

Integrated Graphical Representations
for Development of Programs
with Real-world Input and Output

実世界入出力を伴うプログラムの
画像表現を用いた開発支援手法

加藤 淳 <http://junkato.jp/>

東京大学大学院 情報理工学系研究科
コンピュータ科学専攻 五十嵐研究室

アウトライン

- イントロダクション
- 研究紹介
- ディスカッション
- 結論

アウトライン

- インTRODクシヨン
 - 研究の背景
 - 既存研究
 - 提案手法
- 研究紹介
- ディスカッション
- 結論

研究の背景

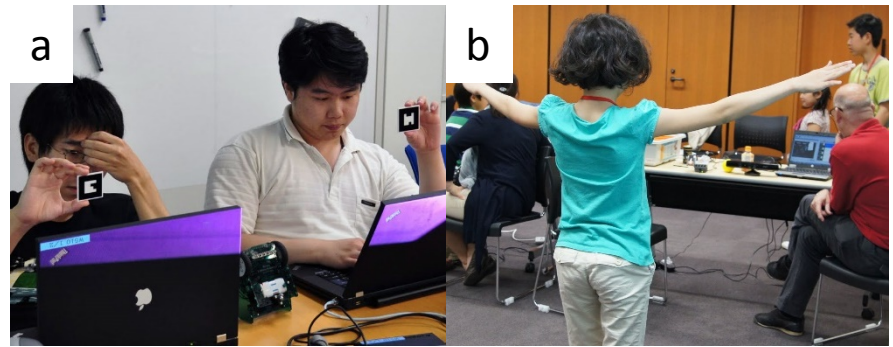
実世界入出力を伴うプログラムの増加

実世界**入力**

(センサの生データ)

a. RGBカメラ → 映像情報

b. 深度センサ → 姿勢情報



実世界**出力**

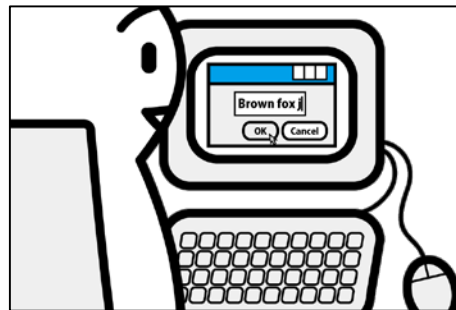
(アクチュエータの生データ)

c. 姿勢情報 → ロボット

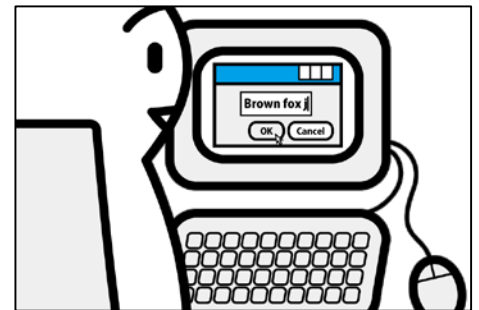


実世界入出力を伴うプログラム 開発の特徴 (1/2)

従来型入出力:

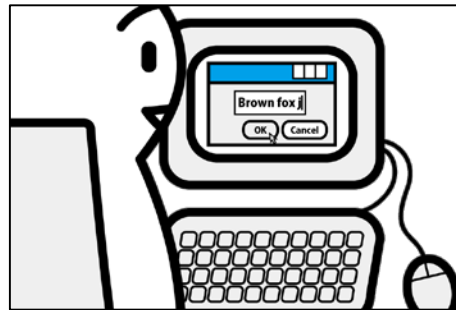


開発環境

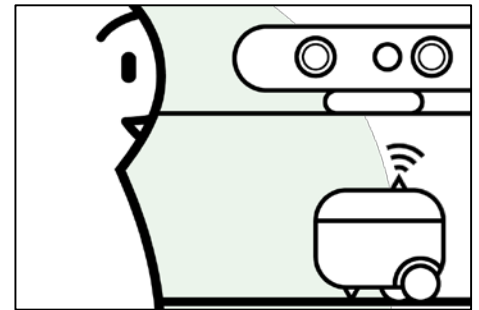


実行環境

実世界入出力:



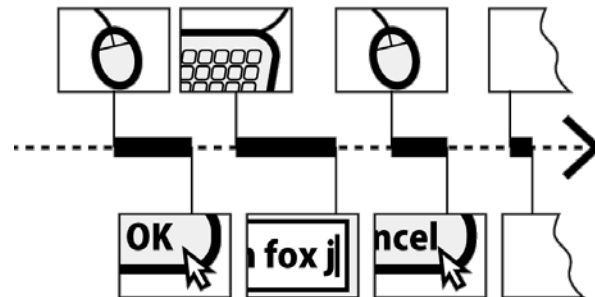
開発環境



実行環境

実世界入出力を伴うプログラム開発の特徴 (2/2)

従来型入出力:

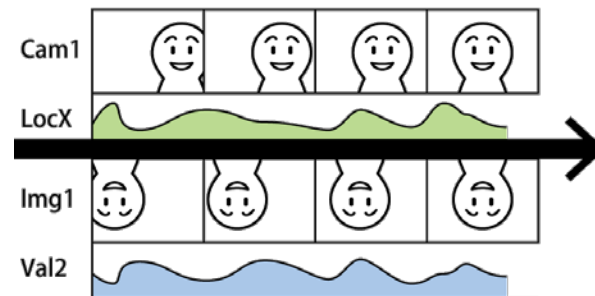


シンプルで断続的

numAbc	int, 26983
message	string, "Quick..."
isClicked	boolean, false

文字表記が容易

実世界入出力:



複雑で連続的

numAbc	int, 26983
message	string, "Quick..."
isClicked	boolean, false

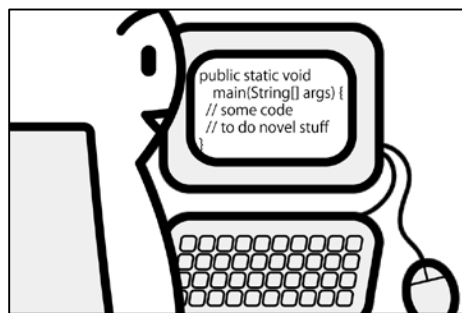
文字表記が困難

実世界入出力を伴うプログラム 開発の問題

既存の統合開発環境は、従来型の入出力を伴う
プログラム開発に最適化されている



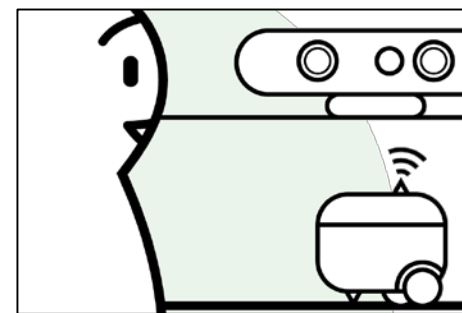
文字ベースのユーザインタフェースが多い



開発環境



実行結果を理解・
想像するのが困難

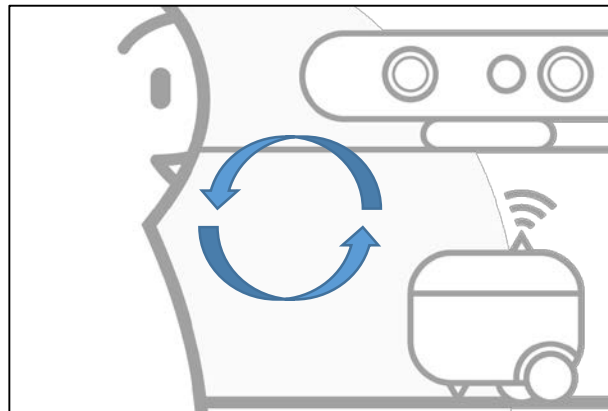


実行環境

既存手法

Programming by Example (PbE)

ユーザの動作を観察してプログラムを推論する手法



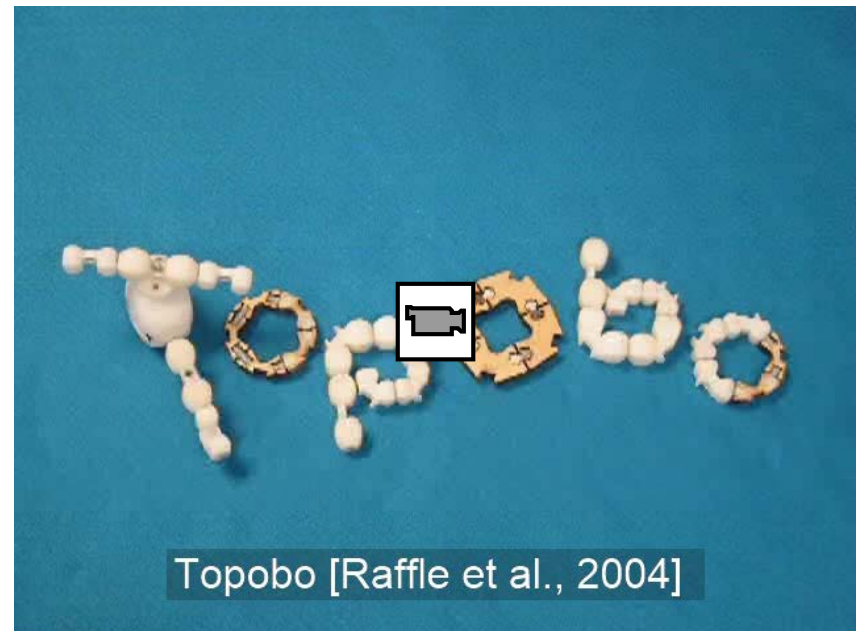
実世界(開発環境＝実行環境)

- 開発環境と実行環境が同一
- 何が起きるかその場ですぐ確認できる

Programming by Example: Topobo

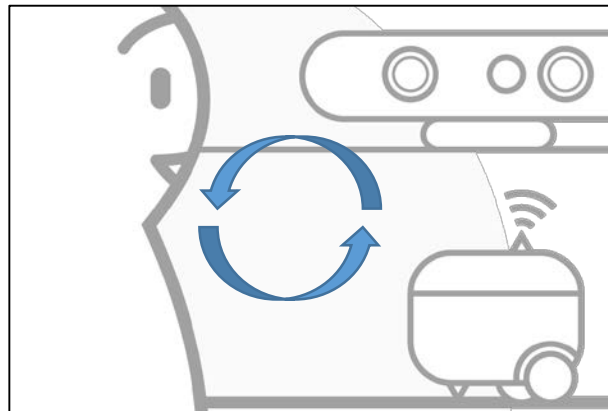
[Raffle et al., CHI '04]

- 関節を手で操作すると、動作を繰り返し再生
- 全身を組み上げて自由な動作を設計できる



Programming by Example

ユーザの動作を観察してプログラムを推論する手法



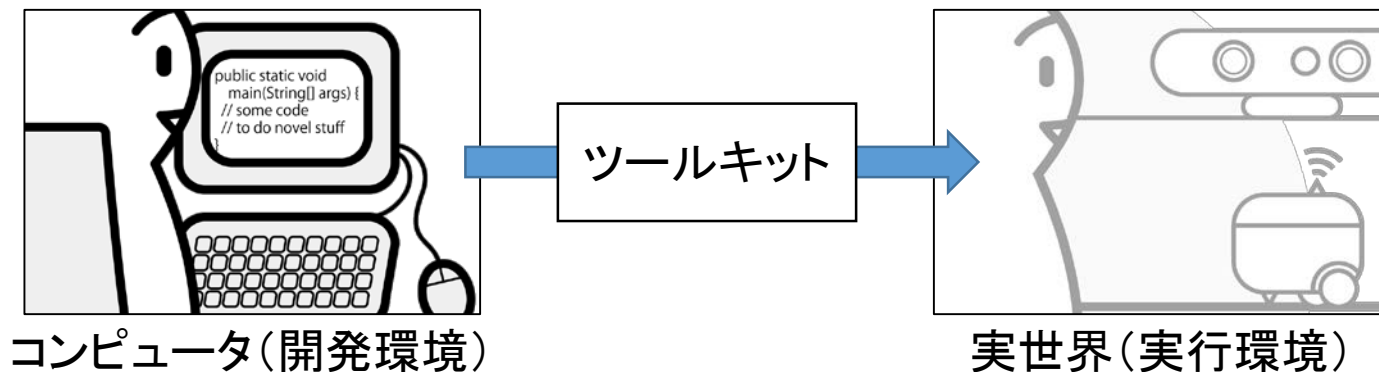
実世界(開発環境＝実行環境)

プログラムの設計をシステムの推論に頼る

➡ ロジックを精密に指定できない

ツールキット

アプリケーション開発に便利なユーザインタフェース (API) を提供する

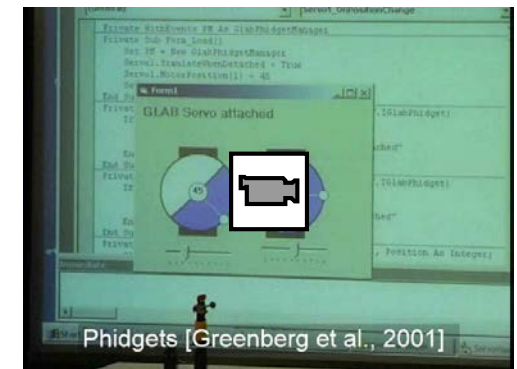
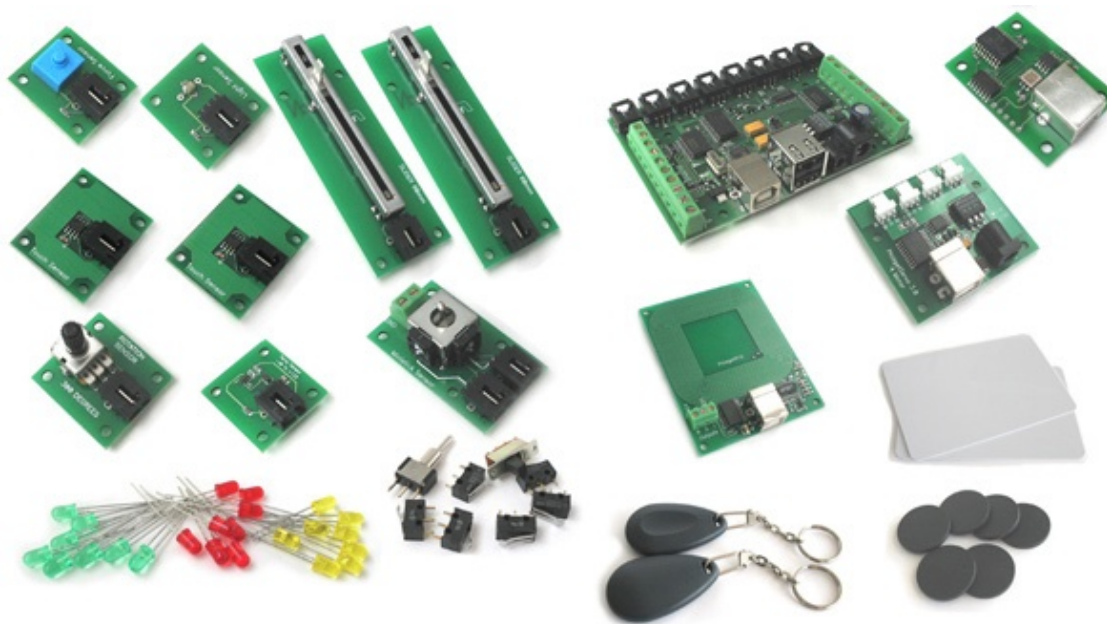


- 実行環境に対するアクセスを包み隠してくれる
- 開発環境に留まってコーディングできる

ツールキット: Phidgets

[Greenberg et al., UIST '01]

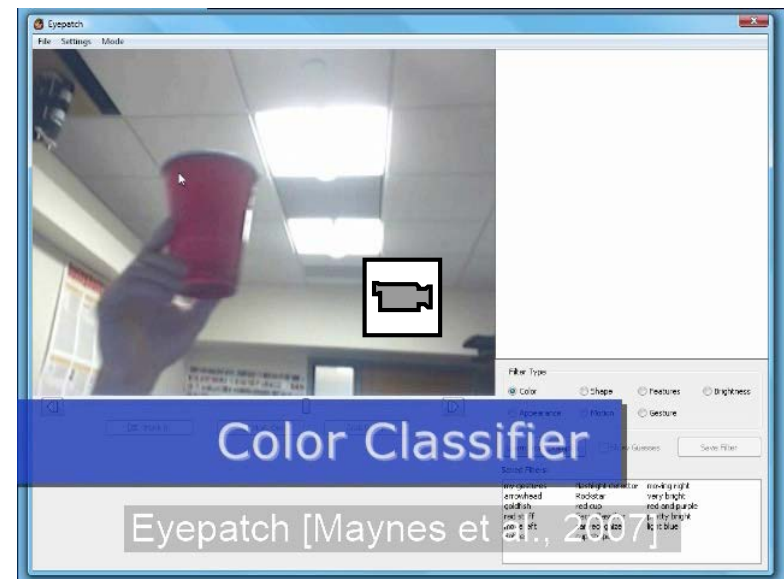
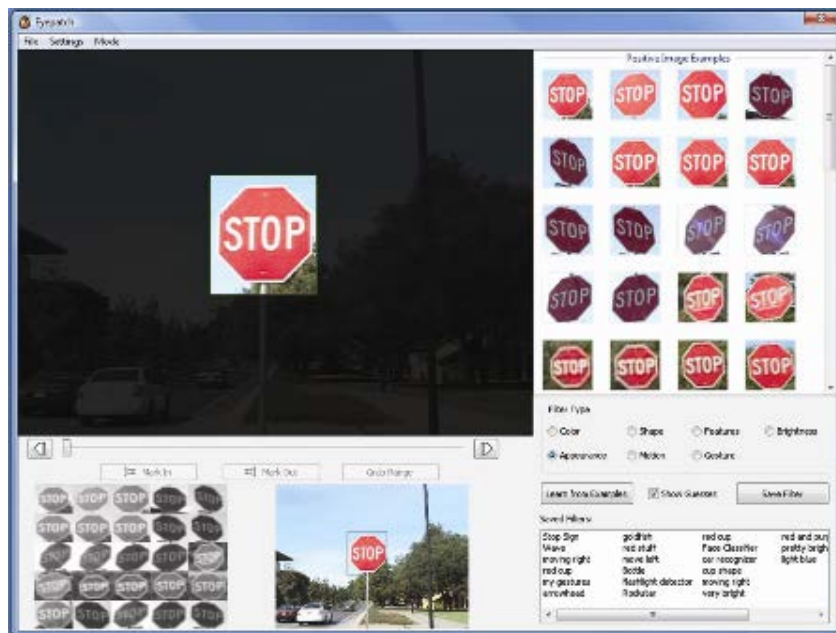
- GUIウィジェット (Widget) の物理デバイス版
- センサやアクチュエータの値を操作できるAPI



ツールキット: Eyepatch

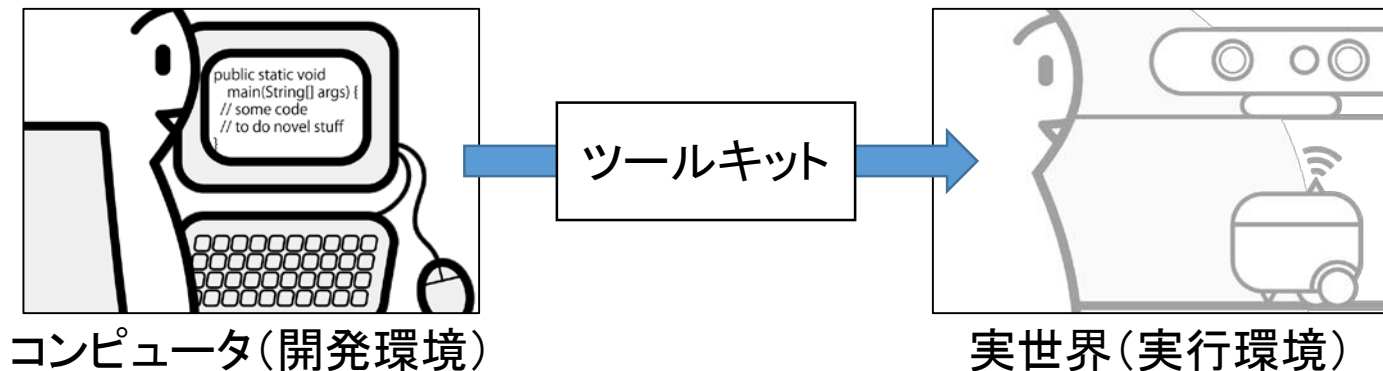
[Maynes et al., UIST '07]

- 画像処理のパラメタを画面上で指定
- カメラ入力に対する処理結果を取得できるAPI



ツールキット

アプリケーション開発に便利なユーザインタフェース (API)を提供する



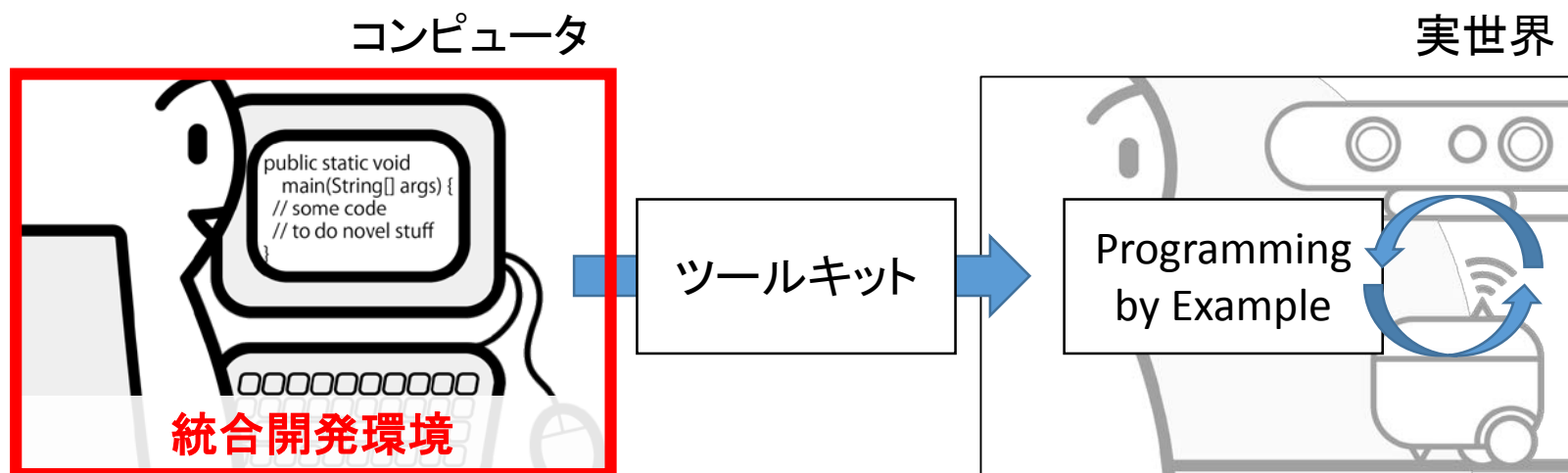
開発環境の外部ツールとして動作する

➡ ワークフロー全体に対する支援ではない

既存手法の問題点

Programming by Example ロジックを精密に指定できない

ツールキット ワークフロー全体に対する支援ではない



➡ 文字ベースの統合開発環境の改良が必要

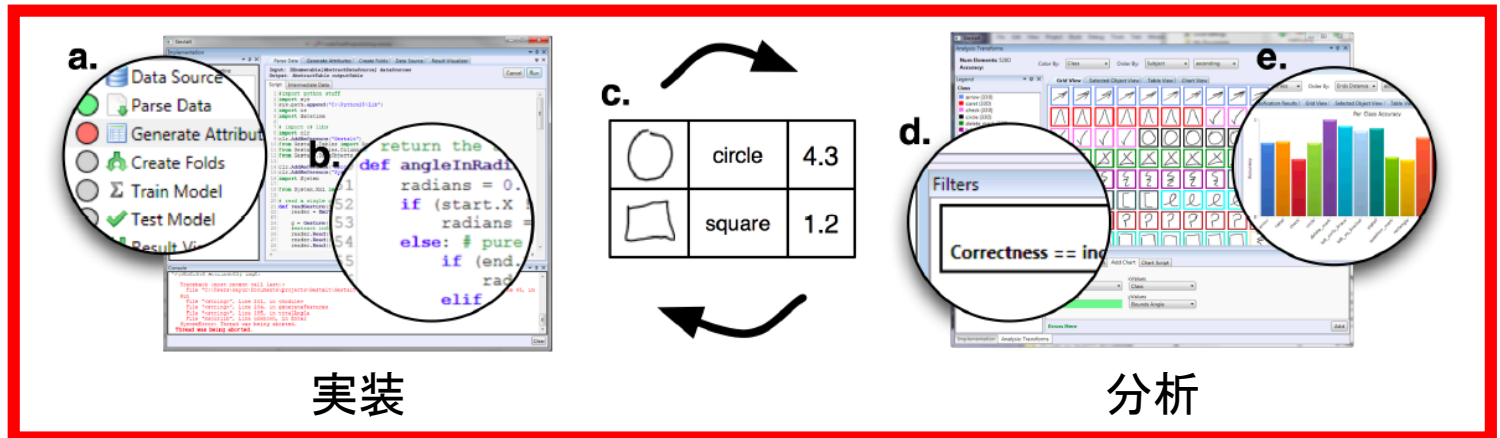
提案手法

統合開発環境の改良

実世界入出力を伴うプログラム開発のワークフローを支援し、プログラミングの効率向上を目指す

参考

機械学習アプリケーション開発のワークフローを支援し、プログラミングの効率向上を目指した



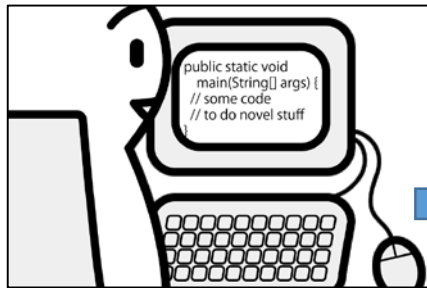
[Patel UIST '10]

本論文で支援するワークフロー: Programming with Example (PwE)

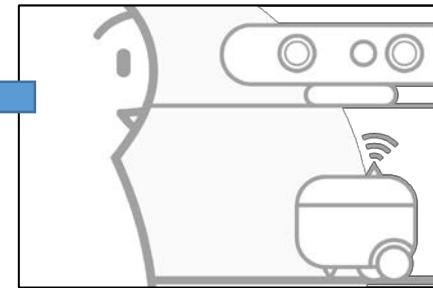
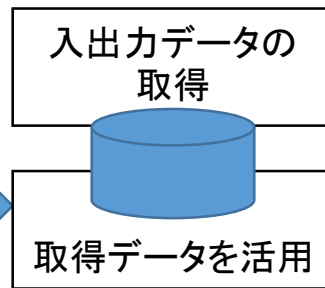
- プログラムの動作が実行環境に大きく依存する
- 抽象的なロジックだけを設計することが難しい



入出力データのExampleを取得・活用して開発する



コンピュータ(開発環境)



実世界(実行環境)

➡ Exampleの適切な表現手法が必要

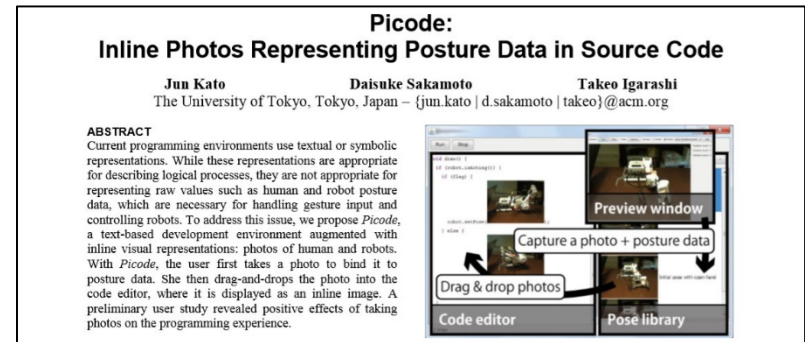
ワークフロー支援手法: Integrated Graphical Representations

文字表現 抽象的で精密なロジックを表現できる

画像表現 具体例を分かりやすく表現できる

Cf. 科学論文における
文章と図表の関係

“Orality and Literacy” [Ong '82]

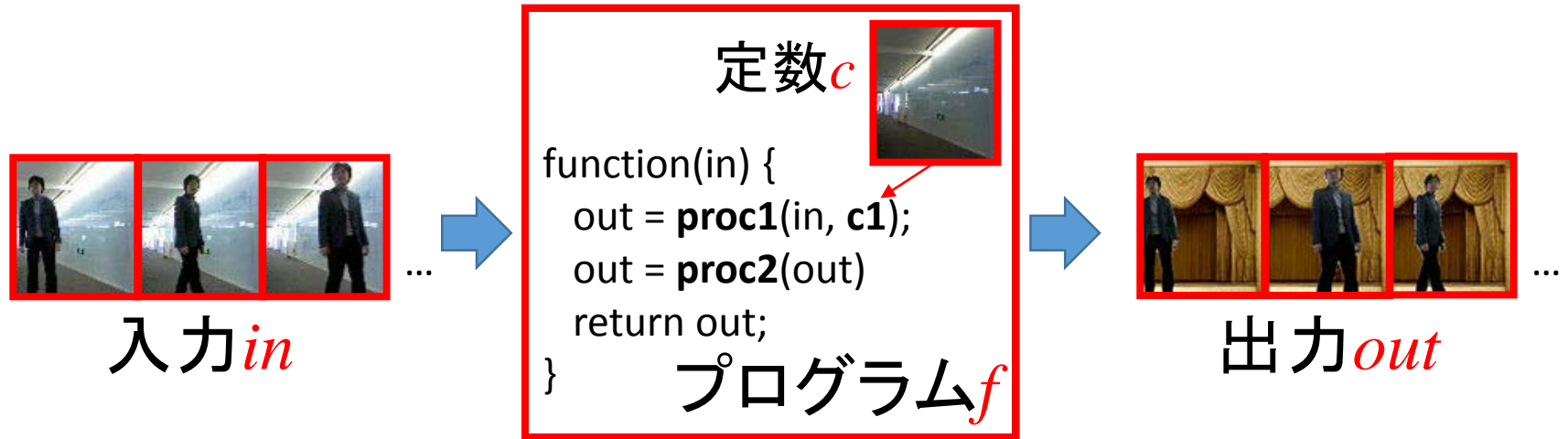


➡ Exampleを画像表現で表すことにより、
文字プログラミングの効率向上をねらう

各論の進め方

1. 実世界入出力を伴うプログラムをモデル化
2. モデルの各構成要素に対して
 1. 対応するExampleの活用法(PwE)を議論
 2. Exampleを画像表現で表す統合開発環境を提案
 3. ユーザスタディにより有効性を検証

実世界入出力を伴うプログラムのモデル: $out = f(in, c)$



構成要素	Example	画像表現による支援	章
c	実世界入出力データ	写真を用いた理解支援	4
in, out	プログラムの振る舞い	動画を用いた理解支援	5
f	実装済みプログラム	編集操作による実装支援	6

アウトライン

- イントロダクション
- 研究紹介
 - 写真を用いた実世界入出力データの理解支援 (4章)
 - 動画を用いたプログラムの振る舞い理解支援 (5章)
 - 画像表現の編集操作によるプログラム実装支援 (6章)
- ディスカッション
- 結論

研究紹介 (4章):

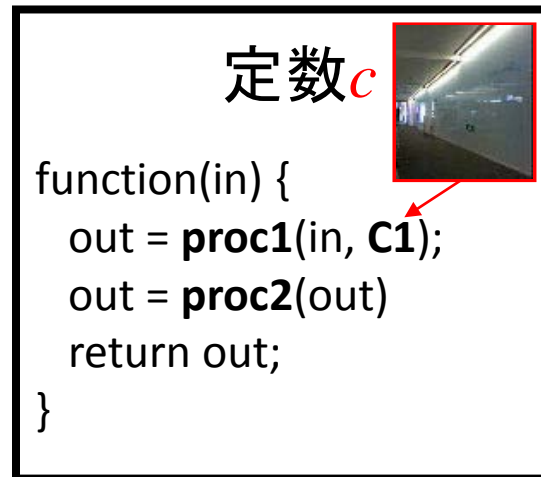
写真を用いた 実世界入出力データ 理解支援

Chapter 4. Using Photos to Understand Static Data

本研究の位置づけ

プログラムで用いる実世界入出力データ(定数 c)が
記号・文字表現では分かりづらい

➡ 画像表現(写真)を用いて分かりやすくしたい



Programming with Example

実世界入出力を伴うプログラムの特徴

センサやアクチュエータの操作に
使う定数が実行環境に依存する



1. センサやアクチュエータから
実世界入出力データを取得
2. 定数 c として保存
3. プログラムで c を読み込んで利用



Human 姿勢情報 (c)
-0.1177 0.0527 2.0215
-0.1211 0.0991 2.0098
-0.1343 0.3329 1.9649
...

```
if (pose.eq("c1.data"))  
  doSomething();
```

Programming with Example

定数 c の値の取得と保存:

- 必ずしも名前をつけられない
- 名前をつけるのが面倒



プログラムで c を利用:

- 文字で表示され、直感的でない

```
HumanPose pose = PoseLibrary.query("Whoa");  
HumanPose pose = new HumanPose(  
-0.0139, -0.0856, 2.2563, -0.0255, -0.0284, 2.3022, -0.0226, 0.2881, 2.2856, 0.012, 0.4823, 2.2662,  
-0.1898, 0.1982, 2.248, -0.4059, 0.2173, 2.2121, -0.393, 0.4257, 2.1835, -0.3946, 0.4749, 2.1794,  
0.1536, 0.1636, 2.3229, 0.3501, 0.1161, 2.3202, 0.4462, 0.3234, 2.3088, 0.4632, 0.3953, 2.3172,  
-0.0926, -0.1627, 2.2235, -0.0298, -0.5136, 2.1015, -0.0223, -0.7798, 2.0743, -0.0015, -0.8185, 1.9894,  
0.0693, -0.1669, 2.2707, 0.371, -0.3215, 2.0965, 0.5015, -0.6241, 2.0075, 0.5338, -0.6467, 1.9204  
);
```

➡ 従来の統合開発環境では難しい

Example(実世界入出力データ)を表す画像表現: 写真

- 実世界入出力データを写真と紐づけて管理
- 文字列ベースのエディタ中でインライン表示

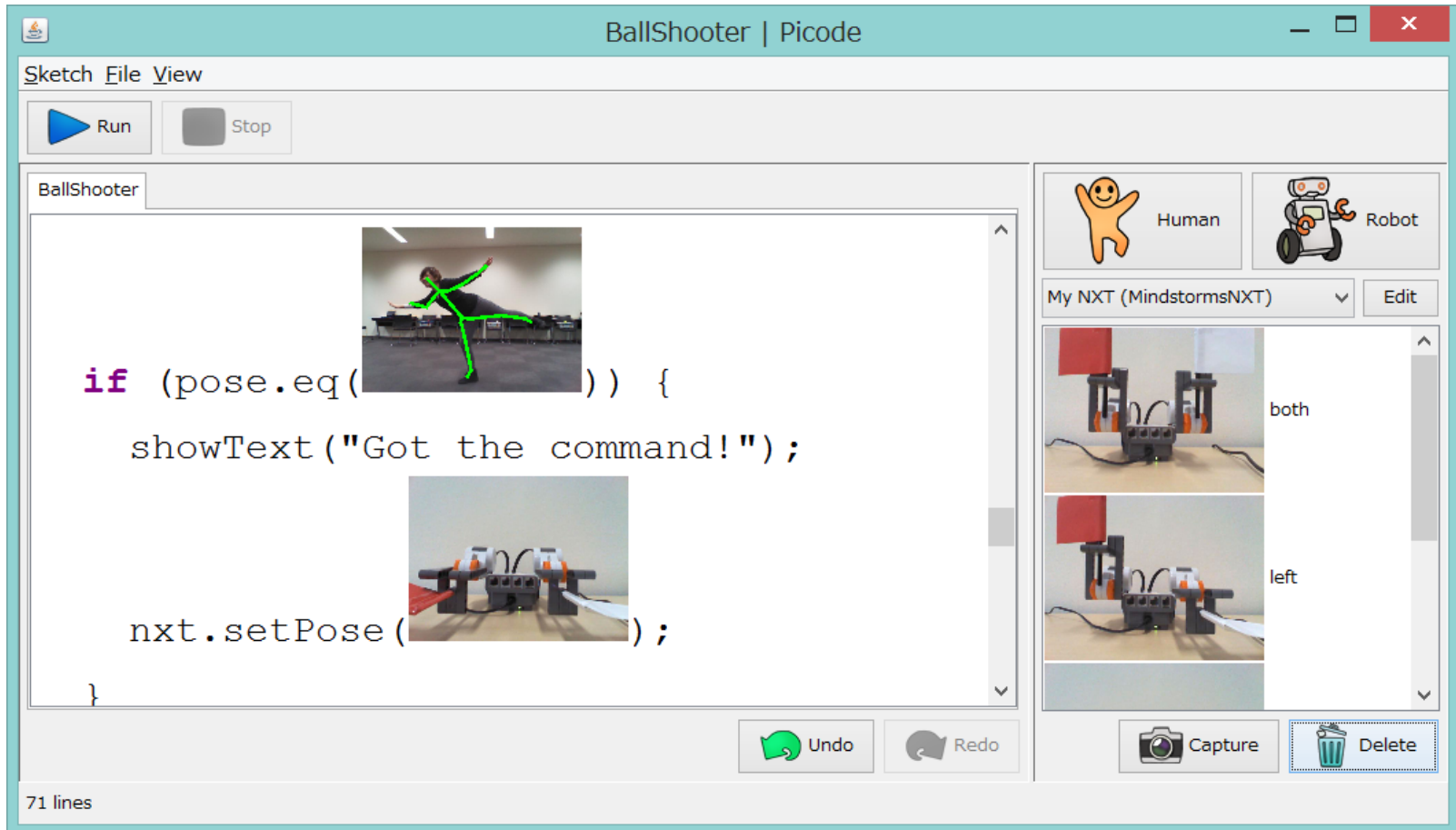
HumanPose pose =



試作システム: Picode

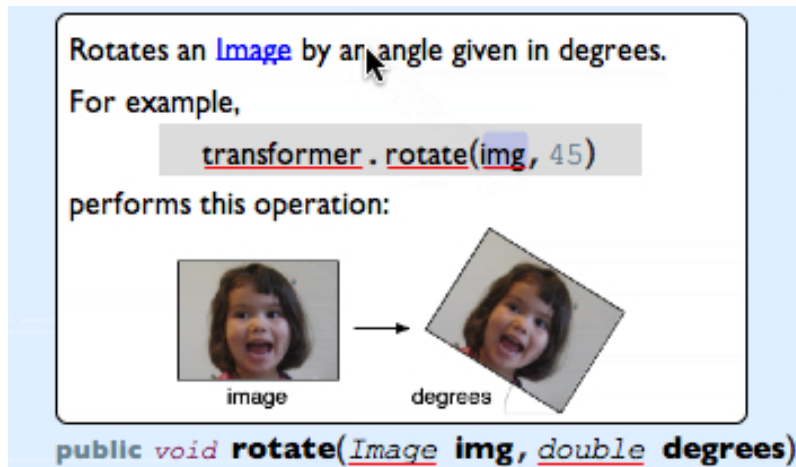


[Kato CHI '13]







関連研究

Barista [Ko CHI '06]



コードエディタ中に
画像表現を貼りこめる

Sikuli [Yeh UIST '09]

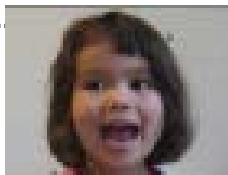
```
switchApp("CleanMyMac.app")  
click()  
click()  
while not find( Scan is finished  
Select the items you want to clean):  
sleep(5)  
click()
```

GUIスクリーンショットの画像
をエディタに貼りこめる

関連研究

Barista

データ
なし



プログラムと無関係な画像

Sikuli

画像
データ



GUI部品を画像で表現

提案手法: 姿勢データを写真で表現

姿勢データ



評価実験:

非プログラマ向けワークショップ

- 小学生、中学生 計29名 3時間
- 写真撮影とソースコードの編集をしてもらった
- 写真の利用方法を調査した

日本科学未来館でのワークショップの様子:



実演を交えた説明



演習



成果発表

評価実験の結果

全員、写真撮影を通してプログラムを編集できた

よい写真の例:



悪い写真の例: 似た写真、複数の被写体、前面物による遮蔽

➡ 状況把握に適すが、精密な情報提示には向かない

アウトライン

- イントロダクション
- 研究紹介
 - 写真を用いた実世界入出力データの理解支援 (4章)
 - 動画を用いたプログラムの振る舞い理解支援 (5章)
 - 画像表現の編集操作によるプログラム実装支援 (6章)
- ディスカッション
- 結論

研究紹介 (5章):

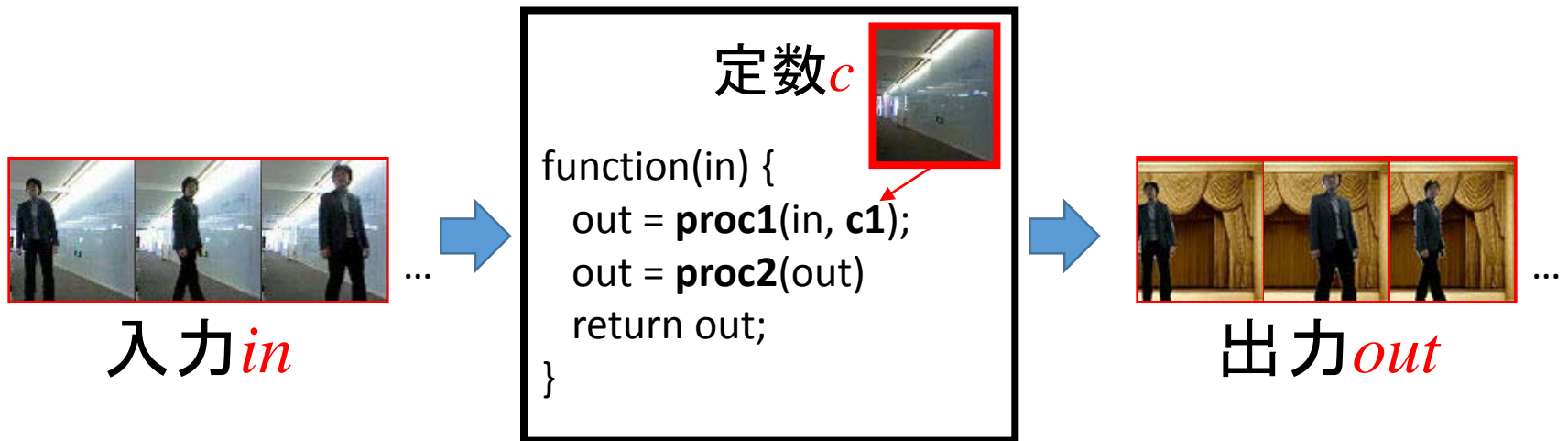
動画を用いた プログラムの振る舞い 理解支援

Chapter 5. Using Videos to Understand Dynamic Behavior

本研究の位置づけ

- 定数 c を写真で直感的に表せた(4章)
- プログラムの振る舞い(変数 in , out)が分かり辛い

→ 画像表現(動画)を用いて分かりやすくしたい



Programming with Example

実世界入出力を伴うプログラムの特徴

1. 処理が連続的に実行される
2. 同一の入力を二度与えられない

既存の統合開発環境での問題

1. ブレークポイントなどが使えない
2. バグの再現が難しい



1. 変数 in , out を保存して分析する
2. 変数 in を利用して再現性を確保する

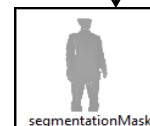
カメラ入力 (in)



API

プログラム

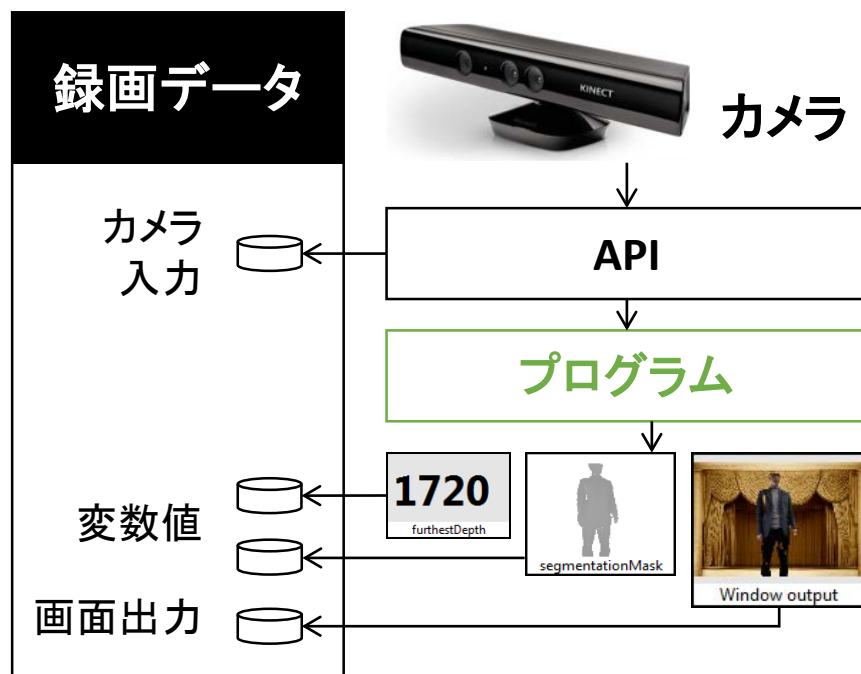
1720
furthestDepth



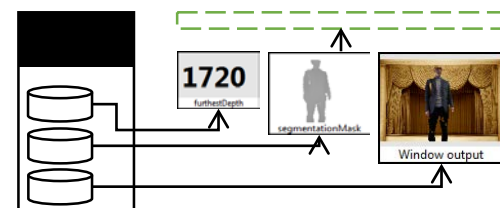
変数・画面出力 (out)

Programming with Example

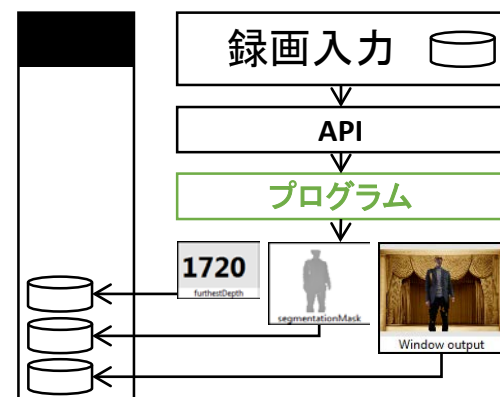
変数 in , out の録画:



録画データの分析:



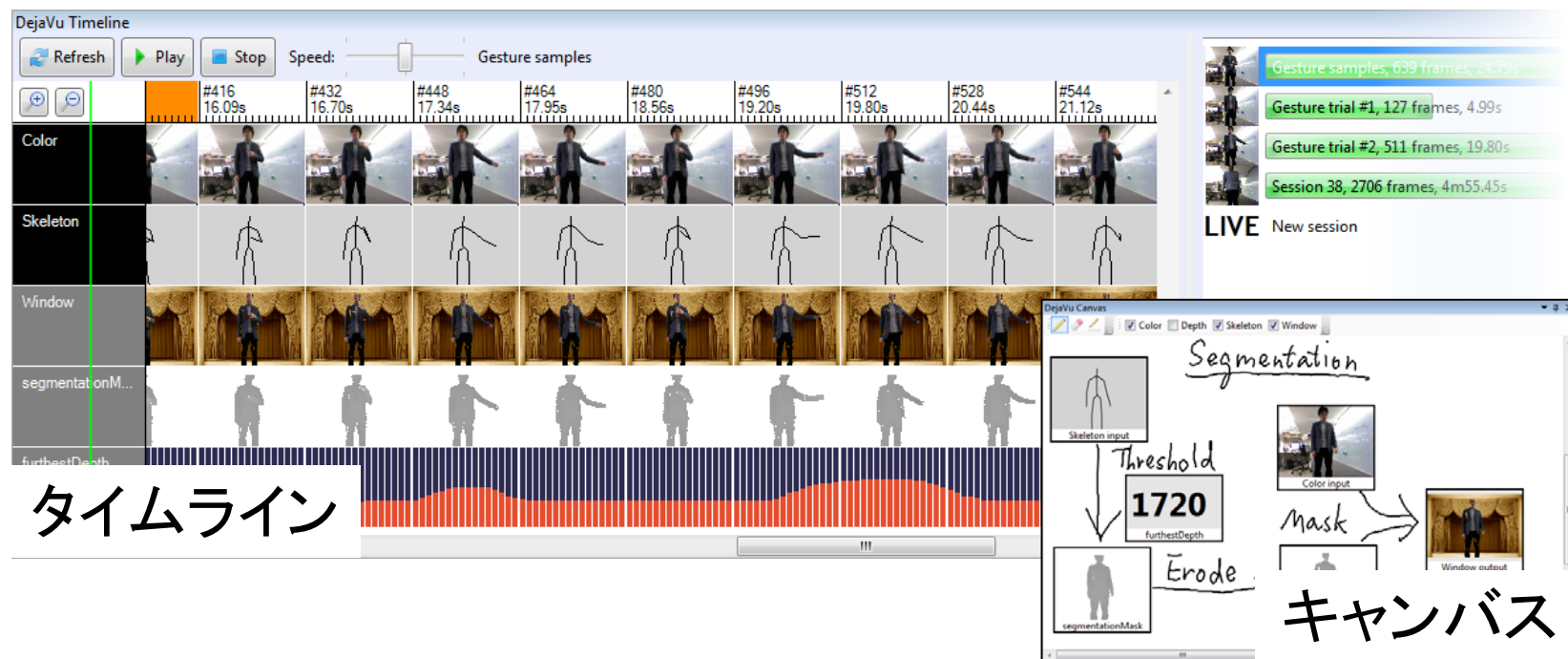
プログラム再実行:



➡ 従来の統合開発環境では難しい

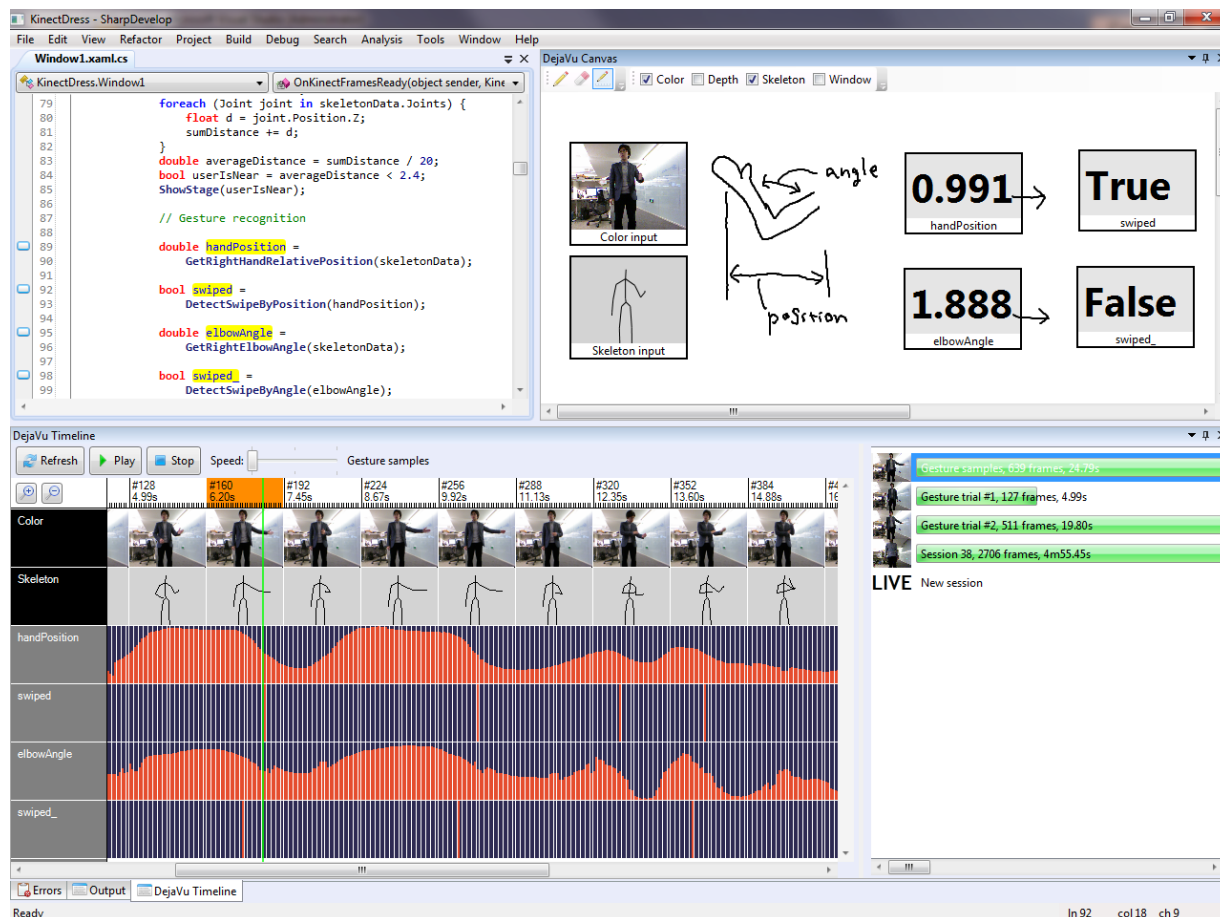
Example(プログラムの振る舞い)を表す画像表現: 動画

- プログラムの振る舞いを動画として記録・管理
- 2つのユーザインタフェースがコードエディタと連携



試作システム: DejaVu

[Kato UIST '12]



録画データの再生



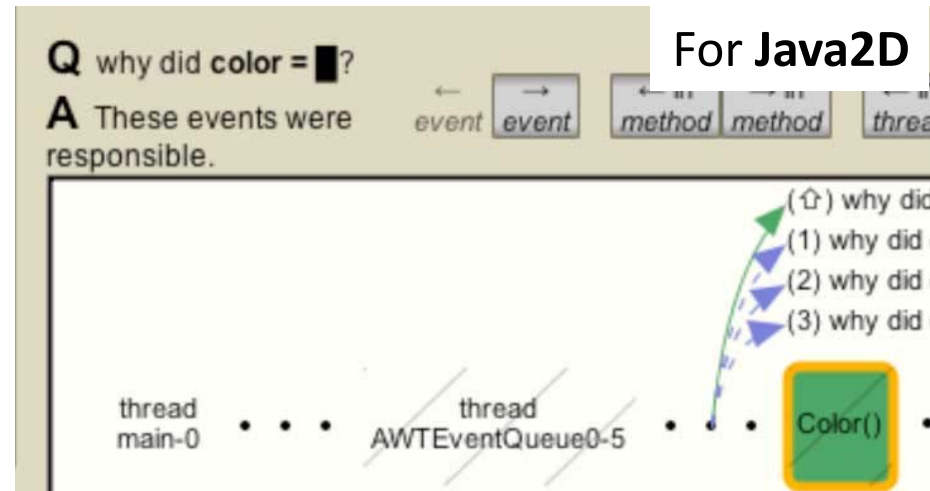
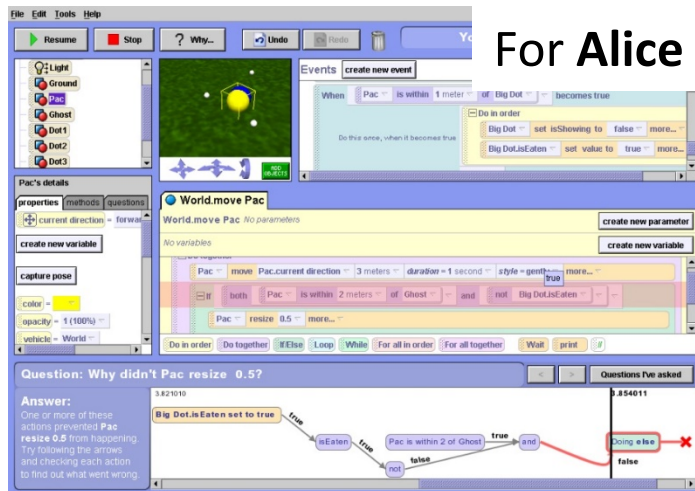
プログラム再実行



関連研究: Whyline

[Ko CHI '04, CHI '09]

ソースコードと出力の因果関係を表示



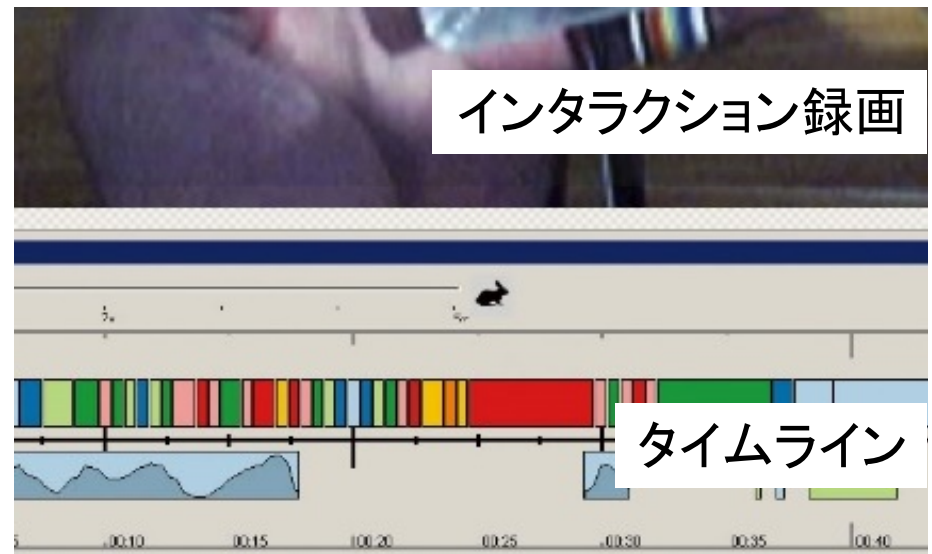
断続的なイベントを見せる目的で作られている

➡ 大量の視覚的データを見せる目的には適さない

関連研究: d.tools

[Hartmann UIST '06]

インタラクション録画をタイムラインと共に表示



予め決められた情報のみ可視化されている

➡ 自由な変数を選んで可視化できない

評価実験:

Microsoftプログラマによる試用

- 対象アプリケーションの業務開発経験がある3名
- 典型的アプリケーションを考え、DejaVuで開発してもらった
- 口頭でアンケートを行った



試用時のセットアップ

3名が考えたアプリケーション:

- A) ボールの位置を追跡、表示するアルゴリズム
- B) 顔の動きに追従するユーザインタフェース
- C) ジェスチャ認識アルゴリズム

評価実験の結果

全員から実用したい旨の高評価

アンケート結果:

- 録画データを自由に編集して使いたい
- 自分で定義した型も可視化したい
- 可視化手法をGUIで編集したい(画像フィルタなど)

➡ 画像表現を簡単に編集できるAPI・ツールが必要

アウトライン

- イントロダクション
- 研究紹介
 - 写真を用いたデータの理解支援 (4章)
 - 動画を用いたプログラムの振る舞い理解支援 (5章)
 - 画像表現の編集操作によるプログラム実装支援 (6章)
- ディスカッション
- 結論

研究紹介 (6章):

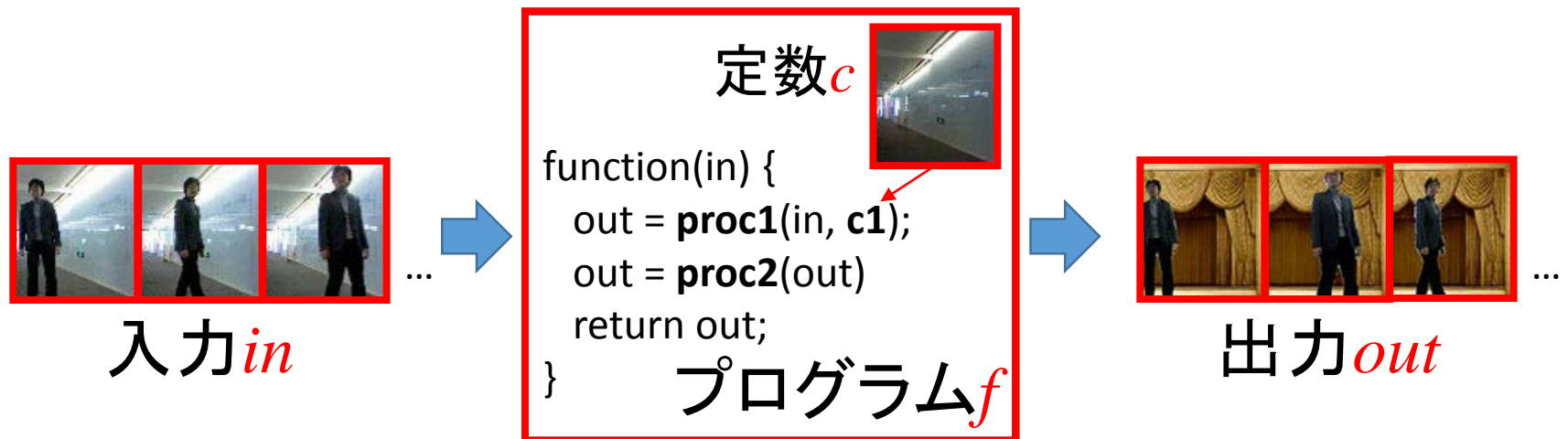
画像の編集操作による プログラムの 実装支援

Chapter 6. Graphical Editing to Specify Program Behavior

本研究の位置づけ

- 定数 c を写真で直感的に表せた(4章)
- 変数 in, out を動画で直感的に表せた(5章)

➡ プログラム(関数 f)の実装を直接支援したい



Programming with Example

実世界入出力を伴うプログラムの特徴

複数アルゴリズムの組み合わせで構成される



1. 様々なアルゴリズムから適したものを選ぶ
2. 実行結果を見て反復的にプログラムを実装する

Programming with Example

1. 様々なアルゴリズムから適したものを選ぶ
2. 実行結果を見て反復的にプログラムを実装する



既存の統合開発環境での問題

1. コード補完の選択肢が多すぎる

例) OpenCV 2.4.8 画像処理クラスのメソッド数: 168個

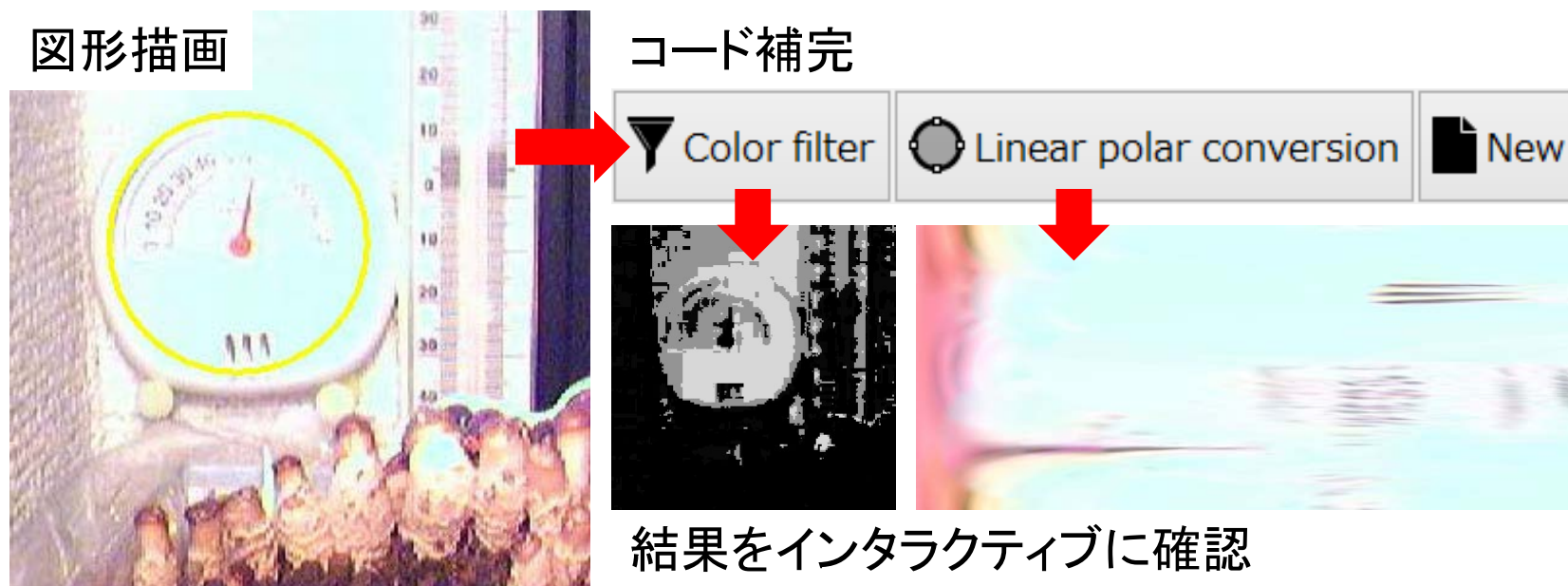
2. プログラム再実行のコストが高い



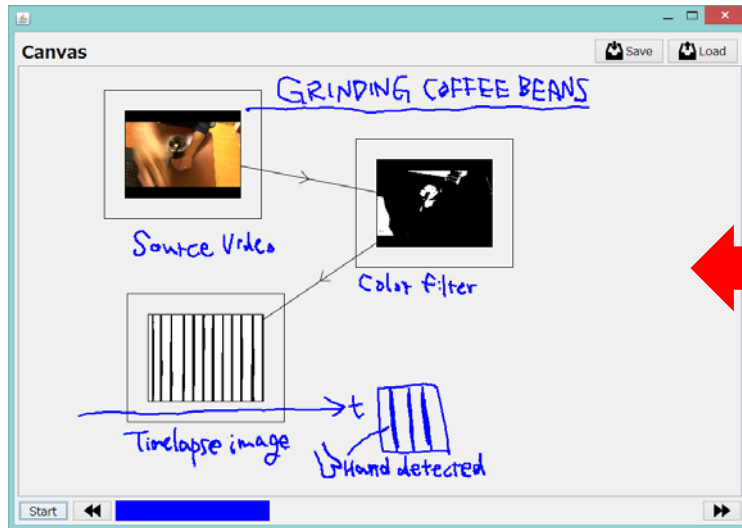
➡ 従来の統合開発環境では難しい

画像表現の編集操作による プログラム実装支援

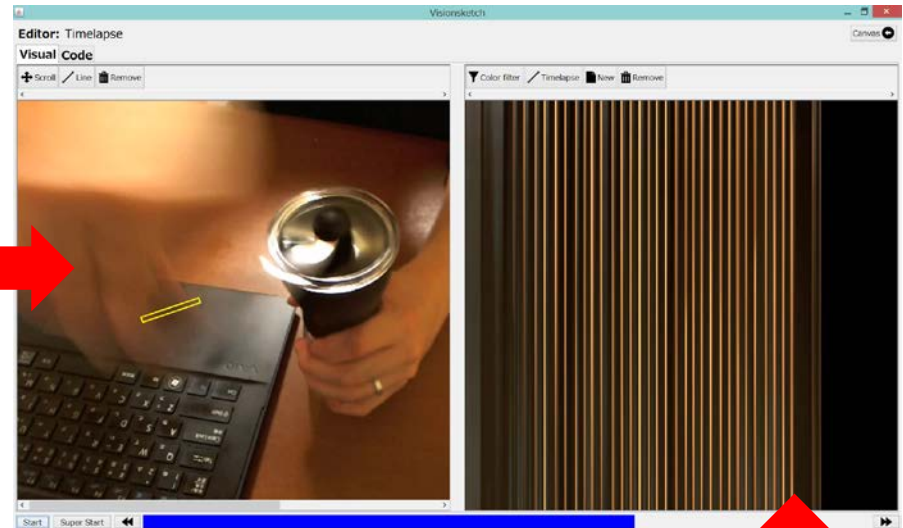
- Example (直前の実行結果) への描画で実装を行う
- 結果をインタラクティブに確認



試作システム: Visionsketch



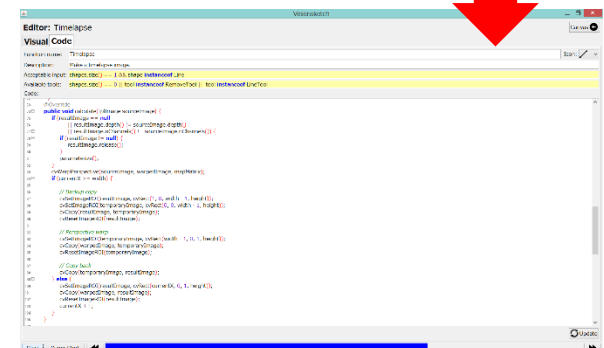
Canvas



Visual Editor



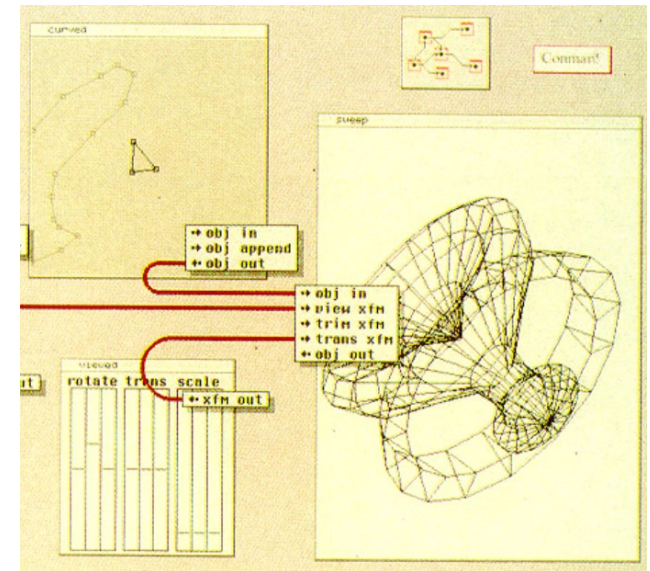
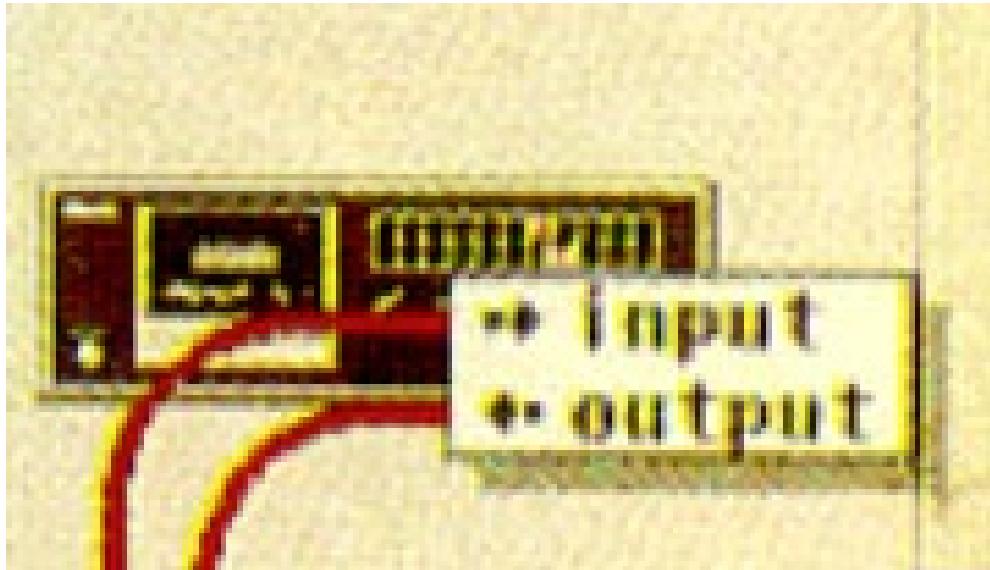
Code Editor



関連研究: ConMan

[Haeberli SIGGRAPH '88]

データの再生操作によるプログラム実行



レンダリング用のパラメタ調整が主目的

➡ 可視化のみで編集操作をサポートしない

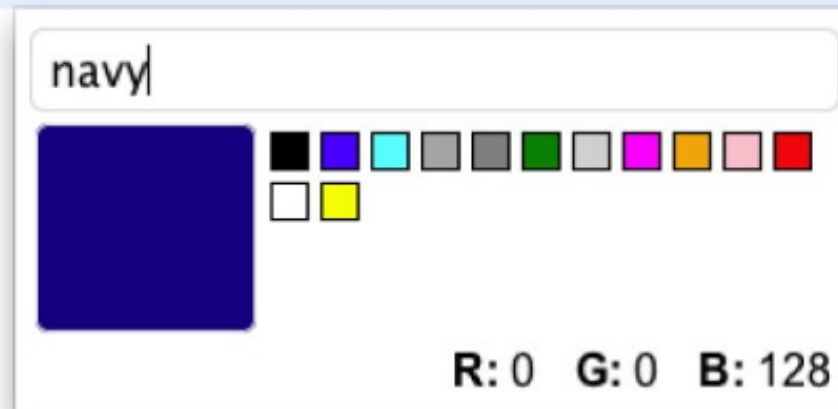
関連研究:

Active Code Completion

[Omar ICSE '12]

型に応じた値入力用インタフェースの表示

```
public Color getDefaultColor() {  
    return  
}
```

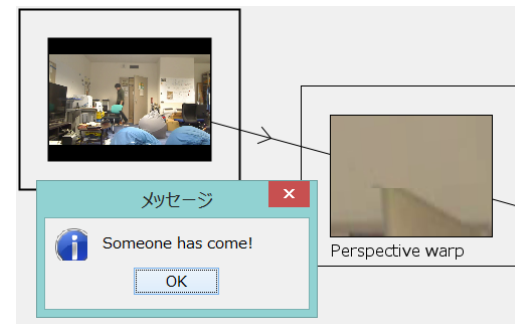


値入力用アルゴリズムは予め用意されたもの

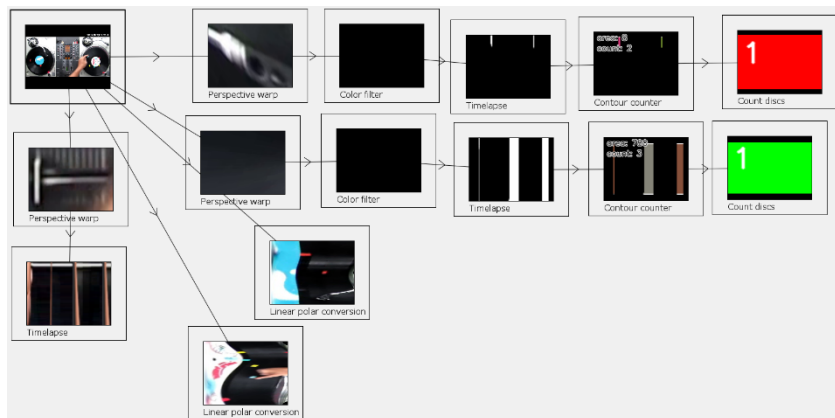
➡ 本研究は描画図形をもとにアルゴリズムを選択可

評価実験: プログラマによる試用

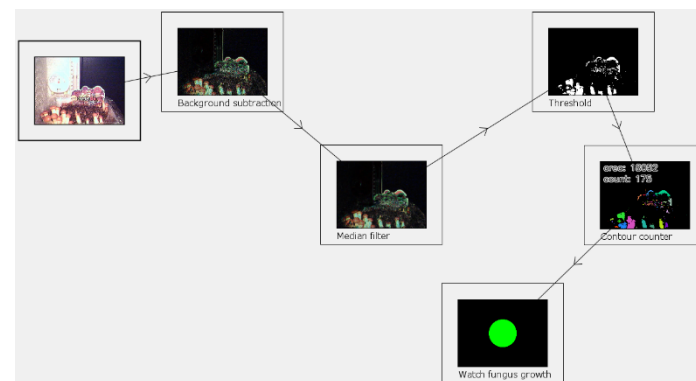
- 研究開発・製品開発経験のあるプログラマ5名
- 4名がOpenCVを用いた静止画処理経験あり



Door Watcher 



Disc Jockey Analyzer



Good-for-eating Sensor



評価実験の結果

- 開発環境の画面を、そのままプログラムのユーザインタフェースとして利用する例が見られた
- 文字ベースでコンパクトな実装が書ける場合でも、実装が容易な画像表現の編集操作が好まれた

# 質問	Canvas	Visual Editor	Code Editor
1 頻繁に使いたいと思う	5/5	5/5	3/5
2 不必要に複雑だった	0/5	1/5	3/5
3 簡単に使えた	5/5	5/5	2/5
4 利用に技術支援が必要だった	2/5	2/5	4/5

事後アンケート結果(抜粋)

➡ コードエディタに関して4,5章の内容を併用すべき

アウトライン

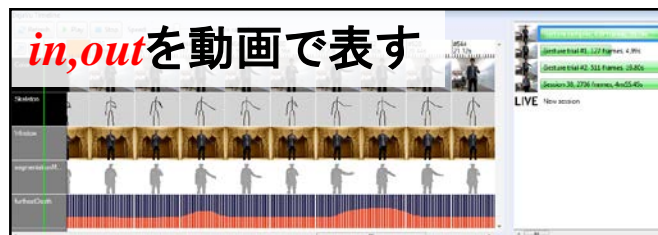
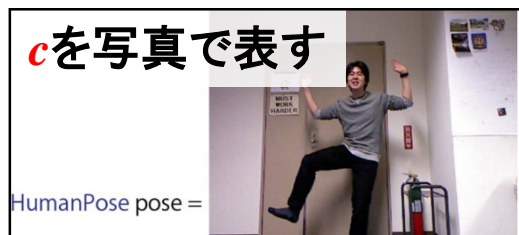
- イントロダクション
- 研究紹介
- ディスカッション
 - 3つの研究のまとめ
 - 将来展望
- 結論

3つの研究のまとめ:

Integrated Graphical Representations

1. 実世界入出力を伴うプログラム開発のワークフロー (Programming with Example, PwE) を分析
2. Exampleを画像表現で表すことでPwEを支援する統合開発環境を試作
3. ユーザスタディで有用性を検証し、課題を議論

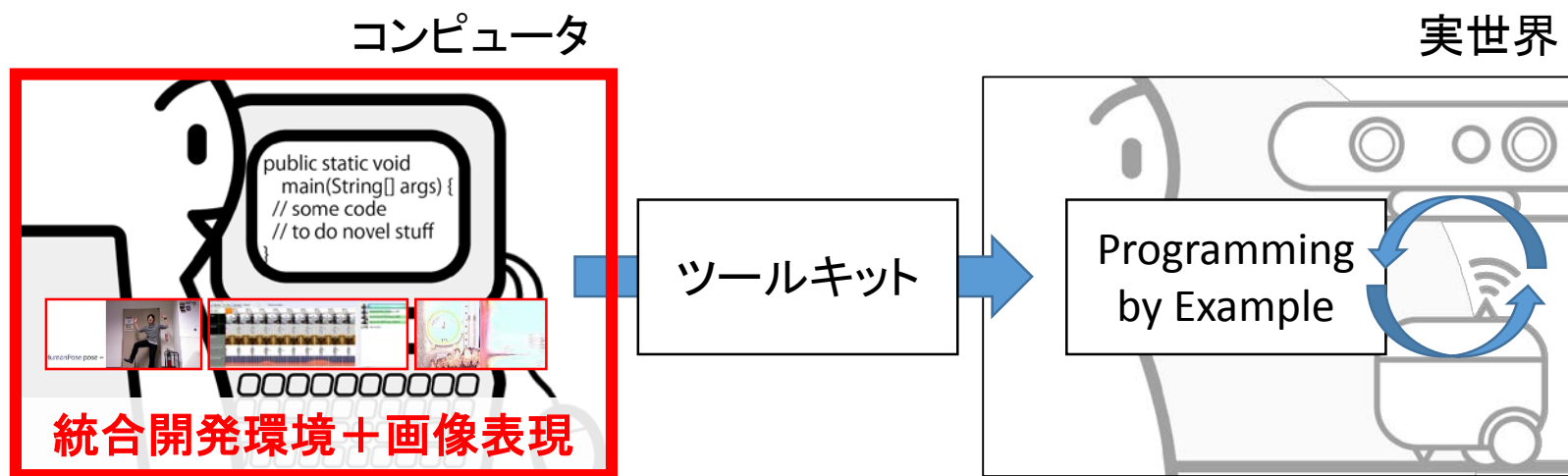
➡ モデル $out = f(in, c)$ の各構成要素に対して行った



3つの研究のまとめ:

具体例と文字表現の両用

1. PbEが苦手な**ロジックの精密な記述**を実現
2. ツールキットにない**ワークフロー全体支援**を実現



➡ 既存手法の苦手とする機能を両立

3つの研究のまとめ:



画像表現の使い方を分析

- 画像表現のみだと曖昧さが残るため、文字やスケッチによる補足が必要 (4章)
- 画像表現の使い方をプログラマが自由に拡張できるような統合開発環境のアーキテクチャが望ましい (5章)
- 画像表現は直接操作できるべきで、文字ベースのエディタと切り離さないことが望ましい (6章)

➡ 画像表現の統合についての設計指針を与えた

将来展望

- 画像表現のリミテーション克服
 - 画像処理による差異の強調 [Wu et al., SIGGRAPH 2012]
 - 三次元拡張による遮蔽の解決 [Takahashi et al., IEEEVis 2004]
- 視覚以外の五感の活用

		開発対象のアプリケーションが扱う情報	
		視覚的な情報	それ以外
統合開発環境が 利用する感覚器	視覚	本研究 	本研究の直接的な応用  = 鰻の香りデータ
	それ以外	問題設定が利用可能 ?	モデルとPwEが利用可能 ?

結論

統合開発環境に画像表現を導入し、実世界入出力を伴うプログラム開発のワークフローを支援した

構成要素	Example	画像表現による支援	章
<i>c</i>	実世界入出力データ	写真を用いた理解支援	4
<i>in, out</i>	プログラムの振る舞い	動画を用いた理解支援	5
<i>f</i>	実装済みプログラム	編集操作による実装支援	6



ご清聴ありがとうございました (代表的な発表文献一覧)

- 4章: “**DejaVu: Integrated Support for Developing Interactive Camera-Based Programs**”, In *Proc. of the 25th annual ACM symposium on User Interface Software and Technology*, pp.189-196, Oct. 2012.
(Microsoft Research Asia; Dr. Sean McDirmid, Dr. Xiang Cao と共著)
- 5章: “**Picode: Inline Photos Representing Posture Data in Source Code**”, In *Proc. of the SIGCHI conference on Human Factors in Computing Systems*, pp.3097-3100, Apr. 2013. (坂本大介氏, 五十嵐健夫氏と共著)
- 6章: “**Visionsketch: Integrated Support for Example-Centric Programming of Image Processing Applications**”, In submission to *the 2014 Graphics Interface conference*. (坂本大介氏, 五十嵐健夫氏と共著)

