データを扱うためのデータ型である．b はこの画像につけた適当な名前である．image 関数により実際に画像をスクリーンに描画する．

```
image(PImage,  x, y, width,  height);
```

描画時の画像サイズを決める高さ height と横幅 width については省略することも可能である．省略した場合は元の画像のサイズが適用される．画像ファイルをフォルダの下に置く以外の方法に，パスの指定や URL の利用も可能である．

A.1.4 ピクセルデータの操作

PImage 型の画像ピクセルを操作するメソッドには，get() と set() が用意されている．

ピクセルデータの取得

get() は座標 (x,y) のピクセルの RGB 値を取得する．ある座標 (x=150, y=160) における RGB カラー値がそれぞれ R=255，G=200，B=131 のとき，以下に示す例にて指定したピクセルの RGB を取り出すことができる．

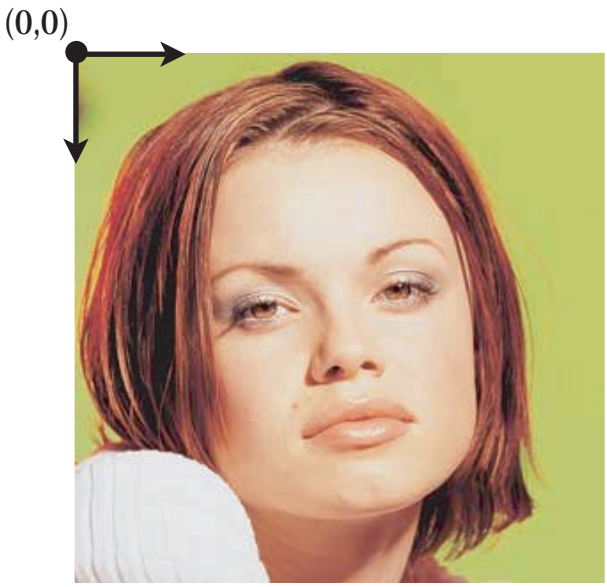


図 A.4: 画像の表示例

```
color c = get(150, 160);    -> RGB カラーを扱う変数 c に (255, 200, 131) を代入
```

- 座標 (150, 160) の赤色の値だけを取り出すには
  `int a = red(get(150, 160));    -> a に 255 代入される`
- 座標 (150, 160) の緑色の値だけを取り出すには
  `int a = green(get(150, 160));  -> a に 200 代入される`
- 座標 (150, 160) の青色の値だけを取り出すには
  `int a = blue(get(150, 160));   -> a に 131 代入される`

ピクセルデータのセット

set() は座標 (x,y) のピクセルに RGB 値をセットする.

```
color c = color(255, 255, 255);    -> RGB カラー変数 c に白色 (255, 255, 255) を代入
set(150, 160, c);                  -> 座標 (150, 160) にカラー変数 c の値をセット
```

A.1.5 文字の描画

ディスプレイウィンドウ上にフォントを指定して文字列を描画するには，text 関数を使用する.

タイポグラフィ

タイポグラフィのプログラム

```
PFont f;
void setup(){
   size(300, 300);
   f = loadFont("HelveticaNeue-UltraLight-48.vlw");
   textFont(f, 50);
}

void draw(){
   background(255);
   fill(0);
   text("Hello World", 10, 150);
}
```

}

`Font f = loadFont("フォント名");` フォントファイルを変数 f に格納

`textFont(f, size);` テキストの種類とその大きさを指定，テキストを描画する前に宣言

`text("文字列", x, y);` テキストを指定位置に描画

フォントデータのセット

使用するフォントは，メニューの Tools → Create Font を選択し，一覧から使用したいフォントを選択する．これにより画像と同様 data フォルダ内にフォントデータが作成される．

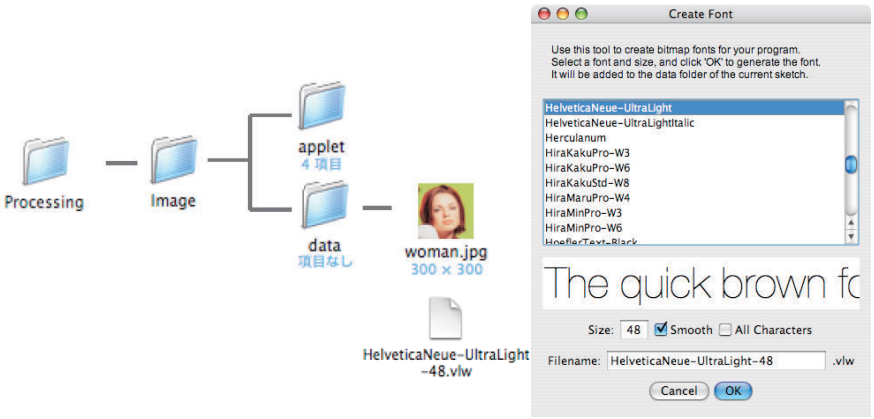


図 A.5: フォントの置き場所

以下に文字 “A” を用いて画像を表示するプログラムとその結果を図 A.6 に示す．

文字の表示のプログラム

```
PImage b;
PFont f;

void setup(){
   b = loadImage("woman.jpg");
   size(b.width, b.height);
   f = loadFont("HelveticaNeue-UltraLight-48.vlw");
   textfont(f, 30);
}

void draw(){
   background(255);
   for(int i = 0; i < b.height; i+=10){
      for(int j = 0; j < b.width; j+=10){
         fill(b.get(j, i);
         text("A", j, i+10);
      }
   }
}
```

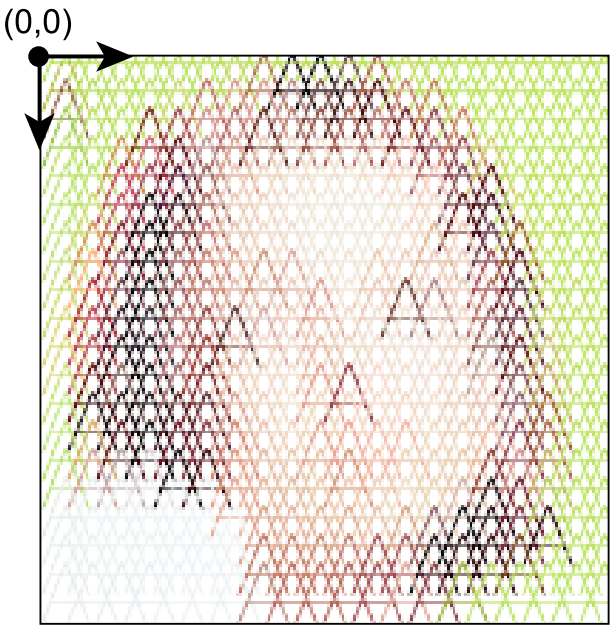


図 A.6: 文字 “A” で画像を表示

A.1.6 動画像の表示

以下に動画像を表示するサンプルプログラムを示す．動画像として，連番のJPEGファイル群，Quick Timeムービー，WEBカメラからの画像を取り込むことができる．Quick Timeムービーの表示は，現在のβバージョン(124)における動作は不安定である．図A.7に連番のJPEGファイル名の描画結果を示す．

動画像のサンプルプログラム

```
////////////////////////////////////////////////////////////////
// DisplayImages.pde
// Dec.2004, Mar.2007
////////////////////////////////////////////////////////////////
import processing.video.*;                    //ライブラリのインポート
//入力ストリームの選択
//再生形式選択 画像(.jpg)ファイル = 1, 動画(.mov)ファイル = 2, カメラ = 3
int input_stream = 3;

//画像ファイルのパラメータ
int g_width = 320, g_height = 240;            //画像サイズ
String DIR = "./DATA/WALK/";                  //フォルダのプレフィックス
String prefix = "f";                          //画像ファイルのプレフィックス
int frame_s = 1;                              //開始フレーム#
int frame_e = 528;                            //終了フレーム#
int _fps = 30;                                //frame rate (default:30fps)

PImage curImg;                                //現在フレームの画像を保存
int frame;                                    //現在フレーム

Capture video;                                //カメラからの映像を保存
Movie movie;                                  //動画を保存
Movie movie;                                  //動画を保存

void setup()
{
  frameRate(_fps);                            //フレームレートを設定
  size(320, 240);                             //スクリーンサイズ
  //入力ストリームが画像ファイルの場合
  if(input_stream == 1){
    frame = frame_s;
  }
  //入力ストリームが動画ファイルの場合
  if(input_stream == 2){
    movie = new Movie(this, "./DATA/ex.mov", _fps);  //読み込む動画・フレームレート設定
    movie.loop();                             //動画のループ
  }
  //入力ストリームがカメラの場合
  if(input_stream == 3){
    video = new Capture(this, width, height, _fps);  //カメラサイズ・フレームレート設定
  }
}
```

続・動画像のサンプルプログラム

```
//キャプチャーイベント
void captureEvent(Capture video)
{
  video.read();                                    //カメラから現在フレームを読み込み
}

//ムービーイベント
void movieEvent(Movie movie)
{
  movie.read();                                   //動画から現在フレームを読み込み
}

void draw()
{
  //入力ストリームが画像ファイルの場合
  if(input_stream == 1){
    getNewImg();
    image(curImg, 0, 0);                           //画像の描画
  }
  //入力ストリームが動画ファイルの場合
  if(input_stream == 2){
    image(movie, 0, 0);                            //動画像の描画
  }
  //入力ストリームがカメラの場合
  if(input_stream == 3){
    image(video, 0, 0);                            //キャプチャ映像の描画
  }
}

//画像ファイル取得
void getNewImg()
{
  String filename = DIR + prefix + nf(frame, 4) + ".jpg";  //ファイルパスを指定
  curImg = loadImage(filename);                    //指定画像を読み込む
  if(frame == frame_e){
    frame = frame_s;
  }else{
    frame = frame + 1;
  }
}
```

図 A.7: 連番画像ファイルの表示化

Web カメラを利用するためには以下の条件を満たす必要がある．カメラ自身のドライバのインストールに加え，下記 URL にてカメラが取得したムービーを Quick Time 形式に出力に変換する

機能を追加する必要がある (Windows User). "http://www.vdig.com/WinVDIG/"

A.1.7 動画像処理サンプル

本テキストで述べた動画像処理技術の一部は，下記 URL よりサンプルコードと実行例が確認できる．

`http://www.vision.cs.chubu.ac.jp/VU/`

A.1.8 サンプルコードの実行方法

ここでは，TemporalDifferecing を例に，実際にサンプルコードを Processing で実行する方法を紹介する．使用するコンピュータに Processing がインストールされていない場合は，あらかじめ最新版の Processing をダウンロードし，インストールをしておく．

ソースファイルとデータファイルの準備

http://www.vision.cs.chubu.ac.jp/VU/html/index.html から TemporalDifferecing のソースをダウンロードする． ダウンロードした zip ファイルを任意の場所で解凍する． 解凍すると「TemporalDifferecing」というフォルダができるので，そのフォルダを「`My Documents\Processing\`」内にコピーする．Mac の場合は「`~\Documents\Processing\`」 内にコピーする．また，このとき解凍してできたフォルダ名と，その中にある pde ファイル名は同じになるようにする（例えば，フォルダ名が TemporalDifferecing なら pde ファイル名も TemporalDifferecing.pde）．

次に，TemporalDifferecing で使用する画像データを http://www.vision.cs.chubu.ac.jp/VU/html/data.html からダウンロードする．ここでは例として WALK というデータを使用する． ダウンロードした WALK.zip を解凍すると，連番の画像ファイルが含まれる「WALK」というフォルダが作成される． この WALK フォルダを先ほどコピーした TemporalDifferecing フォルダ内の「data」フォルダ内にコピーする．このように，Processing で使用するデータは data フォルダ内に格納する．

シーケンスとして WALK を使用する場合は，ここまでの設定で実行することが可能である．他のシーケンスを使用する場合は，ソースファイル（*.pde）の設定を行う必要がある．

pde ファイルの設定

Processing を起動する．メニューバーの「`File`」->「`Sketchbook`」->「`TemporalDifferecing`」を選択すると pde ファイルが開く． ただし，フォルダ名と pde ファイル名が一致していない場合は選択できないため，必要に応じて修正する．
入力ストリームの設定を行う．

$$\textbf{int input\_stream} = \textbf{1};$$

入力ストリームが 1 のときは，画像データ（例．WALK）を使用する．Web カメラからの映像を使用するときは，入力ストリームを 2 に設定する．

※すべてのソースが Web カメラからの入力に対応しているわけではないため注意する．
画像データを使用するときは，以下の設定を行う必要がある．

$$\textbf{String prefix} = \text{"}\textbf{f}\text{"};$$

これは画像ファイルのプレフィクスで，例えば"f0001.jpg"というファイルならば"f"，"ex-0000001.jpg というファイルなら"ex-"と設定する．

$$\textbf{String DIR} = \text{"}\textbf{WALK}/\text{"};$$

これは画像ファイルが存在するフォルダのプレフィクスで，例えば画像ファイルが存在するフォルダが WALK なら"WALK/"，room なら"room/"と設定する．

$$\textbf{int frame\_s} = \textbf{1};$$
$$\textbf{int frame\_e} = \textbf{528};$$

frame_s は開始フレーム番号，frame_e は終了フレーム番号で，実際に使用する画像データにあわせて設定する．WALK の場合は 1〜528 となる．存在しないフレーム番号を設定した場合はエラーとなる．

$$\textbf{String filename} = \textbf{DIR} + \textbf{prefix} + \textbf{nf}(\textbf{frame}, \textbf{4}) + \text{".}\textbf{jpg}\text{"};$$

これは画像ファイル番号の桁数の設定と画像ファイルの拡張子の設定である．
Processing では".gif"，".jpg"，".tga"，".png"形式の画像ファイルを読み込むことが可能である．ここでは".jpg"ファイルを読み込む．また，nf() で画像ファイル番号の桁数を設定する．例えば読み込む画像ファイルが"f0024.jpg"ならば"nf(frame, 4)"，"ex-0000001.jpg"なら"nf(frame, 7)"のように，0 を含む桁数を設定する．

pde ファイルの設定は以上である．Run ボタンで実行し，ESC キーで処理が終了する．

B.1 分散式の展開

$$
\begin{aligned}
\frac{1}{K}\sum_{i=1}^{K}(I_{(t-i)}-\bar{I})^2 &= \frac{1}{K}\sum_{i=1}^{K}(I_{(t-i)}^2-2I_{(t-i)}\bar{I}+\bar{I}^2) \\
&= \frac{1}{K}\sum_{i=1}^{K}I_{(t-i)}^2-2\sum_{i=1}^{K}I_{(t-i)}\bar{I}+\bar{I}^2 \\
&= \frac{1}{K}\sum_{i=1}^{K}I_{(t-i)}^2-2\bar{I}^2+\bar{I}^2 \\
&= \frac{1}{K}\sum_{i=1}^{K}I_{(t-i)}^2-\bar{I}^2 \\
&= \frac{1}{K}\sum_{i=1}^{K}I_{(t-i)}^2-(\frac{1}{K}\sum_{i=1}^{K}I_{(t-i)})^2 \\
&= \frac{K\sum_{i=1}^{K}I_{(t-i)}^2-(\sum_{i=1}^{K}I_{(t-i)})^2}{K^2}
\end{aligned}
\tag{B.1}
$$