ヒューマンインタフェース　95－5
コンピュータビジョンと
イメージメディア　129－5
（２００１．９．１３）

移動能動視覚によるプラント点検画像の蓄積

喜多伸之、喜多泰代、楊海圏
産業技術総合研究所　知能システム研究部門

あらまし：原子力プラントの各種点検情報を長期間に渡り保存し、運転・保守の安全性向上に役立てることを目指している。本論では巡回点検を行う移動ロボットに搭載した能動視覚により収集した画像情報を、効率よく蓄積し、自在な提示が行えるために、仮想環境を場として蓄積する手法を提案する。

Archiving Technology of the Plant Inspection Images captured by Mobile Active Cameras

Nobuyuki KITA, Yasuyo KITA, Yang HAI-QUAN
Intelligent Systems Research Institute
National Institute of Advanced Industrial Science and Technology

Abstract：Aiming to ensure the safety operation of nuclear power plants, we are developing the technology of archiving various information about plants for a long period. In this paper, we propose the system for archiving the image information gathered by mobile robots performing round inspection into virtual world so as to get better compression and more freedom of visualization.

1　はじめに

原子力プラントの安全な運転・保守のためには、プラントに関する情報を長期間にわたり蓄積し、それを自在に参照できれば有用である。近年、情報技術の急速な発達により、実世界の大量の情報を計算機に取り込み、蓄積することが可能となりつつある。ただし、一回の巡回点検で得られる情報を考えてみても、その量は膨大なものであり、計算機ハードウェアの進歩がいくら急速であるとはいえ、逐一蓄積したのではたちまち容量を超えてしまうことは明らかである。さらに、回が異なる巡回点検ではセンシングに関わる条件が変化しうるため、収集した情報を生で蓄積したのでは後に本質的な情報のみ取り出すのは至難の業である。このような問題を解決するために次の３つの技術が必要と考える。

(1) 環境の変化やセンシング目的に応じて効率良く点検情報を収集するための注意制御技術
(2) 収集した点検情報の時空間的な整合性を保つための変動除去技術
(3) 時空間的に広がった膨大な点検情報をコンパクトに蓄積するための情報蓄積技術

分散センサやロボットが集めた各種情報を長期間にわたり蓄積し、それぞれの情報を連携して有機的に活用するためには、すべての点検情報を統合的に計算機により管理することが不可欠である。情報の統合的蓄積の試みとしてマッピングシステムがある。これは情報源の 3 次元位置をインデックスとして情報を統合するが、それ自身広がりを持つ情報、例えば画像情報の扱いは考慮されていない。これに対して、CG において現実味の高い画像を合成するテクニックであるテクスチャマッピングは、実際の画像データを仮想物体表面に張りつけることにより、仮想世界上に蓄積する技術と捉えることもできる。我々はこれを基本として、画像情報のみならず空間的な場に依存した各種情報を仮想物体に貼り付け、さらに時間軸方向の厚みを持たせることにより、空間的な広がりを持った情報までも、時空間的に統合して蓄積することを目指す。このように蓄積された情報は、ビジュアリゼーション技術と組み合わせることにより、視覚的に自在に提示できるものとなる。

本論では、巡回点検ロボットが収集する画像情報を例として、仮想世界への情報の効果的な蓄積手法について提案する。

2　仮想環境への情報蓄積

2-1　何故仮想環境か？

仮想環境を画像情報を蓄積する場とするアプローチには次のようなメリットがある。

- 画像上に得られた情報を本来の 3 次元位置に還元して蓄積できるので、空間的・時間的に大きな圧縮効果が得られる。
- 多様なビジュアリゼーション技術が利用でき情報提示の自由度が拡大する。
- VR 技術、特にハードウェア技術の応用が容易となる。

2-2　環境サーバー

上記アプローチの基礎は、仮想環境を維持する機能である。我々はロボットが動き回る実環境のオンラインシミュレーションを目的に次のような機能を、環境サーバーとして開発してきた[1]。

(1) ロボットが動き回る環境を忠実にモデル化した仮想環境（幾何モデル・反射モデル・カメラモデル・照明モデルなどを VRML で記述）を維持する機能。
(2) 仮想環境内でロボットや能動視覚装置の実際の動きをオンラインで再現する機能。
(3) 能動視覚をはじめとして、ロボットの持つセンサの入力をシミュレーションする機能。
(4) サーバーとして多様なクライアントと情報の授受をおこなうための通信機能。

2-3　環境サーバーの拡張

本論では、巡回点検移動ロボットをはじめ、プラントの運用に携わる様々なエージェント（環境サーバーにとってはクライアント）から送られてくるプラントに関する情報を、仮想環境に効果的に蓄積するための機能、さらに蓄積した情報をエージェントからの多様な要求に応じて提示する機能を環境サーバーに付加する（図 1）。その主なものは次のようである。

- 入力情報と仮想環境の位置合わせ機能
- データの整合性を保つ機能
- 情報の仮想環境へのマッピング機能
- マッピングされた多様な情報の管理と検索機能

図 1　情報蓄積機能を付加した環境サーバーの概念図

3　画像情報の蓄積と提示

3-1　全体の流れ

クライアントからの画像蓄積や提示の要求は、環境サーバーが管理する仮想カメラを通して行う。ロボットに搭載したカメラで入力した画像を仮想環境上に蓄積したり、オペレータのシースルーＨＭＤに仮想環境を重畳表示する場合は、この仮想カメラの視野が実際の視野と正確に連動している必要がある。しかし、ロボットのオドメトリー情報や、オペレータ頭部の 3 次元位置姿勢センサ情報を頼りに仮想カメラを設定した場合、通常少なからぬ誤差を含む。この誤差を解消するために入力画像と仮想視野の間で位置合わせ処理が必要となる。このために様々な手法が研究されてきたが、対象が曲面だけで構成されており、かつ、表面のテクスチャがつかえない場合に適用できる手法はなかった。プラント内ではしばしばこのような状況が起こりうるので、我々は遮蔽輪郭を用いた 3D-2D 位置合わせ手法を開発した。シミュレーションでは所望の位置合わせ精度が得られており[2]、現在は実画像を対象として精度の追い込みを行っている[3]。

いったん、位置あわせされれば、入力画像から必要な部分を切り出し、仮想物体表面に貼り付けて蓄積する。

また、提示要求に対しては情報が貼り付いた環境をレンダリングした画像を返す。将来的には照明条件などの画像入力特性の変動を取り除いた上で蓄積したり、提示時に多様な修飾を行えるようにする予定である。

3-2 システム構成

図2にシステムの構成を示す。これまで環境サーバーは、Javaを本体として、仮想環境の記述であるシーングラフを維持したり画像を合成するためにJava3Dを、クライアントとのインタフェースにはORBacus（CORBAのフリー実装）を用いて開発してきた。Java3Dに限らず、通常の3Dグラフィックソフトウェアはテクスチャマッピング機能を持っているので、それを使って仮想物体表面に容易に入力画像情報を貼り付けることができる。ただし、異なる時間の情報や、異なる種類の情報をテクスチャとして蓄積したり提示するためには、同じ場所に何枚ものテクスチャを貼り付けたり、それを切り替えてレンダリングできる必要があるが、Java3Dにはそのような機能はない。そこで、関係データベースサーバーを環境サーバーのバックエンドサーバーとし、そこに大量のテクスチャを格納し、提示の際には、関係DBから必要なテクスチャを検索し、Java3Dによりマッピングしてレンダリングする。DBサーバーには関係データベースSQLの拡張言語をUNIX系に実装したフリーソフトPostgreSQLを採用し、環境サーバーからはJDBCを介してアクセスする。現在、環境サーバーはWindowsNTマシン上に、DBサーバーはLinuxマシン上に稼動させている。

位置合わせについては、Linuxマシン上にCとMESAをもちいて環境サーバーとは独立して開発してきた。今後もしばらくその開発環境を維持するために、位置合わせプロセスをCORBAサーバーとし、環境サーバーのバックエンドサーバーとする。

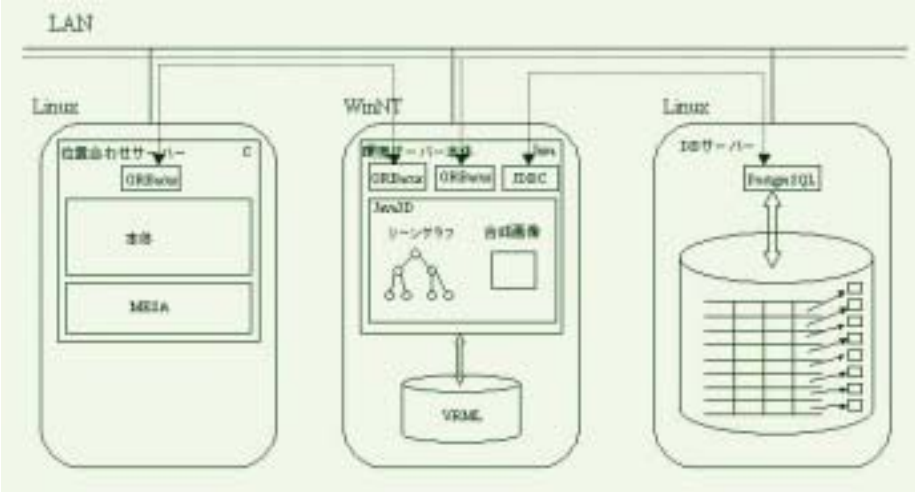

図2　環境サーバーのシステム構成

4　画像情報の貼り付け

ここでは、位置あわせのあと、入力画像から必要な部分を切り出し、仮想物体表面に貼りつける過程について説明する。

4-1 三角パッチへの分解

基本的には視野内に存在する仮想物体の輪郭に沿って画像を切り出し、物体表面に貼り付ければよい。ただし、表面の法線方向が観測方向と大きく異なる場合、貼り付けた情報の品質は悪くなる。したがって、同一対象にたいして多くの方向から観測した画像が入力されるような場合には、表面の法線方向により近い観測方向からの情報を採用してマッピングすることが重要である。

対象物体が平面で構成される場合は、平面ごとに採用すべき画像を決定すればよいが、曲面で構成される場合、同一曲面でも部位によって法線方向が大きく変わりうるので、それを単一表面として扱うことはできない。したがって、曲面は法線方向の変化が一定とみなせる程度におさまる小ささに分解する必要がある。ここでは、扱う曲面をパイプの円筒部分に限定し、一定の大きさの三角パッチに分解し、各三角パッチごとに画像情報をマッピングできるようにした。

ただし、すべてのパイプを小さな三角パッチに分解するとシーングラフが巨大になり、記憶に必要なメモリ容量が急増するとともに、レンダリングの時間も極端に増加する。そこで、妥協策として情報の蓄積が必要となる可能性のある部分だけ三角パッチに分解することとした。

4-2　必要な情報の選択

プラント点検の場合、巡回ロボットが自身で必要な情報を選択してそれを収集したり、オペレータが必要な情報を指示して提示を求める場合もあれば、他のクライアント、例えば劣化予測システムが必要な情報を指示し、それを巡回点検ロボットが収集したり、運転システムが必要な情報を指示して、それをオペレータに提示させる場合も想定される。このために、クライアントごとに異なる情報選択を可能にしたり、あるいは、その情報をクライアント間で共有できるメカニズムが必要である。注意制御のために環境サーバーに組み込んでいる能動指標[1]により、これを実現する手法を検討しているが、これについては別の機会に述べる。

必要な情報の選択は、現在は、簡単のためにパイプを単位として行っている。例えば、パイプ５とパイプ６の画像情報が必要であると選択されているとすると、2 本のパイプを構成するすべての三角パッチがテクスチャマッピングの対象となる。パイプの指定は、パイプ番号を陽に指定してもよいが、クライアントの視線をポインティングデバイスとして暗に指定することもできる。視野中心の感度が高いセンサ入力を蓄積したり、人間の目に提示する場合には便利な機能である。

4-3　マッピング

指定されたパイプを構成するすべての三角パッチについて、入力画像において可視か否かを法線方向チェック、隠蔽チェックにより判定する。可視と判定された各三角パッチに対して、3 頂点の入力画像への投影位置を計算し、その 3 点を含む矩形領域を画像から切り出し、Java3D の関数を用いて三角パッチにマッピングする。

図 3 はパイプ 39 について、1 枚の入力画像(a)から切り出した情報を(b)に貼り付けたところ(c)である。(d)はそれを違った角度から見たところである。

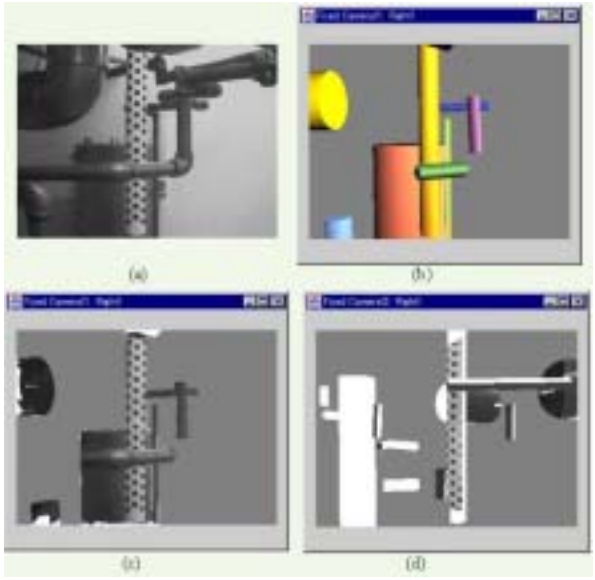

図 3　パイプへの画像情報のマッピング

5　テクスチャの管理

三角パッチに対して切り取られた画像は、三角パッチ ID、入力日付、データの種類をインデックスとして、マッピングに必要な情報とともに一端、ワーキング DB（WDB）に格納する。三角パッチ ID は Java3D によりパイプに割り振られた番号と、分割したときに Java3D が各三角パッチに割り振る番号から成る。後者の番号は分割の仕方を変えなければ不変であるが、前者はパイプなどを Java3D のロードする順に依存するので注意が必要である。データの種類はカラー画像、白黒画像、あるいは温度分布などである。

画像情報を WDB に格納する際、すでに同じ三角パッチ ID とデータ種類を持つレコードが存在した場合、それらは同一対象についての同種の情報であり、高い圧縮効果が見込める。特にプラントのような構造物の場合、表面反射特性が短時間に変化することはないので、1 回の巡回点検で得られた画像情報は本質的に同じとみなせる[1]。したがって、情報量が多い、つまり画像サイズが大きいほうのレコードを残し、小さいものは WDB から削除することとした。図 4 は、巡回点検時の入力画像を次々に WDB に蓄積したときの様子を示す。上段が入力画像であり、下段は WDB に格納されたテクスチャをマッピングして表示した結果である。異なる方向からの入力画像により、テクスチャを持つ三角パッチが増えていくことと、すでにマッピングしていた情報がより解像度の高い情報に置き換えられていくことがわかる。このようにして、観測方向が異なる入力画像についての情報圧縮を実現している。

[1] ただし、このためには先に述べた整合性をとる過程で、照明方向やカメラパラメータによる画像情報の変動を完璧に取り除けている必要がある。

こうして WDB に格納した情報は、情報入力の区切り（巡回点検の場合、一巡回の終了）において、DB に転送し永久保存する。図 5 は永久 DB 内の保存レコード列の一部である。PostgreSQL では、画像のような大きなオブジェクトはポインタで管理し、実際の画像データはレコード列とは別の場所に格納されている。例の中に、同じ三角 ID とデータ種類を持つレコードが存在するが、それらは異なるラウンドの巡回点検中に得られた情報である。先にも述べたように、プラントの見えの変化は非常に緩慢であるので、これらの画像情報についても大きな圧縮効果が期待できる。ただし、現状では時間軸方向への画像の圧縮は行っていない。

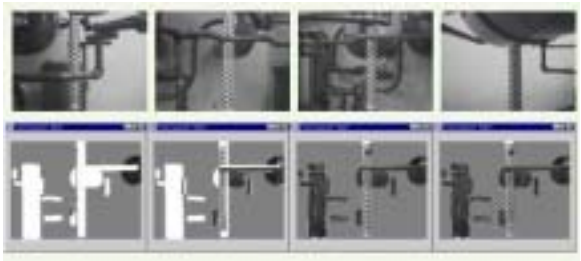

図 4　異なる観測方向の画像の圧縮

Triangle ID	Input Date	Property	Mapped data
00100100	010701	color	* data
00100100	010701	intensity	* data
00100100	010701	friction	* data
00100100	010715	color	* data
00100100	010715	intensity	* data
00100100	010715	friction	* data
00100100	010801	color	* data
00100100	010801	intensity	* data
00100100	010801	friction	* data
00100101	010701	color	* data
00100101	010701	intensity	* data
00100101	010701	friction	* data

図 5　DB 内の保存レコードの例

6　テクスチャの提示

6-1　提示要求の種類

クライアントからの情報提示要求には、今のところ、次の 2 種類を想定している。

1) パイプ 39 の 2001 年 8 月 17 日のカラー情報を提示しろ

2) パイプ４と５の、1995 年 1 月 1 日から 2001 年 12 月 31 日までの濃淡情報を提示しろ

1)は提示対象とデータ種類を特定した上で、ある時刻の情報の提示を求めるものであり、2)は時間軸方向に連続的な提示を要求している。図 6 は 1)に相当する要求を GUI 上でインタラクティブに指定し、サーバーからその情報提示を受け取ったところである。ここでは情報提示が必要なパイプを番号で指定したが、4-2 で述べたようにクライアントの視線を用いることにより、クライアントの視点位置変化に追従して情報表示するパイプを切り替えながら画像を提示するようなことが可能となる。

図 6　画像情報の提示例

6-2　情報提示処理の実装と実行時間

クライアントからの情報提示要求に対して環境サーバーは、必要なテクスチャ情報を DB から検索し、シーングラフにマッピングし、レンダリングするという処理を行う。時間軸方向に連続的な提示は、検索する情報の入力日付だけを変えながら、これら一連の処理を繰り返すことにより実現する。したがって、提示処理に要する時間が短ければ短いほど滑らかな情報提示が行える。

図 7 に現状の実装において提示処理に要した時間を示す。使用した計算機環境は表 1 のとおり。それぞれの処

理は以下のようである。

① シーングラフから古いテクスチャ画像をクリアする
② DB サーバーに情報提示に必要なレコードだけからなる WDB の作成を要求し結果を受け取る
③ WDB 上のポインタ情報により DB サーバーからテクスチャ情報を受け取る
④ 受け取ったテクスチャを三角パッチにマッピングする
⑤ レンダリングする

図でわかるように、高速化のために③と④の処理は 2 つのスレッドにわけて並行に実行している。上段の数字は表 1 に示した実行環境において、パイプ 1 本（三角パッチ数 459）だけを提示対象としたときの処理時間であり、下段はパイプ 18 本（総三角パッチ数 5194）を提示対象としたときの処理時間である。

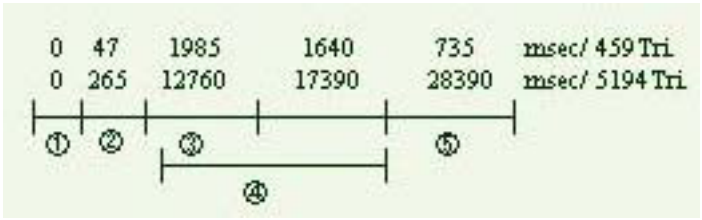

図 7　情報提示の実装と処理時間

表 1　計算機環境の主な仕様

	環境サーバーマシン	DBサーバーマシン
CPU	PentiumIII 700MHzx2	PentiumIII 1GHx2
RAM	512Mb	256Mb
VIDEO	Spectra8000 4xAGP	Millenium G200 AGP

パイプ 1 本だけを提示対象とした場合でさえ、環境サーバー上で提示画像を作成するために 5〜8 秒もかかっており、さらに、提示対象が増えると処理時間が急増することがわかる。処理時間が予想以上に大きい原因について、これまでに以下のような検討を行った。

a) ③の間、DB サーバーの CPU 負荷は数十パーセントである
b) ③④の処理の間ずっと、環境サーバーの CPU 負荷はほぼ 100 パーセントである
c) ④の処理においては DB サーバーからバイト型で受け取ったテクスチャ情報を実数型に変換するために時間がかかっているようである（Java の問題）。
d) ⑤では、テクスチャマッピングされた場合のレンダリングスピードの低下が問題であり、その原因としては 3D グラフィックカードのマッピング機能をうまく使えていない（WindowsNT,Java3D、OpenGL、グラフィックカードドライバのコンビネーションの問題）、あるいは、Java3D のシーングラフから OpenGL への変換に時間がかかっている（Java3D の問題）などが考えられる

上記のほかにも、DB サーバーの使い方、処理の並列化の工夫、ハードウェアの有効利用など検討の余地はまだまだ残っており、大幅な高速化をはかれると考えている。

7　まとめ

多種多様な点検情報を仮想環境に蓄積するアプローチを紹介し、画像情報について具体的なシステムを提案した。紹介したシステムの実装は開発の緒についたばかりであり、今後様々な改良を加えることにより現実的なものとしていく。そのために、VR 技術や DB 技術についての知識を深めるが、本稿を読んでのご助言をいただければ幸いである.

実装手法のほかにも多くの検討課題がある。三角パッチのサイズの最適化、幾何構造変化への対応、時間軸方向への情報圧縮、などなどである。

最後になったが、環境サーバーの構築に関して尽力いただいているカーネル株式会社の河村さんに感謝する。

参考文献

[1] 喜多伸之：反射的・意図的注視制御の統合的な実現、信学論 D-II, Vol.J84-D-II, No.8, 1701-1709(2001).
[2] 喜多泰代、喜多伸之：複雑環境中の遮蔽輪郭を利用した位置・姿勢検出, MIRU2000, II-43-48(2000).
[3] Y. Kita, N.Kita：On accuracy of 3D localization obtained by aligning 3D model with observed 2D occluding edges, to appear in ACCV2002(2002).