卒業論文

インターネットを通した
物理的インタラクションを有する
ゲームシステムの研究

40558 **段原　尚輝**

指導教員　石川　正俊教授
並木　明夫講師

2005 年 2 月

東京大学工学部計数工学科システム情報工学コース

概要

近年、遠隔地の人とインターネットを通して交流する為のロボットなどが開発されているが、物理的インタラクションをもって遠隔地の人と交流するための機構は殆ど開発されていない。そこで本研究では、物理的インタラクションを持って遠隔地の人と交流するため、ゲームシステム、特にエアホッケーの研究を行った。遠隔地から物理的インタラクションをもってエアホッケーを行う為の情報を得る手段として実時間性に優れたビジョンチップを用いることで様々なゲームシステムに応用できると考えている。

目次

第 1 章

序論

1.1 はじめに

近年ＨＣＩ（Human-Computer Intaraction）に関する研究が行われている。ＨＣＩとは人間とコンピュータ、あるいは人間と機械の接点における相互関係、対話型操作などのインタラクションに関する研究領域のことである。コンピュータのソフトウェアやハードウェアのインターフェイス、デザインの問題だけでなく使い方の学習方法や利用環境、利用者としての人間の特性などに関する考察も含むコンピュータサイエンス、工業デザイン、社会学、人類学、認知工学、認知科学、認知心理学、対話理論人工知能、彩理論、言語学など広範囲にまたがる学際的分野であり、近年あらゆる分野で使われるコンピュータを有効かつ安全に活用する為の技術を実現するのに欠かすことの出来ないものである。

近年ではＨＣＩを遠隔地の人との交流に活用しようと言う研究も行われている。遠隔地の人との交流手段としてインターネットを通して、文字や声を届けるだけでなく、映像を通して情報を交換する（例　テレビ電話）手段も近年は用いられている。現在では色々なところでインターネットを使えるようになり、ＨＣＩを用いて遠隔地の人と交流することが出来るようになったこと、更に交流手段を広げることで遠隔地にいる人の介護を行えたり、より多くの情報を相手に与え効率的な仕事を行えるようになることからもＨＣＩを用いて遠隔地にいる人と交流する研究が進められている。このように、ＨＣＩを応用して遠隔地にいる人と接する為のロボットなどは現在開発されているが、物理的インタラクションを持ったものは殆ど研究されていない。

そこで本研究では、遠隔地の人との交流手段を広げることの出来る、物理的インタラクションを有する交流手段のシステムに関する研究を行っている。物理的インタラクションとは、声や映像だけでなく物理的な力を遠隔地の相手に伝えることが出来ることを言う。テレマニュピュレーションなどがその例である。テレマニュピュレーションはインターネットを通してロボット制御を行うものである。現在は遠隔地にあるロボットアームを動かすなどに用いられている。この技術は、患者がいるのに医者のいない、もしくはその患者を治す技術を持った医者がいない状態の際にテレマニュピュレータを用いて遠隔地にいる医者が患者の容態をみる、などの応用を考えて研究が行われている。

1.2　関連研究

Florian Mueller らは遠隔地にいる人と交流するのにボールを用いている [1]。壁にプロジェクターを用いて遠隔地にいる相手側の映像を映し、さらにその壁のどこにボールがあたるかを判別し得点を得てゲームをするということを実現している。ところが、この手法は物理的な力を使ってゲームをする、ということは出来ているが自分の動きと相手の動きが殆ど独立している。

舘・川上研究室では、ロボットを人へのインターフェイスとして用いる RUI（Robotic User Interface）をコミュニケーションの手段として研究を行っている。現在 RobotPHONE を開発し、遠隔地の人と物理的にコミュニケーションをとる手法の研究を行っている [2, 3] RobotPhone とはロボットの持つ身体性に着目し、インターフェースに熊のぬいぐるみを使い声を相手に届けるだけでなく、ぬいぐるみの動きを遠隔地の人のぬいぐるみの動きに連動させ伝えるコミュニケーションデバイスである。離れた場所に存在するロボットの形や動き、位置などをリアルタイムに同期させることでロボットの動作をお互いに交換可能にしている新しいコミュニケーションデバイスである。

1.3　本研究の目的

本研究の目的は、遠隔地の人とコミュニケーションをとる手段として物理的インタラクションを有するシステムを構築することである。インターネットを介して物理的インタラクションを有するシステムを実現することが出来れば、遠隔地にいる人と体を動かして遊んだり、相手の行動に対して自分が思ったとおりの行動を行うことができるようになり、遠くにいる人への援助、身障者、老人などの介護、握手など触れ合いの交流などを行うように応用できる可能性を持っている。本研究では特にゲームシステムへの応用を考えた。特にゲームとしては比較的簡単で実現が可能なエアホッケーを対象とする。

エアホッケーのパックがどのように動くか [4]、エアホッケーをロボットと物理的にゲームをする研究は行われている [5, 6]。このときに用いられているのが「ビジョンチップ」である。２章でも述べているが、ビジョンチップが高い実時間性を持ち、対象を捉えるのに高い精度を持っているため、エアホッケーを行う上での時間のロス、パックの動きを追う上での誤差を最小限に抑えることが可能である為である。エアホッケーに限らず、ビジョンチップを用いることで、実時間性、精度の面でよりリアルに遠隔地にいる人にゲーム性を感じてもらうことが出来る。ゲームシステムを構築する上でビジョンチップを用いるのは有効な手段である。本研究ではこの点から、ゲームシステムの構築にビジョンチップを用いるのが重要であると考え、いかにビジョンチップを有効に活用するかに関しても研究している。

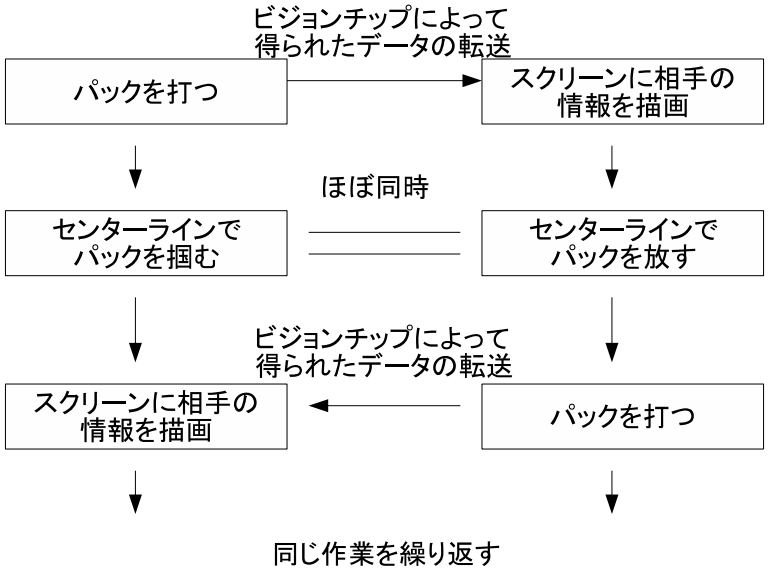

図 1.1. エアホッケーゲームのフローチャート

1.4 研究の概要

エアホッケーを遠隔地にいる人と行う為にはどのようなシステムでどのような装置を用意すればよいのかを以下に記す。まず、装置としてエアホッケー台、パック、ビジョンチップ、スライダ、パックを掴む為の機構、スクリーンを用意する。エアホッケーをやる相手のいる遠隔地でも同じ装置を用意する。

まず、自分のフィールドにパックがあるときは普通にエアホッケーを行う。その際の自分のフィールドの状態はビジョンチップによって情報を得る。その情報はインターネットを通して遠隔地にいる人に伝えられ、遠隔地にいる人はその情報を元に相手がどのように動いているかを写したスクリーンを見て待機する。パックがセンターラインに到達した時（つまり相手のフィールドにパックが行った時）センターラインにあるパックを掴む装置がパックを掴む。ほぼ同時に、そのパックの動きが相手のフィールドで連続しているように、相手のフィールドのセンターラインでパックを掴んでいた装置がパックを放す。パックをどの方向にどの速度で放すかはビジョンチップによる情報から判断する。そして、相手のほうにパックが行った後は今度はこちらにパックが戻ってくるまで相手の方のビジョンチップによって得られた情報を元にした画像を写したスクリーンを見て待機する。相手のパックがセンターラインを超えた時、先ほどセンターラインで掴まれたいたパックが相手のパックの動きと連続しているように速度を持って放される。どちらかのゴールにパックが入った時、逆のフィールドにいる方が一点追加となり再びゲームを再開する。おおまかなフローチャートを図 1.1 に示した。

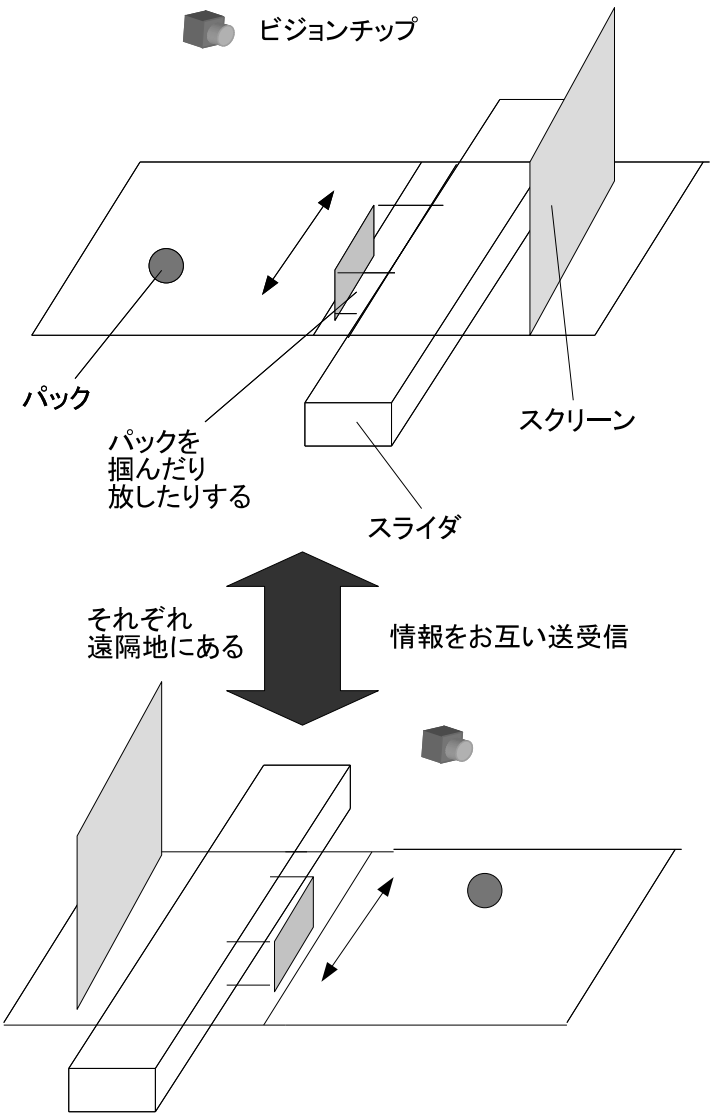

図 1.2. エアホッケーゲームの全体図

このような流れによってエアホッケーを行う。
エアホッケーを遠隔地にいる人とやる為のシステムは、図 1.2 のようになっている。
本研究では、主にこのシステムのソフトの部分の開発を目指した。

第 2 章

高速ビジョンを用いたエアホッケーシステム

2.1 ビジョンチップ

本研究では、移動する対象物体を識別するのに、ビジョンチップを用いている。

ビジョンチップとはセンサ（ＰＤ）と処理装置（ＰＥ）を同一チップ上に集積し、画素ごとの超並列処理系を構成するものである。センサと処理装置が一体化しているため、画像を走査後伝送する必要がなく、リアルタイムで処理を行うことが出来る。本研究では、特に小室らが開発したデジタルビジョンチップを用いている [7, 8]。デジタルビジョンチップは、 光センサ、画素レベルの並列度を持つデジタル回路の並列処理装置、特徴量抽出の為の出力回路をワンチップに集約することで、高速な視覚情報処理を実現している [9]。デジタルビジョンチップを用いることで、アナログ回路では実現できなかったモーメント抽出の為の専用回路により [10, 11]、μｓのオーダーで処理することが可能になり [12]、高い実時間性を得ることが出

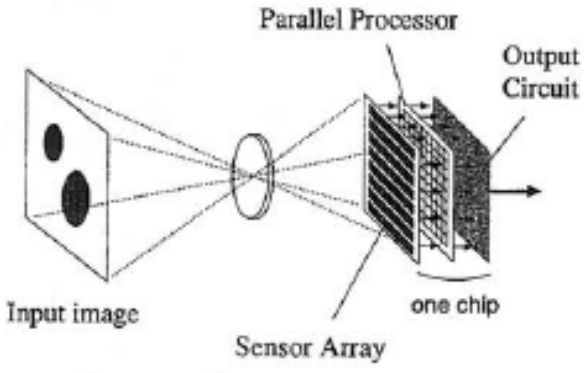

図 2.1. ビジョンチップのモデル [13]

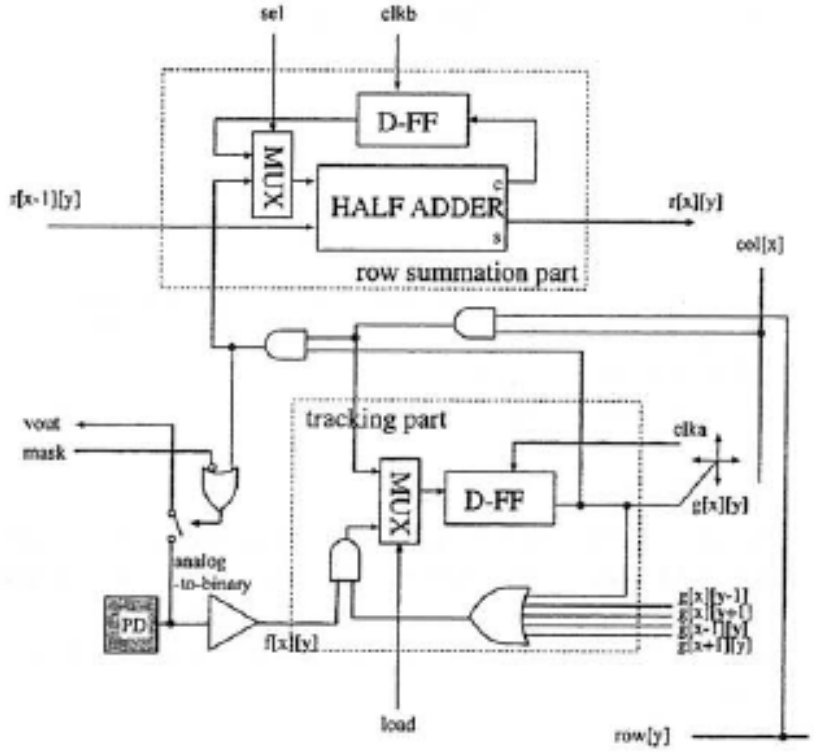

図 2.2. ＰＥの構成 [13]

来るようになっている。

本研究では、実時間性が重要な要素となる為に、高い実時間性をもつデジタルビジョンチップを用いるのが適当であると判断し、用いている。このデジタルビジョンチップのモデルを図 2.1 に示す [14]。

デジタルビジョンチップは均一な構造を持つＰＥをアレイ状に並べ、上下左右に接続、近傍通信を行っている [15]。

ＰＥの構成は図 2.2 のようになっており、大きく分けてトラッキング部、総和演算部に分けられ、トラッキング部はＰＤからの入力に対し、入力画像の抽出を行い、ＰＥを行方向に直列につなげることで、総和演算部を用いて各行の出力の総和を計算する。また、行、列それぞれにグローバルパスが引かれ、シフトレジスタを通じて外部からデータが与えられるようになっている。そして、処理結果は右端の列総和演算回路によって出された列総和情報と行方向の総和の計算を元にアレー全体の総和を計算して算出する。(図 2.3)

今回用いたビジョンチップは石川研究室と共同開発したＮＰＣのＳＲ３３００であり、32 × 48 のＰＥを内蔵した高速対象追跡ビジョンセンサである。これは、処理をターゲットトラッキングに特化したチップである。専用ハードウェアで構成されていることで高速動作が可能であり、ごく小さい回路で実現されている。

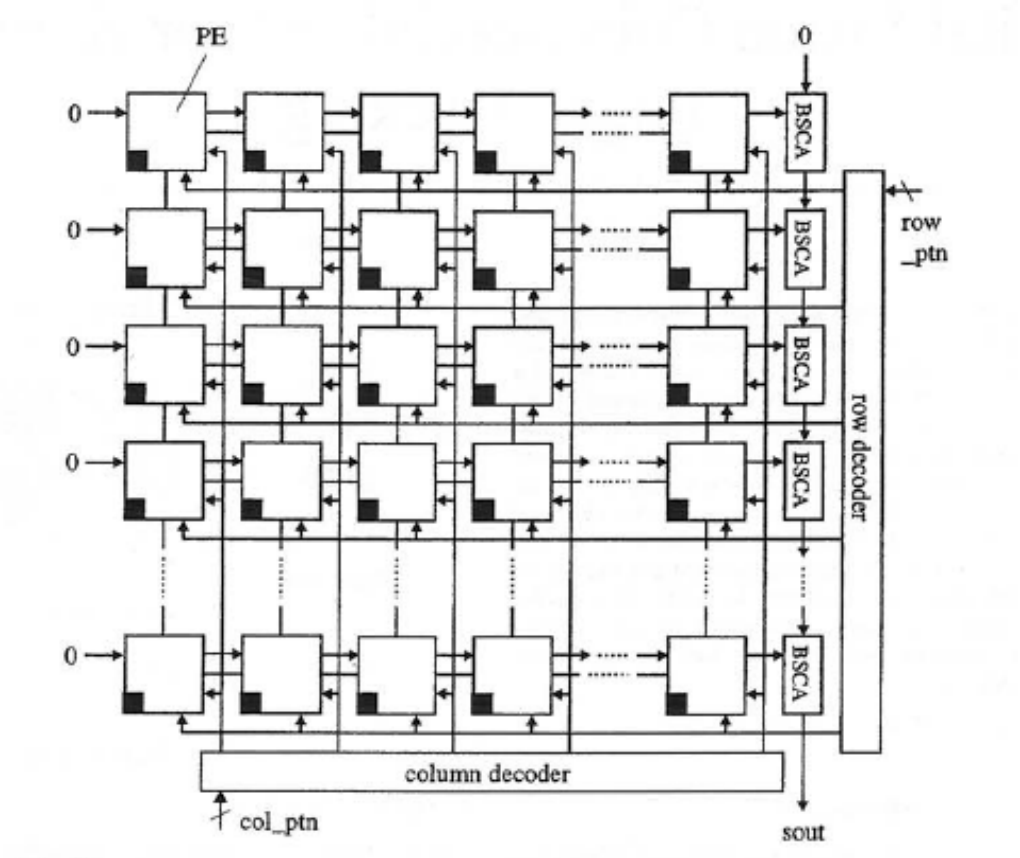

図 2.3. ビジョンチップの構成 [14]

2.2 ビジョンチップによる高速画像処理

ビジョンチップのＰＤから得られた光信号は、光電変換により電圧値に変換された後２値化されてＰＥに送られる。この情報をＰＥ内のデジタル部の変換を通すことによってネガポジ反転、トラッキング、行総和演算を行えるようになっている。これにより、電圧値の閾値を設定し、それを越えた部分全体の重心をトラッキングすることが出来る。

本研究ではパックをビジョンチップで捕らえ、トラッキングすることでパックの位置情報を得ている。この位置情報を元に、OpenGL でパックの位置を描画する。ただし、ビジョンチップの画像は四角形の左隅を (0,0) とし、48 × 32 で位置情報をとらえ、OpenGL は中心 (0,0)、幅 2 × 2 で描画する為にそのままでは OpenGL に描画することが出来ない。そのため、本研究ではビジョンチップから得られた位置情報を OpenGL に描画できるように変換している。

2.3　フィルタ

ビジョンチップで対象をトラッキングする方法として２値化を使っているが、閾値を越えた部分全体の重心をとるためパックのみが閾値を超える状態を作らねばならない。しかし、そのままビジョンチップに写してしまうと、着てる服や周りの環境に影響されて閾値を越える部分がパック以外に出てきてしまう可能性があり、ゲームをする上で問題が生じてしまい、この問題は閾値の調整で解決するのは難しい。

そこで、本研究ではビジョンチップにフィルタをかけ、ある一定の幅の波長しか通さないようにした。フィルタとして LEE181 の Congo Blue のシートと LEE106 の Prymary Red のシートを重ねた物を用いた。LEE181、LEE106 それぞれフィルタを通した時の光の波長による透過度はそれぞれ図 2.4、2.5 のようになっている。

この２つを通すことで波長が 800nm 以下の光を殆どカットすることが出来、更に、800nm 付近の光を特に強く透過することが出来る。一定波長以下の光をカット、更にある波長付近の透過度を強くすることが出来れば電圧値がその波長付近の光だけを高くすることが出来る。つまり、パックに 800nm 付近の波長の光を出すことが出来れば、パックにだけ強く反応する。そこで、パックの中心に波長が 800nm 付近の光を発するＬＥＤを取り付け、ビジョンチップがＬＥＤを捕え、他の部分がフィルタによって２値化したとき閾値を超えないようにすることでＬＥＤを取り付けたパックの場所だけを正確にトラッキングできるようにした。元の映像が図 2.6、フィルタを通さなかった場合の２値化された映像は図 2.7、フィルタを通した場合の２値化された映像は図 2.8 である。十字が付いているところが閾値を越えた部分の重心となる。この映像からも分かるとおり、フィルタを通さないとパック以外の部分も２値化したときビジョンチップが読み取ってしまう。特に、電気をつけていると電気が表面に反射して強い光量をもってしまうので、どうしてもパック以外の部分を閾値を越えた部分として判別してしまう。ところが、図 2.7 と図 2.8 を比べて貰うと分かるように、フィルタをかけると背景の光などをほぼ無視して、パックの重心位置を正確にとらえることが可能であることが分かる。

2.4　アクチュエータの構想

1 章の研究の概要に書いた、センターラインでパックを止めるアクチュエータには、スライダを用いる。パックを掴む為の装置としては電磁石を用いる。スライダでは、ビジョンチップによって得られた情報からパックの位置、速度を検出しパックが到達する位置にパックの速度と同じ速度で移動する。遠隔地にあるホッケー台のアクチュエータも同様に動作する。パックが自分のフィールドから来た場合は、スライダによってパックの位置まで移動し、電磁石でパックをくっつける。逆に相手のフィールドでは、同様に速度をもったままパックがくっついた位置まで移動しそのままパックを放す。これによって、パックの横方向の速度はセンターラインに入ったときと同じ速度となる。

パックを掴む電磁石は、電圧でその磁力の向きと力が変わるように設定する。パックの縦方

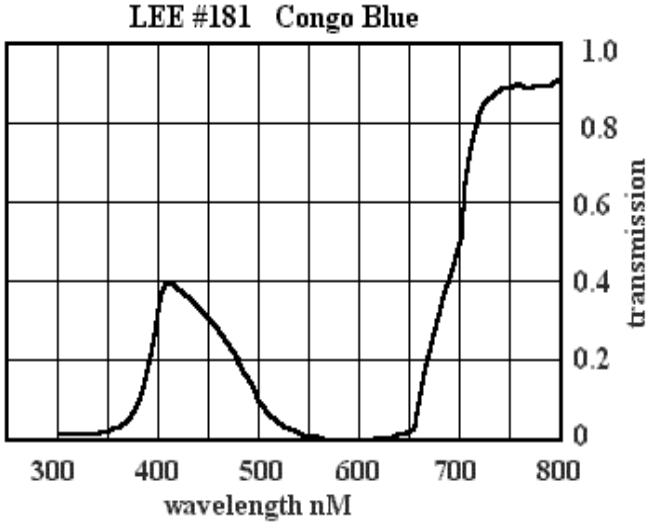

図 2.4. LEE181 を付けたときの光の透過度

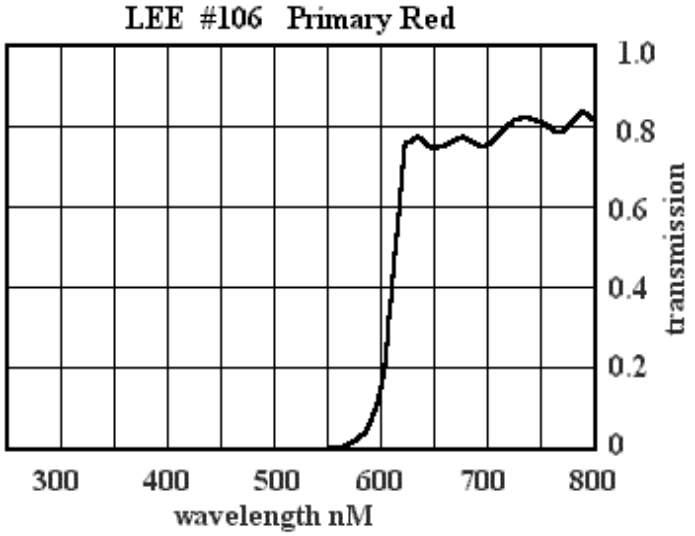

図 2.5. LEE106 を付けた時の光の透過度

向の速度に応じて電圧を変化させ、相手側の電磁石がパックをくっつけた時の速度に応じた電圧を電磁石にかけ、相手側でパックがくっついたと同時に、その速度と同じ速度を縦方向にだせるようにパックを放す。こうすることでパックは縦方向でも同じ速度を得ることが出来る。

このようにして、センターラインで適当な速度でパックを掴んだり放したりすることで遠隔地の人が打ったパックがセンターラインに到達した位置で横方向にも縦方向にも同じ速度でセンターラインを超えて返ってきたように見せることが出来、まるでその場で向こう側からパックがそのままこちらに来た様に見せることが出来、遠隔地の人エアホッケーをやることができ

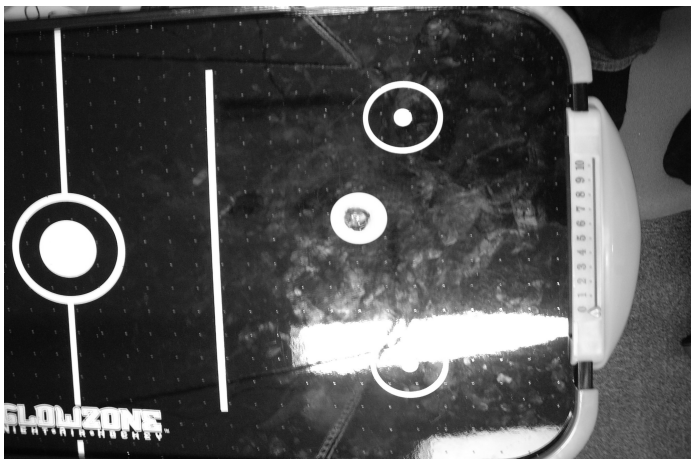

図 2.6. 元の映像

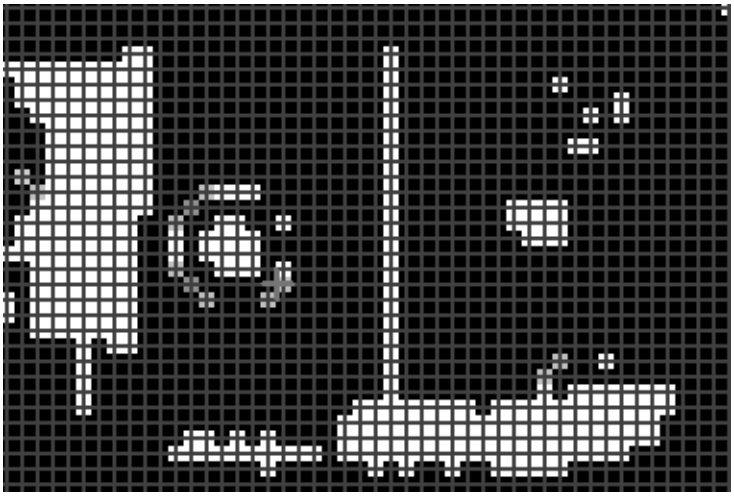

図 2.7. フィルタをかけていないときのビジョンチップで見た 2 値化された映像と重心位置

るようになる。

ただし、本研究ではこのアクチュエータに関することはやっていない。

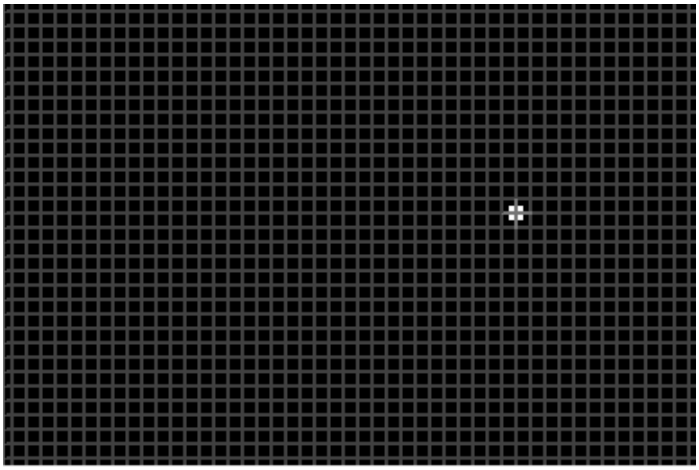

図 2.8. フィルタをかけたときのビジョンチップで見た 2 値化された映像と重心位置

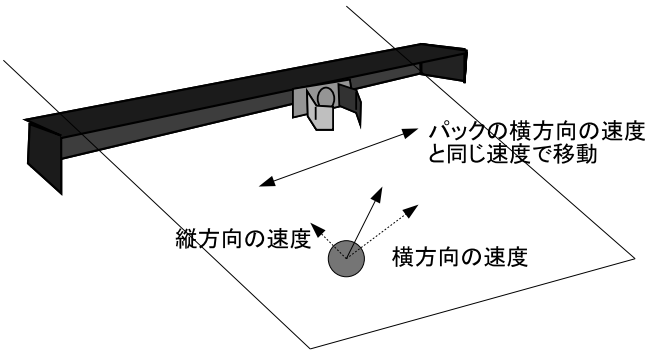

図 2.9. スライダの動きとパックの動き

第 3 章

インターネットを介したデータの送受信

3.1 Winsock

3.1.1 Winsock とは

今回インターネットを介した情報の送受信のアプリケーションとして Winsock を用いた。Winsock とは、Windows 用のネットワークに関するＡＰＩ（アプリケーションプログラミングインターフェイス）であり、明確に定義されたデータ構造タイト関数呼び出しがダイナミックリンクライブラリ（ＤＬＬ）の形式で実装されている。Winsock は一種の翻訳機のようなものであると考えてくれると理解しやすい。たとえば、自作のアプリケーションで汎用のネットワークサービスを要求する関数呼び出しを行ったとすると、Winsock はこの汎用の要求をプロトコル固有の要求に翻訳、必要なタスクを実行することが出来る。

つまり、Winsock とはネットワークとアプリケーションの実装の間で下位のネットワークプルトコルの詳細をアプリケーションプログラマから隠す役割を担っているものである。

Winsock の位置づけは図 3.1 のようになっている。Winsock 対応のアプリケーションは、Winsock 対応のどの実装上でも再コンパイルをせずに実行することが出来る。

3.1.2 ソケット

Winsock は Beerkeley ソケットモデルを用いている。これは、ネットワーク通信をネットワーク通信の端点を存在させ、その端点を「ソケット」と呼び、その２つのソケットの間で通信を行うというものである。Beerkeley ソケットライブラリの大半は Winsock でも同じ関数名とパラメータで使用できる。Beerkeley ソケットモデルは２つのネットワークアプリケーションを同時に起動するのでなく、一方のアプリケーションを理論的に使用可能な状態に置き（サーバー）、もう一方のアプリケーションから必要に応じて情報を要求する（クライアント）ようにする、クライアント/サーバーモデルを用いている。

端点となる２つのソケットはこのことから「サーバーソケット」と「クライアントソケット」

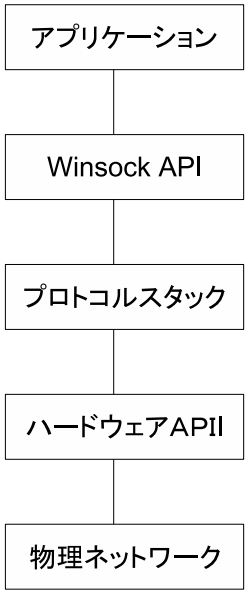

図 3.1. Winsock の位置付け

と言う。その名のとおり、サーバーソケットがサーバーの役割を、クライアントソケットがクライアントの役割を担っている。サーバーアプリケーションを行う側は、まずソケットを作成し、ソケットに名前を付ける。そして、クライアントからの要求を待つ状態になる。クライアントアプリケーションを行う側は、ソケットを作成し、サーバーソケットを名前、もしくはアドレスによって探し、それにデータを要求することで２つのソケット間でデータの行き来が可能になる。Winsock プログラムを使うには、サーバーアプリケーション、クライアントアプリケーション２つ１組のアプリケーションを使うクライアント/サーバーモデルに基づいて設計されている。

クライアントソケットがサーバーソケットと接続、通信する為には、次の３種類の情報が必要となる。

- プロトコル
- サーバーのアドレス
- サーバーのポート番号

この、プロトコルが何かを考える上で必要になってくるのが接続型と非接続型の選択である。サーバーソケットとクライアントソケットを用いたアプリケーションを用いる際に接続型と非接続型の２種類から選ばねばならない。接続型は端点を２つ確立させ、その端点間のみで情報のやり取りを行うもので、アプリケーション毎に相手の識別を行う必要がない。これは、一般的にＴＣＰ/ＩＰを用いるアプリケーションに使われ、大量の情報を交換するのに適している。非接続型は端点をデータの送受信の度にソケットの識別情報を必要とする。これは、一

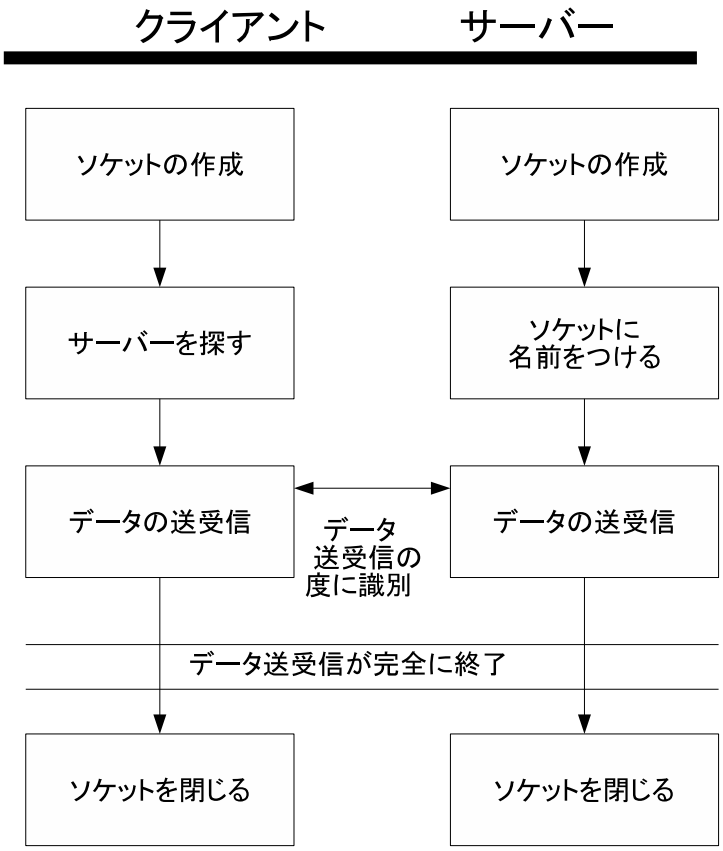

図 3.2. 非接続型アプリケーションの流れ

般的にＵＤＰ/ＩＰを用いるアプリケーションに使われ、少量の情報を高速に交換するのに適している。

本研究では、送受信する情報が少なく、より早い送信速度を必要としたため、非接続型のアプリケーションを用いた。非接続型アプリケーションを用いた時の流れを図 3.2 に示す。非接続型アプリケーションの場合、あて先のアドレスをデータの送信の度に指定せねばならないため、sendto() の処理内でアドレスを指定し、recvform() を使ってデータを受け取る。

3.2　サーバー・クライアント間の情報の交換

対象の位置情報は、様々なゲーム、使用目的に必須のものである。エアホッケーというゲームでも例外ではない。

実際にエアホッケーをやっている所を想像してもらえると分かるようにエアホッケーを行う

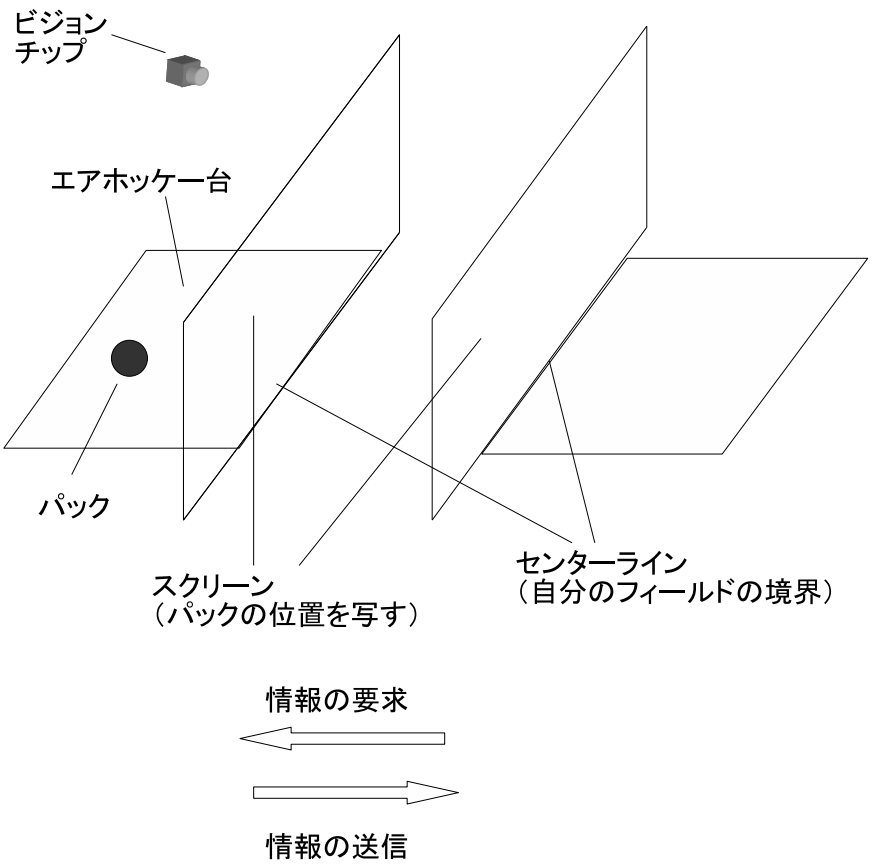

図 3.3. 全体の流れの概念図

ときに必ず必要となるものは、パックの情報である。遠隔地の人物とエアホッケーを行う際に, 相手側のフィールドでパックがどのように動いているかをリアルタイムで知ることが出来なければ、ゲームとして成り立たなくなる。本研究では、パックの位置情報を得る為に、２章で説明したビジョンチップを用いた。相手のフィールドにあるパックの情報を要求する方をクライアント、情報を送信する方をサーバーとして考える。ビジョンチップによって得られた情報をクライアント側に送信、その情報を元に OpenGL によって描画を行っている。

この流れは図 3.3 のようになっている。

本研究では、一方を仮想的にエアホッケーをしているようにプログラムを打った状態、もしくはエアホッケー台にあるパックの位置情報をビジョンチップによって算出、パソコンに入力し相手側に送信するようにし、もう一方をプログラムによって仮想的にビジョンチップがあるものとしてパックの位置情報を算出し、パソコンに入力している。

3.3　サーバー・クライアントの役割交換

パックの情報を送信するのは、パックが自分のフィールドにある時だけでよく、自分のフィールドにパックがない時に送信する必要はない。これは、パックが自分の所にない時に相手側に情報を送信してもあまり意味がなく、送受信の時間が無駄になり実時間性を重要とするゲームシステムを構築する上で（少なくともエアホッケーでは）無駄にかかる時間は省いた方が良いためである。つまり、パックがセンターラインを超え相手のフィールドに入った瞬間に情報を送信する必要はなくなり、情報を要求する側になる。

逆にパックが自分のフィールドになく、今まで情報を要求していた側はセンターラインを越えた瞬間に情報を送信する側になる。つまり、サーバーアプリケーションとクライアントアプリケーションがセンターラインを境に交代するようにせねばならず、どちらの側もサーバーアプリケーション、クライアントアプリケーションを行えるようにしなければならない。

本研究でデータの送受信に用いている Winsock は、２つのソケット間での情報の送受信を行うのが特徴だが、今回はどちらのソケットにも「サーバーソケット」「クライアントソケット」両方の働きが出来るようにし、センターラインを境にサーバーソケットとクライアントソケットの役割が変わるようにした。

非接続型アプリケーションでは、データを送信する度に相手のアドレスを確認する為、データの送信先のあて先を指定せねばならない。init ファンクションでクライアントソケットにサーバーソケットのアドレスを指定すれば、サーバーソケットはクライアント側が送信してくるのを待つ状態になり、クライアントから情報の要求が送信されてきた時に、自動的にクライアントソケットのアドレスを得ることが出来、自動で分かるようになっている。アドレスが自動で分かった後は、no evet ファンクションで情報の送受信を行う。これが一般的に使われるプログラムの流れである。

しかし、init ファンクションは一度呼び出されると二度と通過しない。つまり、これではサーバーソケットとクライアントソケットの役割が変わった時に今までサーバーソケットの役割を果たしており、送って来たアドレスに返せばよく、アドレスを自動的に更新していた側がクライアントソケットに役割が変わった時、今度は相手に情報の要求を送信しなけらばならない。しかし、それまではアドレスを自動で更新しており相手のアドレスを確定させていなかったので相手のアドレスが分からず、役割交換をした際にデータの送受信が出来ない、と言う問題が起こる。

そこで、今回は init ファンクションに相手側のアドレスをどちらも事前に入力しておき、ソケットの役割が変更しても相手側のアドレスが分かるようにしている。このプログラムの流れを図 3.4 に、ソケットの役割交換の流れを図 3.5 に示した。

こうすることで、サーバーソケットとクライアントソケットの役割が変更されたときアドレスが分からず情報の送受信ができない、という問題を解決することが出来る。

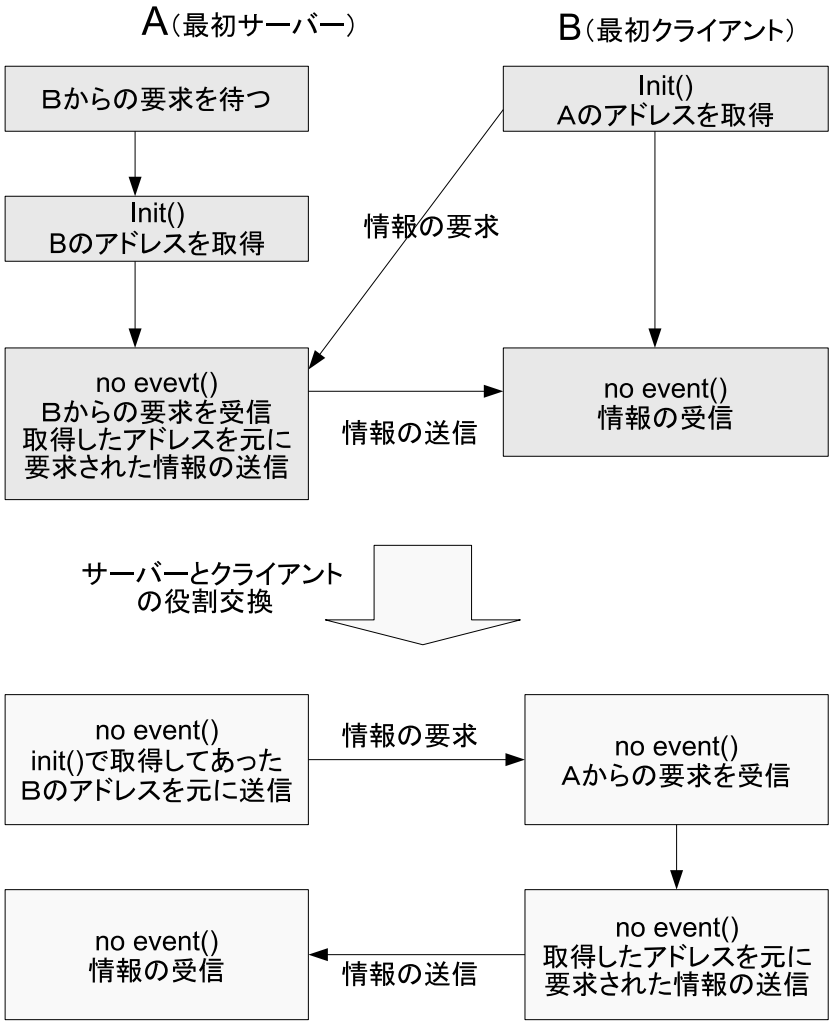

図 3.4. サーバー、クライアント交換の時のプログラムの流れ

3.4 時間差に対する対処

センターラインでスライダを動かす時、求めた速度に達するまでに加速しなければならず、その時間分だけ時間差が生じてしまう。この問題を解決する為に軌跡予測をし、速度、到達位置を予測し時間差分だけ先に到達地点にてパックを放す、という方法を考えている。こうすれば時間差を感じることなくゲームをすることが出来る。

また、情報を送受信する際にはどうしても時間差が生じてしまう。パックの位置を把握した後相手に情報を送信し、スクリーンに映すまでの時間の流れは図 3.6 に示した。情報の送受信にかかる時間を　t とすると、パックがセンターラインを超えて相手のスクリーンに映し出され相手のほうからパックが放出され自分の方に情報が送信されスクリーンに描写されるまでの

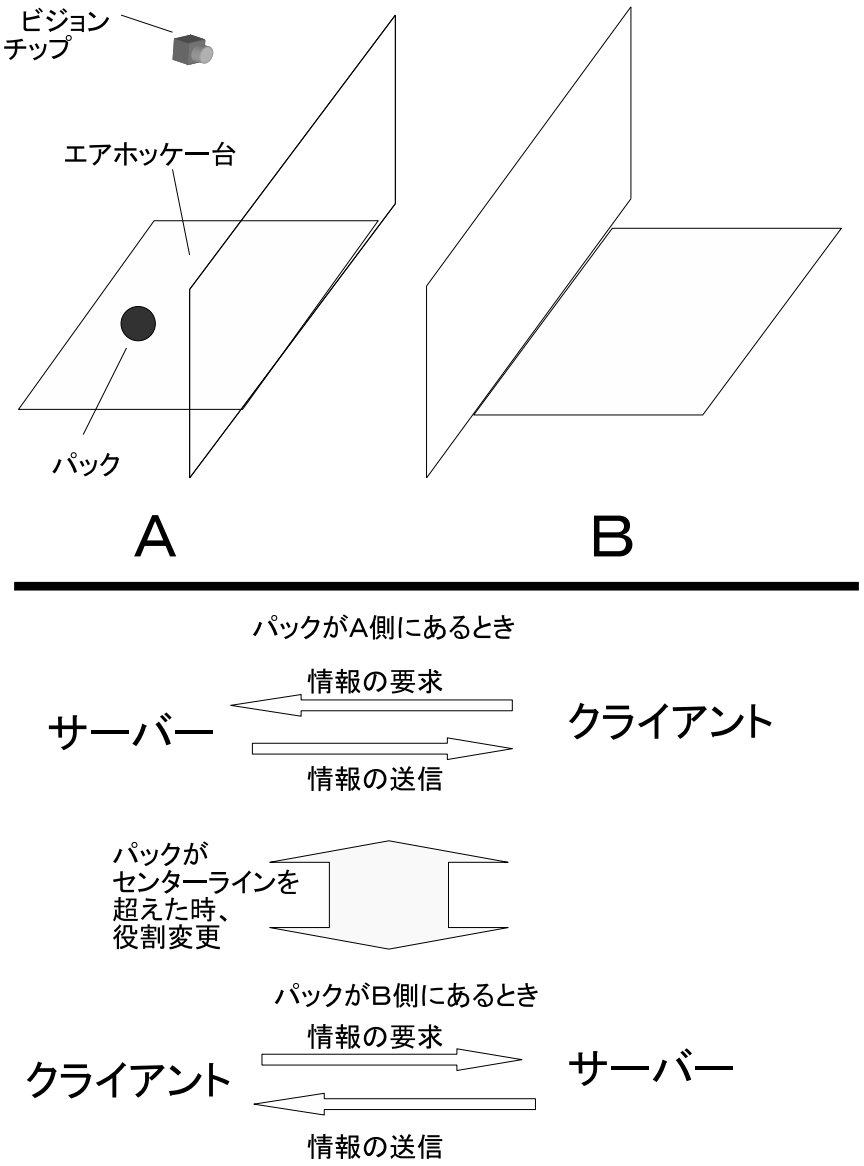

図 3.5. サーバー、クライアント交換の流れ

時間差は２　ｔとなる。この２　ｔの時間差が生じる時間差の最大である。人間が止まっていて違和感を感じる時間は 100ms と言われている。つまり、この２　ｔを 100ms 以内に収めれば違和感なくゲームを行うことが出来る。この為に、時間差を測定してこれに対処している。

3.5　送信時の情報の形

これまでに情報の送受信の方法を述べてきたが、エアホッケーを遠隔地にいる人と行うときに必要な情報を選択した後に、送らなければならない。エアホッケーをする際にはパックの情

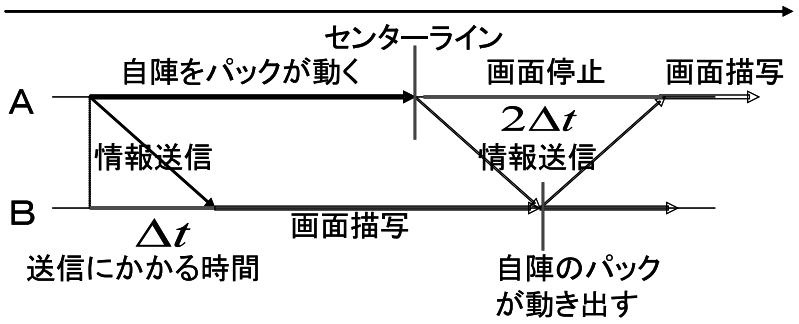

図 3.6. サーバー、クライアント交換の流れ

報が相手に伝わればよい。本研究ではパックの位置座標、速度などを要求、送信している。情報を一つ一つ送信する、と言う方法もあるが、それでは個々の情報それぞれに対してサーバーソケットがクライアントソケットの応答を待ち、その後に送信しなければならず、情報の個数が多くなれば多くなるほど送受信の回数も増えてしまうと言う問題が出てくる。送受信の回数に比例して、相手に情報を送る時間も増えてしまう。そうなると、実時間性が低くなりゲーム性を低くしてしまう。そこで本研究では複数の情報を一回の要求でまとめて送信出来る手法を用いている。複数の情報をまとめる作業は情報の送受信に比べて格段に短い時間で行うことが出来、送信する情報の数が増えても致命的な時間差を生むことがなく、実時間性を保つことが出来ると考えられる。以下に、情報を一つにまとめる方法を記す。

まず、サーバーソケットでデータを格納する配列 Data[n] を作る（n は送信したいデータの数)。そして、Data[0]、Data[1]・・・にそれぞれ必要なデータを格納する。その後送信するデータのアドレスの最初を Data[0] の最初のアドレスに指定。そこから、配列 Data[n] 個分のサイズの情報を送信する、と言う指令をする。こうすることで、複数の情報が一つの情報としてクライアントソケットに送信される。もちろん、送信回数は一回である。

クライアントソケットにも Data[n] という配列を作っておき、一つ一つの情報を n 個の情報に切り取っていく。こうすることで、クライアントソケットに渡った情報が再び必要な複数個の情報として取得できる。この流れは図 3.7 のようになっている。

3.6 スクリーンへの描画

ゲームをする為には相手の動きを見なければならない。遠隔地の人とエアホッケーをする為には、スクリーンに相手の動きを写さなければならない。更に、ゲームをやりやすくする為にはスクリーン上に自分のフィールドが自分側、相手のフィールドが向こう側にあるように見え、パックの動きも自分から見て右に動いたら右に、左に動いたら左に動くようにする必要がある。

ところが、サーバー側から送られたパックの位置情報はクライアント側から見たら全く逆の位置座標になる。クライアント側で描画した映像は、サーバー側と同じではゲームがやりに

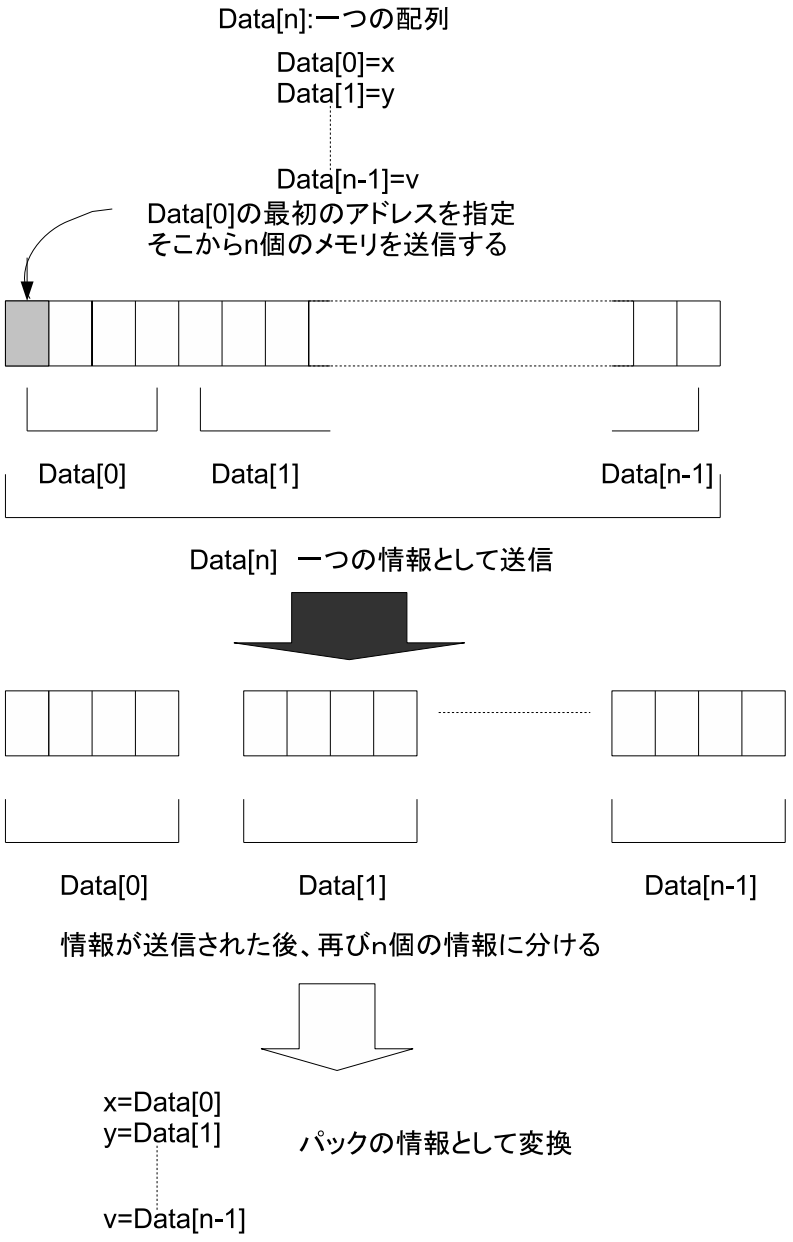

図 3.7. 情報の送信の形

くくなり、ゲーム性が低くなってしまう。本研究では、位置情報をそのまま使うのではなく、送った位置座標をクライアント側から見た位置座標に変換することで対応している。

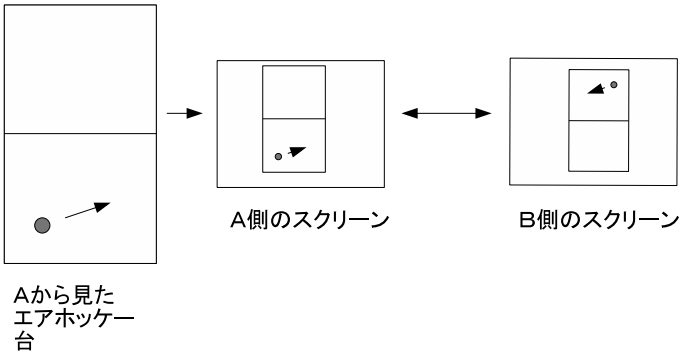

図 3.8. スクリーンの写り方はサーバー側とクライアント側で異なる

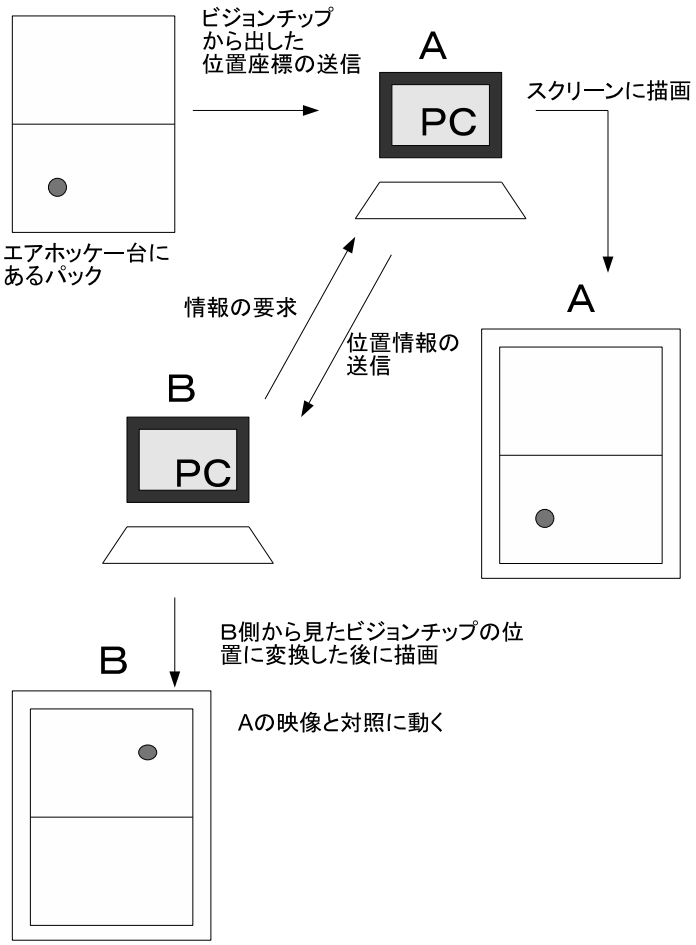

図 3.9. 描画の流れ

第 4 章

実験

4.1 実験装置

図 4.1、図 4.2 は実験装置を示す。
本研究の研究装置は次のとおりである。

- 高速対象追跡ビジョンチップ

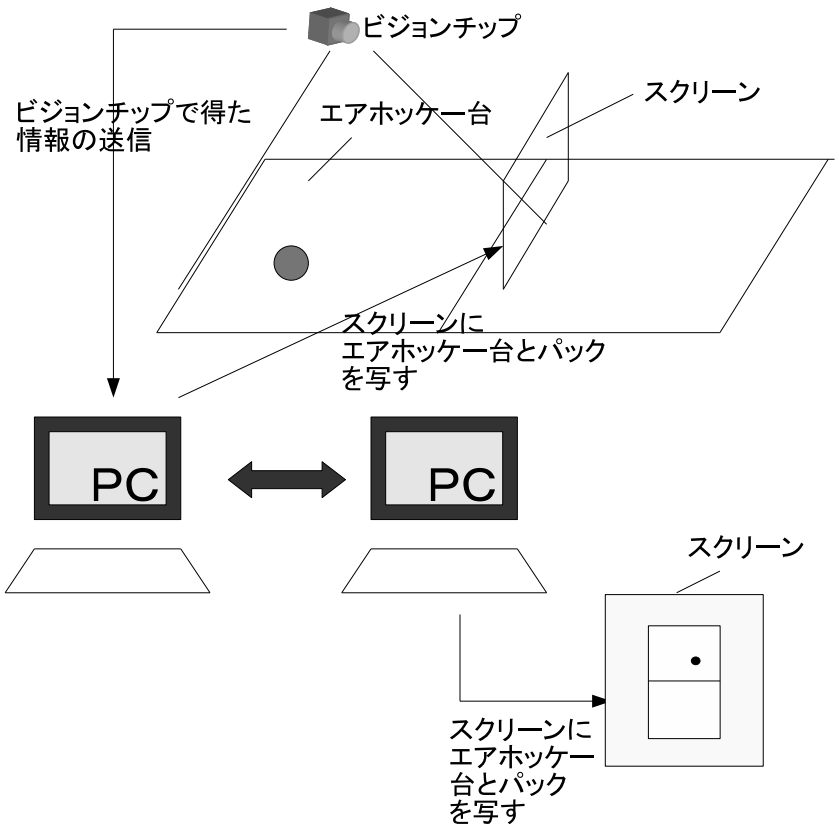

図 4.1. 実験装置の概念図

図 4.2. 実験装置の写真

- エアホッケーの台
- エアホッケーのパック
- 赤色ＬＥＤ
- ＰＣ×２

以下に、どのように研究装置を配置しているかを書く。

エアホッケー台を一つ置き、その上にビジョンチップを取り付ける。ビジョンチップを取り付ける為に、固定する金属棒を用いている。エアホッケー台では通常のエアホッケーを行うのと同様にパックを置く。パックにはビジョンチップでパックを判別する為の波長約 850nm の赤色ＬＥＤを取り付けている。パックのほかに、パックをはじく為のスマッシャーも用いる。更に、ウィンドウズＸＰをＯＳとして積んでいるパソコンを２台用い、そのうち１台にビジョンチップによる情報を送信している。スクリーンには OpenGL によって描画されたエアホッケー台の映像を写している。

図 4.3. SR3300 の写真

4.2　実験設定

今回はＮＰＣ製品であるＳＲ３３００の高速対象追跡ビジョンセンサを用いた。ＳＲ３３００は図 SR3300 のようになっている。以下にビジョンチップの設定を示す。

- 画素数　32 × 48 1536 画素
- 処理周期 30ms
- 露光時間 26ms
- トラッキングモード　シンプルトラッキング・白のトラッキング

今回は、赤色ＬＥＤのみがビジョンチップに写る設定のため、白色をトラッキングし、パック一つをトラッキングすればいいのでシングルトラッキングモードで行った。

4.3　Winsock によるインターネットを通した通信の確認の実験

4.3.1　実験方法

ビジョンチップがない状態で遠隔地の人と Winsock を用いてエアホッケーをやれるかどうかを実験した。エアホッケーをスタートすると、パックが動き出し、それをスマッシャーで跳ね返す。壁に当たったら反射して、相手のゴールに入ったらパックが再び初期位置からスタートするという一般のエアホッケーと同様のゲームのやり方で行えるように設定した。エアホッケーのパックは簡単の為に単純な反射によるもので行う。スマッシャーはマウスにより動かせ、パックがスマッシャーに当たれば反射するようにした。また、ゴールを設定し、ゴール

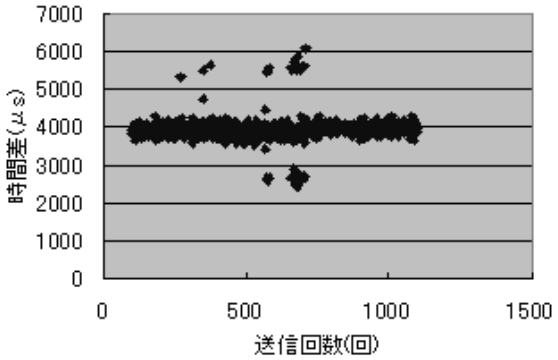

図 4.4. パックの動きと相手のスクリーンへの描画の時間差

の中に入るとパックが初期位置から再びエアホッケーをやれるようになっている。画面では、実際エアホッケーをやった時と同じに見えるようにする為、スクリーンの下のほうを自分のフィールドにしゲームをやってる当事者が見ているパックの動きをスクリーンに映すようにした。

便宜的に最初パックがあるほうをA、逆側をBと呼ぶ。

4.3.2 実験結果

実行した時の映像は図 4.5、図 4.6 のようになった。図からわかるように、パックがA側にあるためにA側から見た映像では自分の方にパックがあるように見え、B側から見た映像では向かい側にパックがあるように見えている。また、パックの動きもお互いに自分から見たとおりに動かせることが出来た。

パックがスマッシャーもしくは壁によって跳ね返り、センターラインを超えてB側のフィールドに入り、サーバーソケット、クライアントソケットの役割が交換した後は図 4.7 図 4.8 のようになった。それぞれでパックが対照の位置にあり、またパックの動きが滞りなく動くことが確認できた。スクリーンに映るまでの時間差は図 4.4 のようになった。

- 平均　3973 μ s
- 分散　310

図 4.4 からも分かるように最高でも 7ms 以下しか時間差が生じていないので、人が違和感を感じる時間 100ms より小さい為違和感を感じることなくゲームを行うことが出来る。

このことから、遠距離の人とインターネットを介して、仮想空間上でエアホッケーのゲームを行うことに成功したことが分かる。

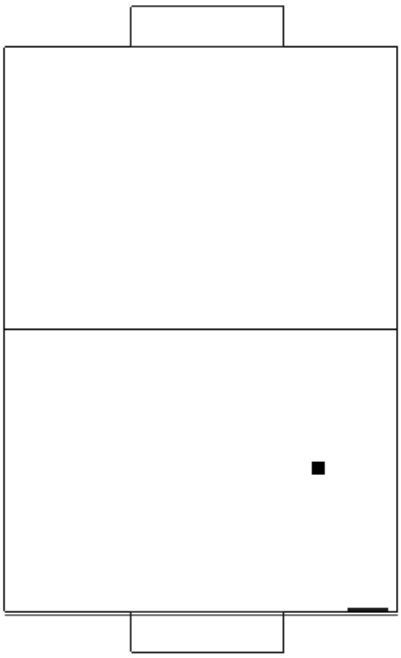

図 4.5. ビジョンチップなしのときのＡ側にパックがあるときのＡ側の映像

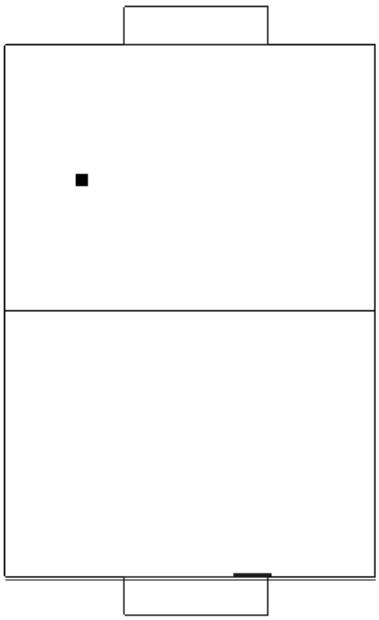

図 4.6. ビジョンチップなしのときのＡ側にパックがあるときのＢ側の映像

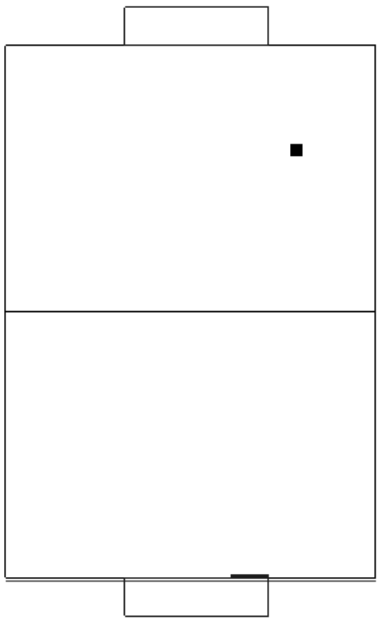

図 4.7. ビジョンチップなしのときのＢ側にパックがあるときのＡ側の映像

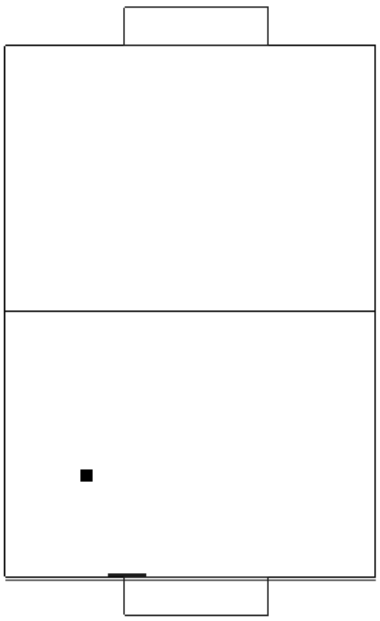

図 4.8. ビジョンチップなしのときのＢ側にパックがあるときのＢ側の映像

図 4.9. ビジョンチップありのときの元の映像

4.4　ビジョンチップを用いた視覚フィードバックの実験

4.4.1　実験方法

パックの位置をビジョンチップにより検出するようにした。ビジョンチップからの情報をＡ側に渡し、その情報からパックを描画、スクリーンに映るようにしている。Ｂ側は前の実験と同じようにヴァーチャル上でパックが動くように設定した。

4.4.2　実験結果

実行した時のパックの位置、Ａのスクリーン映像、Ｂのスクリーン映像はそれぞれ図 4.9、図 4.10、図 4.11 のようになった。

図 4.9 と図 4.10 から確かにパックの位置がＰＣ上でも再現されている。また、インターネットで繋がったＢの方にも図 4.11 から、パックの情報が伝わっており、スクリーン上にＢから見たようにパックが動いてるように描画できた。

また、パックがセンターラインを超えてＢのフィールドに入ると、スクリーン上でスムーズにパックが反対側に行ったように描画出来ており、またＡのフィールドに戻ってくるとパックの位置をトラッキングしていた。このことから、ビジョンチップを用いてエアホッケーをする為のシステムが構築できた。

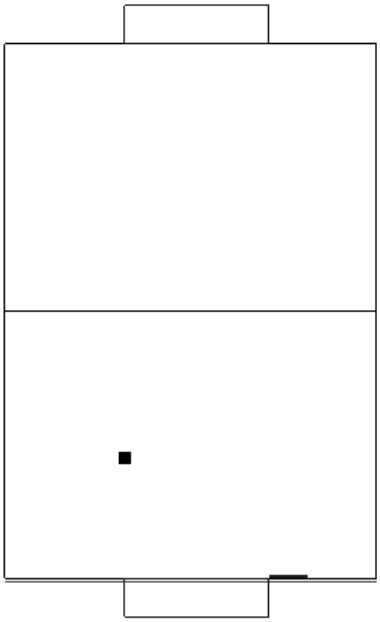

図 4.10. ビジョンチップありのときのA側の映像

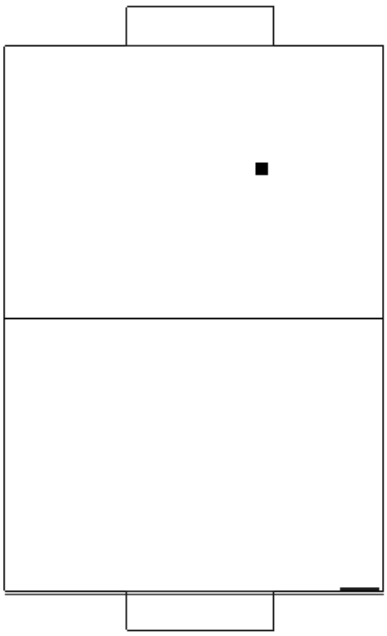

図 4.11. ビジョンチップありのときのB側の映像

第 5 章

結論

5.1 考察

本研究では、遠隔地にいる人とインターネットを通して物理的インタラクションを持ったゲームシステム構築の為の基礎についての提案を行った。ビジョンチップを用いてエアホッケーを行える環境を作ったが、エアホッケーのパックの動きをビジョンチップによってトラッキング、その情報を遠隔地にいる人に送信することが出来た。また、遠隔地にいる人とエアホッケーを行う上で欠かせない、相手側のフィールドでのパックの動きを画面に出すことが出来た。この画面には、それぞれ自分を主観としたパックの動きをみることのできる画像を流すことが出来、本研究で問題点であると考えられた相手のフィールドからの情報を得なければならない際の時間差についても情報の一括化、サーバー・クライアント交換を円滑に行うことが出来ており高い実時間性を持たせることが出来、ゲーム性を持たせることが出来た。

本研究では、まずどちらも仮想空間上でエアホッケーが出来るようにし、その後、一方をビジョンチップによるパックのトラッキングを行い、エアホッケーを出来る環境を設定している。仮想空間上で行えていたエアホッケーを、片方をビジョンチップによるトラッキングに変えてもパックの動きをどっちの側でも変わらずに描画できており、仮想空間をビジョンチップに切り替えることに成功したことが分かる。このことから、仮想空間上で行った側をビジョンチップを用いて行ったとしても同じようにゲームが出来ると考えられる。

5.2 今後の課題・展望

今後の課題として、エアホッケーを相互に物理的インタラクションをもったゲームとして行う為に、相手側でもビジョンチップを用いたパックのトラッキングを行えるようにすることが考えられる。これは、考察でも述べたように今回行った仮想空間上での動きをビジョンチップを用いた場合に切り替えられたように割と容易にできることが想像される。また、ゲームとして実際にパックを打つ為にはセンターラインでパックを止め、相手側でパックを速度を付けて出す為の機構が必要になる。本研究では速度を検出することが出来ているので、その速度を出すことの出来るスライダをセンターラインに取り付け、パックが自分のフィールドからセン

ターライン付近に来たらパックの速度と同じ速度でパックを掴む。逆に、相手側のパックがセンターラインを超えた時に、相手側のパックの速度と同じ速度をだしてパックを放すようにすることが考えられる。これは、横方向だけの速度なので、パックを掴み、縦方向に速度を出す為には電磁石を使うことが考えられる。電磁石を使い、縦方向の速度と同じ速度を出せるように電圧をかけパックを放せばエアホッケーを実際にやったように感じることが出来る。

エアホッケーは２次元で行うものであるが、この研究を更に進めていけば３次元で遠隔地にいる人とゲームすることが可能である。たとえば、卓球などを行うことが出来るようになると考えている。

謝辞

今回この研究を始めた時は、道具も知識もほぼない状態からのスタートであり、自分の目標としたところまで達成できるかとても不安な気持ちで臨んでいました。しかし、研究室の皆さんの助力を得て、なんとか自分の考えていた結果を出すことが出来ました。この場を借りてお礼を申し上げたいと思います。

まず、研究環境を整えてくださった石川正俊教授に心から感謝したいと思います。また、本研究に関する知識が全くなく一体どうすればいいか悩んでいた時に、忙しいときでも常にやさしく指導してくれた AlvaroCassinelli 助手にはとてもお世話になりました。色々と面倒をおかけしたことを申し訳ないと思いつつも、心から感謝しています。有難うございました。また、実験の進め方、論文の書き方等に困っていた時に適切なアドバイスを下さった修士課程１年の伊藤 崇仁にもこの場を借りてお礼申し上げたいと思います。そして、小室孝講師、奥寛雅助手には研究に詰まったときに何度も助けて頂き、ご指導していただいたことを心から感謝しています。また、並木明夫講師には論文を見ていただき、更に研究の助言をしていただきました、有難うございました。そして、何度も相談に行った時に助けてくださった研究室の皆さんに、この場を借りてお礼申し上げます。有難うございました。

参考文献

[1] Florian Mueller,Stefan Agmanolis,Rosalind Picard: Exertion Interfaces Sports for the distance User Interface Software and Technology Symposium in Paris in 2002.

[2] D. Sekiguchi, M.Inami, S. Tachi: RobotPHONE: RUI for Interpersonal Communication, CHI2001 Extended Abstracts, pp. 277-278, 2001.

[3] 関口, 稲見, 舘, オブジェクト指向型テレイグジスタンスによるロボティックユーザインタフェース-形状共有システムの提案と試験的実装- , インタラクティブシステムとソフトウェア VIII: 日本ソフトウェア科学会 WISS 2000, 近代科学社

[4] Mark W.Spong:On the Controllability of an Air Hockey Puck IEEE Int CCA/CACSD 2000

[5] Bradley E. Bishop and Mark W. Spong: Vision-Based Control of an Air Hockey Playing Robot in Proc. IEEE Int.ControlSystems pp23-32 1999

[6] Wen-June Wang; I-Da Tsai; Zhi-Da Chen; Guo-Hua Wang; A vision based air hockey system with fuzzy control: Control Applications, 2002. Proceedings of the 2002 International Conference on Volume 2, 18-20 Sept. 2002 Page(s):754 - 759 vol.2

[7] 小室孝, 鈴木伸介, 石井抱, 石川正俊: 汎用プロセッシングエレメントを用いた超並列・超高速ビジョンチップの設計, 電子情報通信学会論文誌, Vol.J81-D-I, No.2, pp.70-76 (1998)

[8] 吉田 淳，小室 孝，石川正俊: 高速対象追跡ビジョンチップエバレーションボードの紹介, 第 19 回日本ロボット学会学術講演会 (東京,2001.9.20) ／予稿集, pp.237-238

[9] 小室孝, 石井抱, 中坊嘉宏, 石川正俊：デジタルビジョンチップのためのモーメント抽出アーキテクチャ, 電子情報通信学会パターン認識・メディア理解研究会 (函館, 1999.7.16) ／電子情報通信学会技術報告, Vol.PRMU99-51, pp.17-22

[10] 石井抱，小室孝，石川正俊：ディジタルビジョンチップのためのモーメント計算法，電子情報信学会論文誌 D- , Vol.J83-D- , No.8, pp.1733-1740 (2000)

[11] 奥寛雅, 石井抱, 石川正俊：マイクロビジュアルフィードバックシステム，電子情 報通信学会論文誌 D-II, Vol.J84-D- , No.6, pp.994-1002 (2001)

[12] 小室孝, 石井抱, 石川正俊, 吉田 淳: 高速対象追跡ビジョンチップ, 電子情報通信学会論文誌 D-II, Vol.J84-D-II, No.1, pp.75-82 (2001)

[13] Takashi Komuro, Shingo Kagami, Masatoshi Ishikawa: A Dynamically Reconfigurable SIMD Processor for a Vision Chip, IEEE Journal of Solid-State Circuits, Vol.

39, No. 1, pp. 265-268 (2004)

[14] Takashi Komuro, Idaku Ishii, Masatoshi Ishikawa and Atsushi Yoshida: A Digital Vision Chip Specialized for High-speed Target Tracking, IEEE transactions on Electron Devices, Vol.50, No.1, pp.191-199 (2003)

[15] 小室 孝, 鏡 慎吾, 石川 正俊: ビジョンチップのための動的再構成可能な SIMD プロセッサ, 電子情報通信学会論文誌 D-II, Vol. J86-D-II, No. 11, pp.1575-1585 (2003)