2019 年度(令和元年度)

創造工学セミナーⅡ Final Report

Interactive Floor Interface を用いた教育支援システムの開発

研究メンバー

赤坂 総司 S5-15006

井上 海翔 S5-16010

大野 智哉 S5-16012

大原 広暉 S5-16013

新海 大志 S5-16038

新屋 吉昭 S5-16040

指導教員

金丸 隆志 教授

所属研究室

知能機械研究室

要旨

近年の日本の教育現場（中等教育）では情報社会化に伴って、ICT 機器の導入が進められている。例えば、90％を超える中学校がプロジェクタを導入している。また、文部科学省による新学習指導要領では、主体的・対話的で深い学びを実現する生徒参加型授業が推奨されており、生徒参加型授業は座学学習より高い理解度を得られるという研究結果も出ている。

そのため、我々はプロジェクタで床に映像を投影するインタラクティブなインターフェース（Interactive Floor Interface、以下 IFI）を開発することにした。IFI はユーザーのジェスチャーを認識し、それに対応した情報を提示することができるインタラクティブなシステムである。また、IFI で利用できる教育コンテンツ「水面波の干渉アニメーション」の開発も行った。そして、IFI と「水面波の干渉アニメーション」を組み合わせた教育支援システムの開発を目指した。

開発した教育支援システムの教育効果を評価するために評価実験を行った。高校物理の教科書で学習したグループと教育支援システムで学習するグループに分け、学習前後の試験結果を比較することで評価する。また、結果を統計的に評価するためにt検定を行った。

実験の結果、教科書で学習した学生と教育支援システムを利用した学生はともに学習前よりも学習後の方がよい試験結果となった。t検定を行った結果、どちらの試験結果に対しても有意差が認められ、どちらの方法も学習効果によって点数が上がったと言える。

また、教育支援システムを利用したグループの方が、平均点の増加量が大きくなった。こちらもt検定を行ったが、有意差は認められなかった。そのため、教育支援システムを利用することが教科書を読むことよりも優れた教育効果を持つとは断定できなかったが、今後のシステムの改善により、優れた教育効果を生みだせる可能性はある。

1 緒論

本章では、まず、近年の日本において情報社会化が教育現場に与える影響について述べる。また、教育分野では授業形式の多様化が進んでいることを述べる。そして、その 2 つの事実から「プロジェクタ」と「生徒参加型授業」に注目し、〈教育効果の高いプロジェクタを利用したシステム〉を目標としたことを説明する。

1.1 近年の日本における中等教育 *大原広暉*

本節では、はじめに情報社会化が進む近況における学校での各 ICT 機器の導入率を示し、続いて、生徒参加型授業は教育効果の高い教育方法として各学校で実施されていることを紹介する。

1.1.1 情報社会化 *大原広暉*

近年、IoT[*1] が社会に大きな影響を与えるようになってきた。たとえば図 1-1 に示すように 2019 年の国内 IoT 市場全体の規模は約 6,100 億円となっており、対前年比で約 45%増となった。2023 年には市場規模がおよそ 3 倍に拡大することが見込まれている。これにより、IoT が社会に及ぼす影響はこれからも拡大すること(＝情報社会化)が予想される。

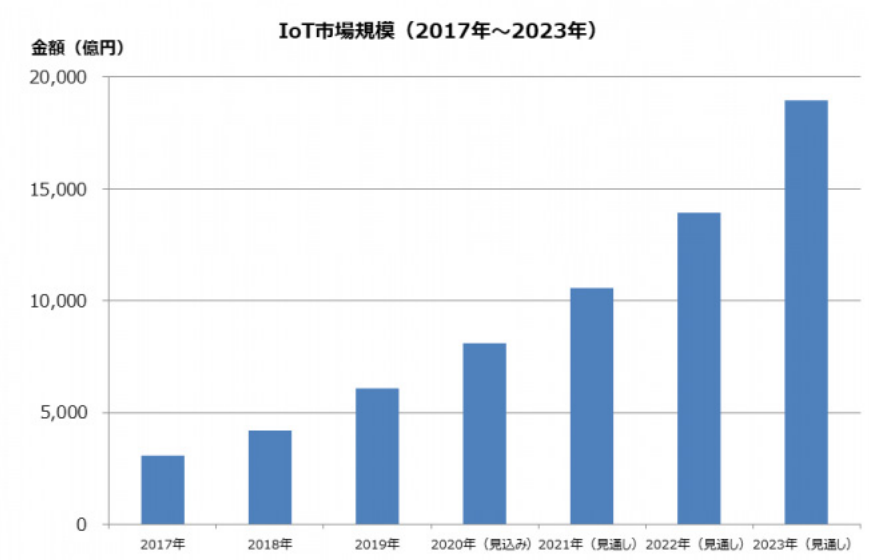


図 1-1 国内の IoT 市場規模(2019)【出典:[1]】

---

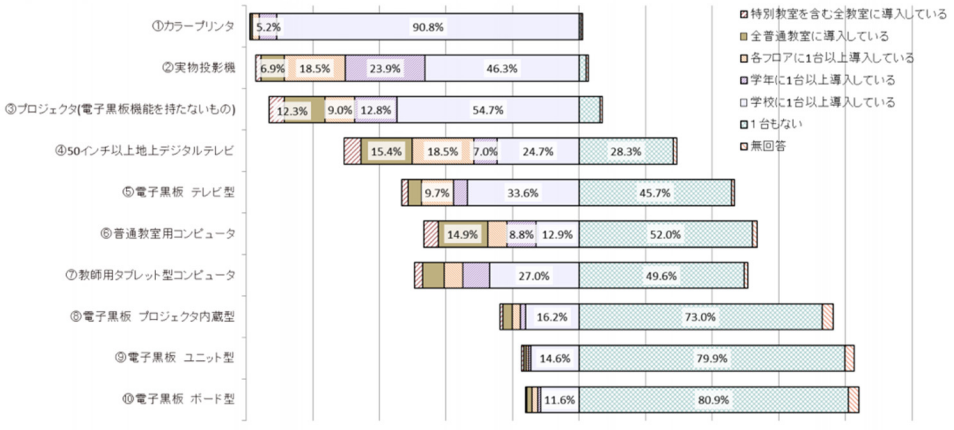
*1 Internet of Things の略称であり、モノとモノがインターネット経由で繋がっていること。

このように、IoT が普及していくことは情報社会化が進んでいくことに繋がり、その対策として教育分野が見直されている。情報社会では、今後の社会がどのように変化していくか予測することが難しいと言われている。しかし、次世代の子どもたちは義務教育の後、情報社会のなかで自立しなければならない。その対策として、2018 年度、文部科学省は新学習指導要領総則【出典:[2]】に「情報活用能力の育成」と「学校における ICT 環境整備と ICT を活用した学習活動」を加えた。

それにより、教育現場では ICT を活用した授業の実施が進められている。ICT を活用するとは、プリントを印刷することから、授業内で生徒がパソコンを使うことまで意味し、ICT を活用した授業は学力向上に繋がることが報告されている【出典:[3]】。

本研究では〈ICT を活用した授業〉とは〈ICT 機器を使いながら教える授業〉であるとする。たとえば、書道の時間に、実物投影機を使って先生が書く様子を全体に見せながら説明することが挙げられる。そのため、プリンターで印刷したプリントを配布することは除く。言い換えると、教員が ICT 機器を使って〈リアルタイムで〉生徒に教えることを〈ICT を活用した授業〉と定義する。

〈ICT を活用した授業〉を実施するために各学校に整えられている ICT 機器のなかで、導入率が 90%を超える ICT 機器は実物投影機とプロジェクタである（図 1-2）。カラープリンターは導入率がもっとも高いが、今回の〈ICT を活用した授業〉を実施することができないため除外した。



＊上記の表・グラフは、「1台もない」「無回答」の値が小さい順に各機器の集計値を並べたものである。

図 1-2 中学校における ICT 機器の導入率【出典:[3]】

1.1.2　生徒参加型授業の普及　　*新海大志*

文部科学省は「新しい学習指導要領の考え方−中央教育審議会における議論から改訂そして実施へ−」【出典:[4]】において、情報化の社会的変化が人間の予測を超えて進展するようになってきたため、予測できない変化に主体的に関わることが重要であると述べている。そこで、文部科学省は学校教育では知識や技能だけでなく、思考力や判断力の育成の充実を目指している。それを実現するために、新学習指導要領では主体的・対話的で深い学びを実現する生徒参加型授業を推進している。マナビラボが実施した 2016 年の調査【出典:[5]】によると、授業参加型授業を取り組んでいる教科がある高校は 75.5%であり、生徒参加型授業が普及されてきていることを示している。

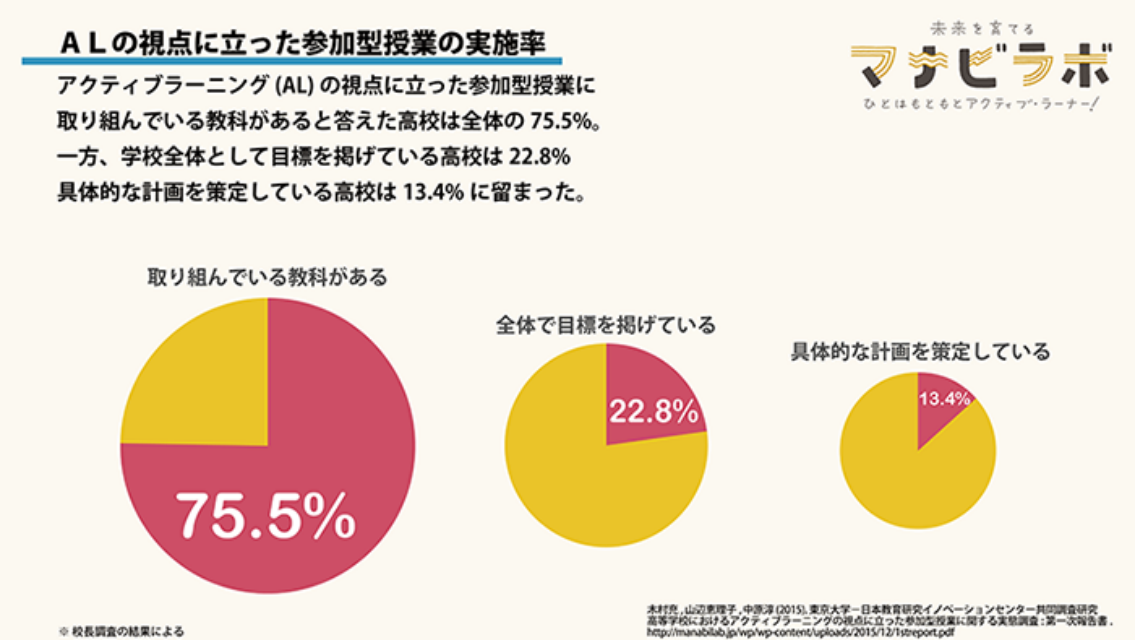


図 1-3 参加型授業に関するアンケート結果【出典:[6]】

1.2 プロジェクタの授業内活用

大原広暉

本節では 1.1 節で触れた ICT 機器の中から、プロジェクタについて注目する。

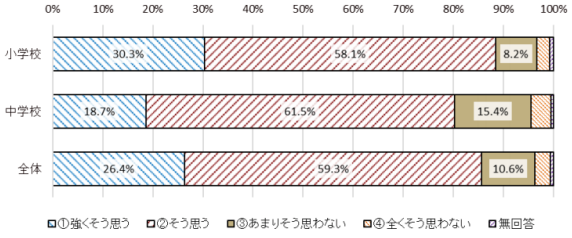
プロジェクタは、大型提示装置に分類される。大型提示装置とは、ICT 機器のなかのプロジェクタ、電子黒板、インタラクティブホワイトボード、電子情報ボードのことを指す。〈ICT を活用した授業〉によく使われる ICT 機器は、大型提示装置と実物投影機、コンピュータである。

プロジェクタを含む大型提示装置とデジタル教材の導入で、教員はわかりやすい講義が実施できると感じている。図 1-4 のアンケート結果では、85.7%の小学校・中学校の教員が大型提示装置とデジタル教材を活用することが授業にプラスになると答えた。すなわち、ICT の活用は教員にとって〈良い影響〉を与える。

大型提示装置の中で、プロジェクタの使用率がもっとも高い。さらに、プロジェクタの導入率も高い。実際に授業内でプロジェクタは、スクリーンに映像を投影し、教室全体に映像と音を使って説明するために使用されている。たとえば、生徒のプレゼンテーションや、情報の授業で教員自身の PC 画面を表示する場合などである。

しかし、この方法では教員が投影映像の前に立って説明することが難しい。たとえば、教員が投影映像の前に立って説明すると、教員に映像が被ってしまう。このとき、授業に参加する生徒は映像が見にくくなるという課題がある。

	小学校		中学校		全体	
	回答数	回答率	回答数	回答率	回答数	回答率
①強くそう思う	365	30.3%	114	18.7%	479	26.4%
②そう思う	700	58.1%	376	61.5%	1,076	59.3%
③あまりそう思わない	99	8.2%	94	15.4%	193	10.6%
④全くそう思わない	31	2.6%	24	3.9%	55	3.0%
無回答	9	0.7%	3	0.5%	12	0.7%
回答者数（母数）	1,204	100.0%	611	100.0%	1,815	100.0%



※ 全体で「①強くそう思う」「②そう思う」を合わせて 85.7%となっており、大型提示装置とデジタル教材でよりわかる授業の実践が進んでいるものと思われる。

図 1-4 質問「電子黒板やプロジェクタ等の大型提示装置、デジタル教材の導入で、よりわかる授業を実施できるようになった。」に対する回答結果（2017 年）【出典:[3]】

1.2.1　授業内における新たなプロジェクタの活用方法　*赤坂総司*

1.2 節で述べた課題を解決するために、我々は〈床〉に注目した。スクリーンでは教員が前に立って説明すると見えない。しかし、図 1-5 に示すように床に投影することで生徒は 360 度から映像を見ることができる。ただし床に映像を投影する場合でも教師の影で映像が見えなくなる。例えば、生徒 A は教員の影で映像を見にくい。

そこで、我々は次の項目で紹介する類似研究を参考にし＜完全に＞360 度から映像を見られる授業を目指す。

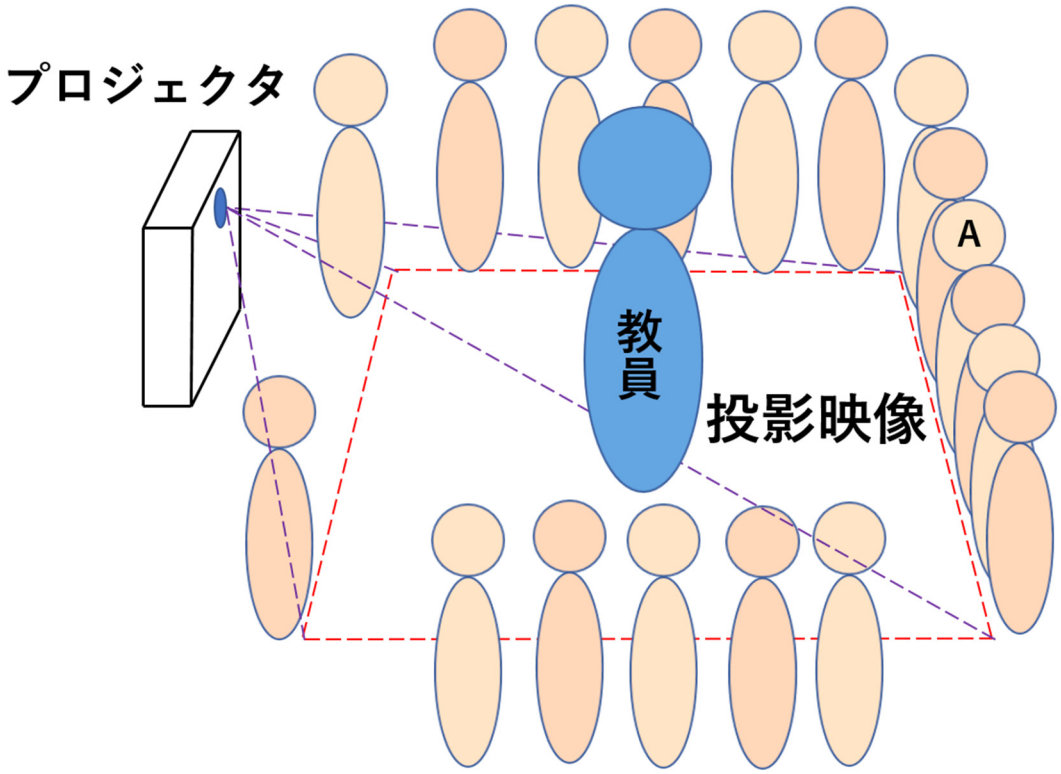


図 1-5 床に映像を投影し授業を行っている様子

1.2.1.1 類似研究:「類似研究「2 台のプロジェクタを用いた Interactive Floor の構築」

赤坂総司

ここでは、小西由香理氏らによる人物の認識およびプロジェクションマッピングを行えるシステムを紹介する。2つのプロジェクタの投影範囲をフィールドとし、フィールド内の人物の位置を Kinect で認識する。その人物から見て3D に見える映像を床面にマッピングすることで、動的な3D トリックアートを実現することを目的としている。

この研究では、2つのプロジェクタから投影した映像の重畳によって、光の遮蔽で映像が消える問題を回避している。これを参考にし、1.2.1 項で述べた問題を解決する。

この研究では、投影領域を黒のテープでマーキングし、Kinect とプロジェクタを予め天井付近に設置しているため、使用できる環境に制限がある。そこで、我々は Kinect と 2 台のプロジェクタの相対位置を固定し、移動可能なハードウェアを開発することで様々な場所で Interactive Floor を使用できるようにする。

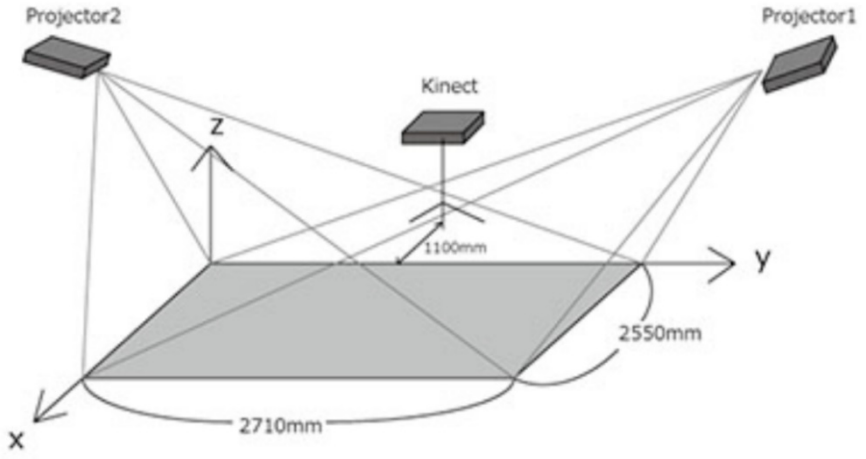


図 1-6 Interactive Floor の概略図【出典:[7]】

1.3 生徒参加型授業の効果 *新海大志*

本節では 1.1 節で触れた生徒参加型授業の効果について解説する。

辻らの「アクティブラーニングの学習効果に関する検証」【出典:[8]】、「アクティブラーニングの学習効果に関する検証(2)」【出典:[9]】では、アクティブラーニング形式の授業での理解度が座学形式での授業の理解度を上回る結果になったと述べられている。これは学習者同士の議論や発表により、学習者の自学自習行動が促進されたためであるとされている。そのため、我々は生徒参加型授業が学習意欲や学習効果の向上を実現できると考える。

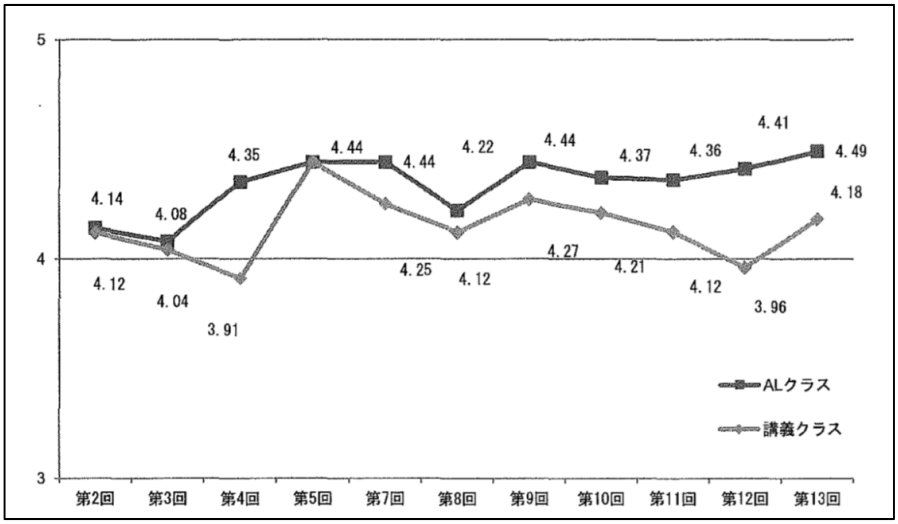


図 1-7 アクティブラーニングと座学の授業理解度【出典:[8]】

1.3.1 類似研究:「Kinectを用いたARによる鏡像シミュレーション教材の活用 −虚像の理解を促す指導法の検討− 」 *赤坂総司*

生徒参加型授業に関連し、中野博幸氏らによるARとKinectを組み合わせた鏡像シミュレーション教材を紹介する。イメージすることの難しい虚像と鏡像をコンピュータ上でシミュレーションすることにより、学習者に現実の現象と概念理解を結び付け鏡像に対する理解の向上を目的としている研究である。図 1-8 に示すように、Kinect で人間の骨格座標を取得し、大型ディスプレイを用いることで骨格情報をリアルタイムでコンピュータのカメラ画像上に表示している。

我々は生徒が参加し体験できるという点、Kinect とディスプレイを使用したインタラクティブな教育コンテンツ作成の 2 点を参考にした。また、この研究では映像のみで数式の表示がなかった。そのため、我々は映像と数式の両方を表示することにした。

図 1-8 Kinectを用いたARによる胸像シミュレーション教材の概略図【出典:[10]】

1.4 本研究の目的 *大原広暉*

1.1 節から 1.3 節より、本研究の目的は以下の条件にあてはまる〈教育効果の高い教育支援システム〉を開発することと定める。

1. 2 台のプロジェクタを使用して、床に映像を投影する。
2. 生徒参加型の教育コンテンツを提供する。
3. ユーザーと教育支援システムがインタラクティブである。

1 の条件を解決するために、2 台のプロジェクタを搭載できるハードウェアを作成することが要求される。

2 の条件を解決するために、生徒が動き回って教科書の内容を理解できるアプリケーションを開発することが要求される。今回は、物理現象の「水面波の干渉」を支援対象とする。

3 の条件を解決するために、ユーザーがシステムにコマンドを送り、システムが応答することが要求される。そのために人のジェスチャーによって、投影映像が変化するようにする。

また、1 の条件に 3 の条件のインタラクティブ性をあわせ持つシステムを本研究では Interactive Floor Interface と呼ぶ。Interactive Floor Interface と、その上で動作する教育コンテンツを開発することが本研究の目的であると言い換えることができる。

2 Interactive Floor Interface

2.1 概要

大野智哉

Interactive Floor Interface(以下、IFI)の概要を図 2-1 に、フローチャートを図 2-2 に示す。IFI の入力はユーザーの動作とし、出力は情報を投影することとする。まず、プロジェクタによって投影されている映像の範囲内にて、ユーザーが提示されている情報に合わせた動きをする。次に、その動きを Kinect によって取得し、その解析結果を用いて提示する情報をリアルタイムに更新する。

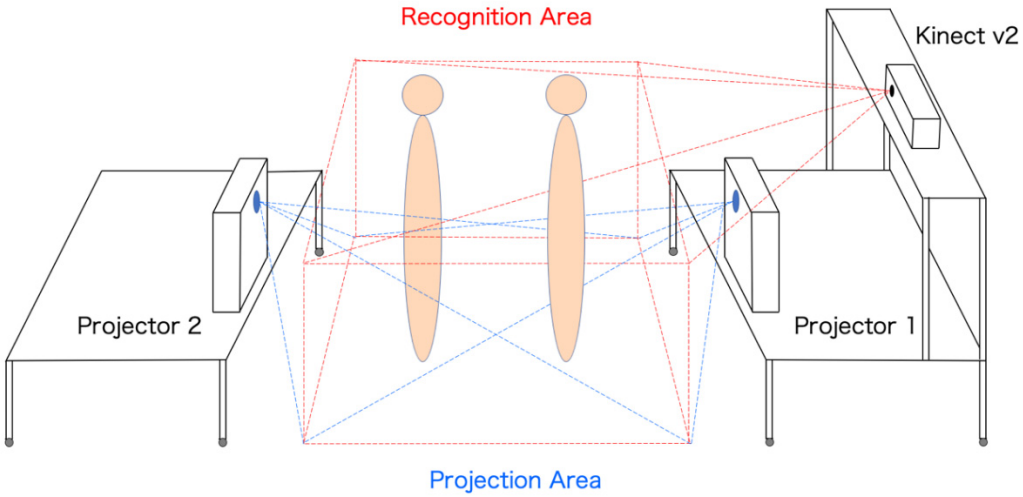


図 2-1 IFI の概要

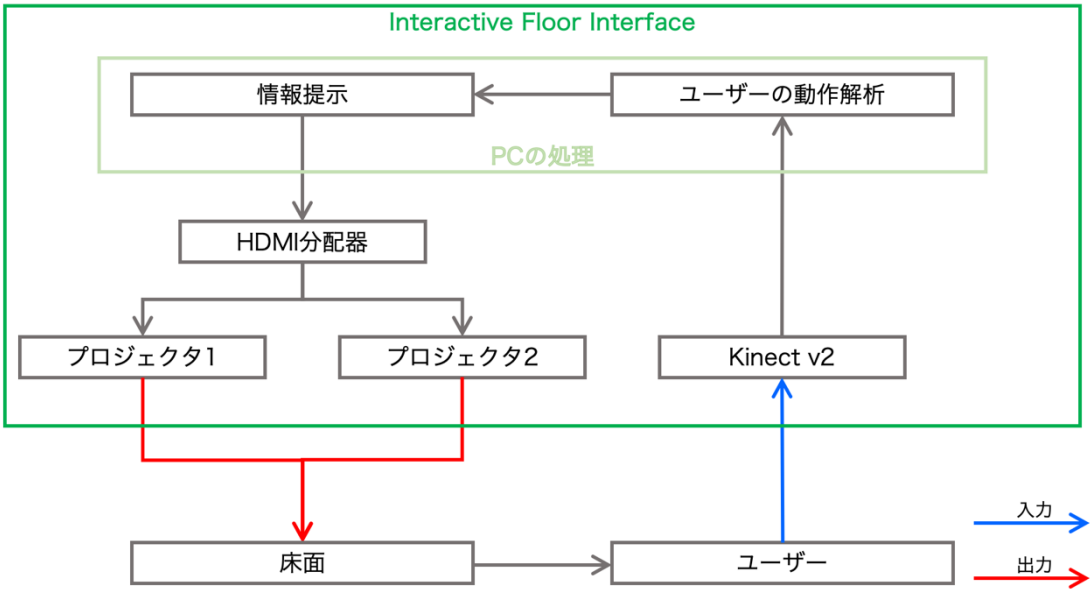


図 2-2 IFI のフローチャート

2.2 搭載機器 *大野智哉*

2.2 節以下では IFI に搭載されている Kinect v2、プロジェクタ、デスクトップ型 PC、HDMI 分配器の詳細を述べる。

2.2.1 Kinect v2

モーションセンサーとして Microsoft 社製の Kinect v2(以下、Kinect)を使用する。カラー映像の解像度は 1920×1080pixel、人物の検出範囲は 0.5～4.5m であり、最大 6 人まで検出可能である。詳しい仕様は表 2-1 に示す。

また、Kinect を PC で利用するためには SDK である「Kinect for Windows SDK 2.0」*1 をインストールする必要がある。さらに、Unity 内で Kinect を利用するためには個別のプロジェクト内にパッケージ「Unity Pro package」*2 をインストールする必要がある。

図 2-3 Kinect【出典:[11]】

表 2-1 Kinect の仕様

色	解像度	1920×1080
	fps	30fps
深度	解像度	512×424
	fps	30fps
検出可能人数		6 人
関節		25 関節/人
人物の検出範囲		0.5～4.5m
映像が取得可能角度	水平	70 度
	垂直	60 度

---

*1 https://www.microsoft.com/en-us/download/confirmation.aspx?id=44561

*2 https://go.microsoft.com/fwlink/p/?LinkId=513177

2.2.2　プロジェクタ

床に映像を投影するために EPSON 社製ビジネスプロジェクタ EB-685W を 2 台使用する。このプロジェクタは超短焦点モデルであるため、縦置きで設置することで床に映像を投影することができる。詳しい仕様は表 2-2 に示す。

図 2-4 プロジェクタ EB-685W

表 2-2 プロジェクタ EB-685W の仕様

スクリーン解像度	1280×800
焦点距離:f	3.7mm
質量	5.7kg
サイズ(W×D×H)	367×400×149mm

2.2.3　デスクトップ型 PC

表 2-3 に使用したデスクトップ型 PC の仕様を示す。 演算負荷が大きいアニメーション等も滑らかに再生できるようノート型 PC ではなく高性能な GPU を搭載したデスクトップ型 PC を使用する。

図 2-5 使用したデスクトップ型パソコン

表 2-3 デスクトップ PC の仕様

OS	Windows 10
CPU	Intel® CORE™ i7-8700
RAM	16GB
GPU	NVIDIA GeForce GTX 1070 Ti

2.2.4　HDMI 分配器

2 台のプロジェクタから同じ映像を投影するため、1 台の PC の映像を複数のデバイスに同時に出力できる HDMI 分配器を使用する。使用したのは GREEN HOUSE 社の GH-HSPC4-BK である。

図 2-6 HDMI 分配器 GH-HSPC4-BK【出典:[12]】

2.3 IFI を用いた教育支援システム　*大原広暉*

1 章で述べた研究目的と類似研究を踏まえ、我々は IFI を用いた教育支援システム（本文中では以下、本システム）を開発する。本システムは

- IFI
- 教育アプリケーション：〈水面波の干渉アニメーション〉（以下、本アプリケーション）

から構成される。

IFI の概要は 2.1 節に述べたとおりであり、本アプリケーションについては 2.3.1 項で述べる。また、IFI の詳細・開発方法については 2.4・2.5 節で述べ、本アプリケーションについては 2.6 節で述べる。

2.3.1　教育アプリケーション:〈水面波の干渉アニメーション〉の概要　*大原広暉*

様々な科目があるうち特に理系科目は、教員は ICT を活用して教えるほうが口頭・板書のみで教えるよりも生徒に伝えやすいと考える。たとえば、ドップラー効果を教えるときに、救急車の例を上げても生徒全員が理解できるとは限らない。一方、ICT を活用すればあらかじめ用意しておいたデジタルコンテンツを使って生徒に教えられるため、生徒たちは理解しやすいと考えられる。

そこで、本システムの教育アプリケーションの内容として〈水面波の干渉〉を選んだ。なぜなら、〈現実の波〉と〈テキスト上の波〉には〈ギャップ〉があると考えたからである。〈現実の波〉は1つの波紋が連続的に広がっていく様子を見て理解できる。それに対し、〈テキスト上の波〉は波源から発生するすべての波が固定で表示されている。これでは双方のつながりがわかりにくい。この〈ギャップ〉を本システムで埋めることを目指す。

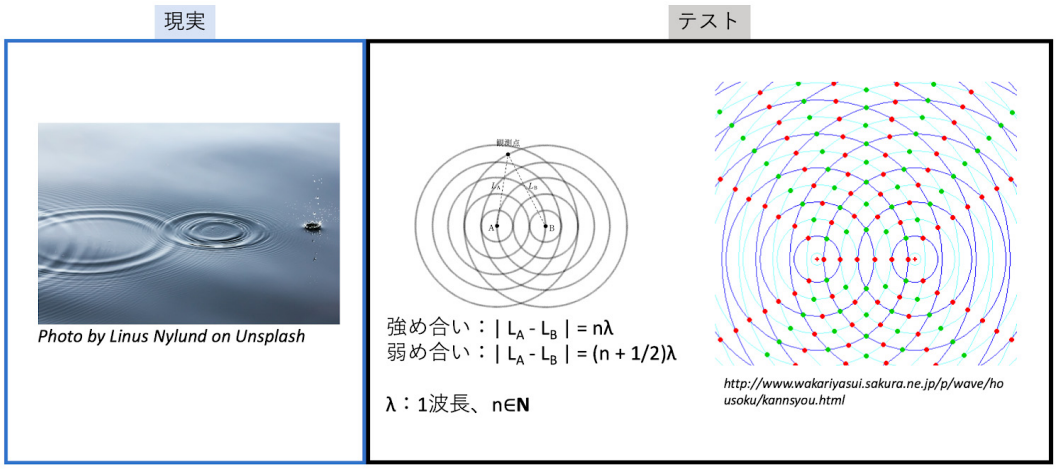


図 2-7 〈現実の波〉と〈テキスト上の波〉の違い

具体的には、本アプリケーションの利用者（以下、ユーザー）は、

- 水面波の強め合いの公式：$|LA - LB| = n\lambda$
- 水面波の弱め合いの公式：$|LA - LB| = (n + \frac{1}{2})\lambda$
- 干渉縞

について学習できることを目標とする。

1.3.1 項で解説した類似研究では、生徒はインタラクティブな教材を使うことで直感的に鏡像と虚像の関係を理解できた[*1]。しかし、数式的な理解をできた生徒は少なかったと結論づけていた。

そのため、上記の目標を達成するには、生徒が公式を理解できるようなアプリケーションとなるよう工夫しなければならない。

本アプリケーションのモードは Distance、Real と Moiré の 3 種類がある。それぞれの詳しい機能については 2.6 節にて述べる。

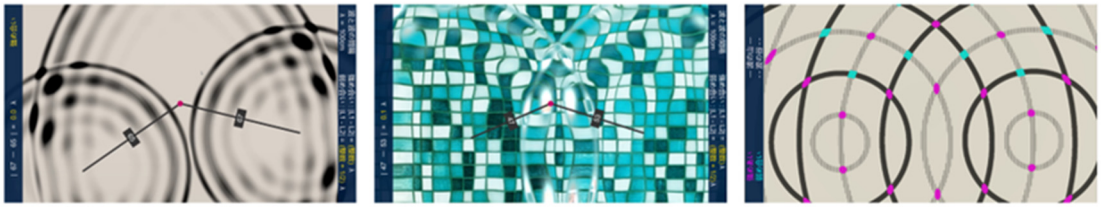


図 2-8 本アプリケーションに搭載されている 3 種類のモード

---

*1 1.3.1 項を参照

2.4 IFI のプログラム実装 *赤坂総司*

本節では IFI に組み込んでいる以下のプログラムの機能・開発方法について述べる。

1. 人のジェスチャー認識
2. キャリブレーション

人のジェスチャー認識の機能を実装することで IFI におけるインタラクティブな操作を実現できる。詳細については 2.4.1 項で述べる。

キャリブレーションの機能を実装することで Kinect における人の位置座標と、プロジェクタにおける床上の位置座標を合わせることができる。詳細については 2.4.2 項で述べる。

2.4.1　人のジェスチャー認識　　*赤坂総司，大野智哉*

我々は IFI にジェスチャー認識の機能をもたせることで、IFI にインタラクティブ性を導入する。それにより、ユーザーがジェスチャーを行うことによって、提示する情報の内容を変化させることができる。

図 2-9 に示すのが、ユーザーがジェスチャーをした時の場合の IFI のフローチャートである。ジェスチャーを行うことにより、2.1 節のフローチャートにて示されている「情報提示」にて提示されている情報が変化する。

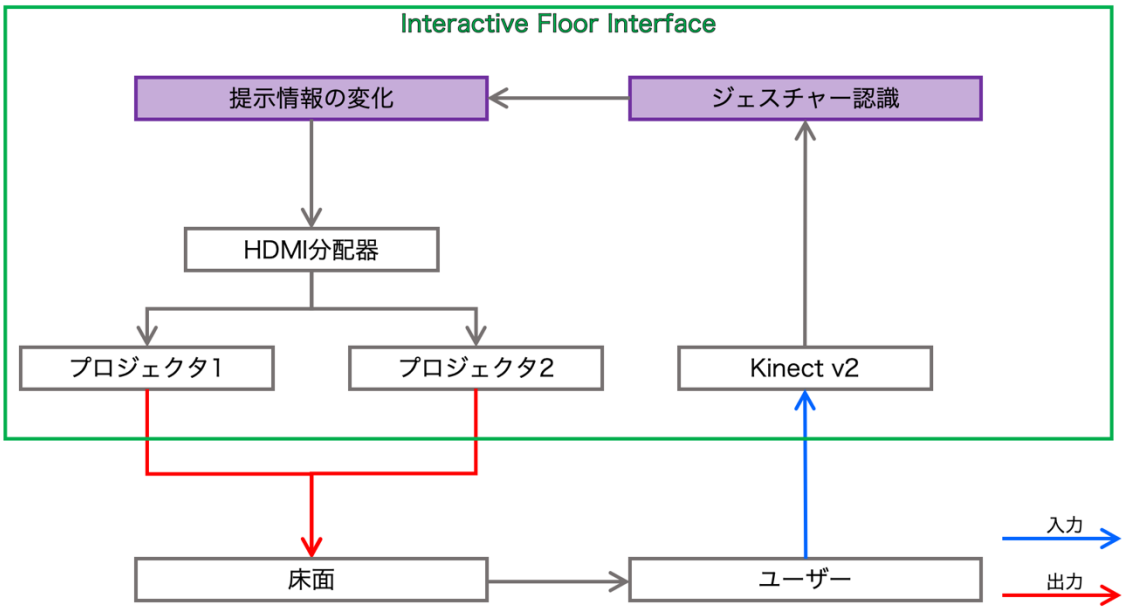


図 2-9　入力をジェスチャーとした時の IFI

2.4.1.1 識別器の作成 *大野智哉*

前述の通り、IFI にジェスチャー認識の機能を持たせるためには、IFI に含まれる Kinect の機能を用いる。Kinect でジェスチャーの認識を行うためには、Kinect の SDK 付属の「Kinect Studio」と「Visual Gesture Builder」(以下、VGB)を用いて識別器を生成すればよい。以下 2 つの手順を踏むことで、識別器を作成できる。

I. SDK 付属の Kinect Studio を用いて認識したいジェスチャーを含むクリップを作成する。
II. VGB を用いてクリップのジェスチャー部分にラベルを付けて識別器を学習させる。

まず、I について説明する。図 2-10 に示されているのが Kinect Studio の画面であり、センサーからの距離によって色が表示されているのが深度情報、人の体に線で表示されているのが骨格情報である。また、識別器を作成するためには画面左側にある

- Nui Body Frame(骨格情報)
- Nui Depth(深度情報)
- Nui IR(赤外線情報)
- Nui Sensor Telemetry(録画をするために必要な内部データ)

のデータが必要であるためチェックを入れる。そして、左上の赤いボタンを押し、Kinect を使ってジェスチャーを録画する。この際、〈認識したいジェスチャーを行っている状態と行っていない状態の両方を含む〉クリップを撮影する。なお、撮影されたクリップは xef ファイルとして保存される。

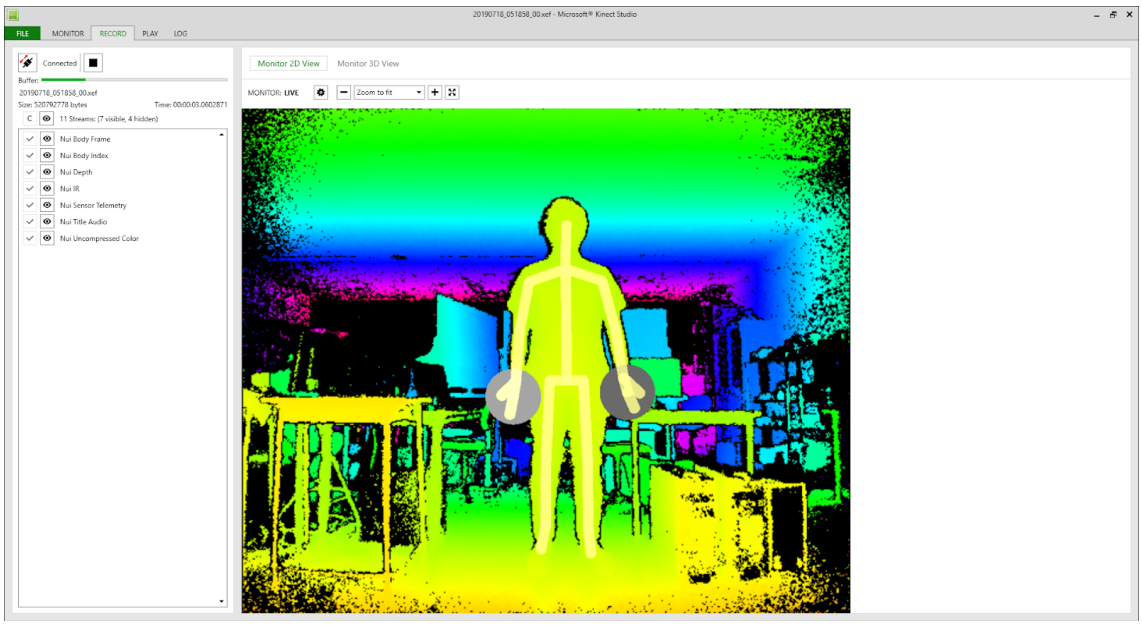


図 2-10 Kinect Studio の画面

次に、Ⅱについて説明する。まず、VGB を起動し、左上の[File]→[New Solution]で新しいソリューションを作成する。次に、作成したソリューションの名前を右クリックし、[Create New Project]を左クリックするとジェスチャーの設定画面が表示される（図 2-11・図 2-12）。本論文では例としてソリューション名を Test としている。

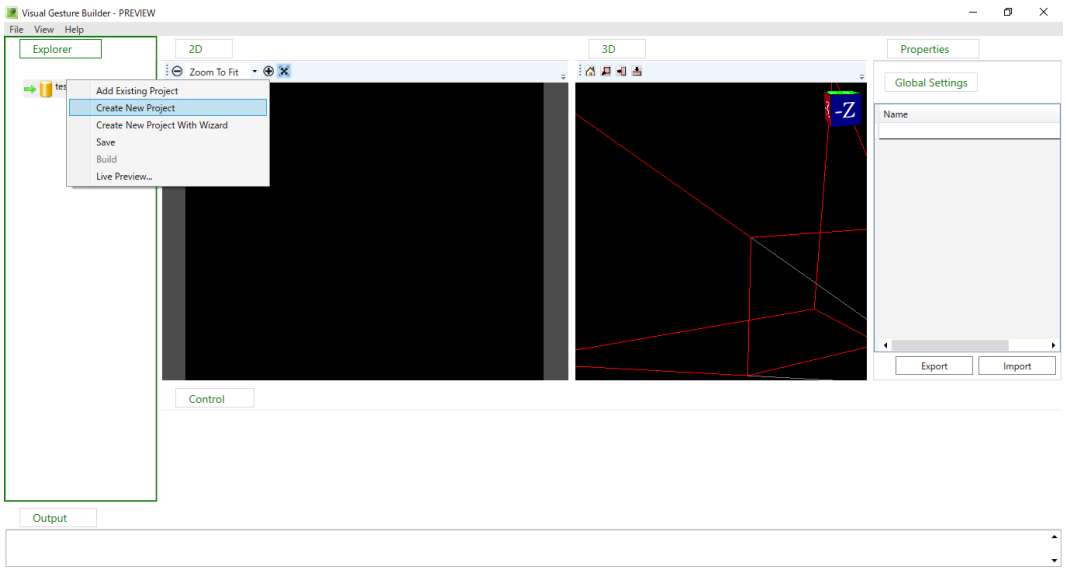


図 2-11 ソリューション作成後の VGB の画面

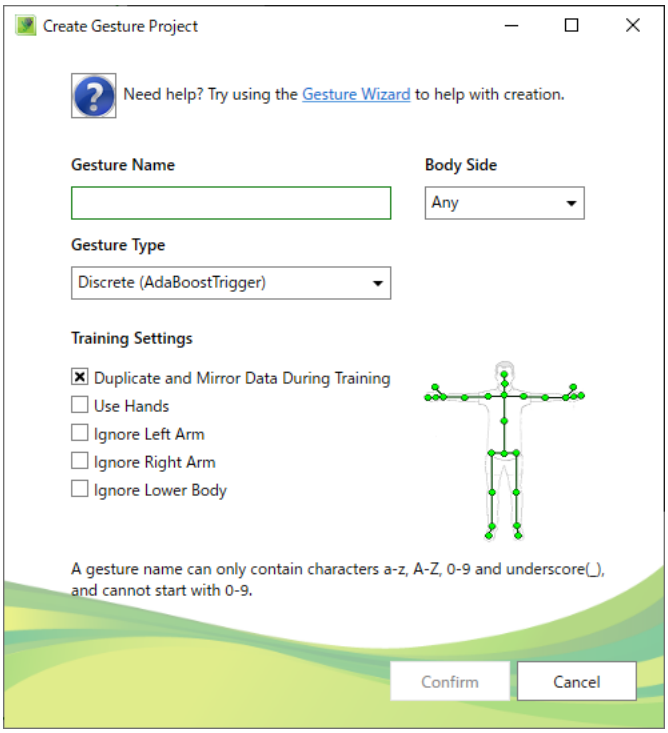


図 2-12 ジェスチャーの設定画面

図 2-12 のジェスチャーの設定について項目ごとに説明する。

- Gesture Name にはジェスチャーの名前を英数字で入力する。
- Body Side は[Any]・[Left]・[Right]から選択する。これはジェスチャーが左右の判別を必要とする場合は[Left]か[Right]どちらかを選択する。例えば〈手を上げる〉ジェスチャーを認識したい場合、上げる手を右左どちらかを指定したいときは[Right]・[Left]、どちらの手でもいいときは[Any]を選択する。
- Gesture Type は[Discrete]・[Continuous]から選択する。これはジェスチャーの途中の動作が必要な場合は[Continuous]を選択する。例えば、〈立つ〉とジェスチャーを認識したい場合、座っている状態から立ち終わるまでの過程を認識するときは[Continuous]、立っている状態のみを認識したいときは[Discrete]を選択する。また、[Discrete]タイプの認識結果が真偽値で、[Continuous]タイプの結果は進捗状況が実数値で帰ってくる。
- Training Settings は大きく分けて[Duplicate and Mirror Data During Training]・[Use Hands]・[Ignore Left Arm / Right Arm/ Lower Body]の 3 つがある。

  ① [Duplicate and Mirror Data During Training]はジェスチャーが左右対称の場合選択する。
  ② [Use Hands]はジェスチャーを認識する時に手の状態も条件に含める場合選択する。手の状態とはグー・パー・チョキの3種類のことを指す。
  ③ [Ignore Left Arm / Right Arm/ Lower Body]はジェスチャーを認識する時に左手/右手/下半身の動きを無視する場合選択する。例えば、〈手を上げる〉ジェスチャーを認識したい際に下半身の動きを認識の条件としない場合は[Ignore Lower Body]を選択する。

本論文では例として Gesture Name を RaiseRightHand、Body Side を Right、Gesture Type を Discrete とし、Training Settings は Ignore Lower Body のみを選択した。なお、1つソリューションに対して複数のジェスチャーを設定することが可能である。

ジェスチャーの設定が終わり、Confirm をクリックするとソリューションの下に設定したジェスチャーの名前が表示される。次に、ジェスチャーの名前を右クリックし[Add Clip]を左クリックし、I で作成したクリップ（xef ファイル）を選択し、追加する（図 2-13）。

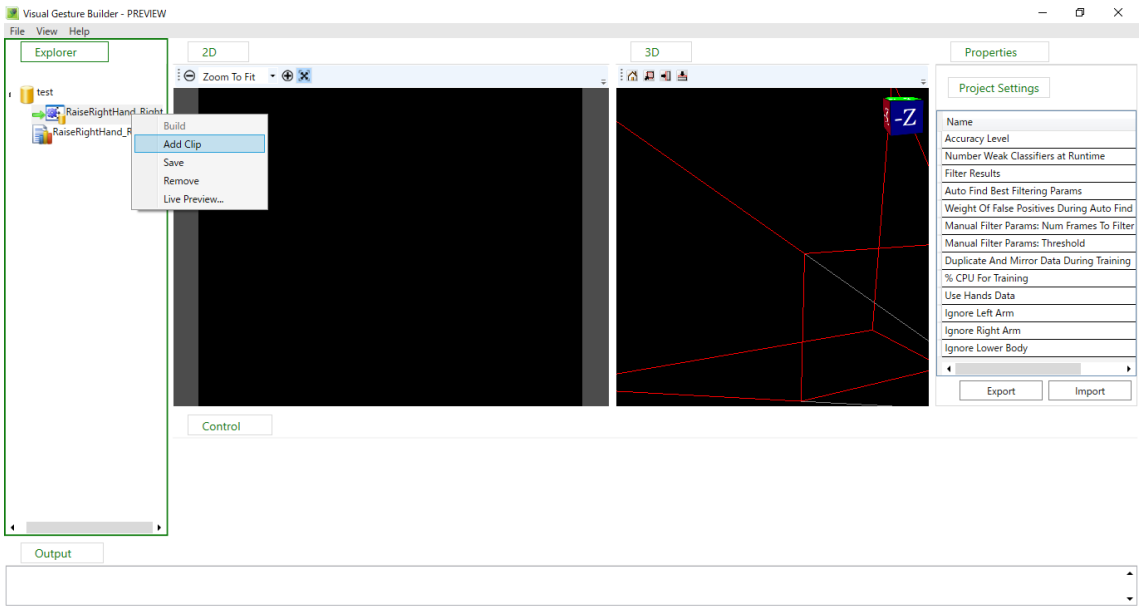


図 2-13 ジェスチャーの設定を終えた後の VGB の画面

クリップを追加し、追加したクリップを左クリックするとVGBの画面が図 2-14のようになる。次に、画面下部の Control タブをマウスとキーボードで操作し、クリップ内のジェスチャー部分にラベルを付ける。Control タブはシークバーのようになっており、マウスにてクリップ内のジェスチャー部分の開始地点をクリックする。次に、Shift キーと右矢印キーを押してラベル付けを行う。両キーを押し続けている間はジェスチャーが true になり続け、図 2-15 のように色が変わる。そして、ジェスチャー部分の終了地点までラベル付け終了後 Enter キーを押してラベル付けを確定する。なお、ラベル付けされていない箇所は false として扱われる。また、ジェスチャーを Continuous タイプとした場合はラベル付けを true で行わず、0～1 の値で行う。ジェスチャーの開始地点を 0 とラベル付けし、終了地点を 1 とする。

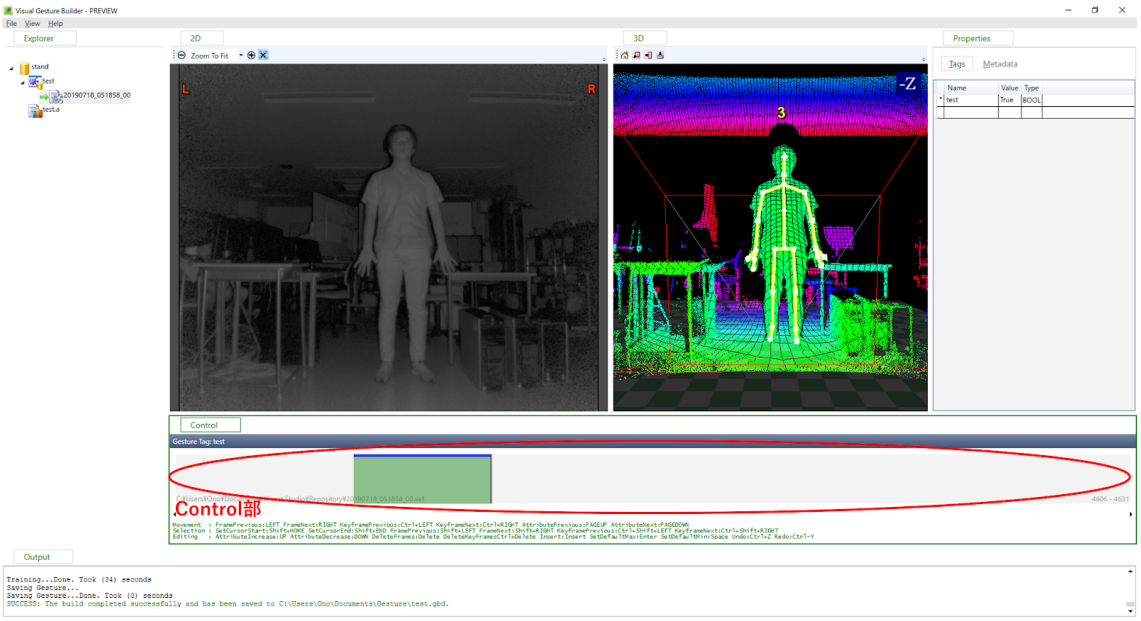


図 2-14 クリップ追加後の VGB の画面

図 2-15 ジェスチャー部分のラベル付け（Control タブ拡大）

ラベル付け終了後、ソリューションの名前を右クリックして[Build]を左クリックすると識別器（gbd ファイル）が作成される。以上が識別器の作成法である。

2.4.1.2 使用したジェスチャー *大野智哉*

我々は、ユーザーにとってわかりやすいと考えられる〈右手を上げる〉・〈左手を上げる〉・〈両手を上げる〉・〈両手を水平に上げる〉の計 4 つのジェスチャーに対する識別器を作成した。なお、本インターフェイスにおいては、ジェスチャーを提示する情報を変化させるスイッチとしたため、4 つ全てのジェスチャーを Discrete タイプとした。

また、水面波の干渉アニメーションにおける機能との対応は以下の通りである。水面波の干渉アニメーションにおける機能については 2.6 節にて説明する。

波の周波数の変更 ⇒ 右手を上げる

Real モードへの切り替え ⇒ 左手を上げる

Moiré モードへの切り替え ⇒ 両手を上げる

波の一時停止 ⇒ 両手を水平に上げる

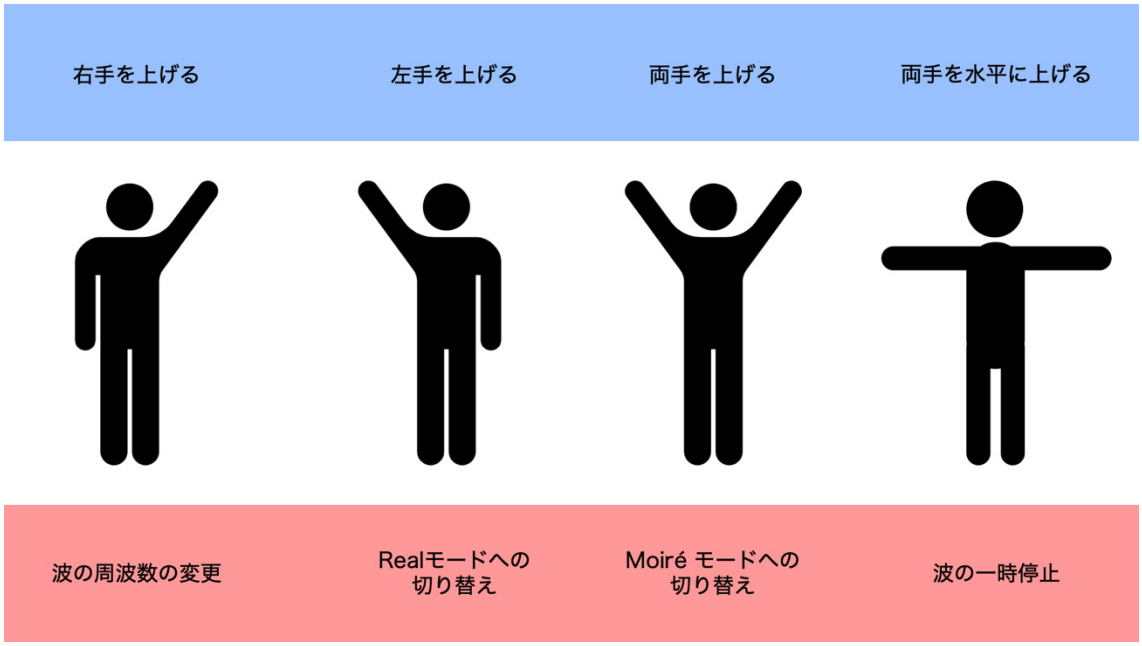


図 2-16 各機能とジェスチャーの対応図

2.4.1.3 ジェスチャーを認識する条件 *大野智哉*

本項では IFI がジェスチャーを認識する条件について述べる。なお、IFI では Discrete タイプのジェスチャーのみを用いるため、以下の説明は Discrete タイプのジェスチャーについてのものとする。

識別器はジェスチャーを認識したかを bool 型(true / false)で出力し、ジェスチャーを認識した場合は Confidence という値を float 型(単精度浮動小数点数)の 0～1 の値で出力する。なお、Confidence とは識別器がジェスチャーを認識した際の確信度である。この仕様の問題として、識別器は Confidence の数値が低い場合、つまり登録されているジェスチャーと似た動きをしている場合でもジェスチャーを認識してしまうことが挙げられる。例えば、〈頭を掻く〉動きが〈右手を上げる〉ジェスチャーだと認識されてしまう可能性がある。

そこで、我々はジェスチャーを認識する条件を〈所定のフレーム数F_0以上連続して閾値C_0以上の Confidence の値が出力された場合〉とした。なお、以降所定のフレーム数F_0は 12 とし、Confidence の閾値C_0は 0.7 とした。これらの値を決めた方法は次項にて述べる。

認識する条件のイメージを図 2-17 に示す。なお、Confidence の値をCとし、Confidence の値が$C_0(=0.7)$以上の時の経過フレームをFとする。CとFを用いて条件を表すと、$C \geqq C_0$の時にFが毎フレーム 1 加算され、$F = F_0(=12)$の時にジェスチャーが認識される。また、$C < C_0$の時は$F = 0$となる。

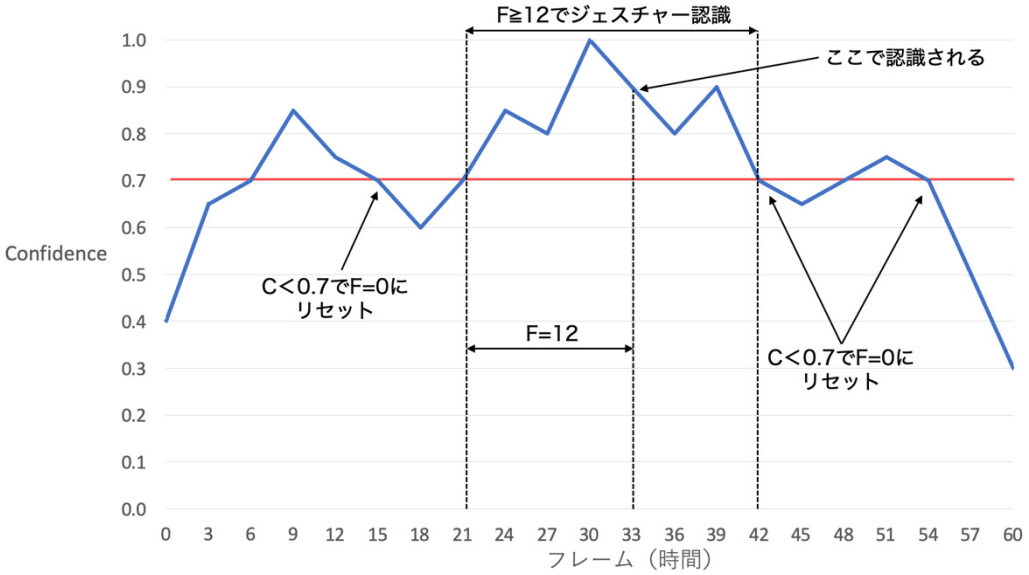


図 2-17 認識条件のイメージ

以下から図 2-17 を例としてジェスチャーを認識する条件を説明する。

まず、7 フレームの時に、$C \geqq C_0$となるためFの値が 1 フレーム毎に 1 加算される。そして、$C \geqq C_0$であるのは 15 フレーム目までであるため、16 フレーム目にFは 0 にリセットされる。15 フレーム目におけるFの値は 15-6＝9 であり、$F < F_0$であるためジェスチャーは認識されない。

次に、図のように 21 フレーム目から 42 フレーム目まで$C \geqq C_0$が満たされるので、その間Fが加算され続ける。32 フレーム目にFの値は 12 となり、$F \geqq F_0$となるため、ジェスチャーが認識される。

最後に 48 フレームから 54 フレームの場合は$F < F_0$であるため、ジェスチャーは認識されない。

2.4.1.4 実験 *赤坂総司*

ここでは最適なC_0とF_0を決定するための実験を行う。既に述べたように、本システムにおいてジェスチャーを認識する条件は、〈フレーム数F_0以上連続して閾値C_0以上の Confidence の値が出力された場合〉である。C_0は誤認識を防ぐため大きい数値、F_0は認識までの時間を短くするため小さい数値であることが要求される。

さらに、識別器にジェスチャーを登録していない人に対しても、識別器がジェスチャーを登録している人と同様の認識精度を保てるかを検証した。

以上より、本実験の目的は次の 1・2 の 2 つである。

1. システム上で使用者に過度の負担をかけずかつ認識精度が高いC_0とF_0の値を決定する
2. 識別器にジェスチャーを登録している被験者（以下、登録者）と登録していない被験者（以下、非登録者）の違いが認識精度に影響を及ぼすかの検証を行う

本実験を行う上で、使用者に過度の負担を与えないためにF_0を 3 から 30 の範囲で実験する。本実験は〈登録者と非登録者各 1 名ずつ計 2 名でジェスチャーを 10 回×3 セット行い、認識精度（認識成功回数/30）を記録する〉ことで行う。本実験は I からⅦの手順で行う。

I. 4 つのジェスチャー*1 が登録されている識別器を用意する
II. 初期値としてC_0の値を 0.9、F_0の値を 15 に設定し、実験を行う
III. C_0の値を 0.1 減らした値に設定し、実験を行う
IV. C_0の値が 0.1 の実験が終わった後、実験結果よりC_0の値を決定する
V. IV で決定したC_0の値を用い、F_0の値を 30 に設定して、実験を行う
VI. F_0の値を 3 減らした値に設定し、実験を行う
VII. F_0の値が 3 の実験が終わった後、実験結果よりF_0の値を決定する

---

*1 2.4.1.2 項を参照

図 2-18 から図 2-21 はC_0の値を決める実験結果である。赤で囲われたグラフは、ジェスチャー登録者と非登録者それぞれにおいて、認識精度が 100%かつC_0が最も大きくなったグラフである。

また、図 2-26 は4つのジェスチャーの平均認識精度である。非登録者の実験結果の中で、赤で囲われたグラフの認識精度が一番高く、C_0の値も高いため〈C_0=0.7〉を本システムで使用する。

右手を上げる		
C_0	登録者	非登録者
	認識精度	認識精度
0.1	96.7	80.0
0.2	70.0	96.7
0.3	90.0	96.7
0.4	86.7	100.0
0.5	100.0	96.7
0.6	100.0	100.0
0.7	100.0	100.0
0.8	100.0	96.7
0.9	100.0	93.3

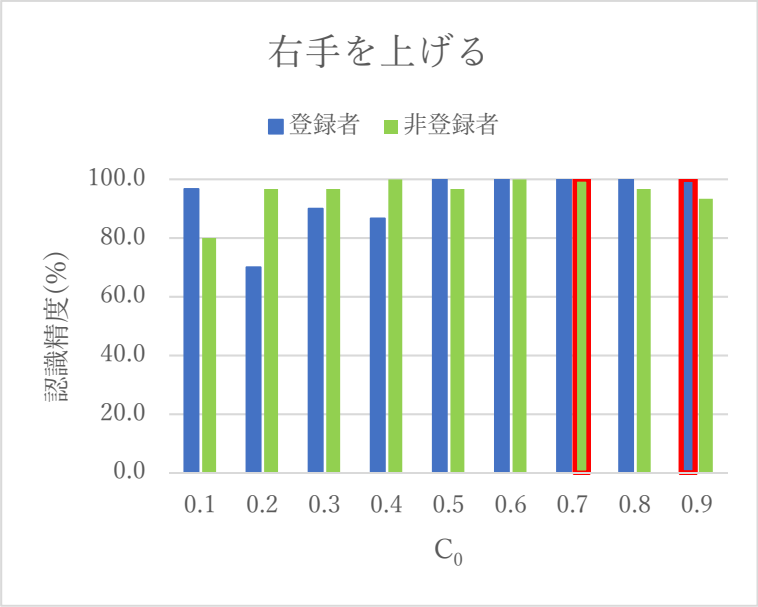


図 2-18 C_0認識精度実験結果

左手を上げる		
C_0	登録者	非登録者
	認識精度	認識精度
0.1	66.7	76.7
0.2	100.0	100.0
0.3	100.0	100.0
0.4	100.0	93.3
0.5	100.0	100.0
0.6	100.0	100.0
0.7	100.0	100.0
0.8	100.0	100.0
0.9	93.3	100.0

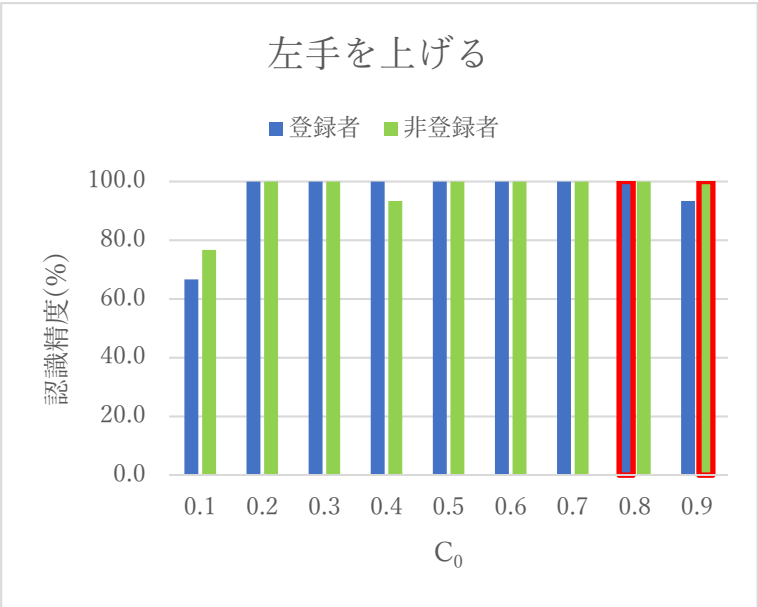


図 2-19 C_0認識精度実験結果

両手を上げる		
C_0	登録者	非登録者
	認識精度	認識精度
0.1	63.3	43.3
0.2	100.0	90.0
0.3	100.0	100.0
0.4	100.0	100.0
0.5	100.0	96.7
0.6	100.0	100.0
0.7	93.3	100.0
0.8	100.0	100.0
0.9	80.0	96.7

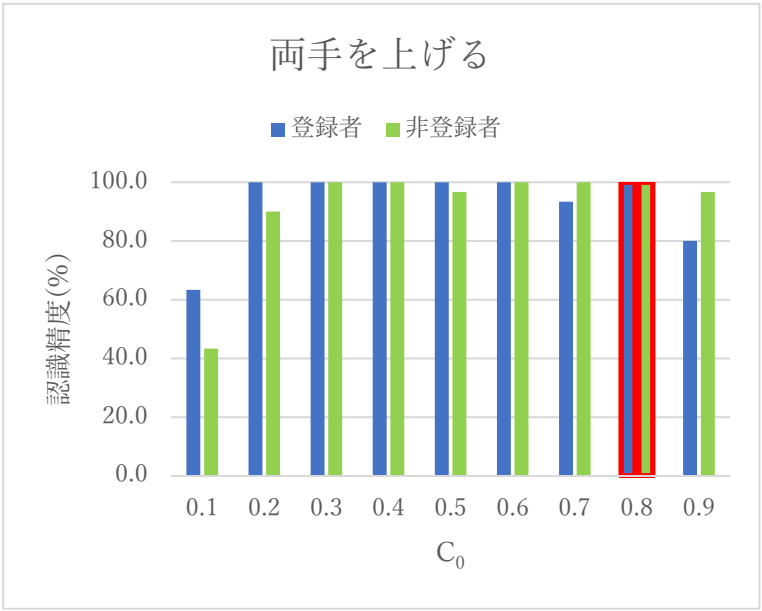


図 2-20 C_0認識精度実験結果

両手を水平に上げる		
C_0	登録者	非登録者
	認識精度	認識精度
0.1	93.3	73.3
0.2	100.0	100.0
0.3	100.0	100.0
0.4	100.0	96.7
0.5	80.0	96.7
0.6	100.0	100.0
0.7	100.0	100.0
0.8	100.0	100.0
0.9	100.0	100.0

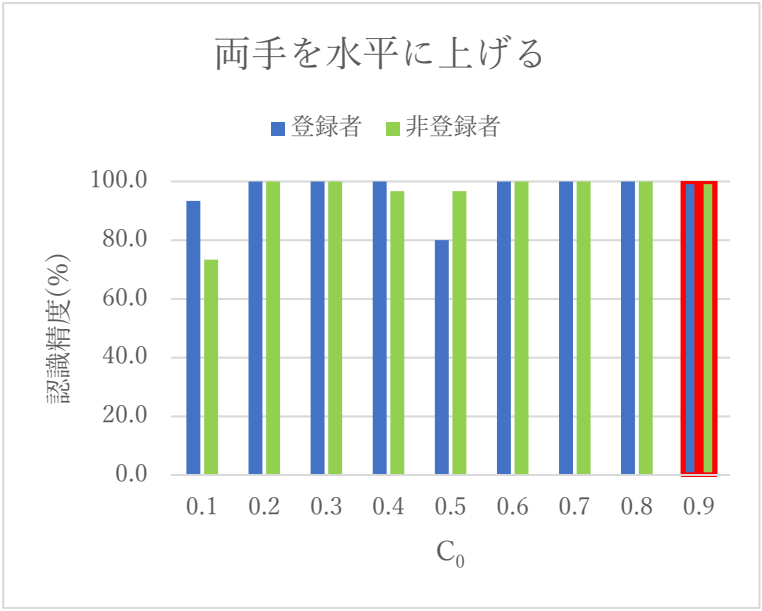


図 2-21 C_0認識精度実験結果

図 2-22 から図 2-25 はF_0の値を決める実験結果である。黒で囲われたグラフは、ジェスチャー登録者と非登録者それぞれにおいて、ジェスチャー認識精度が 100%を超えているグラフである。F_0が小さいことで、誤認識の回数が増えるためこの現象が起こる。

図 2-27 は4つのジェスチャーの平均認識精度である。認識精度が 100%を超えてしまったグラフを黒で囲っている。ジェスチャーの登録者と非登録者の両方で認識精度が 100%となるF_0の最小値は$F_0 = 9$であった。安全のためその 1 つ上の値である〈$F_0 = 12$〉を本システムで使用する。

右手を上げる		
F_0	登録者	非登録者
	認識精度	認識精度
3	123.3	123.3
6	103.3	103.3
9	100.0	100.0
12	100.0	100.0
15	100.0	100.0
18	100.0	100.0
21	93.3	100.0
24	100.0	93.3
27	33.3	83.3
30	3.3	0.0

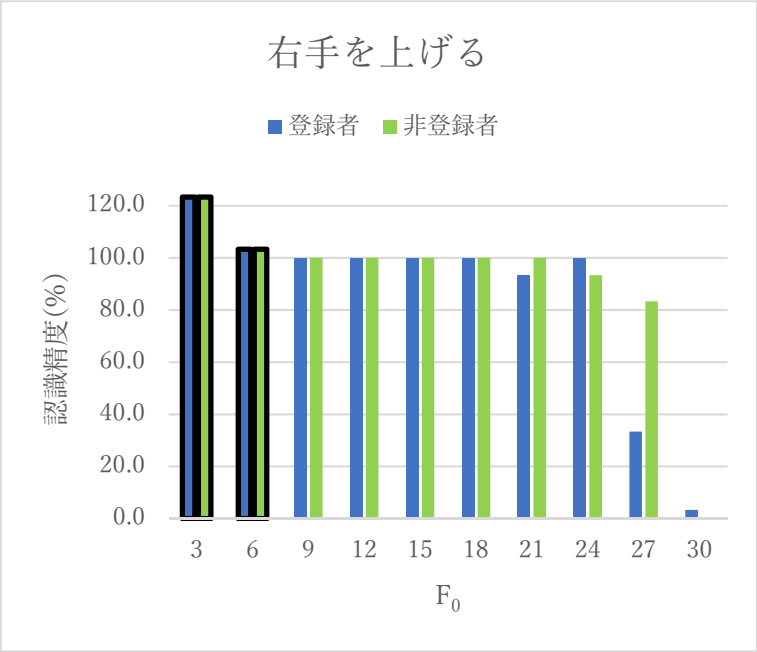


図 2-22 F_0認識精度実験結果

左手を上げる		
F_0	登録者	非登録者
	認識精度	認識精度
3	100.0	156.7
6	100.0	126.7
9	100.0	100.0
12	100.0	100.0
15	100.0	100.0
18	100.0	100.0
21	83.3	100.0
24	93.3	100.0
27	50.0	80.0
30	6.7	0.0

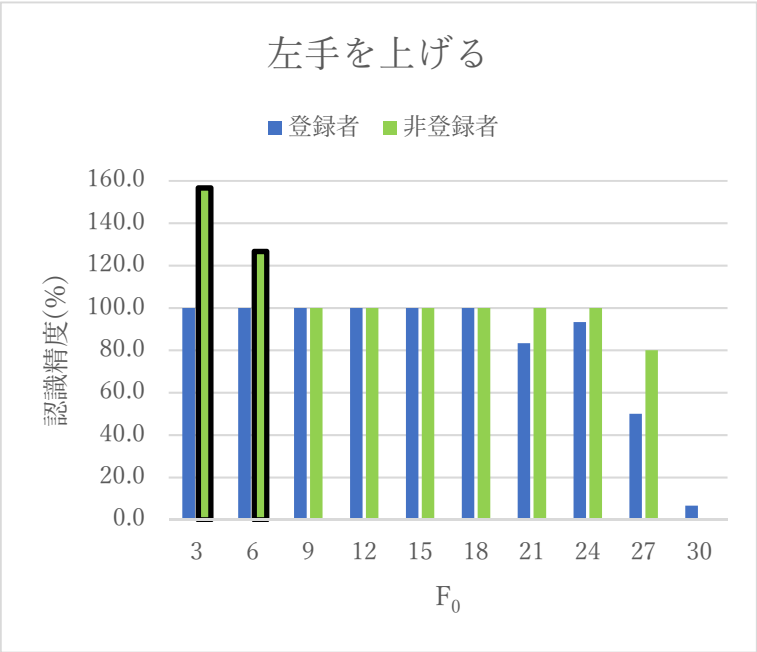


図 2-23 F_0認識精度実験結果

両手を上げる		
F_0	登録者	非登録者
	認識精度	認識精度
3	103.3	103.3
6	100.0	100.0
9	100.0	100.0
12	100.0	100.0
15	93.3	100.0
18	100.0	100.0
21	100.0	100.0
24	86.7	100.0
27	80.0	83.3
30	3.3	0.0

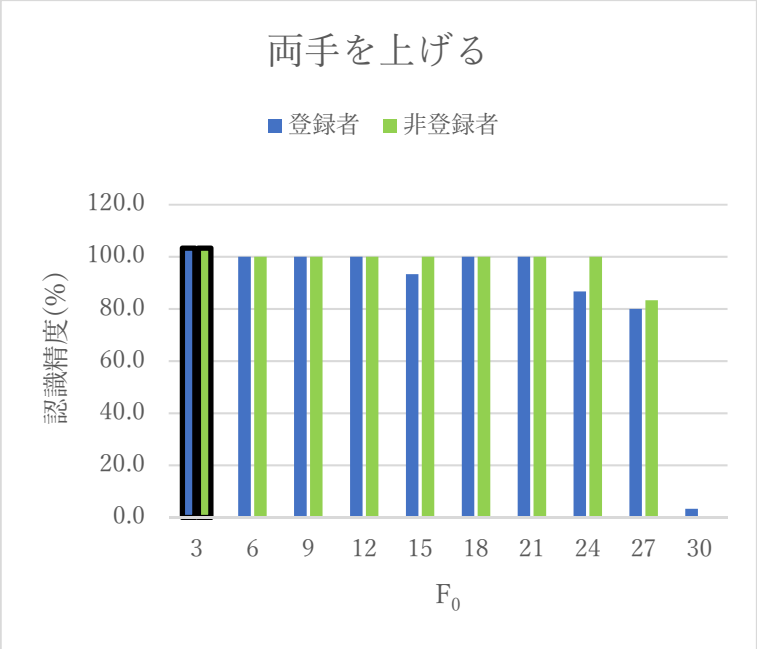


図 2-24 F_0認識精度実験結果

両手を水平に上げる		
F_0	登録者	非登録者
	認識精度	認識精度
3	116.7	100.0
6	100.0	100.0
9	100.0	100.0
12	100.0	96.7
15	100.0	100.0
18	100.0	100.0
21	96.7	100.0
24	76.7	73.3
27	20.0	63.3
30	0.0	0.0

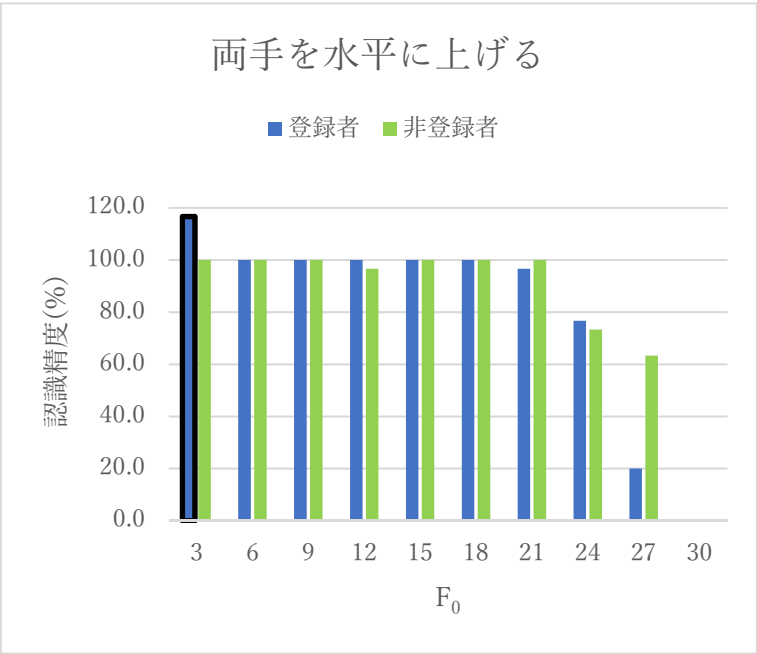


図 2-25 F_0認識精度実験結果

図 2-26 と図 2-27 より登録者と非登録者ではほぼ同じ認識率を出しており、認識精度に影響はなかった。

平均認識精度		
C_0	登録者	非登録者
	認識精度	認識精度
0.1	80.0	68.3
0.2	92.5	96.7
0.3	97.5	99.2
0.4	96.7	97.5
0.5	95.0	97.5
0.6	100.0	100.0
0.7	98.3	100.0
0.8	100.0	99.2
0.9	93.3	97.5

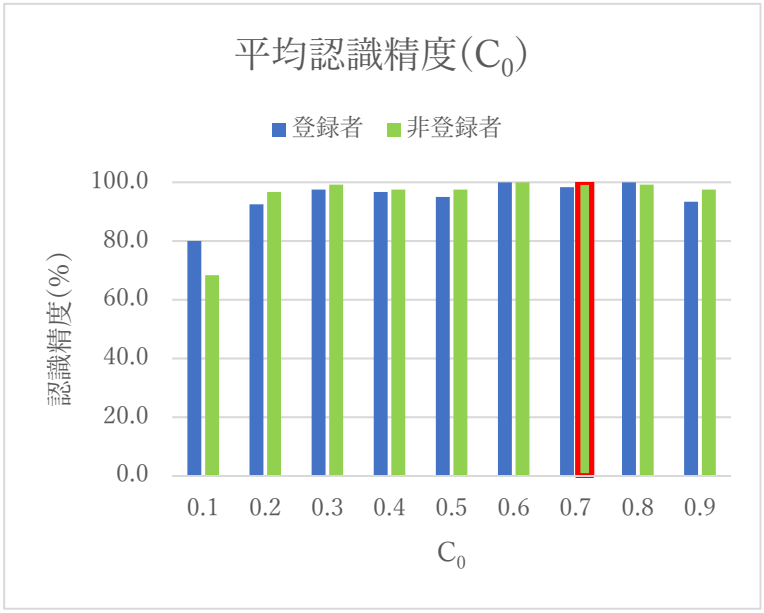


図 2-26 C_0平均認識精度

平均認識精度		
F_0	登録者	非登録者
	認識精度	認識精度
3	110.8	120.8
6	100.8	107.5
9	100.0	100.0
12	100.0	99.2
15	98.3	100.0
18	100.0	100.0
21	93.3	100.0
24	89.2	91.7
27	45.8	77.5
30	3.3	0.0

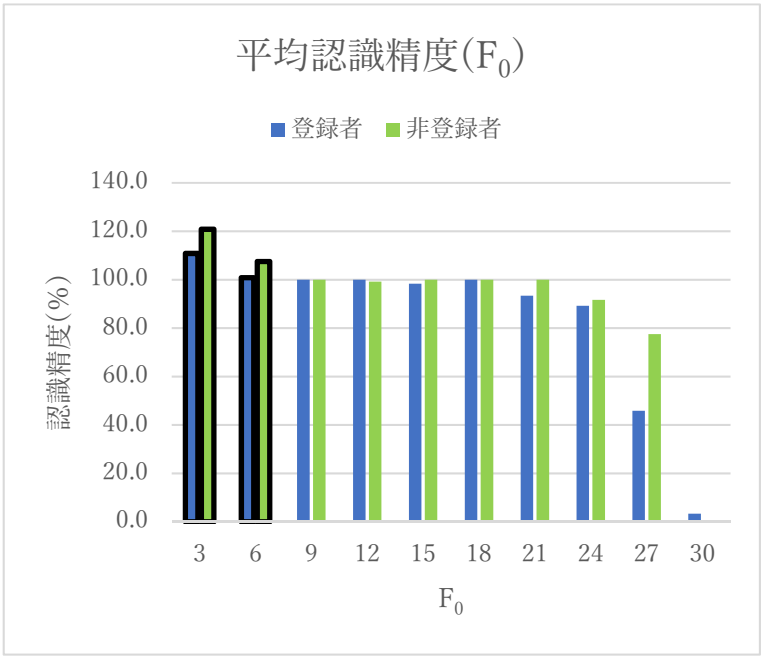


図 2-27 F_0平均認識精度

2.4.2　キャリブレーション（Calibration）　*大原広暉*

2.4.2 項以下では、IFI の投影映像とユーザーの位置を対応させる方法であるキャリブレーションについて述べる。類似研究の Interactive Floor では、〈フィールド[*1] の座標と投影映像の座標〉と〈フィールドの座標と Kinect で計測した人の位置〉をそれぞれ合わせることで、〈投影映像の座標と Kinect で計測した人の位置〉を対応させている。これは、〈フィールドの座標と投影映像の座標〉が一致していることが必要であるが、それらの位置合わせには誤差があったと小西らは述べている。そこで我々は Homography 行列[*2]を利用し、〈投影映像の座標と Kinect で計測した人の位置〉を直接対応する方法を Unity 環境下で確立した。それにより、たとえばキャリブレーションをすることで、ユーザーが立っている足元に波を発生させることができる（図 2-28）。

図 2-28 キャリブレーションによって足の位置と波源の位置が一致している

---

*1 床に黒いテープで囲ってある 2550[mm]x2710[mm]の矩形領域であり、映像が投影される

*2 射影変換に用いられる行列の一種

2.4.2.1 Interactive Floor と IFI のキャリブレーションの違い

ここでは、キャリブレーションについて類似研究である Interactive Floor の方法と IFI の方法を述べ、IFI の方法のほうが〈投影映像の座標と Kinect で計測した人の位置〉を正確に対応させられることを示す。

Interactive Floor にはフィールドの座標 **F**(3 次元)、投影映像の座標 **P**(2 次元)、Kinect で計測した人の位置の座標 **U**(3 次元)の 3 つがある。**F** は 2550[mm]x2710[mm]の矩形領域で固定し、床に黒いテープで可視化してある。**P** は、黒いテープで囲われた領域(フィールド)を RGB カメラから取得後に画像処理され、**F** に合わせてある。そして、**F** と **U** は変換行列 **M** を用いて式(1)で関係づけられている。

$$\mathbf{F} = \mathbf{MU} \qquad \cdots\text{式}(1)$$

変換行列 **M** は最小二乗法による近似で求められており、式(2)で計算される。**M** を求めるために、複数の箇所でフィールドの座標 **F** と Kinect で計測した人の位置の座標 **U** を計測しなければならない。

$$\mathbf{M} = \mathbf{F}\mathbf{U}^{\mathrm{T}}(\mathbf{U}\mathbf{U}^{\mathrm{T}})^{-1} \qquad \cdots\text{式}(2)$$

ただし、これではフィールドの座標 **F** と投影映像の座標 **P** が一致していなければ、**P** と Kinect で計測した人の位置の座標 **U** も一致せず、**P** と **U** が一致しない可能性がある。**F** を介在させた方法では「**P** と **U** の一致」というキャリブレーションの目的を達成しにくい。

そこで、〈ズレ〉の可能性を減らすために、IFI では、直接 Kinect で計測した人の位置の座標 **U** と投影映像の座標 **P** を 2 次元で対応させる。まず、人が投影映像内を移動するときに胴体は平行移動すると考えた。したがって、**U**(3 次元)は 2 次元(左右・前後方向)の値のみ扱うことで十分ではないかと考えた。そして、**P** と **U'**(**U** の左右・前後方向)を同一平面上に存在するようにした。

そして、Homography 変換によって投影映像の座標 **P** と **U'** を対応させた。あらかじめ指定した投影映像の 4 つ角の座標と、人が 4 つ角それぞれに立ったときの骨盤中心の骨格座標を用いて Homography 行列 **H** を求める。そして **P** と **U'** は **H** を用いて式(3)で関係づけられる。

$$\mathbf{P} = \mathbf{H}\mathbf{U}' \qquad \cdots\text{式}(3)$$

我々の方法は床の環境に依らないため、床に黒いテープが貼れない環境下(ex. 砂場)でも投影映像とユーザーの位置を対応できると考える。

2.4.2.2 IFI に実装した Homography 変換プログラム

ここでは、Unity 環境で Homography 行列を生成するプログラムを実装する方法を示し、プログラムで Kinect が取得した人の位置と投影映像を対応づける流れを説明する。

Kinect が計測する座標$(x,\ y)$と投影する映像の座標$(x',\ y')$の 4 組を取得できれば Homography 行列を作成できる。Homography 行列 **H** は式(4)で定義される。そして、**H** の要素は式(5)の行列演算から求めることができる。式(5)を式(5')とおくと、$\mathbf{h}=\mathbf{A}^{-1}\mathbf{b}$ に式変形できる(**A**: 正則行列)。すなわち、Kinect が計測する座標$(x,\ y)$と投影する映像の座標$(x',\ y')$の 4 組を取得できれば $\mathbf{A}^{-1}$ が計算でき、**H** の変数をすべて求められる。

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \cdots\text{式(4)}$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix} \qquad \cdots\text{式(5)}$$

$$\mathbf{Ah} = \mathbf{b} \qquad \cdots\text{式(5')}$$

それにしたがって、Homography 行列を得るソースコードを作成した(コード 2-1)。コード 2-1 では、Kinect が計測した人の座標と投影する映像の座標(図 2-29)を使って **A** を生成している。その後、コード 2-2 では **A** の逆行列 $\mathbf{A}^{-1}$ を.NET Framework 用数値計算ライブラリ「Math.Net.Numerics」の関数「Inverse()」を使って求めた。そして、$\mathbf{A}^{-1}\mathbf{b}$ を計算したものを変数〈homo〉に代入している。

```
void MakeInitialMatrix(Matrix<double> mat, Vector<double> bPoi, Vector<double> bPos){
    int step = 0;
    int index = 0;
    for (int n = 0; n < dimension; n++){
        var o1 = Math.Round(bPos[step], 2);
        var o2 = Math.Round(bPos[step + 1], 2);
        var o3 = Math.Round(bPos[step], 2) * bPoi[index];
        var o4 = Math.Round(bPos[step + 1], 2) * bPoi[index];
        switch (n % 2){
            case 0:
                        mat[index, 0] = o1;
                        mat[index, 1] = o2;
                        mat[index, 2] = 1;
                        mat[index, 3] = 0;
                        mat[index, 4] = 0;
                        mat[index, 5] = 0;
                        mat[index, 6] = -o3;
                        mat[index, 7] = -o4;
                        break;
            case 1:
                        mat[index, 0] = 0;
                        mat[index, 1] = 0;
                        mat[index, 2] = 0;
                        mat[index, 3] = o1;
                        mat[index, 4] = o2;
                        mat[index, 5] = 1;
                        mat[index, 6] = -o3;
                        mat[index, 7] = -o4;
                        step += 2;
                        break;
        }
        index += 1;
    }
}
```

コード 2-1 式(5')の A を生成する

```
void CalHmographyMatrix(Matrix<double> mat, Vector<double> bPoi){
    var mInverse = mat.Inverse();
    var homo = mInverse * bPoi;
    for (int i = 0; i < 8; i++){
        homoVector[i] = homo[i];
    }
}
```

コード 2-2 コード 2-1 で生成した 8x8 行列の逆行列(A^{-1})を求め、式(5')の h を生成する

Unity 環境下で、Kinect で計測した人の座標は World 座標系（3 次元）の値で扱われ、投影する映像の座標は Viewport 座標系（2 次元）の値で扱われる。2.4.2.1 でも述べたように、人の位置の鉛直方向の値は扱わないことで Homography 変換が適用できる。それぞれの座標系のイメージを図 2-29 に示す。

Homography 変換により人が動き回る領域（：投影映像領域）の四隅と投影する映像の四隅を一致させる。Kinect が計測する人が動く領域は不等辺な四角形である一方、投影する映像は長方形である。Homography 変換は異なる 2 つの四角形を一致させ、それぞれの面内の点を一対一で対応させることができる。それにより、人が動き回る領域の四隅と投影する映像の四隅を一致させ、それぞれの領域内の座標を一対一に対応できた。

キャリブレーションが終わった後は、毎フレームごとに人の位置を Homography 変換している（図 2-30）。Homography 行列 H の生成はキャリブレーション後に行われる。生成した後にシーンが切り替わるため、H の値を保存した変数「homoVector」を public な static 変数として定義する。IFI は、プログラム終了まで Homography 行列 H を保持している。

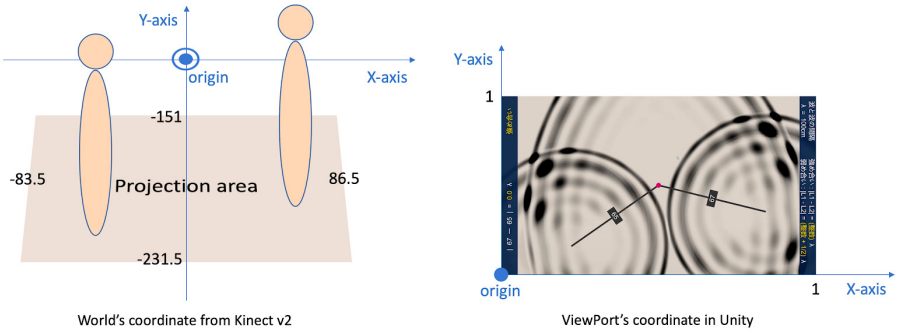


図 2-29 左：World 座標系と右：Viewport 座標系

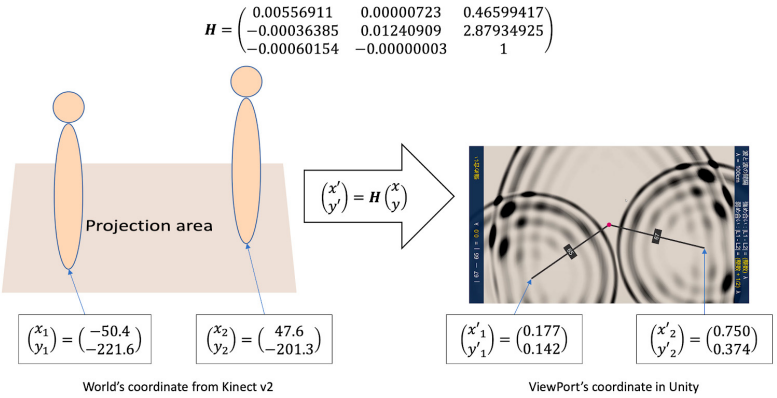


図 2-30 毎フレームごとに Homography 変換によって座標が変換される

2.4.2.3 IFI のキャリブレーション

ここでは、キャリブレーションを実行するためにユーザーが IFI 起動後に行う動作について解説する。

IFI 起動後、次ページの図 2-31 の画面（四隅すべての〈足跡の画像〉が表示されているが、実際は 1 つずつ）が投影され、ユーザーは以下の手順を行う。

I. 表示されている〈足跡の画像〉の上に立つ。
II. 中心にある円ゲージ 0%から 100%まで変化するまで待機する。
III. 円ゲージが 100%になったら、次に表示される〈足跡の画像〉に移動する。
IV. Ⅰ～Ⅲの手順を四隅全てで行う。

上記の操作が無事に終了すると、2.4.2.2 で実装したプログラムから Homography 行列 **H** が生成され（図 2-31）、シーンが切り替わる。

もしキャリブレーションが失敗して **H** が生成できない場合は、投影映像に再度前述の動作をするように求めるメッセージが表示され、上記のⅠ～Ⅳが再び実行する必要がある（次ページ図 2-32）。

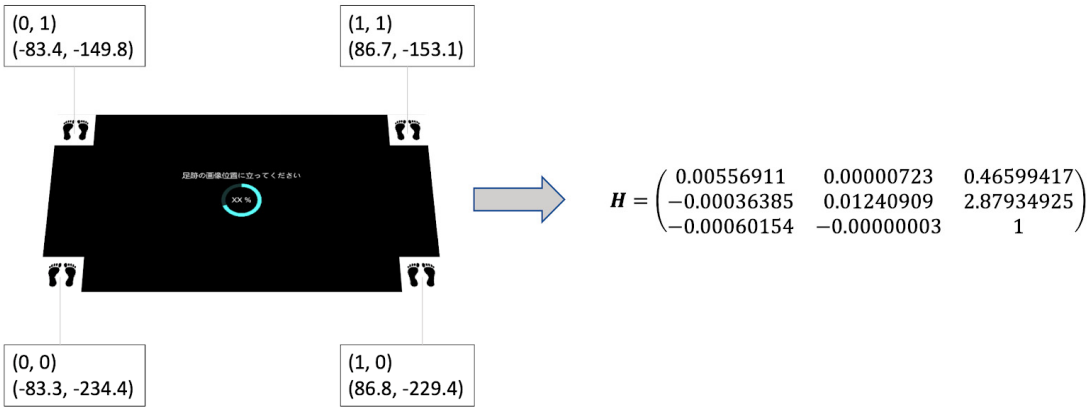


図 2-31 キャリブレーションによって Homography 行列を生成するイメージ図

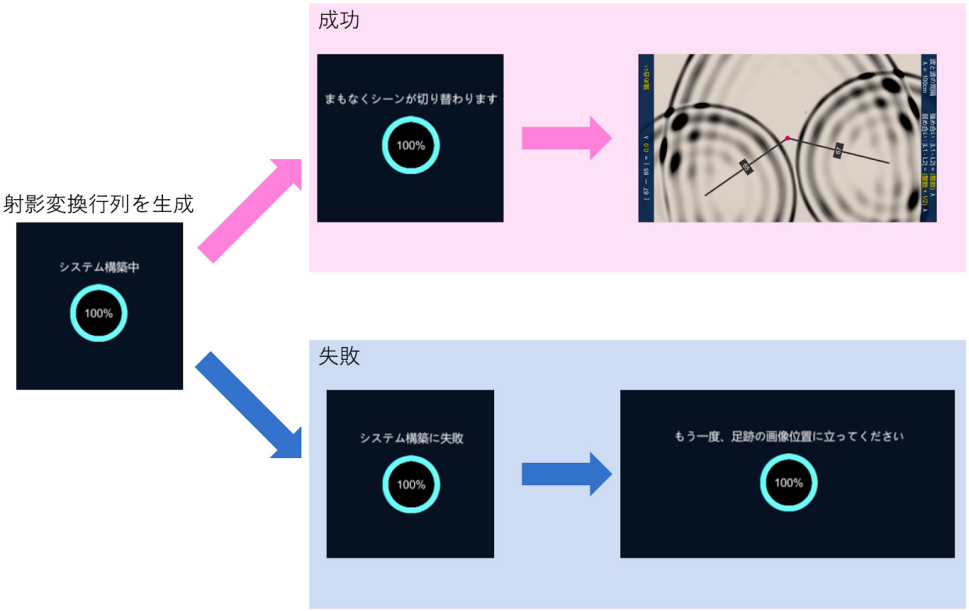


図 2-32 キャリブレーションが終わった後の流れ

2.5 IFI のハードウェア開発 *井上海翔*

本システムでは、〈Kinect と 2 台のプロジェクタの相対位置〉を固定することが必要である。そのため、SUS 株式会社の G-fun シリーズを用いて図 2-33 にある

- KP 固定装置（Kinect とプロジェクタを固定する装置）
- P 固定装置（プロジェクタを固定する装置）

を製作した。

図 2-33 KP 固定装置(左)、P 固定装置(右)

図 2-34 Kinect とプロジェクタを固定した装置(左)、プロジェクタのみを固定した装置(右)

2.5.1 Kinect・プロジェクタ固定装置の要求仕様

本システムに適した環境を構築するために KP 固定装置・P 固定装置の製作を行った。以下に、Kinect とプロジェクタを固定する装置に必要な仕様を示す。それぞれの詳細を 2.5.2 項以降で述べる。

- 適当な投影領域を床に確保する。
- ジェスチャーの認識領域に投影領域を含むよう Kinect を設置する。
- 投影映像を一致させるために、2台のプロジェクタの相対位置を固定する。
- プロジェクタが装置から落下することを防ぐ。
- プロジェクタの排熱を処理する。

2.5.2 IFI が展開する環境

我々が製作した KP 固定装置・P 固定装置によって実現される環境のイメージを図 2-35 に示す。IFI では Kinect によって人のジェスチャーを認識するため、その認識領域内にプロジェクタの投影領域を含む必要がある。

プロジェクタを床から 500 ㎜の高さに立てて設置したときに床に映像が投影される領域を投影領域とする。すると、投影領域の短辺は 1400mm となり、プロジェクタと投影領域の距離は 500mm となるため、図 2-35 のように両固定装置間の距離は 2400 ㎜になる。

投影領域内でユーザーのジェスチャーを常に認識できるようにするために、以下の条件を満たす位置に Kinect を設置する必要がある。

1. ユーザーの全身が常時 Kinect の認識領域に含まれるよう最低 1000 ㎜以上の距離を確保する。
2. Kinect とユーザーの間に障害物が存在しない。

条件 1 を満たすために、投影領域内でユーザーが Kinect に最も近づいたときに、ユーザーと Kinect が最低 1000 ㎜離れるようにした。

条件 2 を満たすために Kinect を床から 1500 ㎜の高さに設置した。これ以上高さを下げると、プロジェクタが Kinect の認識領域に入ってしまう。

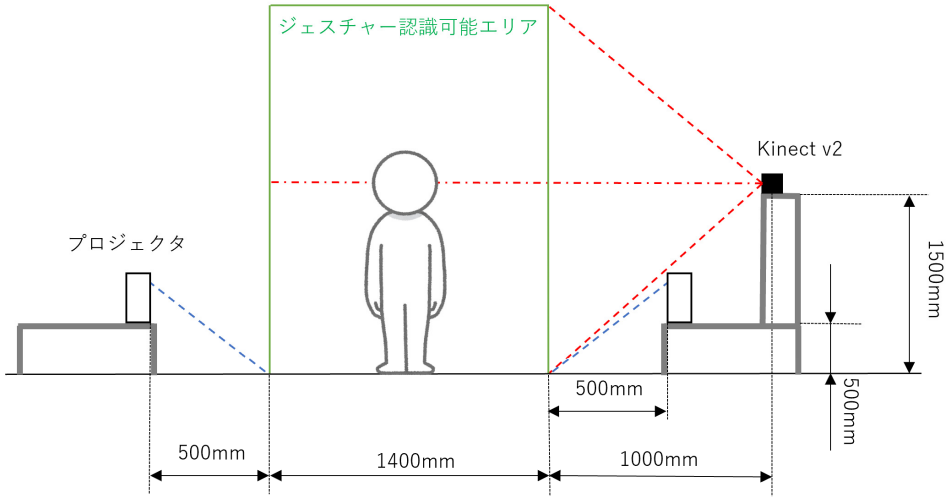


図 2-35 各機器の設置位置

2.5.3　Kinect・プロジェクタ固定装置の相対位置

2 台のプロジェクタから投影された映像を正確に重ねるために、KP 固定装置と P 固定装置の相対位置を固定する必要がある。その目的で、KP 固定装置と P 固定装置の前面同士を 2400 ㎜程離して設置するための仕組みを作成する。

図 2-36 に示す 2400 ㎜の固定棒を KP 固定装置と P 固定装置の前面に取り付けることにより、KP 固定装置と P 固定装置の距離を 2400 ㎜に保つことが出来る(図 2-37)。

図 2-36 固定棒

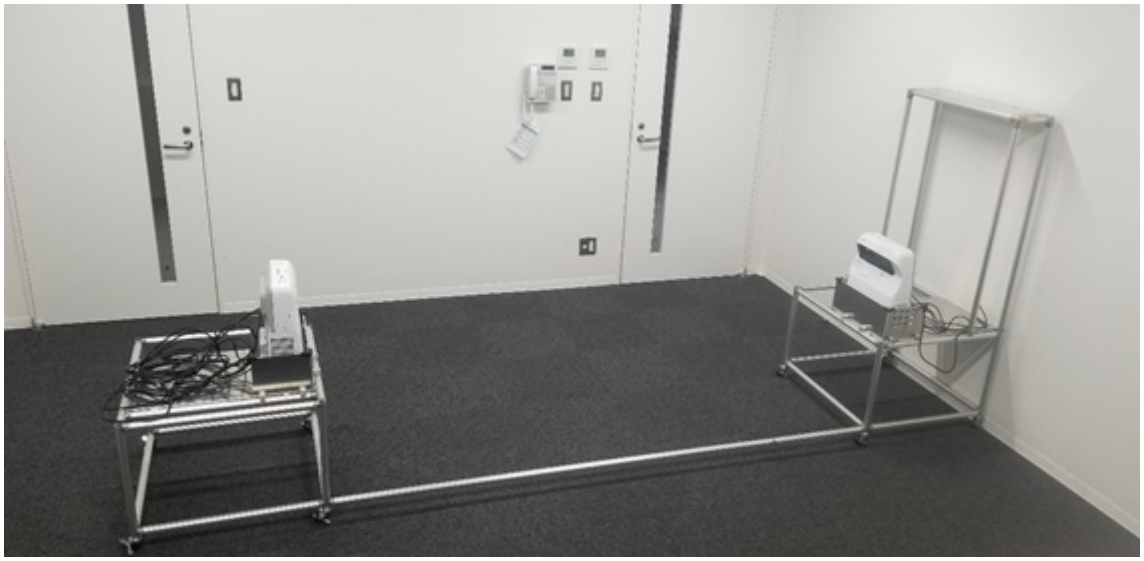


図 2-37 固定時の全体図

2.5.4　プロジェクタの落下防止板

プロジェクタの落下を防ぐために、Kinect・プロジェクタ固定装置に落下防止板（図 2-38）を取り付けた。

しかし、この落下防止板によってプロジェクタの排熱部分を塞いでしまった。プロジェクタの排熱機構は左側面にあり、落下防止板の一部に空気穴を開けなければ高温になってしまう。排熱部分を塞ぐ落下防止板の表面温度を測定した結果、30 分経過した段階で約 55℃になった。

そこで、図 2-39 に示すように空気穴を開け、図 2-38 のように設置した。空気穴を開けた後に再度表面温度を測定した結果、30 分経過した段階もで約 45℃に保たれていた。

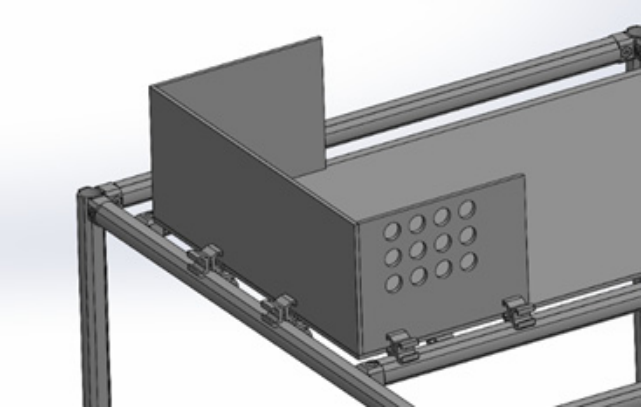


図 2-38 落下防止板

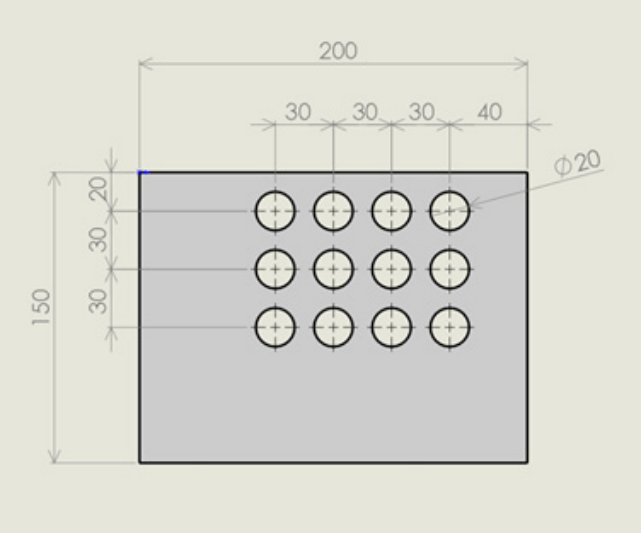


図 2-39 排熱処理を施した落下防止板

2.6 IFI を用いた教育コンテンツ開発:〈水面波の干渉アニメーション〉

新屋吉昭, 大原広暉

2.6.1 波の発生の仕組み

2.6.1.1 波のメカニズム

波を円が広がっていく様子を IFI で表現したい。現実世界では、音波や電磁波、水面波などの振動、波動現象は波動方程式で解析できる。式(6)は 3 次元の波動方程式であり、hは波の高さ(振幅)、tは時間、x , yは平面座標、v^2は位相速度の 2 乗である。

$$\frac{\partial^2 h}{\partial t^2} = v^2\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}\right) \qquad \cdots式(6)$$

しかし波動方程式をそのまま用いて水面波の干渉アニメーションを表現すると、学習に用いるには計算量が多すぎるという問題がある。そのため以下の方法で円状に広がる波紋を簡易的に表現した。

I. 発生時刻t_0のときの波源(x_0, y_0)から画面にある全てのピクセルの距離を計測
II. それらを、現時刻tの波源から波の先頭までの距離$v(t - t_0)$から引く
III. その結果に応じた色で各ピクセルを色付けする。負の値をとるピクセルは無色で描かれる。

図 2-40 を例に説明すると、d1 のピクセルは d-d1 で負の値なので色がつかない。d2のピクセルは d-d2 で正の値をとり、それに対応した色が描かれている。

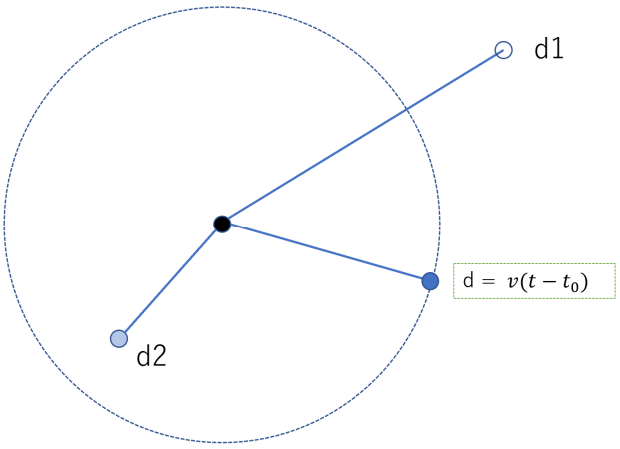


図 2-40 波の表現方法

2.6.1.2 波を表現するコードを実装

〈現実の波に近い〉波の干渉表現をするために ShaderLab*1 で作成したコードを実装した。まず波を表現するにあたり、ふたつのコード、CPU 処理を用いた Unity 上の C#のコードと GPU 処理を用いた ShaderLab のコードを作成した。その二つのコードを検討した結果、C#のコードは複数の波をシングルスレッドで処理しているのに対し、ShaderLab の方は複数の計算を同時に処理しているため、ShaderLab の方が C#の方よりも処理速度が 10fps 速かった。そのため Shader で作成したコードを採用した。そして 1f(フレーム)ごとに波源に波を起こすプログラムを実装した。また作成するにあたり、keijiro 氏のサンプルコードを参考にした【出典:[13]】。なお、ここで発生した波は下記のパラメータを持つものとする。

- 波長 100cm
- 周波数 3Hz

2.6.1.3 波源の設定

波の波源を IFI 内にいるユーザーの骨盤の位置に設定するため、Kinect を用いてユーザーの骨格座標を取得する。骨盤を選んだのは、ユーザーが立って床に投影されている画面を見おろす際に上半身が反れてしまっても、骨盤であれば位置がずれないと考えたからである。それをホモグラフィー変換*1 した座標を波源とする。骨格情報は常時取得しているため、ユーザーが動いた場合波源もその動きに追従する。

図 2-41 Kinect による人間の骨格情報の取得

---

*1 ShaderLab とは Unity で使える Shader のことである。

2.6.2　Distance モード

Distance モードは、IFI 内に二つの波源が存在するときに、波の強め合いと弱め合いを示す干渉の状態を図示するモードである。波が強め合うか弱め合うかを判定する位置は投影領域の中央であるとする。例えば図 2-42 において、ユーザー2 人が動くことで 2 つの波源が移動したとき、画面中央のピンク点において波が強め合うか弱め合うかを判定する。このモードは、波の干渉の公式について理解することを目的としている。

2.6.2.1　画面情報

Distance モードの画面には波の干渉の公式を理解するために必要な以下の情報を表示している。

- 波の波長(100cm)
- 波の強め合い、弱め合いの公式
- (「強め合い:|L1-L2|=(整数) λ」、「弱め合い:|L1-L2|=(整数+1/2)」)
- ユーザーとピンク点の距離
- 距離を代入した数式
- ピンク点において波が「強め合い」状態と「弱め合い」状態のどちらであるかの表示

図 2-42 を用いて説明すると、右上にある 100cm とは波の波長であり、右下にある二つの数式(「強め合い:|L1-L2|=(整数) λ」、「弱め合い:|L1-L2|=(整数+1/2」)は波の強め合い、弱め合いの公式である。真ん中に存在するのがピンク点である。そこから伸びている2本の黒い線(以下、キョリ線)はピンク点と各波源を結んだものであり、その線上にある数値(67, 65)はピンク点から各ユーザーの距離である。そして左下にある数式(|67 - 65| = 0.0 λ)は公式に現在の距離を代入したものであり、左上にピンク点で波が強め合っているか弱め合っているかを示す。この場合は強め合っているので「強め合い」と表示されている。

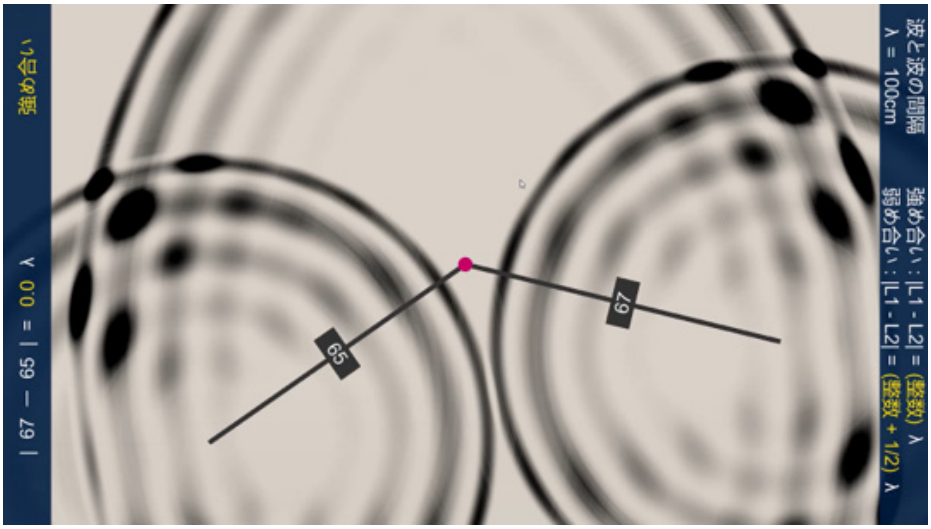


図 2-42 Distance モード

がピンク点と各波源の位置を線で結ぶプログラムをコード 2-3 のように実装した。距離はユーザーとピンク点の座標を用いて算出し、角度は二つの座標を逆三角関数で求めている。

また波の「強め合い」、「弱め合い」の結果表示は次ページのコード 2-4 のように実装した。先程求めた距離を公式に代入し、その算出結果で結果の表示を行う。解が 0.00 から 0.05 の場合は、ピンク点で波が強め合っているから「強め合い」と表示し、解が 0.45 から 0.55 の場合はピンク点で波が弱め合っているので「弱め合い」と表示し、それ以外の場合は無表示になる。条件に幅を持たせているのは、ユーザーが動かずに立っているときにも存在する揺らぎを吸収するためである。

```
void DrawLines(Vector2 dropPos, GameObject l, Text n)
    {
       Vector2 p = Camera.main.ViewportToScreenPoint(dropPos);
      var d = Vector2.Distance(p, screenCenter);
      var rad = Mathf.Atan2(p.y - screenCenter.y, p.x - screenCenter.x);
       l.GetComponent<RectTransform>().sizeDelta = new Vector2(10f, d);
       l.GetComponent<RectTransform>().localEulerAngles = new Vector3(0, 0, rad * Mathf.Rad2Deg
+ 90);
      var convertedD = (d / 10);
       n.text = convertedD.ToString("F0");
       Distances.Add(convertedD);
    }
```

コード 2-3 ユーザーとピンク点を結ぶ線を作成、距離算出するプログラム

```
// リアルタイムで公式を変動させる
txtResult.gameObject.SetActive(true);
difference = Mathf.Abs(Distances[Distances.Count - 2] - Distances[Distances.Count - 1]) / lambda;
 txtResult.text = difference.ToString("f1");
            //公式に実際の距離数値を代入
           if (difference >= 0)
            {
               txtFormula.gameObject.SetActive(true);
            }
           else
            {
               txtFormula.gameObject.SetActive(false);
            }
            // 強弱判定
           if (0 <= difference % lambda && difference % lambda <= 0.05)
            {
               tuyo(); //「強め合い」表示
            }
           else if (0.5 - 0.05 <= difference && difference <= 0.5 + 0.04)
            {
               yowa(); //「弱め合い」表示
            }
           else // どちらでもないので無表示
            {
               txtStrong.gameObject.SetActive(false);
               txtWeak.gameObject.SetActive(false);
            }
```

コード 2-4 ユーザーの距離を代入した公式の結果を用いた強弱判定のプログラム

2.6.2.2　波の周波数の変更

Distance モードでひとりのユーザーが右手を上げるジェスチャーを行うと、波の周波数が「3Hz」から「6Hz」に切り替わる。それにより、周波数の変更が波をどう変化させるかをユーザーに示すことが出来る。その際、画面右上の波長の表示も「100cm」から「50cm」に変更される。そしてもう一度同じジェスチャーを行うとデフォルトの周波数である「3Hz」に戻る。

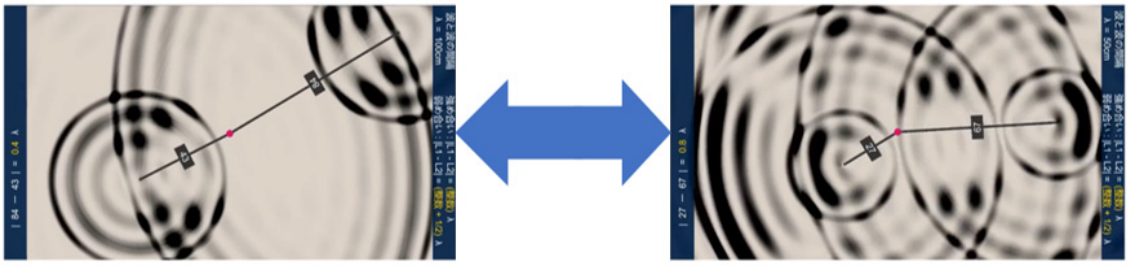


図 2-43 波の周波数変更

2.6.2.3　波の一時停止

Distance モードでひとりのユーザーが両手を水平に上げるジェスチャーを行うと、画面を一時停止することが出来る。それにより、ユーザーが波の様子をその場から離れて観察することが出来る。そしてもう一度両手を水平に上げるジェスチャーを行うと波が動き始める。

2.6.2.4 Real モード

Real モードは、Distance モードの背景をプールのタイル模様に変えたものである。背景を変えることで、現実世界の波に近づけ、現実世界に起きた場合の波の干渉の理解を目的としている。このモードでも「波の周波数変更」と「波の一時停止」を行うことが出来る。

このモードへは、Distance モード時にひとりのユーザーが左手を挙げるジェスチャーを行うことで移行される。そしてもう一度左手をあげるジェスチャーを行うことで Distance モードに戻る。

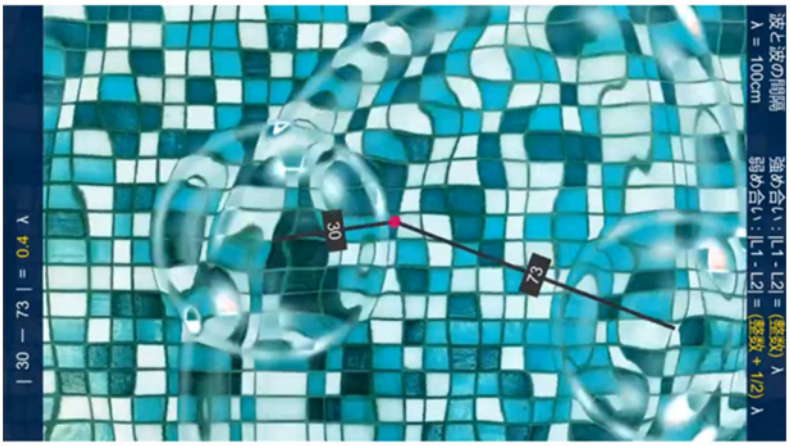


図 2-44 Real モード

2.6.3　Moiré モード

Moiré モードは、図 2-45 のように波を幾何学的に表現したモードである。このモードの波は、試験問題に使われる波の図、すなわち波の山と谷が等間隔で描かれる図と近いため、試験問題対策として優れていると考えている。Distance モードの時に、ひとりのユーザーが両手を上げるジェスチャーを行うことでこのモードに移行する。

2.6.3.1　画面情報

Moiré モードでは波の山と谷をそれぞれ濃い線と薄い線で表し、また山と山および谷と谷が重なる点をピンク色で、山と谷の重なる点を水色で表示している。

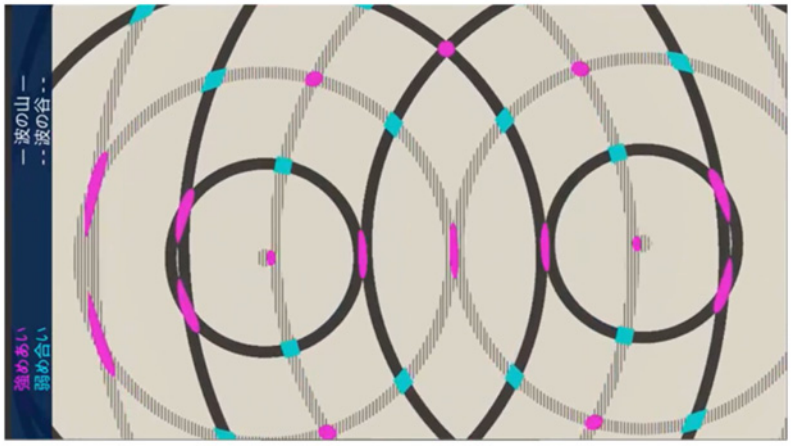


図 2-45 Moiré モード

2.6.3.2 Moiré モードの独自の波源の設定

波源がユーザーの位置に追従する Distance モードとは異なり、Moiré モードでは波源を固定している。すなわち、ユーザーが投影領域内で移動しても波源は追従せずにその場にとどまり波を発生し続ける。

そのような動作を実現するため、Distance モードから Moiré モードに移行する際にユーザー2 人の位置座標を保存する。そしてその座標を固定された波源として幾何学模様の波を発生している。そのためユーザーは少し離れたところで投影領域内の波と干渉点の動きを観察することが出来る。

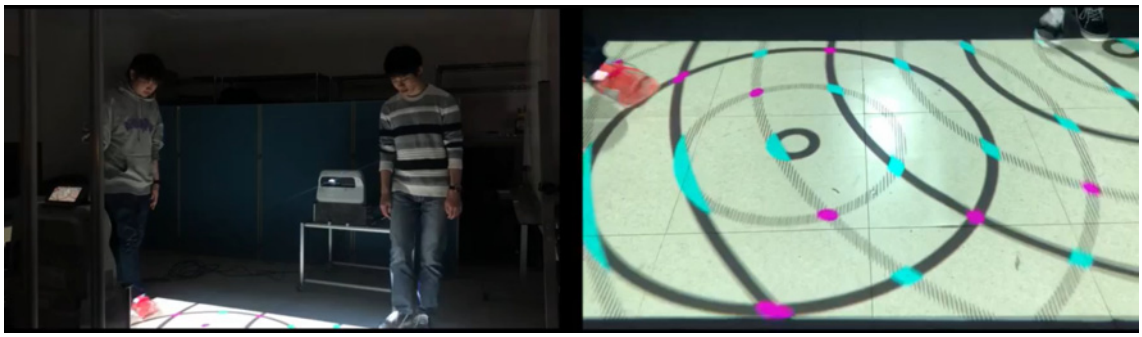


図 2-46 ユーザーが画面を観察している様子

3 IFI を用いた教育支援システムの評価　　*新海大志*

本システムが水面波の干渉の学習に効果があるかを示すために評価実験を行った。

3.1 評価方法

我々は評価実験を 2 つのグループ(以下グループA・グループ B とする)に分けて行った(図 3-1 参照)。今回の実験では、グループ A 及びグループ B をそれぞれ 10 人ずつとした。

I. グループ A・グループ B 共に我々が作成した高校物理の水面波の干渉に関する問題(付録参照)を解いてもらう。問題は 2.3.1 項でも紹介した図 3-2 の内容を問うもので、各 1 点の問題を 4 問出題した。なお実験協力者には問題の正誤及び解答は伝えない。
II. グループ A には高校物理の教科書[*1]を用いて学習を 10 分間行ってもらう。
III. グループ B には本システムの使用方法の説明をし、本システムを用いて学習を 10 分間行ってもらう。
IV. グループ A・B 共に学習前に行った試験と同じ問題を再度解いてもらう。
V. 得られた試験結果を元にt検定を行うことで、統計的に評価する。

t検定とは、2 つの正規分布の母集団の平均が等しいかを検定する際に使われるものである。t検定を行うと、それぞれの平均の差以上の差が発生する確率を示すp値を算出することができる。一般にp<0.05 未満(pが 5%未満)であると、2 つの母集団の間に有意差が認められることになり、同じ傾向を示していないということになる。

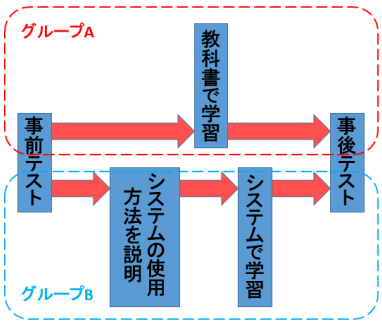


図 3-1 実験手順

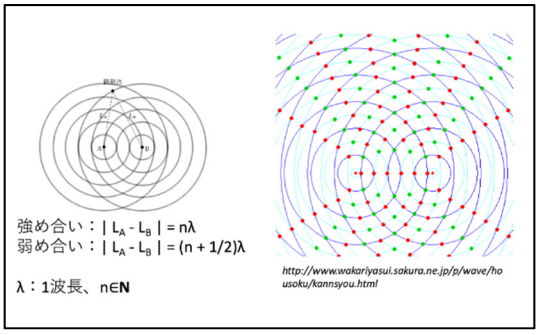


図 3-2 問題の概要

---

*1 中村英二ら,『高等学校　物理』, 第一学習社, 2014

3.2 結果

実験結果を表 3-1・表 3-2 に示す。グループ A における学習前と学習後の得点分布を示したのが次ページの図 3-3 である。人数の多い山型の構造が、学習によって高得点側に移動していることが分かり、学習に効果があったことを示している。グループ B における学習前と学習後の得点分布を示している次ページの図 3-4 も同様の傾向を示している。しかし、この結果は偶然である可能性もある。そのためt検定を行い学習によって得点が増加したかを判定した。

グループ A のp値は 0.0100 となり学習前と学習後の結果に有意差が認められたため、教科書を用いた学習の効果で得点が上昇したと言える。また、グループ B のp値は 0.0002 となったため、こちらも同様に本システムを用いた学習の効果で得点が上昇したと言える。

表 3-1 グループ A の試験結果

実験協力者	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	平均
事前結果	3	3	2	3	2	2	2	1	1	3	2.2
事後結果	3	3	3	4	2	3	4	3	3	3	3.1
差分	±0	±0	+1	+1	±0	+1	+2	+2	+2	±0	+0.9

表 3-2 グループ B の試験結果

実験協力者	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	平均
事前結果	2	1	1	0	1	2	2	1	1	1	1.2
事後結果	4	4	2	2	3	3	2	2	3	3	2.8
差分	+2	+3	+1	+2	+2	+1	±0	+1	+2	+2	+1.6

図 3-5 に両グループの学習前試験と学習後試験の平均点を示す。学習によってグループ A は平均点が 0.9 点上昇し、グループ B は平均点が 1.6 点上昇した。本システムを使用したグループの平均点の方が増加量が大きいため、教科書での学習より効果があると考えられる。しかしこれも偶然である可能性がある。そのためt検定を行い、本システムを使用した学習の方が効果があるかを判定した。

t 検定の結果、p値が 0.085 となり 0.05 を上回ったため、有意差が認められなかった。そのため、本システムを用いた学習が教科書を用いた学習よりも良い成果を上げられると断定はできない結果となったが、今後のシステムの改善により、優れた教育効果を生みだせる可能性はある。

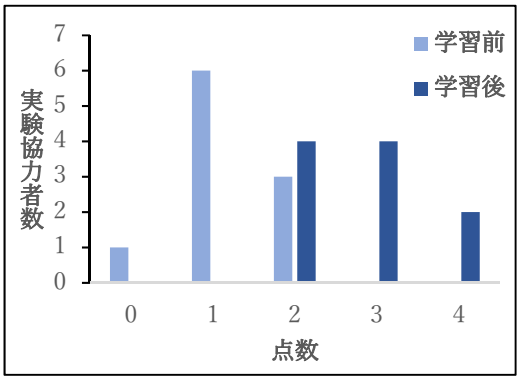


図 3-3 グループ A の得点分布

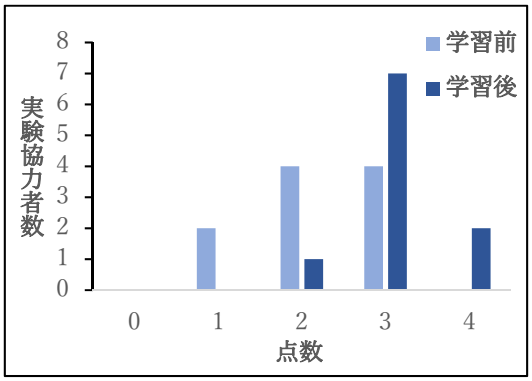


図 3-4 グループ B の得点分布

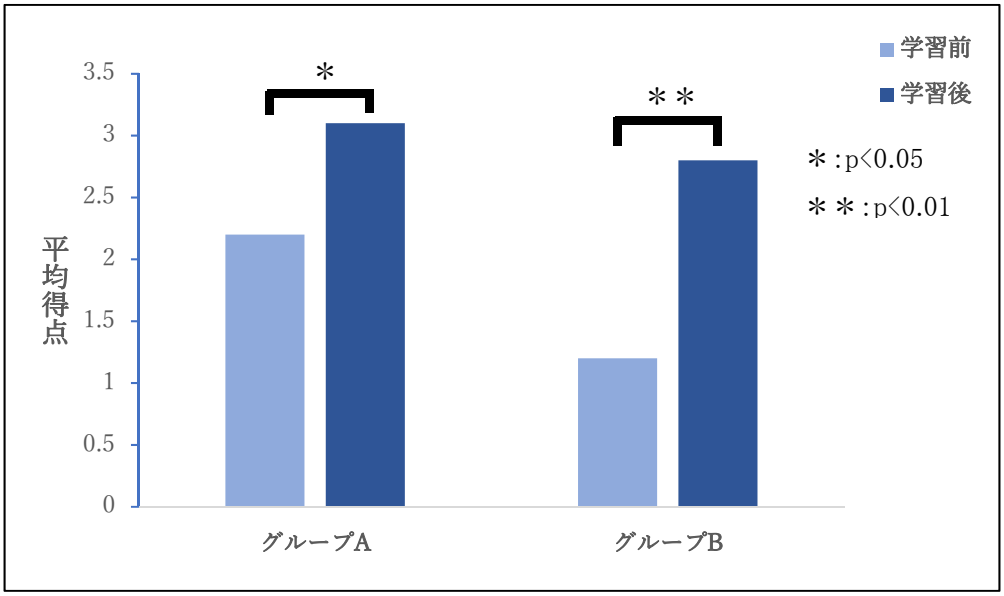


図 3-5 各グループの平均得点

4 結論　　*新海大志*

本研究では、以下の 3 つの条件を満たす〈教育支援システム〉の開発を目指した。

1. 2 台のプロジェクタを使用して、床に映像を投影する。
2. 生徒参加型の教育コンテンツを提供する。
3. ユーザーとシステムがインタラクティブである。

1 の条件については、Kinect・プロジェクタ固定装置の開発によって条件を満たした。

2 の条件については、水面波の干渉アニメーションを製作し、水面波の干渉に関する教育コンテンツを開発することによって条件を満たした。今回は水面波の干渉アニメーションを制作したが、ほかの教科書の内容の〈IFI を用いた教育支援システム〉も実現できると考えられる。

3 の条件については、Kinect を用いることでユーザーのジェスチャーを入力とし、投影映像を操作できるシステムの開発によって条件を満たした。

4.1 課題

本システムの使用者へアンケートを行ったところ、

- 2 人で IFI を利用した時のジェスチャーの認識精度が悪い

という問題が明らかになった。

この問題の原因は、本システムで作成した識別器が3人のジェスチャーから作成されているためだと考えられる。識別器を作成する際により多く人のジェスチャーを含めることでこの問題を解決できるのではと考える。

また本システムを用いた学習が教科書を用いた学習よりも良い成果を上げられることを示せなかったという問題がある。これについても今後改善する必要がある。

5 参考文献

[1] IoT 技術の国内利用動向調査(2019 年 11 月実施)|MMRI 株式会社 MM 総研
https://www.m2ri.jp/news/detail.html?id=387

[2] 新学習指導要領総則|文部科学省
https://www.mext.go.jp/component/a_menu/education/micro_detail/_icsFiles/afieldfile/2019/03/18/1387017_001.pdf

[3] 一般社団法人 日本教育情報化振興会.“第 11 回 教育用コンピュータ等に関するアンケート調査 報告書|一般法人日本教育情報化振興会
http://www2.japet.or.jp/info/japet/report/ICTReport11.pdf

[4] 新しい学習指導要領の考え方-中央教育審議会における議論から改定そして実施へ-|文部科学省
https://www.mext.go.jp/a_menu/shotou/newcs/_icsFiles/afieldfile/2017/09/28/1396716_1.pdf

[5] 高等学校におけるアクティブラーニングの視点に立った参加型授業に関する実態調査 2015|東京大学大学総合教育研究センター中原淳研究室、 日本教育研究イノベーションセンター
http://manabilab.jp/wp/wp-content/uploads/2015/12/1streport.pdf#page=19

[6] 初のアクティブラーニング全国調査の結果を大公開！|マナビラボ
http://manabilab.jp/article/357

[7] 2 台のプロジェクタを用いた Interactive　Floor の構築 ｜ 小西由香理
https://ipsj.ixsq.nii.ac.jp/ej/index.php?action=pages_view_main&active_action=repository_action_common_download&item_id=164916&item_no=1&attribute_id=1&file_no=1&page_id=13&block_id=8

[8] アクティブラーニングの学習効果に関する検証グループワーク中心クラスと講義中心クラスの比較による ｜ 杉山 成・辻 義人
https://barrel.repo.nii.ac.jp/?action=repository_uri&item_id=530&file_id=19&file_no=1

[9] アクティブラーニングの学習効果に関する検証(2)― 学習者の自尊感情が社会人基礎力の獲得に及ぼす影響に注目して ― ｜ 杉山 成・辻 義人
https://barrel.repo.nii.ac.jp/?action=repository_uri&item_id=506&file_id=19&file_no=1

[10] Kinect を用いた AR による鏡像シミュレーション教材の活用 -虚像の理解を促す指導法の検討-｜ 大崎 貢、他 3 名
https://www.jstage.jst.go.jp/article/jssep/39/0/39_191/_pdf/-char/ja

[11] Kinect 2.0 for Windows Available July 15 | Time　https://time.com/2962269/kinect-v2-0-for-windows/

[12] スプリッター(分配器)　GH-HSPC4-BK | GREEN HOUSE グリーンハウス
https://www.green-house.co.jp/products/gh-hspc4/

[13] GitHub - keijiro/RippleEffect: Water surface ripple effect for Unity | Keijiro
https://github.com/keijiro/RippleEffect

謝辞

最後に、本研究を進めるにあたり、2 年間ご指導していただきました工学院大学 先進工学部機械理工学科 金丸隆志教授、協力していただいた研究室の後輩と学科の同級生にメンバー一同心より感謝いたします。これをもって謝辞とかえさせていただきます。

付録

Unity「波紋のアニメーション（Pixel）」のコード　*大原広暉*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MakeRipple : MonoBehaviour
{

    public AnimationCurve waveform = new AnimationCurve(
        new Keyframe(0.00f, 0.50f, 0, 0),
        new Keyframe(0.05f, 1.00f, 0, 0),
        new Keyframe(0.15f, 0.10f, 0, 0),
        new Keyframe(0.25f, 0.80f, 0, 0),
        new Keyframe(0.35f, 0.30f, 0, 0),
        new Keyframe(0.45f, 0.60f, 0, 0),
        new Keyframe(0.55f, 0.40f, 0, 0),
        new Keyframe(0.65f, 0.55f, 0, 0),
        new Keyframe(0.75f, 0.46f, 0, 0),
        new Keyframe(0.85f, 0.52f, 0, 0),
        new Keyframe(0.99f, 0.50f, 0, 0)
    );
    Texture2D drawTexture;
    Color[] buffer;

    float v = 200.0f;
    float f = 0.3f;

    List<WaveFront> waveList = new List<WaveFront>();
    int Height;
    int Width;
    float waveValueMin = 0f;
    float waveValueMax = 2f;
    public GameObject plane;

    void Start()
    {
        Texture2D mainTexture = (Texture2D)GetComponent<Renderer>().material.mainTexture;
        Color[] pixels = mainTexture.GetPixels();
        buffer = new Color[pixels.Length];
        pixels.CopyTo(buffer, 0);

        Height = mainTexture.height;
        Width = mainTexture.width;
        drawTexture = new Texture2D(Width, Height, TextureFormat.RGBA32, false);
        //Debug.Log("画面サイズは、" + Width + "x" + Height + "pixels");
    }

    void Update()
    {
        #region クリックした位置に波を生成
        //if (Input.GetMouseButton(0))
        //{
        //    if (waveList.Count != 0)
        //    {
```

```
        //          if (Time.time > waveList[waveList.Count - 1].t + f)
        //          {
        //              //StartCoroutine(generateWave());
        //              generateWave();
        //          }
        //      }
        //      else
        //      {
        //          //StartCoroutine(generateWave());
        //          generateWave();
        //      }
        //}
        #endregion

        #region ランダムな位置に波を生成
        if (waveList.Count != 0)
        {
            if (Time.time > waveList[waveList.Count - 1].t + f)
            {
                generateWave();
            }
        }
        else
        {
            generateWave();
        }
        #endregion

        //Debug.Log(waveList.Count);

        makeWaveImage();
        drawTexture.SetPixels(buffer);
        drawTexture.Apply();
        GetComponent<Renderer>().material.mainTexture = drawTexture;
    }

    public class WaveFront
    {
        public float t = Time.time;
        public float x;
        public float y;
    }

    public void generateWave()
    {
        #region クリックした位置に波を生成
        //Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        //RaycastHit hit;
        ////yield return new WaitForSeconds(1f);
        //if (Physics.Raycast(ray, out hit, 100.0f))
        //{
        //    WaveFront W = new WaveFront();
        //    W.x = hit.textureCoord.x * 256;
        //    W.y = hit.textureCoord.y * 256;
```

```
        //      waveList.Add(W);
        //}
        #endregion

        #region【テスト】ランダムに波を生成
        WaveFront W = new WaveFront();
        W.x = Random.value * 256;
        W.y = Random.value * 256;
        waveList.Add(W);
        #endregion
    }

    public void makeWaveImage()
    {
        for (int j = 0; j < Height; j++)
        {
            for (int i = 0; i < Width; i++)
            {
                float waveValue = 0f;
                foreach (WaveFront W in waveList)
                {
                    float len = Mathf.Sqrt(Mathf.Pow(i - W.x, 2f) + Mathf.Pow(j - W.y, 2f));
                    if (v * (Time.time - W.t) > len)
                    {
                        float passedTime = Time.time - len / v - W.t;
                        waveValue += waveShape(passedTime);
                    }
                }
                if (waveValue > waveValueMax)
                {
                    waveValue = waveValueMax;
                }
                else if (waveValue < waveValueMin)
                {
                    waveValue = waveValueMin;
                }
                if (waveValue < waveValueMax)
                {
                    Color waveColor = new Color(188f / 255f * (waveValue / waveValueMax), 226f /
255f * (waveValue / waveValueMax), 232f / 255f);
                    buffer.SetValue(waveColor, i + Width * j);
                }
                //else
                //{
            //  重なった部分に色を付ける
                //      buffer.SetValue(Color.magenta, i + 256 * j);
                //}
            }
        }
        if (waveList.Count != 0)
        {
            if (v * (Time.time - waveList[0].t) > Mathf.Sqrt(Mathf.Pow(Width, 2) + Mathf.Pow(Height,
2)))
            {
```

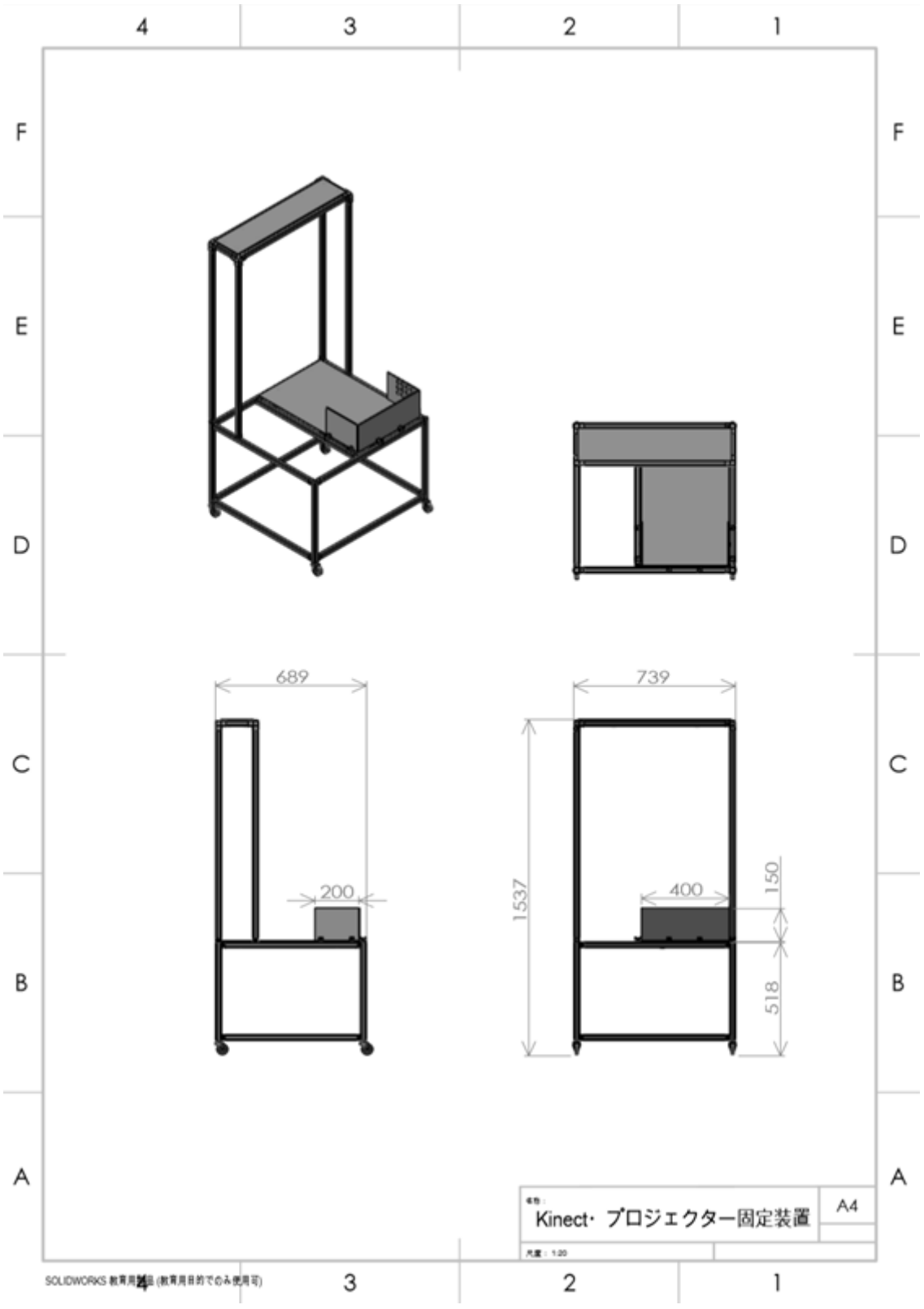
```
                waveList.RemoveAt(0);
            }
        }
    }

    public float waveShape(float pT)
    {
        //float maxindex = 10f / alpha;    //計算する範囲を指定
        //if (pT < maxindex)
        //{
        //    return alpha * pT * Mathf.Exp(1 - alpha * pT);
        //}
        //else
        //{
        //    return 0f;
        //}
        return (waveform.Evaluate(pT * 3) - 0.5f) * 2;
    }
}
```

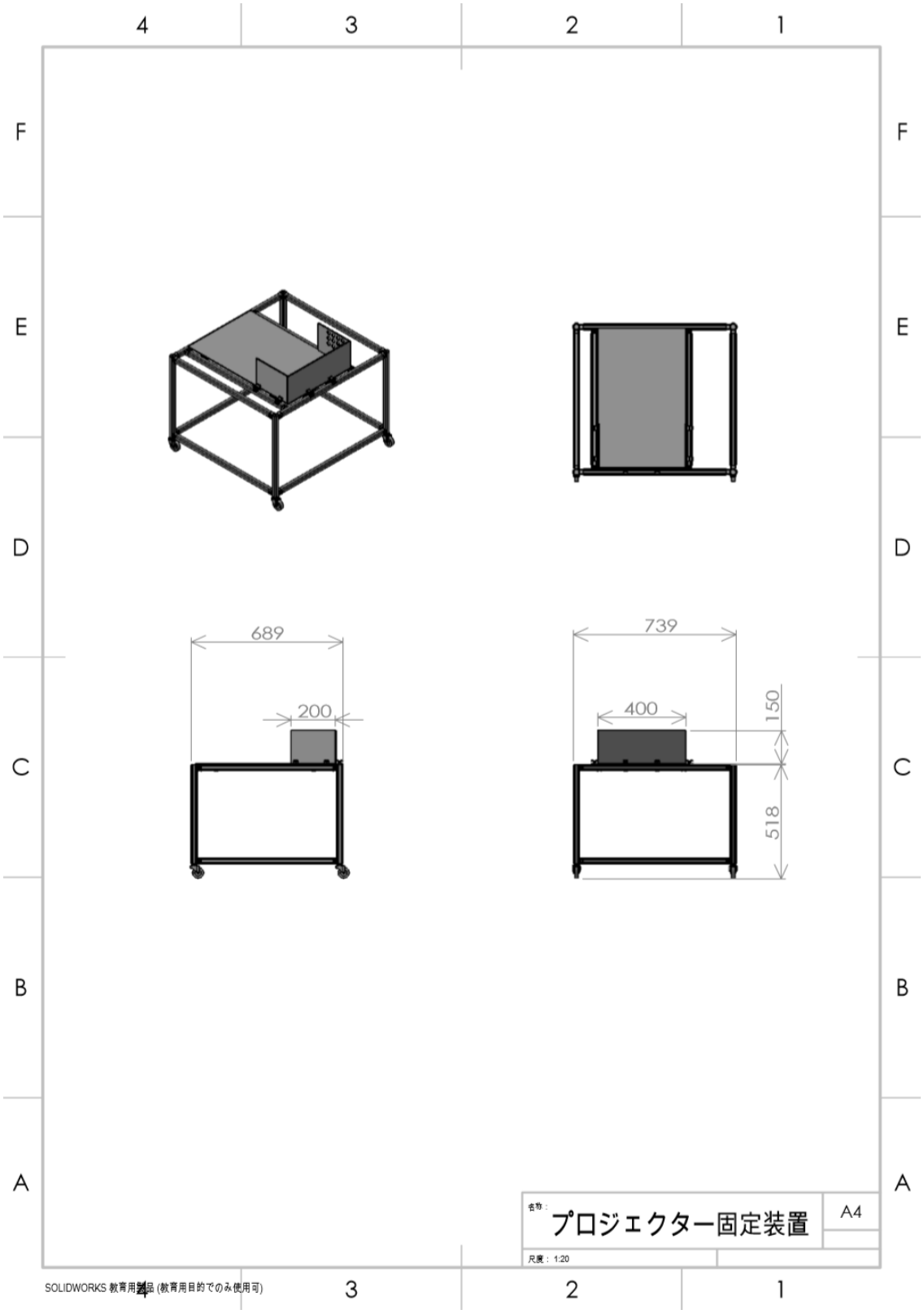
固定装置に使用した材料 *井上海翔*

Kinect・プロジェクタ固定装置	
材料名	使用数
GFun フレーム N 1000mm(切断加工)	4
GFun フレーム N 650mm(切断加工)	6
GFun フレーム N 600mm	5
GFun フレーム N 450mm	4
GFun フレーム N 100mm	2
Gfun マルチコネクタインナー	20
Gfun ストレートコネクタインナー	2
Gfun アングルコネクタ DX	4
Gfun φ50N キャスター	4
Gfun ボードホルダ	14
アルミ板 5×400×640	1
アルミ板 5×150×400	1
アルミ板(5×150×200)	2
アルミ板(5×150×700)	1
M6 ねじ	20
M6 ねじ用ナット	20

プロジェクタ固定装置	
材料名	使用数
GFun フレーム N 650mm(切断加工)	4
GFun フレーム N 600mm	6
GFun フレーム N 450mm	4
Gfun マルチコネクタインナー	20
Gfun φ50N キャスター	4
Gfun ボードホルダ	14
アルミ板(5×400×640)	1
アルミ板(5×150×400)	1
アルミ板(5×150×200)	2
M6 ねじ	20
M6 ねじ用ナット	20

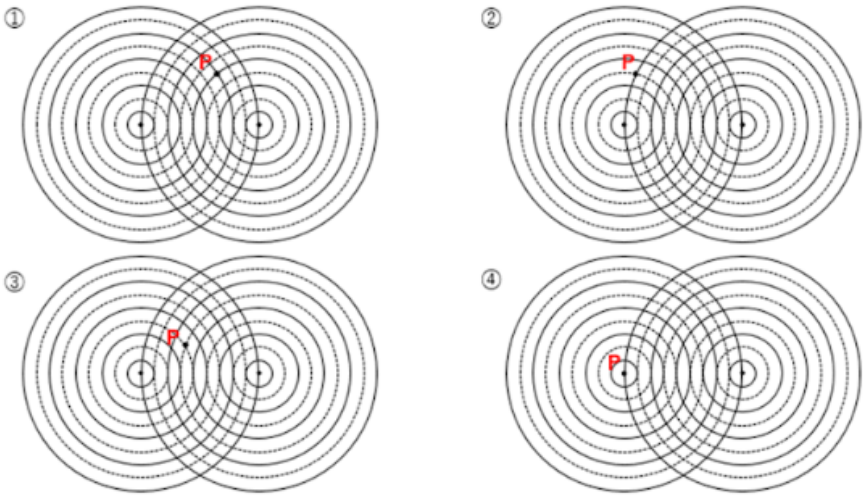
固定装置の CAD 図面 *新海大志*




4
3
2
1
F
E
D
C
B
A
689
200
739
400
150
518
名称：
プロジェクター固定装置
A4
尺度：1:20
SOLIDWORKS 教育用製品 (教育用目的でのみ使用可)

評価実験で使用した問題 *新海大志*

図の点Pにおいて、強めあいの点になっているのはどれか。なお、実線は山、波線は谷を示しているとします。 *



- ○ ①
- ○ ②
- ○ ③
- ○ ④

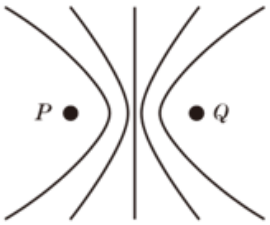
波の干渉において、波同士が強めあう条件を選んでください。ただしl_1, l_2は観測点から波源までの距離、λは波長、mは0,1,2,...の定数とします。 *

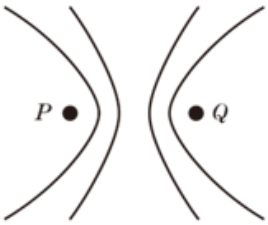
- ○ $|l_1 - l_2| = m\lambda$
- ○ $|l_1 - l_2| = (m+1/2)\lambda$
- ○ $|l_1 + l_2| = m\lambda$
- ○ $|l_1 + l_2| = (m+1/2)\lambda$

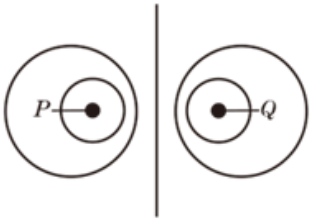
波の干渉において、波同士が弱めあう条件を選んでください。ただしl_1, l_2は観測点から波源までの距離、λは波長、mは0,1,2,...の定数とします。 *

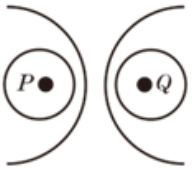
- ○ $|l_1 - l_2| = m\lambda$
- ○ $|l_1 - l_2| = (m+1/2)\lambda$
- ○ $|l_1 + l_2| = m\lambda$
- ○ $|l_1 + l_2| = (m+1/2)\lambda$

波同士の強めあいの点を結んだものを示してるものを選んでください。

① 

② 

③ 

④ 

- ○ ①
- ○ ②
- ○ ③
- ○ ④

Interactive Floor Interface を用いた教育支援システムの開発

指導教員　金丸 隆志　教授
S5-15006　赤坂 総司　S5-16010　井上 海翔　S5-16012　大野 智哉
S5-16013　大原 広暉　S5-16038　新海 大志　S5-16040　新屋 吉昭

1. 緒論

近年、日本の中等教育現場では、情報社会化に伴って ICT 機器の導入が進められている。2018 年には 90%を超える中学校がプロジェクタを導入している。また、新学習指導要領では生徒参加型授業が推進されており、生徒参加型授業の方が座学学習よりも生徒の理解度が高いという研究結果も出ている。以上を踏まえて、我々は Interactive Floor Interface を用いた教育支援システムの開発を目指した。

2. Interactive Floor Interface

床面に影のない映像を投影し、人の立ち位置とジェスチャーを認識できる Interactive Floor Interface（以下 IFI）を開発した。IFI の概要図を図 1 に示す。IFI は Kinect v2（以下 Kinect）、プロジェクタ 2 台、Kinect・プロジェクタを固定する装置と PC の 5 つで構成されている。

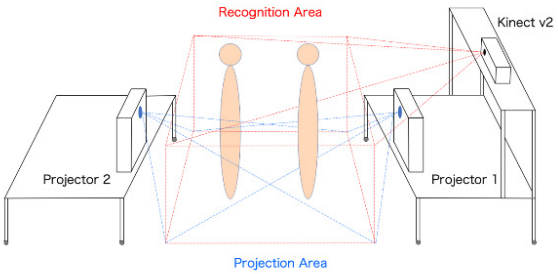


図 1　IFI 概要図

Kinect の認識領域と 2 台のプロジェクタの投影領域を再現性よく一致させるため、Kinect・プロジェクタ固定装置の製作を行った（図 2）。

図 2　Kinect・プロジェクタ固定装置

3. 教育コンテンツ：〈水面波の干渉アニメーション〉

ユーザーは本コンテンツで以下の内容を学習できる。

- 水面波の強め合いの公式：$|L_A - L_B| = n\lambda$
- 水面波の弱め合いの公式：$|L_A - L_B| = (n + ½)\lambda$
- 干渉縞

Distance モード、Real モードと Moiré モードの 3 つが本コンテンツに存在し、ジェスチャーで各モード間を遷移できる（図 3）。Distance モードでは人の位置を波源として波が発生し、波の強め合い・弱め合いを学習できる。また、ジェスチャーによって波の周波数変更と波の一時停止ができる。Real モードは Distance モードの背景を変えて現実の波に近づけた。Moiré モードは他 2 つのモードと波紋の表現が変わり、干渉縞を学習できる。ただし、Distance モードと Real モードはユーザーが 1 人または 2 人で利用できるが、Moiré モードは 2 人でのみ利用できる。

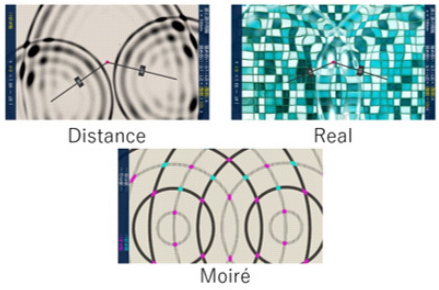


図 3　水面波の干渉アニメーション

4. IFI を用いた教育支援システムの評価

実験協力者 20 人に水面波の干渉のテストを事前に解いてもらった。その後、10 人は教科書で、残りの 10 人は本システムで学習してから再度同じテストを解いてもらった。その結果、両グループともに学習後の平均点の方が高かった。t 検定により両グループの平均点推移には有意差があることが示された。ただし、２つの学習方法の違いには有意差を見いだせなかったため、今後システムを改善することでより優れた教育システムとする必要がある。

5. 結論

Interactive Floor Interface を用いた教育支援システムを開発することができた。今後の展望は、他の単元に対する教育アプリケーションの教育効果を検証することである。

＜参考文献＞

日本教育情報化振興会, “第 11 回教育用コンピュータ等に関するアンケート調査”, 2018.