TechShare

Confidential

ハイレベル制御のプログラム開発（四足歩行ロボット編）

2024/12/12

石川皓詞

TechShare

Agenda

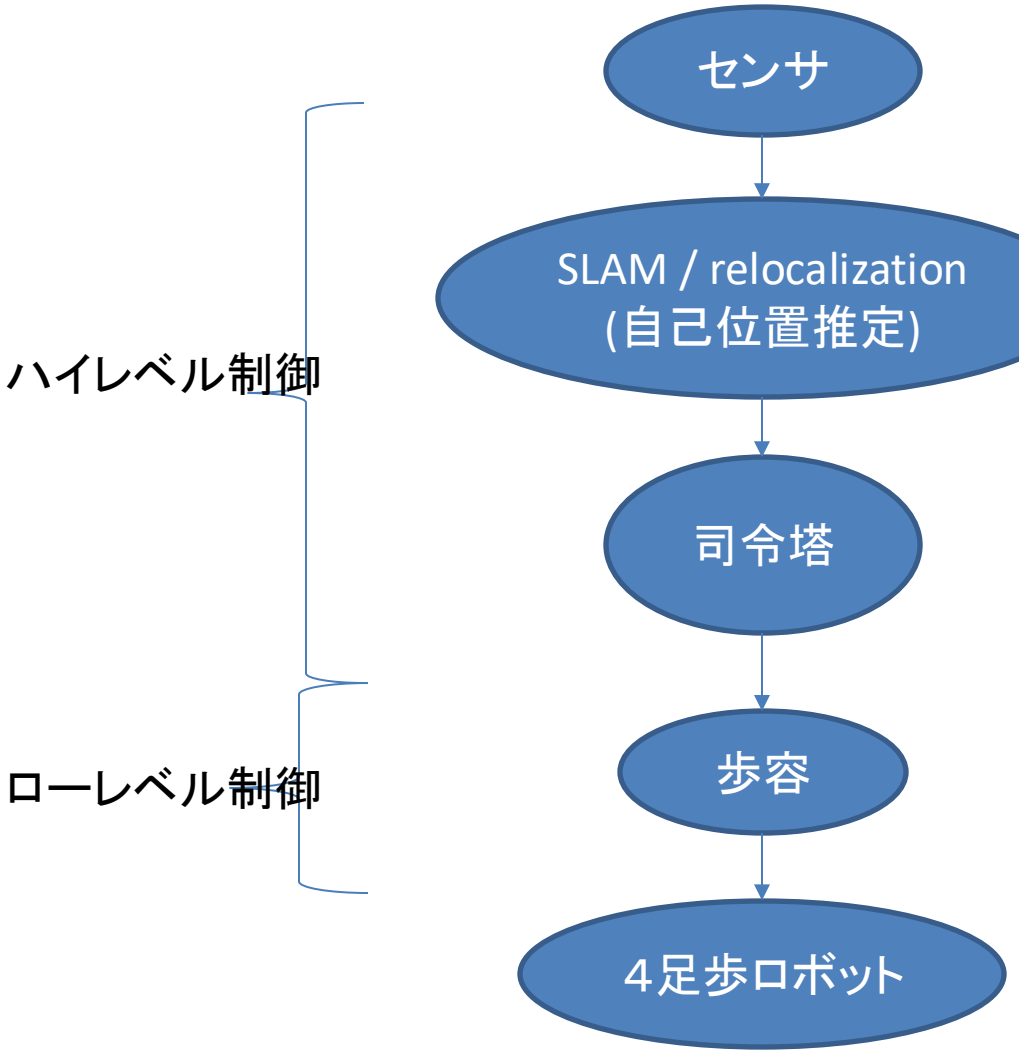
1. はじめに
2. 簡単なハイレベル開発例
3. より実践的な開発例
4. まとめ

１．はじめに

１－１．四足歩行ロボットの制御とは？

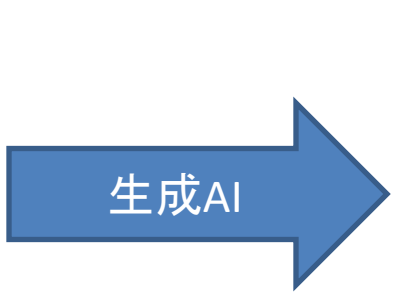
- ROS
  - C++, Python

- 自律走行
  - LiDAR　→　SLAM
  - カメラ　→　マーカー、画像処理、V-SLAM



1－2．開発における難しさ

- 高度な技術レベル
  - ROSやSLAMなど、高度な専門知識が必要
  - 日本語サイトだけでは不十分で英語のサイトを読む必要がある
- 実機がある
  - 複数のセンサー、コンピューター、アクチュエーター等を組み合わせる　→　相性問題、コード調整、パラメーター調整
  - ハードの故障や、通信に関する問題など、不具合原因の調査の複雑化



- ✓ 前提知識の解説
- ✓ コード解説
- ✓ リファクタリングや設計の改善
- ✓ エラー対処
- ✓ コード生成
- ✓ Etc…

開発効率化に貢献しているのは事実だが…

２．簡単なハイレベル開発例

2－1．目標

- キーボードでGo2を動かす

キーボード入力

開発したい
プログラム

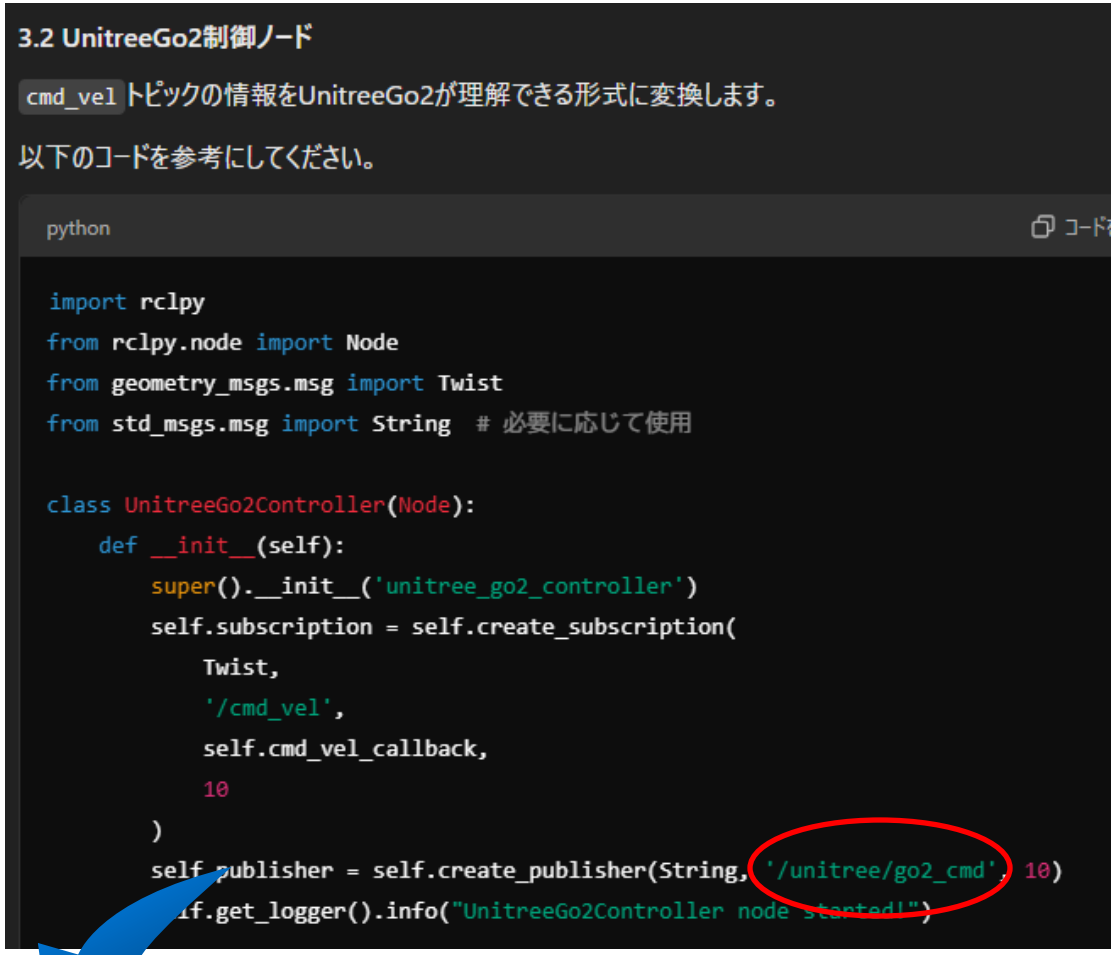
4足歩行ロボット

2－2．生成AIの問題点

- 調査方法
  - Chatgpt 4o
  - 入力例：「UnitreeGo2について、ROS2を用いて、キーボード操作をするプログラム開発をしてください」など

生成AIの出力結果(1)

Teleop_twist_keyboardを使うのは正しい



生成AIの出力結果(2)

/unitree/go2_cmdという存在しないトピック名を作り出している

2－3．Go2のハイレベル制御方法

- 複雑な動きを同じフォーマットで指令可能（＝４輪車とは異なる）

```
RequestHeader header
        RequestIdentity identity
                int64  id
                int64 api_id
        RequestLease lease
                int64 id
        RequestPolicy policy
                int32 priority
                bool noreply
string parameter
uint8[] binary
```

/api/sport/resuestのデータ型

- 1004: Stand Up
- 1005: 伏せ
- 1007: 姿勢制御
- 1008: 速度指令
- 1022: Dance1
- 1031: FrontJump
- Etc…

パラメーターが必要な場合

```
header:
  identity:
    id: 0
    api_id: 1008
  lease:
    id: 0
  policy:
    priority: 0
    noreply: false
parameter: '{"x":0.5,"y":0.0,"z":0.0}'
binary: []
```

/api/sport/resuest情報：
前進方向に0.5m/sの指令値

専用関数で情報作成可能

```
void Move(unitree_api::msg::Request &req, float vx, float vy, float vyaw);
```

2－4．正しい答えを出力させるには

- 調査方法
  - Chatgpt 4o
  - 入力：「下記の情報をもとに、UnitreeGo2について、ROS2を用いて、/cmd_velの入力をもとに動作させるプログラム開発をしてください。（＋参考になる分かりやすい自作のサンプルコード(※)）」

```
void cmdVelCallback(const geometry_msgs::msg::Twist::SharedPtr cmd_vel_msg)
{
    // 受け取った/cmd_velメッセージをスポーツリクエストに変換
    unitree_api::msg::Request req;
    sport_client_.Move(req, cmd_vel_msg->linear.x, cmd_vel_msg->linear.y, cmd_vel_

    // リクエストを送信
    request_publisher_->publish(req);
    RCLCPP_INFO(this->get_logger(), "Published Request: LinearX=%f, LinearY=%f, A
                cmd_vel_msg->linear.x, cmd_vel_msg->linear.y, cmd_vel_msg->angular
}
```

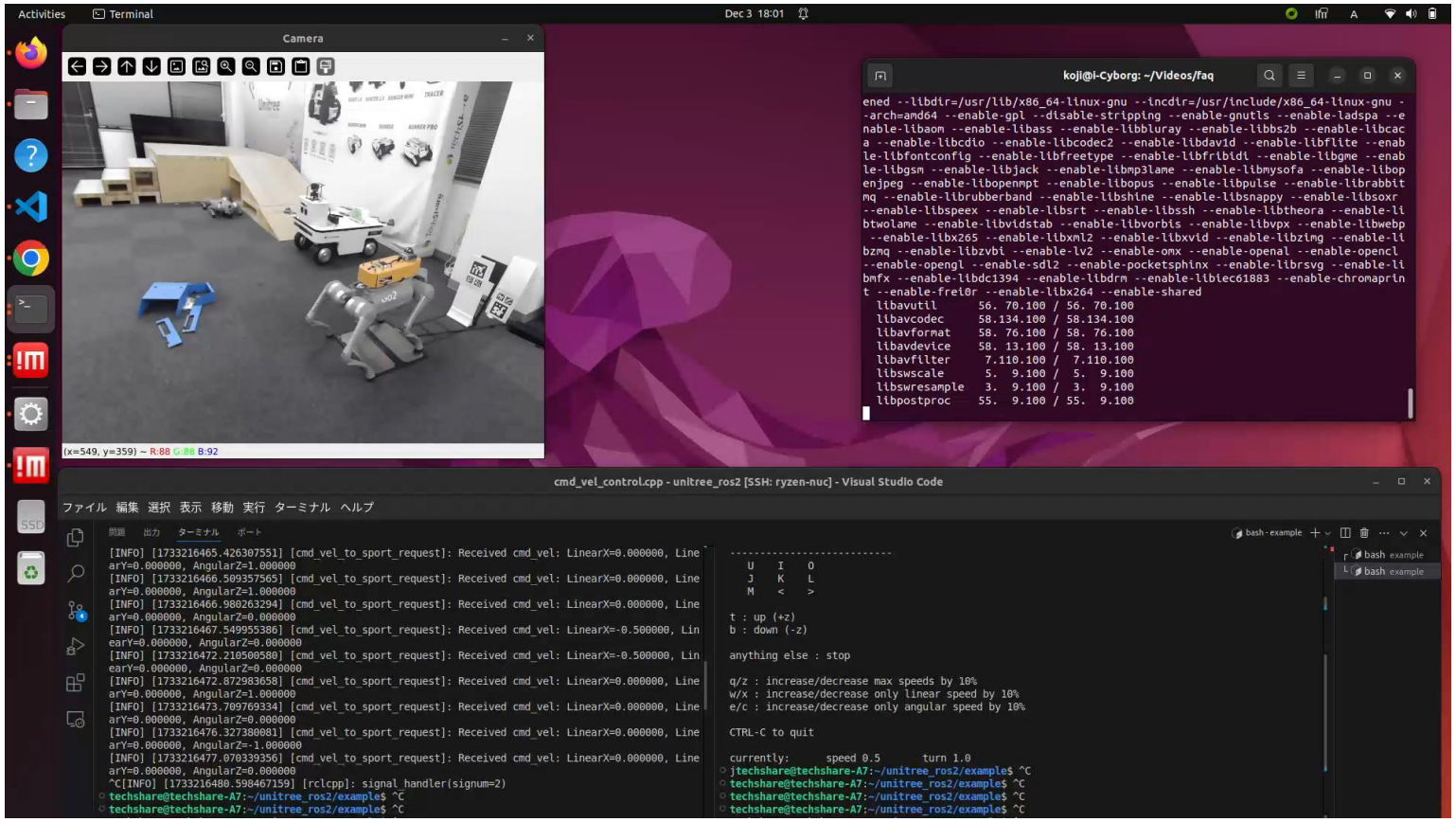
4oの出力結果

4oでも正しい出力

※ 公式Docsの継ぎはぎだと、4o, o1-miniだと難しかった。O1-previewなら正しい答えを出した

2－5．実行

- キーボードでGo2を操作



2－6．分散志向

- ロボット開発の基本

キーボード入力

キーボード入力を
ROSトピックにす
るプログラム

速度指令値
/cmd_vel

/cmd_velでロボットを
動かすプログラム

ROSによる分散

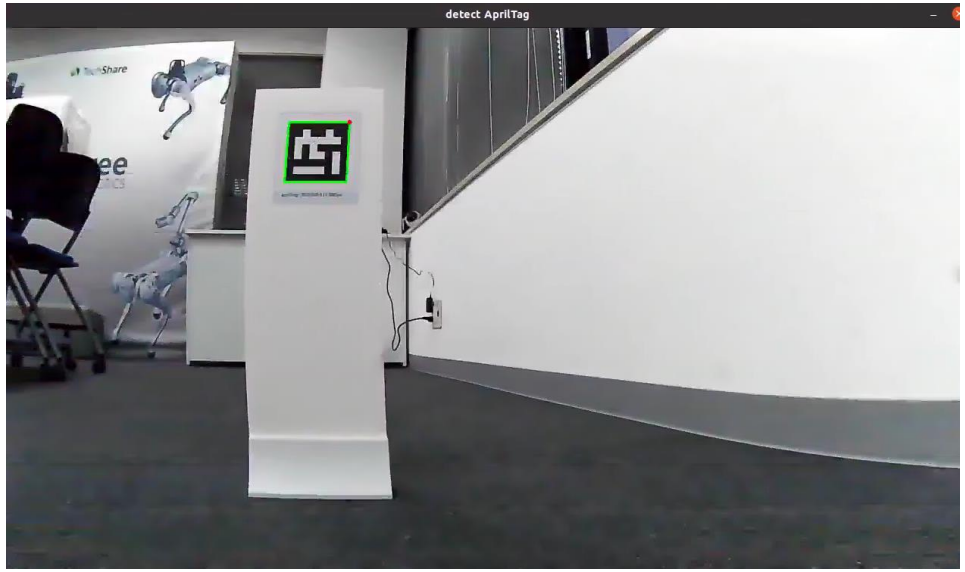
4足歩行ロボット

✓ ROSの親和性が高い
　✓ デバッグの簡略化
　✓ 再利用性
✓ 生成AIのパフォーマンス向上

３．より実践的な開発例

3ー1．目標

- マーカー検出を行い、Go2を動かす

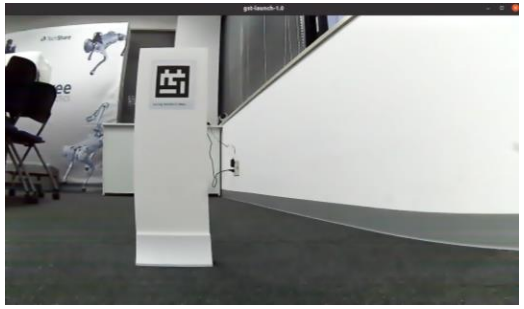


正面カメラでマーカー検知

マーカー検知したら、180° 回転する

3－2．なるべく分割した構成を考える

- マーカー検出を行い、Go2を動かす



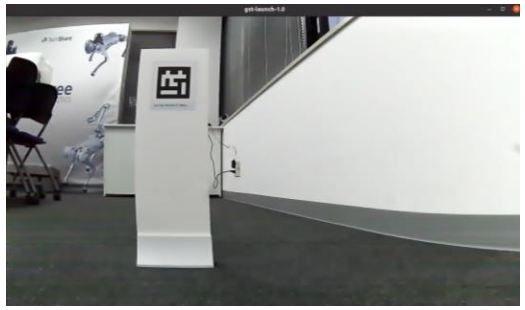
カメラ画像

開発したい
プログラム

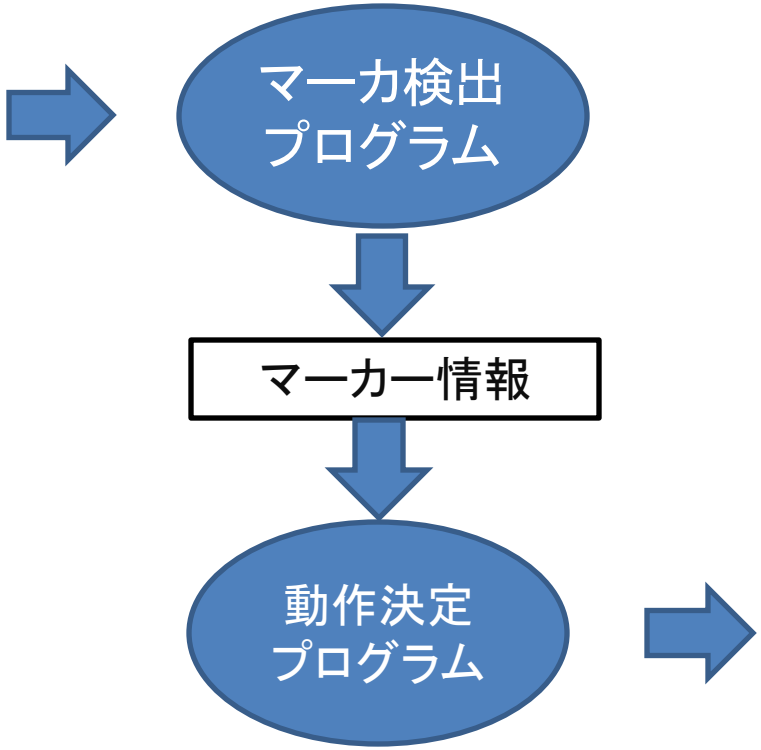
4足歩行ロボット

3－2．なるべく分割した構成を考える

- マーカー検出を行い、Go2を動かす

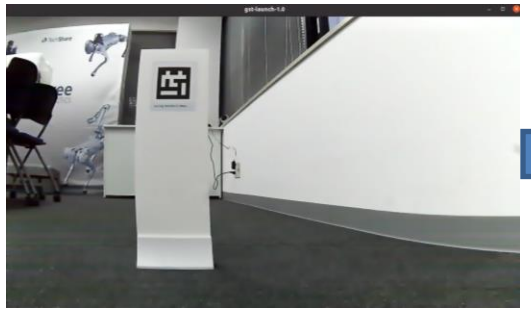


カメラ画像

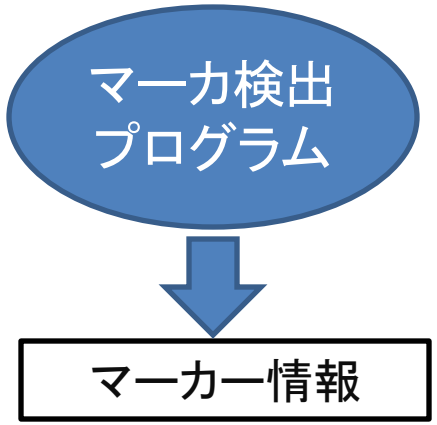


４足歩行ロボット

3－3．マーカー検出：さらに分割した構成

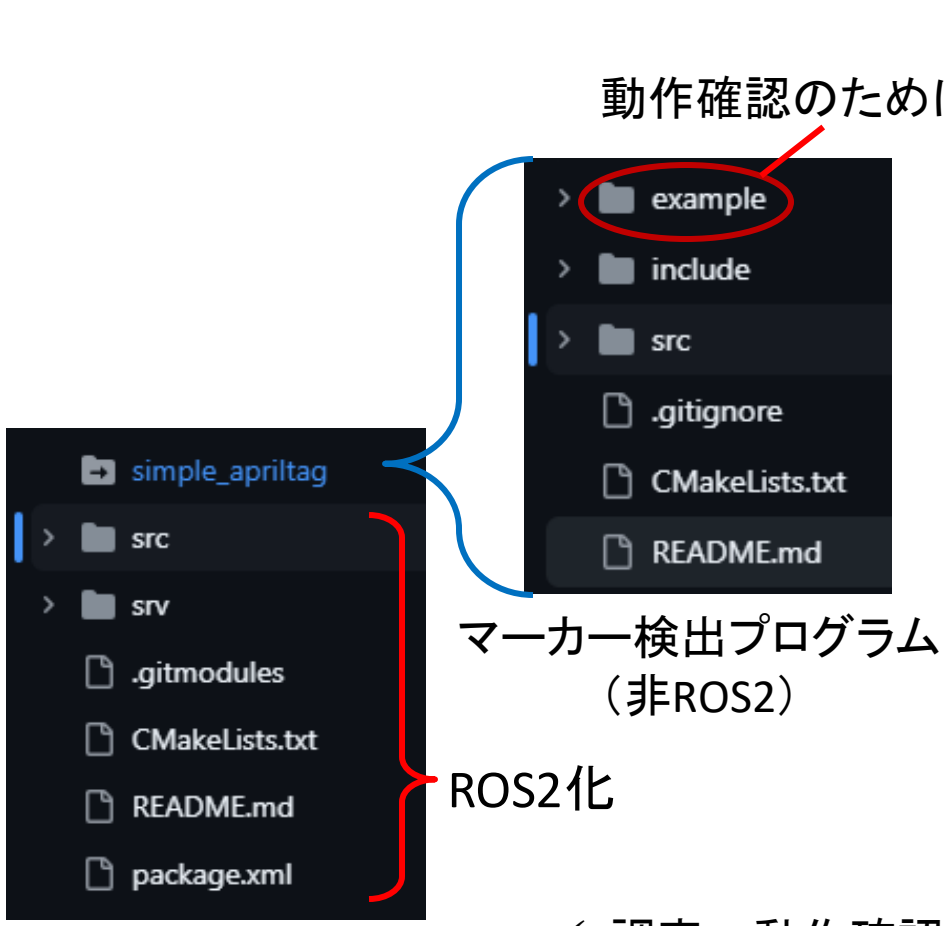


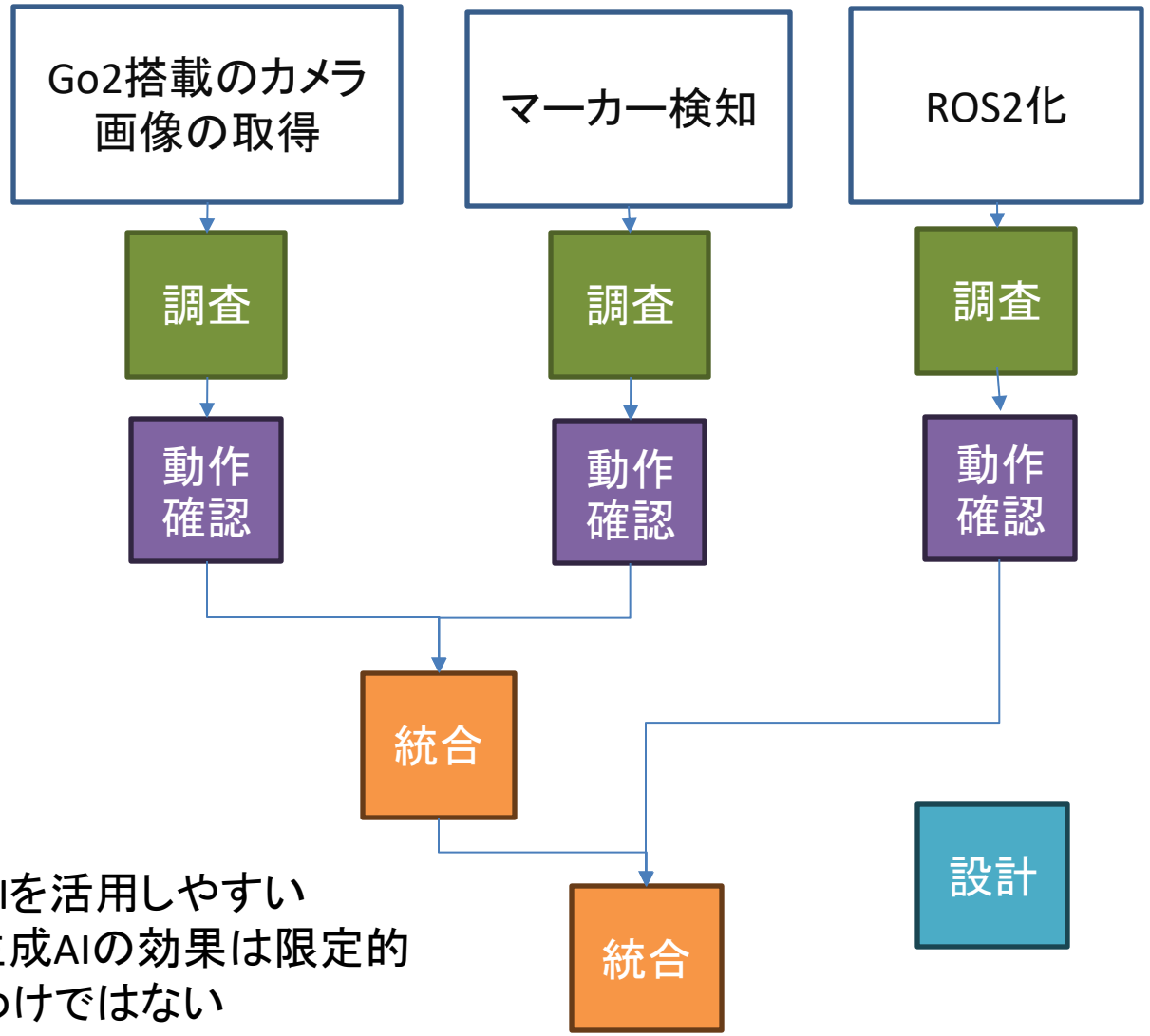
カメラ画像



- 画像の取得
  - UnitreeGo2：GST SDK
  - OpenCV: gstreamer×videoCapture
  - ロボット（Go2）に依存
- 取得した画像で、マーカー検出
  - Apriltag
  - OpenCV
  - ロボット（Go2）に依存しない
- ROS2化
  - ROS2 service
  - マーカー情報の出力

3－3．マーカー検出：開発フロー

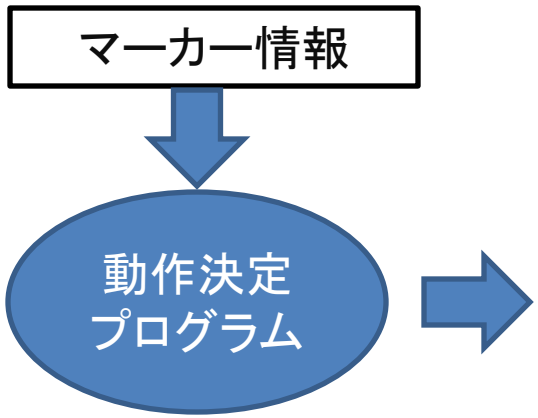




- ✓ 調査～動作確認は生成AIを活用しやすい
- ✓ 設計/統合については、生成AIの効果は限定的
- ✓ 開発手順自体が変わるわけではない

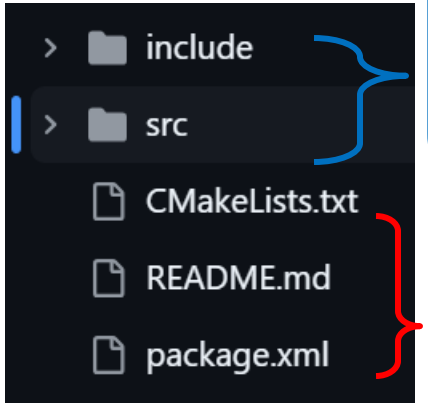
３－４．動作決定：さらに分割した構成

- ロボットの制御
  - UnitreeGo2 Ros2 インターフェイス
  - ロボット（Go2）に依存
- ロボットの状態の取得
  - UnitreeGo2 Ros2 インターフェイス
  - ロボット（Go2）に依存
- 制御アルゴリズム
  - PID制御
  - ロボット（Go2）に依存しない
- ROS2化
  - ROS2 service
  - マーカー情報の取得



４足歩行ロボット

3－4．動作決定：開発フロー



動作決定プログラム

ROS2化

sport_rotation.h
sport_rotation.cpp

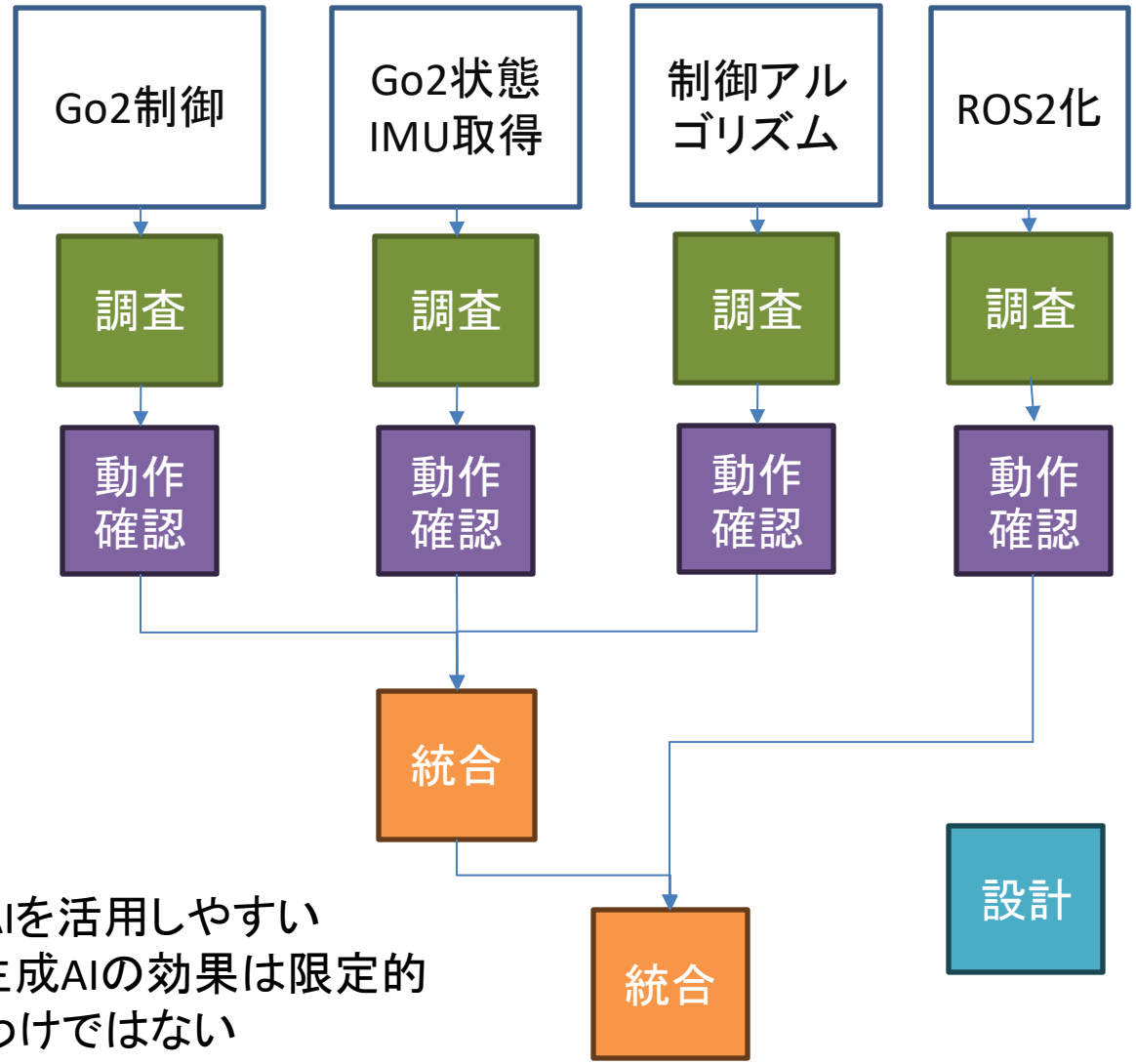
180°　回転制御コード

main_rotation.cpp

ヘッダー動作確認用

tag_move_demo.cpp

最終実行プログラム



- ✓ 調査～動作確認は生成AIを活用しやすい
- ✓ 設計/統合については、生成AIの効果は限定的
- ✓ 開発手順自体が変わるわけではない

４．まとめ

- 生成AIにより、開発のハードルは下がったが、まだまだエンジニアの知識は必要
- ROSにおけるモジュール化と、生成AIの特性はマッチしている
- ロボット開発は１つ１つ動作確認を行いながら進めていくしかない

TechShare