

チーム紹介、目標、意気込み

チーム紹介

がんちゃん+Oneは昨年度の東北地区大会に出場した、[がんちゃんETドリーム]と[I am one]の連携チームです。
★ がんちゃんETドリーム (成績：DA 最下位 人数：4名)
★ I am one (成績：DA東北地区ゴールドモデル 人数：1名)
昨年度は、[がんちゃん]は技術力不足、[I am one]は人手不足という状況でしたが、お互いの不足点を補いあい挑戦します。

目標

競技とモデルの両方で3位以内を獲得し、CS大会総合優勝

意気込み

地区大会は本番環境に適合できず不本意な結果となりました。
CS大会では本来の力を発揮し、総合優勝を目指します。

モデルの概要

Rコースにおいて-11.1秒のリザルトタイムを獲得することを目標とし、それに必要となる要求を分析しました。最大計測時間内でのクリアを想定し、5箇所のパワースポット設置のみを狙う戦略をとります。本モデルは、時間制約を満たしつつ高得点を獲得することができるルートの探索及びそれを実現する制御に主軸を置いています。

ルートの探索においては、ブロック並べエリアの攻略に必要な動作を細分化し、移動時間が少なくなるようなルートの決定手法を記載しています。加えて、走行手法の切り替えにより状況に応じた制御を検討しており、その手法を記載しています。

また、AIアンサーにおいても同様に、短時間で出題数字を認識可能な手法を記載しています。これらの分析結果・制御戦略により、時間制約を満たしつつ、安定して高得点を獲得できるようになりました。

モデルの構成

2. 要求モデル

-11.1秒のリザルトタイム獲得を目標にルール上の制約、ソフトウェア品質、システムに要求される機能、目標達成のための要件を定義し、これらを検討した結果をSysML要求図に記載しました。要件の細分化によって、競技攻略に必要な要求を可視化しました。

3. 分析モデル

ゲームの要素及び走行体の動作の定義、ボーナスタイム獲得のための課題の抽出、及び検討した結果をもとにブロック移動順序及び移動経路を算出する方法を記載しました。移動経路算出については、経路ごとの移動時間を計測し、最適な経路の算出方法を記載しました。

4, 5. 設計モデル

















機能実装とアーキテクチャ、及びテスト容易化設計の方針を立てました。それらをもとに構造の概要となるパッケージの設計、及びブロック並べ攻略ソフトの詳細設計を記載し、構造が方針に従いどのように振る舞うかを記載しました。

6. 制御モデル

ブロック並べエリア走行における8パターンのアクションと、AIアンサーゲームにおける全体の戦略を記載しました。次に、それぞれに必要なとなる課題を検証し、その結果を記載しました。その上で、これらにより最大計測時間内で攻略可能な制御戦略となることを検証しました。

トレーサビリティアイコン

各モデルはアイコンでトレーサビリティを確保しています。

 ライントレース	 色判定	 信頼性
 直進・後進走行	 カラーセンサ位置調整	 保守性
 旋回走行	 ブロック並べエリア走行	 効率性
 座標指定走行	 ブロック移動	 機密性
 PIDパラメータ調整	 ブロック並べルート探索	 使用性
 自己位置推定	 ブロック初期位置管理	
 スタート	 カメラシステム	

2. 要求モデル

要求

分析

設計

制御



がんちゃん
+One

2-1. 目標

CS大会競技部門で優勝するために走行タイム16.9秒を目指し、リザルトタイム-11.1秒を獲得することを目標に設定した。
(走行タイム：16.9秒)-(ブロック並べ：23秒)-(直角駐車：5秒)=-11.1秒

2-2. 制約

今回の競技を行うにあたって競技上のルールおよびボーナス獲得のための制約を満たす必要がある。この制約について明確にする。

ルール上の制約

- 制限時間以内でキャリブレーションが終了すること
- 最大計測時間内にゴールすること
- 失格にならないこと
- リタイアをしないこと
- フライングスタートをしないこと

ボーナス獲得のための制約

- 走行体がパワーブロックに接触しないこと

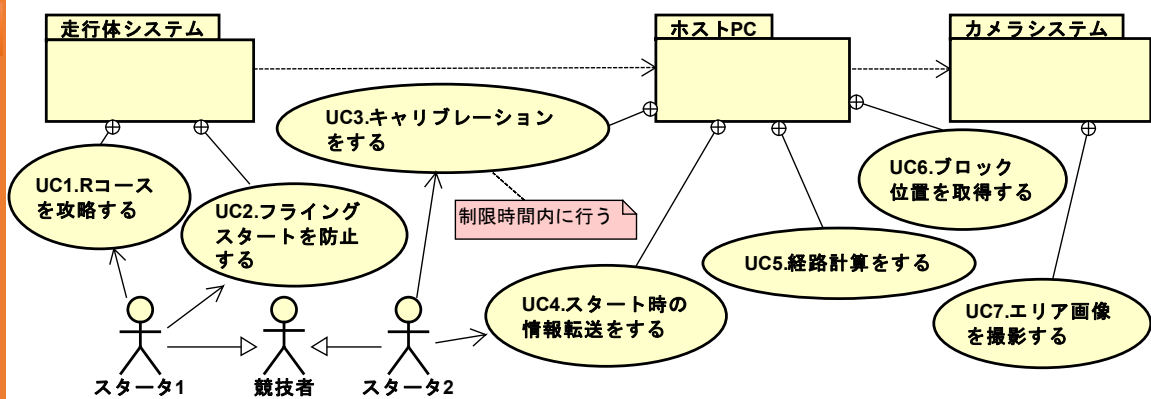
2-3. 品質

ISO/IEC9126の品質モデルに基づき品質要求を明確にする。

- ・信頼性 - コースアウトしないこと
- 色を誤検出しないこと
- 運搬中カラーブロックを離さないこと
- ・保守性 - ハードの異常を検知すること
- ・効率性 - 計算量の重い処理を
ホストPCで行うこと
- ・使用性 - キャリブレーションを自動化すること
- 誤操作によるスタートが行われないこと
- ・機能性 - 障害物に当たらないこと
- 直角駐車場に衝突しないこと

2-4. ユースケース分析

競技を攻略するため、走行体・ホストPC・カメラシステムの3つを用いてシステムを構築する。それぞれのシステムにおいてユーザに提供する必要のある機能の分析を行った。その結果を、ユースケースパッケージ図を用いて検討し、システムごとの責務を明確にした。



<図2-4-1> 開発システムのユースケースパッケージ図

2-5. スピード競技の要件定義

スピード競技エリアの目標である16.9秒以内のゴールを達成するには以下の要件を満たす必要がある。

- Ⅰ. 走行体が最高速度で走行できること
-ETロボコンフェスタ(7/15)でのゼロヨンレース結果より、走行体の最大速度を実測したところ、74cm/秒であったので、上記速度でスタートからゴールまで走行すること
※カーブでは片側タイヤのみ74cm/秒の速度を維持し走行できること
- Ⅱ. GATE1~GATE2間においてショートカットを行うこと
-ラインのない所を走行し、走行距離を短くすることで、タイム短縮ができること



左図のようにショートカットを行うことで1.0秒の短縮を狙う
※障害物がある場合にはライントレースで進む

<図2-5-1> 想定するショートカットのルート例

2-6. ブロック並べの要件定義

ブロック並べでボーナスタイムの23秒を獲得するには以下の要件を満たす必要がある。

- Ⅰ. パワースポット設置を5箇所実現できること
-初期位置コードからパワースポットの位置をデコードできること
- Ⅱ. ブロック並べを攻略するための動作が可能なこと
-経路に沿って走行できること(ライントレース、座標指定、旋回、直進)
-カラーブロックの色と場所を認識できること
-ラインとブロック置き場の色を認識できること
-パワーブロックの場所を認識できること
- Ⅲ. 最大計測時間内に最高得点を取得可能な最短ルートを選択して走行すること
-移動時間最短のルートを探求できること
- Ⅳ. カラーブロック置き場にあるブロックを迂回して移動できること
-白色エリアを走行できること(座標指定、旋回、直進)



<図2-6-1> 目標とするブロック並べの攻略例

最大得点となるのはパワーブロックも移動させた場合の6箇所達成になるが、最大計測時間内に終了しないケースがあるため5箇所達成を目指す。

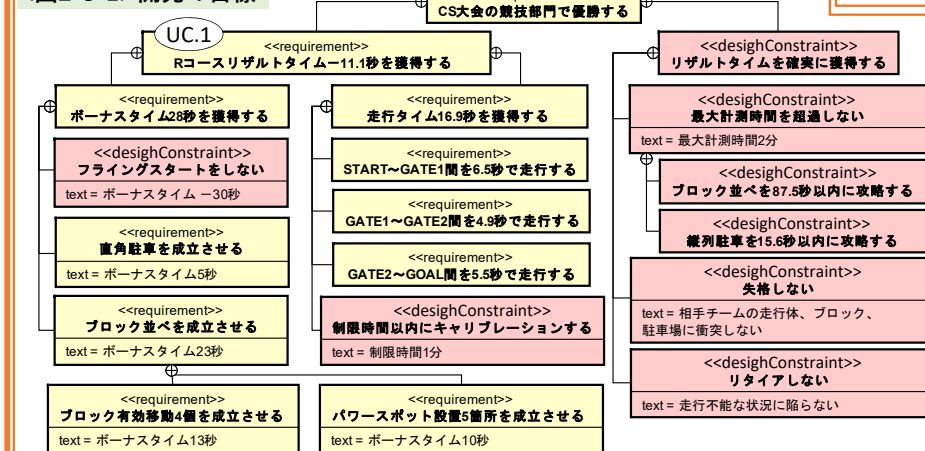
2-7. 直角駐車場の要件定義

直角駐車場でボーナスタイム5秒を獲得するには15.6秒以内に以下の要件を満たす必要がある。

- Ⅰ. 図2-7-1の①②③④の位置から直角駐車用のラインまで走行する
-自己位置推定をもとに走行できること
-旋回及び直進ができること
※①②③④のうち、どれかはブロックが置かれていない場所がある
- Ⅱ. ライントレース走行及び灰色ラインから駐車場へ進入できること
-ラインを検知できること
-自己位置推定で動作できること
- Ⅲ. 直角駐車場到達後に完全停止すること
-モータを完全停止できること

<図2-7-1> 直角駐車場の攻略ルート

<図2-8-1> 開発の目標

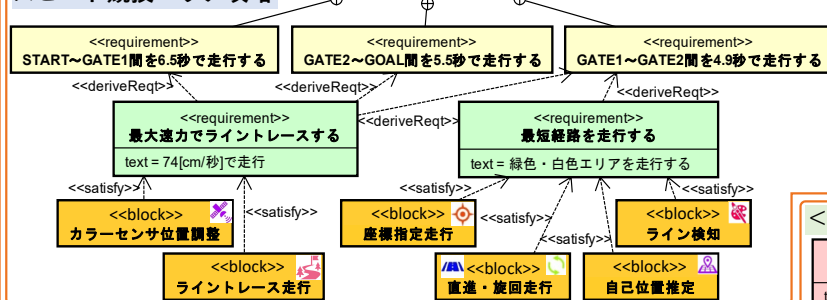


2-1から2-7の内容を基に開発の目標・要求機能・品質・制約を検討した結果から得られた内容をSysML要求図で図2-8-1から図2-8-9に示す。

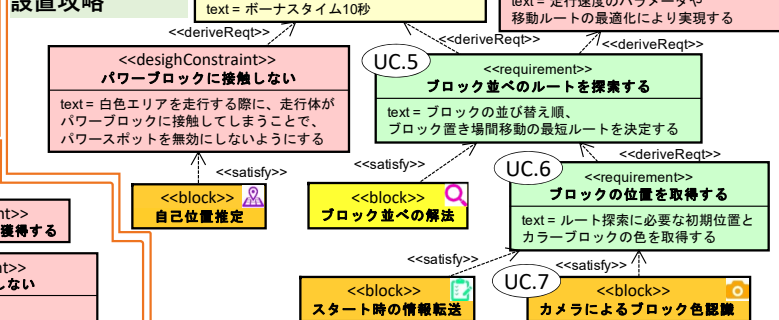
※<<copy>>については紙面の都合上省略している

UC.i: ユースケース記述と対応する箇所

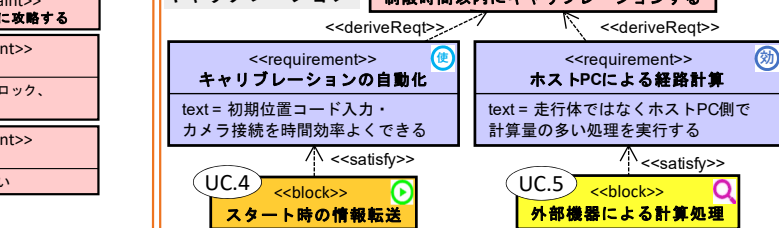
<図2-8-2> スピード競技エリア攻略



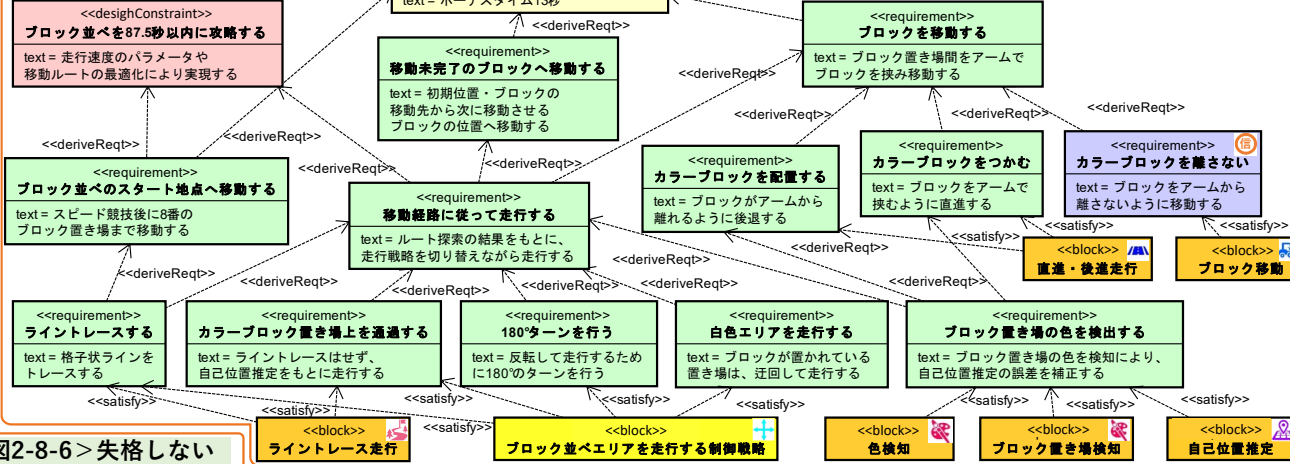
<図2-8-3> パワースポット設置攻略



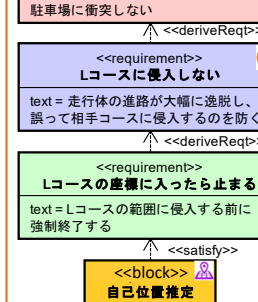
<図2-8-4> キャリブレーション



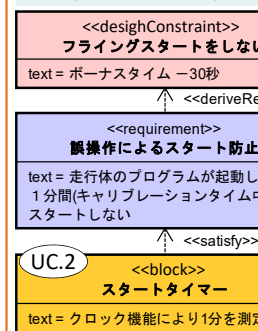
<図2-8-5> ブロック並べエリア攻略



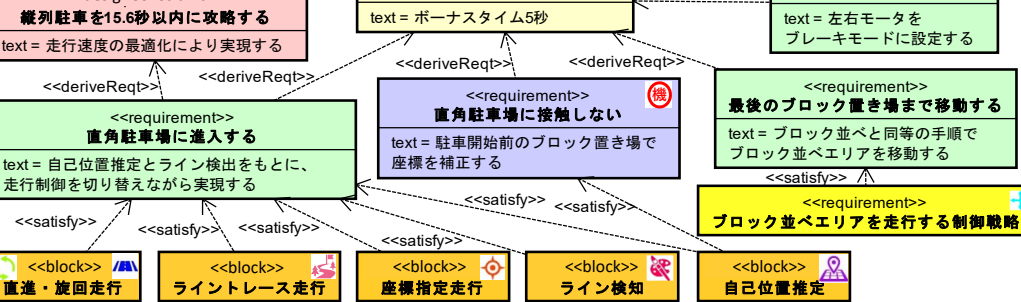
<図2-8-6> 失格しない



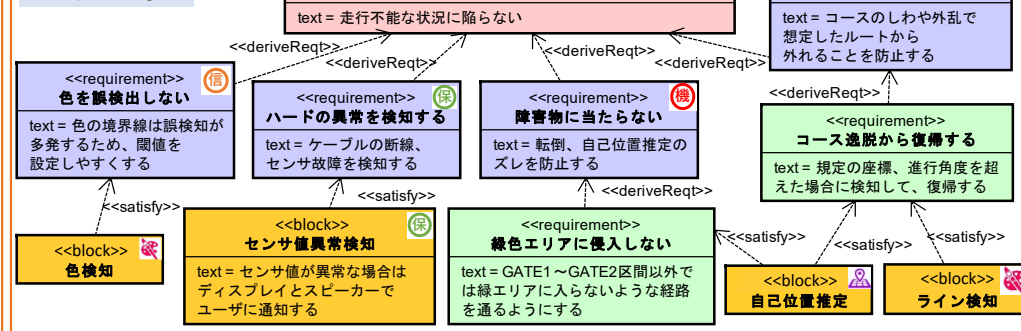
<図2-8-8> フライングスタートしない



<図2-8-7> 直角駐車攻略



<図2-8-9> リタイアしない



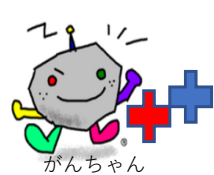
3. 分析モデル

要求

分析

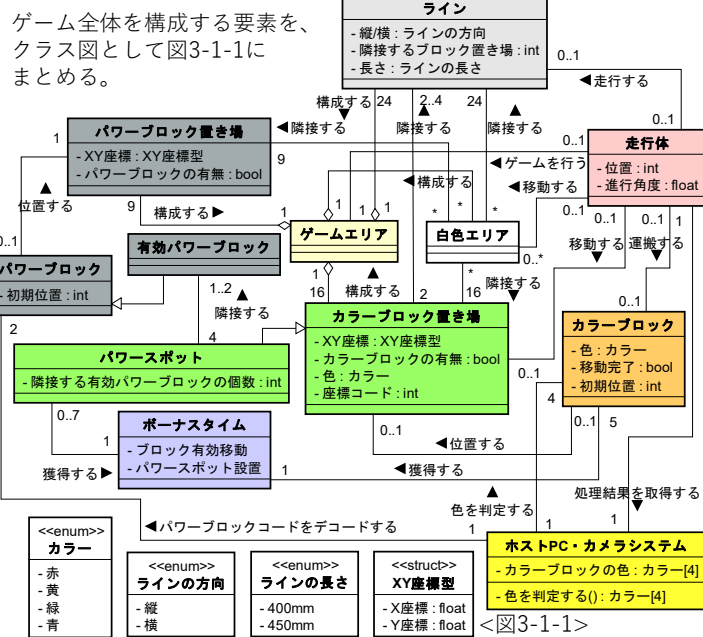
設計

制御



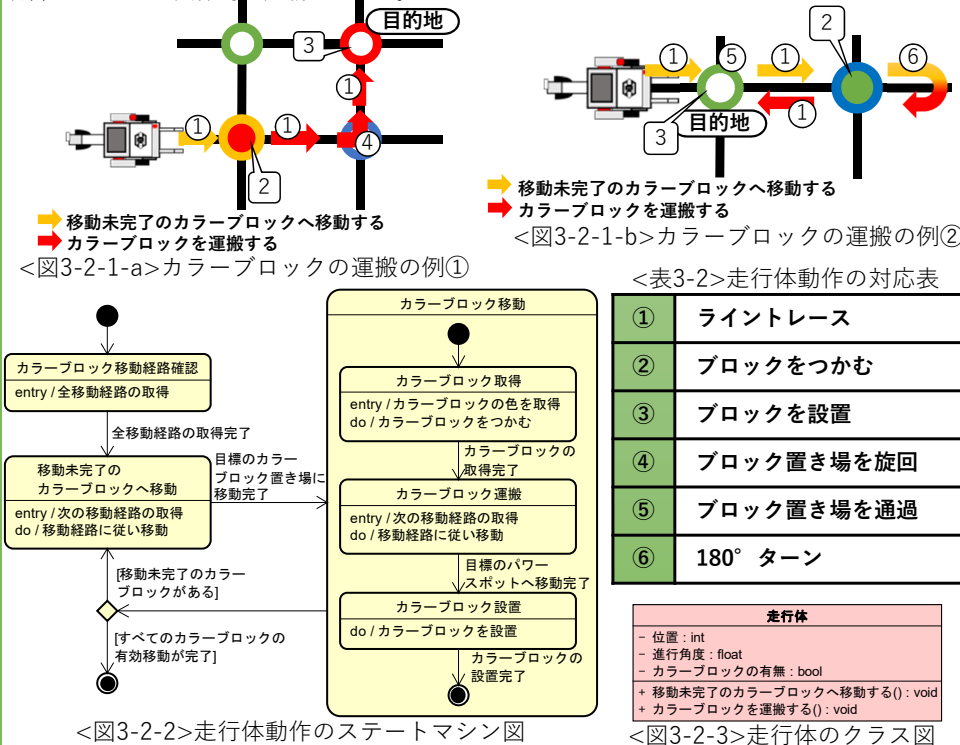
がんちゃん
+One

3-1. ゲームの要素定義



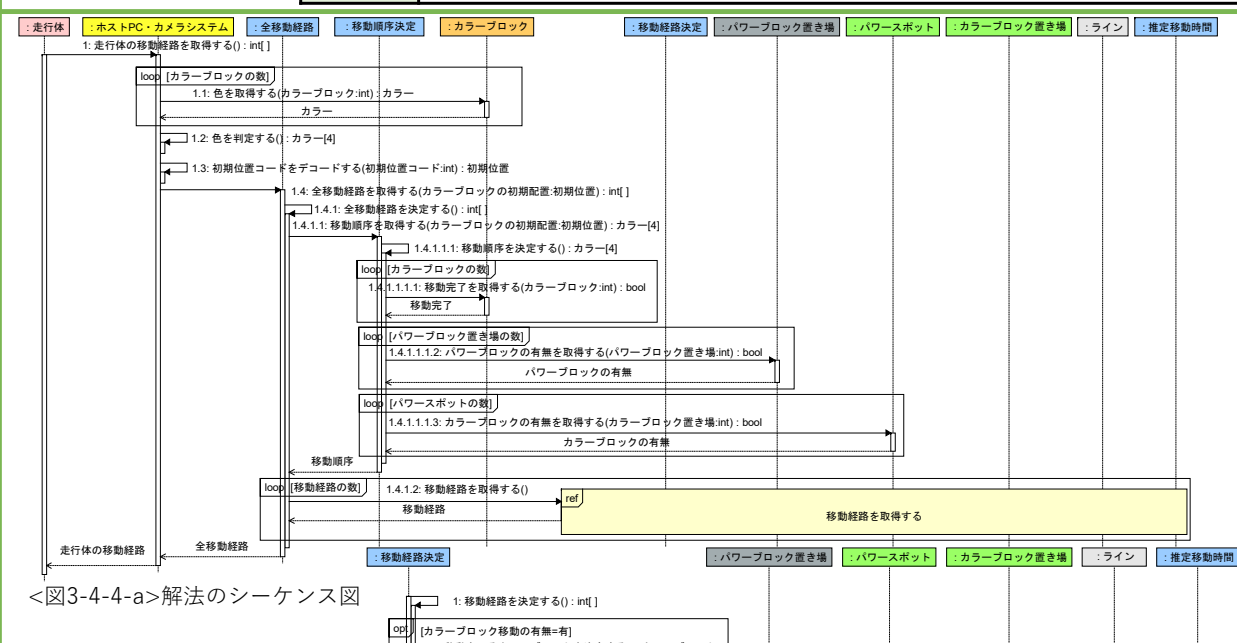
3-2. 走行体の動作定義

図3-2-1-a及び図3-2-1-bに、移動未完了のカラーブロックを運搬する走行体の動作の例を示す。これらの図の①～⑥は、表3-2に示した走行体の動作と対応している。一連の動作は大きく「移動未完了のカラーブロックへ移動する」動作と「カラーブロックを運搬する」動作の2つに分類することができる。各カラーブロックに対し、図3-2-1-aや図3-2-1-bのような動作を繰り返すことでゲームを攻略する。上記の動作例から、カラーブロック運搬に関わる走行体の一連の動作を図3-2-2のステートマシン図で示す。また、走行体は1度に1つのカラーブロックしか運搬できないため、「カラーブロックの有無」の情報が必要である。以上を踏まえ、走行体のクラス図を図3-2-3の通り示す。なお、それぞれの動作をどのように実装したかは、制御モデル6-1に具体的に記載している。



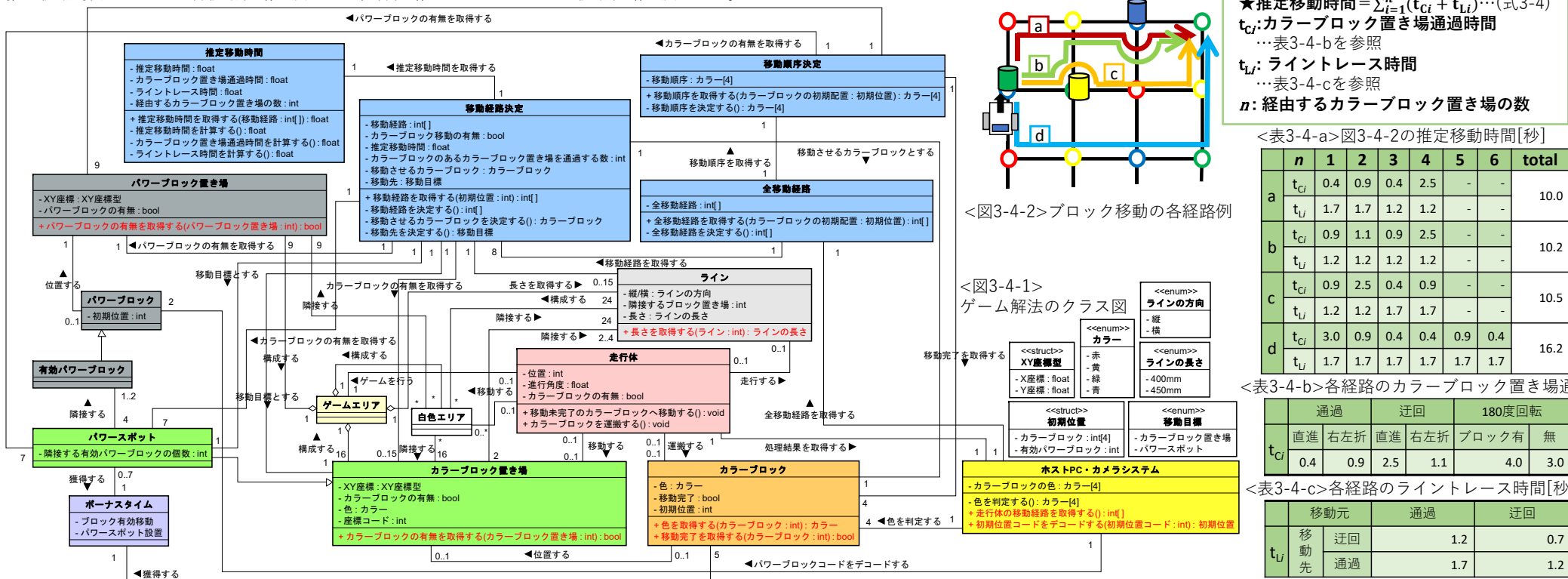
3-3. 指針

課題①	カラーブロックを運搬する順番はどう決定するか？
指針①	走行体の位置から一番近いカラーブロックを移動させる。ただし、課題②の状況の場合は指針②を優先させる。
課題②	図3-1-2のようにパワースポット設置が困難な配置の場合はどうするか？
指針②	パワースポット上にあるカラーブロックを優先して移動させる。
課題③	ブロック並べを87.5秒以内に攻略するために、移動経路をどう決定するか？
指針③	移動時間が最も短くなるように、推定移動時間を用いて経路を算出する。



3-4. 解法

図3-4-1にゲーム解法のクラス図を示す。走行体の動作定義及び指針により図3-1-1から追加されたクラスを青、追加された操作を赤で示す。「全移動経路」は「カラーブロック移動経路決定」と「移動順序決定」からゲームの全移動経路を決定する。指針を検討した結果より、図3-4-2における「移動時間が最も短くなる」経路は式3-4で示す推定移動時間が最短の経路である。ただし、最短の経路が複数存在する場合は、リスク軽減のためカラーブロックのあるカラーブロック置き場を経由する回数が最小の経路とする。表3-4-aでは図3-4-2で示した経路において推定移動時間がどのように計算できるかを示している。なお、推定移動時間は表3-4-b及び表3-4-cの値を用いて算出される。また、走行体クラスはホストPC・カメラシステムクラスから走行体の移動経路を取得することでゲーム攻略を行う。次に図3-4-3には移動順序を導出するアクティビティ図を示す。なお、図3-4-3では地区大会の配置情報を前提としているが、他の初期配置でも適用可能である。最後に、図3-4-4-aおよび図3-4-4-bのシーケンス図に図3-4-1における各クラスの振る舞いを示す。この図に示す順番に従って、ゲームを構成する各クラスから情報を取得し、移動順序を決定した後推定移動時間に基づいて各移動経路を決定して、各経路をまとめることで全移動経路を決定する。



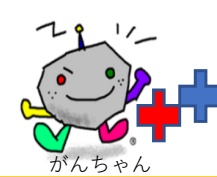
4. 設計モデル (設計方針・構造)

要求

分析

設計

制御



がんちゃん
+One

4-1. 設計意図・方針

要求・分析・制御モデルの内容に基づいて、設計の方針を検討した結果を記載する。

機能実装の方針

課題①	「2-4. ユースケース分析」をどのように実装するか？
方針①	走行制御などのリアルタイム性が求められる処理は走行体で実行し、ルート探索・カメラ処理・ブロック位置管理などの高負荷処理はホストPCで実行する。
課題②	「3-4. 解法」をどのように実装するか？
方針②	ホストPC上に、エリア上の情報を展開してルートを探索する。その結果を走行体が取得し、経路の通りに走行区間ごとに走行する。
課題③	「6-1. ブロック並べエリア上を走行する制御戦略」をどのように実装するか？
方針③	8つのアクションとその走行区間ごとに区間管理し、終了条件に応じて切替ながら実現する。
アーキテクチャの方針	
課題①	「複数人での並行開発」と「ソフトウェアの複雑さ低減」をどのように実現するか？
方針①	ブロック並べルート探索・ブロック初期位置管理・走行体制御・デバイス情報管理・区間切替などに分割することにより並行開発をしやすくする。
課題②	走行体システムのモジュールをどのように分割するか？
方針②	走行体システムについては、STS分割(Source/Transformation/Sink)に準拠し、センサからの入力・区間攻略の切り替え・モータへの出力の分割になるように設計をする。
課題③	来年度以降も競争力を維持するための再利用性や移植性をどう確保するか？
方針③	新タイヤに対応したデバイス管理・走行体制御などは来年度もそのまま使えるように、再利用性や移植性が高い設計にする。

テストを容易にする設計の方針

課題①	2018年版ブロック並べのルート探索が実装できていない状況で、ブロック並べエリア上を走行する制御をどのようにテストできるようにするか？
方針①	走行体とホストPCの接続をシンプルに設計し、シミュレーションできるようにする。その箇所に昨年度までのブロック並べのルート探索を実装しテストをできるようにする。
課題②	膨大な組み合わせのブロック配置のパターンをどのようにテストするか？
方針②	ブロック並べエリア情報の設定箇所と経路決定の方法をシンプルに設計する。その箇所にブロックの初期配置の全パターンをテストパターンとして注入し、全ての解を全自動でテストできるような設計にする。

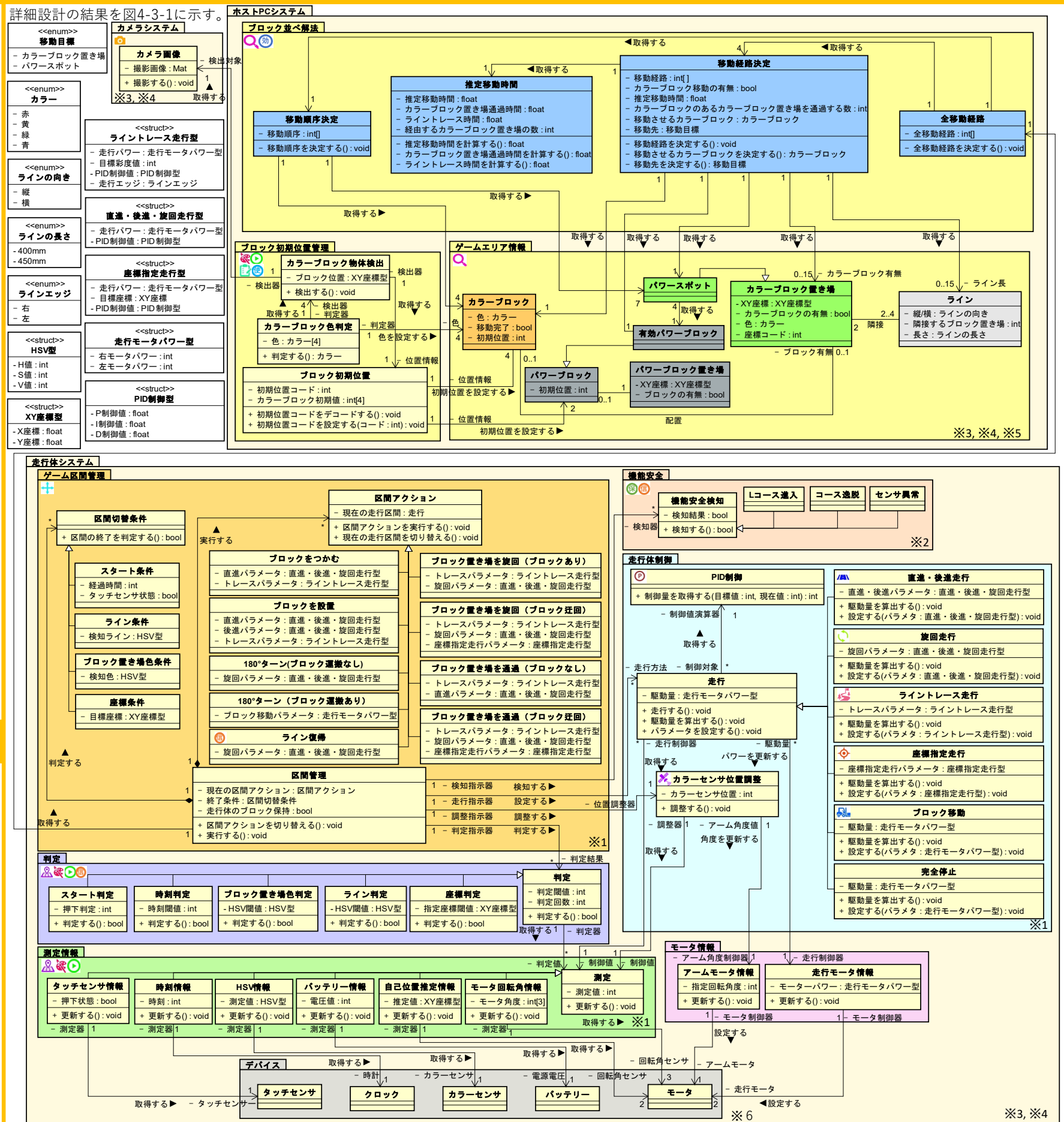
4-2. パッケージ構造

4-1の方針をもとに、実装する機能をパッケージ分割した。その結果を図4-2-1に示す。また、表4-2-1にそれぞれのパッケージの責務を示す。
＜表4-2-1＞各パッケージの責務

パッケージ名	責務
カメラシステム	ブロック並べエリアの画像を取得
ブロック初期位置管理	ブロック色検出・初期位置コードのデコード
ゲームエリア情報	ブロック並べゲームの攻略に必要な情報
ブロック並べ解法	ブロック並べゲームのルート探索
ゲーム区間管理	ルートに沿って区間ごとに動作を切替
機能安全	異常状態の検出
判定	区間切替のための条件判定
走行体制御	モータに与える動作量を算出
測定情報	各センサから現在値を取得
モータ情報	モータに設定する値を更新
デバイス	EV3の各種APIに相当

＜図4-2-1＞ブロック並べゲーム攻略ソフトのパッケージ図

詳細設計の結果を図4-3-1に示す。



＜図4-3-1＞ブロック並べ攻略ソフトのクラス図

※1. 周期駆動タスクのクラスは省略する ※2. 機能安全の各クラスと他のクラスとの関連は省略する ※3. Bluetooth関連クラスは省略する
※4. setter/getterは省略する ※5. クラスの色は分析モデルのクラス図の色と同一にしている ※6. EV3APIの属性・操作は記載を省略する

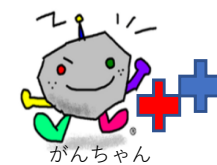
5. 設計モデル (振る舞い)

要求

分析

設計

制御



がんちゃん
+One

5-1. 走行体システムの振る舞い

走行体システムのブロック並べエリアを走行する振る舞いを示す。

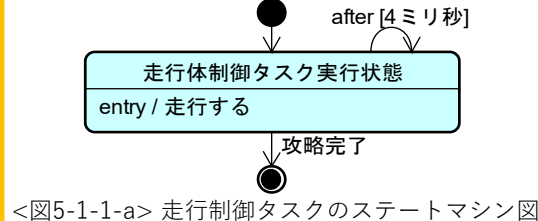
5-1-1. 各タスクの振る舞い

走行体システムにおいては、コース逸脱などを防止するために、走行制御タスクについては優先的に動作させるようにした。また、区間管理についてはCPU負荷が低くなるような設定にしている。

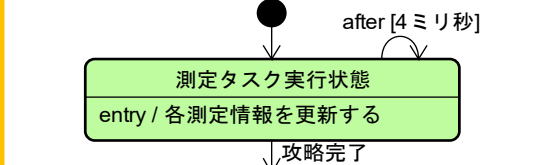
<表5-1-1> 各タスク設定の詳細

タスク名	責務	優先度
走行制御	モータ駆動量の算出 (Sink)	高
測定	各センサ値の更新 (Source)	中
区間管理	区間攻略の切り替え (Transformation)	低

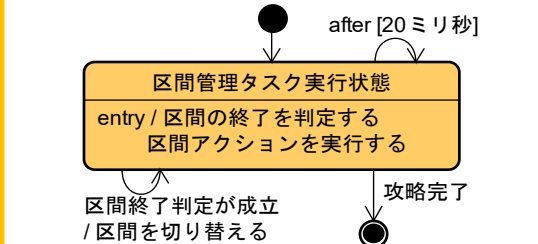
各タスクの振る舞いを、図5-1-1-aから図5-1-1-cに示す。なお、各タスクの駆動周期は実際の走行結果を基に設定した。



<図5-1-1-a> 走行制御タスクのステートマシン図



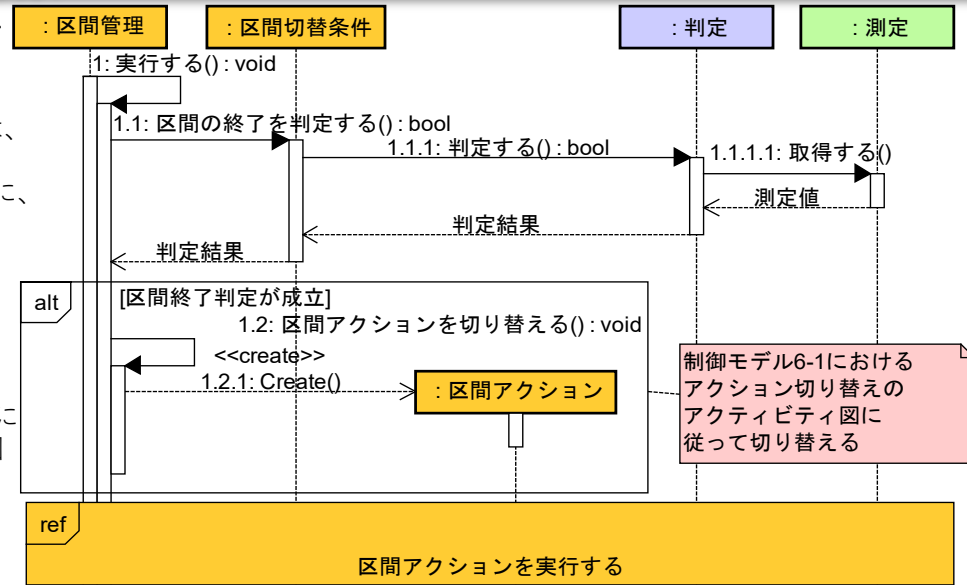
<図5-1-1-b> 測定タスクのステートマシン図



<図5-1-1-c> 区間管理タスクのステートマシン図

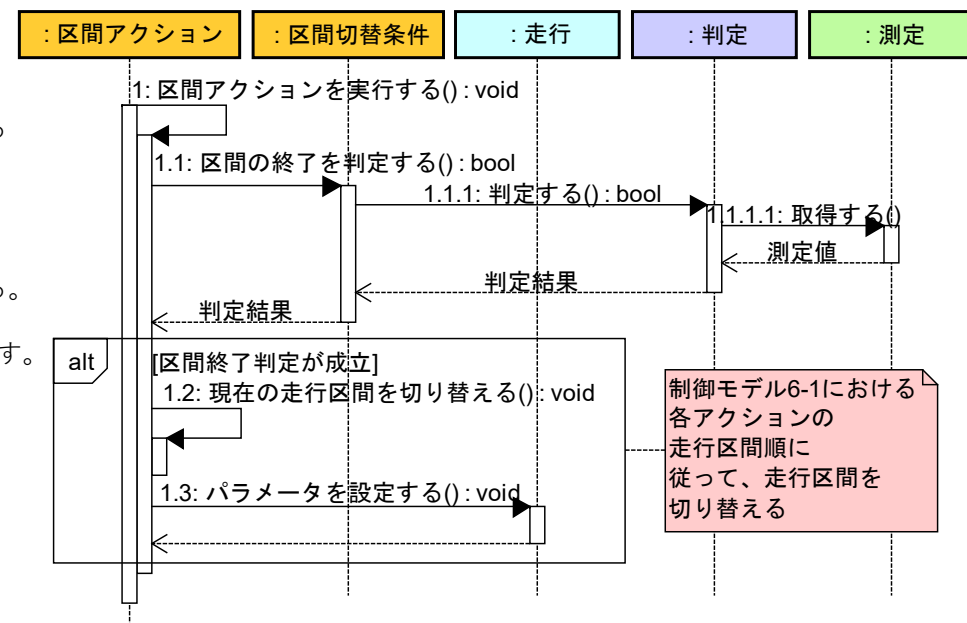
5-1-3. 区間管理の振る舞い

区間管理は区間アクションと走行区間の実行を主な責務としている。全体の処理の流れとしては、現在の区間アクションの終了を判定/切替をした後に、現在の区間アクションを実行するようにしている。その全体の振る舞いを図5-1-3-aに示す。各区間のアクションはホストPCによるルートの探索結果と制御モデル6-1に記載したアクティビティ図に基づいて切り替えを行うようにしている。



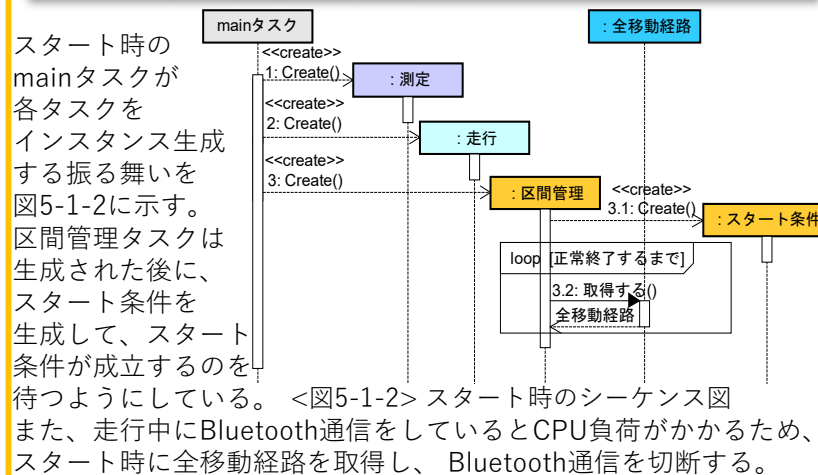
<図5-1-3-a> 区間管理タスク「実行する()'のシーケンス図

区間アクションの実行は走行区間の切替により実現している。走行区間とは直進などの走行ロジックが実行される区間のことであり、その区間を状況に応じて切り替えることで、区間アクションの全体を実現できるようにしている。その区間アクションの振る舞いを図5-1-3-bに示す。なお、全走行区間のインスタンス生成は区間アクション生成と同時に進んでいる。その理由としては、区間アクション生成の時点で実行するべき走行区間が確定しているからである。



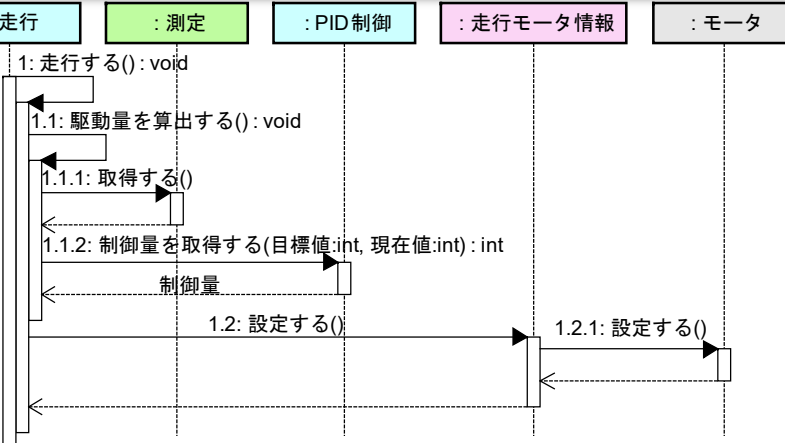
<図5-1-3-b> 区間管理タスク「区間アクションを実行する()'のシーケンス図

5-1-2. スタート時の振る舞い



5-1-4. 走行制御の振る舞い

走行制御タスクは区間管理が設定したパラメータに従って動作している。その振る舞いを図5-1-4に示す。なお、座標指定走行などの各走行は、同様の振る舞いとなるためそれぞれの具体的な走法の記載は省略している。



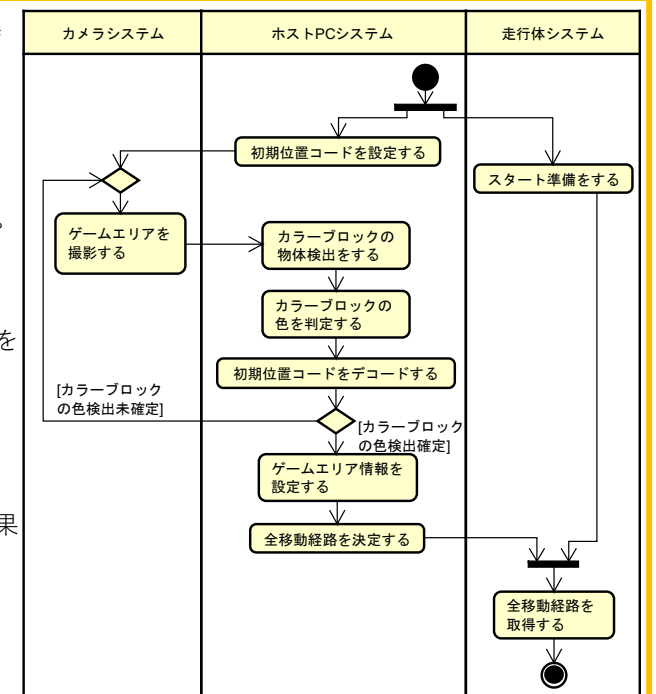
<図5-1-4> 走行制御タスク「走行する()'のシーケンス図

5-2. ホストPCシステムの振る舞い

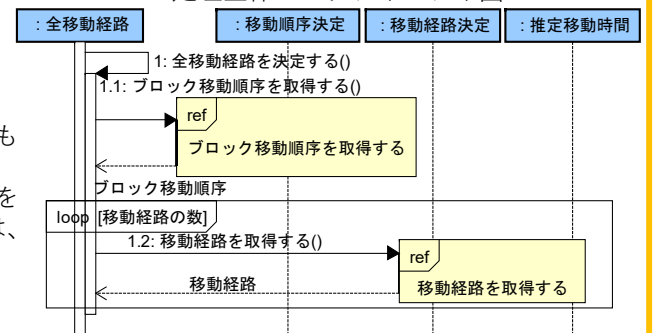
ホストPCシステムのスタート時・ブロック並べ攻略に関する振る舞いを図5-2-1に示す。スタート時にはスタータは2人いるため、ホストPCシステムと走行体システムは並列して、スタートできるようにしている。使用性を考慮して、初期位置コードは最初に入力するようにしており、キャリブレーションタイム中に入力を完了できるようにしている。また、下記の3つのデータを照らし合わせてブロックの色検出の確定/未確定を判定している。

- ・カラーブロックの検出位置
- ・初期位置コードのデコード結果
- ・検出したブロックの色

判定の結果、ブロックの色が確定し、全移動経路の決定がされた後に、走行体システムにデータを送るようにしている。もし仮に、キャリブレーションタイム中に何かのトラブルで走行体システムを再起動する必要があった場合でも、ホストPCと走行体システムは並列して動作しているため、全移動経路を再度計算しなくても良いようにしている。次に、全移動経路を算出する振る舞いを図5-2-2に示す。この振る舞いは、分析モデルの3-4解法と同様の振る舞いとなるため、詳細な記載は省略する。



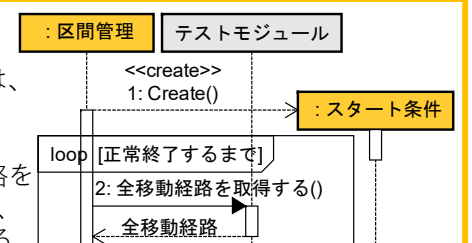
<図5-2-1> ホストPCシステムにおける処理全体のアクティビティ図



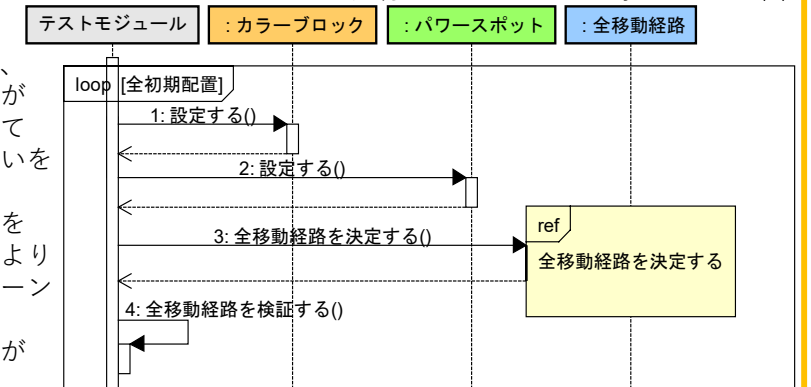
<図5-2-2> 全移動経路決定のシーケンス図

5-3. テスト実施時の振る舞い

テストを実施する際にはテストモジュールを接続して効率よくテストを実施する。ここでは、4-1の方針に記載した2つの項目についてのみその振る舞いを示す。ブロック並べエリアの走行開始時にテストモジュールから全移動経路を取得する振る舞いを図5-3-1に示す。この時の、走行体を通る経路は2018年からのルールであるパワスポットなどは実装されていないが、走行体制御の動作確認には十分なものである。また、全移動経路が決定する経路が妥当であることを、テストモジュールが全初期配置についてテストする振る舞いを図5-3-2に示す。このテストは数日を要するが、これにより想定外の配置パターンが見つかるなど、品質の向上に効果があった。



<図5-3-1> ブロック並べエリア走行テストのスタート時シーケンス図



<図5-3-2> 全移動経路決定のテスト実施シーケンス図

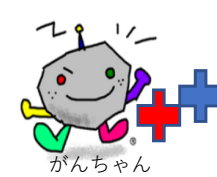
6. 制御モデル

要求

分析

設計

制御



がんちゃん +One

6-1. ブロック並べエリア上を走行する制御戦略

6-1-1. ブロック並べのアクション

ブロック並べはルート探索の結果に従い、下記の8パターンのアクションを切り替えながら攻略する。アクションの切り替え方を図6-1-1-aに示す。それぞれのアクションノード内の各動作は区間ごとに図6-1-1-bから図6-1-1-iに示す。今回、時間内にクリアできることを確認するテストを行ったが、全パターンのテストを行うことは出来ないためランダムに100通りのパターンでテストを行った。その結果平均86.7秒、標準偏差7.8秒でクリアできた。クリア時間が87.5秒を超えるパターンがあるが、直列駐車は15.6秒よりも短時間で成功しているので最大計測時間以内にクリア可能である。

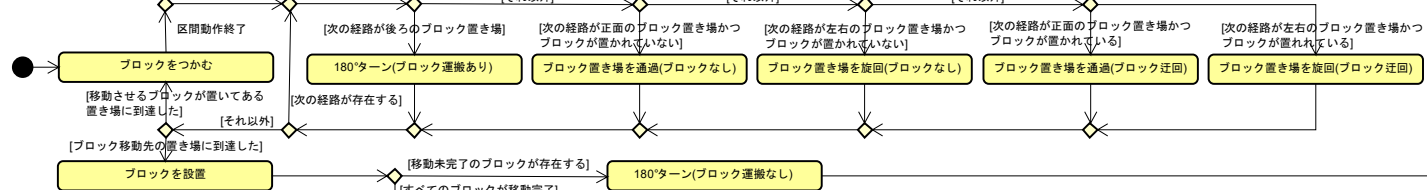


図6-1-1-a>ブロック並べのアクション切り替えに関するアクティビティ図

走行区間の凡例	
順番	動作
終了条件	座標(x,y)[cm] or 角度d[°]
動作に対応する要素技術	
直進/後進/旋回	6-3-1
ライン	6-1-3
座標R	6-3-2
プロ	6-1-4
終了条件に対応する要素技術	
終了条件	要素技術
座標	6-3-2
B検知/L検知	6-1-2

(1) ブロックをつかむ

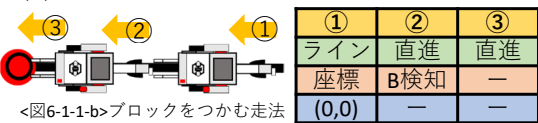


図6-1-1-b>ブロックをつかむ走法

(2) ブロック置き場を旋回[ブロックなし]

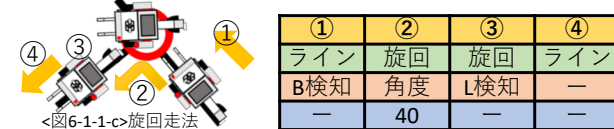


図6-1-1-c>旋回走法

(3) ブロックを設置

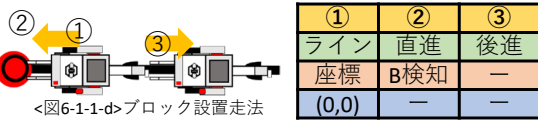


図6-1-1-d>ブロック設置走法

(4) ブロック置き場を旋回[ブロック迂回]



図6-1-1-e>迂回走法

(5) 180°ターン[ブロック運搬なし]

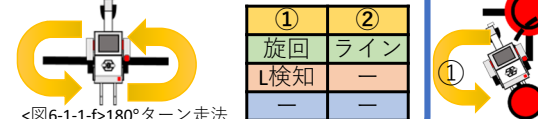


図6-1-1-f>180°ターン走法

(6) 180°ターン[ブロック運搬あり]

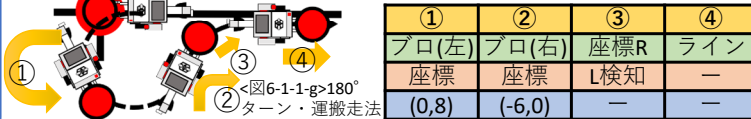


図6-1-1-g>180°ターン・運搬走法

(7) ブロック置き場を通過[ブロックなし]

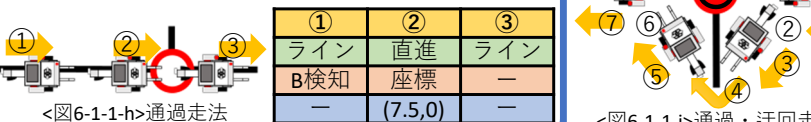


図6-1-1-h>通過走法

(8) ブロック置き場を通過[ブロック迂回]

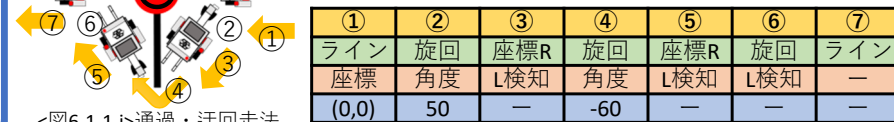


図6-1-1-i>通過・迂回走法

6-1-2. ブロック置き場・ライン検知

<目的>
ブロック置き場・ラインを確実に検知し、自己位置補正したい

<課題>
路面のシワ・光ノイズを、誤検知してしまう

<対策>
連続10回分の履歴から、多数決で検知を確定した

<検証結果>
多数決により、誤検知を低減できた

表6-1-2>多数決対応前後の誤検出回数を測定した結果

	黒	赤	青	黄	緑
対応前	0	0	3	4	0
対応後	0	0	0	0	0

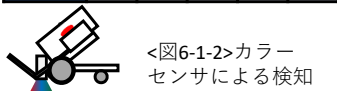


図6-1-2>カラーセンサによる検知

6-1-3. ライントレース

<目的>
コース上のラインに沿って走行できる

<課題>
大会当日にPID制御パラメータ(PID値, 速度, 目標輝度)の調整が間に合わない

<対策>
無線でパラメータを送信して効率よく調整した



図6-1-3>パラメータ送信

6-1-4. ブロック移動

<目的>
ブロック置き場において、ブロックを移動させる

<課題>
旋回動作時に遠心力でブロックを離してしまう

<対策>
左右のパワーを調整した

<検証結果>
実現したい旋回半径で検証し、ブロックを離さないパラメータを採用した

表6-1-4>パワー値と旋回半径を測定した結果

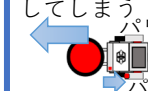


図6-1-4>左右モータのパワー調整

6-3. 2つの制御戦略で共通の要素技術

6-3-1. 直進・後進・旋回走行

<目的>
直進・後進・旋回をズレがなく精密に行いたい

<課題>
左右モータ個体差により回転量にズレが発生する

<対策>
左右のモータ回転角の差を無くすためにP制御を行った

<検証結果>
P制御により、回転角のズレを低減できた

図6-3-1>3.0[m]直進後の回転角のズレ

表6-3-1>3.0[m]直進後のズレ[cm]

P値	0.0	0.5	1.0	1.5	2.0
ズレ	47.8	26.4	13.8	8.6	-0.5

※後進・旋回も同様の結果となる

6-3-2. 座標指定走行

<目的>
コース上の目印のない位置へ自己位置推定を元に移動したい

<課題>
目標座標に向かって途中で位置がズレてしまう

<対策>
目標座標への角度と走行体の進行角度の差分が無くなるようにPID制御で補正した

<検証結果>
P制御により、目標座標との誤差を低減し、競技での十分な精度を実現した

図6-3-2>目標座標への角度と走行体の進行角度の関係

表6-3-2>1.0[m]直進後、出発点に戻した時のズレ[cm]をPゲイン毎に測定した結果

P値	0.0	0.5	1.0	1.5	2.0	2.5
ズレ	9.0	3.9	3.2	2.5	1.3	0.7

6-3-3. 自己位置推定

<目的>
走行体の制御の切り替えに現在位置を使用したい

<課題>
推定座標と実座標に、誤差が生じてしまう

<対策>
自己位置の算出に利用する走行体のパラメータを正確に調整した

<検証結果1> タイヤ半径は5.0[cm]が適正値となった

表6-3-3-a>3.0[m]走行後の推定値ズレ

半径	4.8	4.9	5.0	5.1	5.2
ズレ	14	10	0	-4	-12

※半径,ズレの単位は[cm]

<検証結果2> トレッド幅は14.76[cm]が適正値となった

表6-3-3-b>10回転旋回後の推定値ズレ

幅	17.68	13.72	14.76	14.80	14.84
ズレ	-20	-3	0	3	20

※幅の単位は[cm],ズレの単位は[°]

6-3-4. カラーセンサ位置調整

<目的>
カラーセンサ測定値の精度を向上させたい

<課題>
測定値が、アームの角度により変わってしまう

<対策>
アームの角度とセンサ値の関係を調査し、分解能が一番高くなる位置に調整した

<検証結果>
アーム角度が-10°の時に、カラーセンサの分解能が一番高くなった

表6-3-4>アームの角度とカラーセンサ値の関係

deg	R(白/黒)	G(白/黒)	B(白/黒)
-20	116/7	102/7	182/10
-10	134/7	126/7	195/10
0	126/7	122/6	178/10
10	98/11	96/6	130/12
20	55/11	60/7	66/13

6-2. AIアンサーゲーム出題数字を認識する制御戦略

数字を判定するためのロジックと、実際の走行結果に基づく経路の検討結果を記載する。

6-2-1. 全体の走行経路

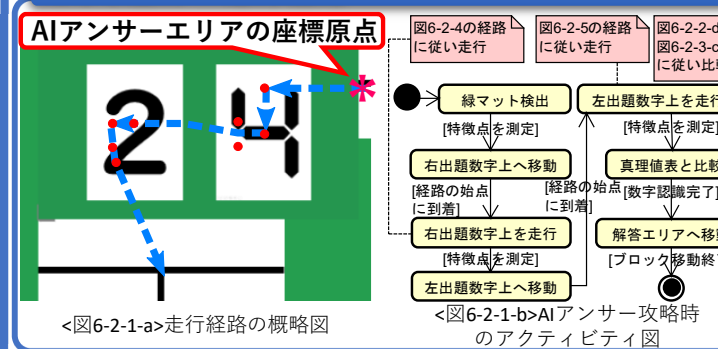


図6-2-1-a>走行経路の概略図

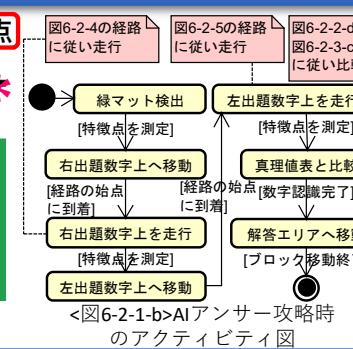


図6-2-1-b>AIアンサー攻略時のアクティビティ図

<課題> 検知に時間をかけすぎると、最大計測時間を超過してしまうので、正解率を落とさずに可能な限り早く数字認識を行いたい。

<対策> 数字を判別するのに最低限必要な特徴点に絞り、走行経路に無駄がないように工夫した。なお誤検知で真理値表と異なる値となった場合は検証時に最も誤検知が多かった数字(右:6, 左:4)と判定する。

<効果> 図6-2-4, 6-2-5の経路で特徴点を測定することで、数字認識を30秒以内で攻略できた。

6-2-2. 右出題数字認識のロジック

<課題> 7セグメント全てを読み取ると、時間がかかり効率が悪い。

<対策> 読み取る辺を絞る。図6-2-2-aの決定木により4つの特徴辺a,e,f,g辺が得られる。同様にa,b,e,g辺も得られる。両者の経路を検討し、より単純に実装可能なa,e,f,g辺を特徴辺とした。

<効果> 4辺のみで0~7を判別することができた。

<特徴辺の導出> 1,7と5,6を区別するためにa,e辺がそれぞれ必須であるため、図6-2-2-aの決定木で最初にa,eで判定を行っている。決定木に従い、例えばae=01, f=1, g=0であるとき数字が3であると判別できる。2段目以降は、どの辺を用いるかで決定木が変わる。

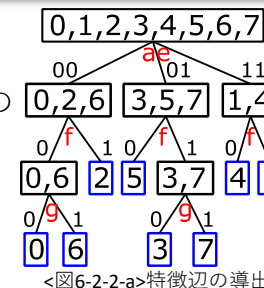


図6-2-2-a>特徴辺の導出

凡例(特徴点の種類)

●: 測定点, ○: 測定点(不定値)

各特徴点の測定結果

0: 黒, 1: 白, x: 不定値

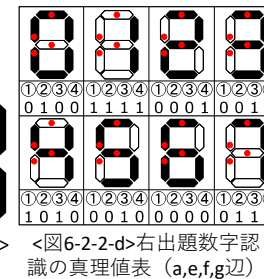


図6-2-2-b>図6-2-2-c>図6-2-2-d>右出題数字認識の真理値表(a,e,f,g辺)

6-2-3. 左出題数字認識のロジック

<課題> 右出題数字とは違い、決まった位置に辺が存在せず、測定しにくい。

<対策> 誤検知しにくい特徴点に絞り、可能な限り少ない測定点で効率良く判別を行った。

<効果> 単純な経路で実現可能な4点のみで0~7を95%の正答率で判別することができた。

<検証> 当初、図6-2-3-aのように特徴点を定めたが誤検知が多く正解率が良くなかった。そこで全ての数字を重ね合わせて見て誤検知しにくい特徴点に絞った結果、図6-2-3-bのように特徴点を決定した。新旧バージョンの正解率の比較結果を表6-2-3に示す。

表6-2-3>新旧の比較

数字	旧	新
0	1000	1000
1	0111	0111
2	1110	1110
3	0110	0110
4	1001	1001
5	0010	0010
6	0000	0000
7	1111	1111

正解率: 旧 70%, 新 95%

図6-2-3-a>図6-2-3-b>図6-2-3-c>左出題数字認識当初の特徴点 決定版の特徴点の真理値表(決定版)