タンジブルなロボットを導入したテーブルトップ環境における直感的なプログラミング手法の提案

藤田智樹[†] 米海鵬[†] 杉本雅則[†]

本稿では，プログラミングの知識や経験が乏しいユーザでも簡単にロボットプログラミングが行える手法を提案する．提案手法は，物理的なロボットをマルチタッチ入力可能なテーブルトップ環境に導入することにより，ロボットへの直接的な入力のみでプログラミングを可能にする．プログラミング経験の無い大学生を被験者とする評価実験を通して，提案手法により被験者がロボットの動きを簡単にプログラムで表現できるようになるなど，直感的なプログラミングを行えることが示唆された．

An Intuitive Programming Technique Using Tangible Robots on Tabletop Environments

Tomoki Fujita,[†] Haipeng Mi[†] and Masanori Sugimoto[†]

We propose an intuitive robot programming technique for supporting users with less programming knowledge or experiences. We use a tabletop environment that can recognize multi-touch input and track physical robots simultaneously. Evaluations of the proposed technique were conducted with nine university students who did not have programming experiences. The results indicated that the proposed technique allowed users to create a program easily and in an intuitive manner.

1. 背　　景

近年，IT（情報技術）の発展に伴い，教育の現場にコンピュータを用いた電子教材の導入が盛んに行われている．これまでに提案されている電子教材の１つとして，コンピュータを用いたシミュレーションが挙げられる．学習者は，シミュレーションを通して環境問題や都市計画など実世界の問題を学ぶことができる．これらの問題に対する解決策をシミュレーション環境上で試行錯誤的に探索することを通して，学習者の知識や理解の深化を促進する効果が報告されている[1)]．さらに，物理的なオブジェクトやロボットを導入することにより，学習者の学習への動機づけや興味のレベルを高められることが示されている[3)]．我々は，シミュレーションに代表される学習コンテンツのデザインや構築を，学習者自身で行うことができる物理的な環境を通して，学習者の自発的な学習を促進することを目指している[2)]．そのためには，シミュレーション環境で用いる物理的なロボットのプログラミングを学習者自身が行える必要がある．このようなロボットプログラミングでは，コンピュータの操作に加え，周囲の状況に応じた振る舞いを実現するための条件分岐や繰り返しなどの概念を理解することが学習者に要求される[4)]．よって，プログラミングの知識や経験が乏しい学習者でも，容易にロボットプログラミングを行えるような支援環境が必要となる．

そこで我々は，物理的かつ直接的な入力のみでロボットの振る舞いをプログラミングできる手法を提案する．本稿では，マルチタッチ入力とロボットトラッキング可能なテーブルトップ環境である RoboTable[10)] を拡張することで，提案手法を実現した．ユーザは，テーブル上でロボットを掴んで動かす，ロボットの周囲に表示されるアイコンを指で触れるなどの直感的な操作で，条件分岐や繰り返しを含むプログラミングを行うことができる．よって，例えば小学校低学年の子どもやプログラミングを普段行わない大学生など，プログラミング経験の乏しい学習者でも容易にロボットプログラミングを行えると考える．これまで，タッチパネルや TUI(tangible user interface) 等のユーザインタフェースを導入した初心者向けのプログラミング手法は多く提案されている[4)~6)]．しかし，物理的なロボッ

† 東京大学
The University of Tokyo

トを活用し，状況に応じた振る舞いをさせるプログラムを容易に構築できるよう支援する手法の提案は，我々の知る限りほとんど存在しない．

評価実験では，グラフィカルなブロックを操作してプログラミングを行うことができる環境を構築し，提案手法と比較した．プログラミングの経験が無い大学生に被験者として参加してもらい，実験後のアンケートへの回答を依頼するとともに，利用中に撮影したビデオの分析を行った．その結果，提案手法が被験者にとってより直感的で容易なロボットプログラミング手法であることが示唆された．

本稿の構成は以下の通りである．第２節では本研究の関連研究を示す．第３節では提案するプログラミング手法について述べる．第４節では実験の詳細および結果について述べる．第５節では評価実験で得られたコメントとビデオの分析を基に，提案するプログラミング手法について考察する．第６節では本稿の結論と今後の展開について示す．

2. 関 連 研 究

プログラミング経験の乏しい初心者を対象としたプログラミング手法は，数多く提案されている．ここでは本稿と関連の強い研究をいくつか紹介する．Scratch[4] は，グラフィカルなプログラミング環境の１つである．Scratch ではブロック状のオブジェクトをマウスで操作し，積み重ねることでコードを記述せずに簡単にプログラミングすることができる．Quetzal[5] は小学校低学年の子供を対象とし，TUI を導入したプログラミング環境である．子どもたちはマーカーが張り付けられたブロックを直接手で掴み，組み立てることでプログラミングを行うことができる．Quetzal では flow-of-control chains という独自のブロック接続手法を用いることで，ループや条件分岐などを含むプログラムを作成することが可能である．Galllardo らは，マルチタッチ入力可能なテーブルトップ環境に TUI を導入した TurTan[6] と呼ばれるプログラミング環境を提案している．TurTan ではテーブル上に配置したタンジブルなブロックを掴んで移動することで，テーブル上に表示されたバーチャルなタートルの経路をプログラミングすることができる．TurTan では入出力が統一された環境でバーチャルなタートルのプログラミングが可能であるが，条件分岐のような複雑なプログラムを作成することはできない．

Horn らは，TUI を導入したプログラミング環境を，コンピュータ上のブロックをマウスで操作するプログラミング環境と比較している[7]．彼らは，博物館に来館した子どもたちが作成したプログラムの数やコード長，および子どもの振る舞いの観察を基に分析を行っている．その結果，TUI をプログラミング環境に導入することで，子どもをプログラミング環境に惹きつけ，自発的に協調させるという効果があると述べている．しかし一方で，実体を持つブロックで作成したプログラムとロボットの振る舞いが関連付けにくいという問題や，実体を持つブロックでは長く複雑なプログラムを作ることが難しいという問題が指摘されている．我々が提案するプログラミング環境では，入出力機能を持つロボットを直接掴んでプログラミングすることができ，さらに作成したプログラムを仮想的なオブジェクトとして表現することができる．そのため，これらの問題を解決できると考える．

Frei らの提案する curlybot[8] では，ロボットを直接掴んで動かしその動きを再現できる．そのため，ユーザの入力とロボットの出力を関連付けやすいという特徴を持つ．Raffle らの開発した Topobo[9] では，ユーザが様々な形状のパーツを組み合わせロボットを構築する．ユーザは，ロボットを手で掴んで動かすことにより，その動きを記録，再生できる．しかし，curlybot や Topobo ではロボットが周囲の状況に応じて振る舞いを自動的に変化させるプログラムを作成することはできない．

3. 提案するプログラミング手法

3.1 テーブルトップ環境

物理的なロボットを導入したプログラミング環境の構築にあたり，マルチタッチ入力とテーブル上の物体をトラッキングすることができる RoboTable[10] と呼ばれるテーブルトッププラットフォームを拡張した．RoboTable は，既存のマルチタッチ入力認識手法である DI(Diffused Illumination)[11] と FTIR(Frustrated Total Internal Refrection)[12] を組み合わせることで，テーブル上の指の接触と，ロボットに取り付けたマーカーを同時に認識できる．テーブル上の入力情報は，reacTiVision[13] と呼ばれる画像認識ライブラリにより取得される．テーブル上のロボットには Bluetooth モジュールが実装されており，RoboTable は Bluetooth を介してそれぞれのロボットに命令を送ることができる．

入力情報を利用してグラフィックの描画を行うために，RoboTable は MT4j[14] と呼ばれるマルチタッチライブラリに加え，オブジェクトの物理的な衝突を表現するために Java JBox2d[15] を組み込んでいる．さらに，異なる駆動機構や無線通信手法を備えたロボッ

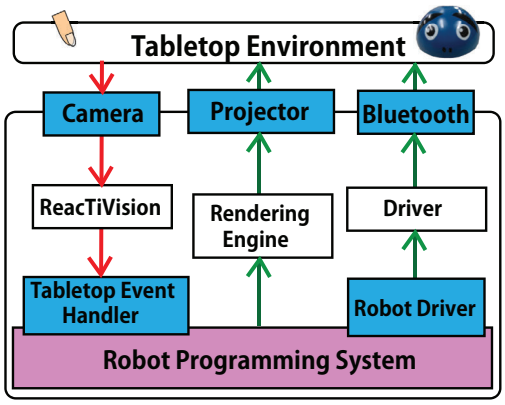

図 1 RoboTable のシステム構成
Fig. 1 System configuration of RoboTable

トに対応するために，独自のシステム開発 Toolkit を実装している[16]．拡張された RoboTable のシステム構成を**図 1** に示す．

3.2 イベント駆動型プログラム

本稿では，プログラムモデルとしてイベント駆動型プログラムを採用する．イベント駆動型プログラムは個々のイベントと，それに対するロボットの振る舞いの組み合わせで構成される．本稿ではこの組み合わせをプログラムブロックと呼ぶことにする．プログラムブロックは RoboTable に登録され，RoboTable はテーブル上の状況と登録されたプログラムブロックとの照合を行い，個々のロボットに送信する命令を切り替える．例えば，「障害物の接近の認識（イベント）」と「右に 90 °回転して回避（振る舞い）」からなるプログラムブロックが該当すれば，ロボットに障害物を回避させることが可能になる．本稿では，作成したプログラムブロックの集合をプログラムとして扱う．

3.3 プログラミング手法

RoboTable を用いることで，テーブル上のロボットには**図 2** に示す直接的な入力を行うことが可能である．

- ロボットを直接掴んで動かす
- ロボットの周囲に表示されるボタンなどのバーチャルなオブジェクトに触れて入力する
- マルチタッチジェスチャで入力する

提案するロボットプログラミング手法では，ユーザはこれらの入力操作を組み合わせることによりプログラミングを行う．

3.3.1 認識イベントの保存

テーブル上のロボットは，他のオブジェクトへの接近，衝突を RoboTable を介してイベントとして認識することができる．本稿では，これを認識イベントと呼ぶ．認識イベントの保存には，あらかじめロボット

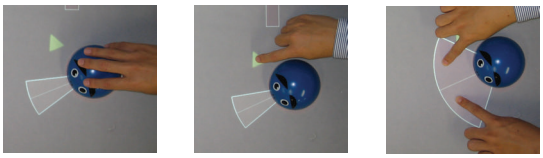

図 2 RoboTable 上の入力手法
Fig. 2 Input techniques on RoboTable

が認識可能な領域を RoboTable に登録しておく必要がある．そこで，認識領域としてロボットの正面に扇型のオブジェクトを配置する．この扇形の領域は，マルチタッチジェスチャ入力でユーザが半径と角度を自由に調整できる (図 2 右)．RoboTable は，個々のロボットに割り当てられた認識領域とテーブル上のオブジェクトの接触を定期的に確認する．認識領域にオブジェクトが接触すれば，接触したオブジェクトと，接触した方向（ロボットから見て左，正面，右）の情報を用いて認識イベントを作成する（例えば，バーチャルな壁をロボットの正面に来るように配置し，そこにロボットを近づけて認識領域と接触させれば「壁：正面」という認識イベントが作られる）．このとき，発生したイベントを保存するためのボタンがロボットの周囲に表示される．ユーザがこのボタンに触れることで，プログラムブロックのテンプレートが表示され，その中に認識イベントを保存できる (**図 3**)．認識イベント発生時に，複数のオブジェクトがロボットの認識領域内に入ることも考えられる．このときは自動的に一番近くにあるオブジェクトのみを選択し，それに対する認識イベントを作成する．なお，複数のオブジェクトを同時に認識して認識イベントを生成することも可能であるが，ユーザのプログラミング経験が乏しいことを考慮して，より単純な機能を実装することとした．

また，認識イベントに対し，認識可能領域に何もオブジェクトが存在しない状態を通常状態と定義する．通常状態のプログラムブロックもロボットの周囲に表示されるボタンから作成できる．プログラムの実行時には，他のイベントが発生しない限り，通常状態のプログラムブロックが繰り返し呼び出される．

3.3.2 振る舞い情報の保存

プログラムブロックのテンプレートに認識イベントが保存されると，ロボットの周囲にはそのイベントを保存したプログラムブロックが表示される．この状況でユーザがロボットを手で掴んで動かすことにより，そのイベントに対応するロボットの振る舞い情報を作成できる．振る舞い情報は，ユーザが動かしているロボットの位置と角度情報を読み取りながら，直進，後

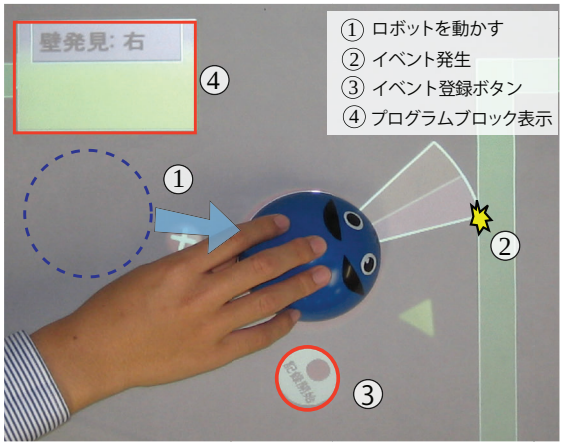

図 3 認識イベントの保存
Fig. 3 Storage of recognition events

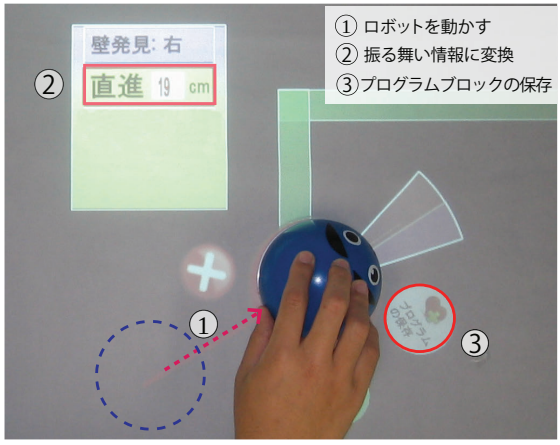

図 4 振る舞いの登録
Fig. 4 Registration of robot behaviors

退，回転の命令に変換することで RoboTable により生成される（例えば，ロボットを掴んで正面方向に動かした場合，プログラムブロックには直進 30cm という振る舞い情報が生成される）．ユーザはロボットを動かしながら，プログラムブロック内の変換された振る舞い情報を確認する．意図した通りの情報が入力されていれば，ロボット周囲に表示される「振る舞い情報保存ボタン」に触れることで，作成された振る舞い情報を保存する．振る舞い情報が保存されると，RoboTable にプログラムブロックとして登録することが可能になる．このとき，ロボットの周囲にはプログラムブロックを登録するためのボタンが表示される（**図 4**）．ユーザはこのボタンに触れることで，作成したプログラムブロックを RoboTable に登録する．その後，同様にロボットを動かして登録ボタンを押すという作業を繰り返すことで，新たなプログラムブロックを登録することができる．

3.3.3 プログラムブロックの表示と編集

ユーザは，RoboTable に登録したプログラムブロックを編集することができる．ロボットの周囲に表示されるメニューボタン中のプログラムブロック編集ボタンに指で触れることで，RoboTable に保存されている全てのプログラムブロックをロボットの周囲に表示することができる．ユーザは，各々のプログラムブロックを指で触れて移動させたり，テーブルの隅に表示されるゴミ箱のアイコンにドラッグすることで作成したプログラムブロックを削除したりできる．

3.3.4 プログラムの実行

ユーザがロボットの周囲に表示されているプログラム実行ボタンに触れると，RoboTable は登録されたプログラムブロックを用いて Bluetooth 通信でロボットを操作する．ロボットの認識領域にオブジェクトが接触し認識イベントが発生すると，その都度登録されているプログラムブロックとの照合を行う．このとき，一致するプログラムブロックが存在すれば，そこから振る舞い情報を取り出し，その情報を基にロボットを操作する．一致するイベントが登録されていない場合，通常状態の振る舞い情報が用いられる．プログラム実行時には，ロボットの周囲にプログラム停止ボタンが表示される．このボタンに触れることで，プログラムの実行が停止する．

3.4 提案手法のシステム構成

図 5 に示すように，提案システムは Tabletop Event Handler からロボットや指などの位置情報を取得し，RobotDriver 経由でロボットの制御を行う．Tabletop Event Handler からからの入力情報は Event Processor で解析され，認識イベントやボタン入力イベント等の特定を行う．特定されたイベントに応じて，Program Block Generator，Robot Controller に処理が移される．各々の処理について以下に述べる．

- プログラムブロックの生成

Event Processor がロボットの認識イベントを同定すると，Program Block Generator により「認識イベント保存ボタン」が表示される．ユーザがこのボタンに触れると，Event Generator が起動され認識イベントとして保存される．次に，ユーザはロボットをテーブル上で動かすことで，保存された認識イベントに対応するロボットの振る舞いを入力する．入力された振る舞いは Behavior Recorder によりロボットの駆動機構（回転、直進など）に応じた命令に変換される（これを振る舞い情報と呼ぶ）．意図通りの動きを入力できたと判断したユーザが「振る舞い情報保存ボタン」に触れると，認識イベントと振る舞い情報が Program Block Assembler に送られる．ユーザは，ロボットを

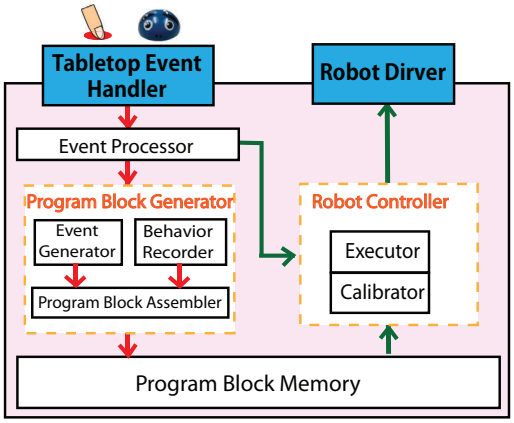

図 5 提案手法のためのシステム構成
Fig. 5 System configuration of the proposed method

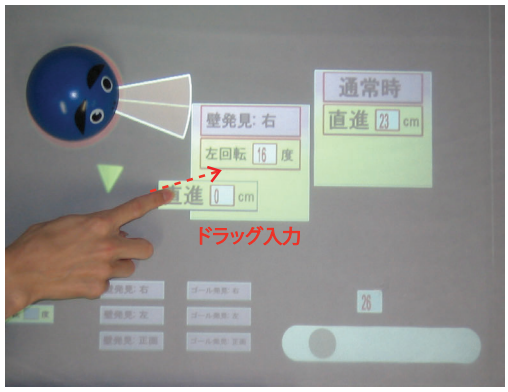

図 6 グラフィカルなプログラミング環境
Fig. 6 Graphical programming environment

動かしながら繰り返し振る舞い情報の入力を行う．より意図通りに近い入力ができたと判断し，ユーザが「振る舞い情報保存ボタン」に触れるたびに，新たな振る舞い情報に更新される．

- プログラムブロックの登録

Program Block Assembler は，受け取った認識イベントと振る舞い情報からプログラムブロックを構成し，テーブル上に「プログラムブロック登録ボタン」を表示する．このボタンに触れると，Program Block Memory にプログラムブロックを送る．Program Block Memory は受け取ったプログラムブロックを格納する．その際，すでに格納済みの認識イベントを確認し重複する場合は上書き処理を行う．

- プログラムブロックの実行

Event Processor が「プログラム実行ボタン」からの入力を確認すると，Robot Controller に処理を移行する．Robot Controller は Program Block Memory にアクセスし，プログラムブロックとして格納されている振る舞い情報を実行する．Event Processor は新たな認識イベントが発生するたびに，Robot Controller にそのイベントの情報を送信する．Robot Controller は受信したイベントと Program Block Memory に格納されているプログラムブロック中の認識イベントを照合し，一致する認識イベントの振る舞い情報を Executor に実行させる．ロボットの動きはトラッキングされるため，指定された振る舞い情報からの誤差を取得できる．Calibrator は，この誤差を補償するために補正された振る舞い情報を Executor に送信する．

4. 評価実験

4.1 実験設定

提案するプログラミング手法を評価するための実験を行った．本稿では比較実験を行うため，プログラミングの知識や経験が乏しいユーザを対象に構築された Scratch[4] を参考に，グラフィカルなブロックを指で操作して組み合わせながらプログラミングを行う環境を，提案手法と同様のテーブル上に構築した．ロボットを掴んでその振る舞いを設定できる提案手法とは異なり，このグラフィカルなプログラミング環境では振る舞いを設定するパラメータを**図 6** 右下のスライダーに触れて設定する．

評価実験のタスクとして，ロボットがバーチャルな障害物を回避しながらゴールを探索する maze(**図 7**) を採用した．被験者は，グラフィカルなプログラミング環境および提案手法の 2 種類のプログラミング環境でプログラミングを行う．実施された評価実験は以下の 2 つである．

- 実験 1：バーチャルなロボットを用いた評価実験.
- 実験 2：物理的なロボットを用いた評価実験.

これらの 2 つの実験を行うのは，物理的なロボットの動作に含まれる誤差（モータやテーブル表面の状態により必ずしも動きを正確に再生できるとは限らない）が，ユーザの評価に影響を与える可能性があるためである。

実験に用いられる maze には同程度の難易度のマップが 4 つ用意されており，それぞれのマップのスタートとゴールの位置が固定されている．ロボットがゴールに接触すると，自動的にロボットは停止し，画面上に GOAL の文字が大きく表示され，タスクが終了する．

評価実験では，プログラミング経験の無い大学生 9

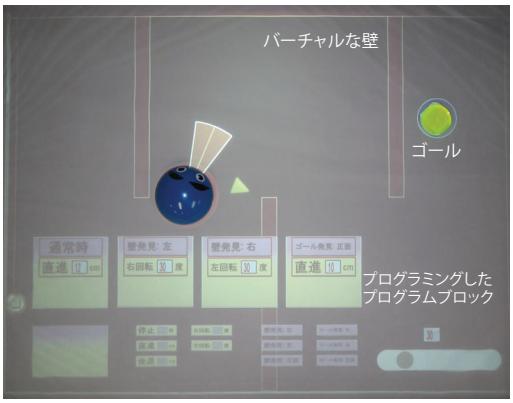

図 7 評価実験に用いた maze
Fig. 7 Maze used in experiments

名（男性 6 名，女性 3 名，平均 21.4 歳）を我々の研究室以外から募集した．被験者には謝礼として 1 人 1,000 円を支払った．まず最初に，実験者が被験者に対しグラフィカルなプログラミングおよび提案するプログラミング手法の説明を行った．被験者がタスクの内容を理解しプログラミングに十分慣れた後，実験を開始した．

実験では各被験者に対し，２種類のプログラミング手法と maze のマップをランダムに選択して提供した．ロボットがゴールに到達，もしくはプログラミング開始から 10 分以上経過した時点でタスクを終了し，各プログラミング手法に関するアンケートに記入してもらった．実験中はプログラミングに要した時間を測定するとともに，手の動きを中心にビデオ撮影を行った．被験者が回答したアンケート項目を以下に示す．

- プログラミング手法の理解しやすさ
- イベントの設定しやすさ
- 振る舞いの組立てやすさ
- パラメータの入力しやすさ
- 実際のロボットの動きの想像しやすさ

アンケートでは，以上の５項目を７段階のリッカート尺度 (1:全く当てはまらない，2:当てはまらない，3:やや当てはまらない，4：どちらでもない，5:やや当てはまる，6:当てはまる，7:非常に当てはまる) で評価してもらった．「実際のロボットの動きの想像しやすさ」は，イメージしたロボットの動きを表現するためには，どのようなプログラムを作成するか，その想像しやすさを評価するための項目である．アンケートではさらに，プログラミング手法に関する問題点や気づいた事などをコメント欄に記入してもらった．

4.2 実験１の結果

実験１では，テーブル上に表示されるバーチャルなロボットに対するプログラミングを被験者に行ってもらった．アンケートで得られたデータに対し，Kolmogorov-Smirnov 検定および F 検定を行い，それぞれ正規分布および等分散の仮説が棄却できないことを確認した．次に，グラフィカルなプログラミングおよび提案するプログラミング手法について t 検定 (対をなすデータの場合) を行った．

実験結果を**図 8** に示す．t 検定により「実際のロボットの動きの想像しやすさ」の項目において，$t(8) = 3.833\,(p < .05)$ となり，有意差があることが確認できた．この項目において有意差が生じた理由は，以下のように考えられる．グラフィカルなプログラミング手法では，ロボットの振る舞いをイメージしそれを具体的なパラメータの値に変換してプログラミングを行う必要があった．一方，提案手法ではユーザがロボットに対して直接行う入力がそのままロボットの振る舞いとなるため，より直感的なプログラミングが可能となったと考えられる．

さらに，提案手法では一部の被験者は手で直接ロボットを動かし，認識領域を maze の壁の接触に合わせて動きを切り替えながら，複数のプログラムブロックによって表現されるロボットの動きをシミュレーションしていた．提案手法の特徴は，プログラム作成時のユーザのロボットへの振る舞いの入力がそのまま実行時にロボットの振る舞いとして再現できる点である．よって，提案手法では作成したプログラムでロボットが意図通りの振る舞いをするかどうか，意図通りでない場合はどのように修正すればよいかを容易に知ることができたと考えられる．

それぞれの手法において，被験者がプログラミングを完了するのに要した時間を表 1 に示す．提案手法を用いることにより，半数以上の被験者のプログラミング時間が短縮されている．しかし，2 名の被験者（No.5，No.9）についてプログラミングに要する時間が明らかに増大していることが分かる．

表 1 被験者が実験１のプログラミングに要した時間（単位：秒）
Table 1 Time spent for programming by each subject in Experiment 1 (unit: second)

	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8	No.9
Graphical	213	195	130	107	223	169	163	148	118
Proposed	232	130	149	97	390	123	152	60	289

4.3 実験２の結果

実体を持つロボットを用いて行った実験結果を**図 9** に示す．実験１と同様，正規分布および等分散の仮説が棄却できないことを確認し，t 検定を行った，図 9 から明らかなように，提案したプログラミング手法の得点平均はいくつかの評価項目においてグラフィカル

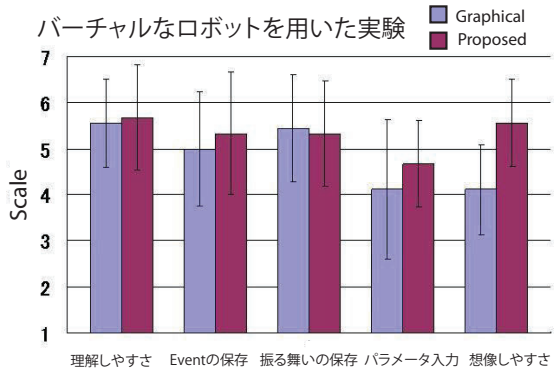

図 8 実験 1 の結果
Fig. 8 Result of Experiment 1

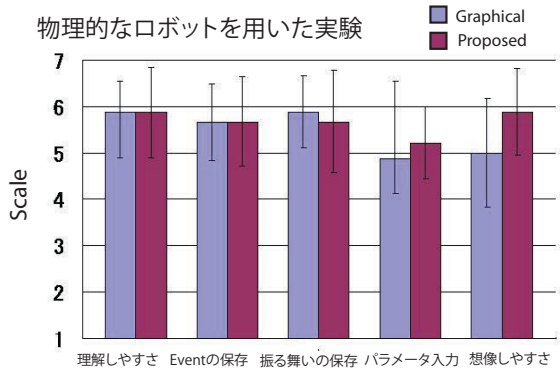

図 9 実験 2 の結果
Fig. 9 Result of Experiment 2

なプログラミング手法を上回っていたが，全ての項目について有意差は見られなかった．実験 1 の結果との大きな違いとして「実際のロボットの動きの想像しやすさ」の評価項目で有意差が見られなかった点が挙げられる．この原因として，プログラム作成時に入力したロボットの動きと実際のロボットの動きの違いが考えられる．バーチャルなロボットの場合と異なり物理的なロボットの場合，常に設定された振る舞いを正確に再現できるとは限らない．そのため，一部の被験者は作成したプログラムと実際のロボットの動きの違いに違和感を感じたと述べていた．このことが，評価結果に影響を与えたと推測される．

それぞれの手法において，被験者がプログラミングを完了するのに要した時間を表 2 に示す．実験 1 と同様に，提案手法により半数以上の被験者のプログラミング時間が短縮されている．しかし，3 名の被験者（No.1, No.4，No.5）のプログラミング時間が明らかに増大していることが分かる．

表 2 被験者が実験 2 のプログラミングに要した時間（単位：秒）
Table 2 Time spent for programming by each subject in Experiment 2（unit: second）

	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8	No.9
Graphical	100	116	220	100	103	107	163	71	78
Proposed	179	83	195	164	222	53	119	54	96

5. 議　　論

評価実験の結果からは，提案するプログラミング手法によりプログラミングの知識や経験の乏しいユーザでもロボットプログラミングを概ね容易に行うことができたと言える．しかし一方で，提案手法の問題点が被験者からのコメントやビデオ分析を通して明らかになった．以下に，いくつかの問題点を示す．

- 振る舞いの登録に関する問題点

一部の被験者はロボットの振る舞いの登録において「少し触れただけで回転命令が作成されてしまうことにストレスを感じる」というコメントした．我々はこの問題の原因が，提案したシステムによる，振る舞い情報の変換にあると考えている．提案手法は，物理的なロボットを動かして振る舞い情報を入力するために，ロボットの位置や角度の変化量を読み取り，振る舞い情報への変換を行っている．そのため，ユーザがロボットを掴んだ瞬間の動きや，ロボットを正面方向に動かしたときに発生するロボットの微妙な回転まで，振る舞い情報として入力されてしまう．これにより，ユーザは意図していない振る舞い情報が出力されないように慎重にロボットを動かすことが要求され，ストレスを感じたと推測できる．事実，ロボットを正面方向に移動させた際，回転命令が含まれていることに気付いた被験者が，ロボットの移動による入力をやり直す様子がビデオで確認された．よって，ユーザの意図しない動きが含まれないようにするためのフィルタリング手法の導入の可否などについて，今後検討する必要がある．

- 情報提示法に関する問題点

提案手法を用いた場合，複数の被験者から「角度の細かい調整にストレスを感じた」というコメントを得た．我々はこの問題の 1 つの原因が，提案手法の情報提示法にあると考えている．提案手法ではロボットを用いて動きの入力を行う際に，ロボットの周囲にプログラムブロックの情報を表示することで，プログラミング中の認識イベントや振る舞い情報を確認できる．しかし，この情報提示ではロボットを動かしてパラメータを調整する際に，操作対象のロボットではなくロボットの周囲に表示された情報に視点を移動させなければならない．「細かい調整」の際のこの視点移動の繰り

返しが，被験者のストレスとなったと考えられる．実験中に撮影したビデオの分析を行ったところ，問題点を指摘した被験者が視点移動を繰り返しつつ，ロボットを何度も左右に回転させながらパラメータの調整を行っている様子が確認できた．表 1 および表 2 において，プログラミングに要した時間が大幅に増大している被験者は，この調整を行っていたためである．よって，微調整を要するロボットの振る舞いを設定する際，より効率的かつストレスなく行える情報提示方法を検討する必要がある．

- グラフィカルなプログラミング手法との比較

グラフィカルなプログラミング手法では，ロボットの移動距離や回転角度を数値で正確に定めることができる．そのため，ロボットを用いた図形描画や，振る舞いを正確に定める必要があるシミュレーション環境などに適している．一方で，提案手法では，ユーザがイメージしたロボットの振る舞いを容易にプログラムとして生成できる．そのため，シミュレーションやゲーム等で, 物理的なロボットの定性的な振る舞いをより直感的なやり方でデザインするのに適していると言える．また，提案手法とグラフィカルなプログラミング手法を組み合わせることにより，両者の利点を生かしたプログラミング環境を実現することも考えられる．

6. まとめと今後の課題

本稿では，テーブルトップ環境を用いることによる直感的なロボットプログラミング手法を提案した．さらに，グラフィカルなブロックを組み合わせるプログラミング環境との比較実験を通して，提案手法の評価を行った．その結果，提案手法を用いることで，ユーザがイメージしたロボットの動きを簡単にプログラムで表現できるようになることが示唆された．さらに，得られた実験データを基に，提案手法の課題について考察した．今後は，提案手法の改善を行うとともに，学習者自身による学習用シミュレーション環境構築とその学習支援効果についての検証や，ゲームプログラミングへの応用についても研究を進めていきたい．

参 考 文 献

1) Yamaguchi, E., Inagaki, S., Sugimoto, M., et al.: Fostering Students' Participation in Face-to-Face Interactions and Deepening Their Understanding by Integrating Personal and Shared Spaces, Transactions on Edutainment II, pp.228-245 (2009).
2) Fischer, G., Sugimoto, M.: Supporting Self-Directed Learners and Learning Communities with Sociotechnical Environments, Journal on Research and Practice in Technology Enhanced Learning, 1(1), pp.31-64 (2006).
3) 神田: コミュニケーションロボットによる学習支援，人工知能学会論文誌，23(2) pp. 229-236 (2008).
4) Resnick, M., et al.: Scratch: Programming for All. Communications of the ACM, 52(11), pp. 60-67 (2009).
5) Horn, M. and Jacob, R.：Tangible Programming in the Classroom: A Practical Approach, Proc. of CHI 2006, pp.869-874 (2006).
6) Gallardo, D., Julia, C., Jorda, S.：TurTan: a Tangible Programming Language for Creative Exploration, Proc. of TABLETOP 2008, pp. 89-92 (2008).
7) Horn, M., Solovey , E.T., Crouser, J., Jacob, R.：Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education, Proc. of CHI 2009, pp.975-984 (2009).
8) Frei, P., Su, V., Mikhak, B. and Ishii, H.：curlybot: Designing a New Class of Computational Toys, Proc. of CHI 2000, pp.129-136 (2000).
9) Raffle, H., Parkes, A., Ishii, H：Topobo: A Constructive Assembly System with Kinetic Memory, Proc. of CHI 2004, pp.647-654 (2004).
10) Krzywinski, A., Mi, H., Chen, W. and Sugimoto, M.: RoboTable: A Tabletop Framework for Tangible Interaction with Robots in a Mixed Reality, Proc. of ACE 2009, pp.107-114 (2009).
11) FTIR vs DI (http://iad.projects.zhdk.ch/multitouch/?p=47/) (retrieved 2010.11.07)
12) Han, J.：Low-Cost Multi-Touch Sensing through Frustrated Total Internal Refl ection, Proc. of UIST 2005, pp.115-118 (2005).
13) Kaltenbrunner, M., Bencina, R.：reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction, Proc. of TEI 2007, pp.69-74 (2007).
14) MT4j Multitouch For Java (http://www.mt4j.org/mediawiki/ index.php/ Main_Page) (retrieved 2010.11.07).
15) JBox2D Demos (http://www.jbox2d.org/) (retrieved 2010.11.07).
16) 小野島, 米, 杉本, Krzywinski.：テーブルトップヒューマンロボットインタラクションのための Toolkit の設計と評価, 情処第 72 回全国大会予稿集 (2010).