



# プライマリークラス

## 性能モデル参考資料

Ver. 1.0.2

2016.5.7

ET ロボコン実行委員会 本部性能審査団

## 目次

目次.....	2
はじめに .....	3
I. ビギナー向けノウハウ .....	4
1. ロケットスタート.....	4
2. ローパスフィルタ.....	5
3. まいまい式.....	9
4. 光センサバンド補正 .....	10
5. PID 制御 .....	12
6. 灰色検知 .....	15
7. 段差検知 .....	17
8. 障害物検知.....	20
9. 位置推定 .....	21
10. スピン .....	23
11. 走行体傾斜.....	24
12. ハードウェア部品の個体差対策.....	25
II. EV3 走行体向けノウハウ.....	27
III. 2015 年プライマリークラス競技規約ダイジェスト .....	29
1. コース全体.....	29
2. フィギュアL .....	30
3. ルックアップゲート .....	31
あとがき .....	32
改訂履歴 .....	33

## はじめに

本資料はETロボコンのプライマリークラス参加者に走行体の性能面に関する参考情報（性能モデル＋補足）を提供し、設計モデルの作成や走行プログラムの生成に役立ててもらうことを意図している。紹介する性能モデルは、ビギナーが参考にすることを前提に、2015年チャンピオンシップ大会に出場したチームのモデル（主にプライマリークラスのモデル）から出来るだけ分かり易い表現を選別して引用した。2015年の競技規約に則した性能モデルとなっているため、参考にするにあたっては現状の競技規約にも適用可能か否かについて、各自で判断する必要がある。

本資料の構成は大きく三つに分かれており、第一部（Ⅰ. ビギナー向けノウハウ）は初めてETロボコンに参加するビギナーが知っておくと役立つ情報が、第二部（Ⅱ. EV3 走行体向けノウハウ）は2016年から全クラスで採用されることとなったEV3に関する情報が、第三部（2015年プライマリークラス競技規約ダイジェスト）は2015年プライマリークラスのコースの特徴に関する情報が提供されている。

プライマリークラスのモデル審査においては、性能モデルの記載は対象外となるため、実際に提出するモデルには性能モデルを記載する必要はないが、モデル審査に提出する設計モデルを検討するに当たり、この資料に記載されている幾つかの情報が役立つかも知れない。

掲載資料に関してさらなる詳細な解説が必要な場合は、各地区の実行委員会が開催する技術教育の機会を活用したり、インターネット上に公開されている技術情報を自ら検索したり、参加チーム間で積極的に技術交流をしたり、モデル出典チームの関係者に直接聞いたりして補うなど、自己研鑽に励んでほしい。

## I. ビギナー向けノウハウ

### 1. ロケットスタート


◆出典チーム：delias（東京地区）

II
ロケットスタート

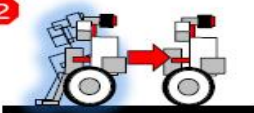
要求	スタート時、走行体を後退させずに短時間でスタートさせる。
課題	スタート時、走行体が後ろに傾いていると倒立振子制御プログラムによりバランスを取ろうとするため、走行体が後退しスタートするのに時間がかかる。
対策	尻尾を付けて静止してしている状態から倒立走行移行時に、走行体の姿勢を直立させ、前方向に進行させる。(下図参照)
効果	後退せずにスムーズなスタートができる。

※スタートとは、尻尾を下して静止している状態から二輪倒立走行状態に移行することである。

**1: 尻尾をおろして静止**



**2: 尻尾をさらにおろし、走行体を直立させ、前方向へ進行する**



補足) 倒立振子制御プログラム(バランス)の特性を利用し、走行体の前方に動く力をスタートダッシュに役立てようとする工夫。バランスは、走行体が後方に傾くと後退し前方に傾くと前進する特性がある。

図 1

◆出典チーム：計算機くれ（中四国地区）

#### 5.1 高速発進

スタート後すぐ車体を前傾させることで速いスタートを実現する。

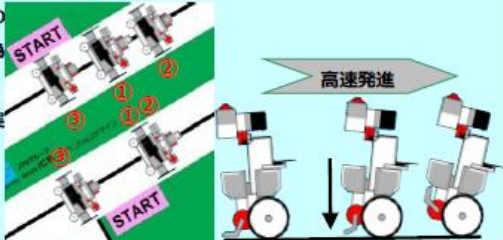
問題点	発進時に少し後退してから発進するため、発進に多少の時間がかかる。
対策	発進時に後退しないようにする。
実現	<p>発進時に尻尾を用いて車体を押し上げ、前傾させることで素早いスタートを可能とする。尻尾で倒立している状態（後傾姿勢）から高速走行（前傾姿勢）に素早く移行でき、また従来のやり方に比べて一度後ろに下がって倒立状態になってから高速走行に移行するのではなく、その場で直接前傾姿勢をとる為一度後ろに下がる必要が無く、素早いスタートを実現できる。</p> 

図 2

## 2. ローパスフィルタ

◆出典チーム：ぼちぼちがんばる（関西地区）

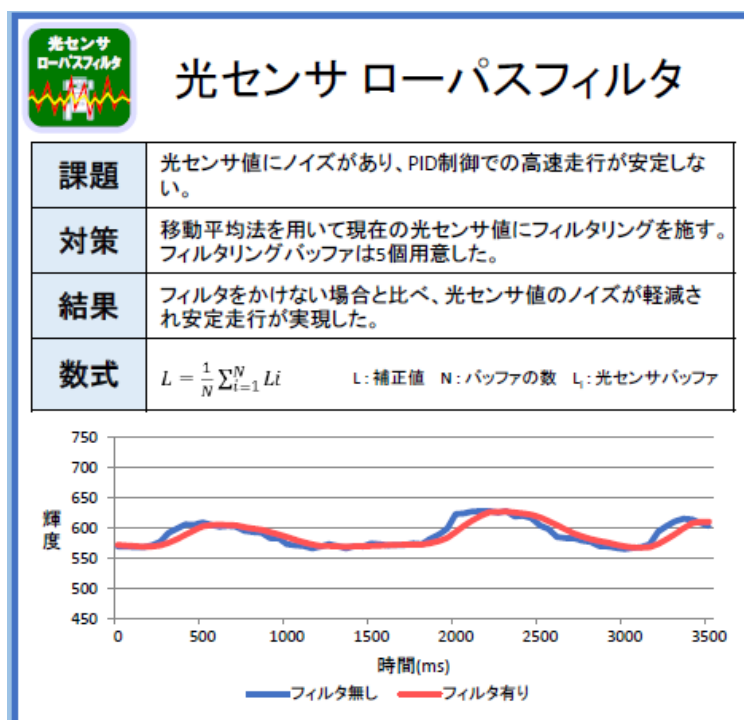


図 3

◆出典チーム：HELIOS（東海地区）

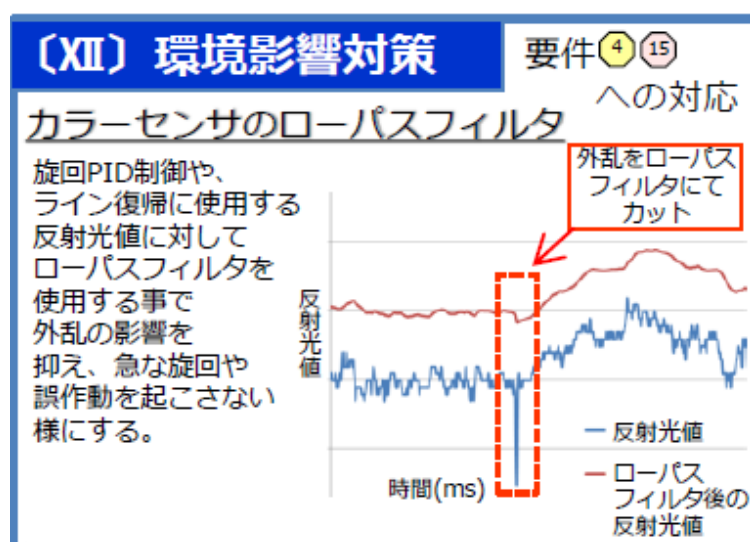


図 4

補足) 光センサのサンプリングデータを平均化処理することで光センサの変動が滑らかになるため、単発データの急峻な変動による走行体の挙動を抑えて滑らかな走行を実現しようとするもの。グラフ縦軸の輝度値は NXT のものであり、EV3 では異なる数値を示す。

補足) グラフの青ラインは反射光のサンプリングデータ、赤ラインは反射光のフィルタ処理後データを示しており、青ラインの急峻な変化が赤ラインでは平滑されていることがわかる。

◆出典チーム：Young Master（東海地区）

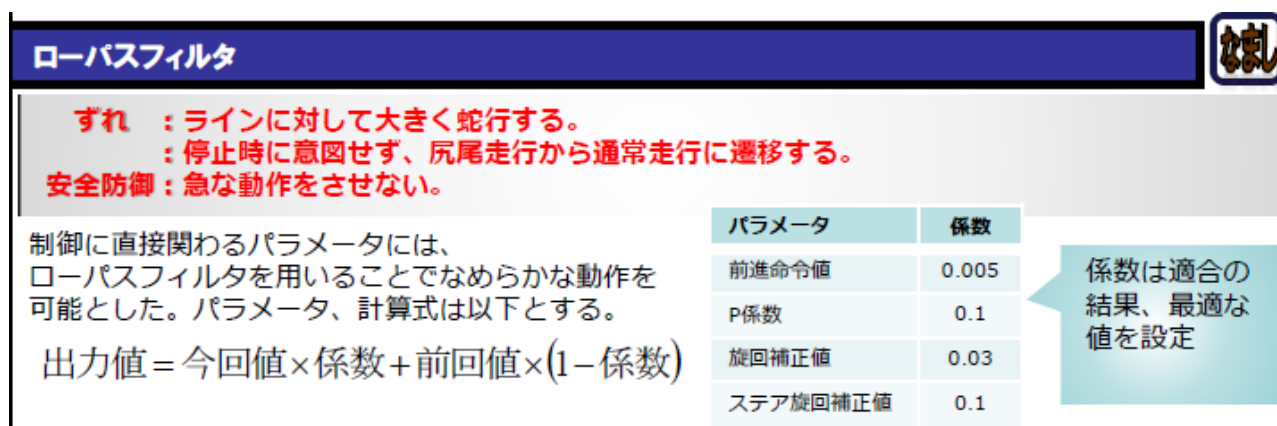


図 5

補足) 係数は重み付けである。前回値と今回値の配分を重み付けにて調整する。

◆出典チーム：AC.ひよこ sonic（南関東地区）

## 光センサ値に混入する外要因対策

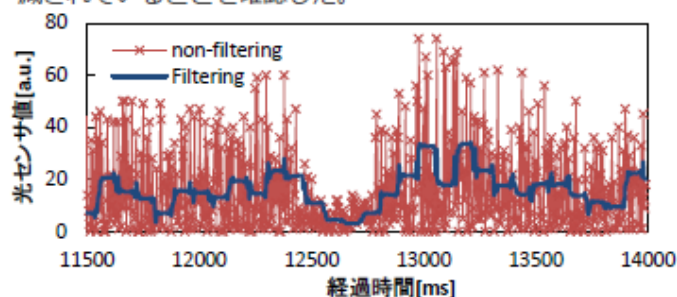
## 光センサ値の正規化

コース各所での光センサ値を事前取得し（白基準値・黒基準値）、光センサ値をコース各所で正規化することで、ライントレース制御に用いる光センサ値から場所ごとの光むら依存を低減する。

正規化光センサ値 =  
 (光センサ値 - 黒基準値) / (白基準値 - 黒基準値)

## ローパスフィルタ

外乱光の蛍光灯成分を除去するためにFIRフィルタを実装した。実験結果：段数11、遮断周波数45Hzで蛍光灯成分が低減されていることを確認した。



光センサ部のブロック線図

図 6

補足) ローパスフィルタとして代表的なFIRデジタルフィルタを採用している。FIR デジタルフィルタとは、「Finite Impulse Response」の略で、簡単なローパスフィルタとして用いられる単純移動平均法ではなく、移動平均の各要素に重み付けのパラメータを追加したローパスフィルタ。図 6 の段数が遅延ブロックの要素数を指している。FIR デジタルフィルタには、ローパスフィルタ、ハイパスフィルタなどのバリエーションがある。



## ◆出典チーム：ごばりき（東京地区）

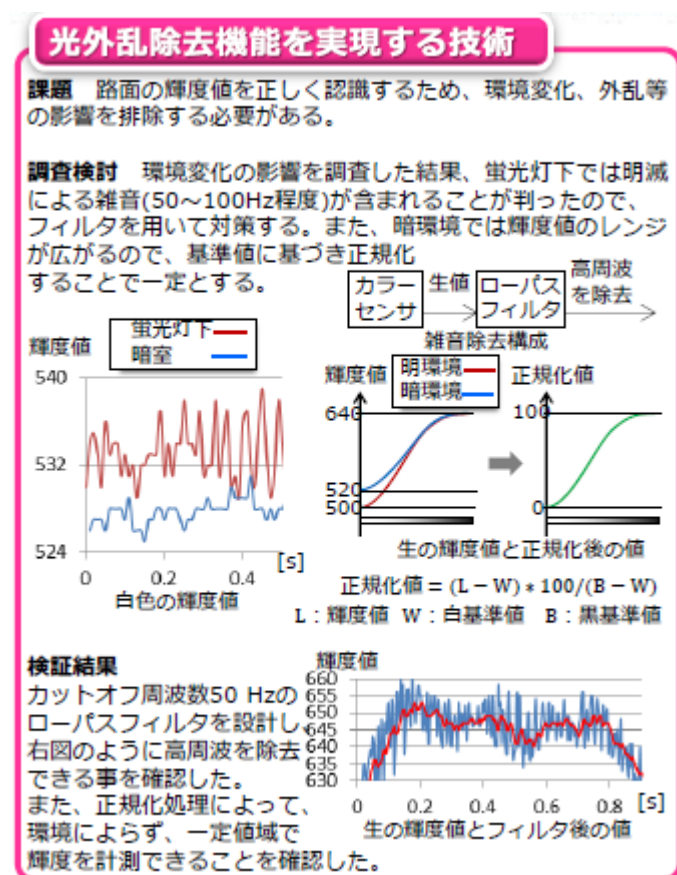


図 7

## ◆出典チーム：ヒカリバクシンオー（南関東地区）

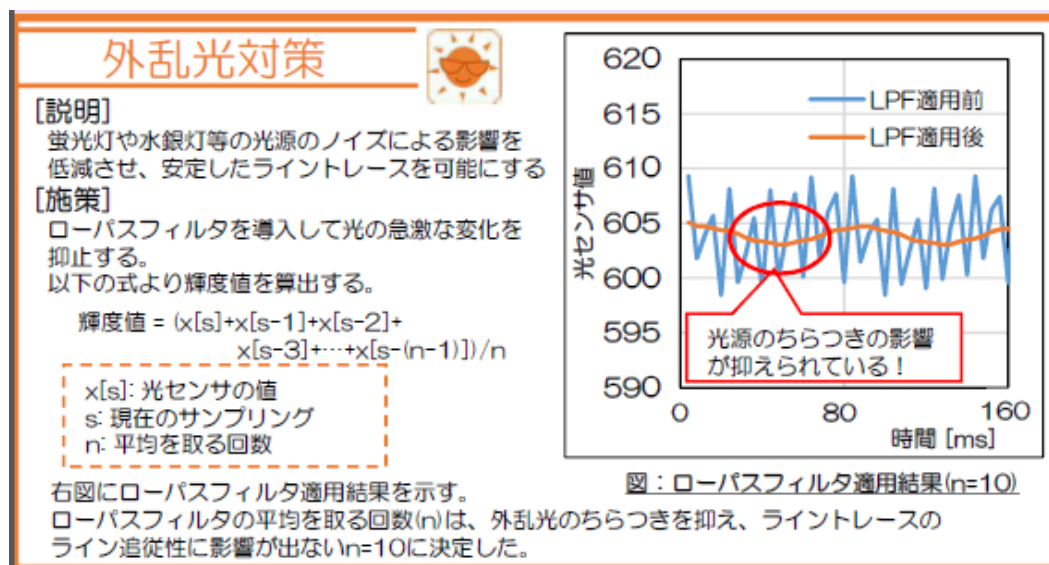


図 8

補足) 10 段の単純移動平均法によるローパスフィルタの実装例である。何段の移動平均にするとよいかは走行体の走行スピードとライン追従性を踏まえて決定するとよい。

◆出典チーム：FUTSAL（北陸地区）

## ローパスフィルタ

- ・ ジャイロセンサや光センサの信号には高周波域に雑音があり、誤動作の原因となる。
- ・ センサ信号に対してスペクトラム解析しローパスフィルタのカットオフ周波数を決定する。
- ・ ローパスフィルタ処理によって高周波域の雑音を低減することができる。
- ・ 一次遅れのデジタルフィルタの数式： $y(t)=\alpha y(t-1)+(1-\alpha)x(t-1)$   
ここで、 $x$  はセンサからの生データ、 $y$  はフィルタの出力であり、 $\alpha(0<\alpha<1)$  はカットオフ周波数により算出される定数である。
- ・ 適切なフィルタリング処理により、走行体の動きを正確に捉えることが可能となった。仕様未確定エリアⅡなどの難所において走行シナリオ切替の誤作動をなくすることができる。

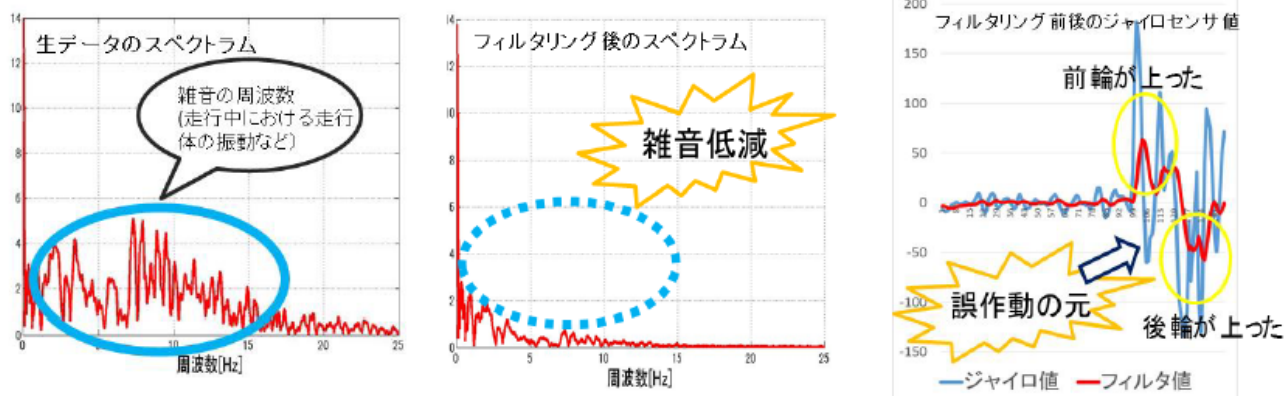


図 9

補足) 周波数のスペクトラム解析により、除去したい周波数帯を識別してローパスフィルタのカットオフ周波数を決定している。



## 3. まいまい式

◆出典チーム：delias（東京地区）

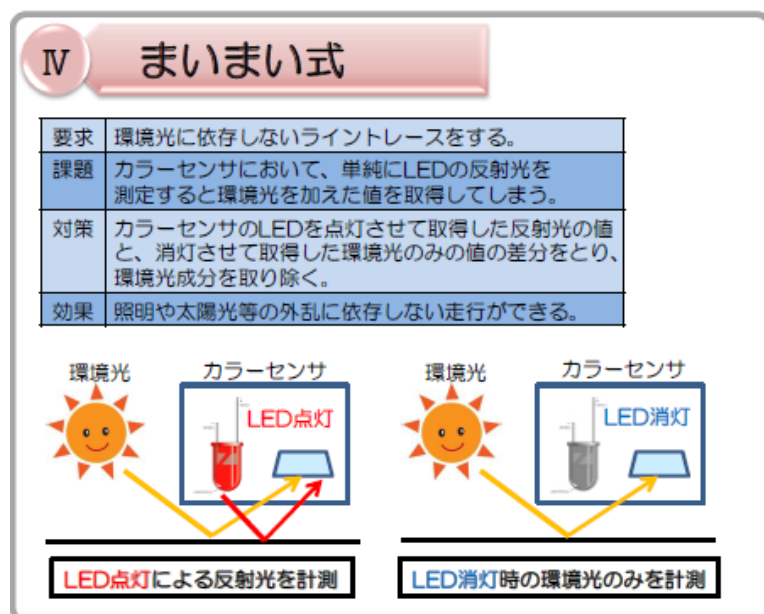
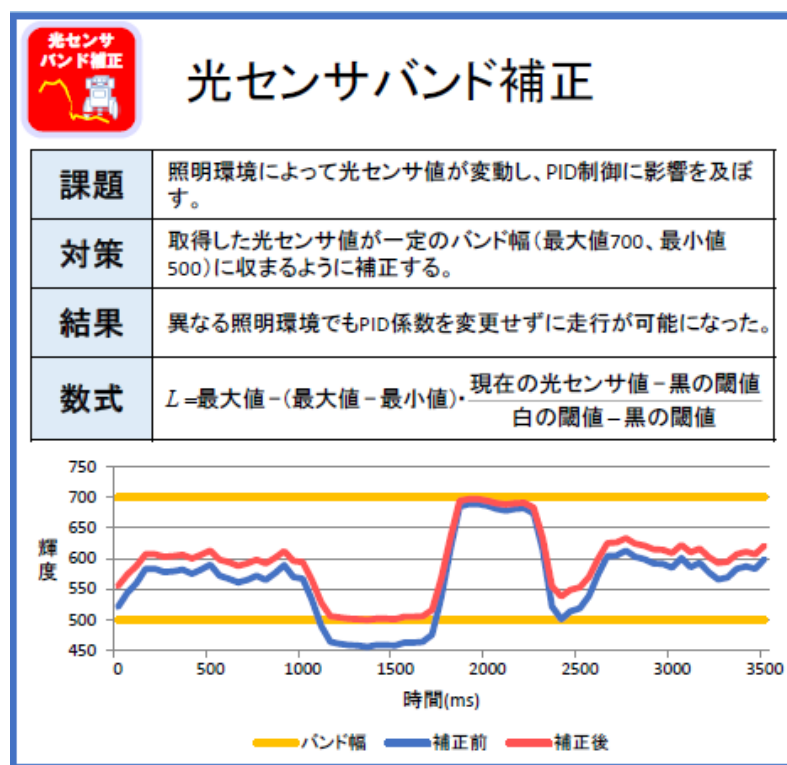


図 10

補足) まいまい式とは、LED 点灯時の反射光輝度 (LED 光+環境光) と LED 消灯時の反射光輝度 (環境光) の差分を真値として利用することで、環境光 (外乱光) の反射光輝度をキャンセルする方法。LED 点灯時と LED 消灯時の環境光が異なる状況では、全ての外乱光をキャンセルできないので留意すること。EV3のカラーセンサを反射モードで使用する場合は、まいまい式に似た処理が予め入っている (カラーセンサに内蔵されたソフトウェアが外乱光をキャンセルする) ようなので、EV3 のマニュアルを参照しながら実現手段を検討・確認してほしい。

## 4. 光センサバンド補正

◆出典チーム：ぼちぼちがんばる（関西地区）



補足) グラフ縦軸の輝度値は NXT のものであり、EV3 では異なる数値を示す。

図 11

◆出典チーム：Young Masters（東海地区）

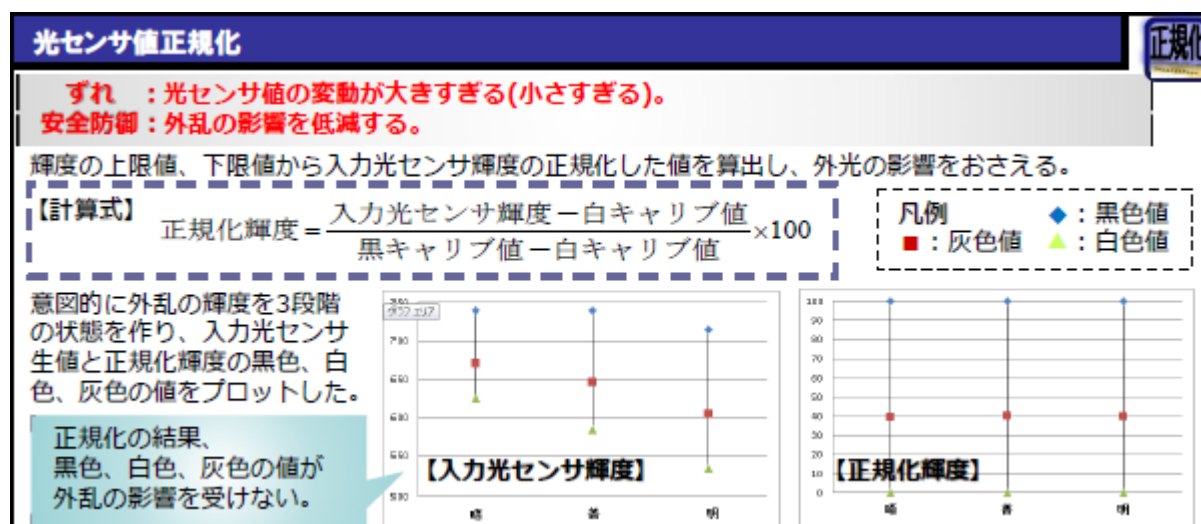
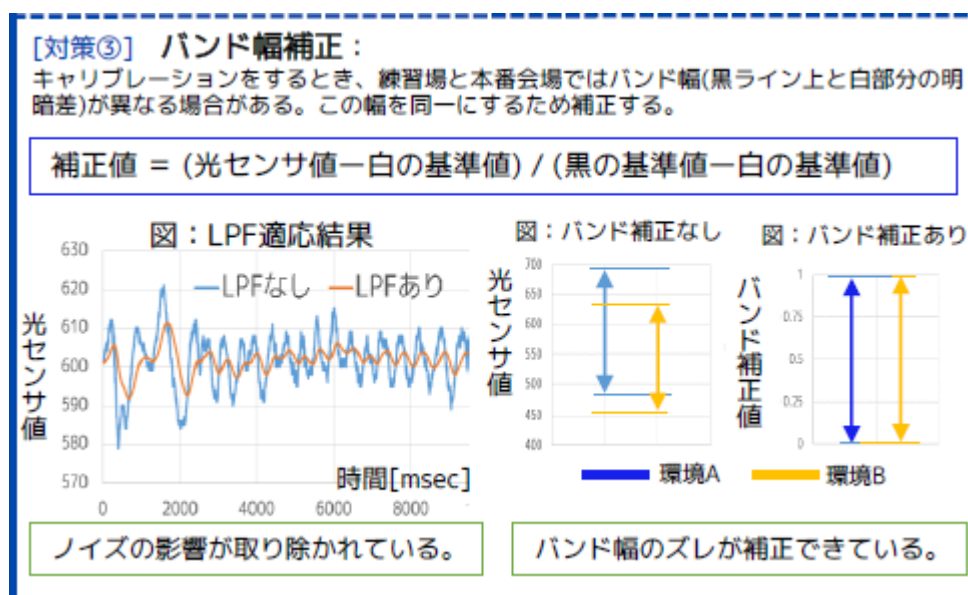


図 12

補足) 環境光が明るい時と暗い時では、黒色と白色の輝度幅が異なる。したがって、環境光の明暗に影響を受けずに、黒色・灰色・白色を判定させるため、輝度の上下限值から正規化処理をおこなって正規化輝度データを生成する。

グラフ縦軸の輝度値は NXT のものであり、EV3 では異なる数値を示す。

◆出典チーム：NiASET（九州北地区）



補足) グラフ縦軸の輝度値は NXT のものであり、EV3 では異なる数値を示す。

図 13

## 5. PID 制御

◆出典チーム：ぼちぼちがんばる（関西地区）

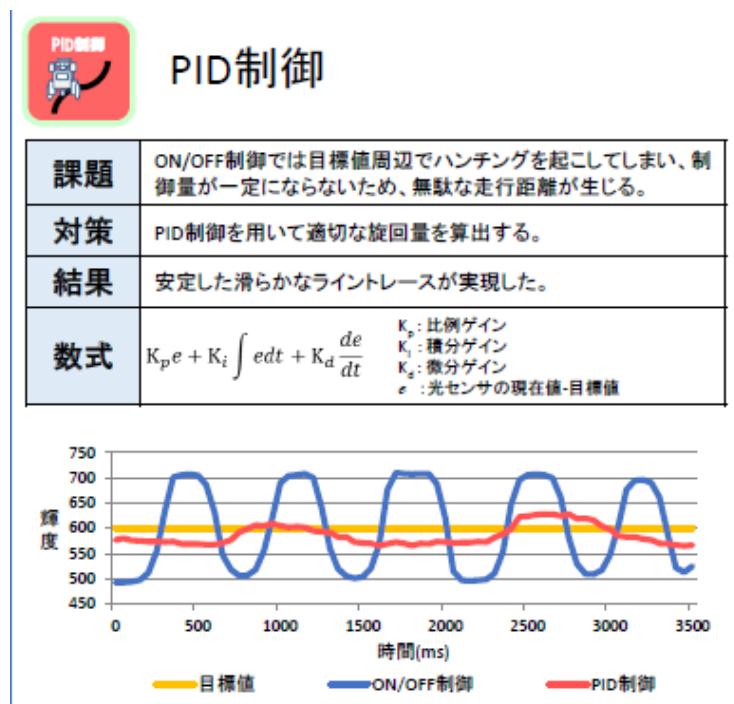


図 14

補足) PID 制御はフィードバック制御としてもっとも良く使われる制御方式である。PID とは P:Proportional (比例)、I: Integral (積分)、D: Differential (微分) の3つの 組み合わせで、目標に対してスムーズな制御を可能とするもの。単純な On/Off 制御は、制御操作量が 100% (On) と 0% (Off) であることから、0% と 100%の間の行き来を繰り返すこととなる。したがって、操作量の変化が大きくなり過ぎてしまい、目標値の上下をいつまでも繰り返す(振動すること)となり、なかなか目標値に収束しない。

グラフ縦軸の輝度値は NXT のものであり、EV3 では異なる数値を示す。

◆出典チーム：からっ風産学隊 2015（北関東地区）



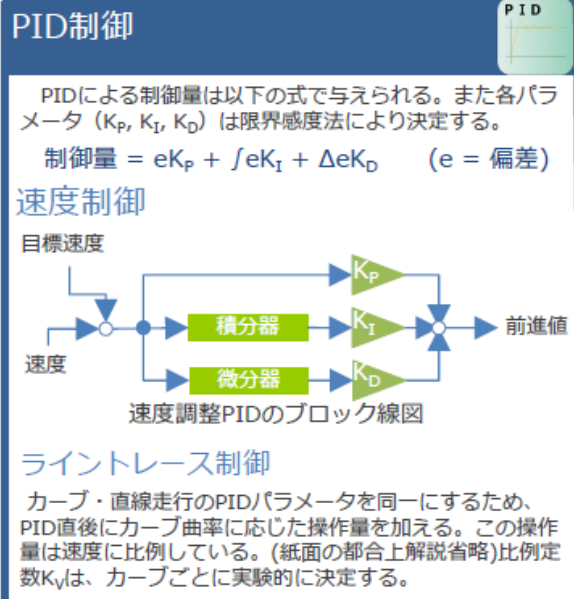
図 15

補足) 目標値に収束させるためのゲイン量を算出するにあたり、P 制御、I 制御、D 制御の3つを組み合わせることで、P 制御、PD 制御、PID 制御などの制御バリエーションが可能となるため、課題解決に適したゲイン量が算出できる制御方法を選択する。

PID 制御に必要な比例ゲイン、積分ゲイン、微分ゲインを決める代表的な方法に「ステップ応答法」と「限界感度法」がある。

◆出典チーム：AC,ひよこ sonic（南関東地区）

## 5.2 走行中の要素技術



補足) スタートからゴールまでをコース形状に応じて幾つかの区間に分割し、それぞれの区間に最適な PID 制御ゲインを設定することもあるが、スタートからゴールまで同一の PID 制御ゲインでまかなうこともある。図 16 は後者であるが、直線とカーブの速度に応じた旋回値を左右モータに指示する工夫をしている。

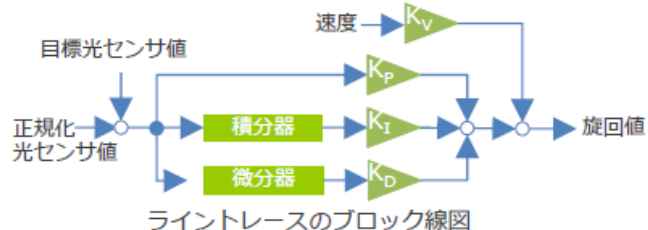
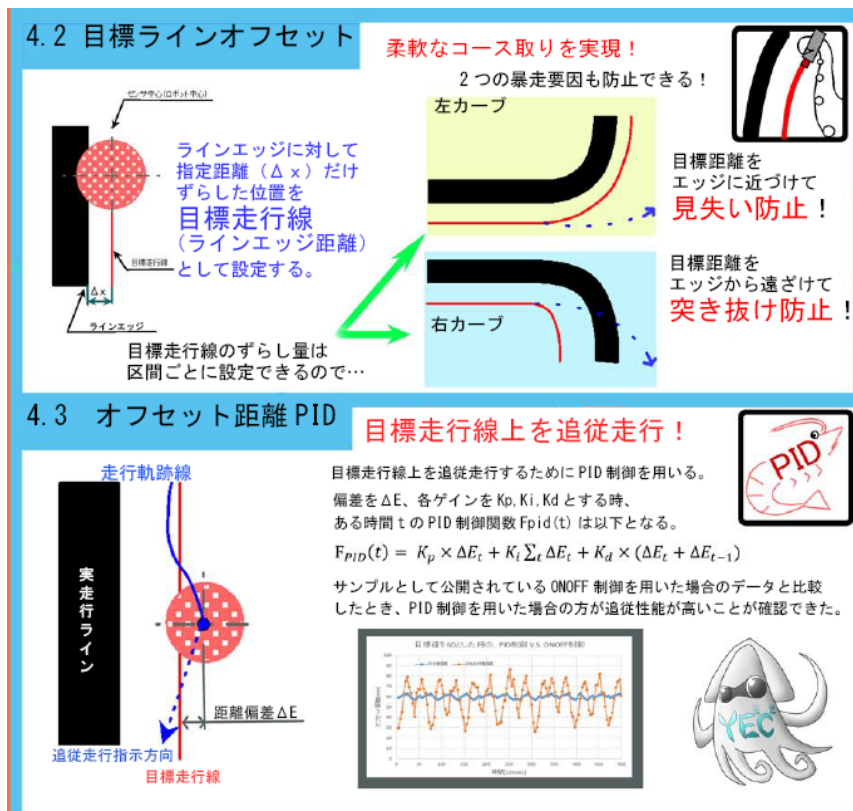


図 16

◆出典チーム：いか（東海地区）



補足) PID 制御は目標との差を収束させる技術であるため、何を目標とするかがポイントとなる。実走行ライン (黒色) と白色の境界ではなく、目標走行線 (目標においてコースを逸脱しないよう) に工夫している。

図 17



## ◆MOTION &amp; CONTROL DP（北関東地区）

## (1) 積分数の検討

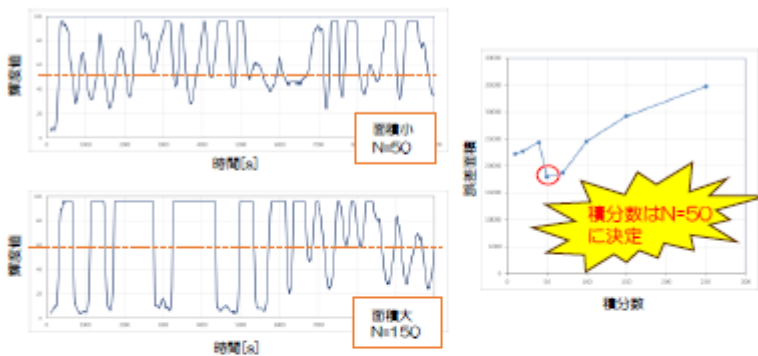


図10. 積分数と誤差面積の関係

挙動を安定させるために、PID制御における積分量の係数  $K_i$  を検討する。 $K_i$  の検討に際して、過去値の積分をし続けることによるオーバーフローを抑制するため、まずは積分に陥る過去値の数(積分数:  $N$ )を検討した。

図10にその結果を示す。最適値を見つけるため、走行中の偏差の絶対値の合計(誤差面積:  $S = \sum |e(t)|$ )を評価に用いた。この値が大のときは挙動が不安定となり、小のときは安定となる。実験から  $N = 50$  が最適であると判断した。

補足) PID 制御のゲイン値のうち、積分ゲインと微分ゲインを実走行によるエラー&トライによって決めるアプローチ例を図 18 および図 19 に示す。

## 図 18

(2) 積分係数  $K_i$  の検討

図11. 積分係数と誤差面積の関係

積分数  $N = 50$  で、積分係数  $K_i$  の値を検討した。ここでも誤差面積を用いて、最適値を検討した。図11から、 $K_i = 0.2$  が誤差面積最小であるが、 $K_i = 0.3$  のほうが収束傾向があり、誤差面積的にも大差がないので、 $K_i = 0.3$  に決定した。

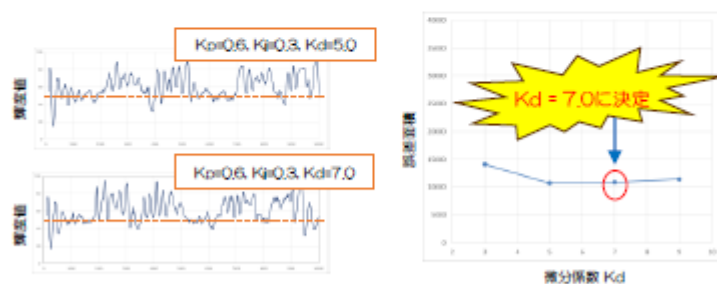
(3) 微分係数  $K_d$  の検討

図12. 微分係数と誤差面積の関係

急峻な輝度値変化に対する応答性を向上させるため、微分係数  $K_d$  の検討する。誤差面積の結果と応答性を考慮して、 $K_d = 7.0$  に決定した。

## 図 19



## 6. 灰色検知

◆出典チーム：オートロボットみとちゃん（東京地区）

## 3-1. 灰色マーカ検知

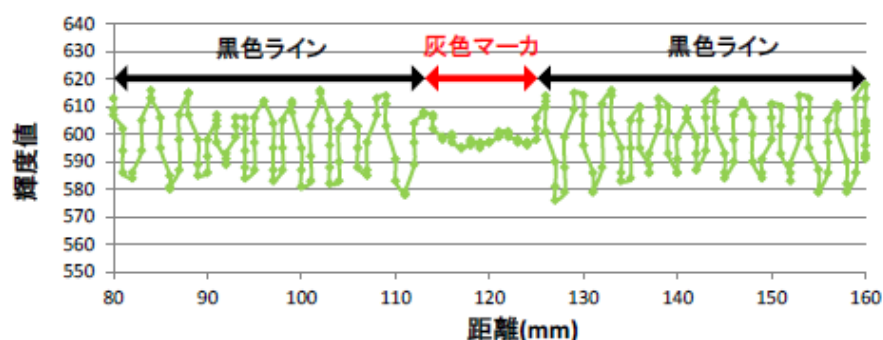
## 課題

灰色マーカ輝度値と黒色ライントレース時の白黒中間輝度値（目標輝度値）が類似しており、灰色マーカを検知できない。

輝度値取得場所	白黒中間	灰色マーカ
輝度値平均	603	616

## 対策

灰色マーカに突入する直前に、黒色ラインに振動的に追従するように比例ゲインを大きく設定する。これにより、黒色ライン通過時と灰色マーカ通過時のセンサ値の振幅の違いが顕著に表れるため、その変化量を検知して灰色マーカを検知する。



【黒色ライン→灰色マーカ→黒色ラインを倒立黒色ライントレースで走行した際の輝度値】

## 対策後の成果

輝度値の変化量を用いることで、灰色マーカ検知の精度が向上した。より確実な攻略が期待できる。

※図は、黒色ライントレースにより灰色マーカを通過できることも同時に示しているため、灰色マーカ通過に関する記述は省略する。

図 20

補足) 図 20 は灰色マーカに突入する前に比例ゲインを変更する戦略であるため、何らかの方法で灰色マーカに近づいていることを認識する必要がある。例えば、「9 位置推定」を使って灰色マーカへの接近を認識することも選択肢のひとつとなり得る。

## ◆出典チーム：薊レーシングチーム'15（東京地区）

#### 4.5 灰色検知

## 概要

灰色ラインの手前で走行体を黒寄りに走行させ、灰色基準値を一定数以上連続で検知したとき灰色検知とする  
検知後は灰色ラインの距離の間、輝度値を補正して走行する

### 效果

- ◆灰色ラインを白色と誤認識せず通過！  
◆難所攻略の開始位置を灰色ラインで判断可能！

## 黒寄り走行 する理由

走行中は黒色と白色の平均値を  
ライトレースの閾値にしているため、  
通常の走行する時も灰色と検知してしまう

区間切替により灰色ラインの手前で  
黒寄りに走行させ、黒色ライン時の閾値を灰色  
基準値より高くする

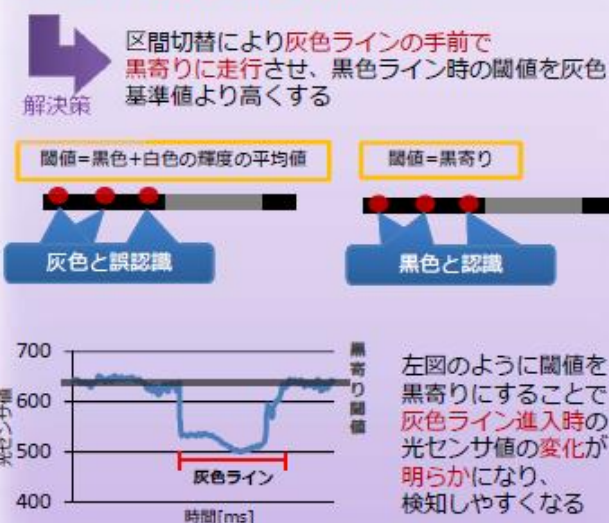
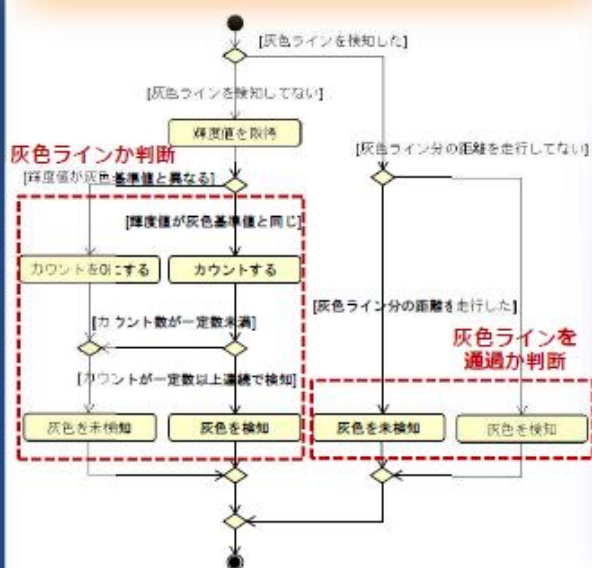


图 21

補足) 図 21 図 20 は灰色マーカーに突入する前に、位置推定を使った区間切替えによって黒色寄りにライントレースの閾値を移動して走行させる戦略である。



## ◆FCTawashi (東京地区)

## 3.2 進入ライン(段差)検知

## 課題

ステージ上りを実行するためには、進入ラインに接触している必要がある。

## 検討

一定速度でライトレース走行し、フィギュアLの進入ライン(段差)にぶつかった場合、移動距離が減少する事を利用する。

## 実現方法

検知走行中に走行距離計算を行い、ある単位時間あたりの移動距離を求める。段差に衝突する前であれば一定速度で移動しているので、単位時間あたりに進める距離は一定である。段差に衝突した場合、それ以上進めなくなり単位時間あたりの移動量は減少する。

単位時間での移動距離の履歴を複数持っておき、差分が閾値を超えた(移動出来なくなった)事で、進入ラインの検知とする。

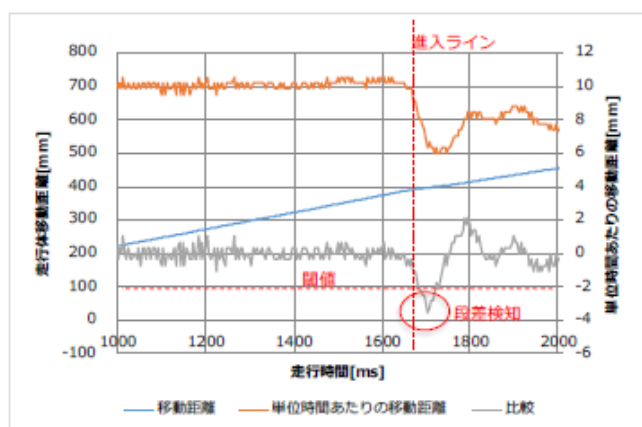
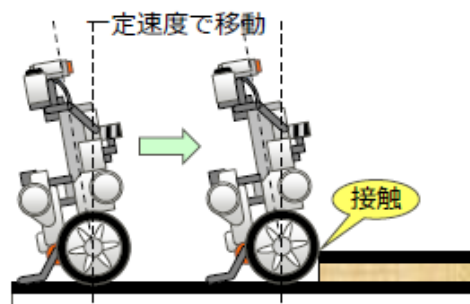


図 24

補足) ロータリーエンコーダからのパルス数をカウントし、単位時間当たりの走行距離を計測することで段差に衝突したことを検知する。

## ◆出典チーム: NiASET (九州北地区)

## 5-9. 段差・障害物検知

**[目的]** 段差・障害物を検知する。

**[方法]** ジャイロセンサの値が基準値を超えたとき、前輪が段差に衝突したと判断することで段差を検知することができる。



図 25

補足) 図 25 はアドバンストクラス用の三輪走行体であるため、前輪の段差検知を示しているが、プライマリークラスの二輪走行体にも応用できる。

◆出典チーム：ヒカリバクシンオー（南関東地区）

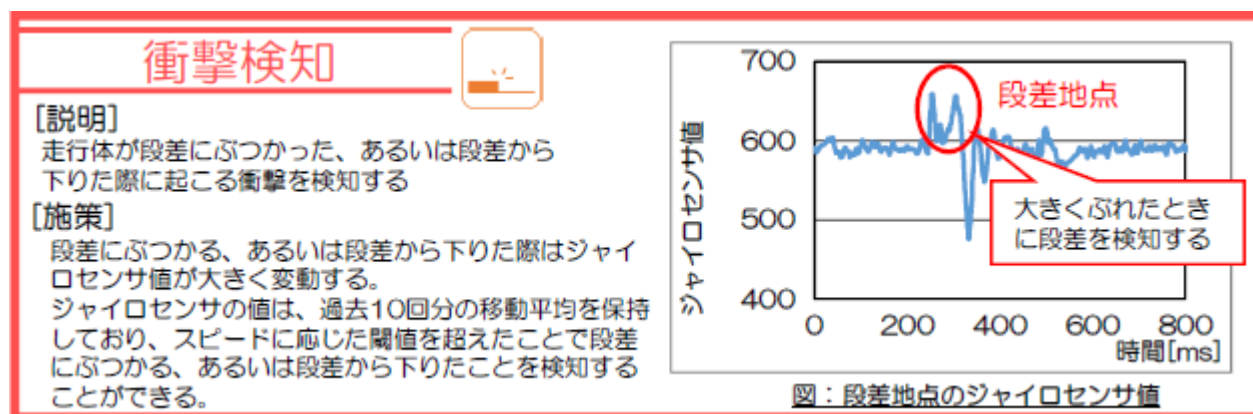


図 26

補足) ジャイロセンサ値による段差の登段や降段の判定にセンサ値の単純移動平均法によるローパスフィルタを使い誤判定のリスクを下げている。



## 8. 障害物検知

◆出典チーム：オートロボットみとちゃん（東京地区）

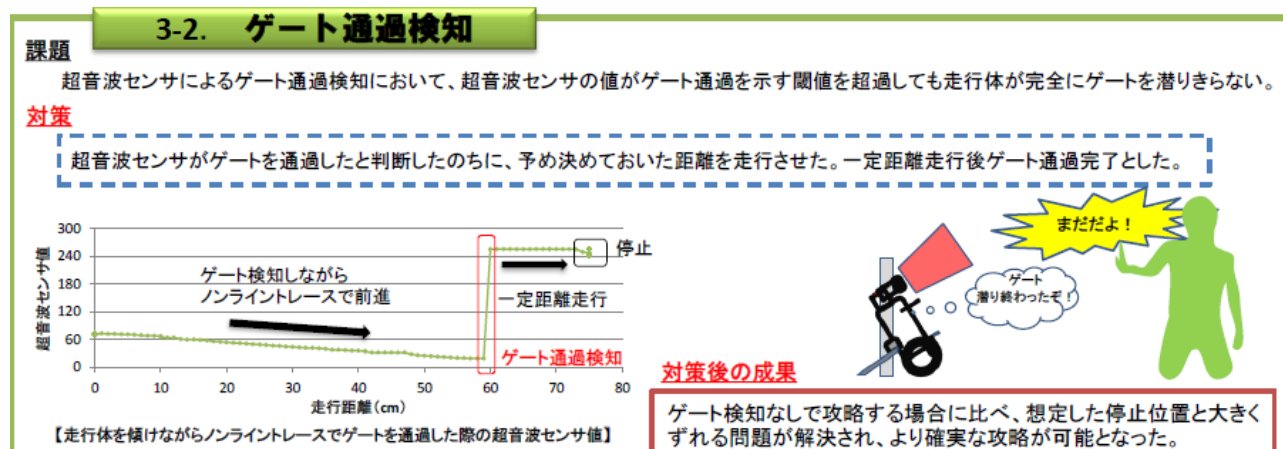


図 27

補足) 超音波センサを使ってルックアップゲートを検知し、走行体がゲートを通過したか否かを判定している。

◆出典チーム：ごばりき（東京地区）

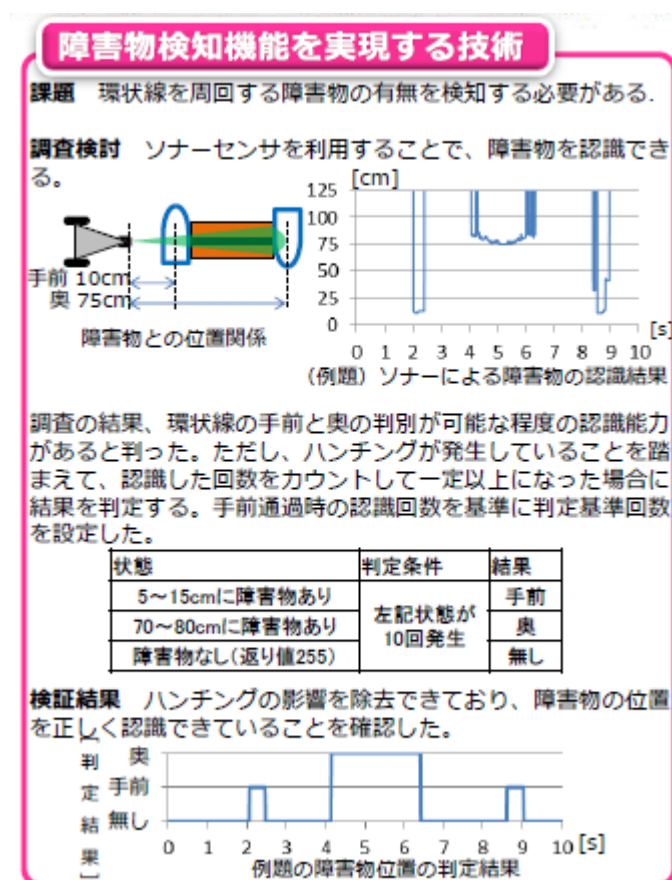


図 28

補足) ソナー（超音波センサ）による障害物の検知性能（障害物との検知距離や上下左右の検知幅）を調べる。

ソナーにも個体差があるため、複数のソナーを有している場合には、自チームの走行戦略に適したソナー特性を有する個体を使用する。

図 28 はアドバンスクラスの事例であるが、ソナーを使用するときのアプローチとして参考にしてほしい。



## 9. 位置推定

◆出典チーム：AC.ひよこ sonic（南関東地区）

## 走行距離/方向検知

車輪間隔  $w$ ・車輪直径  $d$ ・左右エンコーダ値  $E_l, E_r$  から計算する。  
 走行距離  $L$  は、走行体の中心が進んだ距離とする。式を以下に示す。  
 旋回角度  $\theta_c$  が、正ならば左向き、負ならば右向きと検知する。

■ 走行距離  $L = \pi \times d \times \frac{(E_l + E_r)}{2}$

■ 旋回角度  $\theta_c = \pi \times d \times \frac{(E_r - E_l)}{w}$

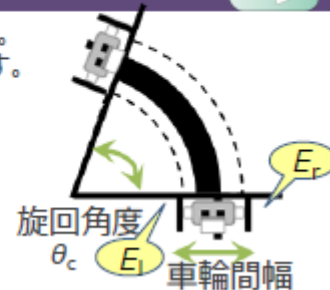


図 29

補足) コース全体の座標と左右モータのエンコーダ値を使って、走行体がコース上のどこを走っているのかを推定することで、走行方法をコース形状によって切り替えることが可能となる。また、コースからの逸脱も判定できるため、通常走行時と逸脱走行時の走行方法を用意して対処することも可能となる。

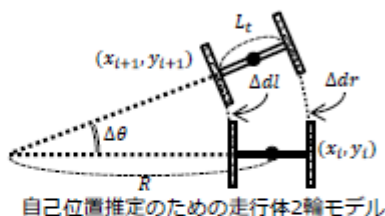
◆出典チーム：ごぼりき（東京地区）

## 自己位置推定機能を実現する技術

課題 走行戦略の切り替えや難所の攻略のため、走行体の走行距離、現在位置、ヨー角を推定する必要がある。

調査検討 走行体の位置変化を下式で曲線運動として近似し、左右輪の回転量に基づいて逐次積算することで、自己位置推定を行うことができる。

$\Delta d_l, \Delta d_r$  : 回転量  
 $\theta$  : 走行体ヨー角  
 $L_t$  : トレッド幅  
 $x_t$  : X座標位置  
 $y_t$  : Y座標位置  
 $\rho$  : 曲率半径  
 $\Delta \theta$  :  $\theta$  の変化量  
 $d_t$  : 移動距離



$$\begin{aligned} d_{t+1} &= d_t + (\Delta d_l + \Delta d_r) / 2 \\ \theta_{t+1} &= \theta_t + (\Delta d_l - \Delta d_r) / L_t \\ x_{t+1} &= x_t + 2\rho \sin(\Delta \theta / 2) \cos(\theta_t + \Delta \theta / 2) \\ y_{t+1} &= y_t + 2\rho \sin(\Delta \theta / 2) \sin(\theta_t + \Delta \theta / 2) \end{aligned}$$

検証結果 この方法では計算誤差が蓄積される。対策として段差を検知した際に自己位置をリセットすることとした。また、初期設置角によってヨー角推定値にオフセットが発生する事が判った。対策として、ストレート走行時のヨー角推定値を基に最小二乗近似した直線からオフセットを推定することとした。補正をかけた自己位置推定値は真値との最大ズレは走行距離  $\pm 2[\text{cm}]$ 、走行体ヨー角  $\pm 2[\text{deg}]$ 、となる事を確認した。

補足) 計算誤差が蓄積されるため、誤差をキャンセルするための幾つかのポイントコース全体の中に設定し、正確な位置推定を実現する工夫をしている。

図 30

◆出典チーム：Champagne Fight（北海道地区）

## 自己位置推定

基本技術

## 技術概要

左右のモータエンコード値を利用して、一般的に知られているオドメトリ手法を用い、走行体の座標(X,Y)、移動距離L、方位 $\theta$ を算出する。

## 詳細：

$$\Delta T = 2\pi r (G \Delta E / 360) \quad \text{※ 左右のモータ各々で計算} (\Delta T_r, \Delta T_l)$$

$$\Delta D = (\Delta T_l + \Delta T_r) / 2$$

$$\Delta \omega = (\Delta T_r - \Delta T_l) / d$$

$$X1 = X0 + (\Delta D \cos(\theta_0 + (\Delta \omega / 2)))$$

$$Y1 = Y0 + (\Delta D \sin(\theta_0 + (\Delta \omega / 2)))$$

$$L1 = L0 + \Delta D$$

$$\theta_1 = \theta_0 + \Delta \omega$$

G:ギア比 1 : G

$\Delta E$ :モータエンコード値の変化量

$\Delta T$ :後輪の移動距離

$\theta_1$ :現在の走行体の方位

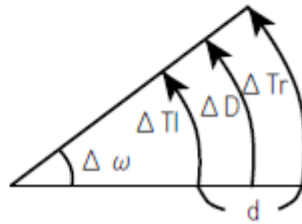
$\theta_0$ :前回の走行体の方位

L1:現在の移動距離

L0:前回の移動距離

(X1,Y1):現在の走行体の座標

(X0,Y0):前回の走行体の座標

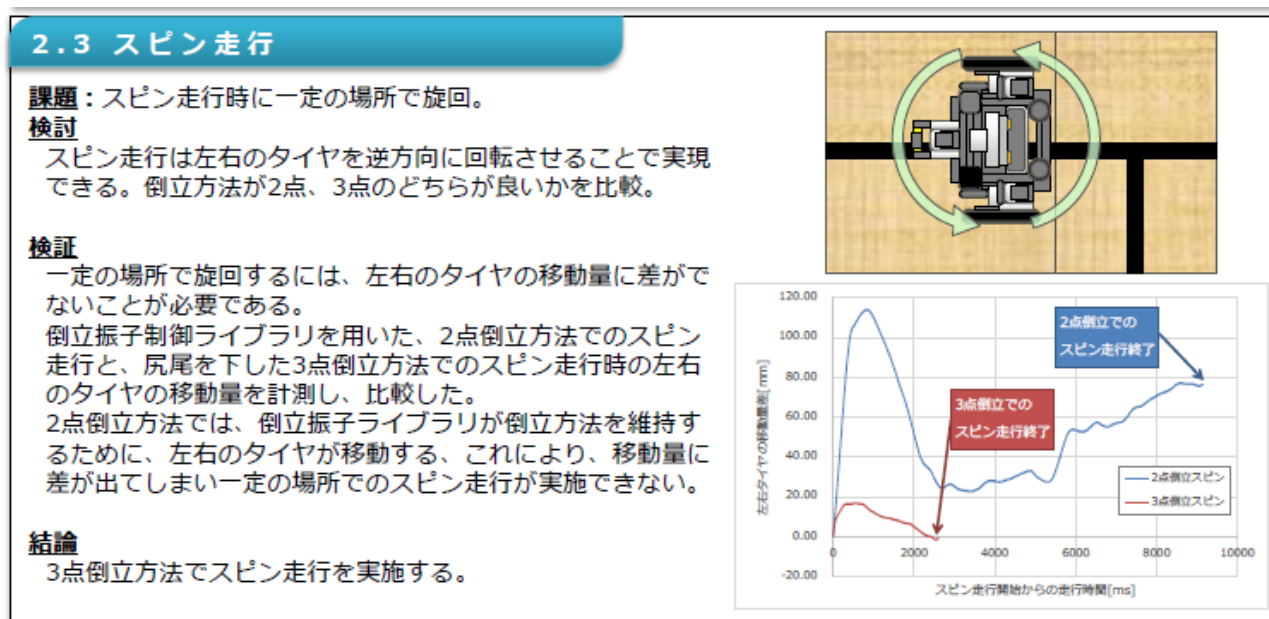


補足) ギア比が示されているが、これはアドバンストクラスにおけるギア比の選択値を示すものであり、プライマリークラスにおいては検討対象外としてよい。

図 31

## 10. スピン

◆出典チーム：FCTawashi（東京地区）



### 図 32

補足)スピンの中心点をずらさずに回転することが必要となる。そのため、左右モータからの指示値によるタイヤの回転量を左右同一にすることが求められるので、個体差を吸収する方法として紹介している「12 ハードウェア部品の個体差対策」を参照するとよい。


## 11. 走行体傾斜

◆出典チーム：オートロボットみとちゃん（東京地区）

### 3-3. 走行体を傾ける

**課題**  
ゲートを潜るために走行体を傾けるが、転倒する恐れがある。

**対策**  
倒立振子を止める際に、前方方向に進むように車輪にトルクをかけることで、慣性力を利用して走行体を確実に後ろ側に傾ける。



※上図の走行体は、前に進んでいるわけではなく、動作の移り変わりを表している。

**走行体を傾ける手順**

- ①走行体が倒立停止
- ②尻尾を少し傾ける(地面にはついていない)
- ③倒立振子を止め、前方に小さなトルクを与える ← 慣性力により後ろに傾く力を利用
- ④目標走行体角度まで少しずつ尻尾を傾ける

**対策後の成果**  
対策前に比べて、より安定して走行体を傾けることが可能となった。


補足) 走行体の全高よりも低いルックアップゲートを通すためには、走行体を傾けなければならない。その際、後方にある尻尾を使って傾斜させるのだが、慎重に傾斜させないと転倒の恐れがあるため注意が必要となる。

図 33

◆FCTawashi（東京地区）

### 2.2 倒立方法の変更

**課題** : 2点倒立から3点倒立への変更。  
**検討・検証** : 変更時の手順を検討・実装し、ジャイロセンサの値をモニタし、安定度を比較。  
**結論** : 下記手順が最も安定度が高い変更可能方法と判断。



手順1	手順2	手順3	手順4
2点倒立方法で停止。	尻尾モーターを制御し、尻尾を指定角度に保つ。	走行体を後ろに傾けるために倒立制御のジャイロオフセット値を減らす。	傾きが終わるのを時間待ちし、倒立制御をOFFにし3点倒立へ移行完了。

図 34

補足) 手順2の尻尾を指定角度（この場合 90 度）に保つところがポイント。倒立制御が OFF となり走行体の荷重が尻尾に掛かっても、指定角度を維持することで転倒を防止する。

## 12. ハードウェア部品の個体差対策

◆出典チーム：からっ風産学隊 2015（北関東地区）

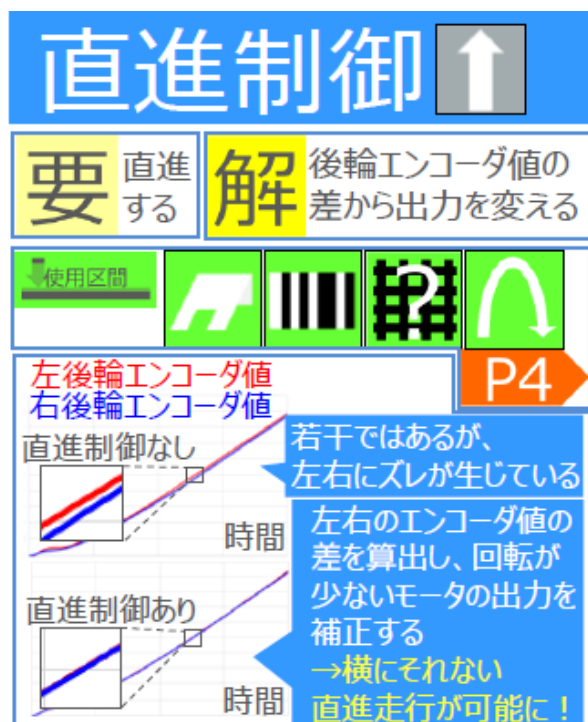


図 35

補足) 同じハードウェア部品であっても製造ロットの違いなどによって性能にバラツキがでる。

部品にはある程度のバラツキがあることを認識して対策することが必要となる。

左右のモータにもバラツキは存在するため、左右のモータに同一の指令値を出して直進走行させた時の左右のエンコーダ値を比較し、バラツキをキャンセルさせる。

◆出典チーム：AC.ひよこ sonic（南関東地区）

## モータ出力時に混入する外要因対策

## 直線走行補正

尻尾走行で直線走行を行う際モータの個体値により、同一のPWM値を左右のモータに付加してもエンコーダ値が同一にならない。これをPD制御で補正する。PD制御の目標値は、暴走抑制のために左右のエンコーダ値のうち、低い方とする。また、車輪の停止を検知する段差衝突検知時はこの機能をOFFにする。

## バッテリー補正

同じPWM値でも、バッテリー残量によってモータに与える電力が変化してしまう。これを補正する。補正式を以下に示す。

補正後PWM値 = 入力PWM値 × 基準電圧 / 電池電圧

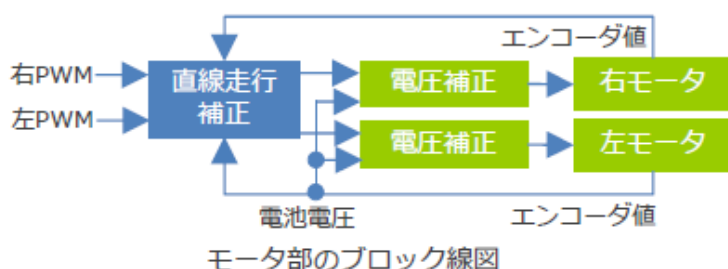
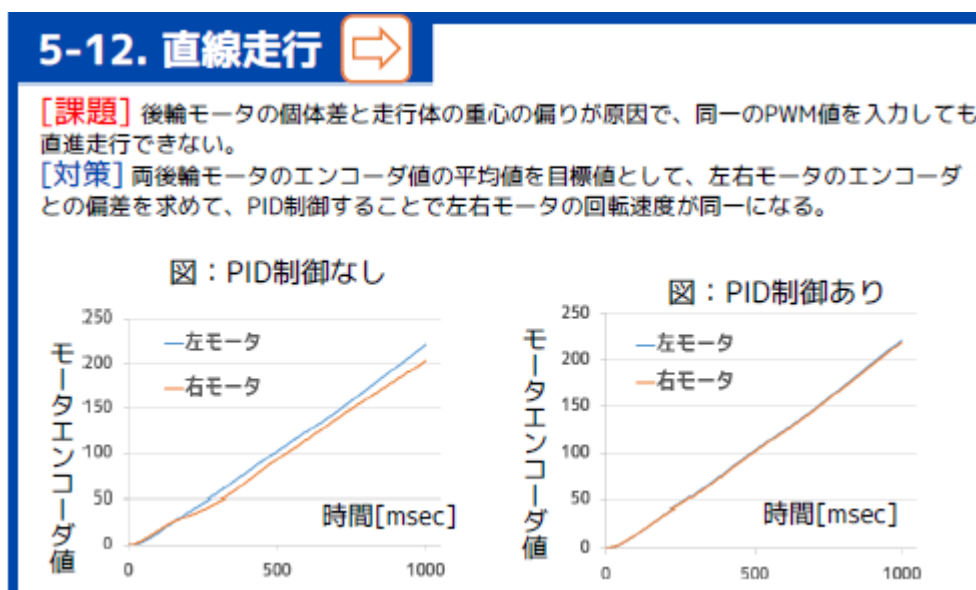


図 36

補足) 左右のモータに指示するPWM値に対して、左右のエンコーダ値の低い値を目標値としたPID制御にて個体値のバラツキをキャンセルしている。さらに、走行中の電池電圧の変動も制御することで安定した直進走行を目指している。



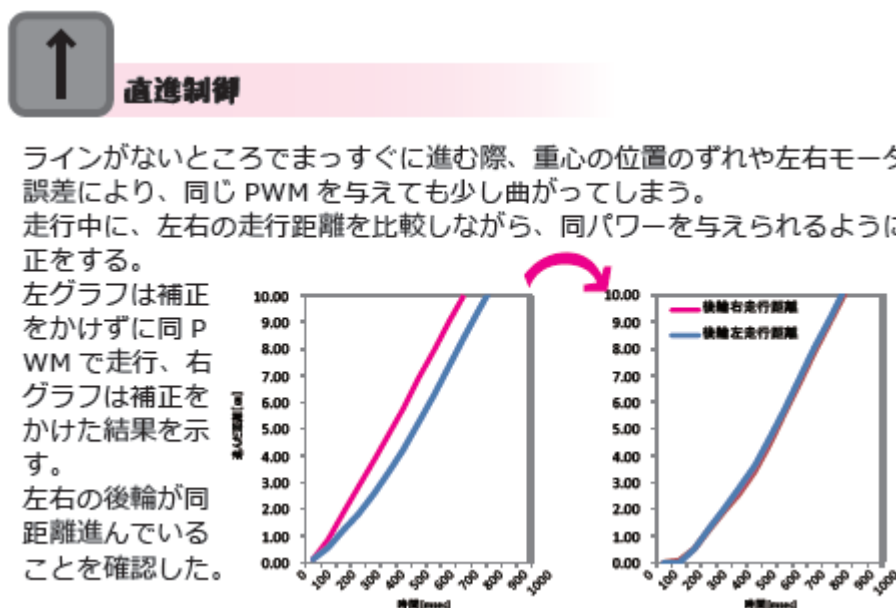
◆出典チーム：NiASET（九州北地区）



補足) 左右モータのエンコーダ値の平均を目標値とした PID 制御により、左右モータに与える PWM 値を決定し、左右のバラツキを補正している。

図 37

◆出典チーム：雪桜（東京地区）



補足) 左右のモータに同一 PWM 値を与えても直進しない要因には、モータの固有さだけでなく、走行体の重心位置が中心でないことが影響している。

図 38



## Ⅱ. EV3 走行体向けノウハウ

NXT と EV3 の大きな変化点にカラーセンサ（光センサ）の性能が挙げられる。NXT は 10 ビットのダイナミックレンジを持っており 1024 段階（0 ～ 1023）の分解能表現が可能であったが、EV3 は 100 段階の分解能表現しか使えない。

これにより、カラーセンサの黒色輝度値と白色輝度値や黒色輝度値と灰色輝度値の差が小さくなり、カラーセンサでコースの黒色ラインに沿ってラインレースをおこなったり、灰色マーカを識別したりすることが難しくなることが予想される（図 40 の緑丸で示した輝度値の分解能表現）。

したがって、カラーセンサの分解能表現が小さくなったことによるセンサの性能ダウンを何らかの方法で補うような工夫が必要となる。

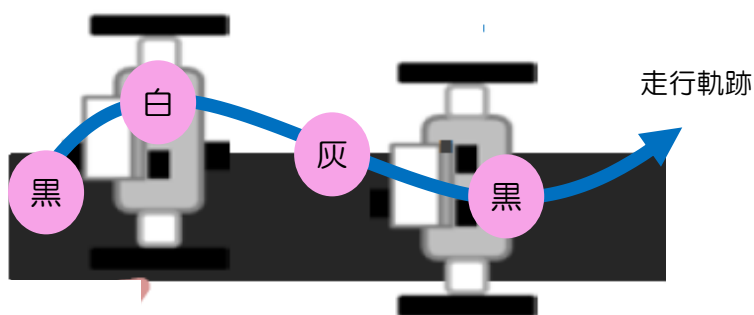


図 39

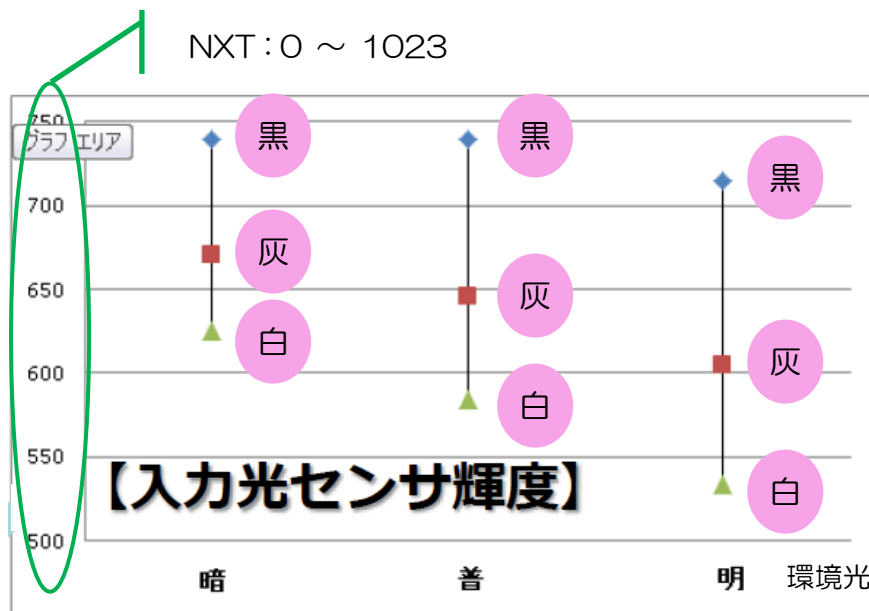


図 40

補足）環境光（暗い、普通、明るい）に関わらず、黒色と白色の幅が NXT よりも EV3 の方が狭くなる。すなわち、識別したい色の輝度値が接近するため、識別が難しくなる。

詳しくは下記の URL を参照。

### ■NXT

[http://lejos-osek.sourceforge.net/ecrobot\\_c\\_api\\_jp.htm#LightSensor](http://lejos-osek.sourceforge.net/ecrobot_c_api_jp.htm#LightSensor)

API 名 : U16 ecrobot\_get\_light\_sensor(U8 port\_id)



U16 ecrobot_get_light_sensor(U8 port_id)	光センサデータ(10ビット A/Dコンバータ)の取得。値が大きいほど反射率が低い(または暗い色からの反射)  引数: port_id: NXT_PORT_S1, NXT_PORT_S2, NXT_PORT_S3, NXT_PORT_S4 戻り値: 0~1023
---	---

### ■EV3

[http://www.toppers.jp/ev3pf/EV3RT\\_C\\_API\\_Reference/group\\_\\_ev3sensor.html#ga6f0760f8a0781fd5397f3365c6d6ac87](http://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3sensor.html#ga6f0760f8a0781fd5397f3365c6d6ac87)

API 名 : uint8\_t ev3\_color\_sensor\_get\_reflect(sensor\_port\_t port)

**uint8\_t ev3\_color\_sensor\_get\_reflect ( sensor\_port\_t port )**

カラーセンサで反射光の強さを測定する。

不正のセンサポート番号を指定した場合、常に0を返す(エラーログが出力される)。

引数

**port** センサポート番号

戻り値

反射光の強さ(0~100)

### Ⅲ. 2015 年プライマリークラス競技規約ダイジェスト

2015 年プライマリークラス競技の特徴（コース、難所の一部）を簡単に紹介する。詳細な情報入手したい場合には、「ET ロボコン 2015 デベロッパー部門競技規約 1.1.0」を参照されたい。

(<https://www.etrobo.jp/2015/gaiou/kiyaku.php>)

#### 1. コース全体

◆出典チーム：' 15（東京地区）



図 41

補足) Rコース、Lコースが用意されている。

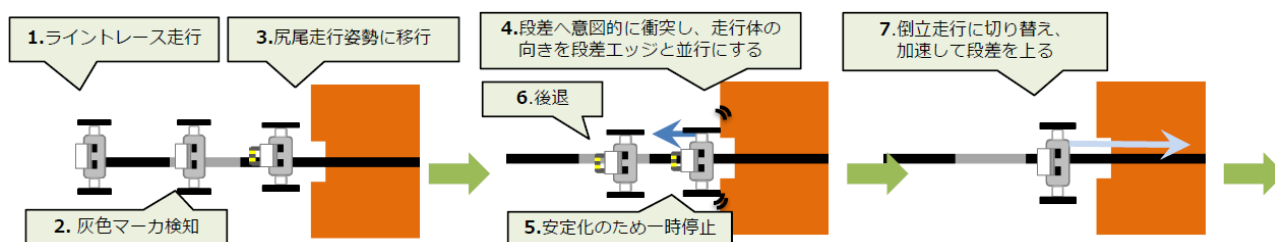
Rコースはスタート直後の長いストレート・エンドに左急カーブがあるコース。難所として、フィギュアLとガレージRがある。

Lコースはスタート直後の長いストレートに続いて複合カーブがあるコース。難所として、ルックアップゲートとガレージLがある。

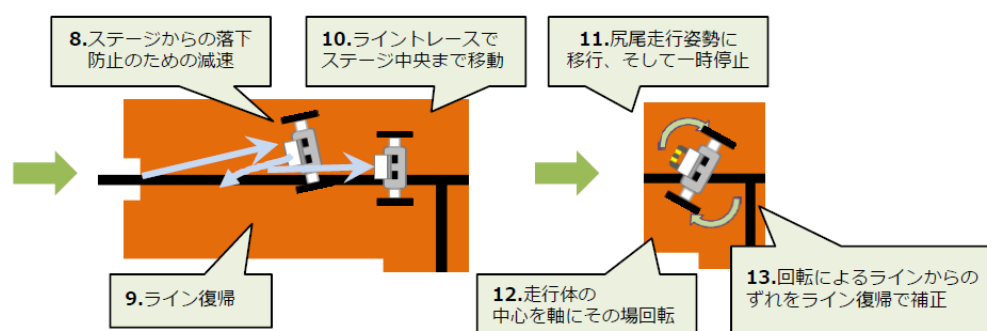
## 2. フィギュアし

◆出典チーム：AC.ひよこ sonic（南関東地区）

### ■ステージを上る走行戦略



### ■ステージ上で一回転するための走行戦略



### ■ステージを降りる走行戦略

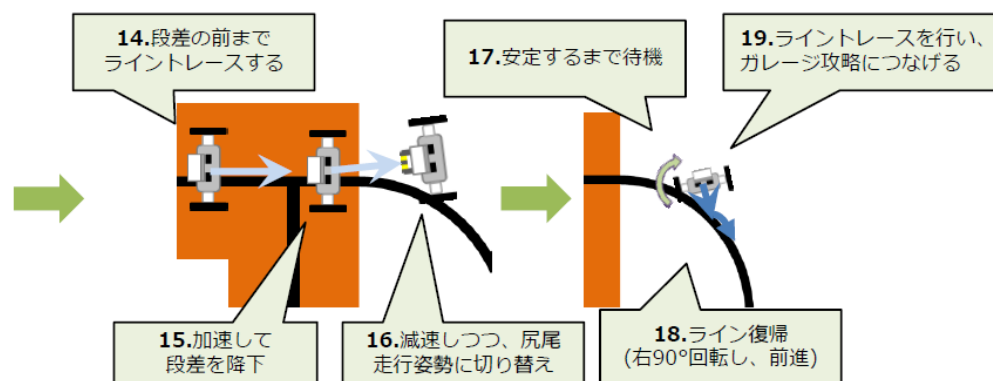


図 42

### 3. ルックアップゲート

◆出典チーム：GrowUp（九州北地区）

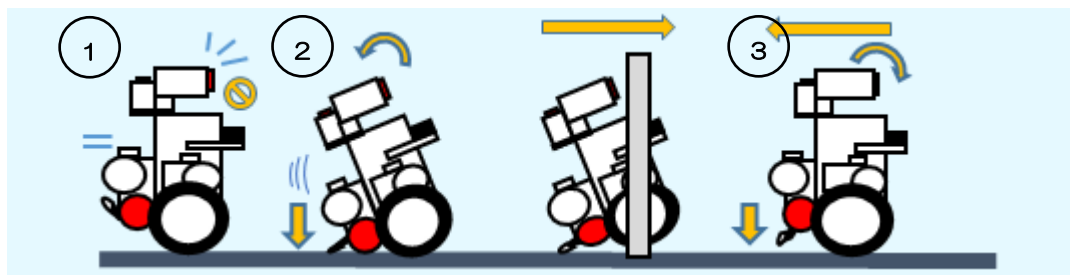


図 43

補足）二輪走行と三点走行の二つの状態を切り替えてルックアップゲート難所をクリアする

二輪走行（倒立振子制御プログラム ON）



- ① 勢角度の急激な変化を抑えるため、段階的に尻尾の角度を増やす
- ② 前方転倒防止のため、低速前進で後ろに重心を移して確実に後方へ倒す

三点走行（倒立振子制御プログラム OFF）



- ③ 姿勢を起こす際、尻尾モータの力だけでは起き上がれないため、若干後退して重心を前方に移すことによりスムーズに姿勢を起こす

二輪走行（倒立振子制御プログラム ON）

## あとがき

本資料で紹介したノウハウは、ET ロボコン 2015 チャンピオンシップ大会出場チームから提出されたモデルの一部に過ぎないため、より多くの情報を得たい場合には、ET ロボコン実行委員会から提供される前年度参加チームのモデルを参照されたい。300 を超えるチームから提出されたモデルには、ET ロボコンに取り組むにあたっての膨大なノウハウが書き込まれており、初めて ET ロボコンに参加するビギナーはもちろんのこと、2 回目、3 回目の参加者にとっても有益な情報をもたらしてくれるであろう。

参加チームが描いたモデルは、ET ロボコンビギナーにとってよき参考書であり、この貴重な情報源を有効に活用しないのは実にもったいないことである。まさに宝の山と言っても過言ではないであろう。

また、インターネット上には ET ロボコン参加者のブログなどが多数公開されており、自ら積極的に必要な情報を検索し入手することで、ET ロボコンの開発だけでなく組込み開発にも役立つ情報を得ることができるであろう。



## 改訂履歴

版数	日付	執筆者	内容
1.0.0	2016/04/28	本部性能審査団) 河野	初版
1.0.1	2016/04/30	//	出典モデルの地区名追記
1.0.2	2016/05/07	本部審査委員) 渡辺	誤記の修正