

2021年度 卒業論文

オブジェクトの動作に基づく
Scratch 作品の直感的検索手法

2022年2月15日

システム工学科
(学生番号: 60236238)

福地 ユキ

和歌山大学システム工学部

概 要

ビジュアルプログラミング言語は、視覚的に表現された命令処理を操作することでプログラムを実装可能なプログラミング言語である。ビジュアルプログラミング学習サービスの1つであるScratchでは、ユーザが制作したプログラム作品をオンラインサービス上に公開可能である。Scratchのオンラインサービス上には、多数のユーザが制作した膨大な数の作品が公開されており、ユーザは他者の作品を参照することで多様な実装方法を学習する。参照する他者の作品を探し出すためにキーワード検索を行うが、ユーザがイメージする動作を言語化することは難しく、動作に関するキーワードをタイトルや説明文に含まない作品は検索不可能であるため、ユーザがイメージする動作を含む作品を検索することは容易ではない。キーワードによる検索ではなく、マウスを用いた描画等によって動作のイメージを入力とする直感的な検索を実現することで、ユーザがイメージする動作を含む作品の検索を容易にできると考える。

本研究では、動作のイメージを入力とする、オブジェクトの動作に基づくScratch作品の直感的検索手法を提案する。入力と検索対象の各動作の類似度合いを測るために、2つの時系列データの距離を算出する動的時間伸縮法を用いる。類似動作を抽出可能な距離を明らかにするために、13,437件の公開作品を対象に入力を行い、距離算出結果の分析を行なった。分析では、入力との距離が異なる複数の動作を被験者に提示し、被験者全員が類似していないと判断する値よりも近い距離を、類似動作を抽出可能な距離とした。結果は、～。 **TODO: [結果を載せる]**

本研究は、動作のイメージを入力とするScratch作品の直感的検索手法を提案することで検索を容易にし、他者の作品を参照することによる実装方法の学習が促進されることを期待する。

目次

第1章	はじめに	1
第2章	ビジュアルプログラミング言語 Scratch におけるプログラム作品検索	2
2.1	Scratch	2
2.2	Scratch 作品の検索	5
第3章	Scratch 作品に含まれる類似動作の抽出手法	6
3.1	概要	6
3.2	オブジェクトの座標の時系列データ取得	6
3.2.1	動作イメージの入力	6
3.2.2	検索対象作品に含まれるオブジェクトの動作	6
3.3	座標の時系列データの距離算出	8
3.3.1	時系列データの正規化	8
3.3.2	DTW 距離の算出	9
第4章	検索者の求める動作を抽出する距離の分析	10
4.1	概要	10
4.2	検索データセット	10
4.3	分析	10
4.4	分析結果	12
第5章	考察	17
5.1	動作の類似性	17
5.2	プログラム	17
5.3	より長い動作の検索	17
5.4	妥当性への脅威	17
第6章	おわりに	18

第1章 はじめに

初等教育からのプログラミング必修化に伴い、プログラミング教育が進められている。特に、テキストの入力ではなく、視覚的に表現された命令処理の操作でプログラムを実装するビジュアルプログラミング言語が利用されている。ビジュアルプログラミング言語は、直感的にプログラムを実装可能なことや、視覚的に表現されたプログラムの理解が容易であること、テキストベースのプログラミング言語で障壁となるテキスト入力の間違いが発生しないことなどから、プログラミング初学者が取り組みやすい言語となっている。

ビジュアルプログラミング言語を用いた代表的な学習サービス Scratch¹は、ユーザが制作したプログラム作品をタイトル・説明文を添えてオンラインサービス上に公開可能である。2022年1月時点で8,300万以上の人がScratchに登録しており、オンラインサービス上に9,400万件以上のプログラム作品を公開している²。Scratchユーザは、オンラインサービス上の他者の作品を参照することで多様な実装方法を学習する[1]。参照する作品を探し出すために、ユーザは作品のタイトルと説明文を対象としたキーワード検索を行う。しかしユーザにとって、イメージする動作を検索キーワードへ変換することは難しく、動作に関するキーワードをタイトルや説明文に含んでいない作品を検索することは不可能であるため、ユーザのイメージを含む作品の検索は容易ではない。作品検索が容易ではないという課題は、他者の作品を参照する学習の障壁になると示唆する。キーワードによる検索ではなく、動作のイメージを入力とするScratch作品の直感的な検索を実現することで、ユーザがイメージする動作を含む作品の検索を容易にできると考える。

本論文では、マウスを用いた描画等によって動作のイメージを入力とし、オブジェクトの動作に基づくScratch作品の直感的検索手法を提案する。入力と検索対象の各動作の類似度合いを測るために、2つの時系列データの距離を算出する動的時間伸縮法を用いる。類似動作を抽出可能な距離を明らかにするために、13,437件の公開作品を対象に入力を行い、距離算出結果から入力と動作の類似性を評価する定性分析を行う。動作のイメージを入力とするScratch作品の直感的検索手法を提案することで検索を容易にし、他者の作品を参照することによる実装方法の学習の支援を期待する。

続く2章では、Scratchにおけるプログラム作品検索の背景と課題について述べる。3章では提案手法について説明し、4章では入力に類する動作を抽出可能な距離の分析とその結果について述べる。5章では分析結果を考察し、最後に6章でまとめを述べる。

¹<https://scratch.mit.edu/>

²<https://scratch.mit.edu/statistics/>

第2章 ビジュアルプログラミング言語Scratchにおけるプログラム作品検索

2.1 Scratch

Scratch¹は、MIT メディアラボが開発しているビジュアルプログラミング言語の1つである。Scratchは「10歩動かす」「15度回す」などの命令処理をブロックで視覚的に表現し、複数のブロックを組み合わせて、画面上のオブジェクトを動かすプログラム作品を制作する。Scratchで制作される作品の種類は、ゲームやストーリー、アニメーションなど多岐に渡る[2]。Scratchは、直感的にプログラムを実装可能なことや、視覚的に表現されたプログラムの理解が容易であること、テキストベースのプログラミング言語で障壁となるテキスト入力の間違いによる構文エラーが発生しないことなどから、プログラミング初学者が取り組みやすいプログラミング言語となっている。従来研究では、Scratchを用いた学習がプログラミング学習に有効であり、テキストベースのプログラミング言語への移行を容易にすることを明らかにしている[3][4]。

図2.1は、Scratch3.0における作品制作画面を示す。作品は主に「ブロック」「スクリプト」「スプライト」の3つの要素で構成されている。

- **ブロック**: 視覚的に表現された命令処理を1つのブロックで表現する。図2.1の(1)に示す複数のブロックを繋ぎ合わせて使用する。命令処理の種類によって形状や色が異なる。表2.1は形状の種類と性質、表2.2は色の種類と性質を示す。
- **スクリプト**: 複数のブロックを組み合わせて制作したひとまとまりのプログラムである。スクリプトはイベント駆動型であり、図2.1の(2)に示すスクリプトの例では、スペースキーが押された際にプログラムが実行される。
- **スプライト**: キャラクター等の画像のオブジェクトである。図の～に示すスクリプトエリアにブロックを配置しスクリプトを制作することで、プログラムをスプライトに実行させる。プログラム作品制作の初期画面では、図2.1の(3)に示すCatがあらかじめ配置されている。ユーザはスプライトを複数配置することができ、Scratchで用意されているデフォルトの画像や、ユーザが創作した画像をスプライトとして使用可能である。

図2.1に示す例では、スペースキーが押された際にCatのスプライトが「こんにちは!」と発言し、その後進行方向に10歩移動する動作を10回繰り返す作品である。

Scratchで制作した作品は、図2.1の(1)に示すタイトルと図2.1の(2)に示す説明文を添えてオンラインサービス上に公開することができる。2022年1月時点で、8,300万人以上の人々がScratchに登録しており、オンラインサービス上に9,400万件以上の作品を公開している²。他者の公開作

¹<https://scratch.mit.edu/>

²<https://scratch.mit.edu/statistics/>

品のプログラムを閲覧できるため、ユーザは他者の公開作品を参照することで多様な実装方法を学習できる [1].

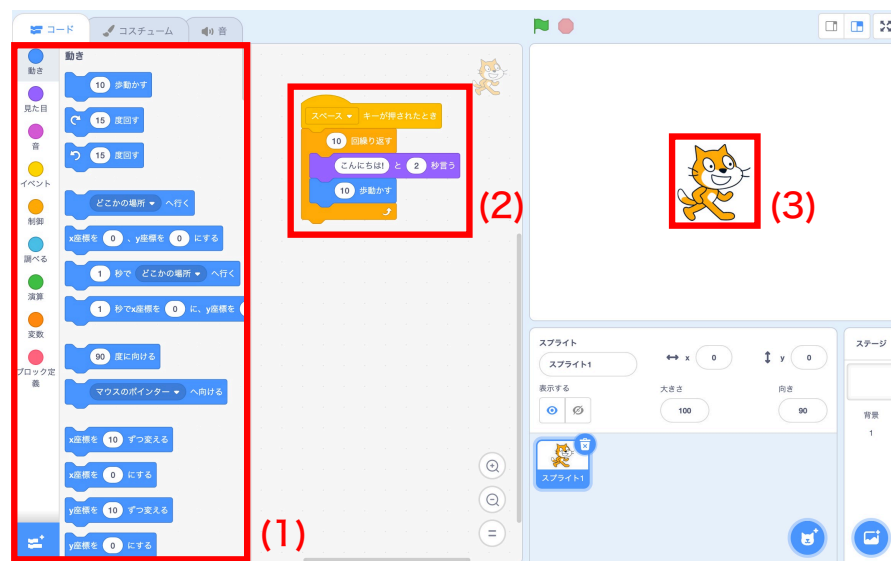


図 2.1: Scratch の作品制作画面



図 2.2: 公開した Scratch 作品画面

表 2.1: ブロックの形状と名前と性質



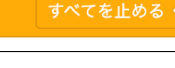
ブロックの形状	名前	性質
	ハットブロック	スクリプトの処理を開始するブロック
	スタックブロック	記述された命令を実行するブロック
	真偽ブロック	真偽値を保持し、他のブロックの引数となるブロック
	値ブロック	値を保持し、他のブロックの引数となるブロック
	C型ブロック	間にあるブロックに対して、条件分岐や繰り返しを実行するブロック
	キャップブロック	スクリプトの処理を停止するブロック

表 2.2: ブロックの色と名前と性質

ブロックの色	名前	性質
	動きブロック	スプライトを動かすブロック
	見た目ブロック	スプライトの見た目を制御するブロック
	音ブロック	BGM や SE をいった音を制御するブロック
	イベントブロック	スクリプト実行のトリガーとなるブロック
	制御ブロック	スクリプトを制御するブロック
	調べるブロック	スプライトや操作などの状態を調べるブロック
	演算ブロック	数値や論理、文字列に対する演算を行うブロック
	変数ブロック	変数とリストを扱うブロック
	ブロック定義ブロック	メソッドを定義するブロック

2.2 Scratch 作品の検索

ユーザは、Scratch オンラインサービス上に公開されている膨大な数のプログラム作品を参照することで多様な実装方法を学習することができる [1]。参照する作品を探し出すために、ユーザは作品のタイトルと説明文を対象としたキーワード検索を行う。C 言語や Java をはじめとするテキストベースのプログラミング言語ではメソッド名などのキーワードを使ったプログラム検索の研究が多数行われている [6]。しかし、Scratch 作品の検索において、ユーザがイメージする動作を検索キーワードへ変換することは難しく、動作に関するキーワードをタイトルや説明文に含んでいない作品を検索することは不可能であるため、ユーザのイメージする動作を含む作品の検索は容易ではない [5]。キーワードではなくマウスを用いた動作描画等の直感的な操作による検索を可能にすることで、ユーザのイメージする動作を入力とする検索が可能となり、作品検索が容易になると考える。

本論文では、動作のイメージを入力とする Scratch 作品の直感的検索システム実現に向けて、類似動作を抽出する手法を提案する。図 2.3 は、本論文で提案する手法を用いることで実現を期待する Scratch 作品の直感的検索システムの概略図を示す。続く第 3 章では、提案手法について述べる。



図 2.3: Scratch 作品の直感的検索システムの概略図

第3章 Scratch 作品に含まれる類似動作の抽出手法

3.1 概要

本章では、動作のイメージを入力とする Scratch 作品の直感的検索手法を述べる。図 3.1 は、手法の概略図を示す。手順 1 では、入力となる動作のイメージの座標の時系列データをマウスによる描画で取得し、検索対象の Scratch 作品に含まれるオブジェクトの座標の時系列データを画像認識を用いて取得する。手順 2 では、手順 1 で取得した入力と各動作の時系列データ同士の距離を算出する。算出結果のうち、距離が近い動作を入力と類する動作として抽出する。類似動作を抽出する距離については、4 章で分析及びその結果について述べ、5 章で考察について述べる。

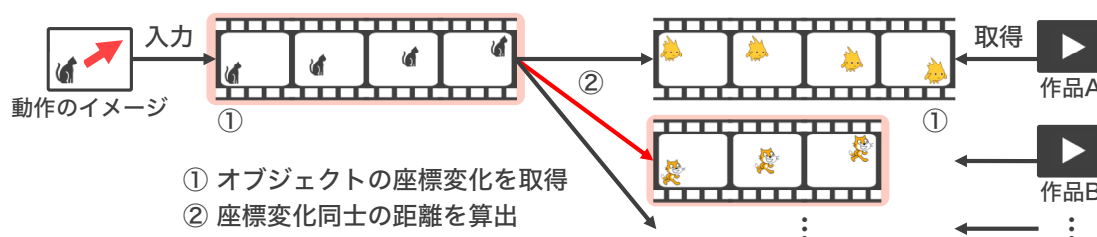


図 3.1: 動作のイメージに基づく検索手法の概略図

3.2 オブジェクトの座標の時系列データ取得

3.2.1 動作イメージの入力

ユーザの持つ動作のイメージを、マウスで描画することによって入力する。動作のイメージを描画中、軌跡を十分再現可能である 0.2 秒ごとにマウスの座標を取得する。取得した座標を、入力の座標変化の時系列データとする。

3.2.2 検索対象作品に含まれるオブジェクトの動作

Web アプリケーションの自動テストツール Selenium¹を用いて、Scratch のオンラインサービス上に公開されている作品のスナップショットを収集する。スナップショットは、入力と同様の間隔で時系列データを取得するため 0.2 秒に 1 枚を収集する。作品の中には実行が終わらない作品も存在するため、該当する作品に対しては、最大 100 枚のスナップショットを収集する。また、スナップショット収集を自動化するため、キー入力やマウス操作を必要としない作品を対象とする。

¹<https://www.selenium.dev/>

収集した各作品のスナップショット中に存在する、テンプレート画像と同じ画像を用いたオブジェクトの座標を、Lowe が提案した SIFT(Scale-Invariant Feature Transform)[7] を用いた画像認識を行うことで座標変化を取得する。SIFT は、画像中の特徴点検出と特徴量記述を行うアルゴリズムであり、照明変化や回転・拡大縮小に頑健である。Scratch では、オブジェクトとなるスプライトの色の変更や回転・拡大縮小を自由にできることから、そのような変更を行なった作品にも対応するため、照明変化や回転・拡大縮小に頑健である SIFT を採用する。SIFT による画像認識の手順は、以下の通りである。

1. **スケール空間における極値検出:** 拡大縮小による不変性を得るため、対象とする点の特徴をより表現可能な近傍領域の範囲を決定する。近傍領域の範囲を決めるパラメータをスケールという。スケール s の異なるガウス関数 $G(x, y, s)$ と入力画像 $I(x, y)$ を畳み込んだ平滑化画像 $L(x, y, s)$ を以下の式により作成する。

$$L(x, y, s) = G(x, y, s) * I(x, y) \quad (3.1)$$

$$G(x, y, s) = \frac{1}{2\pi s^2} \exp\left(-\frac{x^2 + y^2}{2s^2}\right) \quad (3.2)$$

特徴点候補はスケールの異なる平滑化画像間の差分を取る DoG(Difference of Gaussian) 画像 $D(x, y, s)$ から決定する。 $D(x, y, s)$ は以下の式から求める。

$$D(x, y, s) = L(x, y, s_{i+1}) - L(x, y, s_i) \quad (3.3)$$

この処理を複数の異なるスケール間で行うことで複数の DoG 画像を取得し、得られた DoG 画像から極値を検出する。極値の検出は DoG 画像を 3 枚 1 組にして行う。極値を検出する DoG 画像中のある画素に注目し、その周りの 26 近傍を比較する。注目画素が極値であった場合、その画素を特徴点候補として検出する。

2. **特徴点のローカライズ:** 特徴点候補の中にはエッジ上の点が含まれており、ノイズの影響を受けやすい。そのため、エッジ上に存在する特徴点候補を削除することで安定した特徴点を絞り込む。エッジ上の特徴点候補を削除するために、各候補における二次元ヘッセ行列を以下の式から求める。

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (3.4)$$

H から求められる第 1 固有値を α 、第 2 固有値を β とし、その比率を γ とすると以下の式が成り立つ。

$$\alpha = \gamma\beta \quad (3.5)$$

よって、

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(\gamma + 1)^2}{\gamma} \quad (3.6)$$

となり、この値を閾値処理することで不要な特徴点の除去を行う。

3. **回転角の計算**: 検出した各特徴点に対して, 回転角を割り当てる. 特徴点が発見された平滑化画像の各画素の勾配 $m(x, y)$ とその勾配方向 $\theta(x, y)$ を以下の式から求める.

$$m(x, y) = \sqrt{f_x(x, y)^2 + f_y(x, y)^2} \quad (3.7)$$

$$\theta(x, y) = \tan^{-1} \frac{f_y(x, y)}{f_x(x, y)} \quad (3.8)$$

$$\begin{cases} f_x(x, y) = L(x+1, y) - L(x-1, y) \\ f_y(x, y) = L(x, y+1) - L(x, y-1) \end{cases} \quad (3.9)$$

求めた勾配 m の大きさと勾配方向 θ から, 重み付き方向ヒストグラムを作成する. 作成したヒストグラムから最大値となる方向を特徴点の回転角として割り当てる.

4. **特徴量の算出**: 特徴点の周辺領域を, 3 で割り当てた方向を基準とした軸に回転させる. この状態で特徴量を算出するため, 回転に対する不変性を得ることができる. 特徴点に対して周囲の 16×16 画素を取り出し, この領域を 16 個の 4×4 の小領域に分割する. 各小領域に対してビン数が 8 となるヒストグラムを作成する. 結果として $8 \times 16 = 128$ 個の特徴量が算出される.
5. **特徴点のマッチング**: 2 枚の画像間の特徴点のマッチングを, 最も距離の近い点を探す最近傍探索によって行う.

本研究では, インテルが開発・公開しているオープンソースライブラリの OpenCV² を利用し, SIFT を用いた画像認識を行う.

Scratch では, ユーザが自由にオブジェクトの画像を作成することができ, オンラインサービス上に公開されている作品で使用されているオブジェクト画像が多様であることから, 全てのオブジェクトを対象とするのではなく, Scratch においてデフォルトで用意されているオブジェクト画像 339 件のみを画像認識のテンプレート画像とする.

本提案手法では, 動作するオブジェクトを捉えるため, オブジェクトの動作を命令する動きブロックを使用している作品を対象とする. 1 作品には複数の動作を含んでいるため, 連続したスナップショット中で, オブジェクトが登場しないフレームまでを 1 つの動作とする. 1 動作ごとに取得した座標を, 検索対象動作の座標の時系列データとする.

3.3 座標の時系列データの距離算出

3.3.1 時系列データの正規化

入力と抽出対象動作の座標が離れていても, 動作の軌跡が類似していれば類似動作として抽出を行うため, 手順 1 で取得した時系列データを最小値 0, 最大値 1 に正規化する.

²<https://opencv.org/>

3.3.2 DTW 距離の算出

入力の時系列データと、各動作の時系列データの距離を、動的時間伸縮法 (DTW: Dynamic time warping) を用いて算出する。DTW は、2つの時系列データの距離を動的計画法に基づいて算出する手法である。DTW では2つの時系列データの各点の距離を総当たりで算出し、全て求めた上で2つの時系列データの累積距離が最短となる経路を探し、最短経路における累積距離が2つの時系列データの距離を示す。異なる長さの時系列データ同士でも距離を算出可能である。図 3.2 に示すように、距離を算出する2つの時系列データ $S = s_1, s_2, \dots, s_{15}$, $T = t_1, t_2, \dots, t_{10}$ を横と縦に並べ、点 (i, j) における要素 s_i と t_j 間の距離を算出する。図中では、要素のマスの色が薄いほど距離が近く、濃いほど距離が遠いことを示す。時系列データ S と T の各要素の累積距離が最も短くなる経路を探すことで距離を算出する。

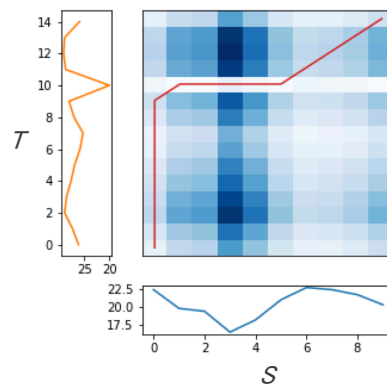


図 3.2: 動的時間伸縮法 (DTW: Dynamic time warping) の概念図

入力と検索対象の各動作の組み合わせで距離を求め、距離が近い動作を入力に類似する動作として抽出する。距離は0に近いほど類似していることを示し、値が大きくなるほど類似しないことを示す。次章では、実際に入力を行い DTW 距離を算出した結果の分析及びその結果について述べ、5章で考察を行うことで入力に類した動作を抽出可能な距離について明らかにする。

第4章 検索者の求める動作を抽出する距離の分析

4.1 概要

本章では、入力に類する動作を抽出可能な距離を明らかにするため分析を行う。著者が実装した Scratch 作品検索システムを用いて複数の入力を行い、DTW 距離の算出結果について分析を行う。本分析は動作の類似性に関するアンケートを行うことで入力との類似動作を抽出する距離を明らかにする。

4.2 検索データセット

3章で述べたように、キー入力やマウス操作などのユーザの操作を必要とするブロックを使用しておらず、動きブロックを使用している作品に含まれる、Scratch にデフォルトで用意されている画像を使用したオブジェクトの動作を検索対象とする。本分析で検索対象外とする作品に使用される、ユーザの操作を必要とするブロックを表 4.1 に示す。Aivaloglou らの公開データセット [9]¹に含まれる作品の中から以上を満たす 13,437 件の作品を用いる。

4.3 分析

本分析では、著者が実装した Scratch 作品検索システムに対して複数の入力を行い、入力と検索対象動作の DTW 距離の結果を分析する。Scratch の作品内で、オブジェクトは直線や曲線の移動を行うため、直線、曲線移動を行う以下の 2 つの動作を入力する。

- 入力 a: 左から右へ直線移動 (図 4.1a)
- 入力 b: 左から上方向へ半円を描く曲線移動 (図 4.1b)

以上に加え、より長い動作を入力した場合の距離や動作の類似性についても確認するため、以下の 2 つの動作を入力する。

- 入力 c: 左上から時計回りに直線移動 (図 4.1c)
- 入力 d: 左上から時計回りに曲線移動 (図 4.1d)

著者が実装した Scratch 作品検索システム画面に表示されるキャンバスに動作イメージの軌跡を描画することで入力を行う。

¹<https://github.com/TUDELFTScratchLab/ScratchDataset>

表 4.1: ユーザの操作を必要とするブロック

種類	ブロック	説明
動きブロック	マウスポインターへ向ける	オブジェクトの向きをマウスポインターの方向に変更する
	マウスポインターへ行く	オブジェクトをマウスポインターの位置に動かす
イベントブロック	(引数) キーが押されたとき	引数に指定したキーが押された際にスクリプトを実行する
	このスプライトが押されたとき	オブジェクトがクリックされた際にスクリプトを実行する
調べるブロック	マウスポインターに触れた	マウスポインターに触れたことを判定する
	マウスポインターまでの距離	オブジェクトとマウスポインター間のユークリッド距離を返す
	(引数) と聞いて待つ	引数に指定した文章と共に、画面下部にテキストの入力ボックスを表示する
	(引数) キーが押された	指定したキーが押されたことを判定する
	マウスが押された	マウスが押されたことを判定する
	マウスの x 座標	現在のマウスポインターの x 座標を返す
	マウスの y 座標	現在のマウスポインターの y 座標を返す

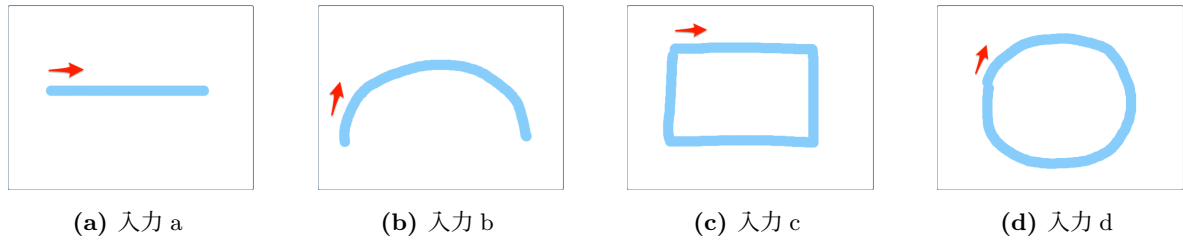


図 4.1: 分析で用いる 4 つの入力

本分析では、入力である動作のイメージに類似する動作を抽出可能な距離を明らかにするために、入力と各検索対象動作の DTW 距離算出結果から距離の異なる複数の動作を抽出し、動作の類似性をアンケートで評価する。動作のスナップショットのみを目視した場合と、動作のスナップショットに加えプログラムを目視した場合で、類似性評価の変動の有無を明らかにするため、アンケートに以下の 2 つの質問を設けた。

- 質問 1: スナップショットの動作は、入力に類似していますか？
- 質問 2: プログラムを参考に入力と同じ動きを実装できますか？

質問 1 に対しては、4 段階のリッカート尺度 (1: 類似する, 2: やや類似する, 3: やや類似しない, 4: 類似しない) で評価し、1: 類似する以外を選択した場合は、理由を記述する。質問 2 に対しては、2 値 (1: 実装できる, 2: 実装できない) で評価する。

また、アンケートで提示する動作は以下の手順で抽出する。

1. 距離が近い動作から 50 件ごとに傾きを算出
2. 傾きが 0.5 を切る距離の動作のスナップショットを目視で確認
3. 2 の結果、入力と類似しない動作であった場合、その距離以下の動作を提示ただし、データセット数が多いため、各動作の距離を小数点第 2 位で四捨五入した値が重複する動作を削除
4. 2 の結果、入力と類似する動作であった場合、2 以降の手順を再度行う

アンケートはプログラムの理解が必要であるため、Scratch のプログラムを理解できる著者を含む 3 人が回答した。アンケート結果から、回答者全員が「類似しない」と判断する動作の距離を、入力との類似動作を抽出可能な距離とする。

4.4 分析結果

入力 a のアンケート結果を表 4.2 に示す。距離 0.8 の動作までは、質問 1 は「類似する」が、質問 2 は「実装できる」が全員に選択された。距離 0.9 の動作は、初めて質問 2 で「実装できない」が選択された。「実装できない」を選択した理由は、質問 1 に対して全員に「類似する」が選択された動作は 1.6 まで続くが、1.7 以降で「やや類似する」が選択された動作が出現している。

表 4.2: 入力 a のアンケート結果

距離	質問 1 の回答数				質問 2 の回答数	
	1: 類似する	2: やや類似する	3: やや類似しない	4: 類似しない	1: 実装できる	2: 実装できない
0.3	3	0	0	0	3	0
0.4	3	0	0	0	3	0
0.5	3	0	0	0	3	0
0.6	3	0	0	0	3	0
0.7	3	0	0	0	3	0
0.8	3	0	0	0	3	0
0.9	3	0	0	0	2	1
1.0	3	0	0	0	2	1
1.1	3	0	0	0	3	0
1.2	3	0	0	0	2	1
1.3	3	0	0	0	3	0
1.4	3	0	0	0	3	0
1.5	3	0	0	0	3	0
1.6	3	0	0	0	3	0
1.7	2	1	0	0	3	0
1.8	3	0	0	0	3	0
1.9	3	0	0	0	3	0
2.0	2	1	0	0	3	0
2.1	2	1	0	0	3	0
2.2	2	0	1	0	2	1
2.3	3	0	0	0	3	0
2.4	1	1	1	0	3	0
2.5	3	0	0	0	3	0
2.6	2	0	1	0	3	0
2.7	3	0	0	0	3	0
2.8	3	0	0	0	3	0
2.9	3	0	0	0	3	0
3.0	0	3	0	0	3	0
3.1	3	0	0	0	3	0
3.2	2	0	0	1	1	2
3.3	3	0	0	0	3	0
3.4	1	0	2	0	3	0
3.5	3	0	0	0	2	1
3.6	3	0	0	0	3	0
3.7	1	0	2	0	2	1
3.8	3	0	0	0	3	0
3.9	2	1	0	0	3	0
4.0	0	1	2	0	3	0
4.1	3	0	0	0	3	0
4.2	3	0	0	0	3	0
4.3	0	0	3	0	1	2
4.4	0	0	0	3	2	1

表 4.3: 入力 b のアンケート結果

距離	質問 1 の回答数				質問 2 の回答数	
	1: 類似する	2: やや類似する	3: やや類似しない	4: 類似しない	1: 実装できる	2: 実装できない
0.9	3	0	0	0	3	0
1.2	3	0	0	0	3	0
1.3	2	1	0	0	2	1
1.4	2	1	0	0	0	3
1.5	2	1	0	0	1	2
1.6	1	1	0	1	1	2
1.7	1	2	0	0	1	2
1.8	2	1	0	0	1	2
1.9	1	1	0	1	1	2
2.0	1	2	0	0	0	3
2.1	2	0	1	0	2	1
2.2	0	0	0	3	0	3
2.3	0	2	0	1	1	2
2.4	0	0	1	2	0	3
2.5	0	1	2	0	0	3
2.6	0	0	0	3	0	3
2.7	0	0	3	0	0	3
2.8	0	0	0	3	0	3
2.9	0	0	0	3	0	3
3.0	0	0	0	3	0	3
3.1	0	0	0	3	0	3
3.2	0	0	0	3	0	3

表 4.4: 入力 c のアンケート結果

距離	質問 1 の回答数				質問 2 の回答数	
	1: 類似する	2: やや類似する	3: やや類似しない	4: 類似しない	1: 実装できる	2: 実装できない
1.7	0	3	0	0	3	0
2.2	3	0	0	0	3	0
2.6	0	2	1	0	0	3
3.0	3	0	0	0	3	0
3.3	2	1	0	0	3	0
3.7	2	0	1	0	3	0
3.8	2	0	1	0	0	3
3.9	1	1	1	0	3	0
4.0	1	1	0	1	1	2
4.1	0	0	0	3	0	3
4.3	0	0	0	3	0	3
4.4	0	0	0	3	0	3
4.5	2	1	0	0	3	0
4.6	0	0	1	2	1	2
4.7	1	1	1	0	2	1
4.8	0	0	2	1	2	1
4.9	1	0	1	0	0	3
5.0	0	1	2	0	3	0
5.1	0	0	0	3	0	3
5.2	0	0	2	1	1	2
5.3	0	0	0	3	1	2
5.4	0	0	0	3	0	3
5.4	0	0	0	3	0	3
5.5	1	2	0	0	3	0
5.6	1	2	0	0	3	0
5.7	0	1	2	0	2	1
5.8	0	0	1	2	0	3
5.9	0	1	2	0	1	2
6.0	0	0	0	3	0	3
6.1	0	0	0	3	0	3
6.2	0	0	0	3	0	3
6.3	0	0	0	3	0	3
6.4	0	0	0	3	0	3
6.5	0	0	0	3	1	2
6.6	0	0	0	3	0	3
6.7	0	0	0	3	0	3
6.8	2	0	1	0	2	1
6.9	0	0	0	3	0	3

表 4.5: 入力 d のアンケート結果

距離	質問 1 の回答数				質問 2 の回答数	
	1: 類似する	2: やや類似する	3: やや類似しない	4: 類似しない	1: 実装できる	2: 実装できない
1.7	3	0	0	0	3	0
3.2	0	0	2	1	0	3
4.3	0	0	1	2	1	2
4.4	0	2	0	1	3	0
4.5	0	0	0	3	1	2
4.6	0	0	0	3	1	2
4.7	0	0	0	3	0	3
4.8	0	0	2	1	0	3
5.0	0	0	0	3	0	3
5.1	0	2	0	1	3	0
5.2	0	1	1	1	3	0
5.3	0	1	2	0	1	2
5.4	0	2	0	1	3	0
5.5	0	2	1	0	2	1
5.6	0	1	1	1	3	0
5.9	0	1	1	1	3	0
6.0	0	0	0	3	1	2
6.1	0	0	0	3	0	3
6.2	0	2	0	1	3	0
6.3	0	0	0	3	1	2
6.4	0	2	0	1	3	0
6.5	0	0	0	3	0	3
6.6	0	1	1	1	1	2
6.8	0	0	0	3	0	3
6.9	0	0	0	3	0	3
7.0	0	0	0	3	0	3
7.1	0	0	1	2	1	2
7.2	0	0	0	3	0	3
7.3	0	2	0	1	1	2
7.4	0	2	0	1	3	0
7.5	0	0	0	3	0	3
7.6	0	0	0	3	1	2
7.7	0	0	1	2	1	2
7.8	0	0	0	3	0	3
7.9	0	0	0	3	1	2

第5章 考察

5.1 動作の類似性

5.2 プログラム

5.3 より長い動作の検索

5.4 妥当性への脅威

第6章 おわりに

本論文では、直感的な Scratch 作品検索実現に向けて、オブジェクトの座標変化を記録した時系列データを分析する手法を提案した。

謝 辭

参考文献

- [1] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Saul Silver, Brian S Silverman, Yasmin Bettina Kafai, Scratch: Programming for all, Communications of the ACM, Vol.52, No.11, pp.60-67, 2009.
- [2] Aniket Dahotre, Yan Zhang, Christopher Scaffidi, A qualitative study of animation programming in the wild, ESEM'10: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, No.29, pp.1-10, 2010.
- [3] David Weintrop, Uri Wilensky, Comparing block-based and text-based programming in high school computer science classrooms, ACM Transactions on Computing Education, Vol.18, No.1, pp 1-25, 2018.
- [4] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, Franklyn Turbak, Learnable programming: blocks and beyond, Communications of the ACM, Vol.60, No.6, pp.72-80, 2017.
- [5] Sheela Surisetty, Catherine Law, Chris Scaffidi, Behavior-based clustering of visual code, Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, pp.261-269, 2015.
- [6] Susan Elliott Sim, Medha Umarji, Sukanya Ratanotayanon, Cristina V. Lopes, How Well Do Search Engines Support Code Retrieval on the Web?, ACM Transactions on Software Engineering and Methodology, Vol.21, No.1, pp.1-25, 2011.
- [7] David G. Lowe., Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, Vol.60, pp.91-110, 2004.
- [8]
- [9] Efthimia Aivaloglou, Felienne Hermans, Jesús Moreno-León, Gregorio Robles, A dataset of scratch programs: scraped, shaped and scored, Proceeding of the 14th International Conference on Mining Software Repositories, pp.511-514, 2017.