第 11 章　二足歩行ロボット＋USB カメラ/不審者を追尾する監視用ロボットの実験

画像処理とロボット制御の実際

神永 拓

実際のロボットで画像情報をどう使っているかを，2種のロボットを例として解説する．一つ目はヒューマノイド・ロボット「PINO」の頭部に USB カメラを取り付け，基礎的な物体追跡を試みた例を説明する．二つ目は，監視ロボット「Einstein」にカメラを一つ接続し，ステレオ視を試みた例を解説する．（編集部）

1. 二足歩行ロボット PINOでの応用

1.1　PINO に USB カメラをつける

もう 2 年ほど前になりますが，ヒューマノイド・ロボット「PINO」にカメラを搭載してトラッキング(物体追跡)を試みたことがあります．PINO は身長約 70cm の 2 足歩行ロボットで，各脚に 6 自由度，各腕に 5 自由度，胴体に 2 自由度，頭部に 2 自由度をもち，全身で合計 26 自由度を有します(**図 1**)．この PINO の頭部に USB カメラを固定し，次のような 2 通りのトラッキングを行うことにしました．

・頭部を用いた画像の追尾

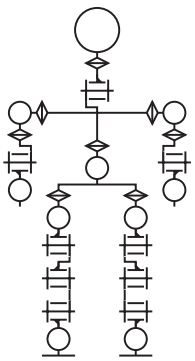

図 1　ヒューマノイド・ロボット PINO(第 2 章参照)

・認識した画像を PINO を移動させ追尾

いずれも単一のカメラを用いた 2 次元画像認識を用いました．

● 処理を PC で行うことがメリットになる

PINO はもともと有線のロボットで，駆動時には関節の目標値をパソコン(PC)から逐次送信します．これは，PC から制御を行うことにすれば，カメラをはじめとする各種センサの拡張を PC に対して行うことができ，既存のリソースを活用できることが理由です．また，処理を PC で行うということは計算リソースなどをさほど意識しなくてよくなるので，処理のアルゴリズムに集中することができることもメリットです(**図 2**)．

今回はカメラを USB 経由で PC に接続し，画像処理は PC 上で行い，PC から PINO に関節駆動指令を行う方式をとりました．こうすることにより既存の画像処理ライブラリを用いて，計算リソースをふんだんに用いることができます．今回はまさに PINO の設計思想が有効に働いた良い例であるといえるでしょう．

搭載するカメラは VGA(640 × 480)のものを用いました．USB1.1 のカメラだったため，フレーム・レートを上げるため，画像は QVGA(320 × 240)で取り込みました．

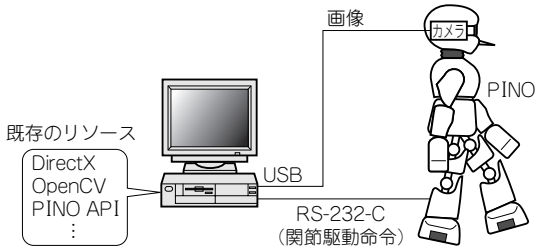

図 2　PINO のシステム構成

注 1：Intel の Open CV のページは http://www.intel.com/technology/computing/opencv/index.htm，Open CV のダウンロード・ページは http://sourceforge.net/projects/opencvlibrary/.

見本

1.2　色抽出と特徴点抽出を使って画像認識

● オープン・ソースの画像処理ライブラリを使う

カメラの画像はWindows Videoインターフェースで PC に取り込み，Intel 社の Open Computer Vision Library(Open CV)を用いて処理を行うことにしました．Open CV は非常に強力な画像処理ライブラリです．また，フリーですので興味がある方はダウンロードして試してみるとよいかもしれません[注1]．

トラッキングする対象は，PC に表示されているカメラの画面上から手動で指定することとしました．これは，表示画面上から，マウスで領域を選択することにより行います．画像認識には，色抽出と特徴点抽出の 2 種類を用いました．

まず，頭部を用いたトラッキングについての説明をしましょう．色抽出では，選択した色の領域の重心が画像の中心になるように PINO の頭部の 2 軸を駆動しました．次の時刻の物体の位置は，現在の位置の近くにある同じ色の領域が画像の中心になるように頭部を駆動します．特徴点抽出では，選択領域内の特徴点にもっとも類似する領域が画像の中心になるように頭部を駆動しました．参考までに，Open CV には，`cvGoodFeatures ToTrack()`という特徴点を抽出するための関数が用意されています．

● テンプレート書き換えによって，対象物の見え方が変わる問題に対処する

特徴点抽出によるトラッキング時には，色抽出にはない問題が発生します．それは，トラッキング中にトラッキング対象物の見え方が徐々に変化してしまうため，特徴点が一致しなくなってきてしまうという問題です．この問題を解決するためには，トラッキングを始めたときの特徴点のテンプレートを書き換えていく必要があります．この処理は次のようなフローで行いました．

(1)領域を指定し，トラッキングする領域の特徴点を抽出する
(2)テンプレートにもっとも合致する領域を検索する
(3)頭部モータを駆動して合致領域が画面の中央にくるようにする
(4)現在の特徴点をテンプレートにする
(5)(2)から繰り返す

この処理を高速に行うことによって画面に対して平行な軸回り，たとえばロボットに背を向けるような動作をしてもトラッキングを継続することができます．しかし，処理の速度が遅いと，テンプレートの更新が間に合わず，トラッキングに失敗してしまいます．

このように，トラッキングを行うためには高速な処理が必要ですが，一方で，正確に特徴点を抽出するためには高解像度の画像が必要になります．処理の速度と画像の解像度はトレードオフとなるため，解像度の決定はノウハウと試行錯誤が必要なことが多い箇所です．

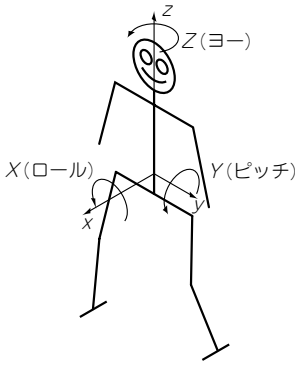

図 3　回転軸はロール軸 / ヨー軸 / ピッチ軸と呼ぶ

ここで，ロボットでよく用いる回転軸の定義を紹介しておきましょう．ロボットの進行方向周りの回転軸をロール，ロボットの体幹周りの回転軸をヨー軸，ロボットの前後方向の回転軸をピッチ軸と呼びます(**図 3**)．この表現を用いると，PINO の脚部の関節は，足首にロール軸とピッチ軸，膝にピッチ軸，股関節にロール軸，ピッチ軸とヨー軸といった書き方になります．

PINO を用いたトラッキングでは，順運動学計算や逆運動学計算は行わず，目標位置が中心よりも右にあるか左にあるか，上にあるか下にあるかで頭部のヨー軸とピッチ軸のモータを駆動することにしました．こうすると，1 回では正しい目標値に至らないのですが，繰り返し位置の補正を行うことにより，漸近的に画像の中心にトラッキング対象をもってくることができます．難しい言い方をすると，これは，ロボットの非線形性を制御系のロバスト性で安定化しているということもできます．

● 色抽出と特徴点抽出での得意・不得意――カラー・ボールと顔でちがう

このシステムでトラッキングを試みたところ，色抽出と特徴点抽出でそれぞれ得意・不得意が見られました．色抽出ではカラー・ボールなどはうまくトラッキングできるのですが，人の顔などはうまくトラッキングできませんでした．これに対して，特徴点抽出では人の顔は特徴点が多く抽出できるのでトラッキングがしやすかったのですが，カラー・ボールなどは特徴点が少なく，うまくトラッキングできませんでした．

また，色抽出では，トラッキング対象を見失っても

見本

再び見つけて捕捉することが可能ですが，特徴点抽出では見失った際にまちがった場所を捕捉してしまい，まちがった特徴点でテンプレートが上書きされたために対象を再捕捉できないといった現象が見られました．

こういった問題は，画像処理ライブラリそのものの問題ではなく，その使い方をもっと工夫しないかぎり解決できそうにありませんでした．残念ながら本プロジェクトは短期時限プロジェクトだったため，そこを追及することができませんでした．

1.3　二足歩行での課題はトラッキング失敗とカメラのブレ

次に，同じ認識ルーチンを使って，認識した物体に対してロボットを歩行させることを試みました．頭部によるトラッキングのルーチンを生かすために，サブサンプション・アーキテクチャ(Subsumption Architecture)と呼ばれる分散制御を行うことにしました．また，頭部のトラッキングの際には，トラッキング性能が特徴点抽出のほうが高かったため，歩行時には特徴点抽出を用いることにしました．

制御のアルゴリズムは，次のとおりです．

・頭部は上記のアルゴリズムと同等にトラッキングを行う

・ロボット自身は頭部と体幹のヨー軸の角度が0になるように旋回を行いつつ歩行を行う

頭部のトラッキングに高いプライオリティを与えることで，まずは動作の速い頭部でトラッキングを行うことができます．ついで，カメラが体の正面を向くように歩行の制御が行われます．

サブサンプション・アーキテクチャは，ロボットのように自由度が高く複雑な対象物を階層に分けて制御する場合に有効です．一方で，各制御系が独立・平行して動作するので，システム全体での安定性を保証することは難しくなります．

図4　監視ロボット Einstein の外観

実際にこのシステムを用いてトラッキングを行ってみた結果，トラッキングに失敗してしまうことがわかりました．原因としては次の点があげられます．

・カメラのシャッタ・スピードが低く，像がブレてしまい特徴点が正しく抽出されない

・カメラのフレーム・レートが低く，特徴点のテンプレートの更新が間に合わない(更新前に見え方が変わってしまう)

いずれの問題も，頭部のみのトラッキングのときも発生していたはずなのですが，歩行時のカメラのブレが大きいことが問題を悪化させている大きな原因です．カメラがブレる原因としては，

・歩行を行う際には足を上げるために重心を移動する必要があるため，上体の移動が必要

・頭部のモータの最高速度がトラッキングの必要な速度に足りていない

・頭部モータの減速機のバックラッシが大きい

の3点があげられます．このうち2番目の項目については，設計変更により改善が可能な項目です．また，細かく速いブレに関しては，ジャイロなどを用いて制振することもできます．しかし，上体の左右移動は歩行に必要なので，改善は難しい項目です．つまり，歩行中にトラッキングを行うためには，フレーム・レートやシャッタ・スピードを上げるか，それを補う方法を用いる必要があります．フレーム・レートを補う方法としてはオプティカル・フローと呼ばれる画像の流れを用いて，移動を予測する方法などがあります．

このプロジェクトにおいては基礎的なトラッキングに成功したと同時に，より精度の高いトラッキングを行うために必要な課題点にぶつかることになりました．これらの経験と技術は，次節の監視ロボット「Einstein」の開発に生かされることになります．

2. 監視ロボット Einstein を使った応用

2.1　さらなる画像の活用をめざして研究開始

PINO を用いたトラッキング実験の結果を見て，さらなる画像の活用と異なるセンサ情報融合の研究をかねて，監視ロボット Einstein を開発することになりました．Einstein は，監視ロボットとはいっても雲台にカメラがついた監視カメラといえます(**図4**)．ただ，当時の一般的なパン/チルト型の監視カメラと異なり，監視に関しての自律性を有しており，カメラの積極的

見本

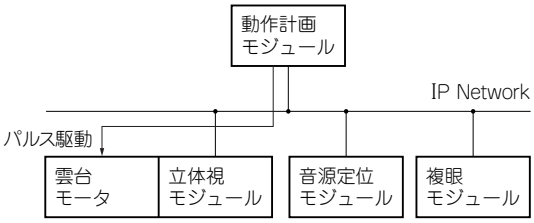

図 5　監視ロボット Einstein のシステム構成図

な駆動を意図しているのが特徴でした．

Einstein は立体視モジュール，音源定位モジュール，複眼カメラ・モジュールと動作計画モジュールから構成されています(**図 5**)．今回は，この中で立体視モジュールと動作計画モジュールについて説明します．現在のコンピュータの性能では全モジュールを 1 台の PC で実行するには処理が重たいため，複数の PC に処理を分散する方式をとりました．各モジュールは UDP(User Datagram Protocol)を用いて相互に接続されています．将来的に PC の性能が上がれば，1 台の PC 上ですべてのプログラムを実行することもプログラムの変更なくできます．また，ネットワークを用いた構造とすることで，拡張性をもたせることが容易になりました．

2.2　ミラーを使い一つのカメラでステレオ視を実現する

立体視の原理は第 10 章のステレオ視の節で説明したとおりなのですが，これには次のような前提があります．

・両眼のカメラが同時に画像を取得する(フレーム同期)

・両眼のカメラの露出が同一になっている(AE 同期)

とくに，1 番目の項目は，この前提が崩れると移動物体の距離がわからなくなってしまいます(**図 6**)．しかし，特殊なカメラを用いないとフレーム同期を取るのは困難です．AE(Automatic Exposure)同期が崩れると，両眼での見え方が変わってしまうため，対応点を見つけられなくなる可能性が高くなります．これは，片目だけ明るい光を浴びた直後に両目でものを見ようとしても距離がうまくつかめないことに近い現象です．AE 同期はディジタル式のカメラを用いる場合は，より深刻となります．なぜなら，ディジタル式のカメラによってはフレーム・レートがシャッタ・スピードによって左右されるため，AE 同期が崩れるとフレーム同期にも影響を与えるからです．AE 同期を取るためには，片方のカメラをマスタとして露出情報を読み出し，もう片方のカメラの露出はマスタのカメラの露出

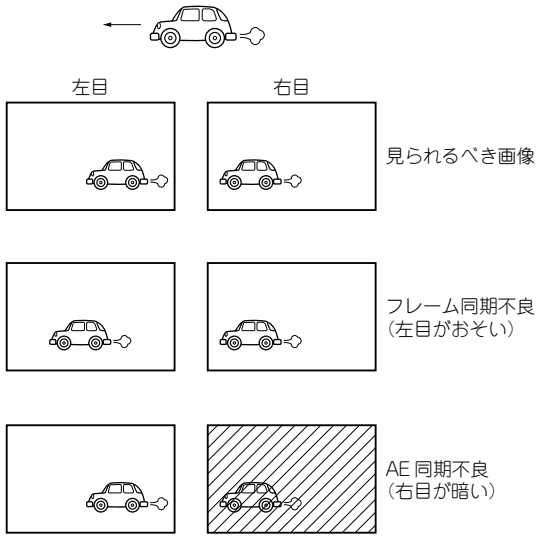

図 6　カメラの同期

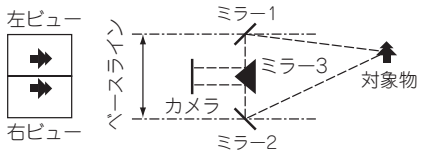

図 7　ミラーを使ったステレオ視アダプタ

にあわせて設定するような操作が必要です．

このような煩雑性を避けるため，Einstein では一つのカメラの画像をミラーを使って左右に分けて実現することにしました(**図 7**)．こうすれば，そもそも同じカメラで撮像するため同期の問題は発生しません．また，一般的なカメラを用いることができるため，使用できるカメラの選択肢が広がります．

2.3　スペックから部品選定へ

Einstein を開発するにあたって，まずはどのような環境でどのような動作をさせるかを考えました．

目的は，不審者の自動検出および自動追尾，そして記録です．検出と追尾にはステレオ視を用いて誤報や検出不備を防ぐことにしました．動作環境としては最大 10m × 10m までの広さで，天井高さが 2.5m の部屋の天井の中心に据え付けることを想定しました．これは，一般的な会議室の大きさと考えられます．また，検出する不審者は身長 170cm 程度で，最大移動速度が時速 10km，地面に伏せたときの高さが 20cm 程度と仮定しました(**図 8**)．このように仮定することで，部品の選定が可能になります．

まず，カメラの選定です．部屋の隅に伏せている不

見本

審者を検出するための解像度を検討した結果，ソニー㈱から発売されているカラー・カメラ・ブロックのレンズと解像度の組み合わせで，前述のミラーを用いたステレオ視を行えば足りることがわかりました．より小型なもので，かつなるべくレンズが広角なものを選択した結果，ソニー製のFCB-IX-10APというカメラを選択しました．このカメラを立体視アダプタに用いた場合の水平画角は約34degです．また，取り込みは640 × 480で30fpsで行えるボードを選択しました．さらにこのとき，ベースラインは120mm前後となり，ミラー部の大きさも現実的です．

次に，このカメラを2自由度で駆動するための機構を考えました．当初，実用領域内に特異点が存在しないという理由でジンバル型(ピッチ・ロール型)の機構を検討しましたが，カメラの視野内にモータが写りこんでしまう構造上の問題を駆動性能を犠牲にせずに解決できるメドが立たなかったため，通常のパン/チルト型の駆動機構を採用することにしました(**図9**)．

さらに，モータを選択するにはカメラの直下を，前述の仮定の不審者が画面の垂直方向に動く際に追従できる速度をもつこと，停止時に1フレーム中に画面の半分以下の遅れで最大速度の不審者に追従できる性能を要することを条件としました．また，バックラッシ(歯車をかみ合わせたときの歯面間の遊びのこと)は許容できないので，ダイレクト・ドライブか，ハーモニック・ドライブ，もしくは特殊な遊星歯車減速機が必要となりました．

選択したカメラの重量は95gだったので，取付金具や，根元側のモータがもう一方のモータを移動しなければならないということを考慮しても，負荷重量は最大で1kg程度になります．そこで，1kgの質量が直径20cm，長さ20cmの円筒に均等に分散しているものを仮想的な負荷としてモータを選定しました．

その結果，次のことがわかりました．

・負荷の慣性モーメントを駆動するには，ダイレクト・ドライブでは慣性比が足りない

・最大速度で移動する不審者を追尾するには最低60RPMの回転速度が必要

・三角関数状の加速パターンを用いて，停止状態から前述のフレーム内の追従遅れで追尾するためには0.62Nmの起動トルクが必要

これらを満たすモータとして，ハーモニックドライブシステム社のFHA-11C-100というACサーボ・モータを選択しました．

2.4 システムの構築――不審者発見/不審者追尾のモードを用意する

不審者を自動認識して追尾するためには巡回して不審者を発見するモードと，不審者を追尾するモードが必要になります．そこで，前者をサーベイランス・モード，後者をトラッキング・モードと呼ぶことにしました．

サーベイランス・モードはあらかじめ定義された点を巡回して不審者を探すので，初回に巡回して不審者がいない状態のテンプレートを作成します．定義された点は，ID番号を振って管理します．以降巡回中はそのIDのテンプレートに対して変化があるかの有無をチェックしていきます(**図10**)．後述しますが，ID = 99だけは特殊な定義位置として使用します．

デプス・マップは基本的には光の当たり方に無関係なのですが，どうしても影の部分ができるとデプスマップに変化をきたすので，巡回中不審者を発見しなかったらテンプレートを適応させていくことにしました．

不審者を発見したらその位置を目標位置として追尾を行うので，トラッキング・モードに遷移します(**図11**)．発見時には同時に不審者を2次元的にもトラッキングするための特徴点のテンプレートを作成します．以降は目標値を中心に移動する動作を行いながら不審者の認識を行います．不審者の認識は，3次元的に突起になっている箇所で，距離が直前の距離の周辺でかつ2次元的特徴点がテンプレートと類似している点と

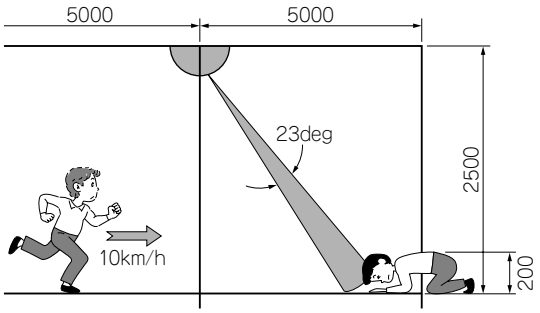

図8　想定した動作環境

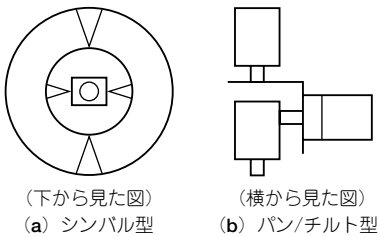

図9　雲台の構造

見本

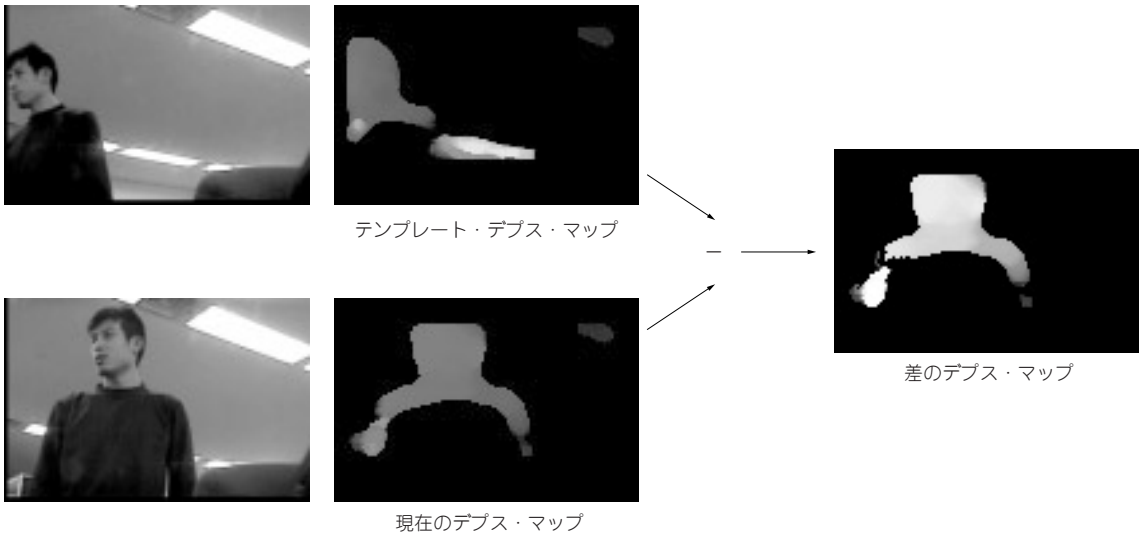

図 10　サーベイランス・モードでの動き検知

図 11　トラッキング中の画像（赤い箱がトラッキングしている点．第 2 章参照）

して行いました．PINO のときと同様に，特徴点のテンプレートは更新していきます．

不審者を追尾中に見失ったときは，いったんその位置で静止してテンプレートを作成します．これは ID＝99 という特殊な ID に対して行われます．そして，5 秒その位置（ID＝99）に静止して，移動物体がいないか確認します．こうすることで，一時的に不審者を見失った場合には再度捕捉できます．静止後も移動物体が発見されなかった場合は，サーベイランス・モードに復帰します．復帰時には不審者を発見した位置のテンプレートは作成しなおします．

この一連のフローを**図 12** に示します．将来的なシステムの拡張も想定して動作計画以外のモジュール，すなわちセンサのモジュールはすべて状況報告のみを行うことにしました．動作計画モジュールは受け取った情報を基にモータの駆動計画，モータの駆動，モードの決定，そして各センサ・モジュールへの指令を行うことにし，モジュール間での分業を定義しました．

2.5　駆動系のモデリング――運動学を取り入れる

Einstein においては PINO の場合と異なり，カメラの駆動に運動学・逆運動学を取り入れることにしました．

まず，座標系を定義しましょう．固定座標系を Σ_O，パン・モータの座標系を Σ_{pan}，チルト・モータの座標系を Σ_{tilt}，カメラの座標系を Σ_{cam} とします．パンとチルトの軸，カメラの光軸は 1 点で直交するように設計したので，この交点を固定座標系の原点とします．そしてパン・モータとチルト・モータは，モータの回転軸が z 軸になるように座標系を設定します．カメラ座標系に関しては，光軸方向が z 軸になるように定義します．また，パン軸の角度を θ，チルト軸の角度を ϕ と表すことにします（**図 13**）．

このように座標系を取ると，各同次変換行列は次のようになります．なお，焦点と座標系の原点のずれは無視できるものとします．

$$
{}^{O}T_{pan} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & -\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \cdots\cdots\cdots\cdots\cdots \quad (1)
$$

$$
{}^{pan}T_{tilt} = \begin{bmatrix} \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cos\phi & -\sin\phi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \cdots\cdots\cdots\cdots\cdots \quad (2)
$$

見本