ETロボコン2023
デベロッパー部門 アドバンストクラス

地区：南関東　地域：神奈川県横浜市

チームNo. 004　チーム名：ろぼ魂.exe

所属：学校法人 岩崎学園 情報科学専門学校

メンバー紹介，目標，意気込み

■チーム紹介
私たち『ろぼ魂.exe』は有志のメンバーで集まった4年生4人、2年生2人の計6人で構成されています。

■目標
CS出場‼

■意気込み
情報科学専門学校初めてのCS大会出場を目指して頑張ります。

モデルの概要

ロボコンスナップにてベストショットを確実に獲得する

☆ベストショットを確実にとる
CS出場には高得点が必要。開発目標としてボーナスポイントが一番高い「ロボコンスナップでベストショットを確実に獲得する」と設定した。達成に向けて以下を実現した。
・ミニフィグの画像提出時に送信が失敗したとき、再送処理を行う
・ミニフィグの角度推定をするときにSPIKEより性能の良いPC側で角度推定を実施する

モデルの構成

1. 要求モデル

目標リザルトポイントを『115ポイント以上』と定義してそれを達成するために、以下の要求を抽出した。
・走行ポイント獲得エリアは15秒(15ポイント)、ロボコンスナップはベストショットを確実に獲得する(計50ポイント)
・ベストショットを確実に獲得するするためにISO/IEC 25010の品質特性(機能適合性、性能効率性、信頼性、保守性、移植性)の5項目を取り入れて品質レベルをあげた。
・課題はロボコンスナップを選択した。

2. 分析モデル

HSV値を用いることでしきい値の設定を簡略化、保守性を高める設計とした。
ロボコンスナップにて確実に画像を提出するため、送信失敗時には再送処理を行うことで信頼性を高める設計とした。

3. 設計モデル

複雑な動作に対応し、外部編集が可能な設計を実現しつつ、
相互の通信と独立したタスク分割を保持、独自データの共有でデータ独自性を表現し、
複数人での効率的な開発とテストを容易にするための方針を取り入れた。

4. 制御モデル

各モータの出力の個体差を補正する機能を搭載することで、安定した直線走行を行えるようにした。
ミニフィグの角度推定AIの開発において開発の効率化を実現することができた。

1.要求モデル

1.1 チーム目標

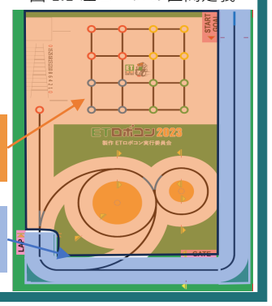
チーム目標は「**ロボコンスナップにてベストショットを確実に獲得する**」ことである。獲得ポイントが計50ポイントと他の難所に比べて高いためロボコンスナップにてベストショットを確実に獲得することをチームの目標にした。
目標を達成するための詳細を以下に記述。

①走行ポイント：15ポイント以上
走行ポイントはロボコンスナップ(ベストショット)に比べあまりポイントが高くないため安全に走行することにした。試走会の結果と普段の走行をもとに15秒でゴールするとしたときに獲得ポイントが15ポイントになる。

②ボーナスポイント：100ポイント以上
難所エリアは獲得ポイントが高いロボコンスナップを確実に攻略する。去年の走行と今年の走行をもとにボーナスポイントは100ポイントを目標にした。

リザルトポイント
①、②より、リザルトポイントを**115ポイント以上**を目標にする。

図 1.1 左コースの区間定義



1.2 システムのユースケース

・目標を達成するためにシステムに必要な機能を【図1.2.1】ユースケース図で抽出し、具体的なふるまいを説明したものを【表1.2.1】ユースケース記述で表した。
・目標であるロボコンスナップにてベストショットを確実に獲得するは [1.3 要件定義]にて詳細を記述。

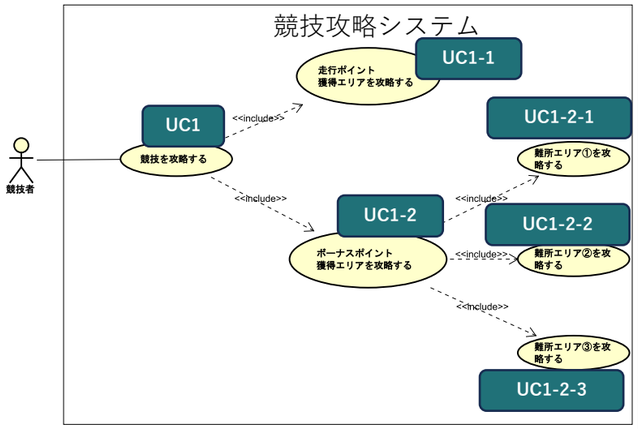


図1.2.1 ソフトウェア全体のユースケース図

ユースケース名	競技を攻略する
概要	競技を攻略する。その中でもロボコンスナップでベストショットを確実に獲得する
アクター	競技者
事前条件	・走行体が無線通信デバイスと接続状態であること ・走行体はスタート操作待ち状態であること ・スタートエリアに設置され、停止状態であること
基本フロー	①走行スタート【UC1】 ②走行ポイント獲得エリアを攻略する【UC1-1】 ③ボーナスポイント獲得エリアを攻略する【UC1-2】 ④競技終了を検知して、走行体を停止する
例外フロー	競技時間経過で強制停止
事後条件	システムが正常終了する

表1.2.1 ユースケース記述

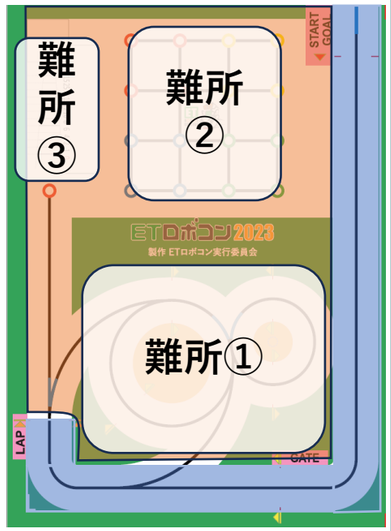


図1.2.2 難所の区間定義

1.3 要件定義

・要求図を用いて【図1.2.1】ユースケース図で抽出した機能ごとに必要な要件を抽出した。
・ロボコンスナップにてベストショットを確実に獲得するため 品質特性についてISO/IEC 25010の観点(機能適合性、性能効率性、信頼性、保守性、移植性)の5項目から要件で検討すべきことを洗い出した。
・紙面の都合上走行ポイント獲得エリアと難所①(ロボコンスナップ)以外の要求は全体が見える程度に記載する。

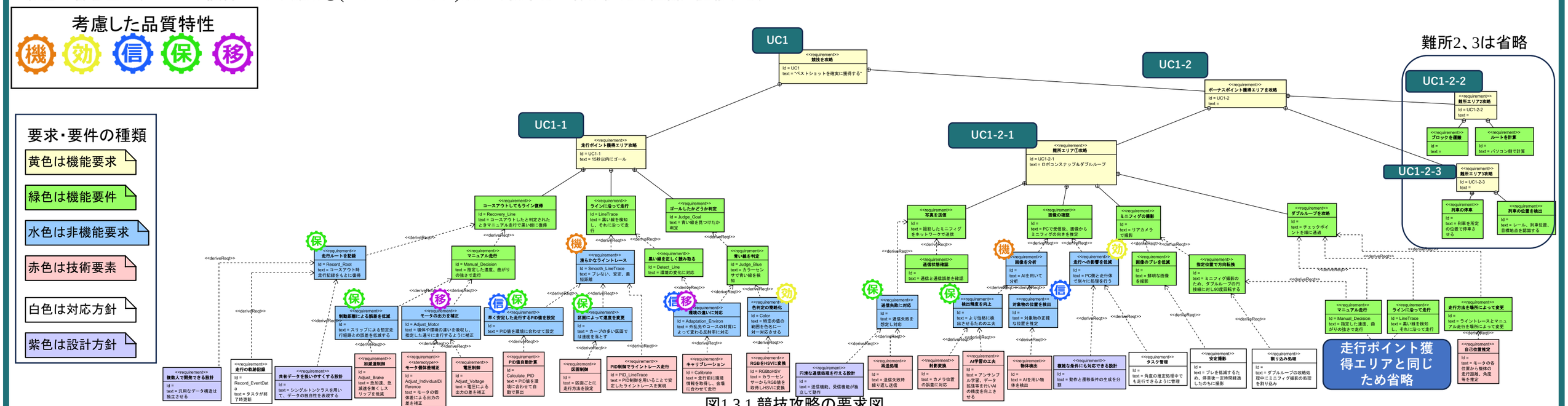


図1.3.1 競技攻略の要求図

2. 分析モデル

2.1 全体構造概要

要求モデルからシステムの構築に必要となる物理要素を抽出する
ミニフィグ角度については「4-2. ミニフィグの認識精度の向上」に詳細を記載
色種別は6種類に定義した。また、より色を分類しやすくするため、RGB値をHSV値に変換し識別を行う。
まずHSVのS(彩度)の値で(赤、青、黄、緑)と(白、黒)の2種類に分類し、以下の処理を続けて行う。
(赤、青、黄、緑)の場合はH(色相)の値により、色を識別する。
(白、黒)の場合はV(明度)の値により、色を識別する。
上記の処理を行うことで、RGB値のように3つの値を同時に比較する
必要が無くなり、分類時のしきい値変更が容易になる。

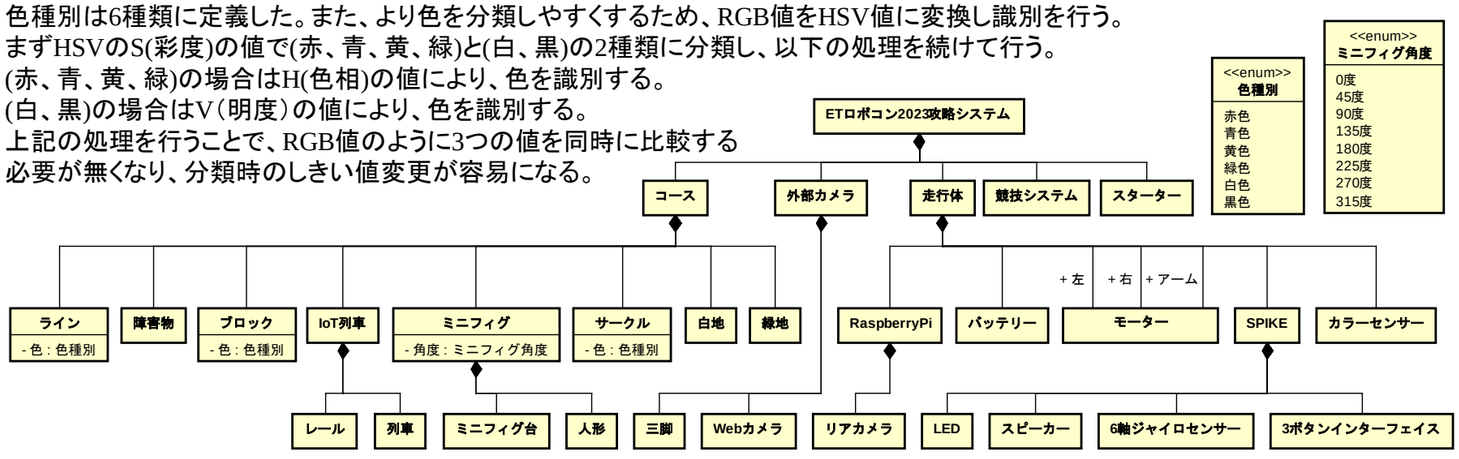


図2-1 全体構造図(クラス図)

2.2 システム間入出力定義

全体の物理部品間の入出力を下図に定義する。画像分析はPCに処理を集中させることで、走行のタイムロスを極力低減する構造としている。

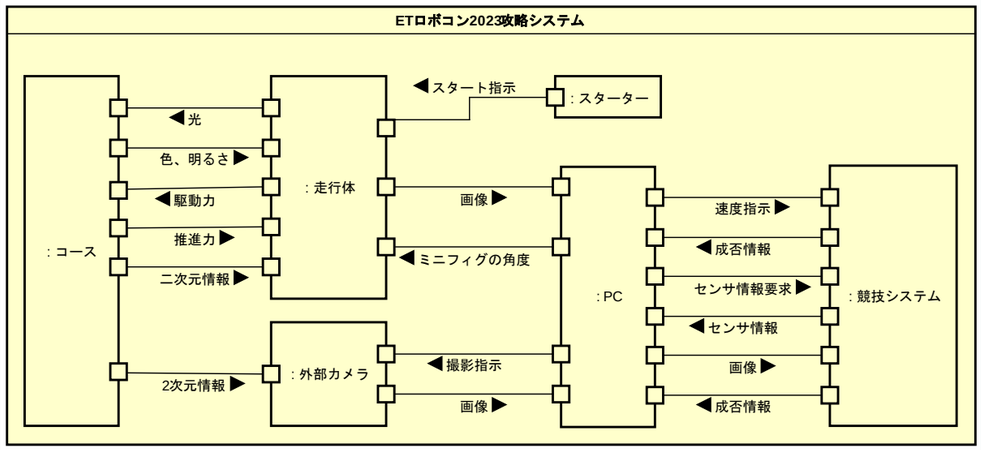
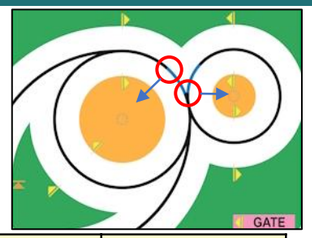


図2-2 全体 I/F 定義図(合成構造図)

2.3 ロボコンスナップ詳細概要

2.3.1 動的構造定義

ミニフィグの「撮影」動作を下図に定義する。
青のラインに侵入した地点で各ミニフィグの「撮影」を行う。
一度目の撮影時にミニフィグの向きを推定し、角度が0度でない場合はミニフィグの正面の地点まで走行し、再度撮影を行う。
このように「撮影」動作で角度を推定、再走行を行うことで一つのミニフィグにつき2回の撮影でベストショットを撮影することができる。



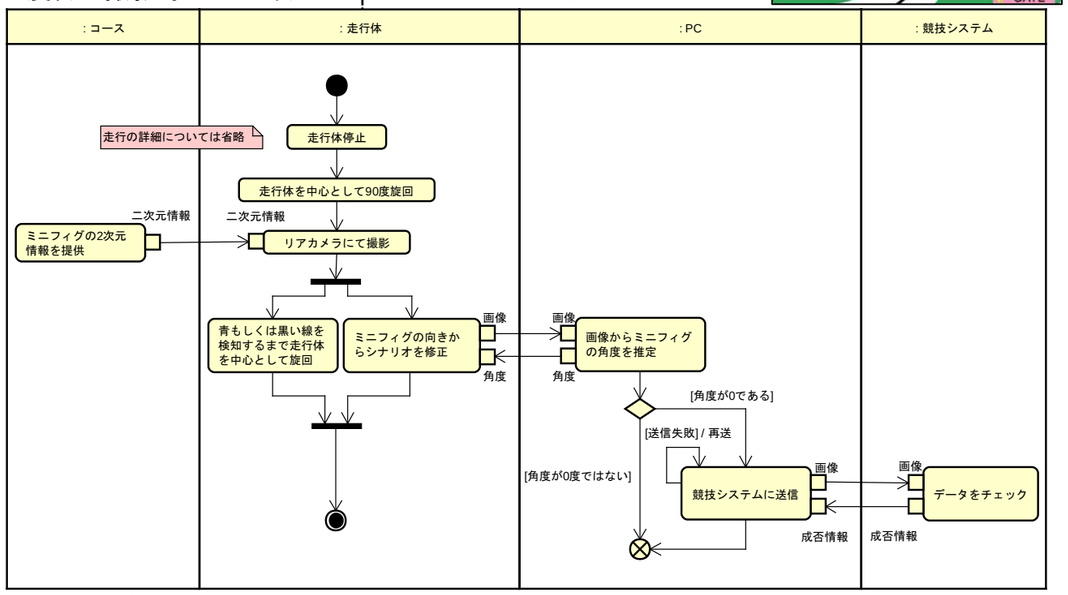


図2-3-1 撮影時動的構造図(アクティビティ図)

2.3.2 静的構造定義

図2-2の走行体内部について詳細を図式化する
図2-3-1の「走行体を中心として90度旋回」動作では車輪回転数からの角度計算(詳細は「4-1走行体制御の精度向上」に記載)を使用する。また、走行体停止状態から旋回を開始することで、正確にリアカメラをミニフィグに向けることができる。
リアカメラによる撮影後は、ミニフィグの角度をPCから受信する。その後RaspberryPi上でミニフィグの正面に停車するまでに必要な走行距離算出、その計算結果をもとにSPIKEにシナリオの修正指示を行う。
上記のようにすることで処理の分担を行う構造としている。

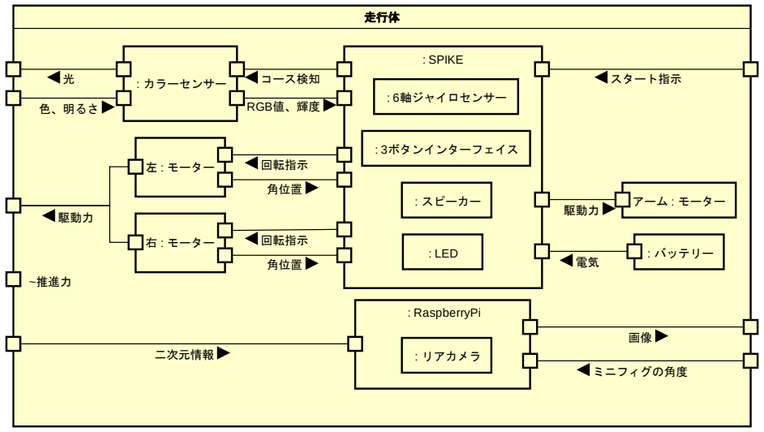


図2-3-2 走行体 I/F 定義図(合成構造図)

2.3.3 システムの振舞

下図にて図2-3-1におけるシステムの振舞を記述する。走行再開準備と角度推定を同時に行うことでミニフィグ撮影を迅速に行うことができるようになる。
また、走行再開準備はカラーセンサーを用いてライン復帰を行うことで確実な復帰を目標としている。

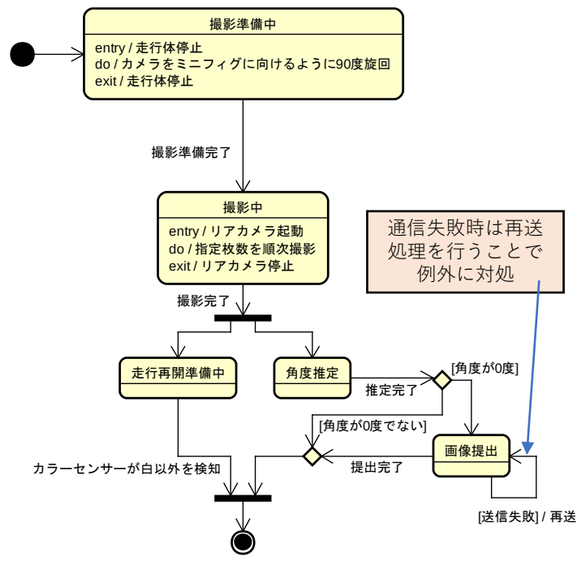


図2-3-3 撮影時振舞定義図(ステートマシン図)

3. 設計モデル（1）

学校法人 岩崎学園 情報科学専門学校 Information Science College

3.1 設計方針

2ページの要求モデルからソフトウェアの設計の方針を決め、下記の表に示した。

要求	目的	設計方針
複雑な条件にも対応できる設計	・走行体動作の保守性(調整)を担保しつつ、複雑な動作に対応する	・動作と、遷移条件の生成を分割する。 ・外部ソフトを用いても編集が可能な設計にする。
円滑な通信処理を行える設計	・通信の送受信が相互にできるようにする	・送信機能、受信機能が独立して動作できるようにタスク分割する
共有データを扱いやすくする設計	・走行体や記録など独自性のあるデータは共有できるようにする。	・シングルトンクラスを用いて、データの独自性を表現する
複数人で開発できる設計	・複数人で開発するため、機能ごとにテストしやすくする。	・各クラスごとに機能を持たせすぎないようにする。 ・汎用なデータ構造は独立させる

表3-1-1 ソフトウェア設計指針

3.2 構造設計（全体）

装置	パッケージ名	役割
走行体	脚本	シナリオとシーンの管理を行い動作と判定の実行の管理をする。
	動作	走行体の一連の動作をまとめている。
	判定	シーンの遷移制御を行うための条件をまとめている。
	情報	データ入出力関連やセンサー値などのデータ保存を行う
	計算	生データを加工、変換する
	デバイス	走行体機能の操作や通信を行う
	Unit	汎用的なデータや構造体を共有で扱えるようにする。
PC	通信管理	走行体、競技システムとの通信を行う
	経路探索	ブロックdeトレジャーの走行経路を計画する
	画像処理	ブロックdeトレジャー及びロボコンスナップの画像分析を行う
	列車管理	IoT列車の攻略を行う

表3-2-1 各パッケージの役割概要

3.1の設計方針、及び要求モデルを元に、
ソフトウェア全体の設計を図3-2-1に示す。
各パッケージの概要については表3-2-1に示す。

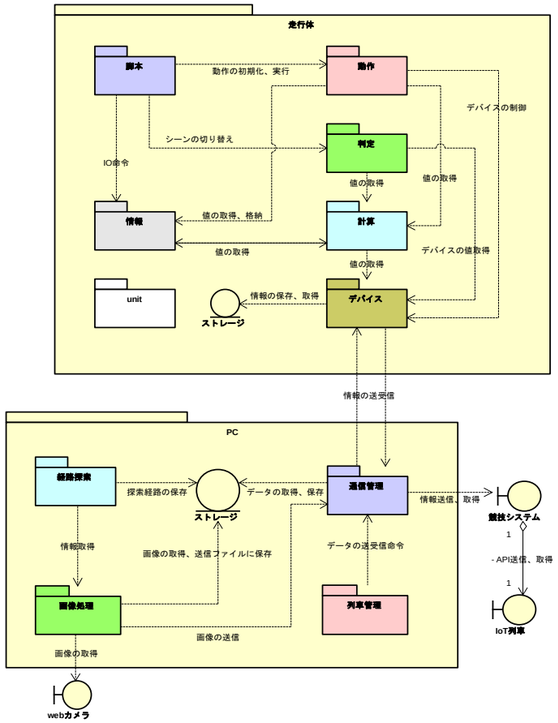


図3-2-1 ソフトウェア設計の全体(パッケージ)

3.3 構造設計（詳細 - 走行体）

3.2の構造設計(全体)にて説明した各パッケージ内部のクラス図を図3-3-1に示す

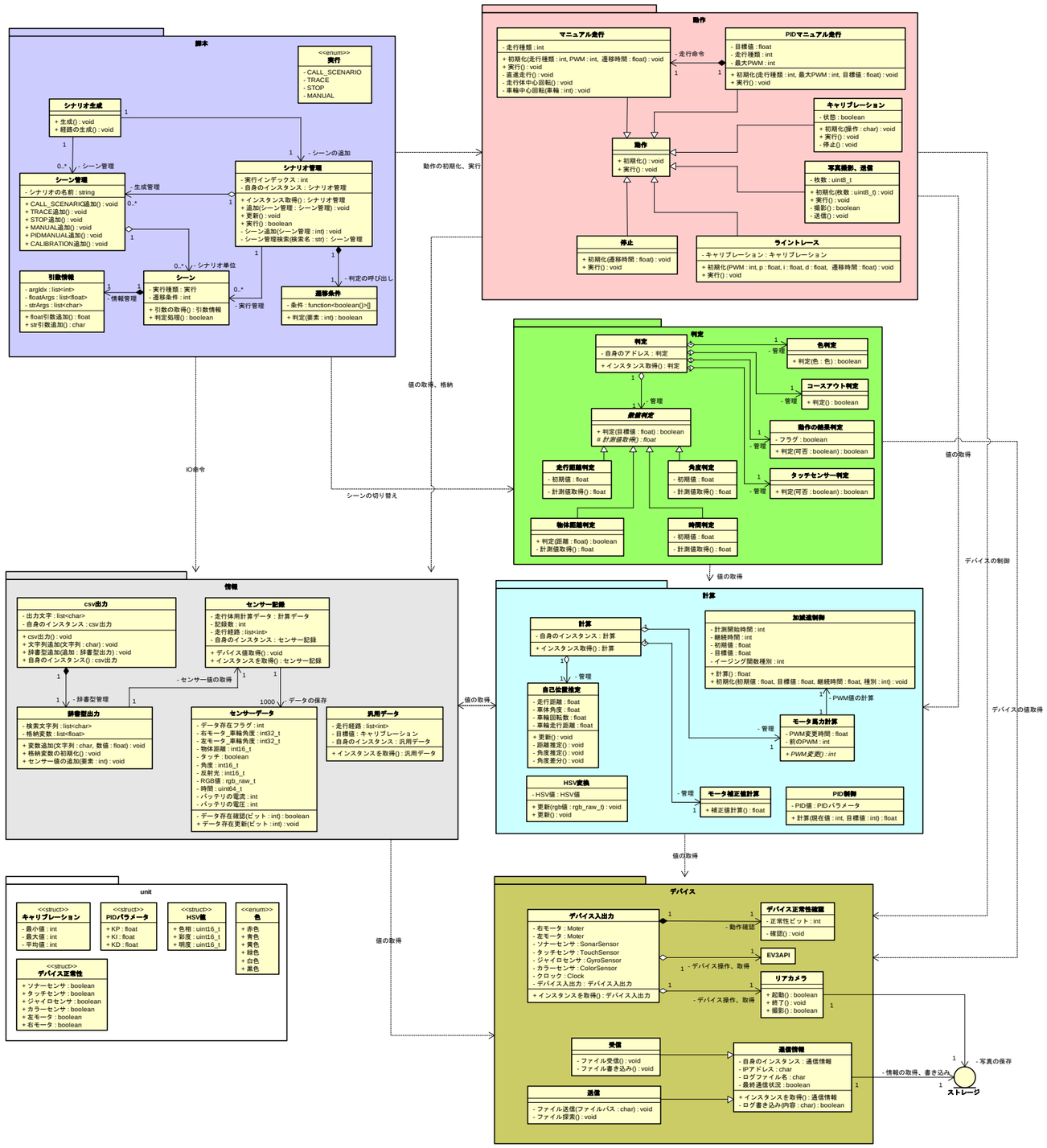


図3-3-1 走行体のソフトウェア設計の全体(クラス)

3. 設計モデル (2)

情報科学専門学校 Information Science College

3.4 構造設計 (詳細 – PC)

3.2の構造設計(全体)にて説明した各パッケージ内部のクラス図を図3-4-1に示す。
ただし経路探索及び列車管理については選択しているものではないため省略する。

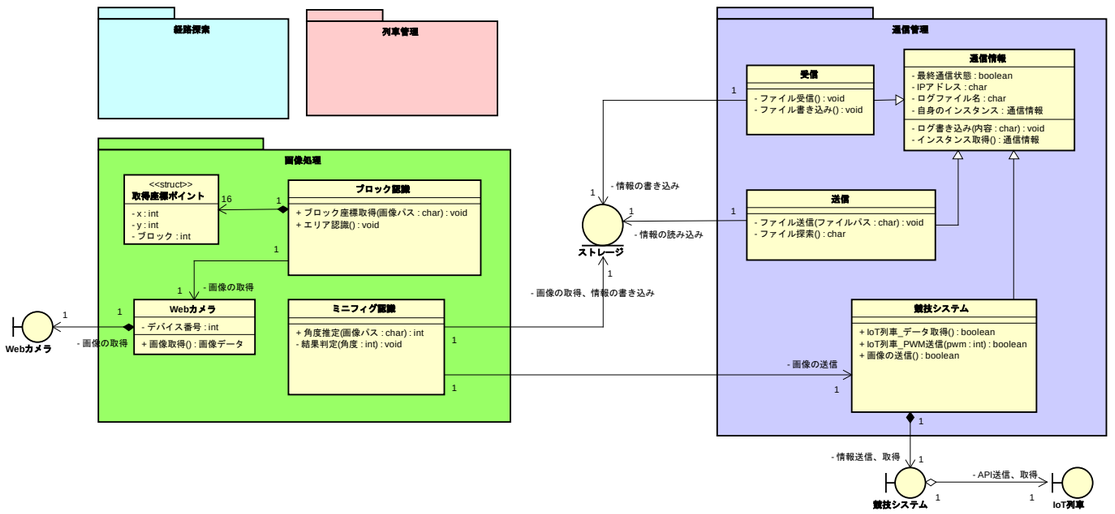


図3-4-1 PCのソフトウェア設計の全体(クラス)

3.5 振舞設計 (タスク)

要求モデル及び設計方針を元にタスクの振舞設計を行い、表3-4-1に示す。
送信、受信タスクについては、走行体とPCとの相互通信を行うためであり、
競技システムとの通信のやり取りは行わない。

装置	タスク	周期	優先度	役割	実行パッケージ
走行体	起動タスク	-	-	各種走行体タスクの立ち上げを行う	-
	主制御タスク	10 ms	高	競技攻略のために走行体全体を制御する。競技中に停止等の問題が発生すると、様々な問題が発生するため優先度は高くする必要がある。	走行体パッケージ全般
	撮影タスク	100 ms	中	主制御で操作することが厳しいため、主制御からの命令があれば撮影する。	デバイス
走行体 PC	送信タスク	50 ms	中	送信用ストレージに格納されたデータを読み取り送信する。	デバイス(走行体) 通信管理(PC)
	受信タスク	50 ms	中	受信待機状態になっており、受信した場合は受信ストレージに格納する。	デバイス(走行体) 通信管理(PC)
PC	難所攻略タスク	10 ms	高	難所を攻略するための情報の導出や競技システムにアクセスし、情報の送受信を行う。	経路探索, 列車管理, 画像処理

表3-5-1 各タスクの概要

3.6 制御システムの状態遷移

3.3 構造設計 (詳細 - 走行体)にて示した脚本パッケージの
具体的な動作について、図3-6-2にて示す。
再コンパイル命令時はブロックdeトレジャーの走行経路を
計算した後、シナリオを追加する必要があるためであり、
再度実行状態に移った場合は途中のシーンからの実行になる。
　また、動作の調整をしやすくするため、ExcelとPythonを用いて
直接コードを書き換え更新できるようなツールを作成した。

	A	B	C	D	E	F	G	H	I
1	動作	引数1	引数2	引数3	引数4	引数5	遷移条件	備考	タイトル
2	STOP	0					j.time.j(500 * 1000)		calibration
3	MANUAL_PID	2	50	-90			j.ret.j(true) and j.time.j(1000 * 1000)		
4	STOP	0					j.time.j(200 * 1000)		
5	CALIBRATION	"record"							
6	MANUAL_PID	2	50	540			j.ret.j(true) and j.time.j(3000 * 1000)	スタート位置から450度回転	
7	CALIBRATION	"stop"							
8	STOP	0					j.time.j(200 * 1000)		
9	MANUAL_PID	5	50	1			j.ret.j(true) and j.time.j(2000 * 1000)		
10	STOP	0					j.time.j(100 * 1000)		

図3-6-1 Excelを用いたシナリオ作成ツール

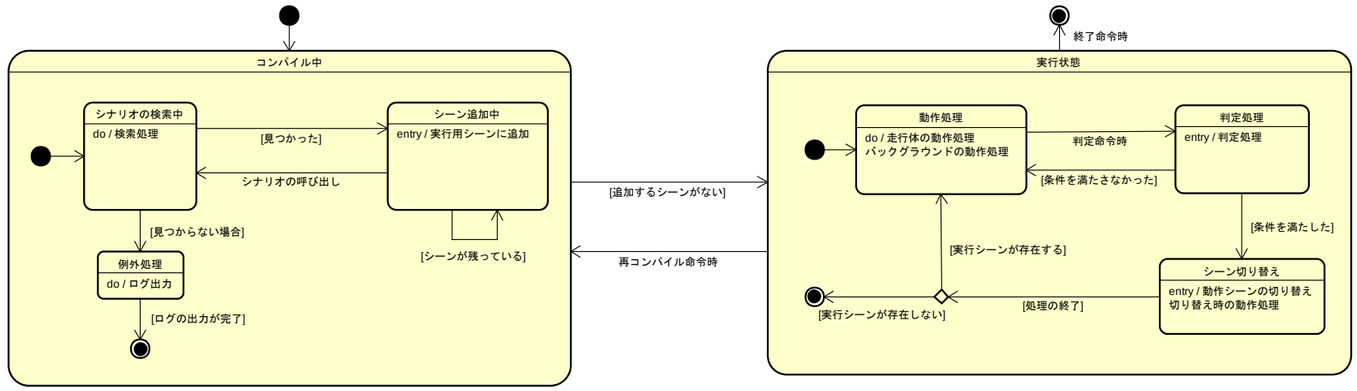


図3-6-2 シナリオのコンパイルと実行の状態遷移

3.7 ロボコンスナップの撮影時の振る舞い

3.1の設計方針、3.3と3.4の構造設計から示した設計から、走行体側の撮影からPC側の画像分析までの
全体の振る舞いの流れを図3-7-1に示す。

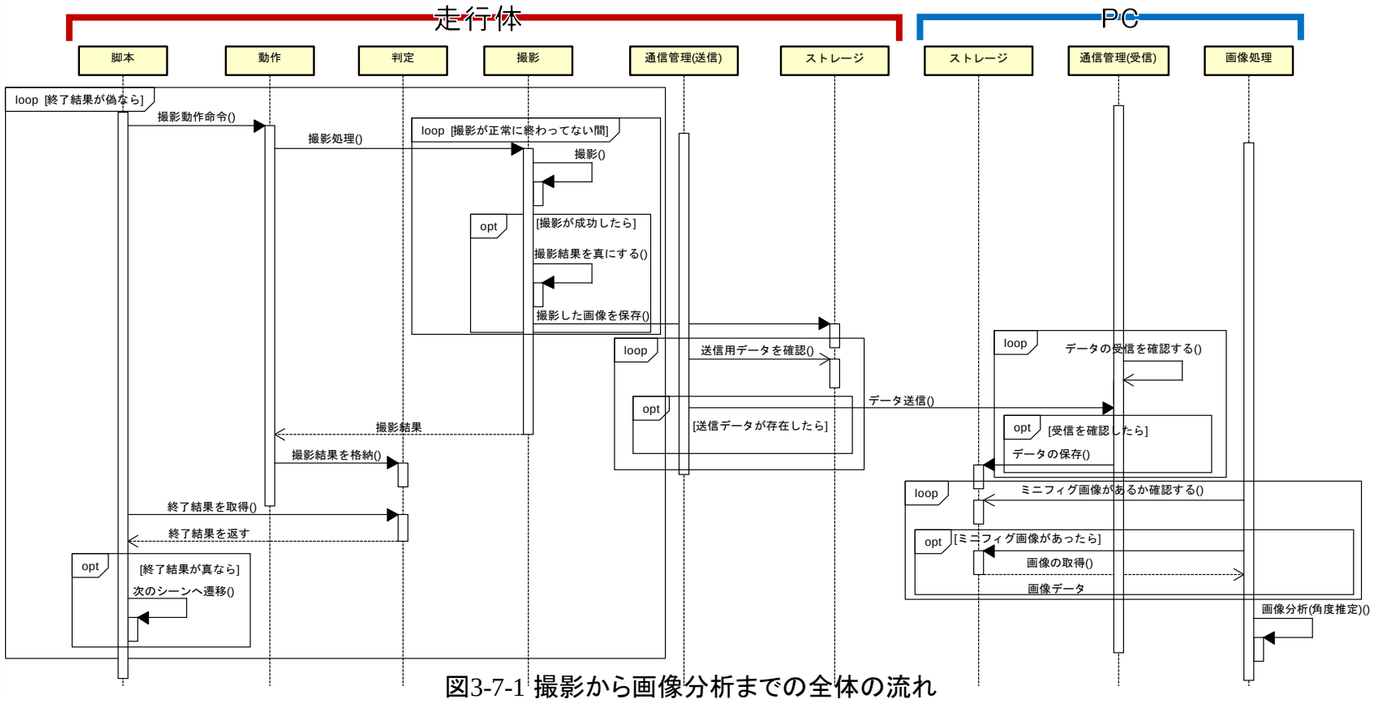


図3-7-1 撮影から画像分析までの全体の流れ

4.制御モデル

4.1 走行体制御の精度向上

[課題] ミニフィグ撮影や一部の走行の動作の過程において目印がない状況で直進で走行する必要があるが、
各モータの個体差や電圧など様々な要因で曲がって走行してしまう。

[対策1 – フィードバック制御とフィードフォワード制御]
各モータの出力には個体差があり、定常的な出力差が生じると考えられる。そこで、フィードフォワード制御を導入して、出力差を補正した。また、走行中に生じる角度の変動に対しては、PID制御を活用して瞬時に補正を実施することで、直進運動を維持する方法であるフィードバック制御を導入した。

[効果]
一部の蛇行を伴うものの、定常的な出力差によるカーブ状態を一定程度まで修正する成果を得ることができた。

[対策2 – フィードバック制御対象の変更]
フィードバック制御の対象が角度であった場合x軸の補正を行うことができないという制約が生じてしまっていた。これを克服すべく、制御の対象を自己位置推定による座標に変更した。
制御対象の値の計算方法を下記に示す。

x軸の移動距離 += Δ移動距離 × sin(π ÷ 180 × Δ車体角度)

[効果]
この方法により、図4-1-1に示すように誤差を±6mm以内に抑制できた。

[対策3 – 角度計測方法の変更]
ジャイロセンサーから得られる角度情報を基準としてPID制御を行った。ジャイロセンサーから得られる角度情報と車輪回転数から計算される角度情報をそれぞれ取得し、比較した結果、図4-1-3に示すようにジャイロセンサーから得られる角度情報では、誤差が次第に増加してしまったことが分かった。そこで、総車輪回転数から計算される角度情報を採用した。
車輪回転数から角度を推定する計算は以下の式で行った。

Δ車輪走行距離 = (π × タイヤの直径 ÷ 360) × (前車輪回転数 – 現車輪回転数)
Δ車体角度 = (360 ÷ (2π × 車体トレッド幅)) × Δ右車輪走行距離 - Δ左車輪走行距離

[効果]
この方法により、図4-1-2に示すようにジャイロセンサーから得られる角度情報よりも、総車輪回転数から計算される角度情報のほうが誤差が小さく、走行が安定したことが分かった。

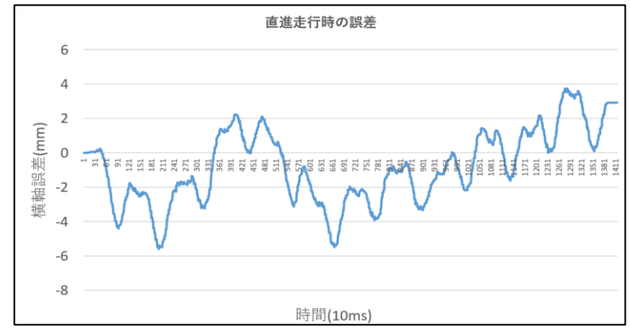


図4-1-1 直線走行時の誤差

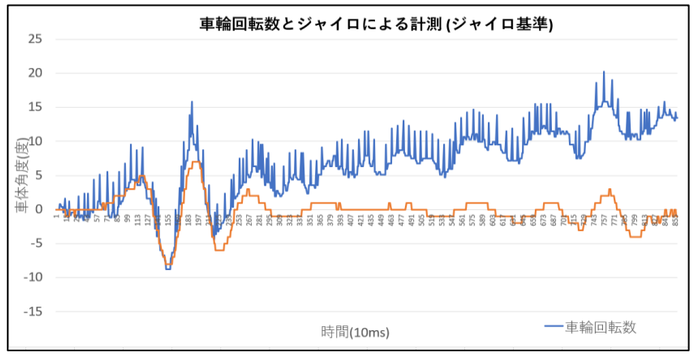


図4-1-3 ジャイロセンサーと角度推定処理の誤差量

走行体寸法
タイヤ半径 = 5
車輪円周 = 31.42
トレッド幅 = 12.6
車体幅 = 14
タイヤから最後部 = 14
アーム間距離 = 5.7

図4-1-2 走行体の寸法

4.2 ミニフィグの認識精度の向上

[課題]
ミニフィグ撮影によるベストショットを獲得するには、角度を高精度で推定する必要がある。

[対策1 – 学習ラベルの削減]
角度推定AIを作成するにあたり、一番の課題は学習ラベルの多さであった。角度は0~360までの数値があるが、画像ごとに手作業でラベル付けを行うのは事実上不可能である。そこで、学習ラベルの削減を試みた。まず、ベストショットの条件として「ミニフィグの右目、左目、口が個々に判別できる写真であること」という条件から、この条件を満たす顔の向きの範囲を調べた。その結果、図4-2-1のように±45度以内なら顔が判別できることが分かった。

[効果]
ラベルを合計8つに削減することが可能となった。

[対策2 – 転移学習のモデル選定]
今回の学習モデルは、2種類のミニフィグと8種類のラベルを学習させる必要がある。そのため、再学習時のコストを抑えるとともに、学習時間と精度のバランスを取ることが重要である。そこで、図4-2-2に示す通り、少ない学習時間で高い精度を達成できるEfficientNetV2モデルを選択した。

[対策3 – 動画からフレーム分割とデータ拡張]
写真を一枚ずつ撮影するのは非効率的であるため、動画から画像をフレームごとに生成する方法を採用した。さらに、画像の水増しと加工を図4-2-3に示すように実施することで、学習データの多様性を高め、汎化性能の向上に寄与することができる。

[効果]
約46000枚のデータを用いて学習を行った結果訓練データの精度は78.4%から93.0%へ、テストデータの精度は68.1%から86.7%へと向上した。
上記の対策により約27%の精度向上が達成できたと言える。

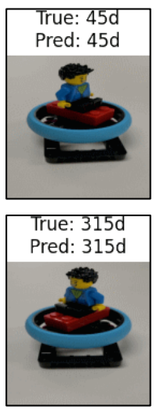


図4-2-1 角度別ミニフィグ画像

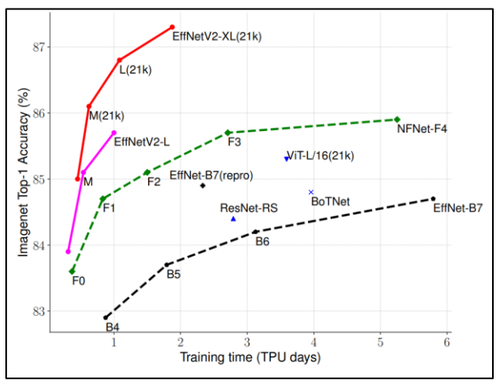


図4-2-2 学習モデルの比較


図4-2-3 データ拡張処理後の画像