九州工業大学 CIR-KIT のつくばチャレンジへの取り組み

有田 裕太（九州工業大学） 田中 良道（九州工業大学）
森田 賢（九州工業大学） 西田 健（九州工業大学）

Arita Yuta(Kyutech), Tanaka Ryodo(Kyutech),
Morita Masaru(Kyutech) and Nishida Takeshi(Kyutech)

1. はじめに

我々は，九州工業大学工学部の学部生を中心としたロボット開発チームであり，屋内外の移動を行う福祉ロボットの開発に取り組んでいる．このような福祉ロボットに必要とされる機能には，案内・荷物の運搬・搭乗者の安全な搬送・周囲環境に対する回避，発見，追従動作などが挙げられる．そこで我々は市販の電動カートを改造し，これらの作業を行うための福祉ロボット (KIT-C3) を開発した．しかし，このロボットでは屋内を走行するには大きすぎるため，比較的小さく，小回りもきくロボット (KIT-C4) の開発を行った．本稿では，これらのロボットの開発とつくばチャレンジ 2015 での実験結果について報告する．

2. ロボットの構成

2.1 KIT-C3

2.1.1 ハードウェア構成

本ロボットでは，シニアカータウンカート (スズキ株式会社製 TC1A4) の走行系を用いた．シニアカーの走行系は，バッテリーや走行安定性，防水性など，様々な面で屋外での走行機能の信頼性が高く，故障などの問題の発生率が低い．さらに，ホイールレングスが 1[m] 程度あるため，直進走行性能が高く，自動走行制御に対して再現性が高いというアドバンテージがある．一方で，つくばチャレンジの他チームの小型走行ロボットよりは比較的車体が大きく，車重も 97[kg] と重い．Fig.1 に KIT-C3 の外観を示す．

2.1.2 センサ構成

KIT-C3 は環境観測用センサとして前方に LRF(北陽電機製 UTM-30LX) を 2 台，後方にも同様の LRF を 1 台搭載している．また，ロボットの状態観測用センサとしてロータリーエンコーダ (MUTOH 製 UM-125) をステアリング軸に 1 つ取り付けている．このセニアカーの後輪は独立に 2 つのモータとエンコーダが搭載されており，そのエンコーダ情報をマイコンにより取得している．

Fig. 1: KIT-C3 の外観

2.1.3 全体の構成

ロボットへの速度指令やロータリーエンコーダの処理用マイコンとして iXs Research 社の iMCs01 を用いた．シニアカーの速度制御は iMCs01 で 0-5[V] の電圧を生成しシニアカーのアクセル信号部に印加することで実現している．またステアリング操舵のためステッピングモータの制御に Arduino Uno を用いた．制御用 PC はラップトップ PC を用いており，OS は Ubuntu 14.04 を使用した．以上の構成の概要を Fig.2 に示す．ラップトップ PC と iMCs01 や URG とは USB 接続である．また，Fig.1 における破線はシニアカーの制御基板によって制御されていることを表す．

2.2 KIT-C4

2.2.1 はじめに

本年度のつくばチャレンジにおいて KIT-C4 の開発を行ったが，予算不足や開発人員の不足により，実際につくばチャレンジの実験走行や本走行を行うことはなかった．よって本稿における KIT-C4 についての記述はすべて開発途中の成果であり，開発の経験，知識を共有することを目的としていることに留意されたい．

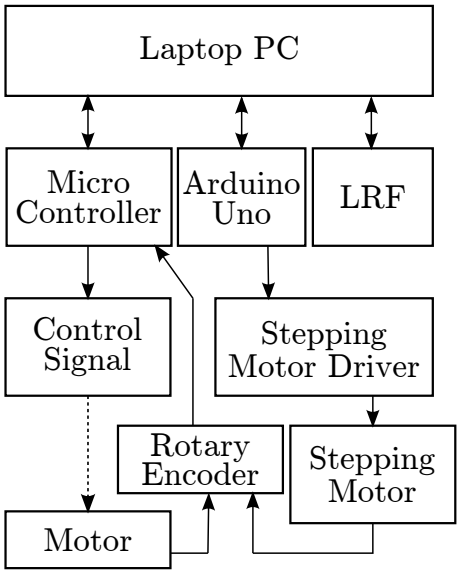


Fig. 2: KIT-C3 のシステムの概要

2.2.2　ハードウェア構成

KIT-C4 は，KIT-C3 と比べて小型化，旋回性能の向上を図り製作した独立二輪駆動方式のロボットである．実際のロボットを Fig.3 に示す．KIT-C4 は昨年度より開発を行っている．ゆえに本年度追加，変更した要素について述べる．まず追加要素について，本年度は車載 PC と無線 LAN ハブを新たに搭載した．これは，ROS[1] を用いた分散処理を行うためである．車載 PC でロボットの走行制御とセンサ情報処理を行い，走行プランナなどの上位層のアプリケーションを実行する PC と分散処理を行う．続いて変更要素について，昨年度スイッチボックスを作成したが，Fig.4 に示すように，ボックスの取り付け位置に問題があり，配線の取り回しが非常に複雑になってしまった．そこで本年度は，ボックスの取り付け位置を変更し，配線の取り回しを行いやすくした．これによりこれまで無駄に使用していた空間を確保することができ，遠隔操縦コントローラ等の設置スペースを設けることができた．

2.2.3　センサ構成

KIT-C4 は昨年度と同様，環境観測用センサとして前方下方に LRF（北陽電機製 UTM-30LX-EW）を，前方上方に LRF と陽動機構（移動ロボット研究所製 Gim30 ）を組み合わせた　3 次元 LiDAR を搭載している．また，ホイールオドメトリ観測のため，駆動輪にそれぞれロータリーエンコーダ（オムロン製 E6B2-CWZ60）を設置した．これらのセンサに加えて，本年度は GPS（Hemisphere A100）を追加で搭載した．この GPS は本年度開発した自律走行計画に用いることは予定していないが，センサ系に冗長性を持たせることで，他のセンサが故障し，替えが効かない際に用いることを想定している．

[1]Robot Operating System の略称．なお，用いたバージョンは Indigo である．

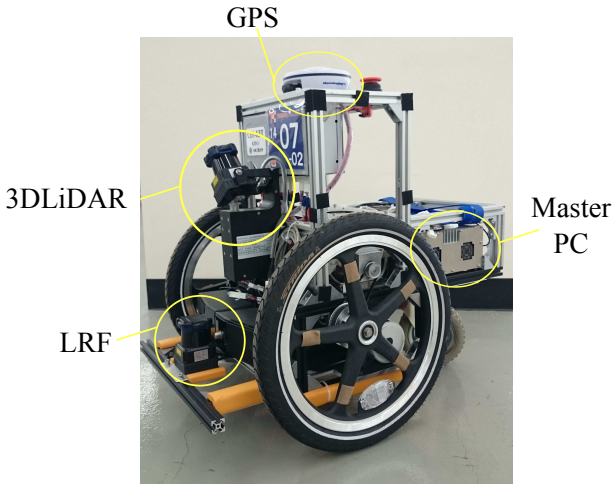
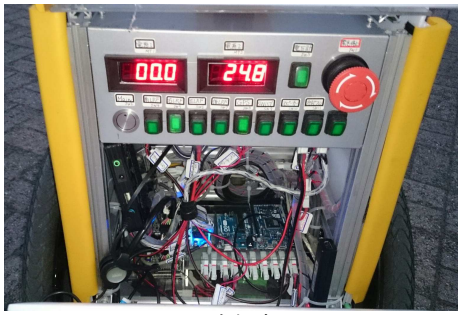
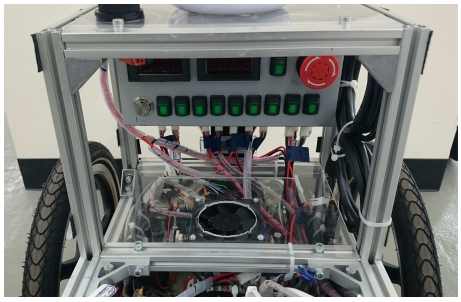


Fig. 3: KIT-C4 の外観



昨年度
ボックスが手前にあり，デッドスペースが多い



今年度
ボックスを奥に設置し，スペースを確保

Fig. 4: KIT-C4 のスイッチボックス

2.2.4　システム概要

KIT-C4 のシステム概要を Fig.5 に示す．先述のとおり，ロボットの走行制御，センサ情報処理は車載 PC で行い，この PC を ROS の MASTER としている．この車載 PC に走行計画を行う PC を，クライアント PC として Ethernet 経由で接続する．これにより走行計画などの比較的リソースを必要とするプログラムに十分な駆動環境を与えることができ，走行の安定化を図ることができる．尚，後ほど紹介する ROS の走行プランナーについては，リソースが不足するとロボットの走行の安定性に大きな悪影響を及ぼすことが実験を行う中で判明している．よって，このような分散処理は，つく

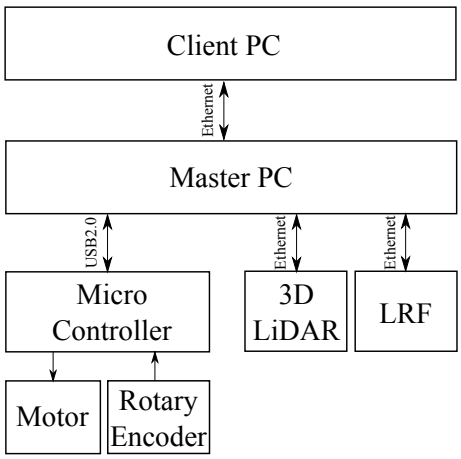


Fig. 5: KIT-C4 のシステム概要

ばチャレンジのような走行の確実性を求められる大会において非常に有効な手段であると考えられる．

2.2.5　モータについて

KIT-C4 の駆動モータには昨年度と同様，三相ブラシレスモータ（オリエンタルモータ製 BLH230K-30）を使用した．昨年度の開発において，これに付属するモータドライバがロボットの速度制御に適していないことが判明しており，ドライバの変更を行うことを考えていた．しかし，予算の都合より，ドライバを交換することが不可能であったため，昨年度と同様，モータドライバへの入力値を制御することで，走行の安定化を図った．付属のモータドライバは速度制御を行うものであり，0〜5[V] の PWM 信号に合わせてモータの回転速度を制御するようなドライバである．昨年度まではこのドライバに対し，試行錯誤により if・then 形式のプログラムを作成していた．昨年度の走行の結果，低速時には比較的安定して走行することが判明している．しかし本年度はハードウェア構成の変更があり，再び試行錯誤を繰り返す時間が無かったため，よりシステマティックな制御器を構築することを目標とした．また，昨年度はロボットの走行時にブレーキ動作が多かった（目標値＋閾値を超える速度になるとブレーキ動作を行うようなアルゴリズムであった）ため，オーバーシュートをなるべく少なくすることも目標とした．そこで本年度は I-P 制御を用いることとした．I-P 制御では，積分要素に偏差を，比例要素に観測値を用いるようなフィードバック制御であり，目標値が急激に変化しても制御値が急激に変化しづらいため，オーバーシュートが起きづらいという特徴がある．I-P 制御について，Fig.6 に制御システムのブロック線図を示す．ただし，図中の r は目標速度を，u は制御入力を，y は観測値を表し，k_p は比例ゲインを，k_i は積分ゲインを表す．また，目標速度 r に対し，制御入力 u が PWM 信号，出力 y が駆動輪に取り付けたロータリーエンコーダの観測値となるようなシステムを制御対象とした．I-P 制御を行うには制御対象の同定が必要であるので，システムを二次遅れ系としてステップ同定を行った．制御器を取り付けた結果，平坦な道を走行する際にはオーバーシュートの少ない，安定した走行を行うことができたが，坂道では不安定な走行となってしまった．これは，坂道を走行しているロボットのモデルが同定したモデルとは異なるために起きた現象である．ゆえに予め同定したモデルを用いた制御を行うのではなく，モデルをオンラインで推定し，推定したモデルに対して制御を行う必要がある．I-P 制御の採用を決定した際には，無視できるほどのモデル化誤差であると予想していたが，これが間違いであることがわかった．以上より，モータドライバを変更しないのであれば，オブザーバを用いた状態フィードバック制御やモデル予測制御など，よりロバストな制御を行うことでこれらの問題に対処できるものと考えられる．

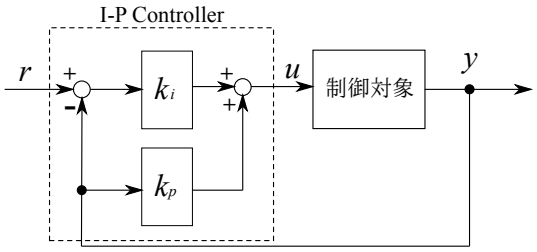


Fig. 6: I-P コントローラ

3.　ソフトウェアの構成

3.1　ROS

開発したロボットには先述のとおり ROS を導入している．これはソフトウェアの再利用性を意識したものである．実際，KIT-C3 で開発したソフトウェアを KIT-C4 で再利用することが可能であり，開発時間の短縮を図ることができる．また，ROS を用いることでデバッグや可視化に有効なツールだけでなく，世界中の開発者が作成したライブラリを容易に利用することができる．ゆえに我々は，ソフトウェア構成において ROS を採用した．

3.2　走行手法

走行手法には，ROS の navigation スタックを利用した．予め環境地図を作成し，環境地図上に Waypoint を数メートル毎に設定し，それらを順に辿っていくようにして自律移動を行った．Waypoint の設定は ROS の可視化ソフトウェア rviz 上でクリックやドラッグ操作で設定できるパッケージを開発しこれを用いた．経路生成および障害物回避には move_base パッケージを用いた．また，自己位置推定には AMCL（AdaptiveMontecalro Localization) を用いた．昨年は前方下に取り付けた 1 台の LRF で自己位置推定を行っていたが，今年は前方と

後方に取り付けた 2 台の LRF で自己位置推定を行った．

3.3　環境地図

環境地図は ROS のパッケージ化された gmapping を用いて作成した．環境地図は 10[cm] 四方の占有格子地図とした．環境地図作成に用いたセンサは昨年は LRF1 台とホイールオドメトリのみであったが，今年は自己位置推定同様に前方と後方に取り付けた 2 台の LRF とホイールオドメトリを利用して作成した．これにより大清水公園などでも比較的精度の高い地図を短時間で作成することができた．また，確認走行を完了したあとは大清水公園と公園の外でそれぞれ地図を作成し画像編集ソフトを用いて 1 つに統合したものを用いた．作成した地図を Fig.7 に示す．作成された地図では，経路は閉じていないものの，重なりは生じていないため，自己位置推定などには問題なく使用できた．作成された地図が歪んでいる原因はオドメトリがずれているためと考えられる．これはオドメトリパラメータの修正や，ジャイロセンサを組み合わせることでオドメトリを高精度化できれば更に高精度な地図を作成できると考えられる．

3.4　下方段差検出

環境地図には存在しない障害物の回避には ROS の local_planner パッケージを利用したが，下方段差検出における問題が発覚した．KIT-C3 には，下り階段や急激な下り坂への進入を回避するために，地面方向に照射する LRF（下方照射 LRF）が搭載されている．ところが，そのような段差で得られる下方照射 LRF の生データを local_planner に入力しても，段差を回避する経路を獲得できない．この理由は，次に示す local_planner の仕様と LRF の特性の不一致にあり，具体的には，1) local_planner は地面よりも高い場所に位置する物体を障害物と検知する，すなわち下方照射 LRF の出力距離が地面までの距離より小さいことを検知するが，2) 実際には段差において下方照射 LRF は地面までの距離よりも大きい距離を出力する，というものである．つくばチャレンジ 2014 では正にこの問題のため，段差に進入の直前でロボットを停止させた経緯があった．

そこで今年度は，下方照射 LRF が地面までの距離よりも大きい値を出力した場合，それを local_planner が障害物と認識できる値に人為的な変更を加える機能を追加した．この様子を Fig.8 に示す．

最後に，本機能の工夫点について述べる．下方段差を機能させるためには，ロボット，LRF，地面との位置関係や，環境に合わせた LRF の感度の調整が必要となることが予想されたため，細かくパラメータを調整できる仕様を定めて実装を行った．これにより，事前の検証において，屋内では段差検出ができたが屋外で検出ができないという問題が発生したが，パラーメータ調整だけでこの問題を容易に解決することができた．

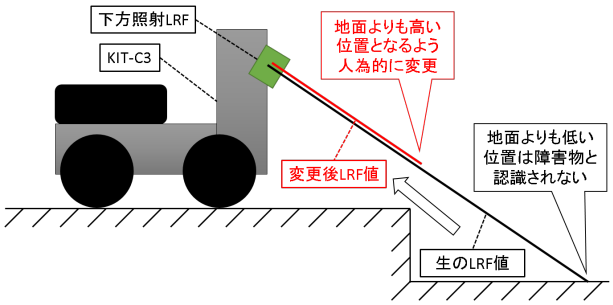


Fig. 8: KIT-C3 の下方段差検出

3.5　遠隔監視システム

つくばチャレンジの注意事項等について，「ロボットの位置や状況のステーションにおけるモニタリング」を強く推奨すると明記されている．我々は本要求事項を達成する遠隔監視システムを構築したので，その機能とネットワーク環境について述べる．ただし，本システムは大会当日に機能させることが出来なかったため，以下では開発時の情報を用いて説明する．

初めに，遠隔監視について述べる．監視画面として，ロボットの現在姿勢を地図上に表示させる形式を採用した．現在姿勢，あるいは現在姿勢と経路履歴の両方，を選択的に表示できるようにした．現在姿勢を表示した様子を Fig.9 に示す．なお，同図で画像の一部のみを拡大をしている箇所は説明のために加工を施したもので，実際のシステムでは背景の大域地図上にロボットの姿勢が表示されるのみである．

次に，通信環境について述べる．監視端末とロボット間の通信回線として商用モバイル回線を利用し，VPN で接続を行った．VPN システムとして，OpenVPN を採用した．遠隔監視 PC をサーバ，ロボットをクライアントとして VPN ネットワークを構成し，ROS ネットワークと連携させることで，遠隔監視を実現した．

ここで，工夫点について記述する．本システム開発当初，本構成で ROS の可視化ツールである rviz による監視を試みたところ，rviz はリアルタイムに更新される位置情報を全て取得しようと試みるのだが，商用回線の通信速度ではその情報量に対応しきれず，情報の更新漏れが頻繁に発生してしまった．また，パケット通信容量が膨大で，契約した通信回線の上限を容易に超えてしまう恐れがあり，遠隔監視そのものが実現できなくなる懸念があった．そこで，ロボットが一定距離を移動する毎に，自身の位置を監視端末に送信する仕様に変更することで，更新漏れとパケット通信料の問題を解決した．検証段階では送信間隔 5[m] 毎に設定したが，ロボットの位置を追跡するには十分な周期であった．

Fig. 7: KIT-C3 の作成した環境地図 (コース全域)

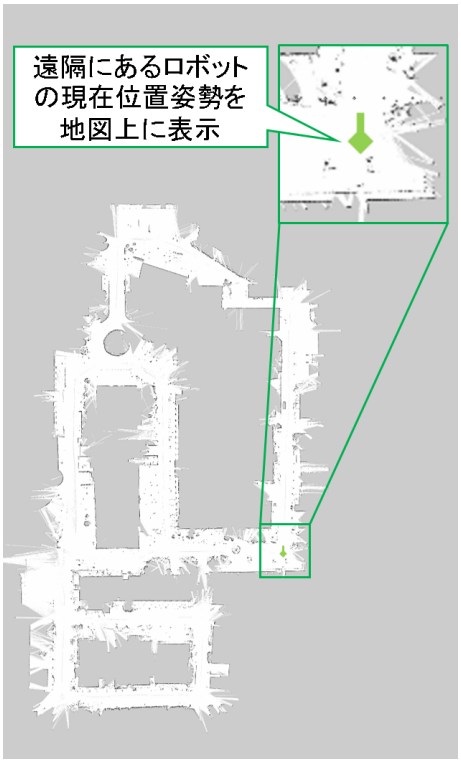


Fig. 9: 遠隔監視システムで地図上に現在姿勢を表示させた様子.

最後に，本機能の展望を記述する．ネットワークシステムを独自で構築したため，位置に限らず任意の情報を遠隔監視するための基盤技術を確立できた．期待される機能の例として，ロボットの現在速度，電池残量，検出した人物の画像等といった，遠隔監視に有益な情報表示機能が挙げられる．

4. 結果と考察

4.1 実験走行

LRF を 2 台用いることによって環境地図が安定して作成できるようになったため，一度手動で操作してデータ取りを行った後，すぐに地図作成を完了することができた．また，実験走行では横断歩道手前まで何度か走行できたものの，横断歩道を通過する実験を行うことが出来なかった．横断歩道へ侵入する際の段差が想定よりも大きく，車体が前のめりに傾いた際に地面を障害物として認識してしまっていた．他にも，コース後半の橋を渡ったあとの下り坂でも同様の現象が発生したことがあった．

4.2 本走行結果

本走行では雨だったものの，ラップトップ PC をビニール袋で覆う以外には特に防水処理は行っていない．LRF が雨を障害物として認識することがあり，頻繁に一時停止するものの，順調に走行し，横断歩道手前までの約 1420[m] 自律走行を行った．しかし，横断歩道手前

で一時停止出来なかったため，オペレータが緊急停止スイッチを押し，停止させた．横断歩道手前の waypoint に到達したら一時停止するような処理を実装していたが，waypoint の位置の設定ミスが原因だった．横断歩道手前での実験が不十分だったことが設定ミスの原因である．

昨年からハードウェア構成やソフトウェア構成をほとんど変更していないものの，地図作成や自己位置推定に用いた LRF を 1 台から 2 台に変更し，360 度から情報を得ることができるようになったことで格段に安定性を大きくすることが出来た．また，オドメトリの精度向上も自己位置推定の安定性向上につながった．しかしながら，広い場所では自己位置推定が安定しない場合があったため，3DLiDAR などを利用した自己位置推定を利用して解決を図りたい．

また，今回は完走を目標としたため人物発見には取組めていない．次回の参加では人物発見を行い，課題の完全達成を目指したい．

5. 開発状況などに関して

本年度の開発では，昨年度までに開発してきたロボットを用いたが，本年度の開発期間は大会本走行の約 1ヶ月前からであった．更に，開発を行った人数は 3 名で，開発期間の約半分は 2 名が KIT-C3 に，1 名が KIT-C4 の開発を行っていた．さらに開発メンバーの一人は，遠隔地におり，実際にロボットが間近にある環境で開発を行う回数は，ほんの数回であった．このような短期間かつ少人数で開発を行う際に，ソフトウェアの仕様統一が行い易い ROS の影響は非常に大きかったと言える．また，開発を行う際に，オンラインでソースコードを共有できる GitHub の利用も非常に効果的であった．開発を行うにあたり，我々のチームは実験走行になかなか参加出来ないため，学内での実験を積み重ねてきた．学内の実験においても約 1000[m] 程度の自律走行が安定してできていた．しかし，学内での実験では他のロボットや未知の障害物などが少なく，道幅は広く，起伏が殆ど無い環境だったためつくばチャレンジの環境では上手く行かないことがあった．本番前の実験走行ではこの辺りの調整が大変であった．

6. 最後に

我々が開発したソースコードは，GitHub 上で公開している．開発を行うにあたって参考にしていただければありがたい．`http://github.com/Nishida-Lab/TC2015`

謝辞

つくばチャレンジ実行委員会やつくば市の方々にはつくばチャレンジのような貴重な実験の機会を与えていただき感謝いたします．