マルチメディア通信と分散処理 91－3
（1999. 1. 21）

ネットワーク透過型仮想ビデオキャプチャデバイスの設計

岡村 耕二*　　村尾 真洋**　　有川 正俊**
*九州大学
情報処理教育センター
**広島市立大学
情報科学部知能情報システム工学科

本稿で提案するネットワーク透過型仮想ビデオキャプチャデバイスは、遠隔にあるカメラや周辺機器があたかもローカルに接続されているようなAPIをユーティリティプログラムに提供することができる。本稿ではこのビデオキャプチャデバイスの目的、概要を説明し、その設計を示す。さらにプロトタイプを用いた応用を通じてその有用性を示す。

Design of Network Transparentable Virtual Video Capture Device

Koji OKAMURA*　　Michihiro MURAO**　　Masatoshi ARIKAWA**
*Educational Center for Information Processing, Kyushu University
** Department of Intelligent Systems Faculty of Information Sciences, Hiroshima City University

The Network Transparentable Virtual Video Capture Device introduced in this paper, offers the APIs of the remote peripheral equipments to the utility programs as if the remote equipments are connected locally. In this paper, the goal and outline of the Video Capture Device are explained and the design is shown. And the advantage is shown through the application with the prototype.

1 はじめに

近年では、ハードウェアの廉価化によって、マルチメディアデータをパーソナルコンピュータに手軽に取り込み処理をする環境が身近になってきた。そのようなマルチメディアデータをネットワークを経由させる処理の応用は、多くの可能性を秘めている。しかしながら、マルチメディア入力機器が接続されている計算機上で表示する処理に比べ、ネットワーク上を送信させる処理は複雑である。そのため、計算機上でローカルに処理されるユーティリティプログラムの豊富さに対してネットワーク対応のプログラムはまだまだ少ない。また、現在ローカルに動作するユーティリティプログラムでネットワーク経由で動作させることを期待されているものは少なくない。そこで、本研究ではこのようなローカルに動作するユーティリティプログラムをネットワークで容易に利用するための機能を、オペレーティングシステムに提供させることを目標とする。本稿ではその機能を果たすネットワーク透過型仮想ビデオキャプチャデバイスについて説明し、その設計と応用を示す。

2 ネットワーク透過型仮想ビデオキャプチャデバイス

本研究で開発しているネットワーク透過型仮想ビデオキャプチャデバイスについてその必要性と概要を説明する。

遠隔にある計算機のビデオキャプチャデバイスで入力した動画像を利用するためには、利用するユーティリティプログラム中で動画像をネットワークで送信す

るための符号化/復号化処理、送受信処理を行なう必要がある。そのため、ネットワークを意識せずに開発された動画を扱うためのソフトウェアをネットワーク対応にすることは非常に手間のかかることである。

また、最近ではビデオカメラのついている雲台と呼ばれる製品が普及している。これは、RS232C 信号でビデオカメラの向きやズームを制御できるものである。通常のオペレーティングシステムでは、ビデオキャプチャデバイスは、ビデオキャプチャデバイスドライバによってあるキャラクタデバイスに抽象化され、RS232C インタフェースはシリアルデバイスドライバによってビデオキャプチャデバイスとは別のキャラクタデバイスとして抽象化されている。そのため、雲台から入力した動画と位置情報を連係して動作させるユーティリティプログラムは、ビデオキャオブチャデバイスおよびシリアルデバイスをそれぞれ制御しまた、それぞれのデバイスから入力したデータをプログラム中で同期させて処理を行なう必要がある。

また、通常、ビデオキャプチャデバイスは、他のデバイスと同様にあるプロセスがそのデバイスをオープンすると、そのプロセスがクローズされるまで他のプロセスはビデオキャプチャデバイスを利用できない。これは、ビデオキャプチャデバイスのフレームバッファが単一であるため、ビデオキャプチャデバイスドライバ内でのその資源の相互排除処理をする必要があるためである。一方で、例えばファイルの読み出しは複数のプロセスが同時に行うことが可能であり、それによって多くの利点がある。そのため、複数のプロセスがファイルからデータを読み出すように、ビデオキャプチャデバイスから複数のプロセスが同時に動画データを読み出ることが可能になることが期待されている。

以上の問題や要件を解決するために、本研究ではネットワーク透過型仮想ビデオキャプチャデバイスを開発している。その機能を以下に挙げる。

- 遠隔ビデオキャプチャデバイスとローカルなビデオキャプチャデバイスの統一的な利用
- ビデオ装置と関連のある装置との統合
- ビデオキャプチャデバイスのプロセス間での共有

それぞれの機能についてその概要を説明する。なお、詳細な説明は次章で行なう。

2.1 遠隔ビデオキャプチャデバイスとローカルなビデオキャプチャデバイスの統一的な利用

ネットワーク透過型仮想ビデオキャプチャデバイスによって、遠隔にある計算機のビデオキャプチャデバイスとローカルなビデオキャプチャデバイスを透過的に利用することが可能となる。デバイスドライバによってローカルなビデオキャプチャデバイスは、特別なデバイスファイルをインタフェースとしてプロセスからアクセスすることが可能となる。それと同様に、遠隔にある計算機のビデオキャプチャデバイスをそのようなデバイスファイルをインタフェースとしてアクセスを可能とする。これによって、プロセスからは、ローカルにあるビデオキャプチャデバイスも遠隔にある計算機のビデオキャプチャデバイスも、open()、read()、ioctl() といったシステムコールで画像データの読み出し、制御を行なうことが可能となる。

2.2 ビデオ装置と関連のある装置との統合

ビデオキャプチャデバイスと、雲台のインタフェースであるデバイスを統合する。つまり、ある一つのデバイスファイルをオープンし、そのデバイスから read() システムコールなどで画像データの読み出したり、雲台を制御するためにパラメータを ioctl() システムコールで設定したりすることを可能とする。また、読み出した動画像に雲台の情報を付加することによって、ユーティリティプログラム内での動画データと雲台の情報を同期させる処理は不必要となる。

2.3 ビデオキャプチャデバイスのプロセス間での共有

本研究で開発する仮想ビデオキャプチャデバイスドライバでは、画像データの読み出しのためのシステムコール呼び出し時にハードウェア的なフレームバッファを直接アクセスしない。それによって、複数のプロセスが同時にビデオキャプチャデバイスから動画像データを読み出すことが可能となり、プロセス間でビデオキャプチャデバイスを共有することが可能となる。

3 ネットワーク透過型仮想ビデオキャプチャデバイスの機能

前章でその概要を説明した本研究で開発しているネットワーク透過型仮想ビデオキャプチャデバイスの機能についてそれぞれ詳細に説明する。

3.1 ビデオキャプチャデバイスのネットワーク透過機能

ネットワーク透過型仮想ビデオキャプチャデバイスは、ビデオキャプチャデバイスを利用者にネットワーク透過的に提供する機能を持つ。この機能によって、ある計算機から遠隔の計算機のビデオキャプチャデバイ

スを、ローカルなビデオキャプチャデバイスと透過的に扱うことが可能となる。

遠隔の計算機のビデオキャプチャデバイスとローカルなビデオキャプチャデバイスを透過的にするということは、ユーティリティプログラムからは遠隔の計算機のビデオキャプチャデバイスの特別デバイスファイルが、ローカルなマシンにマウントされているように見えることである。そのため、ユーティリティプログラムからは物理なビデオキャプチャデバイスの地理的な情報を考慮することなくそのデバイスに対してシステムコールを発行し、動画の読み出しや、パラメータの変更をすることが可能となる。

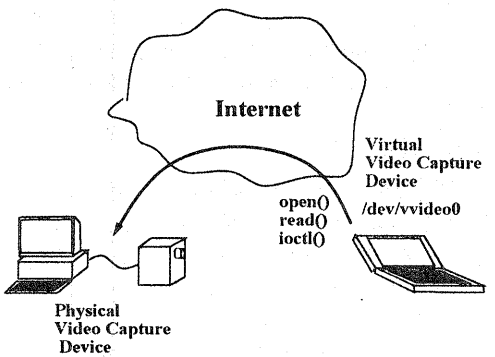

図 1: ネットワーク透過機能

3.2 ビデオ装置と関連のある装置との統合

キャプチャする画像の解像度の変更といったビデオキャプチャデバイスの制御は、ビデオデバイスに対して ioctl() システムコールを呼び出すことで行なわれる。仮想ビデオキャプチャデバイスでは、ビデオキャプチャデバイスだけではなく、雲台の制御もビデオキャプチャデバイスに対する ioctl() システムコールで行なうことができる。

RS232C インタフェースで接続されている物理的な雲台の制御はシリアルデバイスに対してそれぞれの雲台に特有な命令を流し込むことで行なわれるため、雲台を制御するユーティリティプログラムは非常に複雑になる。しかし、仮想ビデオキャプチャデバイスを利用すると、ioctl() を呼び出すことだけで雲台の制御が行なえるようになる。

3.3 ビデオキャプチャデバイスのプロセス間での共有機能

仮想ビデオキャプチャデバイスでは、物理的なフレームバッファを直接制御しない。これによって、プロセス間の相互排除の制約が緩やかになる。通常のビデオキャプチャデバイスではあるプロセスが一旦デバイスをオープンすると、そのプロセスによってクローズされるまでは他のプロセスは利用できない。それに対して仮想ビデオキャプチャデバイスでは複数のプロセスが同時にデバイスをオープンすることが可能である。相互排除の単位はシステムコール呼び出しの粒度となる。つまり、あるプロセスが仮想ビデオキャプチャデバイスに対してシステムコールを発行してそのシステムコールが完了するまでは他のプロセスはシステムコールを発行することはできない。

複数のプロセスが仮想ビデオキャプチャデバイスを同時にオープンすることができると、任意のプロセスによってランダムにデバイスの状態、例えば動画の解像度や雲台のパン角の値が変更されることになるので、ユーティリティプログラムの作りが悪いとそのプログラムに悪影響が及ぼされる可能性がある。しかし、その問題はユーティリティプログラムの作りで解決できる問題であり、それに比べると複数のユーティリティプログラムが同時に画像を読み出せる利点の方がはるかに大きい。

4 vvideo device の設計

本研究で開発しているネットワーク透過型仮想ビデオキャプチャデバイスは vvideo (Virtual Video Capture) device と呼ぶ。本章では vvideo device の設計について説明する。

4.1 モジュール構成

vvideo device は、図 2に示されるように、クライアントデバイスドライバ、クライアントデーモン、サーバデーモンの 3 つのモジュールで構成される。それぞれのモジュールについて以下に説明する。

クライアントデバイスドライバ

クライアントデバイスドライバは、ユーティリティプログラムが発行するシステムコールに対する処理を行なう。vvideo device に用意されているシステムコールは、画像データ、画像の情報、雲台の情報の読み出しおよび書き込みである。

クライアントデバイスドライバは、ドライバ内に画像データを格納するフレームバッファのメモリ

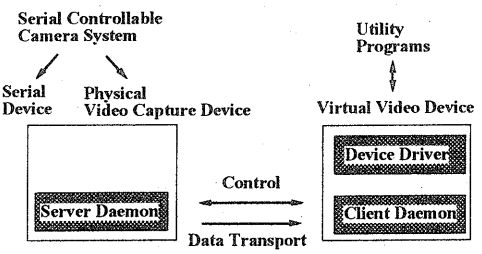

図 2: モジュール構成図

領域を管理している。また、マイナー番号ごとに独立したフレームバッファを管理している。

クライアントデーモン

クライアントデーモンは、後述するサーバデーモンから画像データを受信し、クライアントデバイス内のフレームバッファを更新する。

また、ユーティリティプログラムによって vvideo device の雲台の情報が更新されるとサーバデーモンに物理的な雲台の状態の変更を要求する。

サーバデーモンからの動画の送信は、ユーティリティプログラムによって vvideo device から動画の読み出しが行なわれる時のみ行なわれている。つまり、vvideo device を使用しているユーティリティプログラムが一つもなければサーバデーモンからの動画の送信は行なわれない。この送信の制御はクライアントデーモンによって行なわれる。

クライアントデーモンは一旦起動されると、終了するまで vvideo device を常にオープンしたままであるため、vvideo device をオープンしているプログラムが 2 つ以上になると、クライアントデーモンはサーバデーモンに動画の送信を要求する。

サーバデーモン

サーバデーモンはクライアントデーモンからの雲台の制御、動画の送信制御の要求に従って物理的なビデオキャプチャデバイスおよび雲台の制御、クライアントデーモンへの動画の送信、送信の停止を行なう。

4.2 API

クライアントデバイスドライバは、ユーティリティプログラムへ vvideo device への API を提供する。API は、ビデオキャプチャ用のもの、雲台用のものが用意されている。

ビデオキャプチャ用 API

ビデオキャプチャ用の API は、独自に新たなものを設計せずに、現在ローカルなビデオキャプチャ用に使われているものを使う。これによって、ローカルなビデオキャプチャ用に従来開発されてきた有益なユーティリティプログラムの再利用を可能とする。

雲台用 API

雲台用の API は、パン角、チルド角、ズームの値の読み出し、更新をする。雲台の種類によって、扱うことができる情報の種類や、それぞれの値の範囲も異なるので、雲台用の API は、次に説明するように拡張性に注意して設計されている。

vvideo device では雲台の種類ごとに ID を割り当て、その ID ごとに制御できる雲台の情報を定義したデータベースを持っている。これによって、各雲台はその種類とは独立して情報の定義が可能となるし、新たな種類の雲台の追加も容易である。

4.3 通信プロトコル

クライアントデーモンとサーバデーモンの間では、サーバデーモンからクライアントデーモンへ動画の送信および、クライアントデーモンからサーバデーモンへビデオキャプチャや雲台の制御のための通信が行なわれている。

クライアントデーモンとサーバデーモンの間で用いられてる通信プロトコルについて説明する。

動画通信用プロトコル

vvideo device では、動画像通信プロトコルとして RTP [1] を用いている。ペイロードタイプつまり動画の符号化形式は、クライアントデーモン、サーバデーモンが動作している計算機の計算能力および間の通信路の品質に合わせて適切なものを選択する。

クライアントデーモンは、サーバデーモンから動画を受信するとそれを RGB の形式に復号化してクライアントデバイスドライバに渡すので、クライアントデーモンとサーバデーモンの間で用いられている動画の符号化形式は特に問題にならない。

また、RTP のトランスポートアドレスとしてはユニキャストアドレスだけではなく、マルチキャストアドレスも用いることができるので、一つのサーバデーモンに対して複数のクライアントデーモンの構成をとることが可能である。

遠隔制御用プロトコル

現在、インターネット上には雲台のような機器を遠隔に制御するための決定的な遠隔制御プロトコ

ルは存在しない。遠隔制御を指向しているプロトコルとしては、RTSP [2] を挙げることができる。しかし、RTSP は連続メディアの制御のみを行なう機能を有しており、雲台といった周辺機器を制御する機能はない。また。RTSP は HTTP を拡張したもので、仮に雲台を制御する機能を拡張しても本研究の用途に対しては仕様が冗長的に膨大である。そのため、RTSP に新たな機能を拡張し、実装するのは適切ではない。

そこで、vvideo device の遠隔制御では、単純で汎用性のある遠隔制御プロトコル (SRCP: Simple Remote Control Protocol) を用いている。SRCP は、アスキーベースのプロトコルで、あるキーに対して文字列で記述された情報をセットする機能とあるキーから情報を読み出す機能を持っている。また、一つのコマンドで複数のキーに対するアクセスを可能とする。

vvideo device の 雲台 API では、雲台の種類ごとに ID を割り当て、その ID に対して情報を定義しているデータベースを持っていることを説明した。SRCP は、これを用いる。つまり、まず、ID を読み出し、遠隔の雲台の種類を調べる。次に、雲台データベースに従って、その雲台で利用可能なキーに必要な情報をセットすることによって遠隔制御を行なう。

図 3 に vvideo device の拡張性を図示する。図に示されるように、データベースを参照しながら、雲台の種類ごとにそれぞれ適した制御の通信、アプリケーションへの API をすることが可能となる。

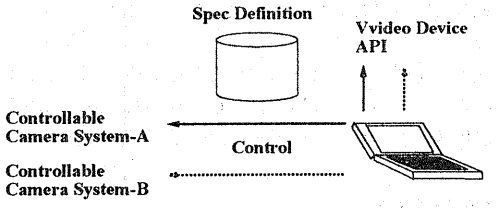

図 3: 拡張性の実現

5 vvideo device の応用と課題

現在、vvideo device の遠隔制御プロトコルおよびデバイス API を任意に決めてプロトタイプを試作し、その有用性に対する考察を行なっている。本章では、vvideo device を用いた応用を示し、vvideo device の課題について述べる。

5.1 応用

アプリケーション **Name-it** は、ビデオキャプチャデバイスのネットワーク透過機能を利用して、遠隔の計算機のビデオ装置が映す映像に文字を張り付け、簡単な拡張現実 (*Augmented reality*) を生成するものである。Name-it を実行して様子を図 4に示す。

ユーザは映像中のオブジェクトに対して注釈として文字を書き込むことができる。またオブジェクトはクリッカブルで、関連するウェブページを開く。これらの情報は複数のユーザで共有することができ、他のユーザが書き込んだ情報を読んだり、それに追加したりすることも可能である。

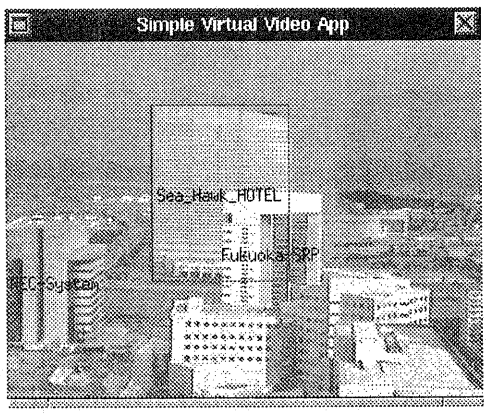

図 4: ビデオ映像と注釈

次に Name-it の構成について述べる。基本的な構成図を図 5 に示す。

ユーザは GUI のスクロールバーを使ってビデオカメラを操作する。GUI の内部で、遠隔にあるカメラの動画像ならびに雲台情報の取得は read()、雲台の制御は ioctl() といった vvideo device のシステムコールを呼び出すだけで必要な処理は全て完了する。オブジェクトに対する注釈が空間データベースに存在すればビデオ映像に文字が表示される。

ビデオ映像に映るオブジェクトに対してユーザは自由に注釈を書き込むことができる。ビデオ映像の映るウインドウでマウスによりオブジェクトを矩形で囲む。GUI に Name,URL, Info といった属性を入力し、"Name-it" のボタンを押すと先のオブジェクトに Name が付く仕組みである。これらの情報は空間データベースに追加され、他のユーザもこのデータを利用することができる。またこれだけではビデオカメラとオブジェクト間の距離がわからないので、ユーザが雲台のパン・チルト角を変化させ、オブジェクトの位置が変わったところでもう一度矩形で囲む作業を行う。ビデオカメ

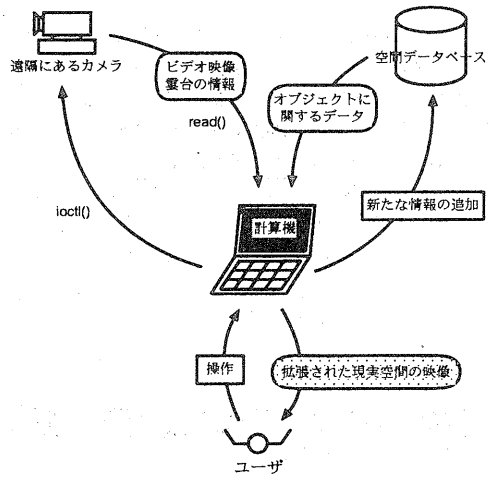

図 5: 構成図

ラからの距離を知ることによりビデオカメラのパン・チルト角の変化に対応し、オブジェクトの近くに文字を配置することが可能となる。さらにユーザが書き込んだ注釈の文字をマウスでクリックすることによりウェブのハイパーリンクのように、指定された URL を開く機能もある。

本来、Name-it のようなインターネットを跨いで動作するユーティリティプログラムの開発は複雑で非常に困難である。しかし、vvideo device を活用することによってその問題は克服されている。図 5 で、遠隔にあるカメラは計算機からはローカルに接続されているように扱うことができるからである。

次に、Name-it のように遠隔に設置されたカメラを使用する場合、カメラに映し出される対象のプライバシの問題が発生することがある。この問題は、vvideo deice のレベルで解決することができる。vvideo device のフレームバッファは、クライアントデーモンによって周期的に更新されている。この処理はソフトウェアで行なわれているため、画像データをフィルタリングする処理は容易に行なうことができる。例えば、雲台で角度を変更しながら撮影している時にプライバシの問題で部分的に撮影を避ける必要がある場合がある。このような場合、雲台の情報から判断して動画に局所的にマスクをかける処理はクライアントデーモンで行なうことができるので、ユーティリティプログラムは一切変更せずに対応することが可能である。

5.2 課題

vvideo device を実現する上で、いくつかの課題が残されている。まずは、遠隔制御プロトコル (SRCP) および、デバイス API の仕様を早急に設計する必要がある。遠隔制御プロトコルを設計する時、クライアントとサーバは完全にステートレスに動作できるようにする必要がある。つまり、クライアントデーモンとサーバデーモンはそれぞれ独立に再起動や停止をしてもシステム全体で矛盾が生じないための仕組みを遠隔制御プロトコルに入れる。また、遠隔制御するクライアントをサーバ側で限定できように認証技術を取り込む予定である。さらに、ネットワーク的に直接接続されていないファイアウォールを越えた利用を可能にするアーキテクチャも今後考察してゆく予定である。

6 おわりに

雲台をネットワーク経由で利用する研究は、LBL (Lawrence Berkeley National Laboratory) でも行なわれている [3]。LBL では雲台を制御するための仕様として、 *Camera Remote Control Command Language* を定義している。 *Camera Remote Control Command Language* は、パン、チルト角およびズームの値をはじめ、サポートしている雲台で設定できるプロパティを全て包含している。そのため、仕様が膨大になる反面、拡張性がないという問題がある。これに対して、vvideo device では、雲台の種類ごとにプロパティの定義を行なうので、拡張性に富んでいる。さらに、雲台に限らず TV チューナーなどの機器の制御の機能を拡張することも容易に行なうことができる。

現在、試作したプロトタイプを用いた応用の開発を通じた定性的な評価では、雲台やビデオ入力、ネットワークプログラミングをスクラッチから行なう場合に比べ、格段に容易にプログラムが作成できることが確認されている。今後は先に示した課題を考慮しながら具体的にシステムの実現を行なう予定である。

参考文献

[1] H. Schulzrinne,"RTP: A Transport Protocol for Real-Time Application",RFC 1889, (1996).

[2] H. Schulzrinne, Anup Rao and Rob Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326 (1998).

[3] http://www-itg.lbl.gov/mbone/devserv/