ET ロボコン 2012
東北地区独自技術教育

自己位置同定の概念と実装

ET ロボコン東北地区
技術委員会
新井　義和

自己位置同定とは

ロータリー・エンコーダ
… モータの回転数を検出するセンサ
⇒　車輪の転がった距離を算出可能

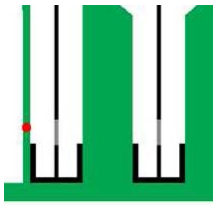
☆前提条件
- 車輪の半径が既知
- 車輪は接地面に密着　→　滑らない※

転がった距離を積算　⇒　ロボットの現在位置

※実際はすべりが避けられないので，位置誤差が発生する

応用例

- ライン無視走行（ショートカット）
  ラインがなくても，現在位置が分かれば
  目的地までの距離／方向が分かる
- ラインの区間分割
  任意に分割した区間の出入りを
  マーカーに頼らずに検出
- ガレージイン
  ガレージインはマーカー後の距離が重要



ロボットの速度と回転角速度

ロボットの速度を v，回転角速度を ωとする．

$$v_R = \left(R + \frac{W}{2}\right)\omega \quad \cdots (1)$$

$$v_L = \left(R - \frac{W}{2}\right)\omega \quad \cdots (2)$$

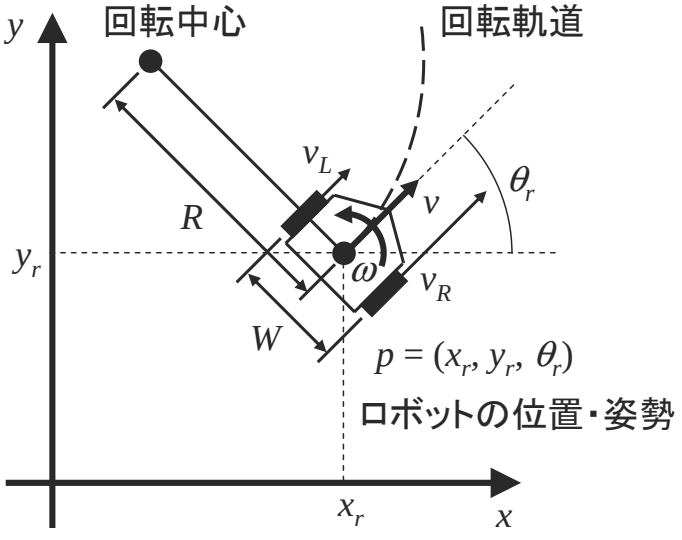
$$v = R\omega \quad \cdots (3)$$

式 (1), (2), (3) より

$$v = \frac{v_R + v_L}{2}$$

$$\omega = \frac{v_R - v_L}{W}$$

※角度の単位は
ラジアン！



左右の車輪の移動速度 v_L, v_R から v, ω を算出可能！

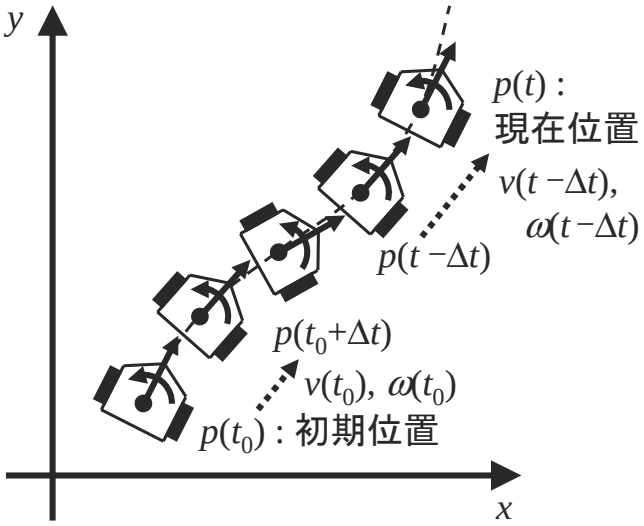
自己位置の算出

微小時間 Δt の間の微小移動量を
積算して現在位置を算出する.

$$x_r(t) = \int_{t_0}^{t} v \cdot \cos\theta_r(\tau)\, d\tau + x_r(t_0)$$

$$y_r(t) = \int_{t_0}^{t} v \cdot \sin\theta_r(\tau)\, d\tau + y_r(t_0)$$

$$\theta_r(t) = \int_{t_0}^{t} \omega(\tau) d\tau + \theta_r(t_0)$$



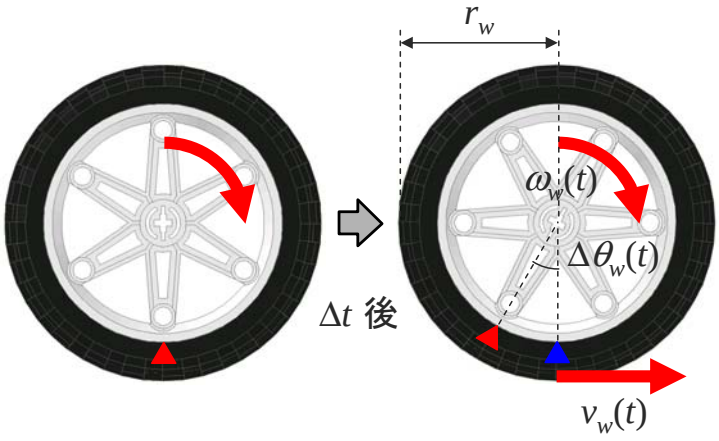
車輪の移動速度

微小時間 Δt の間の車輪の回転角度変化量 $\Delta\theta_w(t)$ から
車輪の回転角速度 $\omega_w(t)$ を算出し, 車輪の移動速度 $v_w(t)$ を得る.

$$\Delta\theta_w(t) = \theta_w(t) - \theta_w(t-\Delta t)$$

$$\omega_w(t) = \frac{\Delta\theta_w(t)}{\Delta t}$$

$$v_w(t) = r_w \cdot \omega_w(t)$$

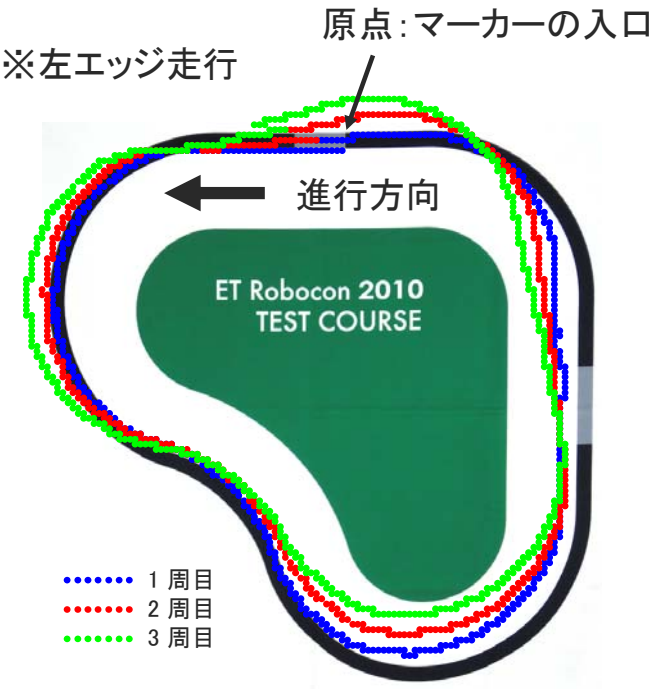


自己位置同定の結果

サンプルコースを 3 周したときの走行体の自己位置同定結果をプロット

徐々に誤差が大きくなっていくことを確認！！

※短期的にはそれなりに
正しい推定が期待できる



実装の準備

本部提供のサンプルプログラム `sample_c4` をベースに実装してみよう！

① `sample_c4` と同じディレクトリに `sample_c4` をフォルダごとコピー　※フォルダ名：`sample_s1`

② `sample_s1` ディレクトリ内の `Makefile` を修正

```
# ターゲット実行形式ファイル名
TARGET = sample_s1
#TARGET = sample_c4
```

後は，`sample_s1/sample.c` を修正するだけ

sample_c4 をベースとした実装（TASK() 関数）

```
balance_init();                                          /* 倒立振子制御初期化 */
nxt_motor_set_count(NXT_PORT_C, 0); /* 左モータエンコーダリセット */
nxt_motor_set_count(NXT_PORT_B, 0); /* 右モータエンコーダリセット */
while(1) {
  tail_control(TAIL_ANGLE_DRIVE); /* バランス走行用角度に制御 */

  if (sonar_alert() == 1) { /* 障害物検知 */
    forward = turn = 0; /* 障害物を検知したら停止 */
  } else {
    forward = 50; /* 前進命令

    if (ecrobot_get_light_sensor(NXT_PORT_S3) <= (LIGHT_WHITE + LIGHT_BLACK)/2) {
      turn = 50;  /* 右旋回命令 */
    } else {
      turn = -50; /* 左旋回命令 */
    }
  }

  /* 倒立振子制御(forward = 0, turn = 0で静止バランス) */
  balance_control( ･･･(省略)･･･ );
  nxt_motor_set_speed(NXT_PORT_C, pwm_L, 1); /* 左モータPWM出力セット(-100～100) */
  nxt_motor_set_speed(NXT_PORT_B, pwm_R, 1); /* 右モータPWM出力セット(-100～100) */

  systick_wait_ms(4); /* 4msecウェイト */
}
```

無限ループの先頭で
自己位置同定をしてみよう
→ その結果に基づいて，
戦略を決定

車輪の回転角度の取得

■ nxt_motor_get_count() を利用

- 戻り値は int 型（符号あり 32 ビット）
- 車輪が約 600 万回転するとオーバーフロー
  → 事実上，オーバーフローを気にする必要なし
- 走行体の進行方向が正回転（カウントアップ）
- バランス制御でもカウンタを利用しているので，
  nxt_motor_set_count() は使用してはならない

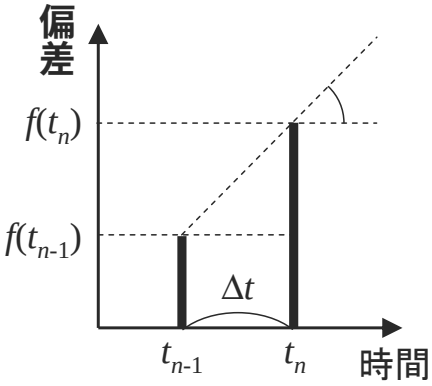
■ 回転角度の初期化

回転角度の変化量を算出するために，1 ステップ前の値が必要
→ 1 ステップ前の値を適切に初期化

微分と積分

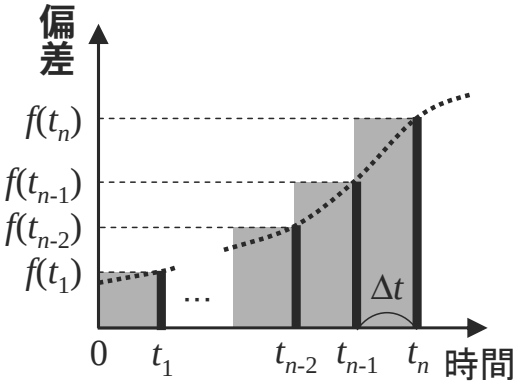
まじめに微分／積分をやろうとすると大変！！
→　ほどほどの手法で近似してみよう

■ 微分の近似



$$\frac{df(t_n)}{dt} \fallingdotseq \frac{f(t_n) - f(t_{n-1})}{\Delta t}$$

■ 積分の近似



$$\int_0^{t_n} f(\tau)d\tau \fallingdotseq \sum_{i=1}^{n} f(t_i) \times \Delta t$$

ロボット座標の送信の実装例

使用変数

送信バッファ：

```
unsigned char tx_buf[256];
```

ロボットの座標：

```
int temp_x;
int temp_y;
```

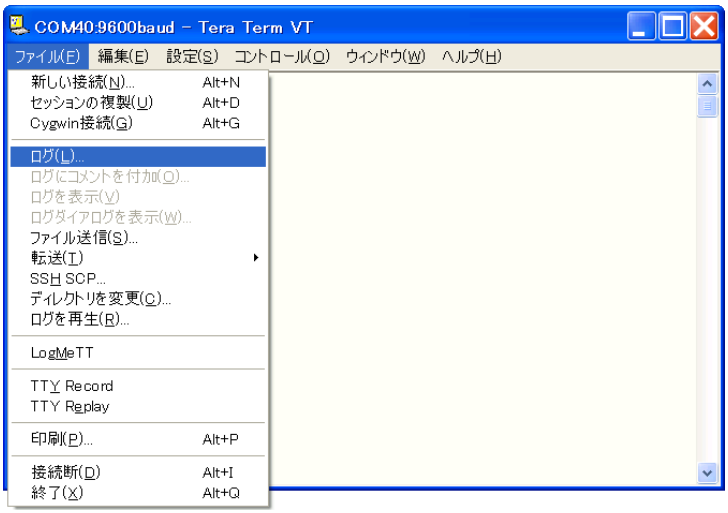
```
/* 送信データをバッファにセット */
if (temp_x >= 0)
  tx_buf[0] = ' ';
else {
  tx_buf[0] = '-';
  temp_x *= -1;
}
tx_buf[1] = (unsigned char)(temp_x / 1000) + '0';
temp_x %= 1000;
tx_buf[2] = (unsigned char)(temp_x / 100) + '0';
temp_x %= 100;
tx_buf[3] = (unsigned char)(temp_x / 10) + '0';
temp_x %= 10;
tx_buf[4] = (unsigned char)temp_x + '0';
tx_buf[5] = ' ';

if (temp_y >= 0)
  tx_buf[6] = ' ';
else {
  tx_buf[6] = '-';
  temp_y *= -1;
}
tx_buf[7] = (unsigned char)(temp_y / 1000) + '0';
temp_y %= 1000;
tx_buf[8] = (unsigned char)(temp_y / 100) + '0';
temp_y %= 100;
tx_buf[9] = (unsigned char)(temp_y / 10) + '0';
temp_y %= 10;
tx_buf[10] = (unsigned char)temp_y + '0';

tx_buf[11] = 0x0d; /* CR */
tx_buf[12] = 0x0a; /* LF */

ecrobot_send_bt(tx_buf, 0, 13);
```

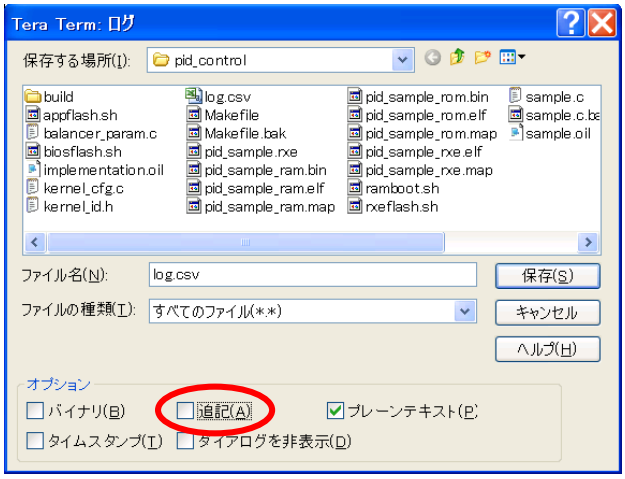
Windows における
改行コードは CR+LF

ログの保存開始（Tera Term の場合）

①「ファイル」→「ログ」

②ログダイアログから保存する場所とファイル名を設定
※「追記」チェックボックスのチェックをはずす

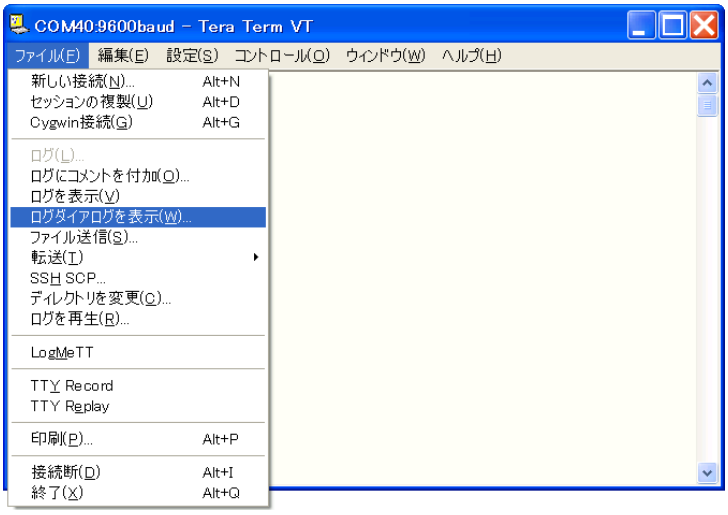


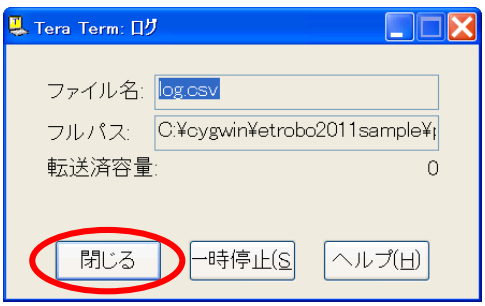


ログの保存終了（Tera Term の場合）

①「ファイル」→「ログダイアログを表示」

②ログダイアログから「閉じる」ボタンをクリック
※「一時停止」も選択可　→　「再開」で保存再開





自己位置の補正

誤差が大きくなる前に自己位置を補正するのが基本

タイミング	基準	補正対象
マーカー検出時	マーカーの絶対座標	座標
直線の検出時	直線の向き	姿勢

走行距離の誤差よりも，姿勢角の誤差が致命的

補正の戦略によって，自己位置同定の成否が決まる