

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	オブジェクトの動作に基づく Scratch 作品検索手法	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	類似動作を抽出可能な距離の分析	10
4.1	概要	10
4.2	検索データセット	10
4.3	検索システム	10
4.4	入力	12
4.5	定量分析	13
4.6	定性分析	13
4.7	検索評価	17
第5章	考察	22
5.1	動作に基づく検索手法の実用性	22
5.2	抽出した類似動作のプログラム	24
第6章	おわりに	26

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[3]。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[3]。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[4][5]。

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].

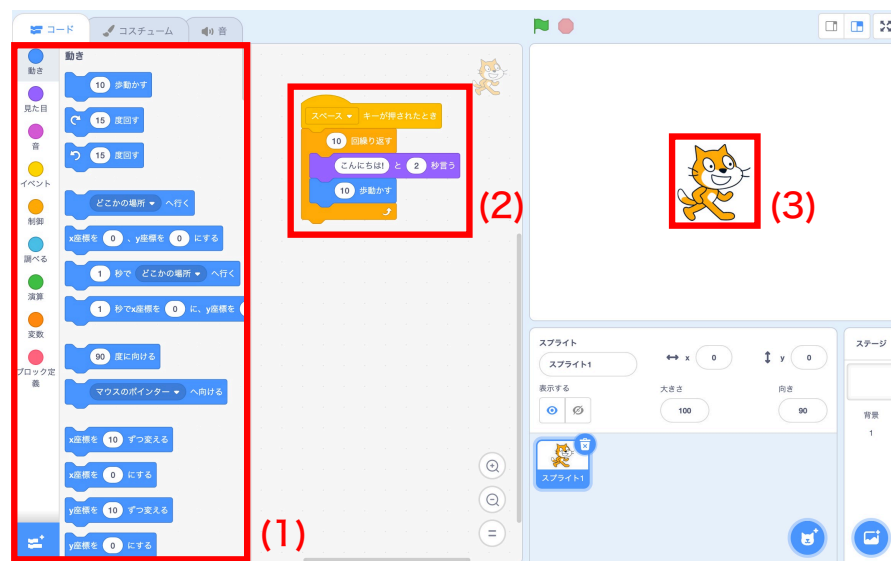


図 2.1: Scratch の作品制作画面



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



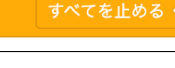
ブロックの形状	名前	性質
	ハットブロック	スクリプトの処理を開始するブロック
	スタックブロック	記述された命令を実行するブロック
	真偽ブロック	真偽値を保持し、他のブロックの引数となるブロック
	値ブロック	値を保持し、他のブロックの引数となるブロック
	C型ブロック	間にあるブロックに対して、条件分岐や繰り返しを実行するブロック
	キャップブロック	スクリプトの処理を停止するブロック

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は検索結果に表示されないため、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 [2]。 **TODO: [検索のストーリーで書く]** 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。

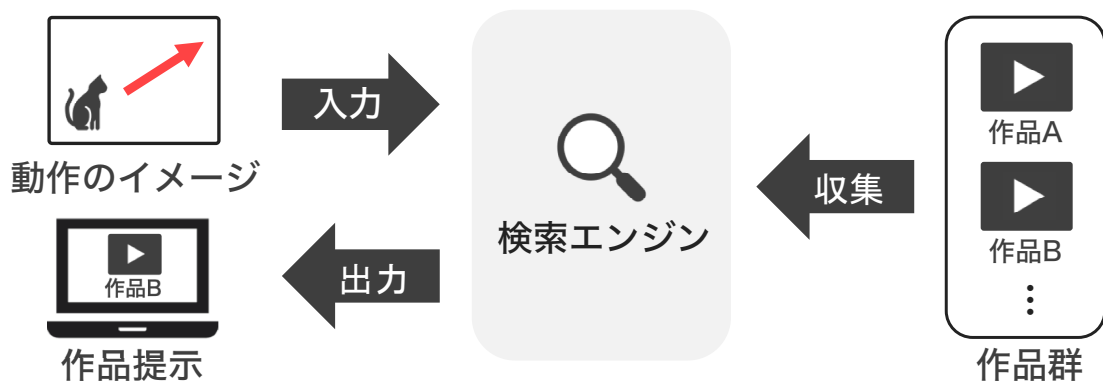


図 2.3: Scratch 作品の直感的検索システムの概略図

第3章 オブジェクトの動作に基づく Scratch 作品 検索手法

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。

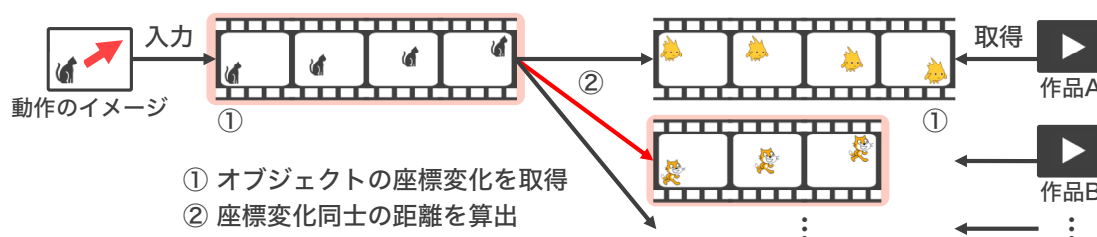


図 3.1: 動作のイメージに基づく検索手法の概略図

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

検索対象作品の収集

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

¹<https://www.selenium.dev/>

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。

画像認識による時系列データ取得

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。

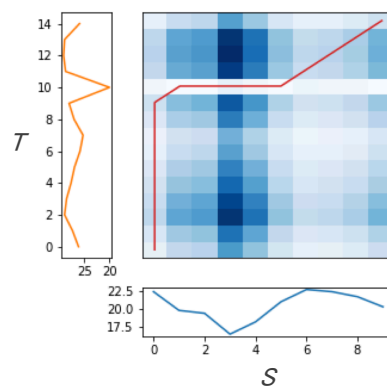


図 3.2: 動的時間伸縮法 (DTW: Dynamic time warping) の概念図

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。

続く4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る。

第4章 類似動作を抽出可能な距離の分析

4.1 概要

本章では、入力に類似する動作を抽出可能な距離を明らかにするため、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定量分析と定性分析を行う。定量分析では、距離を昇順に並べたある 2 点の傾きから距離が収束する値を求めることで、移動方向が全く異なる動作が抽出される距離を明らかにする。定性分析では、定量分析で明らかにした距離以下の動作を提示し、動作の類似性を評価するアンケートを実施することで、類似動作を抽出可能な距離を明らかにする。

4.2 検索データセット

3.2.2 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条件を満たす 13,437 件の作品を用いる。

4.3 検索システム

本分析で入力・距離算出を行うために、著者が実装した Scratch 作品検索システムを用いる。図 4.1 は検索画面を示す。図 4.1 の (1) に示すキャンバスにマウス操作で動作のイメージの軌跡を入力し、検索ボタンをクリックすることで入力と検索対象動作の距離算出を開始する。全ての動作に対しての距離算出完了後、図 4.1 の (2) に示すように、「各動作の含まれる作品の URL」「動作を行うスプライトの名称」「入力との距離」を、距離の昇順に一覧表示する。

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: Scratch 作品検索システムの画面

4.4 入力

Scratch の作品内では、オブジェクトは直線や曲線の移動が行われるため、直線・曲線移動を行う 2 つの動作を入力する。

- 入力 a: 左から右へ直線移動 (図 4.2a)
- 入力 b: 左から上方向へ半円を描く曲線移動 (図 4.2b)

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、さらに 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 4.2c)
- 入力 d: 左上から時計回りに曲線移動 (図 4.2d)

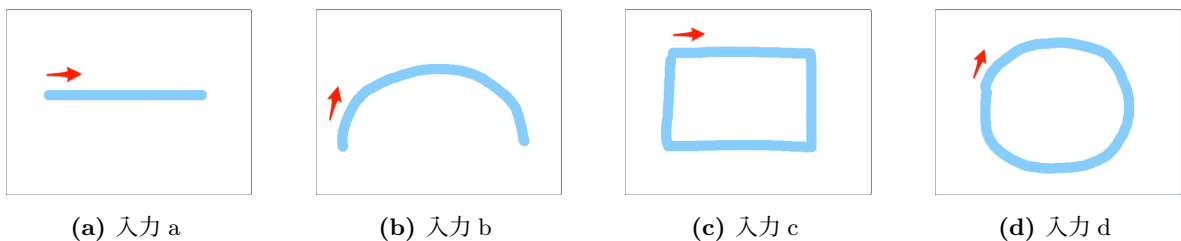


図 4.2: 分析で用いる 4 つの入力

4.5 定量分析

定量分析では、移動方向が異なる動作が抽出される距離を明らかにするために、距離を昇順に並べたある2点の傾きから距離が収束する値を求める。距離が収束する値は次に示す手順で求める。

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

各入力に対して定量分析を行なった結果を表4.2に示す。定量分析で明らかにした距離は、4.6の定性分析で実施するアンケートに提示する動作の抽出に用いる。

表 4.2: 移動方向が異なる動作が抽出される距離

入力	距離
a	4.4
b	3.2
c	6.9
d	7.9

4.6 定性分析

定性分析では、入力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提示し、動作の類似性をアンケートで評価する。提示された動作が、入力と類似するかどうかを判断するために、次の1つ目の質問を設けた。

- 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1に対して、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)で評価を行う。初めてアンケートの回答者全員が「4: 類似しない」を選択した距離を、本分析で明らかにする類似動作を抽出可能な距離とする。また、回答者にとってどのような理由で評価に繋がるかについて明らかにするために、「1: 類似する」以外を選択した場合の理由を記述してもらった。

さらに、本研究は他者のプログラムを参照することによる学習の支援を期待するため、提示された動作のプログラムを参考に入力と類似する動作を実装できるかについて判断する2つ目の質問を設けた。

- 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2に対して、2値(1: 実装できる, 2: 実装できない)で評価を行う。アンケートの回答結果から、「動作は類似しており、プログラムも参考になる」「動作は類似するがプログラムは参考にならない」といった事例の存在を明らかにする。

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名が回答した。

入力 a

入力 a のアンケート結果を表 4.3 に示す。

- 質問1: 最も距離の近い 0.3 から 1.6 の動作までは、回答者全員が「1: 類似する」を選択した。距離 1.7 の動作について1名の回答者が「2: やや類似する」を選択しており、選択理由を「移動距離が短いため」としている。距離 3.2 の動作では初めて「4: 類似しない」が選択された。1名の回答者が選択しており、選択理由を「移動距離が非常に短いため」としている。距離 1.7 と距離 3.2 の事例から、オブジェクトの移動方向が入力と類似する場合にも、回答者によっては移動距離を理由に類似しないと判断することが考えられる。距離 4.3 の動作では回答者全員が初めて「3: やや類似しない」を選択し、次の距離 4.4 の動作では回答者全員が初めて「4: 類似しない」を選択した。距離 4.3 の動作は、右方向へ移動しながらやや上方向への移動もしているため、回答者全員が「上方向への移動を含む」を理由に「3: やや類似しない」を選択した。また、距離 4.4 の動作は移動方向が類似していないために、回答者全員が「4: 類似しない」を選択した。距離 4.3 と距離 4.4 の事例から、移動方向が回答者全員の類似性の判断に影響することが考えられる。

初めて回答者全員が「4: 類似しない」を選択した距離が 4.3 であるため、入力 a との類似動作を抽出可能な距離は 4.3 未満となる。

- 質問2: 最も距離の近い 0.3 から 0.8 の動作までは、回答者全員が「1: 実装できる」を選択した。距離 0.9 の動作について1名の回答者が「2: 実装できない」を選択した。距離 0.9 の動作のプログラムは、オブジェクトの持つ現在の「向き」に移動させる「(引数) 歩動かす」ブロックを用いている。「2: 実装できない」を選択した回答者は自由記入欄に実装できない理由を記述しており、「オブジェクトの向きが左に初期化されているため、右方向への移動が実装できない」としている。一方、「1: 実装できる」を選択した回答者は「オブジェクトの向きを右に初期化することで実装できる」と述べている。距離 0.9 の事例から、オブジェクトの向きを入力の変数に初期化することで実装できる事例の存在が明らかになった。「2: 実装できない」が選択された他のいくつかの動作についても、移動方向の初期化により実装できる事例が見られた。距離 4.4 の動作は、質問1で回答者全員が「4: 類似しない」を選択したにもかかわらず、2名が「1: 実装できる」を選択した。距離 4.4 の動作は、距離 0.9 の動作と同様「(引数) 歩動かす」ブロックを用いており、オブジェクトの向きを初期化することによって右方向への実装が可能となることから、動作は類似しないにもかかわらず、プログラムのみを見ることによって入力との類似動作を実装可能な事例の存在が明らかになった。

入力 b

入力 b のアンケート結果を表 4.4 に示す.

- 質問 1: 最も距離の近い 0.9 と, 2 番目に距離の近い 1.2 の動作に対して, 回答者全員が「1: 類似する」を選択した. 距離 1.5 から 1.8 の動作まで 1 名の同じ回答者が「2: やや類似する」を選択している. いずれの動作も, 入力より横方向の移動距離が短く縦方向の移動距離が長くなっていることから, 選択理由として「角度が異なる」と述べている. また, 距離 1.6 の動作について初めて「4: 類似しない」が選択された. 1 名の回答者が選択しており, 理由として「(画面端に当たって) 反射しているように見える」と述べている. 距離 1.5 から 1.8 の動作の事例から, 類似性の判断のために移動角度に着目することが考えられる. 距離 2.2 の動作は左上から右下へ移動する動作であり, 移動方向が類似しないために初めて回答者全員が「4: 類似しない」を選択した. 以降の距離で, 2 つの動作で「移動方向は類似するが, 曲線移動をしていない」などを理由に「2: やや類似する」が選択されたが, 多くの動作で「3: やや類似しない」「4: 類似しない」が選択されている. 距離 2.8 以降の全ての動作について, 回答者全員が「4: 類似しない」を選択した.

初めて回答者全員が「4: 類似しない」を選択した距離が 2.2 であるため, 入力 b との類似動作を抽出可能な距離は 2.2 未満となる.

- 質問 2: 最も距離の近い 0.9 から 1.3 の動作まで, 回答者全員が「1: 実装できる」を選択した. 距離 1.4 の動作について, 質問 1 で「1: 類似する」「2: やや類似する」を選択したにもかかわらず, 回答者全員が「2: 実装できない」を選択した. 一見, 入力のような曲線を描くように見えたが, プログラムは曲線移動でなく直線移動が実装されていることが明らかになった. そのため, 回答者全員が入力のような動作を実装できないとしている. また, 距離 1.5 や 1.8 の動作のプログラムはいずれもオブジェクトの向きを乱数を生成する「(引数 1) から (引数 2) までの乱数」ブロックで決定しているため, 2 名の回答者が「2: 実装できない」を選択した事例がみられた. 本研究では「(引数 1) から (引数 2) までの乱数」ブロックを用いた作品を除外していないため, 3.2.2 で作品を収集した際の実行結果に依存する. 距離 1.4 や 1.5, 1.8 などの動作から, 動作が類似するにもかかわらず, プログラムを見ることで実際には入力のような実装ができない事例の存在が明らかになった. 距離 2.4 以降の全ての動作について, 回答者全員が「4: 実装できない」を選択した.

入力 c

入力 c のアンケート結果を表 4.5 に示す.

- 質問 1: 最も距離の近い 1.7 の動作に対して, 回答者全員が「2: やや類似する」を選択した. 距離 1.7 の動作は, 入力同様, 始点が左上の時計回りに四角形を描いているが, 4 辺のうち 3 辺が斜めに描かれるため, 綺麗な四角形でないとして「2: やや類似する」を選択した. また, 距離 2.6 の動作でも同様の理由で「2: やや類似する」あるいは「3: やや類似しない」が選択された. 距離 3.7 から 4.0 までの動作は, いずれも同じ作品に含まれ, 四角形を描く最初の右方向への 1 辺が短い, あるいは欠けている動作である. 最初の右方向への 1 辺を理由

に、1名が全てに対して「3: やや類似しない」を選択した。一方、「1: 類似する」「2: やや類似する」を選択した回答者もいる。距離 3.3 と 3.8 で「2: やや類似する」を選択した回答者は、理由として「一部動きが欠けており、方向転換の際に曲線を描いているように見えるが、全体的に類似している」と述べている。距離 4.1 の動作について、初めて回答者全員が「4: 類似しない」を選択した。また、距離 4.1 から 4.4 までの全ての動作について、四角形ではなく円を描く動作であることを理由に回答者全員が「4: 類似しない」を選択した。しかし、以降の距離には、2名以上が「1: 類似する」「2: やや類似する」を選択した動作が複数存在する。距離 4.5, 4.9, 5.5, 5.6 はいずれも四角形の4つの頂点にのみ移動する動作であり、辺を描いて少しずつ移動するものではないが、「2: やや類似する」を選択した回答者は「頂点を移動しているが四角形に見える」と述べている。

初めて回答者全員が「4: 類似しない」を選択した距離が 4.1 であるため、入力 c との類似動作を抽出可能な距離は 4.1 未満となる。

- 質問 2: 最も距離の近い 1.7 から 4.0 の動作のうち、距離 2.6 の動作を除いて回答者全員が「1: 実装できる」を選択した。距離 2.6 の動作のプログラムは画面端に跳ね返りながら移動を繰り返す実装であり、四角形に移動させるための実装ではない。そのため、距離 2.6 の動作について回答者全員が「2: 実装できない」を選択した。質問 1 では「2: やや類似する」を選択した回答者が 2 名いるため、動作が類似していても、プログラムを見ることで入力のような移動をするための実装がされていない動作の存在が明らかとなった。また、質問 1 で距離 2.6 の動作が綺麗な四角形でないと述べられた理由は、入力のような移動をするための実装でなかったためであると考えられる。

入力 d

入力 d のアンケート結果を表 4.6 に示す。

- 質問 1: 回答者全員が「1: 類似する」を選択した動作は最も距離の近い 1.7 の動作のみとなった。以降の距離では「3: やや類似しない」「4: 類似しない」を多く選択している。距離 4.5 の動作において、初めて回答者全員が「4: 類似しない」を選択した。距離 4.5 の動作は、四角形に移動する動作であり、「4: 類似しない」を選択した理由として回答者全員が「四角形を描く直線移動である」と述べている。ただし、距離 4.5 以降において、2名以上が「2: やや類似する」を選択した動作も存在する。一例として、距離 5.1 の動作は入力よりも非常に小さい円を描いており、2名の回答者は円が小さいことを理由に「2: やや類似する」を選択した。残りの 1 名は、その場でオブジェクトが回転しているように見えるとして「4: 類似しない」を選択した。距離 5.4, 6.2, 6.4 の動作も同様の理由で同じ回答者が同じ選択をした。この事例から、回答者によって動作の見方が異なることが明らかになった。

初めて回答者全員が「4: 類似しない」を選択した距離が 4.5 であるため、入力 d との類似動作を抽出可能な距離は 4.5 未満となる。

- 質問 2: 質問 1 において、回答者全員が「1: 類似する」を選択した距離 1.7 の動作以外にも、回答者全員が「1: 実装できる」を選択した動作が見られる。質問 1 の結果において挙げた

距離 5.1, 5.4, 6.2, 6.4 の動作は、回答者全員が「1: 実装できる」を選択した。自由記述欄にて、質問 1 で「4: 類似しない」を選択した回答者は「プログラムを見ることで、その場でオブジェクトが回転しているのではなく、円を描いていることがわかった」と述べている。一見動作が類似していなくても、プログラムを参照することにより実装可能な類似動作を発見できることが明らかになった。

4.7 検索評価

本研究で提案する検索手法の精度を評価するために、各入力に対する距離算出結果のうち、距離の近い上位 20 件を用いて 4.6 と同様にアンケートを実施した。検索評価のために、平均逆順位 (MRR: Mean Reciprocal Rank) を算出する。MRR は、ランキング中に初めて正解が出現した順位 $rank$ の逆数を全ての入力 Q について平均したランキング指標であり、式 4.1 で求められる。

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \quad (4.1)$$

アンケートの結果から、初めて回答者全員が質問 1 に対して「1: 類似する」を選択した動作を正解とし、MRR を算出した結果 0.88 となった。入力 c のみ正解が出現した順位が 2 となり、入力 a, b, d において正解が出現した順位はいずれも 1 であった。MRR の算出結果は 0.5 を超えており、提案手法による検索において上位に類似動作が出現することが期待できる。

表 4.3: 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 4.4: 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 4.5: 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
5.5	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 4.6: 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

本章では4章で実施した分析結果から、動作に基づく検索手法の実用性と、抽出した類似動作のプログラムについて考察する。

5.1 動作に基づく検索手法の実用性

入力 a から入力 d まで、類似動作を抽出可能な距離は異なる結果となった。入力 a, c, d の類似動作を抽出可能な距離はそれぞれ 4.3 未満, 4.1 未満, 4.5 未満であるのに対し、入力 b は 2.2 未満と距離が近い結果となった。

入力 b において類似動作を抽出可能な距離が近くなった要因を探るために、回答者全員が類似すると判断した距離 0.9 の動作 1 と、回答者全員が類似しないと判断した距離 2.2 の動作 2 の距離算出および図 5.1 に示す座標変化のグラフ作成を行なった。2つの動作の距離は 1.6 であり、入力 b との類似動作を抽出可能な距離 2.2 未満となっている。図中の横軸は時間 (取得したスナップショットの番号) を示し、縦軸は図 5.1a では x 座標、図 5.1b では y 座標を示す。各動作の各点にとって最も距離の近い点が灰色の線で結ばれている。動作 2 のスナップショットは左上から右下への直線移動をしているため、動作 2 の y 座標は値が大きくなっていくと考えたが、図 5.1b から実際には y 座標が動作 1 と同様に、値が小さくなったあとに 3 番目のスナップショットから大きくなっていることがわかる。スナップショットを確認した結果、スナップショット中でオブジェクトの画像が変更されており、画像認識の際に取得したオブジェクトの中心座標に変化があることが、動作の見た目と実際の座標変化に違いが見られた要因であると考えられる。そのため、座標変化取得のための画像認識を行う際に、テンプレート画像の中心座標を調整することが必要であることが明らかになった。

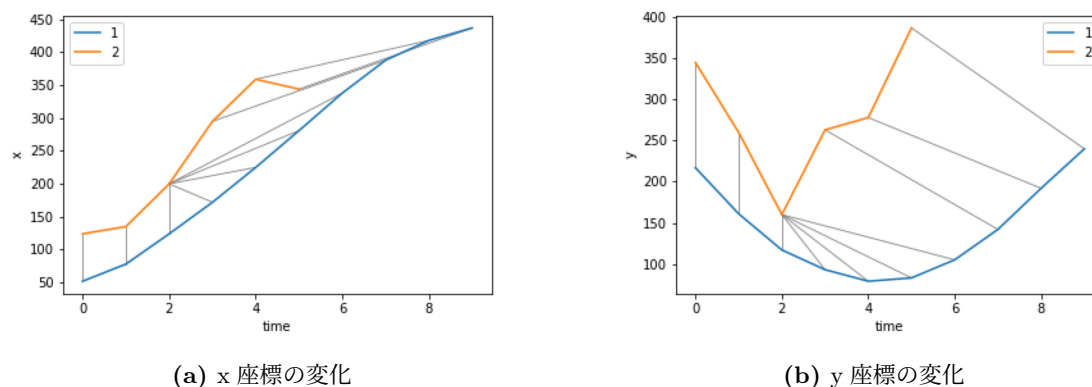


図 5.1: 動作 1 と動作 2 の座標変化グラフ

また、入力 b, c, d において移動方向は類似するが曲線 (直線) 移動ではなく直線 (曲線) 移動をする動作が見られた要因を探るために、入力 c において回答者全員が類似すると判断した、入力同様直線移動を行う距離 2.2 の動作 3 と、移動方向は類似するが曲線移動をする距離 4.1 の動作 4 の距離算出および図 5.2 に示す座標変化のグラフ作成を行なった。2 つの動作の距離 3.8 であり、入力 c との類似動作を抽出可能な距離 4.1 未満となっている。図中の横軸は時間 (取得したスナップショットの番号) を示し、縦軸は図 5.2a では x 座標、図 5.2b では y 座標を示す。各動作の各点にとって最も距離の近い点が灰色の線で結ばれている。直線移動を行う動作 3 は、x 座標が変化する際に y 座標が変化せず (スナップショット 0 6 番など)、y 座標が変化する際に x 座標が変化しない (スナップショット 6 10 番など) ことが、曲線移動を行う動作 4 は、x 座標・y 座標共に変化し続けていることがグラフから読み取れる。しかし、値が増加した点同士や減少した点同士で灰色の線が引かれていることから、値の増減の仕方が類似しており距離が近くなったと考えられる。直線移動と曲線移動を正しく区別し検索するためには、より細かい時間で座標変化を取得する必要があると考える。

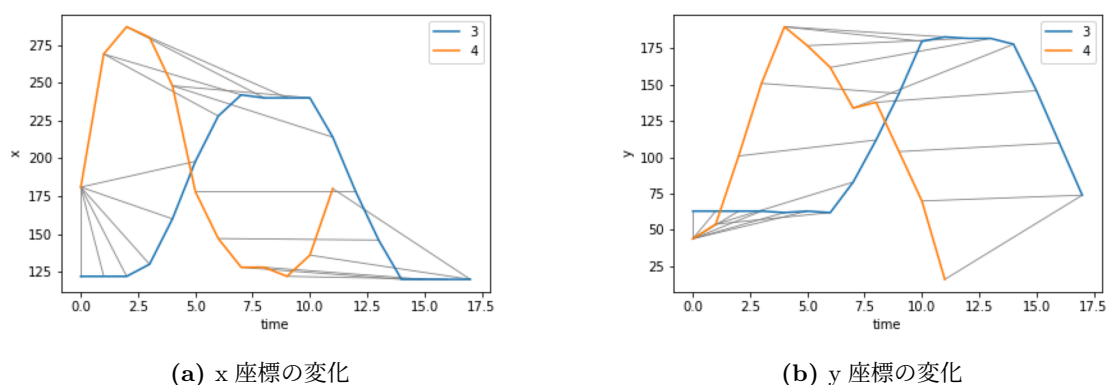


図 5.2: 動作 3 と動作 4 の座標変化グラフ

本研究では複数の入力における類似動作を抽出可能な距離を明らかにしたが、入力には無数のパターンが存在するため、全ての入力に対して類似動作を抽出可能な距離を決定することは不可能である。また、類似動作を抽出可能な距離以降にも回答者が類似すると判断する動作がいくつか見られた。これらのことから、検索結果の動作を抽出する距離を一意に定めるのではなく、4.5 のような手法を用いて、入力と移動方向が異なる動作を抽出する距離をおおまかに求めることが適切である。

動作に基づく検索手法の精度を向上させるためには、画像認識や座標変化を取得する時間間隔といった検索の前処理における工夫や類似動作を抽出する距離の見直しの必要性が明らかになった。しかし、4.6 で実施したアンケート結果からいずれの入力においても回答者全員が類似すると判断する動作の存在が複数明らかになり、4.7 の検索評価の結果からランキング上位に類似動作が出現することが期待できるため、動作に基づく作品検索は実用性があると考えられる。

5.2 抽出した類似動作のプログラム

4.6で実施したアンケートへの回答時に、複数の実装方法の存在する動作が見られた。入力bとの距離算出結果が0.9の動作5と、2.1の動作6のプログラムを例に挙げ、複数の実装方法が検索可能であることの与える学習への影響について考察する。図5.3は動作5と動作6のプログラムを示す。ただし、「見た目ブロック」といったオブジェクトの動作に直接関係しないブロックは省略している。図5.3aに示す動作5のプログラムは、テキストベースのプログラミング言語におけるfor文に該当する「(引数) 回繰り返す」ブロックを用いて、オブジェクトの移動と移動方向の変更を行なっている。一方、図5.3bに示す動作6のプログラムは、座標の指定と向きの変更を行うブロックを必要数並べることで実装している。動作5のプログラムは、「(引数) 回繰り返す」ブロックを用いることで動作6のプログラムよりも可読性が高くなっている。動作6のプログラムは可読性が低いですが、ブロックが1つずつ並べられていることから、どのような順序でオブジェクトが動作しているかを理解しやすくなっている。動作5, 6は、動作は類似するが実装方法に相違点があり、いずれのプログラムにも利点がある。このことから、実装方法の異なる類似動作を提示することは、プログラムの実装方法とより良い実装の両方の学習に繋がり、特に初学者も多く利用しているScratchにおいて有用であると考ええる。完全一致による検索であるキーワード検索やプログラム検索でなく、動作に基づく検索をすることによって類似動作の多様な実装方法を検索することが可能となった。



(a) 動作 5 のプログラム



(b) 動作 6 のプログラム

図 5.3: 動作 5 と動作 6 のプログラム

第6章 おわりに

本研究では、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提案し、複数の入力と検索対象動作の距離算出結果から、類似しない動作を抽出する距離について定量分析や動作の類似性について定性分析を実施した。分析結果から、次の知見が得られた。

- 本研究で提案した手法により、類似動作を検索可能であることが明らかになった。さらに、画像認識や座標変化取得の時間間隔といった検索の前処理における工夫や、類似動作を抽出する距離の見直しによって検索精度の向上が期待できる。
- オブジェクトの動作に基づく検索により、類似動作の多様な実装方法を検索可能であることが明らかになった。実装方法の異なる類似動作を提示することは、プログラムの実装方法とより良い実装の両方の学習に繋がり、特に初学者も多く利用している Scratch において有用である。

本研究で提案した手法により、ユーザのイメージする動作から容易に作品を検索することができ、他者の作品を参照することによる実装方法学習の支援を期待する。さらに手法を改善することによって、より精度の高い作品検索の実現を期待する。

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] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [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.