

2020 年度（秋学期）

卒業研究 3 (FD)：THAWONMAS Ruck 担当

卒 業 論 文

決定木による Minecraft の集落の自動撮影

立命館大学 情報理工学部
知能情報コース 4 回生 2600170477-9

JIA HAO

要旨

背景: 近年, Minecraft の集落生成という新しい課題に関する研究が行われた. Generative Design in Minecraft competition(GDMC) とは, 未知な Minecraft ワールド上で AI プログラムを用いて集落を生成し, 複数の観点から評価される大会である. 現時点では, ほとんどの作品は少数の審査員によって手動で評価され, 宣伝動画が作成される. しかしながら, 提出した作品数が増えることに伴い, 審査と宣伝動画の作成の負荷も大きくなった. この問題に対し, 本研究は審査動画を自動生成できる自動撮影システムを提案した. 審査動画を自動生成し動画を公開することで, より多くの人が評価プロセスに参加できる. これにより, 問題を解決できる可能性がある.

目的: 決定木による Minecraft の集落の自動撮影システムの開発とその性能検証を目指す. 決定木予測に基づき, 人間が Minecraft の集落を撮影しているときの動きを学習することで, 人間らしいカメラワークを自動生成できるシステムの構築の実現を目指す.

評価方法: 決定木予測による生成したカメラワークの人間らしさを評価するため, 学習モデルの学習精度を評価する必要がある. k -分割交差検証法で各モデルの平均精度を求め, その中最も得点の高いモデルを選んで精度検査を行う. 精度検査の結果から, 人間の動きに関する予測がどの程度までできるかを評価する.

結果: 訓練マップにおいては, 木の最大深さ制限がないときの決定木の予測精度は 1.0 に到達し, アクションの再現ができることが確認できた. テストマップに対して, 最も性能が良いモデルを選択し予測した結果, 予測精度は 0.70 に達した.

結論: 本システムにおいて訓練マップに対する有用性が確認できた. また, テストマップにおいては, 正確の予測が七割程度となった.

目次

第 1 章	はじめに	1
1.1	研究背景	1
1.2	研究目的	2
1.3	関連研究	2
1.4	論文構成	2
第 2 章	提案手法	3
2.1	概要	3
2.2	前処理	3
2.2.1	特徴量とラベルの収集手法	3
2.2.2	画像の分割処理	4
2.2.3	特徴量の抽出	5
2.2.4	ラベルの定義	6
2.3	決定木モデルの構築	6
2.4	ワールドへの適応	8
2.5	評価方法	9
第 3 章	実装	10
3.1	前処理	10
3.2	決定木モデルの構築と交差検証	10
3.3	精度検査	10
3.4	ワールドへの適応	10
3.5	ハイパーパラメーター	13
3.5.1	オプティカルフロー	13
3.5.2	決定木	13
第 4 章	実験	14

4.1	実験環境	14
4.2	ワールドに関する初期条件	14
4.2.1	静的初期条件	14
4.2.2	動的初期条件	15
4.2.3	動的初期条件の設定方法	16
4.3	実験で使用するワールド	22
4.4	訓練データの取得	23
4.5	実験結果	23
4.5.1	モデルの比較	23
4.5.2	モデルにおける精度検査	25
4.6	考察	25
第 5 章	おわりに	26
5.1	まとめ	26
5.2	今後の課題	26
謝辞		27
参考文献		28
付録		30
付録.A	PreprocessingAndDataCollection.py	30
付録.B	CrossValidation.py	38
付録.C	DataTest.py	41
付録.D	replayPredict.py	43
付録.E	NBTEdit.py	52

目次

2.1	特徴量とラベルを収集する手法	4
2.2	画像の分割処理	5
2.3	One frame of the Yosemite sequence and the corresponding true velocity field. G Farneback, Two-Frame Motion Estimation Based on Polynomial Expansion, (2003)[6] より引用	6
2.4	本研究におけるオプティカルフローの可視化	7
2.5	決定木の入力と出力	7
2.6	ワールドへの適応	8
4.1	ワールドを上空から見るイメージ	17
4.2	Minecraft におけるブロックの独自タイプ ID(20 以内)[9] より引用 . . .	18
4.3	Minecraft におけるブロックの独自タイプ ID(100-200 以内)[9] より引用	19
4.4	上空から見る集落がないときのブロックの種類分布	20
4.5	上空から見る集落があるときのブロックの種類分布	20
4.6	例：重み付き表記とカメラの初期方向	21
4.7	9 つの実験ワールド	22
4.8	本実験における k-分割交差検証法	24
4.9	モデルの精度比較	24

表目次

3.1	本研究で用いる calcOpticalFlowFarneback 関数のハイパーパラメーター	13
3.2	本研究で用いる決定木の構築のハイパーパラメーター	13
4.1	静的初期条件	15
4.2	動的初期条件	16
4.3	精度検査の結果	25

第 1 章

はじめに

1.1 研究背景

Minecraft は、立方体のブロックで世界が構成されるオープンワールドサバイバルゲームである。主なプレイ方法は、既存のブロックを採掘し、そのブロックを用いてツールの製造や建築物の構築をすることである。このゲームでは多くのプレイヤーが家や集落の建設に熱中しており、プレイヤーたちが建築物を建てる過程を表す動画はインターネット上で注目されている。Minecraft の中で 1 つのゲームシーンはワールドと呼ぶ。Generative Design in Minecraft competition (GDMC) は参加選手たちが、未知の Minecraft ワールドで集落を生成できる AI プログラムを作成し競技する大会である。この大会に提出されたワールドは顧問委員会のメンバーである審査員によって適応性、美学など様々な観点から評価される [1]。大会は 2018 年から始まり、2020 年 9 月まで 3 回行われた。現時点の作品の評価は、少数の審査員によって手動で行われる。しかしながら、現行の評価方式は、日々増えている作品数への対応が難しいという問題点がある。この問題に対し、本研究では審査動画を自動生成するための自動撮影システムを提案する。審査動画を生成し動画を公開することで、より多くの人が評価プロセスに参加することができる。それにより、問題点を解決することが可能であると考えられる。

機械学習には、特徴量を抽出して特徴量にラベルをつけ、指定した学習手法で既存の特徴量を学習することで予測のラベルを出力するという、一般的な流れがある。近年、機械学習の手法には、決定木学習がよく使われており、これによる分類モデルは、その分類にいたる過程が容易に解釈できるため人間の意思決定の中のあいまいな情報と複雑な意思決定も階層的に記述し処理することができる。本研究において人間が撮影時のカメラワークに対する学習は、まさに複雑であいまいな決定である。よって、決定木方式を使用することにした。また Minecraft のゲーム画面は 2 次元であり、その特徴を捉えるため、決定木に入力する特徴量はゲーム画面から計算したカラーエントロピーとオプティカルフローと

する.

1.2 研究目的

本研究では, 決定木による Minecraft の集落の自動撮影システムの開発とその性能検証を行う. 決定木予測に基づき, 人間が Minecraft の集落を撮影しているときの動きを学習することで, 未知のワールドで人間らしいカメラワークの予測を目指す.

1.3 関連研究

決定木に関する研究は既に行われている. データの情報利得を基本とする決定木の交通行動分析への適用可能性を示した研究 [2] やコーパスから決定木を構成し日本語係受け解析に適用する手法を提案した研究 [3] がある. カラーエントロピーに関して, エントロピーを用いてデジタル画像の色による分割を試みた研究 [4] があり, オプティカルフローに関して, オプティカルフローの予測誤差を基づく Network Creation Games(NCGs) model を構築する研究 [5] も存在している.

1.4 論文構成

本論文の構成は以下の通りである. まず, 第 1 章で研究の背景, 目的及び関連研究について述べた. 次に, 第 2 章で本研究の提案手法について述べる. そして, 第 3 章では, 本システムの詳細の実装と機能について説明する. さらに, 第 4 章では, 実験の初期条件, 評価実験及び実験の結果を記録し, 実装したシステムの有用性を検証する. 最後に第 5 章では, 本論文のまとめと, 今後の展望について述べる.

第 2 章

提案手法

2.1 概要

本章では、本研究において提案した手法について説明する。機械学習には、特徴量を抽出し、特徴量にラベルをつけ、指定した方式で既存の特徴量を学習することで予測のラベルを出力するという、一般的な流れがある。本研究では、決定木を用いて学習することとした。前処理として、カメラを Minecraft ワールドで回し、プレイ中のゲーム画面の特徴量とラベルを収集する。そして収集した特徴量とラベルを用いて決定木の学習モデルを構築する。その後、構築した決定木を未知なワールド上に適応することによって、カメラワークの予測を行うシステムを実現した。以下、提案手法の詳細について述べる。

2.2 前処理

2.2.1 特徴量とラベルの収集手法

特徴量とラベルを収集する手法として、本研究では Minecraft ワールドにおけるカメラワークの運動形式を提案した。図 2.1 に示すように、ゲームに入ってから、カメラを前方にしか進めないように設定する。その後、2 秒ごとに前進を止め、停止した瞬間の特徴量を抽出し、人間に判断を求める。判断としては、左折、右折、直行の 3 種類のカメラにおける改変がある。左折は左に 30 度、右折は右に 30 度曲がる改変であり、直行は角度不変のままである。人間が判断を下す後は、その改変をカメラに適応し、ラベルとして抽出された特徴量とペアにして保存する。次にカメラは前進を続ける状態に戻る。前に述べたのプロセスを繰り返すことで、特徴量とラベルを収集する。

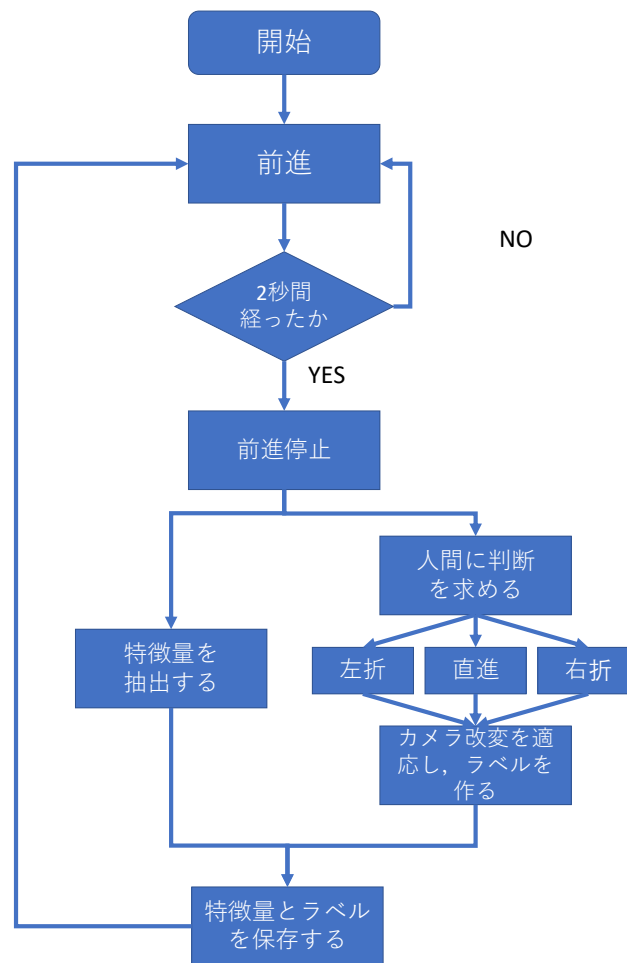


図 2.1 特徴量とラベルを収集する手法

2.2.2 画像の分割処理

ラベルには3方向があるため、ゲーム画像の3つの方向から特徴を捉えることが必要だと考えられる。そのため、まずはゲーム画像の分割を行う。図 2.2 に示すように、ゲーム画面は長方形であり、縦辺は h 、横辺は w とする。ここで画像を横辺 w に沿って平均3分割する。分割した画面の縦辺は h 、横辺は $w/3$ となる。右、中央、左の画像はそれぞれ P_{left} , P_{center} , P_{right} とする。

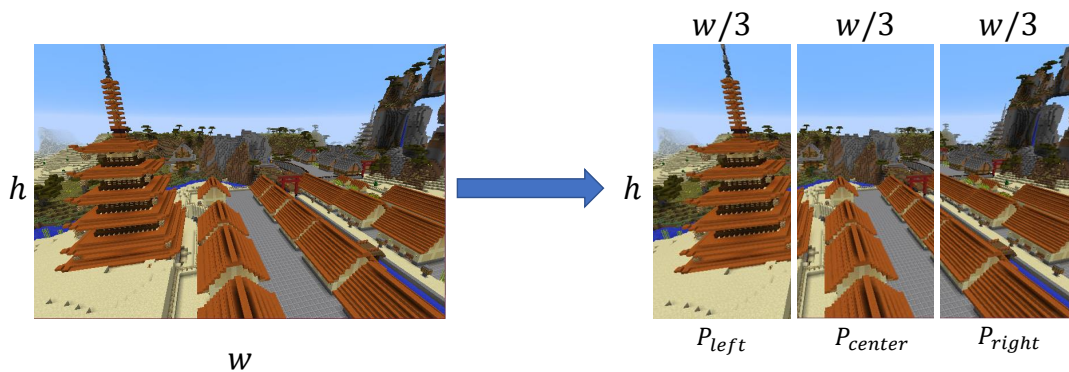


図 2.2 画像の分割処理

2.2.3 特徴量の抽出

決定木の学習モデルを構築するには、まず特徴量を入力する必要がある。本研究で用いる特徴量には2種類があり、それぞれカラーエントロピーとオプティカルフロー強度が含まれている。カラーエントロピーは各画像の色に関する情報量を示す。そして、オプティカルフロー強度は各ピクセルの動きの強度を測ることができる。各特徴量に関する先行研究より、この2つの特徴量によるカメラアクションの予測は可能であると仮説を立てることができる。

カラーエントロピー

デジタル画像は各ピクセルにRGBのデータ(0-255)が格納されるため、各ピクセルを調べ、カラーエントロピー H を算出することができる。式(2.1)ではその処理を行い、 p_i はピクセルが持つ数値の確率分布である。また一つのピクセルには3つのチャンネルR, G, Bがあるため、本研究では3つのチャンネルのエントロピー H_R , H_G , H_B をそれぞれ計算し、平均を取った。式(2.2)ではその計算を行う。その後、分割した画像 P_{left} , P_{center} , P_{right} から算出したカラーエントロピーをそれぞれ H_{left} , H_{center} , H_{right} として標準化を行う。

$$H(x) = - \sum_{i=0}^{255} p_i(x) \log_2(p_i(x)) \quad (2.1)$$

$$H = \frac{H_R + H_G + H_B}{3} \quad (2.2)$$

オプティカルフロー強度

Gunnar Farneback のアルゴリズム [6] を用いて、密なオプティカルフローを求めることができる。このアルゴリズムでは、図 2.3 に示すようにオプティカルフローベクトルを格納した 2 チャンネルの配列が返される。その 2 チャンネルの配列の中に、オプティカルフローの強度と角度が格納されている。そして本研究でゲーム画像の入力によって計算したオプティカルフローの強度と角度を可視化するものは図 2.4 に示す。本研究では画像のピクセルごとにオプティカルフローの強度のみを取り出し総和を計算する。分割した画像 P_{left} , P_{center} , P_{right} から算出するオプティカルフロー強度の総和をそれぞれ S_{left} , S_{center} , S_{right} として標準化を行う。

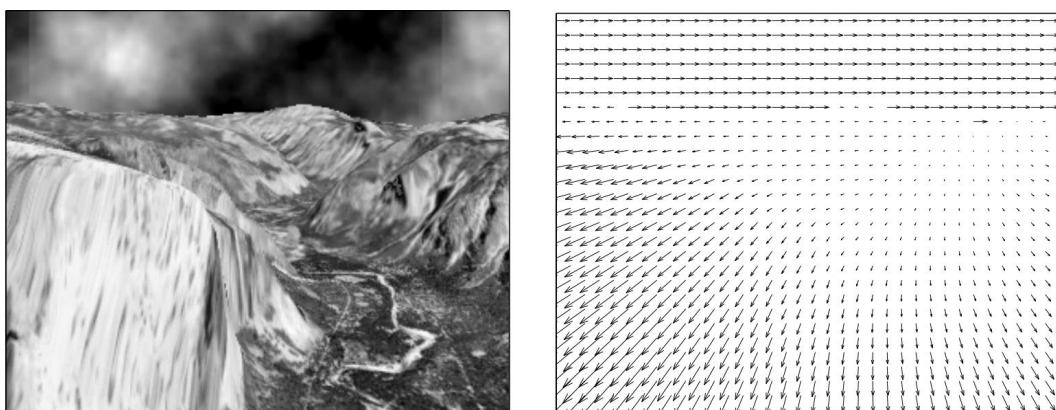


図 2.3 One frame of the Yosemite sequence and the corresponding true velocity field. G Farneback, Two-Frame Motion Estimation Based on Polynomial Expansion, (2003)[6] より引用

2.2.4 ラベルの定義

現時点では、提案したカメラワークの運動形式に左折、直行、右折の計 3 つのアクションがある。それを決定木の学習に用いるラベルとする。

2.3 決定木モデルの構築

前処理から得られたデータを用いて、決定木を訓練し、モデルを作成する。決定木の構築について、本研究では Scikit-learn[7] のアルゴリズムを用いる。そしてデータの均衡を

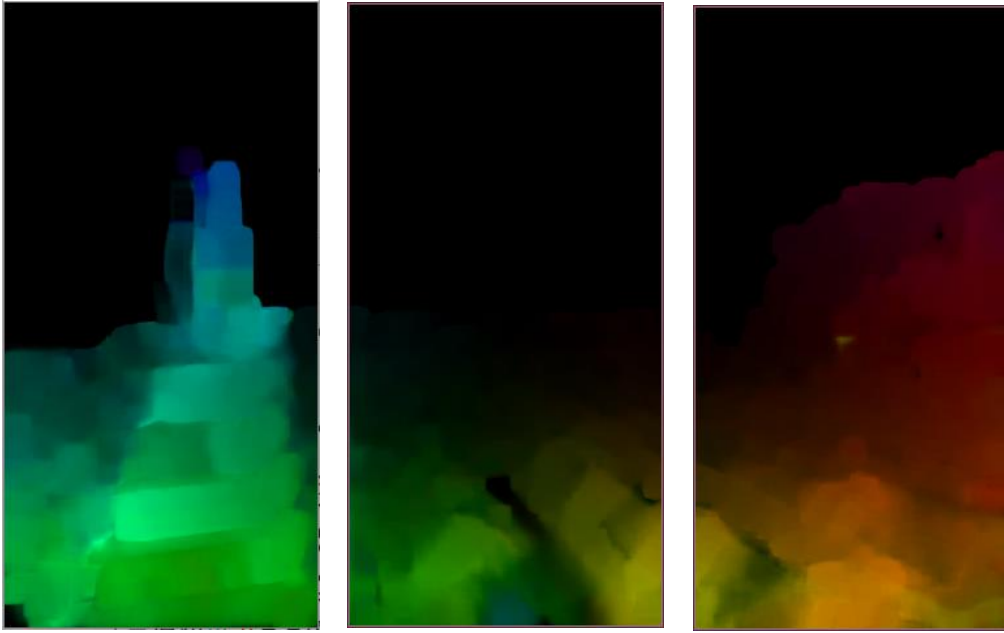


図 2.4 本研究におけるオプティカルフローの可視化

保証するため、クラスの重みを同様に設定する．また、同じ結果を再現できるように、決定木のランダムシードを指定する．最後に決定木の予測における入力と出力のイメージは図 2.5 が示している．

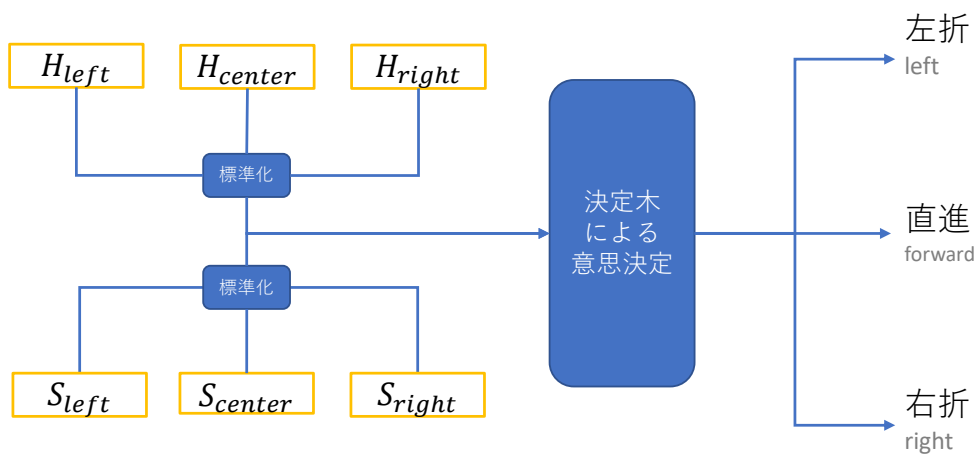


図 2.5 決定木の入力と出力

2.4 ワールドへの適応

図 2.6 に示すように，ゲームに入ってから，カメラを前方にしか進めないように設定する．その後，2 秒ごとに前進を止め，停止した瞬間の特徴量を抽出し，図 2.5 のように決定木による意思決定を出力してカメラの改変を適応する．次にカメラは前進を続ける状態に戻る．前に述べたのプロセスを繰り返すことで，自動撮影システムを実装する．

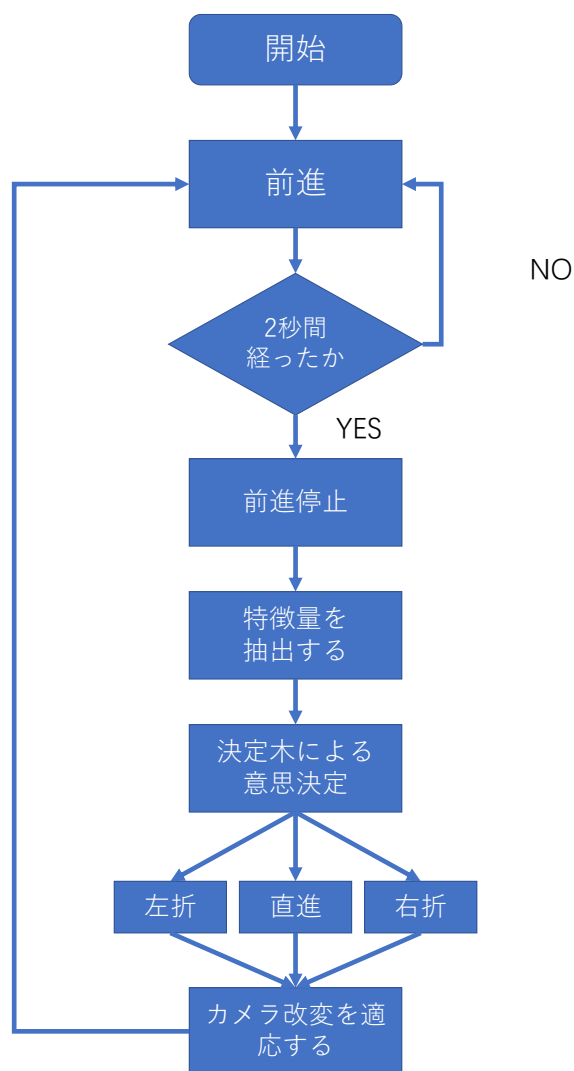


図 2.6 ワールドへの適応

2.5 評価方法

本研究では、いくつかのハイパラメーター (3.5.2 節を参照) と特徴量の組み合わせを変え、異なる決定木のモデルの性能を試みる。実験データをワールド別で分けて k -分割交差検証法を用いて全モデルの平均精度を比較し、最も性能が良いモデルを選択する。これにより、最も人間らしいカメラワークの予測ができるモデルを取得することができる。また、このモデルに対して、訓練データとテストデータにおける精度検査を行い、その人間らしさを評価する。

第 3 章

実装

3.1 前処理

提案手法における前処理には以下の**アルゴリズム 1** を用いて実装している．巻末の付録.A にて，付録としてこのアルゴリズムを実装したソースコードを添付する．

3.2 決定木モデルの構築と交差検証

提案手法における決定木モデルの構築は以下の**アルゴリズム 2** で実装している．巻末の付録.B にて，付録としてこのアルゴリズムを実装したソースコードを添付する．

3.3 精度検査

提案手法における決定木モデルの構築は以下の**アルゴリズム 3** で実装している．巻末の付録.C にて，付録としてこのアルゴリズムを実装したソースコードを添付する．

3.4 ワールドへの適応

提案手法におけるワールドへの適応は以下の**アルゴリズム 4** で実装している．巻末の付録.D にて，付録としてこのアルゴリズムを実装したソースコードを添付する．

Algorithm 1 前処理

```

1: repeat
2:   repeat
3:     カメラ前進
4:     ゲーム画面  $P_{t-1}$  を取得する
5:   until (2 秒間経過)
6:   カメラ停止
7:   ゲーム画面  $P_t$  を取得する
8:   横辺に沿って ( $P_{t-1}$  を  $P_{left_{t-1}}, P_{center_{t-1}}, P_{right_{t-1}}$ ) に平均 3 分割する
9:   横辺に沿って ( $P_t$  を  $P_{left_t}, P_{center_t}, P_{right_t}$ ) に平均 3 分割する
10:  ( $P_{left_t}, P_{center_t}, P_{right_t}$ ) から ( $H_{left}, H_{center}, H_{right}$ ) を計算する
11:  ( $P_{left_{t-1}}, P_{left_t}$ ) から  $S_{left}$  を計算する
12:  ( $P_{center_{t-1}}, P_{center_t}$ ) から  $S_{center}$  を計算する
13:  ( $P_{right_{t-1}}, P_{right_t}$ ) から  $S_{right}$  を計算する
14:   $H \leftarrow (H_{left}, H_{center}, H_{right})$ 
15:   $S \leftarrow (S_{left}, S_{center}, S_{right})$ 
16:  判断ラベル (左折、右折、直進) が入力される
17:  ( $H, S$ , 判断ラベル) を保存する
18:  判断ラベルをカメラに適応 (左折は左に 30 度, 右折は右に 30 度, 直進は角度不変)
19: until (実験終了)

```

Algorithm 2 決定木モデルの構築と交差検証

```

1: 保存している ( $H, S$ , 判断ラベル) を読み取る
2: repeat
3:   決定木に ( $H, S$ , 判断ラベル) とハイパラメーターを入力し, モデルを作成する
4:    $k$ -交差検証法で平均精度を計算する ( $k = 9$ )
5:   output 平均精度
6: until すべてのモデルの構築が完了する
7: 平均精度が最も高いモデルのハイパラメーターを保存する

```

Algorithm 3 精度検査

- 1: 保存している $(H, S, \text{判断ラベル})$ を読み取る
 - 2: 保存した平均精度が最も高いモデルのハイパラメーターを読み取る
 - 3: 決定木に $(H, S, \text{判断ラベル})$ とハイパラメーターを入力し, モデルを作成する
 - 4: (H, S) をモデルに入力し, 判断ラベルを予測する
 - 5: 精度を計算する
 - 6: output 精度
-

Algorithm 4 ワールドへの適応

- 1: **repeat**
 - 2: **repeat**
 - 3: カメラ前進
 - 4: ゲーム画面 P_{t-1} を取得する
 - 5: **until** (2 秒間経過)
 - 6: カメラ停止
 - 7: ゲーム画面 P_t を取得する
 - 8: 横辺に沿って $(P_{t-1}$ を $P_{left_{t-1}}, P_{center_{t-1}}, P_{right_{t-1}}$) に平均 3 分割する
 - 9: 横辺に沿って $(P_t$ を $P_{left_t}, P_{center_t}, P_{right_t}$) に平均 3 分割する
 - 10: $(P_{left_t}, P_{center_t}, P_{right_t})$ から $(H_{left}, H_{center}, H_{right})$ を計算する
 - 11: $(P_{left_{t-1}}, P_{left_t})$ から S_{left} を計算する
 - 12: $(P_{center_{t-1}}, P_{center_t})$ から S_{center} を計算する
 - 13: $(P_{right_{t-1}}, P_{right_t})$ から S_{right} を計算する
 - 14: $H \leftarrow (H_{left}, H_{center}, H_{right})$
 - 15: $S \leftarrow (S_{left}, S_{center}, S_{right})$
 - 16: 決定木に (H, S) を入力し, ラベルを予測する
 - 17: 判断ラベルをカメラに適応 (左折は左に 30 度, 右折は右に 30 度, 直進は角度不変)
 - 18: **until** (実験終了)
-

3.5 ハイパーパラメーター

オプティカルフローの計算と決定木の構築に関して、複数のハイパーパラメーターを設定することができる。実験環境の一致性を保証するため、本項では本研究で用いるハイパーパラメーターについて説明する。

3.5.1 オプティカルフロー

オプティカルフローの計算は Python3.5 のプログラミング環境における opencv で実装した calcOpticalFlowFarneback 関数で行う。本研究でこの関数のハイパーパラメーターを表 3.1 に示す。

表 3.1 本研究で用いる calcOpticalFlowFarneback 関数のハイパーパラメーター

flow	pyr_scale	levels	winsize	iterations	poly_n	poly_sigma	flags
None	0.5	3	15	3	5	1.2	0

3.5.2 決定木

決定木の構築について、本研究では Python3.5 のプログラミング環境における Scikit-learn[7] のアルゴリズムを用いる。構築に関するハイパーパラメーターを表 3.2 に示す。比較に使用するため、分割指標である criterion と木の最大深さ制限である max_depth の値を変数として管理する。

表 3.2 本研究で用いる決定木の構築のハイパーパラメーター

criterion	max_depth	class_weight	random_state
gini/entropy	1~100	balanced	1

第 4 章

実験

本章では，第 2 章に提案した前処理と決定木モデルの構築を行い，自動撮影システムの有用性を確認する．

4.1 実験環境

本研究では，以下の環境で実験を行った．

- OS：Windows 10 professional
- プログラミング環境：Python 3.5
- ゲーム本体：Minecraft 1.12.2 Java Edition (Resolution 720 × 480)
- Minecraft ワールド編集ツール：[Mcedit GDMC Edition](#)
- リモートソフトウェア：[Teamviewer](#)

4.2 ワールドに関する初期条件

訓練データの取得環境はゲームの中にあるため，訓練データの取得結果はゲームの環境設定に影響される．実験の再現性を保証するため，ゲーム環境を制御するパラメータを一定値に維持しなければならない．その中には，異なるワールドごとに異なる動的初期条件と，ワールドに左右されない静的初期条件がある．本節では，設定した静的初期条件と動的初期条件について説明する．

4.2.1 静的初期条件

本研究では，ゲーム内での環境変化 (天気、時間など) を制御するパラメータを静的初期条件と定義する．静的初期条件の詳しい設定値とその説明を表 4.1 に示す．アクセス

は、開発者がパラメータを設置する場所を示す。本研究では、パラメータのアクセスには「NBT ファイル」と「ゲーム設定」の 2 種類を用いる。「NBT ファイル」とは Minecraft ワールドに保存されている設定ファイルの 1 種であり、「Mcedit」[8] というソフトウェア経由で編集する必要がある。MCedit 内でフィルターを書くことでワールドに対する編集ができる。巻末の付録.E にて、付録としてこのフィルターのソースコードを添付する。もう 1 つの「ゲーム設定」は、ゲームを起動し主画面から「options」ボタンを選択して編集することができるものである。

表 4.1 静的初期条件

条件名	設定値	アクセス	説明
DayTime	6000	NBT ファイル	ゲーム世界内の時刻,6000 は 12:00 am
clearWeatherTime	1000	NBT ファイル	天気は晴れ
raining	0	NBT ファイル	雨降らない
thundering	0	NBT ファイル	雷を消す
doDaylightCycle	false	NBT ファイル	時間停止
doWeatherCycle	false	NBT ファイル	天気の切り替えを停止
Flyspeed	0.05	NBT ファイル	カメラの運動スピード
Flying	1	NBT ファイル	空中に浮かべるか 1 は浮かべる
Fov	70	ゲーム設定	Fov 設定, デフォルトは 70
Render Distance	20	ゲーム設定	レンダリング距離
Clouds	off	ゲーム設定	曇をオフにする
the Heads-Up Display	off	ゲーム設定	ワールド内で F1 を押してオフする

4.2.2 動的初期条件

表 4.2 が示したように、動的初期条件には Rotation と Pos がある。Rotation はカメラの姿勢を制御するものである。 $y - rot$ で水平方向の回転を設定する。 -180.0 , -90.0 , 0.0 と 90.0 はそれぞれ北向き, 東向き, 南向きと西向き方向の設定値である。 $x - rot$ で垂直方向の回転を指定する。 -90.0 は真上, 90.0 は真下の設定値である。Pos はカメラが 3 次元空間における座標値である。本実験では、動的初期条件を設定するアルゴリズムを提案した。4.2.3 節でそれについて紹介する。

表 4.2 動的初期条件

条件名	設定値	設定場所	説明
Rotation	$(y - rot, x - rot)$	NBT ファイル	カメラの初期ローテーション
Pos	(x, y, z)	NBT ファイル	カメラの初期位置

4.2.3 動的初期条件の設定方法

集落の撮影を行うため、動的初期条件に属する Rotation と Pos は以下の設定方法で決定し、カメラの初期方向を常に集落に向けるようにする。

Rotation の設定方法

ワールドの上から真っ直ぐ見下ろし、3次元的な空間は図 4.1 が示したのように2次元画像となる。このようにして、ワールドを2次元画像として扱うことで、1番上にあるブロックは2次元画像のピクセルと見なされる。また Minecraft 内でのブロックはタイプによって独自のタイプ ID が付き、図 4.2 と図 4.3 が示すように、ブロックのタイプ ID の値が大きいほど人造物に属する傾向が大きくなる。そのため、ピクセルの値として、図 4.4 と図 4.5 が示したように、タイプ ID を統計したヒストグラムから見ると、集落がある場合とない場合との区別ができる。これにより、1つのエリア内のタイプ ID の総和をこのエリアの重みにすることで、エリア内の集落密度が評価できると考えられる。また、ワールドにおいて Rotation を計算するには、図 4.6 で示すように、まず、原点から4方向300ブロックまでのワールド (600×600 サイズがあるエリア) を選択する。次に、 60×60 のサイズのスライス100個に分け、各エリアの重みを計算する。アルゴリズム 5 を参照し、Rotation にある $y - rot$ は重心に向ける角度を設定する。最後に垂直方向の回転を水平平行にするため、 $x - rot$ の値は0にしておく。以上で Rotation の設定が完了する。巻末の付録.Eにて、付録としてこのアルゴリズムを実現したソースコードを添付する。



図 4.1 ワールドを上空から見るイメージ

































1		Stone (minecraft:stone)	6:4		Acacia Sapling (minecraft:sapling)
1:1		Granite (minecraft:stone)	6:5		Dark Oak Sapling (minecraft:sapling)
1:2		Polished Granite (minecraft:stone)	7		Bedrock (minecraft:bedrock)
1:3		Diorite (minecraft:stone)	8		Flowing Water (minecraft:flowing_water)
1:4		Polished Diorite (minecraft:stone)	9		Still Water (minecraft:water)
1:5		Andesite (minecraft:stone)	10		Flowing Lava (minecraft:flowing_lava)
1:6		Polished Andesite (minecraft:stone)	11		Still Lava (minecraft:lava)
2		Grass (minecraft:grass)	12		Sand (minecraft:sand)
3		Dirt (minecraft:dirt)	12:1		Red Sand (minecraft:sand)
3:1		Coarse Dirt (minecraft:dirt)	13		Gravel (minecraft:gravel)
3:2		Podzol (minecraft:dirt)	14		Gold Ore (minecraft:gold_ore)
4		Cobblestone (minecraft:cobblestone)	15		Iron Ore (minecraft:iron_ore)
5		Oak Wood Plank (minecraft:planks)	16		Coal Ore (minecraft:coal_ore)
5:1		Spruce Wood Plank (minecraft:planks)	17		Oak Wood (minecraft:log)
5:2		Birch Wood Plank (minecraft:planks)	17:1		Spruce Wood (minecraft:log)
5:3		Jungle Wood Plank (minecraft:planks)	17:2		Birch Wood (minecraft:log)

図 4.2 Minecraft におけるブロックの独自タイプ ID(20 以内)[9] より引用

100		Red Mushroom Block (minecraft:red_mushroom_block)	184		Birch Fence Gate (minecraft:birch_fence_gate)
101		Iron Bars (minecraft:iron_bars)	185		Jungle Fence Gate (minecraft:jungle_fence_gate)
102		Glass Pane (minecraft:glass_pane)	186		Dark Oak Fence Gate (minecraft:dark_oak_fence_gate)
103		Melon Block (minecraft:melon_block)	187		Acacia Fence Gate (minecraft:acacia_fence_gate)
104		Pumpkin Stem (minecraft:pumpkin_stem)	188		Spruce Fence (minecraft:spruce_fence)
105		Melon Stem (minecraft:melon_stem)	189		Birch Fence (minecraft:birch_fence)
106		Vines (minecraft:vine)	190		Jungle Fence (minecraft:jungle_fence)
107		Oak Fence Gate (minecraft:fence_gate)	191		Dark Oak Fence (minecraft:dark_oak_fence)
108		Brick Stairs (minecraft:brick_stairs)	192		Acacia Fence (minecraft:acacia_fence)
109		Stone Brick Stairs (minecraft:stone_brick_stairs)	193		Spruce Door Block (minecraft:spruce_door)
110		Mycelium (minecraft:mycelium)	194		Birch Door Block (minecraft:birch_door)
111		Lily Pad (minecraft:waterlily)	195		Jungle Door Block (minecraft:jungle_door)
112		Nether Brick (minecraft:nether_brick)	196		Acacia Door Block (minecraft:acacia_door)
113		Nether Brick Fence (minecraft:nether_brick_fence)	197		Dark Oak Door Block (minecraft:dark_oak_door)
114		Nether Brick Stairs (minecraft:nether_brick_stairs)	198		End Rod (minecraft:end_rod)
115		Nether Wart (minecraft:nether_wart)	199		Chorus Plant (minecraft:chorus_plant)
116		Enchantment Table (minecraft:enchanting_table)	200		Chorus Flower (minecraft:chorus_flower)

図 4.3 Minecraft におけるブロックの独自タイプ ID(100-200 以内)[9] より引用

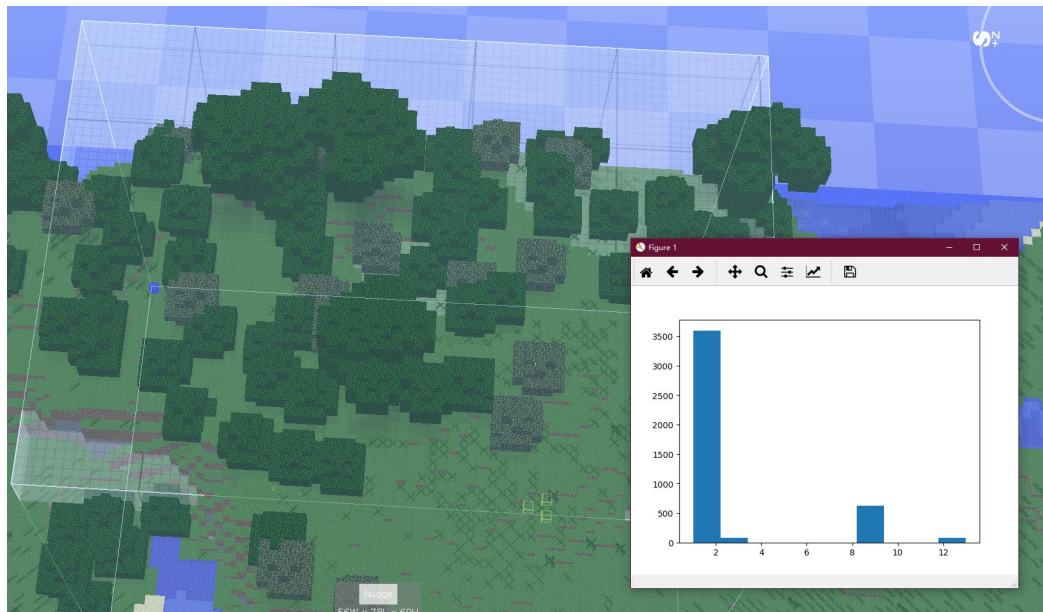


図 4.4 上空から見る集落がないときのブロックの種類分布

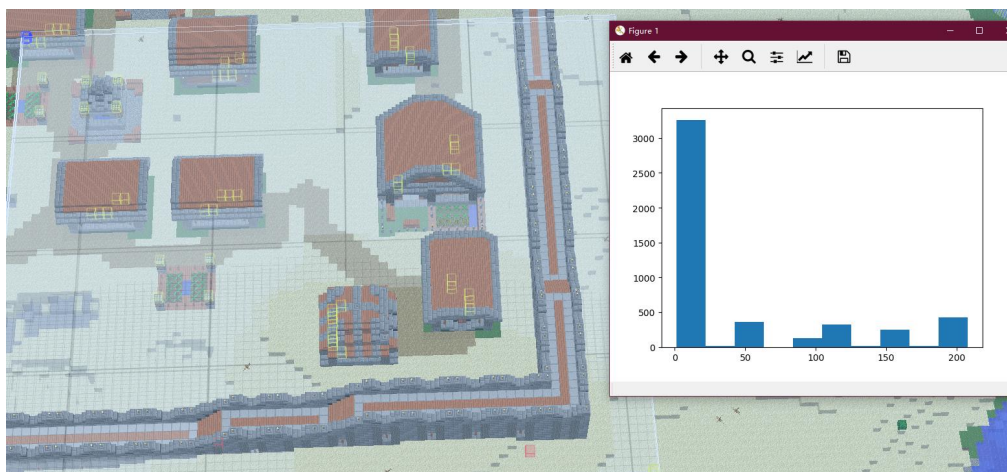


図 4.5 上空から見る集落があるときのブロックの種類分布

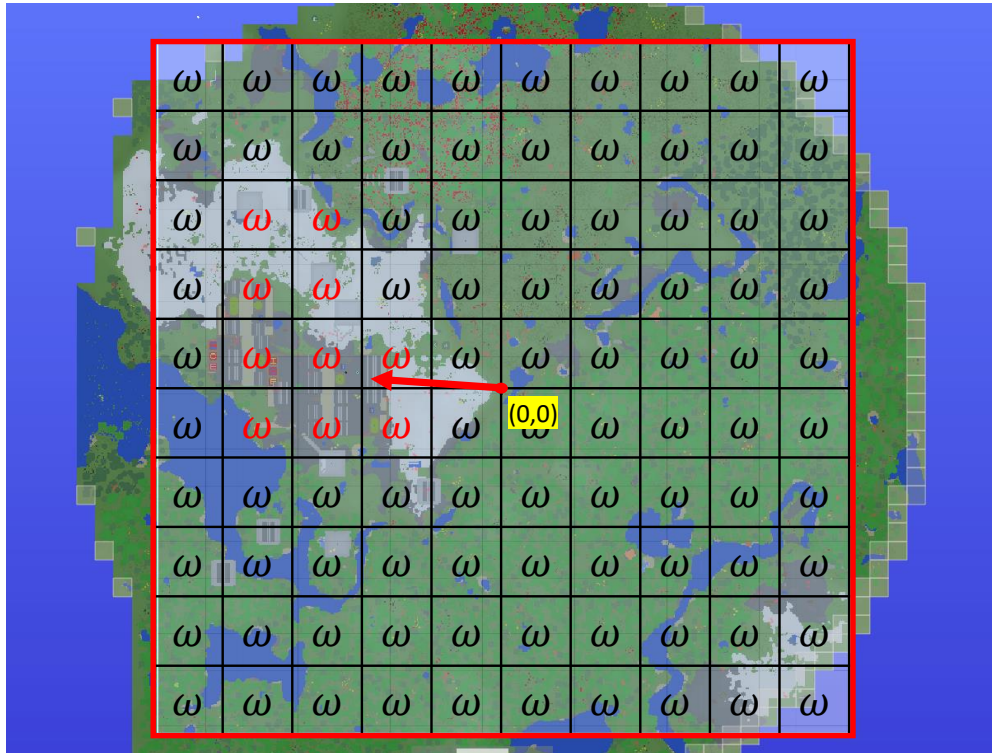


図 4.6 例：重み付き表記とカメラの初期方向

Algorithm 5 Rotation の計算

-
- 1: 重みマップ \mathbf{M} (size:10x10) を生成する
 - 2: $\sum \mathbf{M} = 1$ で \mathbf{M} の値を標準化する
 - 3: 重心座標 $(w_i, w_j) \leftarrow (0, 0)$
 - 4: **for** $i \leftarrow 0; i < 10; i++$ **do**
 - 5: **for** $j \leftarrow 0; j < 10; j++$ **do**
 - 6: $(w_i, w_j) \leftarrow (w_i, w_j) + (i, j) * \mathbf{M}(i, j)$
 - 7: **end for**
 - 8: **end for**
 - 9: $y_rot \leftarrow \text{math.atan2}(-w_i, w_j) * (180/\pi)$
 - 10: **if** $y_rot < 0$ **then**
 - 11: $y_rot \leftarrow y_rot + 360$
 - 12: **end if**
 - 13: $\text{Rotation} \leftarrow (y_rot, 0)$
 - 14: **return** Rotation
-

Pos の設定方法

カメラの3次元空間における座標値 Pos は，ワールドの原点となる ($x = 0, y = 0, z = 0$) に設定しておく．また高さの指標となる y は，0である場合に設定するとカメラは地面の下に埋められてしまうため，高度修正が必要である．Rotation の設定過程に，重心が計算されるため，原点から重心までのルートにあるブロックの中で最も高いブロックの高度より 10 ブロック上げて y を設定する．

4.3 実験で使用するワールド

本実験で用いるワールドは，2020 年度の GDMC 大会に投稿されたワールドの中から選択した．2020 年度 GDMC 大会では内容が異なる 3 つのワールドを配布され，全部の参加選手によってこの 3 つのワールド中で集落の生成が行われた．本実験ではビジュアルの観点に考慮した結果，大会の得点が前三位である参加選手の提出ワールドを選んで総計 9 つのワールドを実験ワールドにした．この 9 つの実験ワールドの景観を図 4.7 に示す．

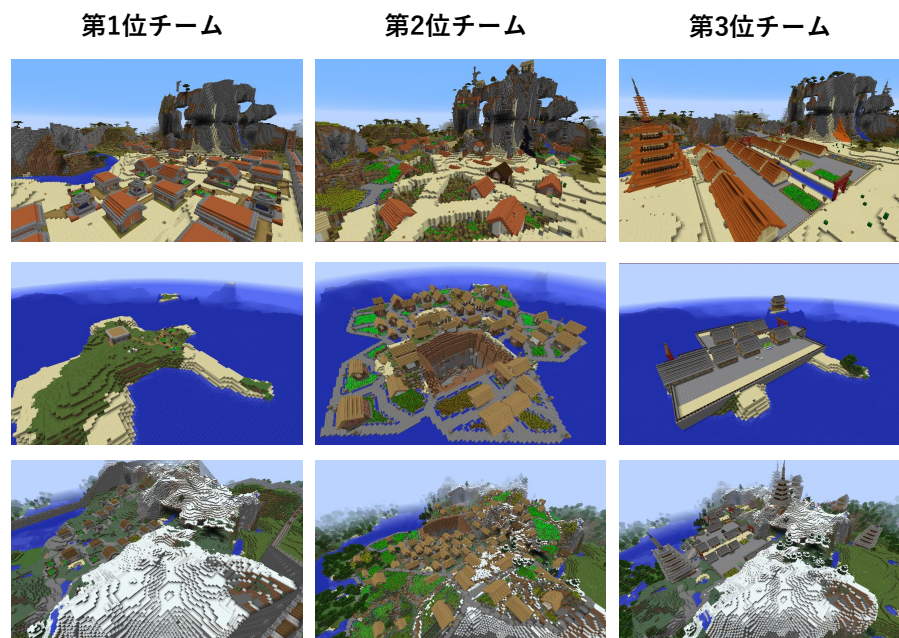


図 4.7 9 つの実験ワールド

4.4 訓練データの取得

2.2.1 節で紹介した手法のように、本実験で以下の手順で集落の動画を撮影することで、訓練データを取得した。

- step 1 ワールドの初期条件を設定し、カメラを初期位置で待機させる。
- step 2 カメラが前進を始める。
- step 3 2 秒間経過すると停止し、被験者にカメラアクション（左折、右折、直行）を決定してもらう。
- step 4 特徴量とカメラアクションをペアで保存する。
- step 5 step 2 と 3 を 50 回繰り返す。

9 ワールド上で同じ操作を繰り返し、取得した全部のデータ数は計 450 個となる。

4.5 実験結果

4.5.1 モデルの比較

本実験でハイパーパラメーターの組み合わせを改変し、いくつかのモデルを比較した。比較に使われたハイパーパラメーターは以下の通りである。

- 決定木分割する際に使う分割指標： エントロピーとジニ不純度
- 木の最大深さ制限（1 から - 100 まで）
- 特徴量の組み合わせ：
 - － カラーエントロピー
 - － Optical Flow 強度
 - － 両方を使う

図 4.8 で示した方法の通りに、 k -分割交差検証法でモデルの平均精度を計算した。その結果を図 4.9 に示す。縦軸は k -分割交差検証法で計算した平均精度であり、横軸は決定木の最大深さ制限である。

Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9
Map 1	Map 1	Map 1	Map 1	Map 1	Map 1	Map 1	Map 1	Map 1
Map 2	Map 2	Map 2	Map 2	Map 2	Map 2	Map 2	Map 2	Map 2
Map 3	Map 3	Map 3	Map 3	Map 3	Map 3	Map 3	Map 3	Map 3
Map 4	Map 4	Map 4	Map 4	Map 4	Map 4	Map 4	Map 4	Map 4
Map 5	Map 5	Map 5	Map 5	Map 5	Map 5	Map 5	Map 5	Map 5
Map 6	Map 6	Map 6	Map 6	Map 6	Map 6	Map 6	Map 6	Map 6
Map 7	Map 7	Map 7	Map 7	Map 7	Map 7	Map 7	Map 7	Map 7
Map 8	Map 8	Map 8	Map 8	Map 8	Map 8	Map 8	Map 8	Map 8
Map 9	Map 9	Map 9	Map 9	Map 9	Map 9	Map 9	Map 9	Map 9

	Test Data
	Training Data

図 4.8 本実験における k-分割交差検証法

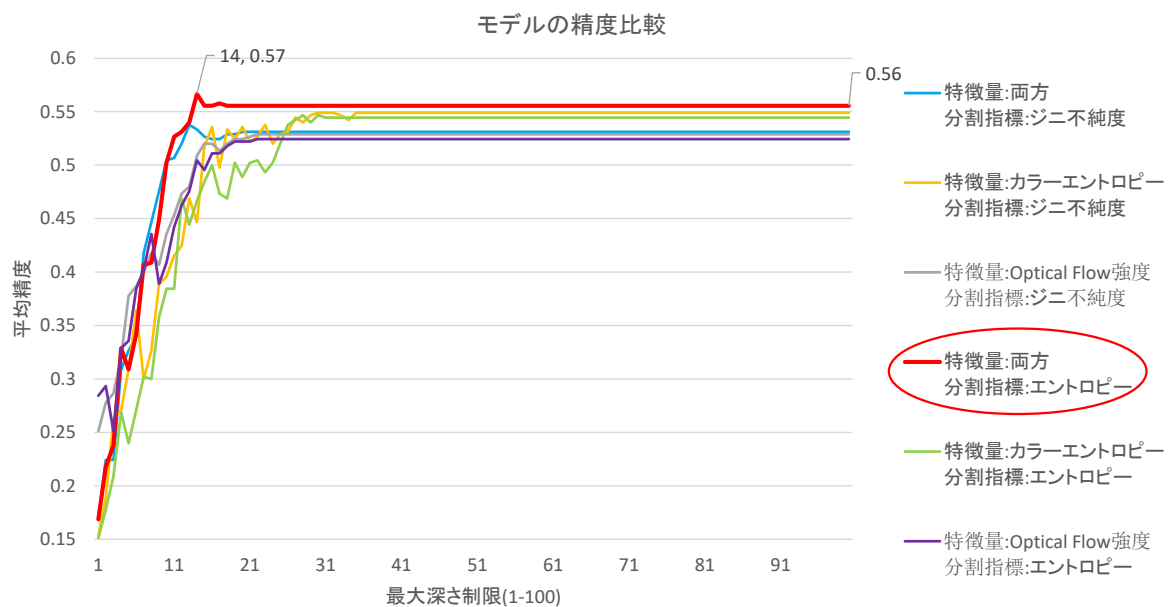


図 4.9 モデルの精度比較

4.5.2 モデルにおける精度検査

図 4.9 から分かるように、モデル (特微量:両方, 分割指標:エントロピー, 木の最大深さ制限:14) の平均精度が最も高いため、このモデルにおける精度検査を行った。精度検査の結果を表 4.3 に示した。訓練データ (データ数:450, Map1-9) に対して予測精度 (有効数字 2 桁, 切り捨て) は 0.99 であり、テストデータ (データ数:50, Map9) に対して、Map1-8 のデータで訓練した予測精度は 0.70 である。

表 4.3 精度検査の結果

	訓練に使われる データセット	予測に使われる データセット	予測精度 (有効数字 2 桁)
訓練データ	Map1-9(450 個)	Map1-9(450 個)	0.99
テストデータ	Map1-8(400)	Map9(50)	0.70

4.6 考察

図 4.9 によると、最も性能が良いモデルは「特微量:両方 分割指標:エントロピー 木の最大深さ制限:14」であり、 k -分割交差検証法で計算した平均精度は 0.57 であった。これは、特微量が多いほど、決定木が分類しやすいと考えられる。また、今回は木の最大深さ制限は 14 に超えると、過学習による精度の下降が発生した。決定木学習を使用するとき、木の最大深さ制限がないと、過学習が起こりうるため、最も精度の高い木の最大深さを探る必要があると考えられる。そして、このモデルを使って精度検査をした結果、「表 4.3」に示したように、訓練データに対する予測精度が 0.99、テストデータに対する予測精度が 0.70 という結果が得られた。また、木の最大深さ制限を無制限に設定すると、訓練データに対して予測精度は 1.0 になる。この結果から、既に学習に使われたワールドに対して、人間らしい自動撮影を実現できることを示した。しかしながら、未知のワールドに対しては、予測精度の下降によって、現段階での人間らしい自動撮影は不可能と考えられる。

第 5 章

おわりに

5.1 まとめ

本研究では，決定木による意思決定によって人間らしいカメラワークを自動生成できるシステムの構築を目指した．具体的に言うと，決定木予測に基づき，人間が Minecraft の集落を撮影しているときの動きを学習することで，人間らしいカメラワークを自動生成できるシステムの構築を目標とした．

提案手法として，ゲーム画面よりカラーエントロピーとオプティカルフローを導き出し，決定木方式でその 2 つの特徴量を用いて学習モデルを構築した．その後，システムを実装し，この 2 つの特徴量でカメラワークの予測を行い，MineCraft ワールドへ適用し，カメラワークの自動生成を実現した．

本システムの性能検証を行うため，実験を行った．その結果，訓練データにおける精度が最大値の 1.0 であることより，本システムにおいて訓練データに対する有用性が確認できた．またテストデータに対しては，7 割程度の正確な予測ができることが確認できた．

今後は，このシステムをもう一レベルを上げるのを望んでいる．

5.2 今後の課題

今回の実験で提案した手法では，未知のワールドでの有用性を確認することができなかった．そのため，今後は枝刈りなどの手法を用いることで決定木の予測精度を上げることを目指す．また，今回得られた特徴を基に，ゲームのワールド内の 3 次元情報も取得し，障害物回避できるようにすることも今後の課題とする．

謝辞

本研究を行うにあたり、熱心な指導と非常に多くの助言をして頂き、またゼミ等で話し会える場を提供して頂いた、THAWONMAS Ruck 教授に心から感謝いたします。講義だけではなく学部運営に関わる業務等で多忙である中、ゼミ時間外にも個別に時間を設けていただき、ご指導と励ましをいただきました。

また、知能エンターテインメント研究室所属の Paliyawan Pujana 先輩と伊藤聡子さんに感謝いたします。ゼミ時間中やそれ以外の場でも研究に関する様々なアドバイスを頂き、有意義な研究議論ができました。同研究室の諸氏にも感謝します。皆様には研究に関するアドバイス等を頂き、大きな支えとなりました。

皆様のご協力のおかげで、学部生として研究に取り組むことができました。研究活動に費やせた成果を持って、将来大学院でより一層努力します。

以上を持って謝辞とさせていただきます。

参考文献

- [1] Salge, C., Green, M.C., Canaan, R. et al. The AI Settlement Generation Challenge in Minecraft. *Künstl Intell*, Vol.34, 19-31 (2020)
- [2] 秋山孝正, 奥嶋政嗣. 交通機関選択分析のためのファジィ決定木手法の比較検討. **土木学会論文集D**, 63 巻, 2 号, 145-157 (2007)
- [3] 春野雅彦, 白井諭, 大山芳史. 決定木を用いた日本語係受け解析. **情報処理学会論文誌**, 39 巻, 12 号, 3177-3186 (1998)
- [4] Amanjot Kaur, Sukhwinder Bir and Harjasdeep Singh. Image Segmentation Using Entropy: A Review. *International Journal of Emerging Science and Engineering*, Vol.2, 7-9 (2013)
- [5] Hagen Echzell, Tobias Friedrich, Pascal Lenzner, Anna Melnichenko. Flow-based Intrinsic Curiosity Module. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Main track. 139-145 (2020)
- [6] G Farneback, Two-Frame Motion Estimation Based on Polynomial Expansion, (2003)
- [7] Scikit-Learn, [online] Available: <https://scikit-learn.org/stable/>, Jan (2021)
- [8] Mccredit GDMC Edition, [online] Available: <https://github.com/mcgreentn/GDMC>, Jan (2021)
- [9] Minecraft ID List, [online] Available: <https://minecraft->

ids.grahamedgecombe.com/, Jan (2021)

付録

付録.A PreprocessingAndDataCollection.py

```
import numpy as np
import cv2
import mss
import threading
import time
import entropy
import mouse
import keyboard
import win32gui
import dbm
from sklearn import preprocessing

hwnd = win32gui.FindWindow(None, r'Minecraft_1.12.2')
win32gui.SetForegroundWindow(hwnd)
dimensions = win32gui.GetWindowRect(hwnd)
print(dimensions)
w = 720
h = 480
exitFlag = 0
prvs_frame = None
next_frame = None
frames = None
```

```

flows = None
flag = False
p_flag = False
Monitor_flag = False
monitor = { 'top': dimensions[1]+31,
            'left': dimensions[0]+8,
            'width': w, 'height': h}

count = 0
map_name = input("name: ")

def get_frame():
    with mss.mss() as sct:
        frame = cv2.cvtColor(np.array(sct.grab(monitor)),
                              cv2.COLOR_BGRA2BGR)

    return frame

def get_flows(prvs, next):
    hsv = np.zeros_like(prvs)
    hsv[...] , 1] = 255
    prvs = cv2.cvtColor(prvs, cv2.COLOR_BGR2GRAY)
    next = cv2.cvtColor(next, cv2.COLOR_BGR2GRAY)
    flow = cv2.calcOpticalFlowFarneback(prvs, next,
                                         None, 0.5, 3,
                                         15, 3, 5, 1.2, 0)
    mag, ang = cv2.cartToPolar(flow[...] , 0], flow[...] , 1])
    hsv[...] , 0] = ang * 180 / np.pi / 2
    hsv[...] , 2] = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)
    bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
    f = [mag[:, 0:w // 3],
         mag[:, w // 3:2 * w // 3],
         mag[:, 2 * w // 3:w]]
    return f

```

[illegible]

```
class ScreenRecord(threading.Thread):
    def __init__(self, thread_id, name):
        threading.Thread.__init__(self)
        self.threadID = thread_id
        self.name = name

    def run(self):
        global prvs_frame
        print(" : start" + self.name)
        while 'Screen_capturing':
            time.sleep(0.1)
            if p_flag:
                prvs_frame = get_frame()

def sum_flows(data):
    a = []
    for one in data:
        a.append(np.sum(one))
    return preprocessing.scale(a)

def calculate_entropy(data):
    a = []
    for one in data:
        a.append(calculate_avg_entropy(one))
    return preprocessing.scale(a)

def calculate_avg_entropy(data):
    a = entropy.calcEntropy(data[... , 0])
    b = entropy.calcEntropy(data[... , 1])
```

```

c = entropy.calcEntropy(data [... , 2])
return (a+b+c)/3

def test(command):
    global p_flag, flag, count
    if flag:
        return
    db = dbm.open('count', 'c')
    f0 = open("data/" + map_name + r"_Entropy.txt", 'a+')
    f1 = open("data/" + map_name + r"_Optical_Flow.txt", 'a+')
    f2 = open("data/" + map_name + r"_All.txt", 'a+')
    if command is "0":
        db['n'] = str(count)
        flag = True
        p_flag = True
    elif command is "1":
        for one in calculate_entropy(frames):
            f0.write(str(one) + '_')
            f2.write(str(one) + '_')
        for one in sum_flows(flows):
            f1.write(str(one) + '_')
            f2.write(str(one) + '_')
        f0.write("forward\n")
        f1.write("forward\n")
        f2.write("forward\n")
        count += 1
        db['n'] = str(count)
        flag = True
        p_flag = True
    elif command is "2":
        for one in calculate_entropy(frames):
            f0.write(str(one) + '_')

```



```
        f2.write(str(one) + ' ')
    for one in sum_flows(flows):
        f1.write(str(one) + ' ')
        f2.write(str(one) + ' ')
    f0.write("right\n")
    f1.write("right\n")
    f2.write("right\n")
    mouse.move(50, 0, absolute=False, duration=0.5)
    count += 1
    db['n'] = str(count)
    flag = True
    p_flag = True
elif command is "4":
    for one in calculate_entropy(frames):
        f0.write(str(one) + ' ')
        f2.write(str(one) + ' ')
    for one in sum_flows(flows):
        f1.write(str(one) + ' ')
        f2.write(str(one) + ' ')
    f0.write("left\n")
    f1.write("left\n")
    f2.write("left\n")
    mouse.move(-50, 0, absolute=False, duration=0.5)
    count += 1
    db['n'] = str(count)
    flag = True
    p_flag = True
f0.close()
f1.close()
f2.close()
db.close()
print(count)
```

```
class Control(threading.Thread):  
    def __init__(self, thread_id, name):  
        threading.Thread.__init__(self)  
        self.threadID = thread_id  
        self.name = name  
  
    def run(self):  
        global flag, flows, frames, next_frame, Monitor_flag, p_flag  
        while True:  
            time.sleep(0.1)  
            if flag:  
                db = dbm.open('count', 'c')  
                db['f'] = str('0')  
                db.close()  
                keyboard.press('w')  
                break  
        while True:  
            time.sleep(2)  
            keyboard.release('w')  
            p_flag = False  
            time.sleep(1)  
            next_frame = get_frame()  
            flows = get_flows(prvs_frame, next_frame)  
            frames = [next_frame[:, 0:w // 3],  
                      next_frame[:, w // 3:2 * w // 3],  
                      next_frame[:, 2 * w // 3:w]]  
            Monitor_flag = True  
            flag = False  
            db = dbm.open('count', 'c')  
            db['f'] = str('1')  
            db.close()  
            while True:  
                if flag:
```

```
        db = dbm.open( 'count', 'c' )
        db[ 'f' ] = str( '0' )
        db.close()
        keyboard.press( 'w' )
        break
    time.sleep(0.1)

keyboard.add_hotkey( 'enter', test, args="0" )
keyboard.add_hotkey( 'up', test, args="1" )
keyboard.add_hotkey( 'right', test, args="2" )
keyboard.add_hotkey( 'left', test, args="4" )

sr = ScreenRecord(0, 'sr')
sr.start()
cl = Control(1, 'cl')
cl.start()
# m = Monitor(2, 'm')
# m.start()
```

付録.B CrossValidation.py

```
from sklearn import tree
import pandas as pd
import numpy as np

path = "D:/OpticalFlow/data.xls"
data = pd.read_excel(path)
data.replace('forward', 0, inplace=True)
data.replace('left', 1, inplace=True)
data.replace('right', 2, inplace=True)

a = {'0': 0, '1': 0, '2': 0,}
for i in range(0, 450):
    a[str(data.action.values[i])] += 1
print(a)

# c_type = 'entropy'
c_type = 'gini'

for dp in range(1, 101):
    scores = []
    for i in range(9):
        test_data = data.iloc[i*50: (i+1)*50, :]
        training_data = pd.concat([data.iloc[0: i*50, :],
                                   data.iloc[(i+1)*50: 450, :]])
        training_data_set_H = []
        for j in range(400):
            training_data_set_H.append(
                [training_data.H_left.values[j],
                 training_data.H_center.values[j],
```

```
        training_data.H_right.values[j]))
training_data_set_S = []
for j in range(400):
    training_data_set_S.append(
        [training_data.S_left.values[j],
         training_data.S_center.values[j],
         training_data.S_right.values[j]])
training_data_set_All = []
for j in range(400):
    training_data_set_All.append(
        [training_data.H_left.values[j],
         training_data.H_center.values[j],
         training_data.H_right.values[j],
         training_data.S_left.values[j],
         training_data.S_center.values[j],
         training_data.S_right.values[j]])

test_data_set_H = []
for j in range(50):
    test_data_set_H.append(
        [test_data.H_left.values[j],
         test_data.H_center.values[j],
         test_data.H_right.values[j]])
test_data_set_S = []
for j in range(50):
    test_data_set_S.append([test_data.S_left.values[j],
                             test_data.S_center.values[j],
                             test_data.S_right.values[j]])
test_data_set_All = []
for j in range(50):
    test_data_set_All.append(
        [test_data.H_left.values[j],
         test_data.H_center.values[j],
```

```
test_data.H_right.values[j],
test_data.S_left.values[j],
test_data.S_center.values[j],
test_data.S_right.values[j]])

clf = tree.DecisionTreeClassifier(criterion=c_type,
                                  max_depth=dp,
                                  class_weight='balanced',
                                  random_state=1)

clf = clf.fit(training_data_set_S, training_data.action)
predicted = clf.predict(test_data_set_S)
# print(predicted)
accuracy = sum(predicted == test_data.action) \
           / len(test_data.action)
# print(accuracy)
scores.append(accuracy)
scores = np.array(scores)
print(scores.mean())
```

[illegible]

```
                                random_state=1,
                                max_depth=14)
clf = clf.fit(training_data_set_All[0:450],
              data.action.values[0:450])
predicted = clf.predict(training_data_set_All[0:450])
# print(predicted)
accuracy = sum(predicted == data.action[0:450]) / 450
print(accuracy)
scores.append(accuracy)
scores = np.array(scores)
print(scores.mean())
```


付録.D replayPredict.py

```
import numpy as np
import cv2
import mss
import threading
import time
import entropy
import mouse
import keyboard
import win32gui
import dbm

from sklearn import preprocessing
from sklearn import tree
from sklearn.model_selection import cross_val_score
import pandas as pd
import numpy as np

path = "D:/OpticalFlow/data.xls"
df_list = []
for i in range(9):
    df_list.append(pd.read_excel(path, sheet_name=i))
data = pd.concat(df_list)
a = {'forward': 0, 'left': 0, 'right': 0,}
for i in range(0, 450):
    a[data.action.values[i]] += 1
print(a)

training_data_set_All = []
for j in range(0, 450):
    training_data_set_All.append(
```

```

        [data.H_left.values[j],
         data.H_center.values[j],
         data.H_right.values[j],
         data.S_left.values[j],
         data.S_center.values[j],
         data.S_right.values[j]])
clf = tree.DecisionTreeClassifier(criterion='entropy',
                                  class_weight='balanced',
                                  random_state=1,
                                  max_depth=100)
clf.fit(training_data_set_All, data.action.values)

def tree_predict(count):
    predicted = clf.predict([training_data_set_All[50+count]])[0]
    print('predicted:', predicted)
    print('target:', data.action.values[50+count])

    if predicted == 'right':
        test("2")
    elif predicted == 'left':
        test("4")
    else:
        test("1")

hwnd = win32gui.FindWindow(None, r'Minecraft_1.12.2')
win32gui.SetForegroundWindow(hwnd)
dimensions = win32gui.GetWindowRect(hwnd)
print(dimensions)
w = 720
h = 480
# plt.figure(figsize=(8, 6), dpi=80)

```

```

exitFlag = 0
prvs_frame = None
next_frame = None
frames = None
flows = None
flag = False
p_flag = False
Monitor_flag = False
monitor = {'top': dimensions[1]+31,
           'left': dimensions[0]+8,
           'width': w, 'height': h}
count = 0
map_name = input("name:_")

def get_frame():
    with mss.mss() as sct:
        frame = cv2.cvtColor(np.array(sct.grab(monitor)),
                             cv2.COLOR_BGRA2BGR)

    return frame

def get_flows(prvs, next):
    hsv = np.zeros_like(prvs)
    hsv[..., 1] = 255
    prvs = cv2.cvtColor(prvs, cv2.COLOR_BGR2GRAY)
    next = cv2.cvtColor(next, cv2.COLOR_BGR2GRAY)
    flow = cv2.calcOpticalFlowFarneback(prvs, next,
                                         None, 0.5, 3, 15, 3, 5, 1.2, 0)
    mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    hsv[..., 0] = ang * 180 / np.pi / 2
    hsv[..., 2] = cv2.normalize(mag, None, 0, 255,
                                cv2.NORM_MINMAX)

    bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
    f = [mag[:, 0:w // 3],

```

```

        mag[:, w // 3:2 * w // 3],
        mag[:, 2 * w // 3:w]]
    return f

class Monitor(threading.Thread):
    def __init__(self, thread_id, name):
        threading.Thread.__init__(self)
        self.threadID = thread_id
        self.name = name

    def run(self):
        print(":start" + self.name)
        while True:
            if Monitor_flag:
                g = np.zeros_like(frames[0])
                g[..., 0] = cv2.normalize(flows[0],
                                          None, 0, 255,
                                          cv2.NORM_MINMAX)
                g[..., 1] = cv2.normalize(flows[1],
                                          None, 0, 255,
                                          cv2.NORM_MINMAX)
                g[..., 2] = cv2.normalize(flows[2],
                                          None, 0, 255,
                                          cv2.NORM_MINMAX)
                cv2.imshow('flow_v_0', g[..., 0])
                cv2.imshow('flow_v_1', g[..., 1])
                cv2.imshow('flow_v_2', g[..., 2])
                cv2.imshow('frame_v_0', frames[0])
                cv2.imshow('frame_v_1', frames[1])
                cv2.imshow('frame_v_2', frames[2])
                cv2.imshow('prvs_frame', prvs_frame)
                cv2.imshow('next_frame', next_frame)

```

```
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break
    time.sleep(0.1)

class ScreenRecord(threading.Thread):
    def __init__(self, thread_id, name):
        threading.Thread.__init__(self)
        self.threadID = thread_id
        self.name = name

    def run(self):
        global prvs_frame
        print(": start" + self.name)
        while 'Screen_capturing':
            time.sleep(0.1)
            if p_flag:
                prvs_frame = get_frame()

def sum_flows(data):
    a = []
    for one in data:
        a.append(np.sum(one))
    return preprocessing.scale(a)

def calculate_entropy(data):
    a = []
    for one in data:
        a.append(calculate_avg_entropy(one))
    return preprocessing.scale(a)
```

```

def calculate_avg_entropy(data):
    a = entropy.calcEntropy(data[... , 0])
    b = entropy.calcEntropy(data[... , 1])
    c = entropy.calcEntropy(data[... , 2])
    return (a+b+c)/3

def test(command):
    global p_flag, flag, count
    if flag:
        return
    db = dbm.open('count', 'c')
    f0 = open("data/" + map_name + r"_Entropy.txt", 'a+')
    f1 = open("data/" + map_name + r"_Optical_Flow.txt", 'a+')
    f2 = open("data/" + map_name + r"_All.txt", 'a+')
    if command is "0":
        db['n'] = str(count)
        flag = True
        p_flag = True
    elif command is "1":
        for one in calculate_entropy(frames):
            f0.write(str(one) + '_')
            f2.write(str(one) + '_')
        for one in sum_flows(flows):
            f1.write(str(one) + '_')
            f2.write(str(one) + '_')
        f0.write("forward\n")
        f1.write("forward\n")
        f2.write("forward\n")
        count += 1
        db['n'] = str(count)
        flag = True

```

```
p_flag = True
elif command is "2":
    for one in calculate_entropy(frames):
        f0.write(str(one) + '└')
        f2.write(str(one) + '└')
    for one in sum_flows(flows):
        f1.write(str(one) + '└')
        f2.write(str(one) + '└')
    f0.write("right\n")
    f1.write("right\n")
    f2.write("right\n")
    mouse.move(50, 0, absolute=False, duration=0.5)
    count += 1
    db['n'] = str(count)
    flag = True
    p_flag = True
elif command is "4":
    for one in calculate_entropy(frames):
        f0.write(str(one) + '└')
        f2.write(str(one) + '└')
    for one in sum_flows(flows):
        f1.write(str(one) + '└')
        f2.write(str(one) + '└')
    f0.write("left\n")
    f1.write("left\n")
    f2.write("left\n")
    mouse.move(-50, 0, absolute=False, duration=0.5)
    count += 1
    db['n'] = str(count)
    flag = True
    p_flag = True
f0.close()
f1.close()
```

```
f2.close()
db.close()
print(count)

class Control(threading.Thread):
    def __init__(self, thread_id, name):
        threading.Thread.__init__(self)
        self.threadID = thread_id
        self.name = name

    def run(self):
        global flag, flows, frames, \
            next_frame, Monitor_flag, p_flag
        while True:
            time.sleep(0.1)
            if flag:
                db = dbm.open('count', 'c')
                db['f'] = str('0')
                db.close()
                keyboard.press('w')
                break
        while True:
            time.sleep(2)
            keyboard.release('w')
            p_flag = False
            time.sleep(1)
            next_frame = get_frame()
            flows = get_flows(prvs_frame, next_frame)
            frames = [next_frame[:, 0:w // 3],
                      next_frame[:, w // 3:2 * w // 3],
                      next_frame[:, 2 * w // 3:w]]
            Monitor_flag = True
            flag = False
```



```
db = dbm.open('count', 'c')
db['f'] = str('1')
db.close()
while True:
    if flag:
        db = dbm.open('count', 'c')
        db['f'] = str('0')
        db.close()
        keyboard.press('w')
        break
    tree_predict(count)
    time.sleep(0.1)
```

```
keyboard.add_hotkey('enter', test, args="0")
```

```
sr = ScreenRecord(0, 'sr')
sr.start()
cl = Control(1, 'cl')
cl.start()
# m = Monitor(2, 'm')
# m.start()
```

付録.E NBTEdit.py

```
from pymclevel.player import nbt
import HeightMap
import numpy as np
import matplotlib.pyplot as plt
from functions import *
import math

displayName = "NBTEditor"

def normalization(data):
    sum = np.sum(data)
    return data / sum

def get_score(ids):
    dict = {}
    for key in ids:
        dict[key] = dict.get(key, 0) + 1
    score = 0
    for key in dict.keys():
        score += key * dict[key]
    return score

def perform(level, box, options):
    level.root_tag['Data']['DayTime'] = \
        nbt.TAG_Long(6000)
    level.root_tag['Data']['clearWeatherTime'] = \
        nbt.TAG_Int(1000)
```

```
level.root_tag['Data']['raining'] = \
    nbt.TAG_Byte(0)
level.root_tag['Data']['thundering'] = \
    nbt.TAG_Byte(0)
level.root_tag['Data']['GameRules']['doDaylightCycle'] = \
    nbt.TAG_String(u'false')
level.root_tag['Data']['GameRules']['doWeatherCycle'] = \
    nbt.TAG_String(u'false')
level.root_tag['Data']['Player']['abilities']['flying'] = \
    nbt.TAG_Byte(1)
player_pos = [nbt.TAG_Double(0),
              nbt.TAG_Double(128),
              nbt.TAG_Double(0)]
level.root_tag['Data']['Player']['Pos'] = player_pos

min_z = 0
while True:
    min_z -= 1
    y = 0
    for y in range(0, 256):
        if level.blockAt(0, y, min_z) != 0:
            break
    if y == 255:
        min_z += 1
        break

max_z = 0
while True:
    max_z += 1
    y = 0
    for y in range(0, 256):
        if level.blockAt(0, y, max_z) != 0:
            break
```

```
        if y == 255:
            max_z -= 1
            break

min_x = 0
while True:
    min_x -= 1
    y = 0
    for y in range(0, 256):
        if level.blockAt(min_x, y, 0) != 0:
            break
    if y == 255:
        min_x += 1
        break

max_x = 0
while True:
    max_x += 1
    y = 0
    for y in range(0, 256):
        if level.blockAt(max_x, y, 0) != 0:
            break
    if y == 255:
        max_x -= 1
        break

if max_x == 0 or max_z == 0 or min_x == 0 or min_z == 0:
    print ("max_x:", max_x, "min_x:", min_x,
           "max_z", max_z, "min_z", min_z)
# return
```

```

max_x = 300
min_x = -300
max_z = 300
min_z = -300
print ("max_x:", max_x, "min_x:", min_x,
        "max_z", max_z, "min_z", min_z)

w = (max_x-min_x)/10
h = (max_z-min_z)/10
print ("w:", w, "h:", h)
i = 0
score_map = np.zeros((10, 10), dtype=float)
for x in range(min_x, max_x-w, w):
    j = 0
    for z in range(min_z, max_z-h, h):
        print(i, j)
        hm = HeightMap.HeightMap(level,
                                   x, z,
                                   x+w, z+h) # Create height map
        # print hm.height_map
        ids = []
        for sx in range(x, x+w):
            for sz in range(z, z+h):
                ids.append(level.blockAt(sx,
                                           hm.height_map[sx-x][sz-z],
                                           sz))
        score_map[i][j] = get_score(ids)
        j += 1
    i += 1

score_map = normalization(score_map)
mean = np.mean(score_map)
sum_x = 0

```

```

sum_z = 0
for x in range(10):
    for z in range(10):
        # we = score_map[x][z]
        # if we < mean:
        #     we = 0.0
        sum_x += x * score_map[x][z]
        sum_z += z * score_map[x][z]
# print(sum_x, sum_z)
t_x, t_z = int(w * sum_x) + min_x, int(h * sum_z) + min_z
a = math.atan2(-t_x, t_z) * (180 / math.pi)
if a < 0:
    a = 360 + a
print("x:", t_x, "z:", t_z, "_angle:", a)

path_list = []
hm = HeightMap.HeightMap(level, 0, 0, 1, 1)
# print hm.height_map
n_height = hm.height_map[0][0]
n = float(t_x)/t_z
m = float(t_z)/t_x
sign = 1
if t_z < 0:
    sign = -1
for z in range(t_z, sign):
    path_pair = (int(z*n), z)
    if path_pair not in path_list:
        path_list.append(path_pair)
sign = 1
if t_x < 0:
    sign = -1
for x in range(0, t_x, sign):
    path_pair = (x, int(x * m))

```

```
        if path_pair not in path_list:
            path_list.append(path_pair)
# print(path_list)
for one in path_list:
    if level.blockAt(one[0], n_height, one[1]):
        while level.blockAt(one[0], n_height, one[1]) != 0:
            n_height += 1
        n_height += 10
print("height:", n_height)

player_rotation = [nbt.TAG_Float(a), nbt.TAG_Float(0)]
level.root_tag['Data']['Player']['Rotation'] = player_rotation
player_pos = [nbt.TAG_Double(0),
              nbt.TAG_Double(n_height),
              nbt.TAG_Double(0)]
level.root_tag['Data']['Player']['Pos'] = player_pos
```