ヒューマンインタフェース 65－11
音声言語情報処理 10－11
（1996. 3. 1）

研究者向け動画通信システム開発環境VICCLの設計と実装

嶋田 総太郎† 山本 吉伸†† 安西 祐一郎†

† 慶應義塾大学 †† 電子技術総合研究所

E-mail: shimada@aa.cs.keio.ac.jp, yoshinov@etl.go.jp, anzai@aa.cs.keio.ac.jp

通信インフラストラクチャの発達に伴い、ヒューマンコミュニケーション研究やパターン認識研究などの各領域で動画通信システムを自分の研究に利用したいという要求が高まっている。本稿では、様々な分野の研究者が動画通信システムを容易に作成するための、動画通信システム開発環境VICCLの設計・実装について述べる。VICCLではプログラム内で映像を容易に扱えるように、「仮想カメラ」の概念を導入し、映像を物理的なカメラから分離する。これにより研究者は様々な種類の映像を、容易に動画通信システムに組み込むことができる。またVICCLは動画通信システムの作成を容易にするための通信機構を備えており、これによりプログラム作成労力を軽減できる。VICCLはC++のクラスライブラリとしてWindows NT/Windows 95上に実装されている。

VICCL: A Development Environment for Video Communication System

Soutaro Shimada† Yoshinobu Yamamoto†† Yuichiro Anzai†

† Keio University

†† Electrotechnical Laboratory

As the communication infrastructure make progress, there is an increasing demand to use video communication in various research areas, i.e. human-communication or pattern matching. This paper describes a development environment for a video communication system, VICCL, that allows various researchers to make various video communication system. We propose "virtual camera" that separate video image from real camera. Using virtual camera, researchers can implement video communication system easily. We also propose network communication mechanism for video communication that allows various connection between servers and clients. We implemented VICCL as a C++ class library on Windows NT / Windows 95.

1 はじめに

最近の通信インフラストラクチャおよび画像圧縮技術の発達により、動画通信を行うための通信基盤が整いつつある。それに伴い、ヒューマンインタフェース研究やパターン認識等の要素技術研究、あるいは心理学といった各領域で動画通信システムを自分の研究に利用したいという要求が高まっている。しかし、動画通信システムを専門としていない研究者が、それらの実装を行うことは容易ではない。既存の動画通信システム開発環境では研究者の望むような条件の設定、機能の改変等が困難なことが多く、商用の動画通信システムもそのような用途に使用することは難しい。

研究者の要求する動画通信システム例として以下のようなものが考えられる。

画像処理の分野では 複数の映像を合成して広範囲の連続的な映像を作り出し、クライアントに提供する動画通信システム等も考えられる。

また参加者の映像の表示を開始する際に、フェイドイン効果をかけるもの [J.C.Tang *et al.* 94] などもある。フェイドイン効果とはデスクトップ上でTV電話などを実現する際に、突然相手の顔が画面に現れたのではユーザに対する心理的圧迫感が大きいので、徐々に相手の顔が浮き上がってくるような画像処理を施すものである。

パターン認識の分野では 画像処理によるユーザ識別の研究を応用して、あらかじめ登録されている研究者のみが参加できる遠隔会議システムが考えられる。このような会議システムがあれば、参加者がコンピュータの前に座っただけで会議に参加できるものと思われる。

コミュニケーションメディアにおける心理学の研究では ユーザの見ている方向だけを動画表示し他を静止画表示する遠隔会議システムとすべての画面を動画表示する遠隔会議システムとの間のユーザに対する心理的影響を調べようという研究 [山本 他 95] もある。

ユーザインタフェースの研究では 参加者を映した動画の配置をシステムの実行中に動的に切り替えたいという要求が出てくると考えられる。

本稿では、このように各分野で様々な要求が生じてくるものと考え、プログラマが柔軟に動画の提供形態を変えたり、ネットワークの接続形態を変えることができる枠組を提供することを目的とし、研究者の様々な要求を満たす動画通信システムを容易に作成できる動画通信システム開発環境 VICCL を設計・実装した。

なお本稿では以後、動画通信システムを作成する人をプログラマと呼び、会議に参加する人をユーザと呼ぶことにする。

2 VICCL

2.1 仮想カメラの導入

広域映像提供システムなどの複数のカメラを扱うシステムを作成する際、物理的な1つ1つのカメラをプログラマが意識しながらプログラムを書くのでは作成が非常に困難になる。このような場合、映像を物理的なカメラから切り離して扱えるとプログラムの記述が容易になると考える。そこで、VICCL では「仮想カメラ」の概念を導入する。

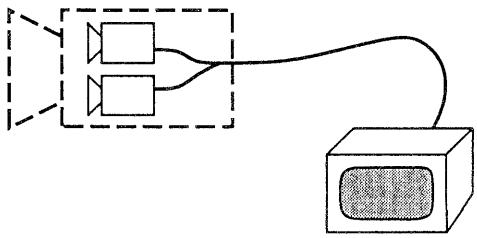

図 1: 仮想カメラ

仮想カメラはプログラムの記述を容易にするだけでなく、クライアントに対してその映像がどのように実装されているのかという詳細を隠すことができる。すなわち、広域映像を取得するカメラの台数が2台であっても10台であっても良いことを意味する。これにより、クライアントのプログラム作成労力を軽減し、またプログラムの保守性を高めることができる。

この仮想カメラは、複数のカメラを入力とする広域映像だけでなく、様々な種類の映像を扱う際にも適用できる。例えば画像処理を施す場合に、物理的なカメラと切り離して、画像処理を行った映像を映す仮想カ

メラを想定するとプログラムの記述が容易になる。またネットワークを介して映像を取得する場合にも、通信部分の処理などを考慮せずに遠隔地の映像を映す仮想カメラがあると考えるのが良い。もちろん、カメラ1つの入力を仮想カメラにすることもできる。このように仮想カメラの概念を用いることによって、あらゆる種類の映像を容易に扱うことができると考える。

2.2 インプリメント

動画通信システムを作成する際、どのような動画を提供するかという映像のコンテンツに関する部分とネットワークの接続形態に関する部分の2つの部分を考慮する必要がある。よって VICCL も映像部分と通信部分の2つに分けて設計および実装を行う。

2.2.1 映像部分

(1) 映像クラス

VICCL では、仮想カメラを映像クラスとして設計・実装する。映像クラスは映像データをカプセル化したものである。すなわち、映像クラスは基本的にビットマップを扱うクラスと同じであると考えることができる。ビットマップクラスと異なる点は、映像データがリアルタイムで更新されることである。

プログラマは様々な映像クラスを作成し、その映像クラスを操作することでプログラムを記述する。映像クラスを用いることによって、プログラムの可読性を高め、開発を容易に行うことができる。

(2) 基底映像クラス (CImage クラス)

映像クラスには前述したように様々なものが考えられるが、もしこれらの映像をそれぞれ独立に扱うとすると、プログラマには多大な負担がかかることになる。そこで、すべての映像を統一して扱えるようなインタフェースを用意することでプログラマの負担を軽減できると考える。本稿では、基底クラスとなる基底映像クラスを1つ定義し、そこから派生映像クラスを定義していくことにする (図 2)。映像に対する基本的な操作や各映像に共通のデータなどはすべてこの基底映像クラス内で定義する。

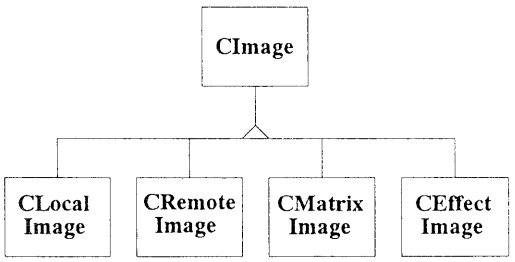

図 2: 映像クラスの派生関係

基底映像クラスのメソッドとして考えられるのは、映像データの更新を行うかどうかを指定する関数である。すなわち、更新開始 (StartSrv())、更新終了 (StopSrv())、更新中断 (SuspendSrv())、およびフレーム更新 (FrameSrv()) である。また、保持している映像データを外部に提供する GetFrame() 関数も必要である。GetFrame() はその瞬間に保持している映像データを指定されたサイズに切り出し、映像データと同じフォーマットのデータを返す。受け取った側は映像データを画面に表示したり、他の画像処理を行ったりできる。

後述する映像の階層構造や映像マネージャによる通信の自動化などの利点は、基底映像クラスを導入することによって得られる。

(3) 映像の階層構造

ネットワークを介して送られて来た映像に画像処理を施したい場合などに、遠隔地の映像を表す映像オブジェクトの出力を画像処理を行う映像オブジェクトの入力に繋げられると、さらにプログラムの記述が容易になる。またこうすることによって、プログラムの再利用性も高めることができる。そこで、VICCL ではこのように映像オブジェクトを繋げていくようにプログラムを書くことを可能にする。

複数の映像を合成して1つの映像にする合成映像クラスを作成する場合、この合成映像オブジェクトは複数の入力映像オブジェクトから構成されることになる。また、画像処理を行う映像オブジェクトは他の映像オブジェクトを入力とする。

このような場合、映像を階層構造に出来ると扱いが容易になる。例えば図 3のように複数のキャプチャ映像を入力とする合成映像クラスを作成する場合を考える。映像を階層構造にできない場合は、プログラマは合成映像クラスを作成する際に複数の映像をキャプチャ

するところから記述しなければならないが、映像を階層構造にできる場合にはすでに作成されているキャプチャ映像クラスをそのまま用いて、その後の処理だけを記述すれば良い。

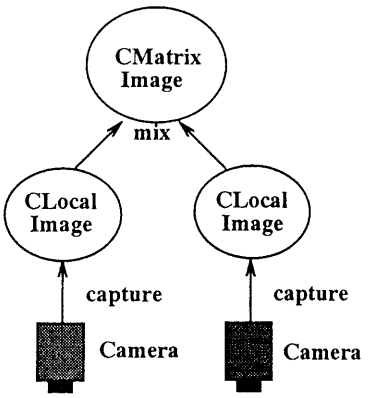

図 3: 映像の階層構造

ここで階層構造になった映像の、入力となる方を子映像、入力映像を受け取って処理を行う方を親映像と呼ぶことにする。

映像の階層構造は基底クラスに用意された `GetFrame()` 関数を用いることで実現できる。すなわち、親映像は子映像の `GetFrame()` 関数を呼び出して得たデータを自分の映像データとして保持すれば良い。全ての映像クラスを基底映像クラスの派生クラスとして作成すれば、どのような映像に対しても `GetFrame()` を呼び出すことができる。これは、どのような組合せの階層構造も可能になることを意味する。従ってプログラマが映像を作成する際に、多大な柔軟性が与えられることになる。

親映像がデータを更新するためには、子映像のデータが更新されたことに気付かなければならない。そこで、子映像は映像データが更新される毎に、親映像にメッセージを送ることにする。親映像は子映像からのメッセージが届くたびに画像処理などを行って自分自身の映像データを更新し、さらに親映像がいればメッセージを送る。

2.2.2 通信部分

動画通信システムを作成するにあたり、プログラマはネットワークの接続形態を柔軟に選択できるべきであるが、その一方でプログラマが複雑な通信の処理を記述せずに済むような配慮が必要である。VICCL はこの相反する 2 つの要求を満たすための単純だが強力な通信機構を提供する。

(1) 遠隔映像クラス (CRemoteImage クラス)

ネットワークの接続形態の選択に柔軟性を持たせることは、遠隔映像クラスを導入することで実現可能となる。遠隔映像クラスは、映像の階層構造をネットワークに拡張したものである。すなわち遠隔映像クラスは遠隔地の映像のデータを入力とし、それと同じデータを保持する映像クラスである。遠隔映像クラスも基底映像クラスの派生クラスであるので、他の映像クラスと同じように階層構造に組み入れることができる。これにより、動画の提供形態に多大な柔軟性をもたらすと同時に、ネットワークの接続形態にも十分な選択の余地を与えることができる。

通信のオン／オフは映像クラスの更新開始 (StartSrv()) や更新中断 (SuspendSrv()) のメソッドを呼び出すことで実現できる。これは、キャプチャ映像クラスがキャプチャのオン／オフを行うのと同じように遠隔映像クラスが通信のオン／オフを行えるということである。従って、プログラマはネットワークをほとんど意識せずに細かな通信制御を記述することができる。

(2) 映像マネージャクラス (CImageManager クラス)

サーバはクライアントとの動的な接続受付と映像の管理・通信という仕事をする。映像マネージャクラスはこういったサーバの仕事を行うためのクラスである。また複数のクライアントとの接続を可能にするために、ストリームクラス CStream を導入する。ストリームクラスは遠隔映像との実際の通信を行う。VICCL の通信機構は以下のとおりである。

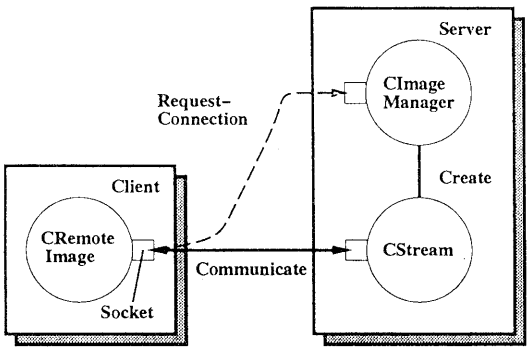

図 4: 通信機構

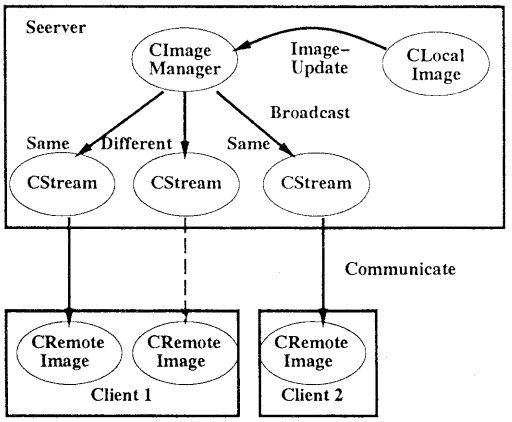

図 5: 自動ストリーム送受信機構

映像マネージャは遠隔映像との接続を行うためのソケットを 1 つ持ち、接続要求があるとストリームオブジェクトを生成する。ストリームオブジェクトが遠隔映像との実際の通信を行い、映像マネージャは次の遠隔映像からの接続要求を待つ (図 4)。映像マネージャが複数のストリームオブジェクトを生成することで、複数の遠隔映像との接続を実現できる。この操作は自動的に行われるので、プログラマが意識する必要はない。

(3) ストリーム送受信

映像のストリーム送受信は、映像データが更新される度に自動的にフレーム送受信する機構を提供することによって実現できる。そのためにはクライアントに対して提供する映像がデータを更新したことを映像マネージャに知らせることが必要になる。これは映像の入れ子構造を拡張して、映像マネージャがその映像の親映像であるとし、映像更新通知メッセージを受け取るようにすれば良い。

VICCL のストリーム送受信機構をまとめると次のようになる。映像マネージャは複数の映像クラスを管理しており、各映像クラスは映像データが更新されるたびに映像マネージャに対してメッセージを送る。映像マネージャは更新された映像の番号を全てのストリームオブジェクトに対してブロードキャストする。ストリームオブジェクトは送られてきた番号と相手の遠隔映像が要求している映像の番号と比較し、同じならその映像データを送信し、異なれば何もしない。この機構によって、自動ストリーム送受信が可能になる (図 5)。

3 サンプルプログラム

3.1 サーバのプログラム例

これは複数のカメラ入力を合成し、広範囲の映像としてクライアントに提供するサーバのプログラム例である。

まず、キャプチャ映像である CLocalImage オブジェクトとキャプチャの実行主体である CCapture オブジェクト、合成映像を表す CMatrixImage オブジェクトを作成する。次にキャプチャを開始し、キャプチャ映像を合成映像への入力映像として登録する。このとき、設計指針で挙げたケーブルで繋げるようなプログラミングが CMatrixImage クラスのメソッド Register() で実現されていることに注意して欲しい。登録が済んだら、合成映像の更新を開始する。その後の処理はプログラム中のメッセージハンドラのように、CMatrixImage クラスに対してだけ行えば良い。

```
//初期化ルーチン
    CCapture cap;        // キャプチャ実行クラス
    CLocalImage local1(0, &cap);
    CLocalImage local2(1, &cap);
    CMatrixImage matrix;     // 合成映像クラス
    cap.InitSrv();       // キャプチャ開始
    matrix.InitImage();
    matrix.Register(local1);// 合成映像に登録
    matrix.Register(local2);
    matrix.StartSrv();  // 映像更新開始
```

```
// メッセージハンドラ
    LPIMAGE lpi;     // 映像データ
    CPoint p;
    CSize s;
    GetParam(&p,&s);
    lpi = matrix.GetFrame(p,s);
    SendImage(client, lpi); // 送信
```

3.2 クライアントのプログラム例

ネットワークの負荷を考慮してユーザの見ていない方向の通信を停止して、見ている方向だけを動画表示する遠隔会議システムのクライアントのプログラム例を示す。

まず、プログラマは必要な映像をとってくるためにCRemoteImageオブジェクトを生成する。次に取ってくる映像のパラメータを設定する。その後、ユーザの見ている方向を取得する関数を呼び出し、その値が異なっていた場合に映像のストリーム送受信をオンにしたりオフにしたりしている。ネットワークをあまり意識せずに映像を扱えていることと、その一方で柔軟なネットワークの操作の両方を実現していることに注意して欲しい。

```
\\初期化ルーチン
    CRemoteImage r1(ipAdress1, port1);
    CRemoteImage r2(ipAdress2, port2);
    CPoint p(0, 0);
    CSize  s(320, 240);
    r1.InitImage(this);
    r2.InitImage(this);
    r1.SetCmd(1, p, s);//サーバ側の1番の映像
    r2.SetCmd(2, p, s);//サーバ側の2番の映像
\\メッセージハンドラ
    static pos, prev;
    pos = getDirection();    //見ている方向
    if(pos != prev){
        switch(pos){
        case SITE_A:
            r1.StartSrv();
            r2.SuspendSrv();
            break;
        case SITE_B:
            r1.SuspendSrv();
            r2.StartSrv();
            break;
        }
    }
    prev = pos;
```

4 今後の予定

今後 VICCL は、Java の拡張言語である HORB(a distributed and pararell Java) [Hirano 95] に移植される予定である。これにより、作成した動画通信システムを Netscape などの WWW(World Wide Web) ブラウザで動作させることができる。

5 結論

本稿では様々な形態の動画通信システムを容易に作成するための動画通信システム開発環境 VICCL を設計・実装した。VICCL では「仮想カメラ」の概念を導入し、

- 映像クラス
- 映像マネージャクラス

などを定義した。プログラマは VICCL を用いることによって、動画通信システムを作成する際の労力を軽減することができる。

謝辞

本稿を進めるにあたり、様々な面で支援して下さった安西研の皆様に感謝致します。

参考文献

[Hirano 95] Satoshi Hirano. ETL HORB a distributed and parallel Java User's Guide. In *http://ring.etl.go.jp/openlab/horb/doc/guide/guide.htm*, 1995.

[J.C.Tang *et al.* 94] J.C.Tang, and M.Rua. Montage: Providing Teleproximity for Distributed Gourps. In *CHI '94*, pp. 37–43, 1994.

[J. ランボー 他 92] J. ランボー, M. ブラハ, W. プレメラニ, F. エディ, W. ローレンセン. オブジェクト指向方法論 OMT モデル化と設計. 株式会社トッパン, 1992.

[山本 他 95] 山本吉伸, 松井孝雄, 梅田聡, 幸島明男, 開一夫, 仁木和久. ネットワークコミュニケーション実験環境の構築. 日本認知科学会第 12 回大会, pp. 146–147, 1995.