RBProceedings 文書クラス サンプル文書

中村 勇太[1]，吉田 侑樹[1]，青木 修平[1]，村上 オト[1]
[1]Abudori Lab.

1　はじめに

つくばチャレンジは 2024 年のロボット大賞に選出された。つくばチャレンジは 18 年間、課題や取り組みを変えながら日本のロボット技術の進歩に大きく貢献してきた。さまざまな研究者に大体的に実験できる環境を提供しつづけロボットの技術を成長させてきた。完走しているチームの技術力は非常に高い。初めて参加するチームや個人で参加するチームとは技術的な差が大きい。初参加から完走するまでの道のりは遠い。その要因は以下の点であると感じた。完走しているチームの多くは高価な 3DLiDAR を搭載している。ROS 2 を使用した 3D ナビゲーションのノウハウがウェブ上にあまり公開されていない。つくばチャレンジレポートは昨年のレポートの差分が書かれていることが多い。つくばチャレンジレポートは特定の課題の解法が書かれていることが多い。つくばチャレンジレポートは過去のレポートがオープンになっていないため、引用先が見られない。以上の点から、初学者や新規参入者に対して敷居が高いように感じた。同時に、完走するために必要なことは以下のように感じた。屋外の自律走行には初学者ほど 3DLiDAR があった方がよい。複雑なシステムを運用するには、質の良いハードウェアが必要である。OSS を駆使すればある程度まではロボットを動作させることができる。OSS を利用するだけでは完走には至らない。AbudoriLab. チームでは、本走行では確認走行区間をクリアできなかった。他のロボットが通路を塞いだため新ルートを作れなかった。決められたルートに何も障害物がない状態であれば、もっと走行することができた。決められたルートを走行することだけであれば、自己位置を喪失することなく、障害物も発見し走行することができた。ゼロからつくばチャレンジに参加する人に必要なことがわかるレポートにしたい。AbudoriLab. が用意したロボットやソフトウェア構成を紹介する。

2　システム

2.1　ハードウェア

自律走行を行う機体で必要な要素は大きく分けると下記 5 点となる。

センサ走行装置計算機バッテリーシャーシ

センサは、自己位置推定や障害物を検出する為に必要となる。自己位置推定には LiDAR が多く使用されるが、GNSS、カメラといったセンサも使用される。走行装置は、ロボットの移動の為に必要となる。ロボットではモータで車輪を駆動する形式が一般的だが、不整地走行に適したクローラ、段差や階段にも対応できる脚といった機構も使用される。計算機は、センサから取得された情報から走行装置への指令を算出する為に必要となる。大量の計算が必要となる自己位置推定には概ね PC が使用される。多くの IO を要求されるモータ制御のため、マイコンも使用される。バッテリーは、上記の機器を使用する為に必要な電力を供給ために必要となる。移動ロボットは消費電力低下のため、重量効率の高い LiPo バッテリーが多く使用される。シャーシは、その他の機器を接続する為の機構物として必要となる。状況と要求されるタスクにより、適切にシャーシを作成する必要がある。

2.1.1　自律走行ロボット：Penguin

今回、自律走行を実現する基本的な構成を持ちつつ、不整地の走行も可能な機体とすることをコンセプトに、機体を作成した。作成した機体とその搭載した機器を、図 n に記載する。以降の章では、各要素について詳細を記載していく。

2.1.2　センサ

自己位置推定を行うための LiDAR センサとして、Velodyne 社の VLP16 を搭載した。自己位置推定用の LiDAR は、遠距離にある環境物 (建物、樹木等) をセンシングすることで、自己位置を推定する。通

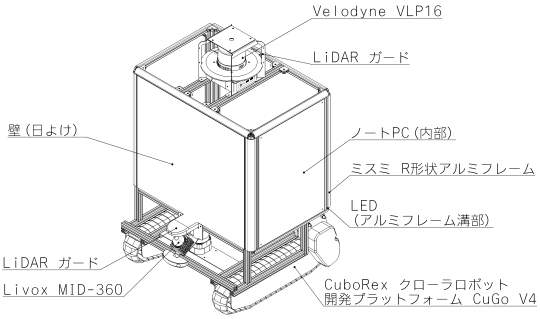

図 1　左の図

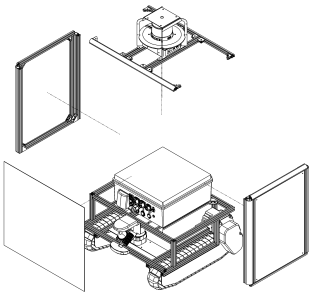

図 2　右の図

行人や障害物があった際にレーザー光を阻害し、自己位置推定の精度低下を発生させないために、機体の最上部に搭載した。また、LiDAR は光を使用したセンサであるため、光学窓に傷が発生すると誤検知が発生する。これを防ぐため、金蔵製のガードを作成した。

ロボットの自律走行を阻害する障害物を検出する LiDAR センサとして、Livox 社の MID-360 を搭載した。自律走行用の LiDAR は、遠距離をセンシングする必要がないため、機体の下部に設置した。また、今回の機体は後進を行わないため、障害物の検出範囲は前方向のみとした。障害物検出は、後述の通り地面の検出を実施するため、障害物検出範囲の地面も検出できる位置とした。上記をみたす位置として、最終的な搭載位置は図 n のような箇所とした。LiDAR を 35° 前傾させる事で、LiDAR 取付位置から 250mm 先の地面を検出できる形とした。障害物の検出範囲は、機体進行方向から左右 78° となった。障害物検出用の LiDAR も、光学窓への傷を防ぐ為、金属製のガードを上下に設置した。

2.1.3　走行装置

走行装置には、CuboRex 社のクローラロボット開発プラットフォーム CuGo V4 を使用した。不整地走行に適したクローラを有するプラットフォームを使用する事で、不整地の走破性向上と開発の高速化を図った。

2.1.4　計算機

ロボットの制御を行う PC として、○○社の○○を搭載した。ノート PC を搭載することにより、ロボットを駆動する電源系列と PC を駆動する計算機の電源分割を実現した。モータの制御には、CuboRex 社製モータドライバを使用した。

2.1.5　バッテリー

機体下部に、24V ○○ Ah の LiFePO4 バッテリを搭載した。本機体は不整地走行の可能性を考慮したため、路面により大きな振動が発生する可能性が考えられた。想定を超える振動によるバッテリへの悪影響を考え、LiPo バッテリより安全性の高い LiFePO4 バッテリーを採用した。バッテリは機体下部に設置することで、重心の上昇による安定性の低下を防いだ。バッテリから供給される電源は、ロボットに搭載した電源分配基板により適切に切替・分配・変圧し、PC を除く各機器に接続した。

2.1.6　シャーシ

ロボットのシャーシは図 n の形とした。屋外でのデバッグ作業を想定し、日よけの壁を搭載した。ロボットは図 n のような 5 ユニットで構成し、各部を計 10 本のボルトで分解できる形とした。分解箇所には付当てを設置し、複数回の分割。組立を実施しても再現性のあるシャーシとなる形とした。

構造部材にはミスミ社 R 形状アルミフレーム を使用することで、衝突しても対象に危害を加えにくい形状を実現した。突起部・巻き込みが発生しうる箇所には、樹脂製のカバーを搭載した。

ロボットの状態表示のため、アルミフレームの溝部に LED を搭載した。これにより、ロボットが自律走行状態か、操作者による操作を実施している状態かを表示した。

2.2　ソフトウェア

自律走行システムのソフトウェア

ロボットを走行させるためのソフトウェア構成は以下の通りである。地図作成。ロボットの活動範囲内は既知の場所。現実の場所とリンクした地図をロボットに持たせる。自己位置推定に使用する。自己位置推定。現在の位置がロボットが持っている地

図のどこに位置するのか推定する。現在位置が分かれば、目的までの向かう経路を計算することができる。障害物認識。自己位置推定ができても、実際には障害物があってたどり着かないことがほとんどだ。向かいたい場所までの経路上に障害物がどのようにあるのか反映させる。経路計画。ロボットは地図をもっているので目的地点と現在位置があれば 2 点間の経路を計算できる。このとき、環境上にある障害物も加味し、障害物に衝突しない経路を計算する。実際の自律走行では、数メートルおきに達成地点を設置し、細かく経路計画を実施する。経路追従。計算した経路にロボットを追従させる。障害物を回避しながら、柔軟に経路通りにロボットを制御させる必要がある。次の章から、それぞれの方法で具体的に使用したソフトウェアを紹介する。

3　地図作成

現代の自律走行の大半は、ロボットの行動範囲の環境地図を作成する。作成した環境地図をロボットが持つことにより、既知の環境内で自由に行動することを目指す。ROS 2 標準の Navigation2 の標準アルゴリズムも含め環境地図は 2D のものが多い。2D 地図は室内など水平であり障害物が単純な環境であれば、とてもよく動作する。しかし、屋外や広大な環境だと以下のような点でうまくいないことがある。設置した 2DLiDAR の設置高さによっては検知できない障害物が多く、無視できないことがある。上り坂など、本来障害物でないものも障害物として認識することがある。2D 地図で表現しきれない障害物が重要な物体であった場合、精度が大幅に悪化する。自動車が止まったりするなど環境の変化があると大きく悪影響を受けることがある。このつくばチャレンジ 2024 では、3D の地図を作成し、上記のポイントで複雑な環境でもうまく地図表現をした。回転式 3DLiDAR は非常に高価であるが、つくばチャレンジでは、無料貸与がある。つくばチャレンジ 2024 では、株式会社アルゴさまより無料で貸与いただいた。もっていない人でも 3DLiDAR に挑戦することができる。ぜひやっていただきたい。地図作成には、回転式 3DLiDAR で、GLIM（引用）を用いて作成した。作成したつくばチャレンジの走行範囲全域の地図は図の通りである。図環境構築をした後、全周のルートを走行させた時のロボットの回転式 3DLiDAR のセンサデータを記録した ROSBAG ファイルを再生するだけで 3D 地図が作成できてしまった。設定にチューニングとしては以下の点である。あああいいいううう AAA の点では、あああパラメータが効く BBB の点では、いいいパラメータが効く CCC の点では、ううううパラメータが効く XXX という点に注意してパラメータチューニングするとよい。最後に、オフライン地図修正ツールが同梱されているのでこれで地図を微修正する。ループクローズがうまくハマるように指定する。つくばチャレンジ全域で自己位置推定ができる地図ができた。3D 地図は https://google-drive で公開しているので、来年同じ構成で使用したい方は使うと良い。

4　自己位置推定

前章で作成した 3D 地図に対して、ロボットの自己位置を表現する。3D 地図で自己位置推定することで、2D 地図よりも以下の点で優れていると考えたため採用した。トラックのような大きな障害物がある場合でも自己位置が破綻しない。2D 地図の場合はバンが止まっている場所に近づいた時、位置が飛んでしまった。2D 地図の場合はロボットの周辺に人だかりがあるとマッチングできない。2D 地図の場合は坂道のように地面が 2DLiDAR に映る場合、位置が飛んでしまった。後述の Navigation では、2D を利用する。地面を走行するロボットであるため、2D の姿勢で十分。3D で自己位置推定して、そのうち X,Y,Yaw だけを抽出して利用する。自己位置推定手法としては、OSS の Lidarlocalizationros2 を利用した。前章で作成した 3D 地図を pcd に変換して読み込んだ。地図はコンフィグファイルに絶対パスを書き込むだけ読み込まれる。LiDAR を起動した状態で自己位置推定アルゴリズムを起動する。起動した後に初期位置を与える。コンフィグで初期設定を与えることもできる。初期姿勢と地図のマッチングが成功すると常に LiDAR スキャンマッチをつづけて自己位置を更新し続ける。設定にチューニングとしては以下の点である。あああいいいううう AAA の点では、あああパラメータが効く BBB の点では、いいいパラメータが効く CCC の点では、ううううパラメータが効く XXX という点に注意してパラメータチューニングするとよい。以下の点で自己位置が破綻することがある。あああいいいううううオドメトリをオンにすることでふっとびを抑えることができた。IMU はチューニング不足で破綻してしまった。結果的に XXX に注意すると良い。

5　障害物認識

自己位置とゴールが分かればロボットが進むべき経路を計算できる。ただし、これだけでは現実にある衝突してはいけない障害物を考慮していない。LiDAR などのセンサを使って障害物を認識し、それを地図に反映させることで障害物を考慮した経路を計算できる。多くのロボットでは、2DLiDAR を障害物に使用している。2DLiDAR を使うと坂道など地面が写ってしまう。2DLiDAR を使うと机のような物体は脚しか映らない。3DLiDAR を使うことで机の柱と天板の部分の全体の点群を得ることができる。2DLiDAR と異なり、ぶつかりたくない部分を抽出して点群として表現できる。メカの章に載せた、MID360 でを使用した。これを実現する ROS パッケージを作成した。簡単な手順としては以下の通りだ。点群をダウンサンプリングする。LiDAR 点群の傾きを設置角度から逆算して水平にする。法線を各点ごとに計算する。法線ベクトルから水平なものは地面など問題ないものと判断した。法線ベクトルから垂直に近いものは壁や障害物と判断した。計算を単純にするために、法線の Z 方向で閾値で区切った。これで坂道や乗り越え可能な 5cm 程度の小物は水平と判断し障害物判定できた。細い鉄パイプやパイロンの根本も障害物として判定できた。出力した点群を OSS の PointCloudToLaserScan パッケージに入力し、2DLiDAR の scan トピックに変換した。この Scan トピックを Navigation2 に入力することで、広く使われる 2DLiDAR のサンプルをそのまま動かすことができた。つくばチャレンジ環境で使用してパイロンや人、生垣などの大半の障害物を検出することができた。パイロンは先端の細いところから根本の一番太いところまで観測できた。この方式では、パイロンの一番太いところを Scan として出力するため、パイロンのぎりぎりを通過することがすくない。苦手な物体として、背のひくい水平な障害物は検出することが難しい。平台車。小さな段ボール。地面成分が大きく平均化され溶け込んでしまう。2.4GHz CPU シングルスレッドで 40ms くらい。８スレッドで分散処理して 7ms くらい。PC の電飾消費が深刻だったので、あえてシングルスレッドの実装で走行した。パッケージは OSS で公開しているので、ぜひ利用してほしい。

6　ナビゲーション

6.1　経路計画

ROS 2 では、経路計画パッケージとして Navigation2 がある。環境地図と自己位置を入力すれば、指定したゴール座標までの経路を出力してくれる。計算アルゴリズムの手法が複数用意されている。各手法が Plugin として実装されており、コンフィグファイルで Plugin 名を変更するだけで反映される。つくばチャレンジ 2024 では、3DMap を利用して自己位置推定を行なった。Navigation2 には Map を入力せず、３ DMap からの自己位置推定の X,Y,Yaw の値をそのまま利用するだけで２ D に転写した。Navigaiton2 には作成した ObstacleToScan から 2DLiDAR を模擬した scan トピックを入力した。これで２ DMap はないが、ロボット周辺の障害物マップは作成される。これで、ロボットから２～１０ m 程度の距離にゴールを指定することで短距離の経路を計算してくれる。この短距離で達成できるゴールを数メートルおきに、完走のゴールまで配置しつづける。近くのゴールが達成できれば、次のゴールを設定する処理を自動化した。これが PenguinNav。penguinnav はお願いします。SmacPlanner。経路計画方法は SmacPlanner を利用した。SmacPlanner は Navigation2 の Plugin に実装されているため、Config ファイルで SmacPlanner を選択するだけで利用できる。SmacPlanner はロボットの形状を考慮した衝突判定をしてくれる。したがって、自ロボットの投影面積であるポリゴンを指定するだけで、無茶な経路は選ばれなくなる。また、最小回転半径も指定できるため、なめらかな円弧で走行経路を利用できる。これにより、障害物ギリギリまで近づいて急に避ける行動を抑えることができた。

6.2　経路追従

前節の SmacPlanner で計算された経路をロボットがどのように再現するか計算するアルゴリズムを経路追従アルゴリズムという Navigation2 では、さまざまな経路追従アルゴリズムを利用できるつくばチャレンジ 2024 では、DWB をつかった DWB はロボットが再現可能な速度や加速度の範囲の中で一番良い速度を選択する最高速度と最低速度と加速度を設定できるクローラの制御の場合、抵抗が非常に大きく初動に大きなエネルギーを必要とする最低速度を速

く設定し、すぐに停止できるように減速加速度を大きくなるように設定した。

7　本走行

以上の構成で 3 DLiDAR を利用した自律走行システムを構築したつくばチャレンジ 2024 の本走行では、市庁舎裏の細い通路でロボットを回避し、元の経路に再計画できずにリタイアした。それまでの走行は、自己位置を見失うこともなく、障害物を未検出になることなく動作した。障害物からは距離を 1m 程度離れるように設定していたため、狭路では、非常に慎重にゆっくりゆっくり進んだが、走行経路はブレることはなかった。このロボットは設定したルートに忠実になぞる機能しか搭載していなかった他ロボットが決められた道を長時間塞いでしまった時、迂回する機能は搭載されていなかったため、復帰できず終了した本年は決められた座標の通りに走行する機能までは 3 DLiDAR を使って実現することができた

8　結論

つくばチャレンジ 2024 では、 3 DLiDAR を利用したナビゲーションシステムを構築することができた。 3 DLiDAR のナビゲーションシステムは文献が少なく構築することに苦労したが、最低限の機能を揃えることはできたつくばチャレンジの多くのロボットは 3 DLiDAR を利用しているが、何から手をつければ良いかわからない人はいるはずであるこのレポートでは、ハードウェアから自律ナビゲーションシステムの各機能のおおよその仕組みを述べた来年初参加を考えている人の参考になれば幸いである環境構築の方法や実際の詳細の使用方法などは引き続きブログで解説していく予定である。次年度では、決められたルートを走行する上での意思決定機能を追加で実装する完走するためには、不測の事態でも、認知判断行動することが必須である完走にむけてナビゲーションシステムをブラッシュアップしていく予定である。

[1]

参考文献

[1] W3C 日本語組版タスクフォース．日本語組版の要件（日本語版），（2020-11 閲覧）．https://www.w3.org/TR/jlreq/.